XOR Binary Gravitational Search Algorithm

Mojtaba Ahmadieh Khanesar, David Branson III

Abstract— **In this paper, an XOR binary gravitational search algorithm is introduced. Gravitational search algorithms, a physics inspired optimization algorithm, have previously been successfully applied to different real-valued optimization problems. In their binary version, the definition of a velocity vector changes to probability of change in corresponding dimensions. However, analysis shows that existing velocity vector update equation for binary gravitational search algorithm do not direct the particle towards better particle in certain cases. To apply this algorithm to binary optimization problems we introduce an XOR operator in the acceleration term. After a mathematical comparison to existing binary gravitational search algorithms it is shown that the proposed modification complies more with the definition of change in each dimension. Extensive simulations are performed showing the superiority of the proposed algorithm over other existing algorithms such as binary particle swarm optimization and an existing version of binary gravitational search algorithm.**

I. Introduction

Calculus of variations, gradient based optimization methods and linear programming approaches are among the most successful computational methods to optimize a nonlinear function. However, there are some cases in which the cost function is not explicitly defined and/or it is difficult or even impossible to calculate derivatives due to high dimensionalities. In such cases, nature inspired optimization algorithms that rely on implicit functions may be viable choices. These optimization algorithms start the process from multiple points and they are less probable to fall in local minima. Evolutionary based optimization algorithms [1], swarm based optimization methods [2] and physic inspired optimization algorithms [3] are different categories of meta-heuristic optimization methods applied in these cases.

In evolutionary based optimization methods, each solution is decoded to a chromosome with its dimension being the same as the number of genes in chromosome. Inspired by nature, these artificial chromosomes are subject to crossover, mutation and natural selection models to find the optimum value of a function. In swarm based optimization algorithms that try to model the graceful and unpredictable movement of birds and fish, a position vector represents every solution. Position vectors are then updated using a velocity vector that has different terms showing tendency towards the best experience of the whole swarm and the best individual experience of each member of the swarm. Physics inspired optimization algorithms, unlike evolutionary optimization algorithms and swarm based optimization methods, which try to model the behavior of living species, imitate the interaction between objects in real life. Examples of physics inspired optimization algorithms are central force optimization algorithm [4] and gravitational search algorithm (GSA) [5]. The latter is a stochastic search algorithm benefiting from superior exploration capability that may results in superior performance.

GSA, an optimization algorithm inspired by the interaction between masses in space from Newton gravitational law, has been successfully applied to different engineering fields such as optimal design of power systems [6], allocation of power devices [7] etc. According to Newton's gravitational law, the gravitational force between two masses in space is directly proportional to the multiplication of two mass and inversely proportional with the Euclidean distance between them. By studying the acceleration of two different masses, one can show that the acceleration of each mass towards another one is proportional to the mass of the other object. In this case, the small object moves with higher acceleration towards the big one. This acceleration causes lighter objects to accelerate towards heavier ones. In GSA, heavier masses are assigned to particles with superior cost functions values that causes the particles with poorer cost functions to be pushed towards them while scanning the whole space. Meanwhile, if any other solutions come up with better performance, a big value of mass is assigned to that particle making it attract other particles.

The original GSA is designed for real-valued optimization problems with position and velocity vector being real-valued. Continuously evolving product design emerges automated task planning, task assignment and programing individual tasks for production elements including robots that may give rise to using intelligent optimization algorithms. In such examples, decision variables may be binary rather than real values and the goal may be to achieve less energy consumption, minimum time and possibly lower cost. Examples of such applications are process-planning problem [8], assembly sequence planning [9] and assembly line design problems [10]. To solve discrete binary optimization problems, the values of decision variables in a binary optimization problem are either *zero* or *one* and a vector representing the probability of change in each dimension is defined instead of velocity vector [11, 12]. The main aim of the vector representing the probability of change is to direct a particle towards a particle with better cost function. However, the analysis presented in this paper show that this does not happen using current update formula for BGSA.

*Research supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant number: EP/R021031/1 - New Industrial Systems: Chatty Factories.

M. Ahmadieh Khanesar is with Advanced Manufacturing Technology Research Group, Faculty of Engineering, University of Nottingham, United Kingdom, NG8 1BB (corresponding author to provide e-mail: ezzma5@exmail.nottingham.ac.uk).

David Branson is with Advanced Manufacturing Technology Research Group, Faculty of Engineering, University of Nottingham, United Kingdom, NG8 1BB (e-mail: ezzdtb@exmail.nottingham.ac.uk).

In this paper, BGSA and its update formulas for acceleration, velocity and position are fully analyzed and the motivation behind writing this paper is established utilizing mathematical models. It is shown that in the existing BGSA, the update equation for the probability vector of BGSA does not direct particles towards better particles as it is expected from the algorithm. To solve the problems with BGSA, the velocity vector is modified using an XOR operator. The proposed optimization algorithm is validated in simulation against a number of benchmark optimization problems. It is shown that the proposed algorithm is capable of optimizing various benchmark optimization problems including unimodal and multi-modal cost functions with several local optimum points, resulting in superior performance when it is compared to BGSA [12], novel binary particle swarm optimization (NBPSO) [13], improved binary swarm optimization (IBPSO) [14] and binary particle swarm optimization (BPSO) [11].

This paper is organized as follows. In Section II, real-valued GSA is discussed. The current version of BGSA is presented and analyzed in Section III. In Section IV, the proposed XOR BGSA is introduced and analyzed. The proposed algorithm is then compared with several existing optimization algorithms for the optimization of several benchmark examples. Finally, the concluding marks are presented in Section V.

II. Gravitational Search Algorithm

GSA is a population based optimization method which imitates the behaviour of interaction between masses due to universal gravitational forces between them. Unlike the basic version of particle swarm optimization (PSO) in which each particle is represented by its position and velocity, particles used in GSA benefit from more features such as acceleration and mass [5]. As particles with better cost functions are assigned higher values of mass they attract other particles. The particles with worse cost functions are absorbed by better particles which results in exploiting the search space. The mass of particles is continuously updated based on their cost function, if during exploitation, the cost function corresponding to a particle becomes better than the previously found better solution, a larger value is assigned to its mass which results in other particles to be accelerated towards newly updated better solution. After a few iterations, all solutions are converged towards the heaviest mass resulting in termination of the algorithm. The algorithm is described mathematically using the following few lines.

The solution to problem is defined as the positions of particles in d −dimensional search space as:

$$X_i = (x_i^1, x_i^2, \ldots, x_i^j, \ldots x_i^d) \;\; i = 1, \ldots, N \qquad (1)$$

where x_i^j is the j^{th} component of the position of the particle and N is the total number of populations. The mass of particles are updated and normalized at t^{th} iteration as [5]:

$$M_i(t) = \frac{m_i(t)}{\sum_{k=1}^{N} m_k(t)} \qquad (2)$$

where $m_i(t)$ is non-normalized mass value corresponding to i^{th} particle at iteration number t which represents the quality of solution and is defined as [5]:

$$m_i(t) = \frac{f(X_i) - f_{worst}(t)}{f_{best}(t) - f_{worst}(t)} \qquad (3)$$

where $f_{worst}(t)$ is the fitness function value corresponding to the worst particle and $f_{best}(t)$ represents the best fitness function observed by any of the particles. The parameters $f_{worst}(t)$ and $f_{best}(t)$ are updated at every iteration as:

$$f_{worst}(t) = max\{f(X_i)\}_{i=1,\ldots,N}$$
$$f_{best}(t) = min\{f(X_i)\}_{i=1,\ldots,N} \qquad (4)$$

It is to be noted that in this case the best and worst particles are defined based on minimization problem. Furthermore, k_b best solutions are selected at each iteration as the masses which absorb the other masses. Hence, in this case $m_i(t)$ belongs to the interval of $[0,\ 1]$ which $m_i(t) = 0$ being the mass of the worst particle which does not absorb any particle and $m_i(t) = 1$ represents the mass of the best particle.

The overall gravitational force acting on the i^{th} particle is obtained as:

$$F_i(t) = \sum_{j \in \{1,\ldots,k_b\}} r_j G(t) \frac{M_j(t) M_i(t) (X_j(t) - X_i(t))}{\|X_i(t) - X_j(t)\|^{r_p} + \varepsilon} \qquad (5)$$

where k_b represents the number of best solution selected, $\|.\|$ stands for Euclidean norm, ε is a small value added to prevent division by $zero$, r_p is the power considered for the Euclidean distance between two particles, $G(t)$ is the gravitational constant and $r_j \in [0,\ 1]$ is a uniform random value. The gravitational constant is updated at each iteration using the following equation.

$$G(t) = G_0 exp\left(-\beta \frac{t}{t_{max}}\right) \qquad (6)$$

where G_0 has a constant real value and t_{max} is the maximum value of the iterations of the algorithm. According to Newton's second law of motion the acceleration of particles are calculated by dividing the applied force to i^{th} mass by its value of mass as:

$$A_i(t) = \frac{F_i(t)}{M_i(t)} = \sum_{j \in \{1,\ldots,k_b\}} r_j G(t) \frac{M_j(t) (X_j(t) - X_i(t))}{\|X_i(t) - X_j(t)\|^{r_p} + \varepsilon} \qquad (7)$$

where $A_i(t) \in R^d$ d −dimensional acceleration of the particles. Considering the fact that velocity of particle can be calculated as the discrete integral of its acceleration, the following equation is obtained for the velocity of the particles.

$$V_i(t+1) = p_i \;\; V_i(t) + A_i(t) \qquad (8)$$

where $V_i(t) \in R^d$ d −dimensional acceleration of the particles at t^{th} iteration and $p_i \in [0,\ 1]$ is a uniform random number. Finally, the newer position of each particle is updated as:

$$X_i(t+1) = X_i(t) + V_i(t+1) \qquad (9)$$

III. Binary Gravitational Search Algorithm

A. Description of the algorithm

Similar to BPSO [10] in BGSA the velocity in each direction changes its meaning to the probability of change in each dimension. In other words, a large absolute value for velocity in a dimension means that the current state is not

suitable and needs to be changed to result in a better solution. Based on such a description, the function $P(V_i^j)$ is defined to implement an approach as [12]:

$$P(V_i^j) = \tanh(|V_i^j|),\ i = 1, \dots, N,\ j = 1, \dots, d \quad (10)$$

where $tanh(.)$ represents hyperbolic tangent function and is defined as:

$$tanh(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (11)$$

Calculating the $P(V_i^j)$, the probability of change in every dimension, the next state in each dimension is calculated as [12]:

$$x_i^j(t+1) = \begin{cases} \overline{x}_i^j(t) & if\ \ r_i^j < P(V_i^j) \\ x_i^j(t) & Otherwise \end{cases} \quad (12)$$

where $\overline{x}_i^j(t)$ represents the complement of $x_i^j(t)$ and $r_i^j \in [0, 1]$ has a uniform random value.

B. Analysis of Binary Gravitational Search Algorithm

As mentioned earlier, to be able to apply GSA to binary problems, the meaning of velocity changes based on the probability of change in each dimension while the other equations remain the same. Hence, the acceleration in each dimension is calculated using (7). The analysis of algorithm is done based on this equation.

Since the parameter $X_k(t),\ k = 1, \dots, N$ only accept binary values of $zero$ and one, the subtraction $X_j(t) - X_i(t)$ in each dimension may have one of four possible conditions.

1,2) If $X_i^k(t) = X_j^k(t) = 0$ or $X_i^k(t) = X_j^k(t) = 1$, in these cases $X_i^k(t) - X_j^k(t) = 0$ which results in no acceleration for the dimension k which is quite desirable.

3) If $X_j^k(t) = 1,\ X_i^k(t) = 0,\ X_j^k(t) - X_i^k(t) = 1$ which results in positive acceleration term. In this case if the velocity in k^{th} dimension is positive, this acceleration term works fine and adds up to the speed. However, if the velocity in k^{th} dimension is negative, the acceleration term decreases the absolute value of velocity and makes the probability of change smaller which is completely undesirable.

4) If $X_j^k(t) = 0,\ X_i^k(t) = 1,\ X_j^k(t) - X_i^k(t) = -1$ which results in negative acceleration term. In this case if the velocity in k^{th} dimension is positive, this acceleration term works decreases the velocity and consequently the probability of change. This case is undesirable. On the other hand, if the velocity in k^{th} dimension is negative, the acceleration term increases the absolute value of velocity and makes the probability of change larger which is desirable.

Motivated by the problems caused in the third and forth case, in this paper, BGSA is modified to have an acceleration term with XOR operator. It will be shown that using such approach, it is possible to avoid undesirable cases which are reviewed previously.

IV. Proposed XOR Binary Gravitational Search Algorithm

To overcome the aforementioned problems in the analysis part of binary GSA, the algorithm is modified to benefit from an XOR operator in acceleration vector. The acceleration term is modified as:

$$A_i(t) = \frac{F_i(t)}{M_i(t)} = \sum_{j \in \{1,\dots,k_b\}} r_j G(t) \frac{M_j(t)(X_j(t) \oplus X_i(t))}{\|X_i(t) - X_j(t)\|^{rp} + \varepsilon} \quad (13)$$

where $\oplus$ represents the XOR operator acting bitwise on $X_j(t)$ and $X_i(t)$. The truth table for XOR operator is defined as in Table I.

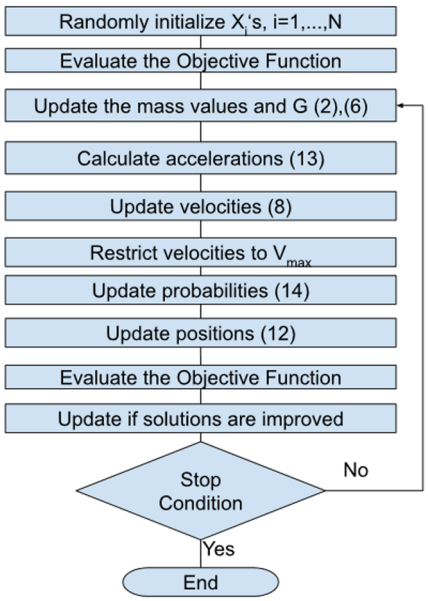

Figure 1. Flowchart of proposed XOR binary gravitational search algorithm

TABLE I. Truth table for XOR operator

A	B	A ⊕ B
0	0	-1
0	1	1
1	0	1
1	1	-1

Furthermore, the function $P(V_i^j)$ is modified as follows:

$$P(V_i^j) = 0.5 + 0.5\tanh(0.5V_i^j),\ i = 1, \dots, N,\ j = 1, \dots, d \quad (14)$$

where $tanh(.)$ is defined as in (11). The remaining parts of the algorithm remain unchanged and the velocity vector as well as position vector in the XOR BGSA are updated as (8) and (12), respectively. The flowchart of the proposed binary optimization method is depicted in Fig. 1.

A. Analysis of proposed binary optimization algorithm

In this section, the proposed BGSA is analysed to show that the previous problems mentioned in the analysis part of BGSA do not exist any more. Similar to BGSA, in the case of

proposed algorithm, since the parameter $X_k(t),\ k = 1,\ldots,N$ only accept binary values of $zero$ and one, the subtraction $X_j(t) - X_i(t)$ in each dimension may have one of four possible conditions.

1,2) If $X_i^k(t) = X_j^k(t) = 0$ or $X_i^k(t) = X_j^k(t) = 1$, in these cases $X_i^k(t) \oplus X_j^k(t) = -1$. According to (13), in these cases negative acceleration occurs for the dimension k which is quite desirable as it decreases the probability of change.

3) In the case $X_j^k(t) = 1,\ X_i^k(t) = 0, X_j^k(t) \oplus X_i^k(t) = 1$ the desired condition is that a positive acceleration which increases the value of velocity vector and consequently results in higher probability of change.

4) In the case when $X_j^k(t) = 0,\ X_i^k(t) = 1,\ X_j^k(t) \oplus X_i^k(t) = 1$. Similar to the previous case, this term contributes a positive value to the acceleration term and would result in higher change probability. This is highly desirable as the bit in the correponding value of bit in better particle is different than that of current particle. Hence change needs to be accelerated.

Hence in the case of XOR BGSA, the problems visited in the previous version of BGSA do not exist any more.

V. Simulation Results

A. Implementation of XOR BGSA on various benchmark examples

The proposed XOR BGSA is implemented on binary minimization of several benchmark problems. In all of simulations, 20 bits are used to decode numerical values into binary domain. The number of solutions considered (population size) are selected as to be equal to 50, maximum number of generations are selected to be equal to 1000. The results obtained using the proposed XOR BGSA are compared with that of BGSA [12], NBPSO [13], IBPSO [14] and BPSO [11].

The considered optimization problems are a collection of functions with single optimum problem and functions with multiple local optimums. In these instances, while function number 1 has a single global optimum, function number 6 suffers from multiple local minimums. On the other hand, although function number 10 has a single global minima, the function changes far from the global optimum are very small making it hard to find its global minima. The parameter N represents the dimension of these optimization problems which is considered to be equal to 3, 5 and 10 to test the optimization algorithms for different dimensions. These dimensions are multiplied by the number of bits taken for decoding real values, which results in optimization in 60, 100 and 200 dimensional space. The mathematical formula for the test functions are as:

$$f_1(X) = \sum_{i=1}^{N} x_i^2 \tag{15}$$

$$f_2(X) = \prod_{i=1}^{N} i x_i^2 \tag{16}$$

$$f_3(X) = \sum_{i=1}^{N} (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \tag{17}$$

$$f_4(X) = 10.N + \sum_{i=1}^{N} x_i^2 - 10cos(2\pi x_i) \tag{18}$$

$$f_5(X) = -\sum_{i=1}^{N} x_i sin(\sqrt{|x_i|}) \tag{19}$$

$$f_6(X) = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 + \prod_{i=1}^{N} cos\left(\frac{\cos(x_i)}{\sqrt{i}}\right) + 1 \tag{20}$$

$$f_7(X) = \sum_{i=1}^{N} |x_i|^{i+1} \tag{21}$$

$$f_8(X) = -20exp\left(\frac{0.2}{N}\sqrt{\sum_{i=1}^{N} x_i^2}\right) + 20 + exp(1) + exp\left(\frac{1}{N}\sum_{i=1}^{N} cos(cx_i)\right) \tag{22}$$

$$f_9(X) = -\sum_{i=1}^{N} sin(x_i)(sin(ix_i^2/\pi))^{20} \tag{23}$$

$$f_{10}(X) = -\sum_{i=1}^{30} \frac{1}{\sum_{j=1}^{N}(x_j - a_{ij})^2 + c_j} \tag{24}$$

where:

$$a_{i1} = c_1 = 1,\ a_{i2} = c_2 = 2,\ a_{i3} = c_3 = 3,\ a_{i4} = c_4 = 4,\ a_{i5} = c_5 = 5 \tag{25}$$

The optimization results are presented in Table II, optimization in each case are repeated 10 times and the mean values are reported in tables. It can be seen that the proposed optimization algorithm performs well in optimizing unimodal as well as multi modal optimization problems. Overall table illustrates that XOR BGSA generally outperforms most of other optimization algorithms. In the cases when it is not the best optimization algorithm, it is still very close to the desired results and often outperforms the others.

TABLE II. Comparison results for minimizing some Benchmark optimization problem

	D	XOR BGSA	BGSA [12]	NBPSO [13]	IBPSO [14]	BPSO [11]
f1	3	**2.73e-10**	**2.73e-10**	**2.73e-10**	0.022604	0.083856
f2	3	**4.51e-30**	**4.51e-30**	**4.51e-30**	5.79e-09	8.35e-08
f3	3	1.6053	**0.44184**	0.925724	41.76314	2.387039
f4	3	1.4754	0.458256	**5.41e-08**	4.28478	2.798389
f5	3	-11.8285	**-11.8336**	-11.8267	-11.8108	-11.8153
f6	3	1.3472	**1.344182**	**1.344182**	1.346191	1.348781
f7	3	**9.10e-11**	**9.10e-11**	**9.10e-11**	0.108468	0.025637
f8	3	-38.2848	**-38.3066**	**-38.3066**	-37.9596	-37.9904
f9	3	-2.73276	-2.87440	**-2.93843**	-2.53957	-2.57601
f10	3	-4.9265	**-5.00000**	**-5.00000**	-4.64909	-4.90473
f1	5	**4.55e-10**	1.20e-06	1.20e-06	0.648581	2.397748
f2	5	**7.47e-49**	**7.47e-49**	5.63e-40	3.41e-12	8.47e-06
f3	5	66.9187	**2.552033**	3.938883	161.9952	249.0183
f4	5	5.9657	1.989918	**0.497177**	13.55983	26.4738
f5	5	-19.6977	**-19.7242**	-19.6962	-19.396	-19.2005
f6	5	1.2804	1.273324	**1.272846**	1.288415	1.28603
f7	5	**9.10e-11**	**9.10e-11**	1.78e-10	2.092338	2.80811
f8	5	-24.2369	**-24.2549**	-24.2356	-23.8808	-23.2707
f9	5	-4.4281	-4.69137	**-4.76623**	-4.07088	-2.92115
f10	5	-1.8449	**-2.00000**	-1.99984	-1.61175	-1.73352
f1	10	**9.09e-10**	7.98e-02	0.054912	12.77626	28.04486
f2	10	**1.41e-94**	3.13e-86	3.08e-34	1.68e-05	24.49671
f3	10	83.8356	**16.98277**	30.5305	4663.245	21640.31
f4	10	25.8926	**5.579333**	17.68303	61.1481	109.696
f5	10	-39.4067	**-39.4207**	-39.3882	-37.7965	-31.9411
f6	10	1.2112	**1.197717**	1.204262	1.233218	1.264858
f7	10	**9.10e-11**	**9.10e-11**	1.72e-04	1222.224	856.3973
f8	10	-13.5442	**-13.5509**	-13.5343	-12.2016	-11.0112
f9	10	-8.5318	**-9.28633**	-7.74886	-5.72851	-3.78485
f10	10	-0.8714	**-1.00000**	-0.95560	-0.63312	-0.44705

B. Implementation of XOR BGSA on knapsack problem (MKP)

For the second class of optimization problem the knapsack problem is considered. In the knapsack problem, it is desirable to maximum the profit while the capacity for resources is limited. Example applications of knapsack problem are databases in a distributed computer system,

project selection, cargo loading, and cutting stock problems [15]. The problem is formulated in an explicit way as:

$$\begin{aligned} \mathrm{f}(\mathbf{x}) &= \max \ \sum_{i=1}^{n} p_i x_i \\ \text{s.t. } & \sum_{i=1}^{n} r_{ij} x_i \le C_j \,, j = 1, \dots, m \\ & x_i \in \{0,1\} \, \forall i \in \{1, \dots, n\} \end{aligned} \quad (26)$$

where n is the number of items, m is the number of constrains, $p_i \geq 0$ represents the profit value for each item, resource consumption is expressed by r_{ij} and the capacity is denoted by C_j. The constrained optimization problem can be converted to unconstrained one as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} p_i x_i + \beta \sum_{i=1}^{n} \sum_{j=1}^{m} min\big(C_j - r_{ij} x_i, 0\big) \quad (27)$$

where the parameter β is penalty coefficient considered to be equal to 10^{10}. There exist several databases for knapsack optimization problem which specify some values for p_i's, C_j's and r_{ij}'s. Among them weing [16] and weish [17] are selected for performance comparison of the proposed XOR BGSA approach with existing optimization algorithms in literature. The number of particles taken for swarms are selected to be equal to 50 with maximum number of iteration selected equal to 1000.

TABLE III. COMPARISON RESULTS FOR MAXIMIZING KNAPSACK PROBLEM WITH WEIGHTS SELECTED AS WEING [17]

	D	Proposed BGSA	BGSA [13]	NBPSO [8]	IBPSO [16]	PSO [7]
weing1	**28**	139580	**139891.1**	134389.7	127604.3	136823.1
weing2	**28**	125255	**126077.6**	117380.6	103826.5	120795.3
weing3	**28**	**92425**	78883.3	72343	61822.1	80410.4
weing4	**28**	112142.3	**115346.3**	105057.1	99356.4	110500.5
weing5	**28**	**94144**	90822.5	80585.5	62798.6	87978.7
weing6	**28**	**128372.7**	126995.3	120242.9	109656.4	121342.8
weing7	**28**	1077578	**1082671**	1016653	913986.3	889170.2
weing8	**28**	438921.7	**455145.8**	255033.7	-3.7E+12	-7.1E+12

TABLE IV. COMPARISON RESULTS FOR MAXIMIZING KNAPSACK PROBLEM WITH WEISH [17] PARAMETER VALUES

	D	Proposed BGSA	BGSA [13]	NBPSO [8]	IBPSO [16]	BPSO [7]
weish01	**30**	**4486.0**	4261	3665.1	3640.6	3903.3
weish02	**30**	**4437.0**	4370.2	3822.4	3266.1	3969.7
weish03	**30**	**4070.7**	3865.8	2887.4	2905.1	3402.8
weish04	**30**	**4542.3**	4048.3	2984.7	2683.8	3099
weish05	**30**	**4173.0**	3693.1	2565.1	2391.5	2973.2
weish06	**40**	**5458.3**	5285.9	4364	4271.8	4612.2
weish07	**40**	**5334.3**	5111.6	3813.7	3621	4600.6
weish08	**40**	5410.3	**5422.6**	4296.2	4228.7	4676.1
weish09	**40**	**5095.0**	4880.6	2768.9	2872.7	3528.3
weish10	**50**	**6095.7**	5692.1	3389.7	3551.2	3496.7
weish11	**50**	**5201.7**	4848	2564.4	-3.5e+11	-5.4e+11
weish12	**50**	**5975.7**	5637.1	3406	-7e+09	3371.8
weish13	**50**	**5883.3**	5646.9	2852.7	3040.9	3208.4
weish14	**60**	**6279.3**	6153.2	3744.8	-1e+11	-7.1e+10
weish15	**60**	**6822.7**	6471.6	3328.5	-3.8e+10	-2e+11
weish16	**60**	**7033.0**	6517.4	3892.9	4191.5	4131.9
weish17	**60**	**8546.3**	8421.5	6706.5	7131.5	7101.7
weish18	**70**	9087.6	**9121.3**	6515.3	7262	7046.6
weish19	**70**	**7017.3**	6611.4	3328.2	-1.7e+11	-7.7e+11
weish20	**70**	**8985.0**	8556.4	5960.3	5606	5699.6
weish21	**70**	**8394.7**	7948.5	4525.6	4691.1	4674.6
weish22	**80**	**8467.7**	7489.1	3937.2	4917.3	-9.6e+10
weish23	**80**	**7542.7**	6932.2	3710.5	-4.7e+11	-1.4e+12
weish24	**80**	**9826.0**	9674.5	6997.2	7434.3	7677.6
weish25	**80**	**9386.0**	8842.2	5621.8	6023	5518
weish26	**90**	**8572.3**	7647.9	4767.2	-5.6e+10	-9.7e+11
weish27	**90**	**8683.7**	8021.3	4860.1	5081.3	-6.5e+11
weish28	**90**	**8276.6**	7855.5	4592.1	-7.3e+11	-1.9e+12
weish29	**90**	**8586.0**	7687.7	4789.6	-4.4e+11	-1.9e+12
weish30	**90**	**10728.3**	10304.8	7097.8	7104.5	7459.9

Tables III and IV summarize the results with D being the dimension of the problem and best results made boldfaced. The experiments are repeated 10 times and mean values are reported. Tables III and IV show that the proposed optimization algorithm outperform other binary optimization algorithms specially in cases when the dimension of the cost function is high. It is further observed that in the cases when it does not obtain the best results, the ranking of the proposed algorithm is *two* within studied algorithms.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper, the binary version of GSA has been analysed and some of the cases for which its probability update rule did not behave as expected are discussed. Based on this analysis, a novel version for BGSA is proposed that benefits from an acceleration term with an XOR operator. Analysis shows that such modification can alleviate the shortcomings of BGSA in that in the proposed algorithm particles stochatically approach best particles in all cases. The proposed XOR BGSA is compared with a number of other optimization algorithms on the optimization of number of unimodal and multimodal optimization problems as well as optimizing knapsack problems. It is shown that results obtained using the proposed algorithm outperform BGSA [12], NBPSO [13], IBPSO [14] and BPSO [11] techniques. Hence the modification proposed in this paper will perform well in binary optimization problems.

B. Future Works

In a flexible manufacturing system, automatic task planning of production elements including robots is required to be done automatically in an efficient and minimum time. As a future work, inspired by appreciable results obtained in this paper, the proposed optimization algorithm will be utilized in the task planning of an assembly application. The assembly task to be optimized will include pick and place of objects, such as fastening nuts and bolts, and a sealing operation. The order of performing basic tasks in this process will be important, resulting in a constrained optimization problem.

VII. ACKNOWLEDGEMENTS

This work is funded and supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant number: EP/R021031/1 - New Industrial Systems: Chatty Factories.

REFERENCES

[1] Jin, Yaochu and Branke, Jürgen. Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on evolutionary computation, 9(3):303–317, 2005.

[2] Engelbrecht, Andries P. Fundamentals of computational swarm intelligence. John Wiley & Sons, 2006.

[3] Biswas, Anupam and Mishra, KK and Tiwari, Shailesh and Misra, AK. Physics-inspired optimization algorithms: a survey. Journal of Optimization, 2013, 2013.

[4] Formato, Richard A. Central force optimization. Prog Electromagn Res, 77:425–491, 2007.

[5] Rashedi, Esmat and Nezamabadi-Pour, Hossein and Saryazdi, Saeid. GSA: a gravitational search algorithm. Information sciences, 179(13):2232–2248, 2009.

[6] Niknam, Taher and Golestaneh, Faranak and Malekpour, Ahmadreza. Probabilistic energy and operation management of a microgrid containing wind/photovoltaic/fuel cell generation and energy storage devices based on point estimate method and self-adaptive gravitational search algorithm. Energy, 43(1):427–437, 2012.

[7] Ibrahim, Ahmad Asrul and Mohamed, Azah and Shareef, Hussain. Optimal power quality monitor placement in power systems using an adaptive quantum-inspired binary gravitational search algorithm. International Journal of Electrical Power & Energy Systems, 57:404–413, 2014.

[8] Janjira Kongchuenjai and Sukan Prombanpong. Binary integer programming to solve the process planning problem for prismatic mixed parts. Proceedings of Academics World 95th International Conference, Pattaya, Thailand, pages 7–10, 2018.

[9] Özmen, Özkan and Batbat, Turgay and Özen, Tolgan and Sinanoğlu, Cem and Güven, Ayşegül. Optimum Assembly Sequence Planning System Using Discrete Artificial Bee Colony Algorithm. Mathematical Problems in Engineering, 2018, 2018.

[10] Oesterle, Jonathan and Amodeo, Lionel. Efficient multi-objective optimization method for the mixed-model-line assembly line design problem. Procedia CIRP, 17:82–87, 2014.

[11] Kennedy, James and Eberhart, Russell C. A discrete binary version of the particle swarm algorithm. Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, pages 4104–4108, 1997. IEEE.

[12] Rashedi, Esmat and Nezamabadi-Pour, Hossein and Saryazdi, Saeid. BGSA: binary gravitational search algorithm. Natural Computing, 9(3):727–745, 2010.

[13] Khanesar, Mojtaba Ahmadieh and Teshnehlab, Mohammad and Shoorehdeli, Mahdi Aliyari. A novel binary particle swarm optimization. Control & Automation, 2007. MED'07. Mediterranean Conference on, pages 1–6, 2007. IEEE.

[14] Yuan, Xiaohui and Nie, Hao and Su, Anjun and Wang, Liang and Yuan, Yanbin. An improved binary particle swarm optimization for unit commitment problem. Expert Systems with applications, 36(4):8049–8055, 2009.

[15] Chu, P.C. and Beasley, J.E., A genetic algorithm for the multidimensional knapsack problem. Journal of heuristics, 4(1):63-86, 1998.

[16] H. M. Weingartner and D. N. Ness, "Methods for the solution of the multidimensional 0/1 knapsack problem," Operations Research, vol. 15, no. 1, pp. 83–103, 1967.

[17] W. Shih, "A branch and bound method for the multiconstraint zero-one knapsack problem," Journal of the Operational Research Society, vol. 30, no. 4, pp. 369–378, 1979.