Probabilistically Driven Particle Swarms for Optimization of Multi Valued Discrete Problems : Design and Analysis

Kalyan Veeramachaneni, Lisa Osadciw, Ganapathi Kamath
Department of Electrical Engineering and Computer Science
277, Link Hall
Syracuse University, Syracuse NY –13244
kveerama, laosadci,gkamathh@syr.edu

***Abstract*—A new particle swarm optimization (PSO) algorithm that is more effective for discrete, multi-valued optimization problems is presented. The new algorithm is probabilistically driven since it uses probabilistic transition rules to move from one discrete value to another in the search for an optimum solution. Properties of the binary discrete particle swarms are discussed. The new algorithm for discrete multi-values is designed with the similar properties. The algorithm is tested on a suite of benchmarks and comparisons are made between the binary PSO and the new discrete PSO implemented for ternary, quaternary systems. The results show that the new algorithm's performance is close and even slightly better than the original discrete, binary PSO designed by Kennedy and Eberhart. The algorithm can be used in any real world optimization problems, which have a discrete, bounded field.**

I. INTRODUCTION

Many real world optimization problems, like design optimization in transistor sizing problems, are discrete and multi-valued. Any field, whose values are discrete, need a discrete, multi-valued optimization algorithm. There is an increasing need to develop these types of algorithms as the uses for optimization algorithms grow. In this paper, a new algorithm based on particle swarm optimization is presented. We analyze the performance of the discrete, binary PSO developed by Kennedy, et al.,[8] to identify the critical properties needed by discrete PSOs. We design a particle swarm for the discrete, multi-valued optimization problems that exhibits the same critical properties as the original discrete, binary PSO. The new algorithm's performance is tested using benchmarks adapted from Liang et al. [11].

The particle swarm optimization algorithm, originally introduced in terms of social and cognitive behavior by Kennedy and Eberhart in 1995 [1], can effectively solve optimization problems in many fields, especially engineering and computer science. The power in the technique is its fairly simple computations and sharing of information within the algorithm as it derives its internal communications from the social behavior of individuals. The individuals, called particles henceforth, are flown through the multi-dimensional search space with each particle representing a possible solution to the multi-dimensional problem. Each solution's fitness is based on a performance function related to the optimization problem being solved.

The movement of the particles is influenced by two factors using information from iteration-to-iteration as well as particle-to-particle. As a result of iteration-to-iteration information, the particle stores in its memory the best solution visited so far, called *pbest,* and experiences an attraction towards this solution as it traverses through the solution search space. As a result of the particle-to-particle information, the particle stores in its memory the best solution visited by any particle, and experiences an attraction towards this solution, called *gbest,* as well. The first and second factors are called cognitive and social components, respectively. After each iteration the *pbest* and *gbest* are updated for each particle if a better or more dominating solution (in terms of fitness) is found. This process continues, iteratively, until either the desired result is converged upon, or it's determined that an acceptable solution cannot be found within computational limits.

The PSO formulae define each particle in the D-dimensional space as $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$, where the subscript i represents the particle number, and the second subscript is the dimension. The memory of the previous best position, *pbest,* is represented as $P_i = (p_{i1}, p_{i2}, ..., p_{iD})$ and velocity as $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$. After each iteration, the velocity term is updated influenced by both its own best position, P_i , and the global best position, P_g. The velocity update equation is

$$V_{id}^{(t+1)} = \omega \times V_{id}^{(t)} + U[0,1] \times \psi_1 \times (p_{id}^{(t)} - x_{id}^{(t)}) + U[0,1] \times \psi_2 \times (p_{gd}^{(t)} - x_{id}^{(t)}) \quad (1)$$

1-4244-0708-7/07/$20.00 ©2007 IEEE

where U[0,1] is a sample from a uniform random number generator, *t* represents a relative time index, ψ_1 is a weight determining the impact of the previous best solution, and ψ_2 is the weight on the global best solution's impact on particle velocity. The next solution to test is

$$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)} \quad . \qquad (2)$$

The next solution in (2) is defined in a continuous valued solution space.

Kennedy and Eberhart [8] designed a discrete version of the algorithm, which is significantly different from the continuous version. The algorithm uses the same velocity update equation but the values for the solution or 'X' are now discrete and binary. Kennedy, et al., preserved the social and cognitive learning components in the algorithm but changed the particle solution updating.

In the next section, the motivation for designing a particle swarm algorithm for discrete multi-valued optimization problems is presented. Analysis of probabilistic transitions made in the binary PSO is presented in Section 3. The new PSO algorithm for discrete multi-valued optimization problem is presented in Section 4. In Section 5, the benchmarks and the experimental settings are described. Results are presented in Section 6 followed by conclusions and future work in Section 7.

II. Motivation : Discrete Multi Valued Particle Swarm Optimization

Recently, there has been an increasing interest in developing particle swarm optimization based algorithm for discrete multi-valued optimization problems [12, 13]. Many real world optimization problems have discrete variable values. It can be argued that discrete variables can be transformed into an equivalent binary representation, and the binary PSO can be used. However, the range of the discrete variable often does not match the upper limit of the equivalent binary representation. For example, a discrete variable of range [0,1,2,3,4,5] requires a three bit binary representation, which ranges between [0-7]. Thus, special conditions are required to manage the values past the original range of the discrete variable. Secondly, the Hamming distance between two discrete values undergoes a non-linear transformation when an equivalent binary representation is used instead. This often adds complexity to the search process. The third reason is that the binary representation increases the dimensions of the particle. For these reasons, an extension to the original discrete, binary PSO converting it to a discrete multi-valued PSO is necessary.

Previously researchers have attempted to enhance the performance of the binary PSO. Al Kazemi, et al. [10], improves the original binary PSO algorithm by modifying the way particles interact. The research on algorithms that optimize discrete, multi-valued problems, however, is sparse.

With the new algorithm, any discrete multi-valued problem can be optimized using particle swarms without converting the problem into equivalent binary representations. The algorithm's properties and probabilistic rules are introduced and discussed in Section IV. The movement of the particles through solution value updates remains probabilistic for performance reasons. Results are presented for a suite of 5 benchmark problems adapted from [11].

III. Properties and Analysis of Discrete Binary PSO

Kennedy and Eberhart designed a discrete version of the PSO algorithm. The algorithm uses the same velocity update equation as in (1) but the values of 'X' are now discrete and binary. For position update, first the velocity is transformed into a [0, 1] interval using the sigmoid function given by

$$S_{id} = sig(V_{id}) = \frac{1}{1+e^{-V_{id}}} \qquad (3)$$

where V_{id} is the velocity of the i^{th} particle's d^{th} dimension. A random number is generated using a uniform distribution which is compared to the value generated from the sigmoid function and a decision is made about the X_{id} in the following manner.

$$X_{id} = u(S_{id} - U[0,1]) \qquad (4)$$

u is a unit step function. The decision regarding X_{id} is now probabilistic, implying that higher the value of the V_{id}, higher the value of the S_{id}, making probability of deciding '1' for X_{id} higher. As $V_{id} \to \infty$, then $S_{id} \to 1$ making it unlikely that X_{id} will become zero again. Figure 1 shows this property of the probability, $X_{id}=1$ increases as V_{id} increases, in the binary PSO. However, $P(X_{id}=1)$ is almost equal to 1 for $V_{id}>10$ but not equal to 1. This is the key to the design of the discrete binary PSO, since this prevents particles from getting stuck once. However, it raises some important questions about the velocity. Will the velocity of a particle tend toward infinity if the local optimum is at (1,1) for (P_{id}, P_{gd})? What parameter values result for higher velocities? Can we analyze swarm's behavior for such conditions? Finally, can we design a similar model for a discrete multi-valued optimization problem?

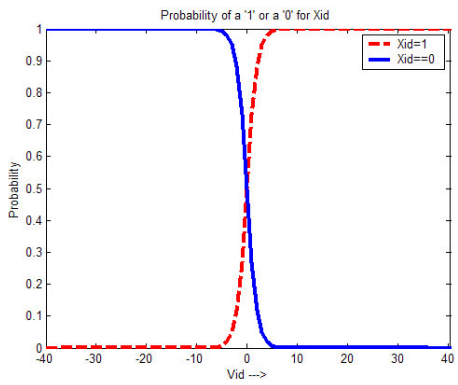

Figure 1. Probability of X_{id} =1 and X_{id}=0 given the V_{id}

In the velocity update function (1) in the Binary PSO, one or both terms of influence become equal to zero depending on the difference between the current solution, local best solution, $P_{id,}$ and the global best solution, P_{gd}. When the influences are not to zero, the number added to the previous velocity (ωV) comes from standard distributions. Unlike the continuous PSO, these distributions do not get scaled, nor are their properties changed over iterations. This makes analysis of binary PSO easier than the continuous PSO. In the next section, four different cases and distributions used in the velocity equation (1) are described.

Ozcan, et al., [4], presented the theoretical framework for the analysis of a continuous PSO. A simple particle's behavior in one-dimensional space is analyzed leading to a conclusion that the particle follows a sinusoidal path, when 'gbest' and 'pbest' are fixed. In this paper, the discrete binary particle swarm optimization algorithm is analyzed for fixed 'gbest' and 'pbest'. Since the values of the 'pbest' and 'gbest' are binary, there are four cases. We generate the probability density of velocity for these different cases analyzing the behavior of a single particle in a one-dimensional space.

The probability density of velocity is formed through Monte Carlo runs. First the algorithm is run by starting the algorithm at different velocities [-10,10] and is run for 2000 iterations each time for a fixed case i.e., fixed '*gbest*' and '*pbest*'. This captures the steady state behavior of the particle. This set of runs can be repeated forming an average probability density of the velocity for each specific case. The probability density of velocity is analyzed for different values of ψ_1, ψ_2 and ω in the four cases.

A. 4 Case Analysis

Considering a single dimension and a single particle. Let us drop the '*id*' notation in (1) as 'i' implies particle number and 'd' dimension number. Four cases occur in the discrete binary PSO with each causing certain behaviors in the velocity. *Pbest* is now P; *gbest* is now G for simplicity. $X^{(t)}$ is the current position of the particle. The analysis assumes parameter values of $\psi_1 \geq 1$, $\psi_2 \geq 1$, $0 \leq \omega \leq 1$. The generalized form of velocity update equation for discrete binary PSO is given by

$$V^{(t+1)} = \omega \times V^{(t)} + \psi \tag{5}$$

ψ is given by

$$\psi = U[0,1] \times \psi_1 \times (P - X) + U[0,1] \times \psi_2 \times (G - X) \tag{6}$$

and is a random number that comes from different distributions for different cases.

1) Case 1 : P=0; G=1;

Depending upon the current position of the particle one of the two influences become zero. Thus, only one influence remains in the velocity update equation. When $X^{(t)} = 1$, ψ is a random number generated from the uniform distribution $U[-\psi_1, 0]$. When $X^{(t)} = 0$, it is a random sample generated from uniform distribution $U[0, \psi_2]$. In this case, the particle will swing between values '0' and '1' until '*G*=0'. The particle then falls into case 4.

Figure 2 shows the probability density of velocity when '*gbest*' is G=1 and '*pbest*'is P=0. The two variables, $[\psi_1, \psi_2]$, are varied to analyze the affect. The probability density of the velocity is centered around zero, and an increase in $[\psi_1, \psi_2]$ increases the standard deviation of the probability density of the velocity but still centers around zero. The statistics collected for the position of the particle revealed a 50-50% distribution for a value of X=1 and X=0. Change in ψ_1, ψ_2 did not affect this distribution.

2) Case 2 : P=1; G=0;

This case is similar to Case 1 with the particle will swinging between values of '0' and '1' until 'G=1'. The particle then behaves as in 3rd case. In case P=0 the particle behaves as in 4th case.

3) Case 3 : P=1; G=1

The velocity changes when the particle value is 0, $X^t = 0$ and

$$\psi = U[0,1] \times \psi_1 + U[0,1] \times \psi_2 \tag{7}$$

Hence, the convolution of two uniform distributions yields the triangular distribution (when $\psi_1 = \psi_2$) that is

$$f(\psi) = \begin{cases} \dfrac{\psi}{\psi_1 \psi_2} & 0 \leq \psi < \psi_1 \\ \dfrac{\psi_2 + \psi_1 - \psi}{\psi_1 \psi_2} & \psi_1 \leq \psi < \psi_2 + \psi_1 \end{cases} \tag{8}$$

, when $X^{(t)} = 1$, the random term is simply, $\psi = 0$ and velocity gets scaled by ω.

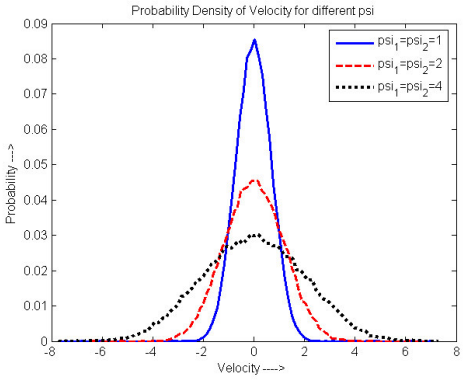

Figure 2. Probability density of the velocity for different ψ_1, ψ_2 for Case 1.

This case is important to analyze since due to the sigmoid function there is always a probability that the particle's current position will become '0' increasing the velocity of the particle whose P stays at 1 and G at 1.

The probability density for the velocity is plotted for different ψ_1, ψ_2 in Figure 3. It can be seen as ψ_1, ψ_2 increases the probability density of the velocity spreads over a wider range of velocities. Hence by controlling the values of ψ_1, ψ_2 one can control the values of the velocity. Three different cases of ψ_1, ψ_2 are shown in the figure. The positions of the particle had a 75% and 25 % distribution for X=1 and X=0 respectively, for $\psi_1 = \psi_2 = 1$, as shown in Figure 4. When ψ_1, ψ_2 are increased to 4 the positions of the particles had a distribution of 86.8% and 13.2% for X=1 and X=0 respectively. Increasing the ψ_1, ψ_2 stabilizes the positions of the particles since there are higher velocities that are possible.

Similar analysis can be done to demonstrate the affect of ω on the probability density of velocity.

4) Case 4: P=0; G=0

The velocity is only changed in this case when the current value of the particle position, i.e., $X^t = 1$ and ψ is a random number generated from the triangular distribution in (8). When $X^{(t)} = 0$, the random value is $\psi = 0$.

This analysis shows that the probability of higher velocities is very low even for high values of $\psi_1 = \psi_2 = 4$.

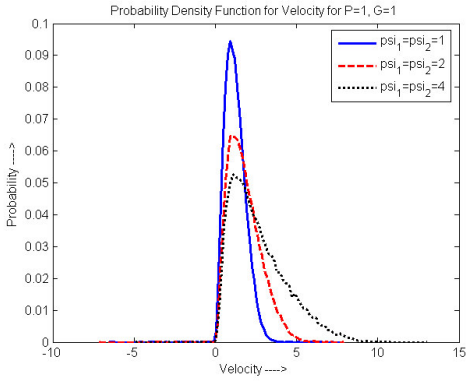

Figure 3. Probability density of velocities for the Case 4 for different ψ_1, ψ_2.

IV. DISCRETE MULTI VALUED PARTICLE SWARM OPTIMIZATION

For discrete multi valued optimization problems the range of the discrete variable values between [0 M-1], where 'M' implies the M-ary number system. The same velocity update and particle representation are used in the algorithm as for the binary valued PSO. The position update equation is however changed in the following manner. The velocity is transformed into a number between [0, M] using the sigmoid transformation,

$$S_{id} = \frac{M}{1 + e^{-V_{id}}} \qquad (9)$$

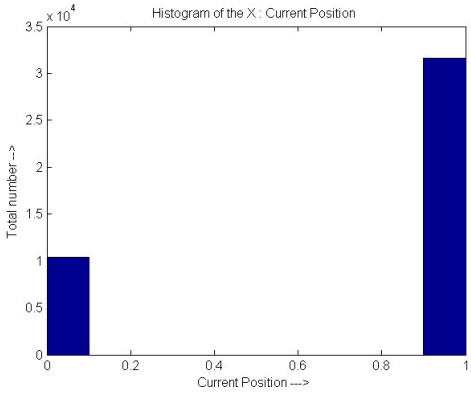

Figure 4. Histogram of the current position X for Case 3 for $\psi_1 = \psi_2 = 1$; $\omega = 0.8$

A number is generated using the normal distribution with parameters $N(S_{id}, \sigma(M-1))$. The result is rounded to

$$X_{id} = round(S_{id} + (M-1) \times \sigma \times randn(1)) \qquad (10)$$

If

$$X_{id} = \begin{cases} M-1, & X_{id} > M-1 \\ 0, & X_{id} < 0 \end{cases}$$

The velocity update equation remains the same as (1). The positions of the particles are discrete values between [0, M-1]. Note that for any given S_{id} there is a probability for choosing a number between [0, M-1]. However, in this paper, the probability of selecting a number decreases based on its distance from S_{id}. In the following subsection the relationship between S_{id} and the probability of picking discrete values is given.

A. Probability of a discrete value 'm'

For a particular S_{id}, the probability of a discrete variable becoming assigned to a value of 'm' is discussed in this section.

For m = 0, the probability is

$$P(X_{id}=0 \mid S_{id}) = \int_{-\infty}^{0.5} g(x)\,dx \qquad (11)$$

$$= 1 - Q\left(\frac{0.5 - S_{id}}{\sigma(M-1)}\right)$$

,where, Q is the error function and the function, g, is

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2(M-1)^2}} \exp\left(\frac{-1}{2\sigma^2(M-1)^2}(x - S_{id})\right) \qquad (12)$$

For m in the range 1 to M—2, the probability is

$$P(X_{id}=m \mid S_{id}) = \int_{m-0.5}^{m+0.5} g(x)\,dx \qquad (13) \text{ or}$$

$$= Q\left(\frac{m - 0.5 - S_{id}}{\sigma(M-1)}\right) - Q\left(\frac{m + 0.5 - S_{id}}{\sigma(M-1)}\right)$$

For m = M-1, the probability is

$$P(X_{id}=(M-1) \mid S_{id}) = \int_{(M-1)-0.5}^{\infty} g(x)\,dx \qquad (14) \text{ or}$$

$$= Q\left(\frac{(M-1) - 0.5 - S_{id}}{\sigma(M-1)}\right)$$

Of course, the sum of the probability is always

$$\sum_{m=0}^{M} P(X_{id} = m / S_{id}) = 1 \qquad (15)$$

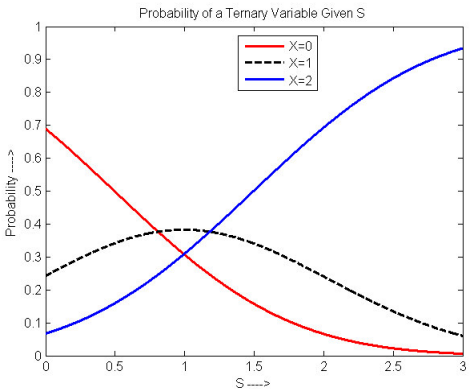

Figure 5. Probability of different discrete variables as S_{id} varies between the limits of [0 M]. This example is shown for a ternary system, M=3, and σ =0.5

Figure 5 shows the probability of various discrete variables for different S_{id} values. The figure is for a ternary system using a σ of 0.5 or standard deviation for the normal distribution of '1'. In Figure 6, the plot is shown for $\sigma = 0.1$ or equivalently standard deviation of 0.2. As σ value decreases, one gets curves with sharper peaks for the probability of discrete value given S_{id}. As $\sigma \rightarrow 0$ the algorithm will simply round the S_{id} value to determine the discrete value. If a higher σ is used, the algorithm approaches a uniform distribution. In the new algorithm, an additional parameter σ is introduced. The setting of this σ is critical to the algorithms performance. Empirical results show that a sigma of 0.2 is a good choice for ternary system.

B. Case Analysis for a Ternary System

For a ternary system, 9 cases exist for different sets of '*pbest*' and '*gbest*'. Due to limitations of space we show the preliminary analysis for one case, (P=2, G=2). Since P=2, G=2, the velocity update equation becomes

$$V^{(t+1)} = \omega \times V^{(t)} + \psi \qquad (16)$$

The distribution is triangular for both X values, X=1 and X=0. However, the triangular distribution for X=0 has twice the support range when compared to the triangular distribution for X=1.

The probability density of the velocity is plotted for different values of (ψ_1, ψ_2) in the Figure 7. The probability density of the velocity spikes near 2 and, eventually, decreases as the velocity increases. With an increase in (ψ_1, ψ_2), a longer tail appears on the distribution indicating higher velocities being selected. Similar to the binary PSO, one controls the velocities using the parameters, (ψ_1, ψ_2) in the new algorithm.

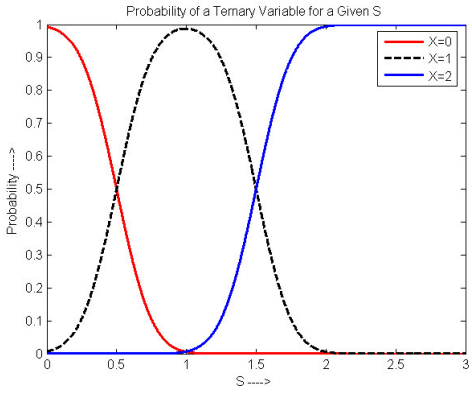

Figure 6. Affect of Selection of Omega on the Probability of different values for a discrete variable given S_{id}. This example is shown for a trenary system, M=3, $\sigma = 0.1$

The new algorithm designed for discrete multi-valued optimization problems is an extension of the binary PSO so has similar properties and reacts to

ψ_1, ψ_2. In the next section, the new algorithm is tested using different benchmark problems.

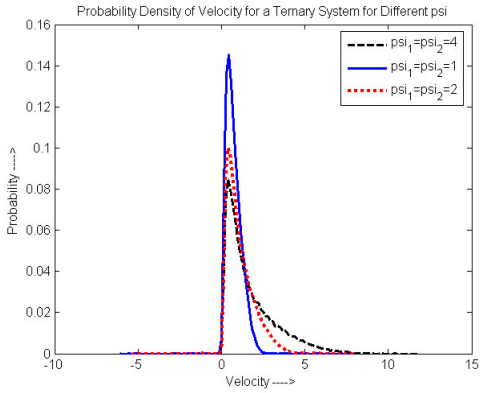

Figure 7. Probability density of velocities for the ternary system for different ψ_1, ψ_2 .

V. EXPERIMENTAL SETUP AND BENCHMARKS

Five benchmark problems are used from Liang, et al., [11] for comparing the binary discrete PSO algorithm with the multi-valued discrete PSO. For each benchmark, the algorithm executes for 25 trials with 10 dimensions of each problem. For more details about the benchmarks, the reader is referred to [11]. A brief description of the benchmark problems is given in the following subsection.

A. Benchmark Problems

The five-benchmark functions are defined in this section with information concerning their optima as in [11].

1) Shifted Rotated Ackley's Function with Global Optima on Bounds

$$f_1(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e + f\_bias$$

2) Shifted Rastrigin's Function

$$f_2(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10) + f\_bias$$

3) Shifted Rotated Rastigrin's Function

$$f_3(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10) + f\_bias$$

4) Shifted Rotated Weierstrass Function

$$f_4(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k \cos(2\pi b^k (x_i + 0.5))\right]\right) - D\sum_{k=0}^{k\max}\left[a^k \cos(2\pi b^k 0.5)\right] + f\_bias$$

5) Schwefel's Problem 2.13

$$f_5(x) = \sum_{i=1}^{D}(A_i - B_i(x))^2 + f\_bias$$

Where, $A_i = \sum_{j=1}^{D}(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j)$

and $B_i = \sum_{j=1}^{D}(a_{ij}\sin x_j + b_{ij}\cos x_j)$

For i=1,…D, A and B are two D x D matrices, a_{ij}, b_{ij} are integer random numbers generated in the range [-100, 100], $\alpha = [\alpha_1, \alpha_2 .....\alpha_D], \alpha_j$ are random numbers generated from $[-\pi, \pi]$.

B. Sampling the Search Space for Different Discrete Domains

The discrete multi-valued PSO is applied to different bases, binary, ternary and quaternary, and tested using standard fitness functions originally designed for continuous functions. Each dimension in the original function has a recommended search space range. A procedure is used to transform these functions into discrete domain. In the following table 1, we define terms used in this and following sections.

Table 1: Term Definitions for Benchmarks

Term	*Definition*
Dimension	Dimension of the original continuous benchmark problem
Base	The base of the number system (e.g. binary = base 2)
Digits	The length of the numerical character string used in the number system (e.g. binary = 2 digits, i.e., [0,1])

Each original dimension is represented using 16 bits for a binary based system or 8 quaternary digits which results in 65536 discrete values each. In a ternary based system, 10 digits would only result in 59049 steps. Thus, these representations have resulted in 65536 samples of search space for binary and quaternary bases, and 59049 samples for ternary. Higher sampling of fitness landscape provides more information about the landscape leading to better performance of the algorithm. As the samples tend toward infinity, we approach the continuous domain again. Hence, it is pertinent for comparisons that the same samples of the fitness landscape are provided to all the algorithms. Fairness is achieved by sampling the exact same points for all bases. The base that yields the smallest number of steps, which is 3 fixes the sample step size. The additional samples for the other bases are adjusted to have the value at the upper or lower range of the problem. It is assumed that the particles won't dwell significantly in these regions of the landscape.

C. Particle Swarm Settings

The benchmarks are compared using the same parameter settings. An equal weight of $\psi_1 = 1$ and

$\psi_2 = 1$ is used. A time varying inertia, ω, is used as described in [14, 15]. For an iteration 'i', the value is given by

$$\omega(i) = \frac{(\omega - 0.4) \times (no.\ of\ Iterations - i)}{no.\ of\ Iterations - 0.4} \qquad (17)$$

, where, $\omega = 0.9$. The number of particles used in the simulations is 20 and the number of iterations is equal to 5000.

VI. RESULTS

A. *Affect of σ on the Discrete PSO algorithm*

One parameter that controls the new algorithm is σ. It is shown in figures, 5 and 6, that higher σ flattens out the probabilities of the three discrete values. The resulting probabilities for σ=0.5 are given in figure 5. If the σ is further increased, the probability of each discrete value for a given S will eventually become $\simeq 0.33$ for a ternary system. This makes the curves appear nearly flat and algorithm completely random and defeats the purpose of using a swarm-based algorithm. A σ=0.4 or less is a better choice for the new algorithm. In this section, results achieved for ternary system using σ=0.4 and σ=0.2 are shown. The results are shown for Function 2 described in previous section. Similar performance results occur with the other functions described in section V.A..
Figure 8 shows that a σ=0.2 produced better results for function 2. Hence in this paper, a σ=0.2 is used for a ternary system.

B. *Results with $\sigma = 0.2$*

In this section, the results for the five-benchmark problems are presented. The new algorithm, designed for higher number systems, performed better than the binary PSO for all the benchmark problems as can be seen in figures 9, 10, 11, 12, 13. Varying sigma significantly affects the performance of the algorithm. 0.2 seems to be reasonable choice for sigma, for a ternary system. The sigma has to be varied for different number systems, i.e., ternary, quaternary and so on. A sigma of 0.1 has been used for quaternary system in this paper.
Table 2 presents the statistics for different algorithms for the 25 trials that were performed. Ternary PSO performed equivalent to the Binary PSO or even slightly better. Quaternary PSO performed better than the binary and ternary PSO. These results are significant and allow the new algorithm to be used for number systems other than binary.

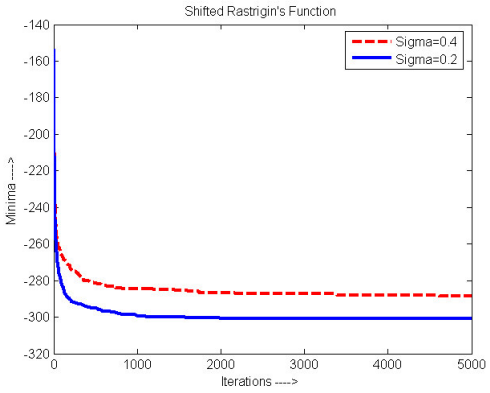

Figure 8. Comparison of minima achieved for different σ used for an algorithm designed for a ternary system for Function 2.

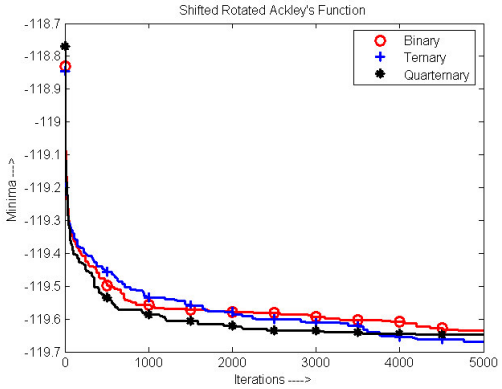

Figure 9. Minima achieved (averaged over 25 trials) for a 10 dimensional Shifted Rotated Ackley's Function

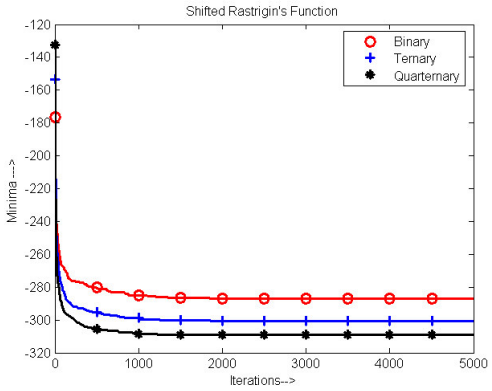

Figure 10. Minima achieved (averaged over 25 trials) for a 10 dimensional Shifted Rastrigin's Function

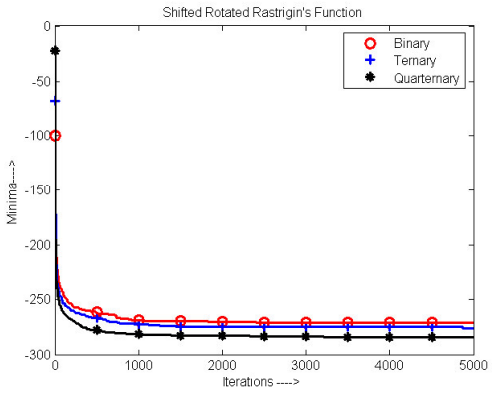

Figure 11. Minima achieved (averaged over 25 trials) for a 10 dimensional Shifted Rotated Rastrigin's Function

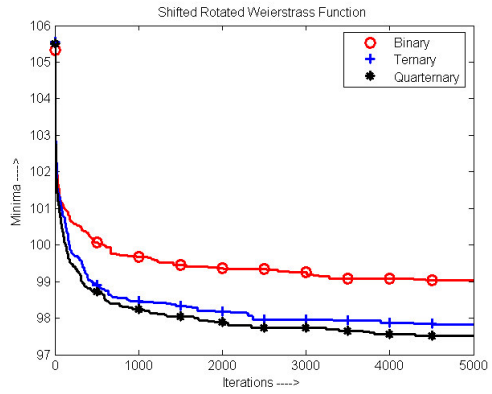

Figure 12. Minima achieved (averaged over 25 trials) for a 10 dimensional Shifted Rotated Weierstrass Function

VII. Conclusions and Future Work

In this paper we proposed a new algorithm for discrete multi valued optimization problems. A theoretical framework employing probabilistic analysis for binary PSO is presented. Specifically, probability density function for the velocity is modeled to investigate the affects of different parameters of the PSO algorithm on the velocity. The new algorithm is analyzed under this framework to examine its stability and affects of the parameters of the algorithm.

The algorithm is applied to five benchmark problems and the results presented show the performance benefits of the new algorithm. The algorithm can be successfully used for any number system.

In future work, we want to formalize the analytic framework for the discrete PSO and analyze the convergence and search behavior. Specifically, the closed form expressions for the probability density functions of the velocity will be derived. This is first attempt to analyze a binary PSO and its behavior. Further analysis for the binary PSO will also be done.

The new algorithm will be tested on deceptive functions; trap functions designed for higher ordered number systems.

Table 2: Averaged Results for Different Functions

	Binary PSO		Ternary PSO		Quaternary PSO	
f	Mean	Std	Mean	Std	Mean	Std
f1	-119.63	0.06	-119.66	0.07	-119.64	0.068
f2	-287.08	5.80	-300.72	4.11	-309.12	4.68
f3	-271.18	6.15	-276.1	8.45	-283.87	6.39
f4	99.03	0.49	97.83	0.87	97.50	0.89
f5	25596.9	6231.9	23199.19	6847.43	14986.13	5215.12

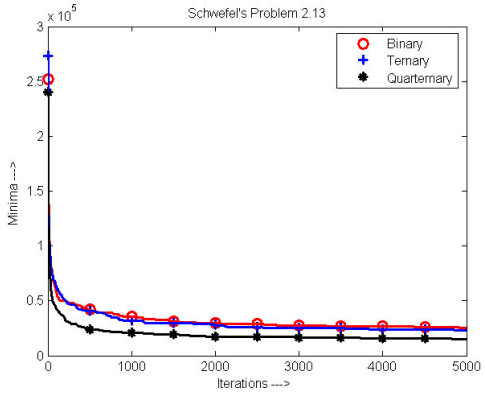

Figure 13. Minima achieved (averaged over 25 trials) for a 10 dimensional Schwefel's Problem 2.13

References

[1] Eberhart, R. and Kennedy, J., "A New Optimizer Using Particles Swarm Theory", Sixth International Symposium on Micro Machine and Human Science, 1995, Nayoga, Japan.

[2] James Kennedy, Russell Eberhart and Shi, Y.H., Swarm Intelligence, Morgan Kaufman Publishers, 2001.

[3] Evolutionary Computation 1: Basics, Algorithms and Operators, Institute of Physics Publishing, 2000.

[4] E. Ozcan and C. K. Mohan, "Particle Swarm Optimzation: Surfing the Waves", Proceedings of Congress on Evolutionary Computation (CEC'99), Washington D. C., July 1999, pp 1939-1944.

[5] Shi. Y, R. C. Eberhart, "Empirical Study of Particle Swarm Optimization", 1999 Congress on Evolutionary Computing, Vol III, pp 1945-1950.

[6] Shi Y. H., Eberhart R.C., "A Modified Particle Swarm Optimization Algorithm", IEEE International Conference on Evolutionary Computation, 1998, Anchorage, Alaska.

[7] C. K. Mohan, B. Al-Kazemi, "Discrete Particle Swarm Optimization," Proc. Workshop on Particle Swarm Optimization, Indianapolis , IN, 2001.

[8] J. Kennedy and R. C. Eberhart, "A Discrete Binary Version of Particle Swarm Optimization," Proceedings of the 1997 Conf. on Systems, Man, and Cybernetics, pp. 4104-4109. IEEE service center, Piscataway, NJ.

[9] Maurice Clerc, James Kennedy, "The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space," IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, February, 2002.

[10] B. Al-Kazemi and C. K. Mohan, "Multi-Phase Discrete Particle Swarm Optimization," Proc. The Fourth International Workshop on Frontiers in Evolutionary Algorithms, 2002.

[11] J. J. Liang, P. N. Suganthan and K. Deb, " Novel Comparison Test Functions for Numerical Global Optimization", IEEE Swarm Intelligence Symposium, pp. 68-75, June 2005.

[12] Elon S. Correa, Alex A. Freitas, Colin G. Johnson, " A New Discrete Particle Swarm Algorithm Applied to Attribute Selection in a Bioinformtics Data Set", GECCO'06, Seattle, Washington, USA, July 8-12, 2006.

[13] Jim Pugh, Alcherio Martinoli, "Discrete Multi-Valued Particle Swarm Optimization", IEEE Swarm Intelligence Symposium' 06, Indianapolis, Indiana, USA, May 12-14, 2006.

[14] Kalyan Veeramachaneni, Thanmaya Peram, Chilukuri Mohan, lisa Osadciw, "Optimization Using Particle Swarm Using Near Neighbor Interactions", GECCO'03, Chicago, Illinois, USA, July, 2003.

[15] Particle Swarm Optimization Code, Yuhui Shi, www.engr.iupui.edu/~shi.