Binary Particle Swarm Optimization with Crossover Operation for Discrete Optimization

Deepak Singh
Raipur Institute of Technology
Raipur, India

Vikas Singh
ABV- Indian Institute of Information Technology and management, Gwalior, India

Uzma Ansari
Raipur Institute of Technology
Raipur, India

ABSTRACT
The field of discrete optimization consists of the areas of linear and integer programming, cover problems, knapsack problems, graph theory, network-flow problems, and scheduling. This paper performs an Experiment for discrete Optimization problem with the Hybridization of Binary Particle Swarm Optimization (BPSO) and Genetic Crossover. There are many algorithms Present for solving discrete optimization problem. Both BPSO and GA have shown to be very effective results. Experiment performed on this paper is for the analysis and behavioral study of Hybridized algorithm. We conclude with the results obtained by the performed experiment on standard benchmark functions, and it is found that proposed algorithm gives better results for few standard benchmark functions.

General Terms
Discrete Optimization, Algorithm.

Keywords
Binary Particle Swarm Optimization; BPSO; Genetic Algorithm; GA; Hybrid Binary Particle Swarm Optimization; HBPSO; Crossover.

1. INTRODUCTION
In mathematics, computer science and economics, optimization, or mathematical programming, refers to choosing the best element from some set of available alternatives. There are many problems, which require ordering or arranging of discrete elements such as: scheduling and routing problems. As any problem discrete or continuous this can be expressed in binary notation, might be advantageous. This work presents alternative of the algorithm to operate on discrete binary optimization problems. In 1997, Kennedy and Eberhart introduced a discrete binary version of PSO for discrete optimization problems [13]. In this paper we performed an experiment on enhanced model of binary PSO with crossover operation. There are many literature found on the hybridized algorithm in which different types of crossover were found to be the better performance [4][5]. Therefore this paper shows the various result obtained with different types of crossover hybridized on BPSO.

Five different types of binary coded crossover operators are applied to Binary PSO to check whether the hybridized algorithm works better on the Benchmark function or not. In this work a crossover step is added to the standard BPSO. The crossover is performed between each particle's individual best positions. After the crossover, the fitness of the individual best position is compared with that of the two off-springs, and the best one is taken as the new individual best position. To improve the solution diversity in BPSO, a crossover BPSO is introduced. Two particles are selected as parents through tournament selection. After selecting a crossover point randomly, new particles are generated using crossover probability. Both the above techniques perform a crossover by swapping the particles around the crossover point. The positions themselves are not altered. In this paper we perform five crossover operators to BPSO [15], some crossover operators alter the position and some crossover operators swap the particles around the crossover point.

The rest of the paper is organized as follows: Basic PSO and Binary PSO algorithm is explained in second section. Crossover operator in genetic algorithm is described in third section. In this paper 5 different types of binary coded crossover operators are considered for experiments with BPSO presented in forth section. BPSO with crossover procedure is described in section five. In Section six, the benchmarks and experimental settings are described [14]. Results are presented in Section seven followed by conclusions in Section eight.

2. PARTICLE SWARM OPTIMIZATION
Particle Swarm Optimization (PSO) is an evolutionary computation technique developed by R.Eberhart and J.Kennedy in 1995 [1], inspired by swarm intelligence, such as birds flocking, fish schooling. The particle swarm concept originated from a simulation of a simplified social system. Initial simulations were modified to incorporate a nearest-neighbor velocity matching, eliminate ancillary variables, and incorporate multidimensional search and acceleration by distance. Here we give the short description of PSO and Binary PSO.

2.1 Continuous PSO
Particle Swarm Optimization (PSO) was originally designed and introduced by Eberhart and Kennedy in 1995. The PSO algorithm is an adaptive algorithm, which involves simulating the social behavior of a group of bees, birds or a school of fish. Previous versions of the particle swarm have operated in continuous space, where trajectories are defined as changes in position on some number of dimensions.

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 hence forth, are flown through the multi-dimensional search space with each particle representing a possible solution to the multi-dimensional optimization problem. Each solution 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.

For an n-dimensional search space, the ith particle of the swarm is represented by an n-dimensional vector, $X_i = (x_{i1}, x_{i2}, \ldots\ldots x_{in})$ .The velocity of this particle is represented by another n-dimensional vector $V_i = (v_{i1}, v_{i2}, \ldots\ldots v_{in})$. The previously best visited position of the ith particle is denoted as $P_i = (p_{i1}, p_{i2}, \ldots\ldots, p_{in})$. The velocity of the ith particle is updated using the velocity update equation given by

$$v_{id}(k+1) = \omega \times v_{id}(k) + c1 \times rand(\ ) \times \left(p_{id}(k) - x_{id}(k)\right) + c2 \times rand(\ ) \times \left(p_{gd}(k) - x_{id}(k)\right) \quad (1)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (2)$$

ω is inertia factor; c1, c2 is the learning factor, or accelerated variable; rand () is the random number between (0, 1); $p_{id}(k)$ is the individual(pbest) Extreme of number i; $p_{gd}(k)$ is the global (gbest)optimal solution.v_{id} (k) id stands for speed, $X_{id}(k)$ is for position of the particle.

2.2 Discrete PSO

Standard PSO is mainly applied in the area of continuous space and rarely in that of discrete space. In order to be used in discrete space, J.Kenney and R.Eberhart developed a binary Particle Swarm Optimization in 1997 to make PSO capable to optimize the combination problem The binary PSO extended particle swarm optimization and is used to optimize the discrete binary space problem, for example, knapsack problem, schedule problem, data mining, biologic information, and graphics and so on, Although it has been proposed for just ten years and has been used in many combination optimizing problems.

Kennedy and Eberhart presented a first binary version of PSO, called Binary PSO. As in classical PSO, velocities are used to determine the next position of a particle. However, as each bit may only obtain discrete values 0 and 1, velocities are used in a stochastic solution construction process. More precise, the velocity value of a bit determines the probability to set this bit to 1 in the next solution construction. The binary PSO is an extended particle swarm optimization and is used to optimize the discrete binary space problem. The parameter $v_{id}(k)$ (predisposition for each particle) is calculated similarly to the classical PSO based given in Equation (1) and it will function as a probability threshold to make one of the two decisions (0 or 1).Such a threshold needs to stay in the range of [0, 1]. The sigmoidal function shown in Equation (3), maps the interval of $v_{id}(k)$ i to a range of [0, 1].

$$Sig(x) = \frac{1}{1 + \exp(-x)} \quad (3)$$

Final binary decision making is based on the following rule:

$$x_{id}(k+1) = \begin{cases} 0 & \text{if } rand(\ ) \le Sig(v_{id}(k+1)) \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

The entire PSO algorithm of the binary version is almost the same as that of the basic continuous version in Section, except that it uses the above decision rule.

3. CROSSOVER OPERATOR

Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. In genetic algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Crossover is genetic operator used in creation of one or more offspring's from the selected parents. Numbers of elements take part in crossover are called parents and crossover results are called offspring's or children. The number of new offspring's generated from crossover operation is equal to the number of members in parent population. The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover probability is a term associated with crossover operation which decides whether crossover happens or not. The crossover operator combines parts of good solution to form new potential solution. Information contained in one solution combine with information contained in another solution and the resulting solution will either have good fitness or survive to exchange this information again. Crossover operators exist for both real coded and binary coded GA. Since this paper explores the application of crossover operators to BPSO for discrete optimization problem therefore for this paper we discuss only real coded crossover operators. A number of real coded crossovers have been introduced for GA and other heuristic technique. Five different binary coded crossovers we used listed in next section.

3.1 One Point Crossover

When performing crossover, both parental chromosomes are split at a randomly determined crossover point. Subsequently, a new child genotype is created by appending the first part of the first parent with the second part of the second parent.

Consider the following 2 parents which have been selected for crossover. The "|" symbol indicates the randomly chosen crossover point.

Parent 1: 10001|110

Parent 2: 00100|011

After interchanging the parent chromosomes at the crossover point, the following offspring are produced:

Offspring1: 10001|011

Offspring2: 00100|110

3.2 Two Point Crossover

In two-point crossover, both parental genotypes are split at two points, constructing a new offspring by using parts number one and three from the first, and the middle part from the second ancestor. Using two point crossovers will enable searching problem space more thoroughly. Using single-point and two-point crossover operator prevents schema to be disrupted, but when population becomes homogeneous, search space becomes smaller.

Consider the following 2 parents which have been selected for crossover. The "|" symbols indicate the randomly chosen crossover points.

Parent 1: 101|010|01

Parent 2: 001|001|11

After interchanging the parent chromosomes between the crossover points, the following offspring are produced:

Offspring1: 101|001|01

Offspring2: 001|010|11

3.3 Uniform Crossover

A crossover operator that decides (with some probability – known as the mingling ratio) which parent will donate each of the gene values in the offspring chromosomes. This allows the parent chromosomes to be mixed at the gene level rather than the segment level (as with one and two point crossover). Uniform crossover disrupts schema with great probability but searches larger problem space. For uniform crossover, the number of effective crossing points is not fixed, but will average to l/2 where l represents string length.

Consider the following 2 parents which have been selected for crossover:

Parent 1: 11001010

Parent 2: 00100111

If the mixing ratio is 0.5, approximately half of the genes in the offspring will come from parent 1 and the other half will come from parent 2. Below is a possible set of offspring after uniform crossover:

Offspring1: $1_1 0_2 1_2 0_1 0_2 0_1 1_1 1_2$

Offspring2: $0_2 1_1 0_1 0_2 1_1 1_2 1_2 0_1$

Note: The subscripts indicate which parent the gene came from

3.4 Half-Uniform Crossover

Half-Uniform crossover is similar to uniform crossover. Only difference is that only half of differing bits between parents will be swapped. In the half uniform crossover scheme (HUX), exactly half of the nonmatching bits are swapped. Thus first the Hamming distance (the number of differing bits) is calculated. This number is divided by two. The resulting number is how many of the bits that do not match between the two parents will be swapped.

3.5 Shuffle Crossover

Shuffle crossover is similar to one-point crossover. First, a single crossover position is selected. Before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled in reverse. This removes positional bias as the variables are randomly reassigned each time crossover is performed. In a way, shuffle crossover is similar to uniform crossover. Difference is that uniform crossover exchanges bits and not segments like shuffle crossover. Further, in uniform crossover bits exchanged follow a binary distribution and in shuffle crossover bits follow uniform distribution, as in single-point crossover.

4. HYBRID BINARY PARTICLE SWARM OPTIMIZATION WITH GA CROSSOVER

It was pointed out that PSO usually suffers from premature convergence, tending to get stuck in local optima, low solution precision and so on. In order to overcome these shortcomings and get better results, numerous improvements to PSO have been proposed. One of the novel versions of PSO with crossover operator is proposed by adding a crossover step to the standard PSO. PSO uses iterative process to search the global optima in solution space. Crossover operator with PSO has a property of better exploration so by using crossover search area is explored in a relatively better manner. PSO has a higher convergence rate, by using crossover with PSO premature convergence is also reduced and PSO does not get trapped in local optima. Crossover can help the particles to jump out of the local optima by sharing the others' information. To improve the solution diversity in PSO, a crossover BPSO is introduced.

In this paper BPSO with five different types of crossover applied to the five benchmark functions for testing which crossover gives the optimum result at what probability. Particles generated by BPSO are randomly selected for crossover operation and two new offspring's are formed. The best offspring (in terms of fitness) selected from the new offspring's. This new best offspring replaces the worst parent particle which is selected for crossover. The replacement is done if the new best offspring has the good fitness value than the parent particle. In this paper we analyze different types of crossover operators used with BPSO, all the crossover discussed in previous section applied to the BPSO one by one according to the algorithm shown in below algorithm.

Algorithm of proposed method is as follows:

for i = 0 to the maximum bound of the number of function evaluation **do**

for s = 0 to the swarm size **do**

for d = 0 to the problem dimension **do**

Update velocity by BPSO method

Update position by BPSO method

end for d

Compute fitness of updated position

If needed, update historical information for

Pi and Pg

end for s

for all current generated swarm

if Crossover criteria met **then**

Select two random particles as parent particles from the current swarm for crossover operation.

Apply crossover operation.

New offsprings generated from parents as a result of crossover. Replace the parent particle with the new offspring.

If Pg meets problem requirement **then**

Terminate

end if

end if

end for i

Figure 1

5. PARAMETER SETUP for BPSO, HYBRID BPSO and BENCHMARK FUNCTIONS

From the standard set of Benchmark problems available in the literature, five important functions are chosen to test and compare the performance of both BPSO and Hybrid PSO. All the benchmark problems chosen have discrete variables and have different degree of complexity. In our experiment the problem size for all the problems is set to 30. The problems are listed in below.

5.1 Benchmark Functions

5.1.1 *Goldberg's order-3*

The fitness f of a bit-string is the sum of the result of separately applying the following function to consecutive groups of three components each:

$$f1(y) = \begin{cases} 0.9 \text{ if } |y| = 0 \\ 0.6 \text{ if } |y| = 1 \\ 0.3 \text{ if } |y| = 2 \\ 1.0 \text{ if } |y| = 3 \end{cases} \quad (5)$$

5.1.2 *Bipolar Order 3*

The fitness is the sum of the result of applying the following function to consecutive groups of six components each:

$$f1(y) = \begin{cases} 1.0 \text{ if } |y| = 0 \text{ or } 6 \\ 0.0 \text{ if } |y| = 1 \text{ or } 5 \\ 0.4 \text{ if } |y| = 2 \text{ or } 4 \\ 0.8 \text{ if } |y| = 3 \end{cases} \quad (6)$$

5.1.3 *Muhlenbein's order-5*

The fitness is the sum of the results of applying the following function to consecutive groups of five components each:

$$f1(y) = \begin{cases} 4.0 \text{ if } y = 00000 \\ 3.0 \text{ if } y = 00001 \\ 2.0 \text{ if } y = 00011 \\ 1.0 \text{ if } y = 00111 \\ 3.5 \text{ if } y = 11111 \\ 0.0 \text{ otherwise} \end{cases} \quad (7)$$

5.1.4 *Clerc's Zebra3*

The fitness f of a bit-string is the sum of the result of separately applying the following function to consecutive groups of three components each. If the rank of the group is even (first rank=0)

$$f1(y) = \begin{cases} 0.9 \text{ if } |y| = 0 \\ 0.6 \text{ if } |y| = 1 \\ 0.3 \text{ if } |y| = 2 \\ 1.0 \text{ if } |y| = 2 \end{cases} \quad (8)$$

5.1.5 *Whitney DF2*

The fitness is the sum of the results of applying the following function to consecutive groups of sixteen components each:

$$f1(y) = \begin{cases} 0.0 \; if \; y = 15 \\ 2.0 \; if \; y = 0 \\ 4.0 \; if \; y = 1 \\ 6.0 \; if \; y = 2 \\ 8.0 \; if \; y = 4 \\ 10.0 \; if \; y = 8 \\ 12.0 \; if \; y = 3 \\ 14.0 \; if \; y = 5 \\ 16.0 \; if \; y = 6 \\ 18.0 \; if \; y = 9 \\ 20.0 \; if \; y = 10 \\ 22.0 \; if \; y = 12 \\ 24.0 \; if \; y = 14 \\ 26.0 \; if \; y = 13 \\ 28.0 \; if \; y = 11 \\ 30.0 \; if \; y = 9 \end{cases} \quad (9)$$

5.2 Parameter Setup for Experiment

In this paper the parameter setting are as swarm size = 40. The inertia weight w is set to 0.73. c1 and c2 both are set to 2. Global variants of BPSO are considered. X_{max} and X_{min} are the upper and lower bounds of the decision variables. Whenever the calculated position of particle exceeds the X_{max} or lowers than the X_{min}, particle position is set to X_{max} or X_{min} respectively. This paper presents an experiment on BPSO and (Hybrid) BPSO with Crossover operator. For Crossover, all five crossover which is explained previously in chosen. For experiments, crossovers probabilities are taken in the range of 0.1 to 0.9. Two criteria are applied to terminate the simulation of the algorithms. The first is the maximum number of function evaluations, set as 10000 and the second is minimum error which is set to be 0.9.

Table 1 BPSO with one point crossover

Benchmark Functions	Fitness	without	pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr< .7	pr< .8	pr <.9
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Goldberg's	Mean Fit	0.049673	**0.049393**	**0.049533**	**0.04944**	**0.0493**	**0.049393**	**0.049393**	**0.049393**	**0.04944**	**0.0493**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Bipolar	Mean Fit	0.055967	0.07268	0.102633	0.0893	0.102633	0.142633	0.149393	0.159347	0.175967	0.1693
	Best Fit	1.3493	0.8493	0.8493	1.3493	1.3493	0.8493	0.8493	0.8493	1.3493	1.3493
Muhlenbein's	Mean Fit	2.315967	**2.1493**	**2.2493**	**2.2493**	**2.182633**	**2.1493**	**2.06596**	**2.115967**	**2.165967**	2.032633
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Clerc's order 3 (Zebra3)	Mean Fit	0.172633	0.182633	**0.155967**	**0.1493**	**0.1493**	**0.162633**	**0.152633**	**0.145967**	**0.152633**	**0.163947**
	Best Fit	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507
Whitney DF2	Mean Fit	2.7662	**2.206387**	**2.73972**	2.863	**2.673**	**2.673053**	3.119627	4.22282	4.263007	3.586293

Table 2 BPSO with two point crossover

Benchmark Functions	Fitness	without	pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr< .7	pr< .8	pr <.9
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Goldberg's	Mean Fit	0.049673	**0.0493**	0.056247	**0.04944**	**0.049393**	**0.049347**	**0.04944**	**0.04944**	**0.049393**	**0.049347**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Bipolar	Mean Fit	0.055967	0.0893	0.086013	0.102633	0.126013	0.1693	0.1293	0.155967	0.15268	0.1093
	Best Fit	1.3493	0.3493	0.3493	0.3493	1.3493	1.3493	0.8493	0.8493	1.3493	0.8493
Muhlenbein's	Mean Fit	2.315967	2.482633	**2.065967**	**1.982633**	**2.132633**	**2.182633**	**2.215967**	**1.969567**	**1.965967**	**1.915967**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Clerc's order 3 (Zebra3)	Mean Fit	0.172633	**0.165967**	**0.165967**	**0.155967**	0.182633	**0.1593**	**0.152633**	0.175967	**0.1593**	**0.1593**
	Best Fit	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507
Whitney DF2	Mean Fit	2.7662	**1.949767**	**2.329673**	**1.979907**	**2.293147**	2.53972	3.65296	3.693147	4.149767	5.232867

Table 3 BPSO with uniform crossover

Benchmark Function	Fitness	without	pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr< .7	pr< .8	pr <.9
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Goldberg's	Mean Fit	0.049673	**0.04944**	**0.04944**	0.052773	0.056153	**0.04944**	**0.049347**	0.049533	0.049627	**0.04944**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Bipolar	Mean Fit	0.055967	0.075967	0.0893	0.13268	0.102633	0.0893	0.122633	0.155967	0.15268	0.142633
	Best Fit	1.3493	1.3493	1.8493	0.8493	0.3493	1.3493	1.3493	1.3493	0.8493	0.3493
Muhlenbein's	Mean Fit	2.315967	**2.2993**	2.415967	**2.1993**	**2.165967**	**2.1993**	**2.1493**	**2.215967**	**2.282633**	**2.132633**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Clerc's order 3 (Zebra3)	Mean Fit	0.172633	**0.162633**	**0.155967**	0.182633	**0.1693**	**0.1593**	**0.155967**	**0.1693**	**0.1493**	**0.155967**
	Best Fit	0.1507	0.1507	0.1507	0.1507	0.1507	**0.1507**	0.1507	0.1507	0.1507	0.1507
Whitney DF2	Mean Fit	2.7662	**1.82648**	**2.493147**	**2.406387**	2.786293	2.956153	**2.586293**	3.33972	3.33972	4.146107

Table 4 BPSO with shuffle crossover

Benchmark Functions	Fitness	without	pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr< .7	pr< .8	pr <.9
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Goldberg's	Mean Fit	0.049673	**0.049487**	**0.049347**	**0.049487**	**0.49487**	**0.49487**	**0.04944**	**0.04944**	0.052867	**0.049487**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Bipolar	Mean Fit	0.055967	0.082727	0.0693	0.1493	0.122633	0.1693	0.1893	0.2093	0.2293	0.222633
	Best Fit	1.3493	0.3493	1.3493	0.8493	1.3493	0.8493	1.3493	1.3493	1.3493	0.8493
Muhlenbein's	Mean Fit	2.315967	**2.082633**	**2.282633**	**1.982633**	**2.265967**	**2.082633**	**2.232633**	**2.0493**	**2.315967**	**2.132633**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Clerc's order 3 (Zebra3)	Mean Fit	0.172633	**0.172633**	0.19268	0.195967	0.182633	0.1793	0.206013	0.202633	0.1893	**0.162633**
	Best Fit	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507
Whitney DF2	Mean Fit	2.7662	**2.663**	3.612773	5.402727	5.146107	5.956153	6.469393	7.73606	8.52613	8.27944

Table 5 BPSO with half uniform crossover

Benchmark Functions	Fitness	without	pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr< .7	pr< .8	pr <.9
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Goldberg's	Mean Fit	0.049673	**0.049487**	**0.04944**	**0.04944**	**0.049393**	**0.0493**	**0.04944**	**0.049347**	0.049347	**0.0493**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Bipolar	Mean Fit	0.055967	0.0893	0.0693	0.075967	0.1093	0.122633	0.162633	0.135967	0.195967	0.1693
	Best Fit	1.3493	0.8493	1.3493	0.8493	1.3493	0.8493	0.8493	1.3493	0.8493	0.8493
Muhlenbein's	Mean Fit	2.315967	**2.282633**	**2.132633**	**2.015967**	**2.0993**	**2.0493**	**2.065967**	**2.032633**	**1.865967**	**1.965967**
	Best Fit	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493	0.0493
Clerc's order 3 (Zebra3)	Mean Fit	0.172633	0.199347	**0.162633**	0.1793	**0.1593**	**0.172633**	**0.172633**	**0.1393**	**0.1593**	**0.155967**
	Best Fit	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507	0.1507
Whitney DF2	Mean Fit	2.7662	**2.616433**	**1.563473**	**2.436527**	**2.05662**	**1.97352**	2.836527	3	3.246573	3.179907

6. EXPERIMENTAL RESULT

This section focuses on the efficiency of BPSO and Hybrid BPSO with five different crossover operators tested on 5 benchmark functions with 30 dimensions, given in Table 1,2,3,4,5. To avoid the choice of a particular initial population and to conduct fair comparisons between crossovers, 30 runs are considered starting from various randomly selected points in the multidimensional search space. The BPSO and Hybrid BPSO with crossovers are implemented in C++. Recorded simulated results are presented in Tables 1, 2, 3, 4, 5. for each benchmark function. For each crossover mean fitness objective function value (Mean OBJ) and the best fitness objective function evaluations (Average evaluations) of 30 runs were calculated and compared. In the tables the bold readings are improved result over the BPSO. Obtained results indicated in tables that BPSO with crossover and suitable crossover probability may provide better results than original BPSO. However there is no general value of crossover probability for which any crossover can improve the results obtained by hybrid BPSO.

7. CONCLUSION

In this paper, Hybrid version of BPSO is compared with BPSO. Hybrid BPSO is obtained by an additional step added to BPSO that is a crossover technique of GA. Code of BPSO and Hybrid BPSO (fig. 1) is developed in c++ and results are shown in Table 1, 2, 3, 4, 5. The experiments are performed on a set of 5 benchmark problems available in the literature. The crossover probability range is set to 0.1 - 0.9. For each benchmark function results are tabulated for crossover probability 0 to 0.9 in terms of mean objective function value (Mean OBJ) and best function evaluation (Best fitness Evaluations) which describes the efficiency and accuracy are depicted in bold. From the result obtained it is concluded that Hybrid BPSO has been found to have successful performance on Goldberg's order-3, Muhlenbein's order-5, Clerc's Zebra 3 and Whitney DF2 benchmark problems.

8. REFERENCES

[1] J. Kennedy and R. Eberhart, Particle swarm optimization, in Proc. IEEE International Conference Neural Networks, vol. 4, 1995, pp. 1942 - 1948..

[2] Y. Shi and R. C. Eberhart, A modified particle swarm optimizer, in Proc. IEEE International Conference on Evolutionary Computation, Piscataway,NJ, 1998, IEEE Press, pp. 69-73.

[3] R. C. Eberhart and Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, 2000 Congress on Evolutionary Computing, vol. 1, 2000, pp. 84-88.

[4] Zhi-Feng Hao, Zhi-Gang Wang and Han Huang, A Particle swarm optimization algorithm with crossover operator, in Proc. of the Sixth International Conference on Machine Learning and Cybernetics, HongKong, 19-22 August 2007.Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[5] Dongyong Yang, Jinyin Chen and Matsumoto Naofumi, Self-adaptive Crossover Particle Swarm Optimizer for Multi-dimension Functions Optimization,ICNC 2007.

[6] J.H. Holland, Adaptation in Natural and Artificial System, The University of Michigan Press, Ann Arbor,1975.

[7] Goldberg D E, Genetic Algorithms in search, optimization, and machine learning. Addison-Wesley Publishing Corporation, Inc, 1989.

[8] Jianhua Liu and Xiaoping Fan, The Analysis and Improvement of Binary Particle Swarm Optimization, International Conference on Computational Intelligence and Security2009.

[9] Xu Jun and Huiyou Chang, The Discrete Binary Version Of The Improved Particle Swarm Optimization Algorithm, IEEE 2009.

[10] Javad Sadri and Ching Y. Suen, A Genetic Binary Particle Swarm Optimization Model, IEEE Congress on Evolutionary Computation2006.

[11] Zhang Li-ping, YU Huan-jun and HU Shang-xu, "Optimal choice of parameters for particle swarm optimization", Journal of Zhejiang University SCIENCE 2005, vol. 6A(6), pp. 528-534.

[12] Jaco F. Schutte and Albert A. Groenwold,"A Study of Global Optimization Using Particle Swarms", Journal of Global Optimization (2005) vol. 31, pp. 93-108.

[13] J. Kennedy and R. Eberhart, A discrete binary version of the particle swarm optimization A. Proceeding of the conference on System, Man, and Cybernetics [C], NJ, USA: IEEE Service Center, 1997, 4 104 - 4 109.

[14] M. Clerc, "Binary Particle Swarm Optimisers: Toolbox, Derivations, and Mathematical Insights," 2005. [Online]. Available: http://clerc.maurice.free.fr/pso/.

[15] Eiben, A. E. et al (1994). "Genetic algorithms with multi-parent recombination". PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: 78–87. ISBN 3-540-58484-6.