Critical Considerations on Angle Modulated Particle Swarm Optimisers

Barend J. Leonard · Andries P. Engelbrecht · Christopher W. Cleghorn

Abstract This article investigates various aspects of angle modulated particle swarm optimisers (AMPSO). Previous attempts at improving the algorithm have only been able to produce better results in a handful of test cases. With no clear understanding of when and why the algorithm fails, improving the algorithm's performance have proved to be a difficult and sometimes blind undertaking. Therefore, the aim of this study is to identify the circumstances under which the algorithm might fail, and to understand and provide evidence for such cases. It is shown that the general assumption that good solutions are grouped together in the search space does not hold for the standard AMPSO algorithm or any of its existing variants. The problem is explained by specific characteristics of the generating function used in AMPSO. Furthermore, it is shown that the generating function also prevents particle velocities from decreasing, hindering the algorithm's ability to exploit the binary solution space. Methods are proposed to both confirm and potentially solve the problems found in this study. In particular, this study addresses the problem of finding suitable generating functions for the first time. It is shown that the potential of a generating function to solve arbitrary binary optimisation problems can be quantified. It is further shown that a novel generating function with a single coefficient is able to generate solutions to binary optimisation problems with fewer than four dimensions. The use of ensemble generating functions is proposed as a method to solve binary optimisation problems with more than 16 dimensions.

Keywords Swarm intelligence · Particle swarm optimisation · Angle modulation · Discrete optimisation · Binary optimisation

B. Leonard · A. Engelbrecht · C. Cleghorn
Department of Computer Science, University of Pretoria, Hillcrest, Pretoria, South Africa
E-mail: bleonard@cs.up.ac.za

A. Engelbrecht
E-mail: engel@cs.up.ac.za

C. Cleghorn
E-mail: ccleghorn@cs.up.ac.za

1 Introduction

The earliest use of angle modulated particle swarm optimisation (AMPSO) is found in (Franken, 2004). However, the technique was formally introduced by Pampara et al (2005). AMPSO employs a particle swarm optimiser (PSO) (Kennedy and Eberhart, 1995) to optimise the four coefficients of a trigonometric function, known as the *generating function*. The generating function is then sampled at regular intervals and each sample value is mapped to a binary digit. The resulting bit string is interpreted as a potential solution to some binary optimisation problem.

Although other algorithms exist to solve binary-valued optimisation problems[1] (e.g. Holland, 1975; Kennedy and Eberhart, 1997), Pampara (2013) pointed out that these binary algorithms are susceptible to problems like hamming cliffs, loss of precision, search space discretisation, and the curse of dimensionality. Because AMPSO is able solve problems with binary-valued solutions using a PSO (with no modifications to the algorithm), which is defined in a four-dimensional continuous search domain, the algorithm is not prone to the same issues noted above. In addition to PSO, Pampara et al (2006), and Pampara (2013) also successfully combined angle modulation with genetic algorithms (GA), evolutionary programming (EP), differential evolution (DE), and artificial bee colonies (ABC).

Leonard and Engelbrecht (2014) identified a number of potential problems with AMPSO and introduced three variants to attempt to overcome these problems. Two of the variants, called Amplitude AMPSO (A-AMPSO) and MinMax AMPSO (MM-AMPSO), introduced additional complexity in terms of dimensionality to the PSO algorithm, but showed performance improvements in a handful of problem cases. The remaining variant, called Increased-Domain AMPSO (ID-AMPSO) showed no statistically significant improvement over AMPSO.

Although previous work (Leonard and Engelbrecht, 2014) has succeeded at making some improvements to the AMPSO algorithm in specific cases, the approach followed to identify potential flaws in the algorithm has, thus far, not been backed by empirical or theoretical evidence. As a result, there is no clear understanding of when or why the AMPSO algorithm (or any of its existing variants) might fail to produce good results. Therefore, the aim of this work is to study various aspects of the AMPSO algorithm in order to identify and provide evidence for any effects or characteristics that may have a negative influence on the performance of AMPSO.

The study begins with an analysis of the periodicity of the standard generating function that is used in AMPSO. The periodicity of the generating function has never been considered as a point of concern in existing literature. It is shown that the periodicity of the standard generating function is not problematic in AMPSO. However, it is discovered that the roots of the generating function increase in frequency along the x-axis. This characteristic is discovered to have potentially severe consequences on the algorithm's performance. To illustrate the potential problem, an empirical analysis is performed. It is shown that – because of the increasing frequency of the roots of the generating function – the distance between solutions in the coefficient space (where PSO is defined) is not linearly correlated with the dis-

[1] This article henceforth refers to algorithms that are defined for binary search domains as binary algorithms. Likewise, algorithms that are defined for continuous search domains are called continuous algorithms.

tance between solutions in the solution space (where the solution to the arbitrary binary problem exists). That is, the assumption made by PSO that good solutions are grouped together is violated. This problem is henceforth referred to as *spatial disconnect*. Spatial disconnect manifests as a result of the generating function and is therefore relevant to anybody using the AMPSO algorithm, regardless of the specific binary problem being optimised.

The study continues with the first ever empirical analysis of particle convergence in AMPSO. It is discovered that particles in AMPSO tend to stabilise at high velocities, relative to the size of the search domain. This problem is referred to as *inadequate convergence* and can also be explained by spatial disconnect. Additional methods are suggested to confirm that spatial disconnect is indeed the underlying cause of inadequate convergence.

A novel approach to finding suitable replacement generating functions is then introduced. A definition is provided to quantify the ability of an arbitrary generating function to generate all possible solutions to any arbitrary binary problem. This new quantity is referred to as *generating function potential*. This definition is applied to construct a new generating function, with a single coefficient, to solve arbitrary binary problems with fewer than four dimensions using AMPSO. Until now, applying AMPSO to binary problems with fewer than four dimensions actually caused an increase in dimensionality, because the standard generating function has four coefficients that need to be optimised.

Unfortunately, calculating the potential of a generating function to solve high-dimensional binary problems proves to be a difficult task. Indeed, the best potential that has been calculated for the standard generating function shows that it is able to solve binary problems with up to 16 dimensions. As a workaround to this problem, the concept of *ensemble generating functions* is introduced. An ensemble generating function refers to the use of many generating functions in AMPSO, where each generating function generates part of the overall binary solution. The new definition of generating function potential is then used to show that ensemble generating functions are able to generate all possible solutions to arbitrarily high-dimensional binary problems.

The rest of this article is structured as follows: section 2 gives a brief overview of the PSO algorithm. Section 3 provides an explanation of AMPSO, while the AMPSO variants that have been introduced in previous work are discussed in section 4. Analyses and discussions of various aspects of angle modulated particle swarms are presented in section 5. A novel approach to quantify the ability of arbitrary generating functions to solve arbitrary binary problems is given in section 6. The study is concluded in section 7.

2 Particle Swarm Optimsation

Particle swarm optimisation is a stochastic, population-based search algorithm. A population of candidate solutions, called *particles*, is maintained throughout the search process. Each particle i has a position $\mathbf{x}_i$ and a velocity $\mathbf{v}_i$ in an n_x-dimensional search space. The best position that a particle i has found during the search process is referred to as the particle's *personal best position*, $\mathbf{y}_i$. The best position found by any particle is known as the *global best position*, $\hat{\mathbf{y}}$, assuming a star neighbourhood topology. At each time step $t+1$, every particle updates its velocity using the following equation:

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 \mathbf{r}_1(t)[\mathbf{y}_i(t) - \mathbf{x}_i(t)] + c_2 \mathbf{r}_2(t)[\hat{\mathbf{y}}(t) - \mathbf{x}_i(t)], \tag{1}$$

where ω is an inertia weight, c_1 and c_2 are acceleration coefficients, and $r_{1j}(t)$ and $r_{2j}(t)$ are random values, sampled from $U(0,1)$ in each dimension $j = 1, \ldots, n_x$. Each particle's position is then updated using

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

The resulting behaviour is that particles stochastically return to regions of the search space where good solutions have previously been found.

3 Angle Modulated Particle Swarm Optimisation

Angle modulated particle swarm optimisation makes use of a PSO algorithm to optimise the coefficients (a, b, c, and d) of the following trigonometric function:

$$g(x) = sin[2\pi(x-a)b\ cos(2\pi(x-a)c)] + d. \tag{3}$$

This function is known as the *generating function*, because it is used to construct (or generate) a binary solution. The resulting PSO maintains a swarm of 4-dimensional particles with position vectors of the form

$$\mathbf{x}_i = (a, b, c, d). \tag{4}$$

To evaluate a particle, the coefficients from the particle's current position are substituted into g. The generating function is then sampled at regular intervals $x = 0, 1, 2, \ldots, n_b - 1$, where n_b is the length of the required binary solution. Each sample value is mapped to a binary digit as follows:

$$\begin{aligned} &\text{if } g(x) > 0 \rightarrow 1, \text{ otherwise} \\ &\text{if } g(x) \leq 0 \rightarrow 0. \end{aligned} \tag{5}$$

Thus, the optimisation problem presented to PSO is to find the optimal coefficients (a, b, c, and d) for the generating function, such that the optimal binary solution may be obtained by sampling the generating function at regular intervals. In this way, an n_b-dimensional binary problem can potentially be solved using PSO in a 4-dimensional continuous search domain. Figure 1 illustrates the generation of a 5-dimensional binary string from g.

4 Angle Modulated Particle Swarm Variants

Leonard and Engelbrecht (2014) identified the following potential problems that were thought to cause the standard AMPSO algorithm to perform poorly:

- the absence of a parameter to control the amplitude of the generating function,
- the small size of the initialisation domain that was used in the original AMPSO algorithm, and

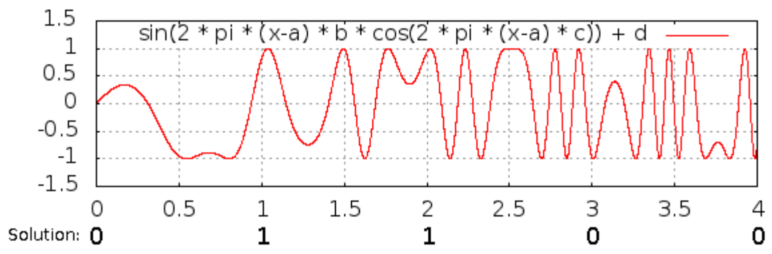

Fig. 1 An example 5-dimensional binary solution is constructed from g with $a = 0$, $b = 0.5$, $c = 0.8$, and $d = 0$.

– the predefined sampling positions at which the generating function was sampled to produce a bit string.

In order to alleviate the identified problems, three variants of AMPSO were introduced. This section provides an overview of the three AMPSO variants.

A-AMPSO is explained in section 4.1. Section 4.2 describes the ID-AMPSO variant, and MM-AMPSO is discussed in section 4.3.

4.1 Amplitude AMPSO

The first variant that was proposed by Leonard and Engelbrecht (2014) is A-AMPSO. This variant adds an additional amplitude coefficient e to the generating function:

$$g(x) = e\ sin[2\pi(x - a)b\ cos(2\pi(x - a)c)] + d. \tag{6}$$

The additional coefficient implies that the dimensionality of particles increases to five dimensions when implementing this variant:

$$\mathbf{x}_i = (a, b, c, d, e). \tag{7}$$

The amplitude of g amplifies the effect of the vertical shift coefficient d: the smaller the amplitude, the greater the effect of d on the generated solution. Results indicated that the e coefficient allows the algorithm to find better solutions faster in only a few problem cases.

4.2 Increased-Domain AMPSO

In the ID-AMPSO variant, no modifications to the AMPSO algorithm was made, other than increasing the initialisation domain of particles in the PSO algorithm from $[-1, 1]^4$ to $[-1.5, 1.5]^4$. The motivation for this variant was that initial coefficients that allowed the vertical shift d to exceed the amplitude of the generating function could potentially be beneficial to the search. However, the results showed no improvement over normal AMPSO. The lack of improvement can be ascribed to the roaming behaviour of particles in the PSO algorithm (Engelbrecht, 2012): because particles in PSO already have a tendency to exit the initialisation domain during the first few iterations of the search process, no benefit is gained by slightly increasing the initialisation domain.

4.3 Min-Max AMPSO

The final variant is MM-AMPSO. In this variant, the position vectors of particles are augmented with two additional dimensions to control the sampling range of the generating function. A particle's position is then given by

$$\mathbf{x}_i = (a, b, c, d, \alpha_1, \alpha_2), \tag{8}$$

where α_1 and α_2 are the bounds of the sampling domain. Let $\alpha_l = min\{\alpha_1, \alpha_2\}$, and $\alpha_u = max\{\alpha_1, \alpha_2\}$. Then, the generating function is sampled at every δ^{th} position in the range $[\alpha_l, \alpha_u)$, where

$$\delta = \frac{\alpha_u - \alpha_l}{n_b}. \tag{9}$$

Results showed that MM-AMPSO outperformed AMPSO for some binary problems with 225 to 432 dimensions. However, the additional complexity in terms of particle dimensions had a negative influence on performance in lower dimensions.

5 Analysis of Angle Modulated Particle Swarm Optimisation

While the variants that were discussed in section 4 showed improved performance in some specific problem cases, none of them yielded improvements across a wide range of problems. A better understanding of why and when the AMPSO algorithm might fail is clearly required in order to truly improve the performance of AMPSO.

This section aims to scrutinise various aspects of angle modulated particle swarm optimisers in an effort to understand why the algorithm might fail to solve arbitrary binary problems.

Section 5.1 investigates the periodicity of the generating function. The relationship between the binary- and coefficient spaces in AMPSO is studied in section 5.2, and section 5.3 presents the first empirical analysis of the convergence behaviour of angle modulated particle swarms.

5.1 Periodicity of the generating function

To begin, consider the periodicity of the generating function. A periodic function is a function whose values repeat at some fixed interval T. Formally, if $f(x)$ is periodic, then there exists a $T \neq 0$, such that

$$f(x + T) = f(x), \ \forall x \in \mathbb{R}. \tag{10}$$

The use of a periodic generating function in AMPSO raises the concern that it could cause repetition in the generated bit string. The interval of repetition in the resulting binary solution is not necessarily the same as the period T of the generating function, but rather depends on T, as well as the sampling domain. Nevertheless, as the dimensionality n_b of the binary problem increases, so does the probability of producing solutions that contain repetition, if the generating function is periodic.

Of course, in order to know whether this problem is a valid concern in AMPSO, it is necessary to know if g (equation (3)) is periodic. One might easily make the assumption that g is periodic, because it is a *sin* wave. However, closer inspection of the function reveals that it is, in fact, not periodic, as long as $bc \neq 0$.

5.1.1 Proof that g is aperiodic

In the case where $b = 0$, equation (3) reduces to a constant value:

$$g(x) = sin(0) + d = d. \tag{11}$$

When $c = 0$,

$$g(x) = sin[2\pi(x - a)b] + d, \tag{12}$$

which clearly is periodic. Now, consider the case where $bc \neq 0$. Without loss of generality, let

$$g(x) = sin[f(x)], \tag{13}$$

where

$$f(x) = bx\ cos(cx). \tag{14}$$

Now, $sin(x) = 0 \Leftrightarrow x = \pi n,\ n \in \mathbb{Z}$. Therefore,

$$g(x) = 0 \Leftrightarrow f(x) = \pi n. \tag{15}$$

Consider the function f on any interval

$$I_k = \begin{cases} \left[\frac{2\pi k}{c} - \frac{\pi}{2c}, \frac{2\pi k}{c}\right] & \text{if } c > 0, \\ \left[\frac{2\pi k}{c}, \frac{2\pi k}{c} - \frac{\pi}{2c}\right] & \text{if } c < 0, \end{cases} \tag{16}$$

where $k \in \mathbb{N}^+$. Observe that $cos(x)$ has roots at $x = \pi n - \frac{\pi}{2}$. Therefore,

$$\begin{aligned} f\left(\frac{2\pi k}{c} - \frac{\pi}{2c}\right) &= b\left(\frac{2\pi k}{c} - \frac{\pi}{2c}\right)\ cos\left(2\pi k - \frac{\pi}{2}\right) \\ &= 0. \end{aligned} \tag{17}$$

Furthermore, observe that $cos(2\pi n) = 1$. Therefore,

$$\begin{aligned} f\left(\frac{2\pi k}{c}\right) &= \frac{2\pi bk}{c} \cdot\ cos(2\pi k) \\ &= \frac{2\pi bk}{c}. \end{aligned} \tag{18}$$

Figure 2 illustrates the relevant intervals of f for $k = 1, \ldots, 6$, $b = 1$, and $c = 1$. From the observations above, it is evident that the range of f on I_k is $\left[0, \frac{2\pi bk}{c}\right]$ if $bc > 0$, or $\left[\frac{2\pi bk}{c}, 0\right]$ if $bc < 0$. In either case, the length of this range is $\left|\frac{2\pi bk}{c} - 0\right|$). Now, because f is continuous, there exists a value $x^* \in I_k$ such that $f(x^*) = \pi n$ for every πn in the range of f on I_k. Furthermore, f is monotonic on any interval

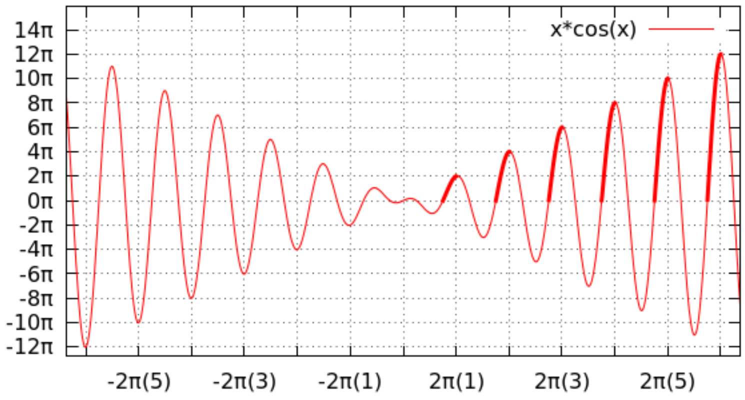

Fig. 2 The range of f on I_k is $\left[0, \frac{2\pi bk}{c}\right]$ when $bc > 0$. In this figure, every interval I_k is highlighted in bold for $k = 1, \ldots, 6$, $b = 1$, and $c = 1$. Note that the range of f on every highlighted interval contains $\left\lfloor \left|\frac{2bk}{c}\right| \right\rfloor + 1$ multiples of π, including 0π.

I_k, because $cos(x)$ is monotonic on any interval $\left[2\pi k - \frac{\pi}{2}, 2\pi k\right]$. Therefore, the number of multiples of π contained in the range of f on I_k is given by

$$\begin{aligned} \Pi(k) &= \left\lfloor \frac{\left|\frac{2\pi bk}{c} - 0\right|}{\pi} \right\rfloor + 1 \\ &= \left\lfloor \left|\frac{2bk}{c}\right| \right\rfloor + 1. \end{aligned} \tag{19}$$

Note that $\Pi(k)$ is equal to the number of roots of g in the interval I_k. Now,

$$\lim_{k\to\infty} \Pi(k) = \infty. \tag{20}$$

The same argument can be used for the sequence of intervals (J_j), where

$$\bigcup_{j\in\mathbb{N}} J_j = (\mathbb{R}^+ \cup \{0\}) - \bigcup_{k\in\mathbb{N}} I_k, \tag{21}$$

Then, by simple inductive argument, it is clear that $\Pi(k)$ also diverges over larger, consecutive intervals. Therefore, the roots of g become more frequent as $x \to \infty$. The same is true for $x \to -\infty$, because g is a sin wave, which is symmetrical.

Because the number of roots of g increases indefinitely over consecutive intervals as $|x| \to \infty$, there can be no sequence of intervals (K_k) such that the length of any interval K_k is equal to the period T, and the number of roots of g in every interval K_k is the same, and

$$\bigcup_{k\in\mathbb{N}} K_k = \mathbb{R}. \tag{22}$$

That is,

$$g(x + T) \neq g(x)\ \forall x \in \mathbb{R}. \tag{23}$$

I.e. g is aperiodic if $bc \neq 0$.

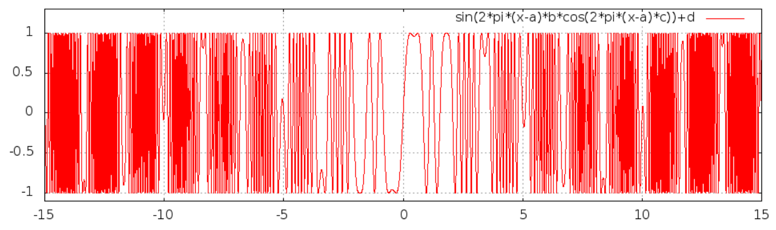

Fig. 3 The roots of g increase in frequency as $|x| \to \infty$.

5.2 Spatial Disconnect

While the problem with periodic generating functions has now been ruled out for the specific case of g, it is clear from section 5.1.1 that the roots of g increase in frequency as $|x| \to \infty$. This effect is illustrated in figure 3. The rate at which the frequency of roots increases is dependent on the c coefficient of g. The perpetual increase in the frequency of roots of g may seem harmless, but it has severe consequences for the algorithm, as explained next.

The angle modulated particle swarm algorithm is concerned with two spaces: the continuous *coefficient space*, where the PSO algorithm is defined, and the binary *solution space*, wherein the solution to the arbitrary binary problem exists. The PSO algorithm works in two primary phases, known as the *exploration* and *exploitation* phases. During the exploration phase, particle step sizes are relatively large so that many parts of the search space are sampled sparsely in order to determine which regions in the search space are likely to contain good solutions. During the exploitation phase, particle step sizes become smaller so that the good regions that were found during the exploration phase can be thoroughly searched in an attempt to find the best solution. When solving continuous problems (which is what PSO was designed for), the magnitude of the velocity of a particle is the distance between its current and previous position. This distance is generally also an indication of how similar two solutions are. The assumption is that if some arbitrary solution in the continuous search space is good, then other close-by solutions are probably also good. However, when particle positions are mapped to binary solutions using angle modulation, it may happen that two solutions which are close to each other in the coefficient space are actually far apart in the solution space. When this happens, the ability of PSO to exploit the solution space is hindered. This problem is henceforth referred to as *spatial disconnect.*

Note that the assumption that good solutions are grouped together is an assumption made by PSO, but not necessarily by all optimisation algorithms. That is, other optimisation algorithms may not be influenced by the spatial disconnect problem. One example of such an algorithm is Simulated Annealing (SA) (Kirkpatrick et al, 1983), where new solutions are not necessarily generated close to the current best solution. The use of SA and other similar algorithms with angle modulation will be investigated in future work.

One way of measuring the similarity between two binary solutions is to calculate the *hamming distance* between them. The hamming distance between two binary strings of equal length is the number of corresponding bits in the two binary strings that differ in value. Figure 4 illustrates the concept of hamming distance. The problem of spatial disconnect can be demonstrated by the following process:

1. Determine a step size s.
2. Uniformly initialise a random vector $\mathbf{p}_1$ in the range $[-1, 1]^4$.
3. Substitute the coefficients of g with the elements of $\mathbf{p}_1$.
4. Generate an 100-dimensional bit string, using g. Call the bit string b_1.
5. Create a uniform random vector $\mathbf{s}$ with length s.
6. Let $\mathbf{p}_2 = \mathbf{p}_1 + \mathbf{s}$.
7. Substitute the coefficients of g with the elements of $\mathbf{p}_2$.
8. Generate an 100-dimensional bit string, using g. Call the bit string b_2.
9. Divide each bit string (b_1 and b_2) into 25 groups of 4 bits.
10. Determine the hamming distance between each of the corresponding 4-bit groups of b_1 and b_2.
11. Repeat steps 2 to 10 n times, and calculate the average hamming distance between the corresponding 4-bit groups of b_1 and b_2.

The process outlined above simulates the movement of a particle from a random point $\mathbf{p}_1$ in the coefficient space to another point $\mathbf{p}_2$, with velocity $\mathbf{s}$. Note that the length s of $\mathbf{s}$ is the amount of change in the coefficient space. By repeating the entire process for decreasing values of s, the average amount of change in the binary solution can be measured and compared to s. By dividing the binary strings b_1 and b_2 into blocks, the amount of change in lower- and higher dimensions of the binary solution can also be compared. For PSO to be able to successfully exploit the binary solution, there must be a direct correlation between the amount of change in the coefficient space and the amount of change in the solution space. Furthermore, because $\mathbf{s}$ is chosen uniformly, and s is constant while repeating steps 2 to 10, the amount of change measured should, on average, be uniform across all dimensions of the solution space.

Figures 5 to 9 show the average hamming distances across all dimensions of the solution space for various values of s. All averages were measured by repeating steps 2 to 10 thirty times. Figure 5 shows that, when $s = 0.1$, the average hamming distance is more or less uniform across all dimensions of the solution space. However, in figures 6 to 9, it is observed that as the value of s is decreased, the distribution of change in the solution space becomes increasingly non-uniform. In particular, while there is a correlation between the amount of change in the coefficient space and the amount of change in the solution space, the correlation is stronger for lower dimensions of the solution space. Hence, a spatial disconnect exists.

The spatial disconnect is caused by a combination of the increasing frequency of roots of g (see section 5.2) and the manner in which g is sampled to generate binary solutions. The original AMPSO algorithm samples g on integer intervals $x = 0, 1, 2, \ldots, n_b - 1$. This relatively fast increase in the value of x means that

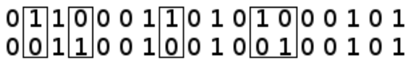

Fig. 4 There are five corresponding bits with different binary values in these two binary strings, so the hamming distance between them is 5.

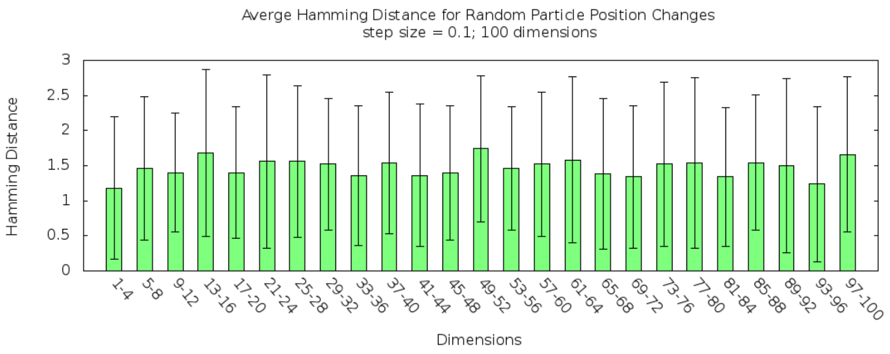

Fig. 5 For particle step sizes of 0.1, the amount of change is uniform across all dimensions of the binary solution.

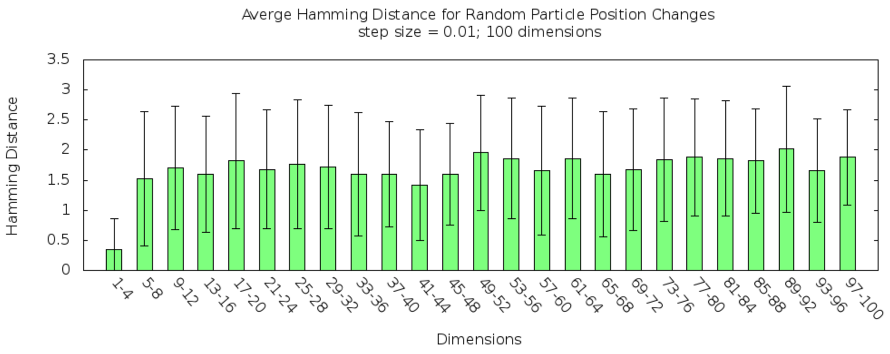

Fig. 6 For particle step sizes of 0.01, the amount of change is low in the first four dimensions of the binary solution.

the frequency of roots of g at the sample positions also increases quickly (albeit, depending on the value of c), as can be seen in figure 3. The higher the frequency at a given sampling position x, the higher the probability that the sign at x will change when the coefficients of g change by a small amount. Recall from section 3 that change in sign at x is equivalent to a bit flip in the binary solution. It is important to realise that the results reported here are independent of any specific binary problem, and that the spatial disconnect is caused by the generating function. Note the absence of a binary problem in steps 1 to 11 above.

The problem of spatial disconnect can be partially overcome by MM-AMPSO, which has the ability to decrease the sampling domain, and thereby the sampling interval, so that x does not increase as fast. This provides an explanation for MM-AMPSO's good performance in some high-dimensional problem cases that were studied in (Leonard and Engelbrecht, 2014). However, this problem persists in both the ID-AMPSO and A-AMPSO variants. Another way to circumvent this problem is to replace the generating function with one whose frequency does not increase with $|x|$.

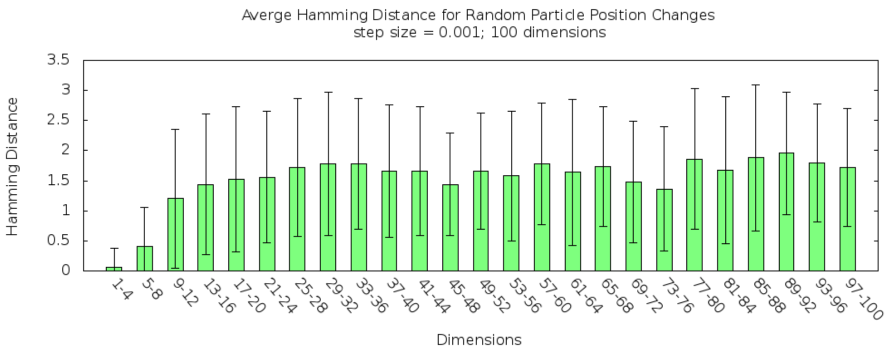

Fig. 7 For particle step sizes of 0.001, the amount of change is low in the first four dimensions of the binary solution, slightly higher in dimensions four to eight, and high in the remaining dimensions.

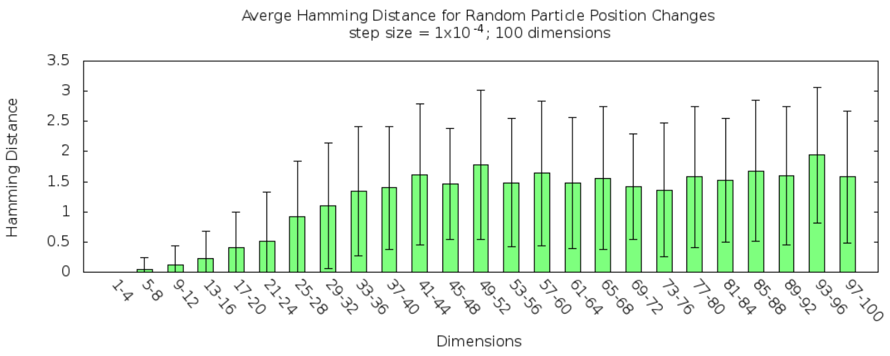

Fig. 8 For particle step sizes of 1×10^{-4}, there is no change in the first four dimensions of the binary solution. The amount of change increases gradually from dimensions five to 40, and remains high in the remaining dimensions.

Another question that arises from the results presented in this section is: do particles in AMPSO eventually slow down sufficiently to exploit the higher dimensions of the solution space? Note that even if particles do slow down sufficiently, the non-uniform distribution of change is still detrimental to performance if the optimal values of lower dimensional bits are in any way dependent on the values of higher dimensional bits. Nonetheless, the question remains interesting. Section 5.3 analyses the velocities of particles in AMPSO throughout the duration of the search process.

5.3 Particle velocities in angle modulated particle swarms

This section presents the first empirical investigation into the behaviour of angle modulated particle swarms, with respect to particle velocities.

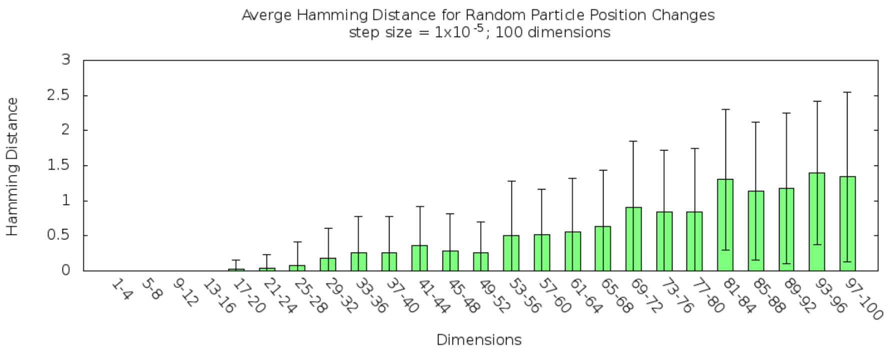

Fig. 9 For particle step sizes of 1×10^{-5}, there is no change in the first 16 dimensions of the binary solution. The amount of change increases gradually from dimensions 17 to 100.

Section 5.3.1 gives an overview of the theoretical aspects of PSO that are of interest to this study. The experimental setup is outlined in section 5.3.2, while the results are presented in section 5.3.3. Section 5.3.4 discusses the implications of the results in terms of the spatial disconnect that was discovered in section 5.2.

5.3.1 Particle swarm optimisation theory

In PSO, a particle i is said to be order-1 stable when the expected position $E[\mathbf{x}_i(t)]$ of the particle converges to some point $\boldsymbol{\mu}$ (Poli, 2009). That is,

$$\lim_{t\to\infty} E[\mathbf{x}_i(t)] = \boldsymbol{\mu}. \tag{24}$$

Poli (2009) also showed that PSO is order-2 stable. That is, the standard deviation σ_i of the particle's sampling position around $\boldsymbol{\mu}$ converges, such that

$$\lim_{t\to\infty} \sigma_i(t) = \frac{1}{2}\sqrt{\frac{c(\omega+1)}{c(5\omega-7)-12\omega^2+12}} \cdot |\hat{\mathbf{y}}(t) - \mathbf{y}_i(t)|, \tag{25}$$

where $c = c_1 = c_2$. Consequently, a particle i will continue searching within a region bounded by σ_i, unless $\mathbf{y}_i(t) = \hat{\mathbf{y}}(t)$. Using the parameter values in table 1, the standard deviation for a particle i at time step t is calculated as follows:

$$\sigma_i(t) = 1.0432 \cdot |\hat{\mathbf{y}}(t) - \mathbf{y}_i(t)|. \tag{26}$$

When σ_i remains stationary, the particle is said to be order-2 stable. The standard deviation $\sigma_i(t)$ can be interpreted as the expected magnitude of the velocity of particle i at time $t+1$, if the particle is order-1 stable. The expected average magnitude of particle velocities in the swarm at time $t+1$ is then given by

$$E[v_{avg}(t+1)] = \frac{\sum_{i=1}^{N} \sigma_i(t)}{N}, \tag{27}$$

where N is the number of particles in the swarm, and the particles in the swarm are order-1 stable.

Table 1 PSO parameter values.

Parameter	**Value**
ω	0.729844
c_1	1.496180
c_2	1.496180

5.3.2 Experimental setup

In order to analyse particle velocities in angle modulated particle swarms, experiments were constructed using AMPSO, A-AMPSO, and MM-AMPSO. ID-AMPSO was not considered further in this study, because there is no difference in performance between ID-AMPSO and AMPSO (see section 4.2).

All algorithms were executed on the same set of problems used in (Leonard and Engelbrecht, 2014), with the exception of the deceptive problems, which were showed to be easily solvable by AMPSO and all three variants. The following problems were evaluated:

– N-Queens (64-, 100-, 400-, and 625 dimensions),
– Knight's Tour (48-, 108-, 300-, and 432 dimensions), and
– Knight's Coverage (64-, 100-, 400-, and 625 dimensions).

In every case, the parameter values in table 1 were used, and the algorithm was allowed to execute for 1000 iterations. The swarm consisted of 20 particles, and the following measurements were recorded at every iteration:

– expected average magnitude of particle velocities (equation (27)),
– actual average magnitude of particle velocities, and
– whether the global best fitness has changed since the previous iteration.

Because the purpose of these experiments is to gain insight into the behaviour of angle modulated swarms by analysing particle velocities during the search process, it does not make sense to average the measurements over a number of independent runs. Thus, although the experiments were performed for 30 independent runs, each of the graphs that are reported section 5.3.3 shows the behaviour of a single execution. However, unless stated otherwise, the results are representative of the kind of behaviour that was observed across independent executions of the experiments.

5.3.3 Results

Figures 10 to 12 show the three most typical cases of the behaviour of angle modulated swarms with respect to average particle velocities. Interestingly, with a few exceptions, these three cases were observed throughout all experiments, across all problems and dimensions. The most typical cases are depicted in figures 10 and 11. The graphs show the actual- and expected average particle velocities in the swarm at each iteration. The intermittent vertical lines indicate when changes in the global best fitness occurred.

In figure 10, particle velocities initially decrease. Changes in the global best fitness are also initially frequent, as expected. However, the average velocity quickly stagnates (after about 100 iterations) at a value between 0.1 and 1. The stagnation is evident from the fact that the expected average velocity $E[v_{avg}]$ converges to a constant value, indicating that the swarm is order-2 stable at this point. Minor fluctuations in $E[v_{avg}]$ are still caused by changes in the particles' personal best positions, but it is not until a change in the global best fitness occurs (around iteration 280 in this case) that the swarm destabilises momentarily. Unfortunately the swarm stabilises at a higher velocity after this event occurs. From equations (26) and (27), it is evident that the global best position has, on average, shifted further away from the particles' personal best positions, causing an increase in average particle velocity.

Another common case is shown in figure 11, where particle velocities initially decrease, until the swarm stabilises, as in the previous case. However, in this case, subsequent changes in the global best position do not destabilise the swarm. The fact that the swarm remains stable sometimes, even though the global best fitness has changed, indicates that the global best position in those cases did not change by a lot. That is, in those cases, the new global best position is relatively close to the old global best position in the coefficient space.

Another, less common, scenario is shown in figure 12. In this case, the initial decrease in average velocity is especially short-lived, while a number of consecutive destabilising changes in the global best position cause the average particle velocity to grow progressively. This indicates that the global best position gradually shifts further away from the particles' personal best positions as the search continues. In these cases, the average velocity of the swarm generally eventually stabilises at a value between 1 and 10.

Finally, figure 13 depicts an uncommon scenario that was only observed for MM-AMPSO in a handful of cases. In this graph, it is observed that the average velocities of particles can grow to values as high as 100 if destabilising changes in the global best position keep occurring.

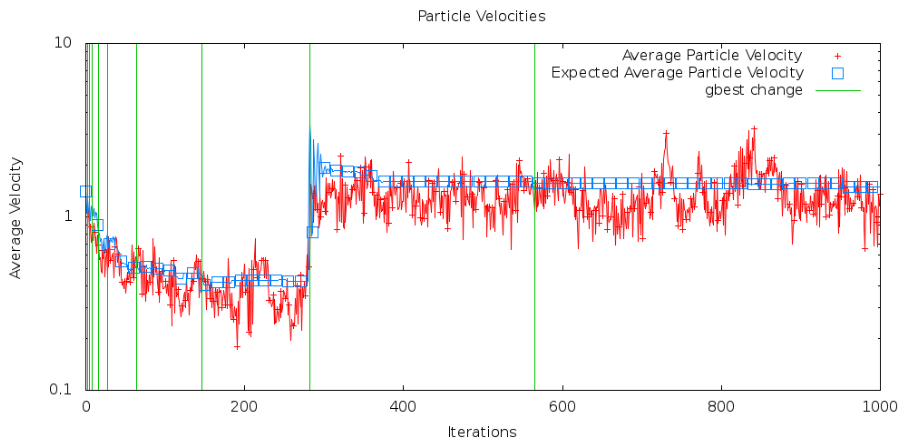

Fig. 10 Particle velocities initially decrease, but stabilise after about 100 iterations. After around 280 iterations, a change in the global best position destabilises the swarm. The swarm then becomes stable at a higher average velocity.

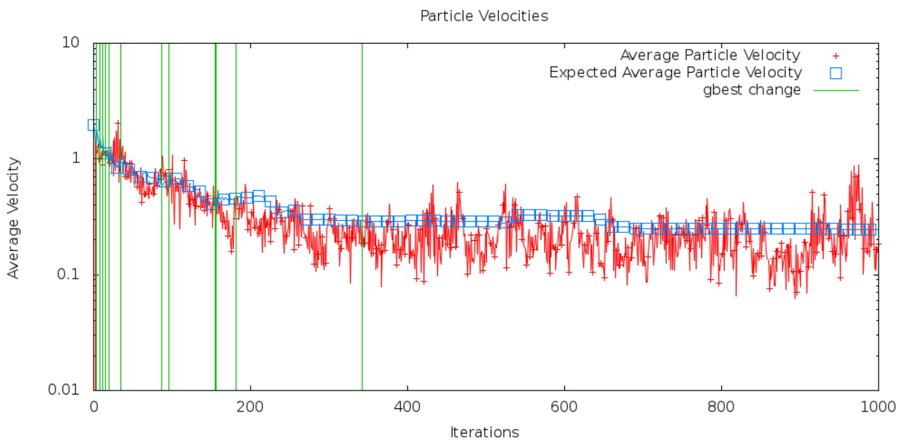

Fig. 11 Particle velocities initially decrease, but the swarm stabilises after a few hundred iterations, with an average particle velocity of between 0.1 and 1.

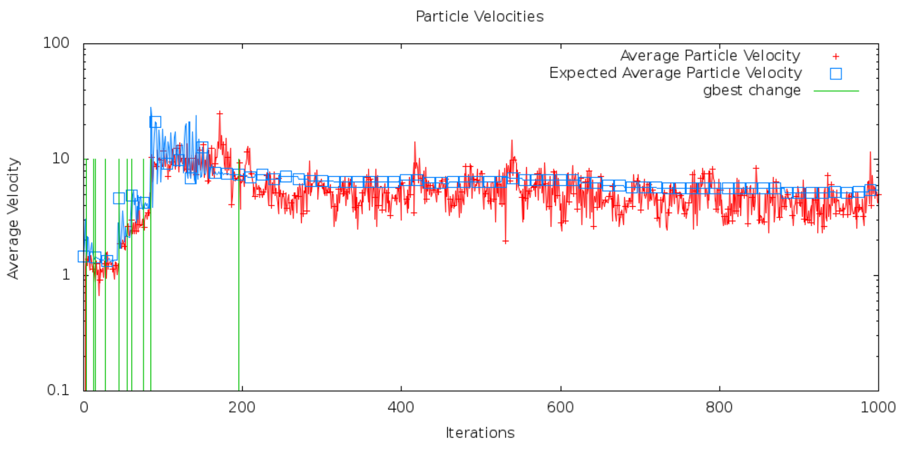

Fig. 12 Particle velocities do not get a chance to decrease, because a number of consecutive changes in the global best position causes the two attractors to become more separated over time.

In a few cases the average velocities of particles were observed to drop below 0.1. However, there were no cases were the average particle velocities ever dropped below 0.01. Henceforth, the phenomenon where particle swarms become order-2 stable at high average velocities will be referred to as *inadequate convergence*.

5.3.4 Discussion

The results shown in section 5.3.3 indicate that particle velocities in AMPSO tend to stabilise at relatively high values. In the most common scenarios, particle velocities stabilised at values between 0.1 and 1.

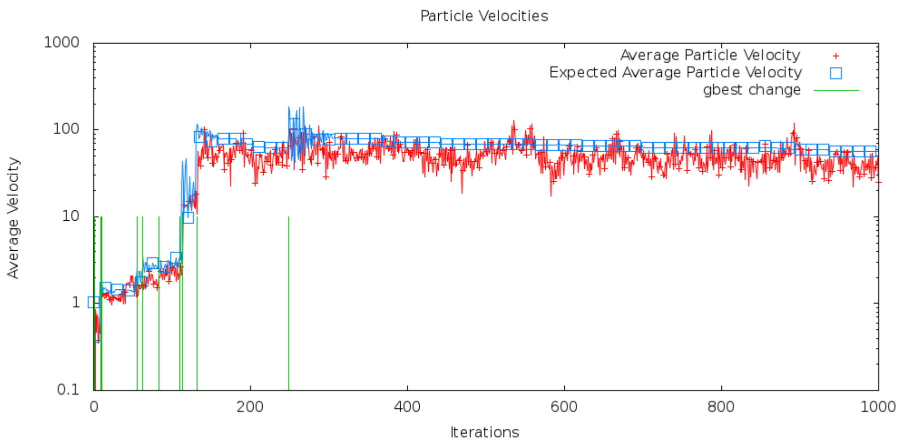

Fig. 13 Particle velocities escalate to extremely high values as the global best position moves very far from the particles' personal best positions, on average.

When these results are compared to those reported in section 5.2, it becomes clear that neither AMPSO, nor any of the variants ever progresses to the exploitation phase of the search process. Recall from figure 5 that a step size as small as 0.1 in the coefficient space causes, on average, 1.5 out of four bits to flip across all dimensions of the binary solution.

While spatial disconnect (see section 5.2) highlights the severity of the problem, it also offers an explanation for inadequate convergence in the case of angle modulated swarms. When a particle i in PSO is attracted towards $\hat{\mathbf{y}}$ and $\mathbf{y}_i$, the assumption is that good solutions are likely to be found around other previously-found good solutions. In moving towards those previously-found locations, the particle progressively discovers better solutions, and $\hat{\mathbf{y}}$ and $\mathbf{y}_i$ move closer together. The effect is that the particle progressively stabilises at lower velocities, until $|\mathbf{y}_i - \hat{\mathbf{y}}|$ becomes sufficiently small, so that σ_i also becomes very small, relative to the search domain. However, in the case of angle modulated swarms, the generating function may cause spatial disconnect, nullifying the assumption that good solutions are grouped together. In this case, while a particle is still attracted towards previously-found good solutions, it is unlikely to find better solutions in those regions, so the distance between $\hat{\mathbf{y}}$ and $\mathbf{y}_i$ does not decrease enough, relative to size of the coefficient space. As a result, the particle becomes order-2 stable at a high velocity. Essentially, a particle in this state indefinitely explores the coefficient space in a region of size σ_i around $\boldsymbol{\mu}$. If, by chance, any particle in the swarm does find a better global best solution, the other particles may become destabilised if the global best position moves by a large enough distance. However, the destabilised particles will yet again stabilise as they fail to find better solutions near the new point of attraction, *ad infinitum*.

The explanation above makes sense, given the evidence presented in this study. However, to confirm that spatial disconnect causes inadequate convergence, a number of additional studies can be performed, for example:

- Investigate the velocities of angle modulated swarms, where the generating function is replaced with one that does not cause spatial disconnect.
- Design a continuous function, where good solutions are deliberately scattered across the search space. If spatial disconnect is the cause of the problem, inadequate convergence should also be observed when using PSO to optimize this function.

A potential solution to the problems that have been discovered so far is to replace the generating function. The problem of finding a new generating function is addressed in section 6.

6 The potential of the generating function

The generating function presented by Pampara et al (2005) was chosen seemingly arbitrarily and without justification. Indeed, the only justification to be found in literature is from Franken (2004), when he postulated the possibility of a "...mathematical function that will allow for more intricate arrangements of 0's and 1's, and adapting this function through the use of PSO."

While all the problems that have been discovered in section 5 have to be considered when choosing a new generating function, another very important question has eluded researchers thus far: can one produce every possible bit string of length n_b by varying the coefficients of the generating function? Or, stated differently, does the generating function have the *potential* to produce any arbitrary bit string of length n_b? This question is of paramount importance, because if the generating function cannot produce any arbitrary bit string, AMPSO might, in some cases, never be able to produce optimal solutions.

A formal definition of generating function potential is given is section 6.1. Section 6.2 demonstrates the use of generating function potential by constructing a novel generating function that can solve simple binary problems. Section 6.3 discusses the implication of generating function potential for lower-than-n_b-dimensional problems once a suitable generating function has been found. Section 6.4 addresses the problem of finding a generating function that has the potential to solve arbitrarily high-dimensional binary problems. Section 6.5 discusses the applicability of generating function potential to AMPSO variants. The section concludes in section 6.6 with a note on the distribution of binary solutions in the search landscape.

6.1 Definition of generating function potential

The following definition assumes that sample values are mapped to binary digits using equation (5).

Given an arbitrary generating function Υ, the potential of Υ to generate any arbitrary bit string of length n_b by varying the coefficients of Υ and sampling the function at regular, fixed intervals is given by

$$P_{\Upsilon}^{n_b} = \frac{|\mathbb{B}_{\Upsilon}^{n_b}|}{2^{n_b}}, \tag{28}$$

where $|\mathbb{B}_{\Upsilon}^{n_b}|$ is the cardinality of the set of binary strings of length n_b that Υ can generate. Equation (28) will henceforth simply be referred to as the n_b-potential of Υ. An n_b-potential of 1 indicates that it is possible to generate every possible bit string of length n_b by varying the coefficients of Υ. Lower values of $P_{\Upsilon}^{n_b}$ indicate that there are bit strings of length n_b that can never be generated by the function.

Note that the n_b-potential of Υ says nothing about the likelihood of generating any specific bit string of length n_b.

To demonstrate the usefulness of generating function potential, an example is given below.

6.2 Example of finding an appropriate generating function

One of the motivations for introducing angle modulation, was the fact that the method reduces the dimensionality of the problem. However, for very low-dimensional binary problems (three dimensions or lower), the dimensionality of the problem actually increases when angle modulation is applied (assuming that g is used as the generating function), because g has four coefficients. This example will show that it is possible to replace the generating function with one that has fewer coefficients to prevent the increase in dimensionality for simple binary problems.

Consider a binary problem with a 3-bit solution, and a simple generating function:

$$h_1(x) = sin(x). \tag{29}$$

This generating function has no coefficients and will therefore always produce the same bit string, assuming that h_1 is sampled at $x = 0, 1, 2$. Table 2 lists all the possible solutions to a 3-dimensional binary problem. Column 2 indicates that h_1 always produces the bit string '011':

$$\begin{aligned} &\text{bit 1: } sin(0) = 0.0 \rightarrow \ 0 \\ &\text{bit 2: } sin(1) \approx 0.8 \rightarrow \ 1 \\ &\text{bit 3: } sin(2) \approx 0.9 \rightarrow \ 1, \end{aligned} \tag{30}$$

which means that $|\mathbb{B}_{h_1}^3| = |\{\text{'011'}\}| = 1$. Therefore, the 3-potential of $h_1(x)$ is

$$\begin{aligned} P_{h_1}^3 &= \frac{|\mathbb{B}_{h_1}^3|}{2^3} \\ &= \frac{1}{8} \\ &= 0.125. \end{aligned} \tag{31}$$

The low 3-potential of h_1 indicates that this function would be a very poor choice of generating function to use with angle modulation. Of course, h_1 is clearly a toy example, because there are no coefficients to optimise.

Consider a slightly more complicated function:

$$h_2(x) = sin(vx). \tag{32}$$

Adding the v coefficient means that the function can now produce different bit strings by varying v. It also implies 1-dimensional particles in the case of AMPSO:

$$\mathbf{x}_i = (v). \tag{33}$$

Table 2 Different binary solutions are obtained by varying the value of v. Assuming that $x \in \{0, 1, 2\}$, a '-' indicates that it is not possible to generate a particular solution. A '*' indicates that a particular solution is always generated.

Solution	$h_1(x) = sin(x)$	$h_2(x) = sin(vx)$	$h_3(x) = sin(v(x+1))$
		v	v
000	–	5	6
001	–	4	5
010	–	2	4
011	*	1	17
100	–	–	2
101	–	–	3
110	–	–	14
111	–	–	1

Table 2 shows that there exist four values for v that produce the first four possible solutions, respectively. These values were found empirically, assuming the same sampling positions as for h_1. However, no values for v could be found to produce the remaining four solutions. The fact that no values for v could be found to produce the last four solutions does not necessarily imply that it is impossible to produce those solutions. However, in this case it is easy to show that h_2 cannot generate solutions whose first bit is '1'. Indeed, consider the calculation of the first bit:

$$h_2(0) = sin(v(0)) = 0.0 \to 0, \tag{34}$$

regardless of the value of v. Therefore, $|\mathbb{B}^3_{h_2}| = 4$, and $P^3_{h_2} = 0.5$. The 3-potential of h_2 is a significant improvement over the 3-potential of h_1, but still not good enough.

The problem with h_2 can easily be corrected by adding a constant horizontal shift term to prevent the first bit from always being sampled at $sin(0)$:

$$h_3(x) = sin(v(x+1)). \tag{35}$$

Table 2 shows that, for h_3, there exist values for v to generate every possible 3-bit binary solution. Hence, $P^3_{h_3} = 1$, meaning that h_3 a generating function, with a single coefficient, that has the potential to solve any 3-dimensional binary problem. Table 2 also shows that all the possible solutions exist for $1 \leqslant v \leqslant 17$. Therefore, a suitable initialisation range for particles in the PSO algorithm has also been found (although a smaller range that is equally suitable might exist).

A point of concern, regarding h_3, is that the function is clearly periodic and may therefore give rise to repetition in the binary solution. However, as indicated in section 5.1, periodic generating functions are of greater concern for higher values of n_b. In the case where $n_b = 3$, a periodic generating function is not problematic.

6.3 Implication for lower-than-n_b-dimensional binary problems

Another useful observation to make from table 2 is that the solutions to every 2-bit binary problem are contained in the 3-bit solutions. That is, if one ignores the final bit of every 3-bit solution in table 2, then every 2-bit binary string is present in the list. The implication is that if a generating function is able to generate all

3-bit binary strings, then it is also able to generate all 2-bit binary strings, simply by not sampling the final bit. By the same reasoning, it then follows that all 1-bit solutions can also be generated. This implication necessarily holds for any value of n_b and can be formally stated as follows:

$$P_{\Upsilon}^{n_b} = 1 \quad \Rightarrow \quad P_{\Upsilon}^{n_b-1} = P_{\Upsilon}^{n_b-2} = \ldots = P_{\Upsilon}^{1} = 1. \tag{36}$$

Therefore, if $P_{\Upsilon}^{n_b} = 1$, then Υ is a generating function with the potential to solve any binary problem with dimensionality n_b or lower.

6.4 Finding generating functions to solve high-dimensional binary problems

It has now been shown that generating function potential can be used to quantify the usefulness of any arbitrary generating function. However, up to this point, only very simple functions were considered. This section investigates the use of generating function potential to find appropriate generating functions to solve high-dimensional binary problems.

Section 6.4.1 presents an empirical approach to estimate the potential of an arbitrary generating function. Section 6.4.2 introduces the use of multiple generating functions as an alternative approach to solving arbitrarily high-dimensional binary problems.

6.4.1 An empirical approach to estimate generating function potential

The most challenging aspect of determining the n_b-potential of a generating function Υ, is finding the set $\mathbb{B}_{\Upsilon}^{n_b}$ of bit strings that Υ can generate. One conceivable way of generating $\mathbb{B}_{\Upsilon}^{n_b}$ is to sample Υ at $x = 0, 1, 2, \ldots, n_b - 1$ for every possible permutation of the coefficients of Υ and recording which bit strings get generated. Of course, this is impossible, because the coefficients of Υ are continuous. However, the values of the coefficients can be limited to some finite subset $\mathbb{S}$ of $\mathbb{R}$. Let C_{Υ} denote the number of coefficients of Υ. By limiting the values that the coefficients may assume, a set $\mathbb{P}_{C_{\Upsilon}}$ of all possible permutations of the coefficients of Υ can be generated. The number of permutations $|\mathbb{P}_{C_{\Upsilon}}|$ is given by

$$|\mathbb{P}_{C_{\Upsilon}}| = |\mathbb{S}|^{C_{\Upsilon}}. \tag{37}$$

For example, if Υ has two coefficients whose values are limited to $0.0, 0.1$, and 0.2, then the permutations listed in table 3 are possible, and

$$\begin{aligned} |\mathbb{P}_{C_{\Upsilon}}| &= |\mathbb{S}|^{C_{\Upsilon}} \\ &= 3^2 \\ &= 9. \end{aligned} \tag{38}$$

The complexity of generating a bit string of length n_b from Υ, using every possible permutation in $\mathbb{P}_{C_{\Upsilon}}$ is

$$O(n_b \cdot |\mathbb{P}_{C_{\Upsilon}}|) = O(n_b \cdot |\mathbb{S}|^{C_{\Upsilon}}), \tag{39}$$

which increases exponentially with the number of coefficients C_{Υ}.

If the method described above is used to calculate the n_b-potential of an arbitrary generating function, the following should be noted:

1. Because $\mathbb{S}$ is a finite subset of $\mathbb{R}$, the calculated n_b-potential is only an estimate of the function's true n_b-potential.
2. The true n_b-potential of the function is necessarily greater than or equal to the estimated value. Therefore, if the estimated n_b-potential of Υ equals 1, then $P_{\Upsilon}^{n_b} = 1$.
3. To obtain the best estimate of the true n_b-potential of the function, $|\mathbb{S}|$ should be as large as possible.
4. Because any given permutation will always generate the same binary solution, $\mathbb{S}$ must be chosen such that
$$|\mathbb{P}_{C_\Upsilon}| \geqslant 2^{n_b}. \tag{40}$$
Without this constraint, an estimated n_b-potential of 1 can never be obtained.

The n_b-potential of g (equation (3)) was estimated using the approach outlined above, with $\mathbb{S} = \{-1.000, -0.975, -0.950, \ldots, 0.00, 0.025, 0.050, \ldots, 1.000\}$, and $n_b = 16$. Therefore, from equation (38),

$$\begin{aligned} |\mathbb{P}_{C_g}| &= |\mathbb{S}|^{C_g} \\ &= 81^4 \\ &= 43\ 046\ 721, \end{aligned} \tag{41}$$

which satisfies the constraint in equation (40):

$$\begin{aligned} |\mathbb{P}_{C_g}| &\geqslant 2^{n_b} \\ 43\ 046\ 721 &\geqslant 2^{16} \\ &\geqslant 65\ 536. \end{aligned} \tag{42}$$

Generating a bit string from g for each of the 81^4 permutations of the coefficients of g, yielded an estimated 16-potential of 1. That is, every possible bit string of length 16 could be generated. Therefore,

$$P_g^{16} = 1. \tag{43}$$

Increasing n_b to 17 still satisfied the constraint in equation (40), but yielded an estimate of $P_g^{17} \geqslant 0.99932$. In order to prove that $P_g^{17} = 1$, $|\mathbb{S}|$ needs to be further increased. Unfortunately, this approach quickly becomes infeasible, because of the increasing computational complexity to estimate $|\mathbb{B}_{\Upsilon}^{n_b}|$.

6.4.2 Ensemble generating functions

An alternative method to solve arbitrarily high-dimensional binary problems is to make use of multiple generating functions to solve parts of the binary solution independently. A separate generating function would then be used for each $\frac{n_b}{\varphi}$-bit group of bits in the binary solution, where φ is the total number of generating functions. Together, the φ generating functions are referred to as an *ensemble generating function* Θ. Assuming that the φ generating functions are all the same function θ, and that $P_\theta^{\frac{n_b}{\varphi}} = 1$, it follows that $P_\Theta^{n_b} = 1$. The PSO that results from using Θ as the generating function has dimensionality

$$\begin{aligned} n_x &= \frac{n_b}{\frac{n_b}{\varphi}} \cdot C_\theta \\ &= \varphi C_\theta. \end{aligned} \tag{44}$$

Table 3 Each line in this table represents one possible permutation of the coefficients of Υ. Each permutation can potentially generate a different binary solution.

Coefficent 1	Coefficient 2
0.0	0.0
0.0	0.1
0.0	0.2
0.1	0.0
0.1	0.1
0.1	0.2
0.2	0.0
0.2	0.1
0.2	0.2

The position of a particle i is then given by

$$\mathbf{x}_i = (\psi_{\theta_1 1}, \psi_{\theta_1 2}, \ldots, \psi_{\theta_1 C_\theta}, \psi_{\theta_2 1}, \psi_{\theta_2 2}, \ldots, \psi_{\theta_2 C_\theta}, \\ \ldots, \psi_{\theta_\varphi 1}, \psi_{\theta_\varphi 2}, \ldots, \psi_{\theta_\varphi C_\theta}), \tag{45}$$

where $\psi_{\theta_i j}$ is the j^{th} coefficient of the i^{th} θ function.

In the specific case of h_3, this implies that an n_b-dimensional binary problem can be solved using an $\frac{n_b}{3}$-dimensional PSO, with particle positions of the form

$$\mathbf{x}_i = (v_1, v_2, v_3, \ldots, v_\varphi). \tag{46}$$

Of course, g is also a potential candidate to replace θ. In the case of g, the resulting PSO has dimensionality $\frac{n_b}{4}$, from equations (43) and (44). However, recall from figure 7 that the problem of spatial disconnect already manifests when g is used to solve 16-dimensional binary problems.

6.5 Using generating function potential with AMPSO variants

One point of concern regarding generating function potential is whether it is applicable to the AMPSO variants, described in section 4. In the case of ID-AMPSO, no modifications were made to the algorithm, so generating function potential automatically applies. In the case of A-AMPSO, an additional coefficient was introduced, but the definition of generating function potential is not dependent on the number of coefficients, so generating function potential also applies to A-AMPSO. The only point of possible confusion is MM-AMPSO (refer to section 4.3). The MM-AMPSO algorithm has the ability to vary its sampling domain. More precisely, the way in which MM-AMPSO calculates the sample positions might cast some doubt regarding the applicability of the implication for lower-than-n_b-dimensional problems, discussed in section 6.3.

Consider the case where $\alpha_l = 0$, $\alpha_u = 5$, and $n_b = 5$. From equation 9, $\delta = 1$ for this particular configuration. Further assume that $a = 0$, $b = 0.5$, $c = 0.8$, $d = 0$. When a bit string is generated, the scenario depicted in figure 1 is obtained. Thus, the generated bit string is '01100'. Now, according to section 6.3, the bit string '0110' should be obtainable by neglecting to sample the final bit. To do so, we reduce the value of n_b to 4. But reducing the value of n_b changes the value of δ, and therefore also the sampling positions. This means that some bits in the newly generated 4-dimensional bit string might change. The concern in this case is that it is no longer guaranteed that the generating function will be able to produce all lower-dimensional bit strings. Fortunately, this effect is easily corrected by adjusting the value of α_u. In this particular case, the value of α_u can be set to 4 to obtain the correct value for δ to produce the desired bit string.

From the example above, it is clear that by making the necessary adjustment to α_u, MM-AMPSO can also generate any lower-than-n_b-dimensional bit string, assuming that the generating function has an n_b-potential of 1. Thus, the concept of generating function potential, including the implication for lower-than-n_b-dimensional problems, is equally applicable to AMPSO and all the AMPSO variants contained in this study.

6.6 Distribution of Binary Solutions

As a final thought on generating function potential, note that $P_{\Upsilon}^{n_b}$ says nothing about the distribution of solutions in the search space. Consider again the illustration given in figure 1. In that illustration, the values $a = 0$, $b = 0.5$, $c = 0.8$, and $d = 0$ produced the binary solution "01100". However, it is trivial to see that choosing a sufficiently small, but non-zero positive value for a would produce the exact same binary solution.

The fact that different permutations of the coefficients can produce the same binary solution has two obvious effects. Firstly, the search space contains plateaus, and, secondly, all solutions are not guaranteed to exist in the search space *with the same frequency*. That is, even if $P_{\Upsilon}^{n_b} = 1$, meaning that every possible binary solution exists in the search space, some solutions may be much more common (and therefore easier to find) other solutions.

This problem would be difficult to overcome by simply replacing the generating function with some new function, since every continuous function would conceivably present the same problem. A thorough investigation into the implications of this problem is left for future work.

7 Conclusion and Future Work

This study investigated various aspects of angle modulated particle swarm optimisers in order to understand why the algorithm might fail to optimise arbitrary binary problems.

The periodicity of the generating function was investigated and found not to be a problem in the standard AMPSO algorithm or any of its existing variants. However, it was discovered that the roots of the generating function increase in frequency along the x-axis. This characteristic was shown to cause a basic assumption made by PSO (that good solutions are grouped together in the search space) to be violated. The problem is referred to as *spatial disconnect*, and was shown to manifest in all existing AMPSO variants, regardless of which binary problem is being optimised. Thus the problem is applicable to anyone who uses AMPSO or any of its variants.

The study also provided the first empirical analysis of particle convergence in angle modulated particle swarm optimisers. It was shown that particles in AMPSO tend to stabilise at high velocities, with respect to the size of the search domain. This problem is referred to as *inadequate convergence*. The most important consequence of inadequate convergence is that particles in AMPSO are not able to exploit good solutions in the search space, because their step sizes remain too high. The problem was explained in terms of the spatial disconnect that was discovered earlier.

In general, the problems discovered with AMPSO in this study were all associated with the generating function. In particular, spatial disconnect seemed to be the cause of these problems. Thus, the best way to address the problems found in this study would be to replace the generating function with one that does not cause spatial disconnect.

To this end, the study outlined the first formal definition to quantify the ability of arbitrary generating functions to solve arbitrary binary problems. This quantity is referred to as the *potential* of the generating function. Using the definition of generating function potential, a new generating function was constructed to solve binary problems with fewer than four dimensions. This new generating function has a single coefficient, meaning that a reduction in dimensionality when solving simple binary problems using AMPSO was made possible for the first time. Furthermore, the use of multiple generating functions in AMPSO was proposed as a method to solve arbitrarily high-dimensional binary problems. Generating functions consisting of multiple functions that solve parts of the binary problem are referred to as *ensemble generating functions*.

The insights gained from this study pave the way for various additional studies, as well as potential improvements to angle modulated particle swarm optimisers. In future work, studies will be carried out to test the various hypotheses that were formulated in this article. Such studies will include tests to confirm that spatial disconnect is the cause of inadequate convergence in angle modulated particle swarms, as well as empirical analyses comparing the use of a single generating function with the use of ensemble generating functions.

Acknowledgements The authors would like to thank the reviewers of this article for their valuable comments and suggestions.

References

Engelbrecht A (2012) Particle swarm optimization: Velocity initialization. In: IEEE Congress on Evolutionary Computation, pp 1–8

Franken N (2004) PSO-based coevolutionary game learning. Master's thesis, University of Pretoria

Holland J (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol 4, pp 1942–1948

Kennedy J, Eberhart R (1997) A discrete binary version of the particle swarm algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol 5, pp 4104–4108

Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simmulated annealing. Science 220(4598):671–680

Leonard B, Engelbrecht A (2014) Angle modulated particle swarm variants. In: Swarm Intelligence, vol 8667, LNCS, pp 38–49

Pampara G (2013) Angle modulated population based algorithms to solve binary problems. Master's thesis, University of Pretoria

Pampara G, Franken N, Engelbrecht A (2005) Combining particle swarm optimisation with angle modulation to solve binary problems. In: IEEE Congress on Evolutionary Computation, 2005, vol 1, pp 89–96

Pampara G, Engelbrecht A, Franken N (2006) Binary differential evolution. In: IEEE Congress on Evolutionary Computation, pp 1873–1879

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