applied sciences

MDPI

Article

XOR Binary Gravitational Search Algorithm with Repository: Industry 4.0 Applications

Mojtaba Ahmadieh Khanesar [1,*], Ridhi Bansal [2], Giovanna Martínez-Arellano [1] and David T. Branson [1]

[1] Faculty of Engineering, University of Nottingham, Nottingham NG7 2RD, UK; Giovanna.MartinezArellano@nottingham.ac.uk (G.M.-A.); David.Branson@nottingham.ac.uk (D.T.B.)
[2] Department of Aerospace Engineering and Bristol Robotics Lab, University of Bristol, Bristol BS16 1QY, UK; ridhi.bansal@brl.ac.uk
* Correspondence: Mojtaba.Ahmadiehkhanesar@nottingham.ac.uk

Received: 2 August 2020; Accepted: 2 September 2020; Published: 16 September 2020

Abstract: Industry 4.0 is the fourth generation of industry which will theoretically revolutionize manufacturing methods through the integration of machine learning and artificial intelligence approaches on the factory floor to obtain robustness and speed-up process changes. In particular, the use of the digital twin in a manufacturing environment makes it possible to test such approaches in a timely manner using a realistic 3D environment that limits incurring safety issues and danger of damage to resources. To obtain superior performance in an Industry 4.0 setup, a modified version of a binary gravitational search algorithm is introduced which benefits from an exclusive or (XOR) operator and a repository to improve the exploration property of the algorithm. Mathematical analysis of the proposed optimization approach is performed which resulted in two theorems which show that the proposed modification to the velocity vector can direct particles to the best particles. The use of repository in this algorithm provides a guideline to direct the particles to the best solutions more rapidly. The proposed algorithm is evaluated on some benchmark optimization problems covering a diverse range of functions including unimodal and multimodal as well as those which suffer from multiple local minima. The proposed algorithm is compared against several existing binary optimization algorithms including existing versions of a binary gravitational search algorithm, improved binary optimization, binary particle swarm optimization, binary grey wolf optimization and binary dragonfly optimization. To show that the proposed approach is an effective method to deal with real world binary optimization problems raised in an Industry 4.0 environment, it is then applied to optimize the assembly task of an industrial robot assembling an industrial calculator. The optimal movements obtained are then implemented on a real robot. Furthermore, the digital twin of a universal robot is developed, and its path planning is done in the presence of obstacles using the proposed optimization algorithm. The obtained path is then inspected by human expert and validated. It is shown that the proposed approach can effectively solve such optimization problems which arises in Industry 4.0 environment.

Keywords: physic inspired optimization algorithms; gravitational search algorithm (GSA); discrete binary optimization problems; assembly task planning; universal robot

1. Introduction

It is highly desirable to prepare products to be served and used for customers with their own dominant needs, interests, energy and use standards. Industry 4.0, the forth generation of industry, is a response to such varied requirements which is predicted to lead to highly customized and re-configurable production. Such technology rely on big data and benefits from machine learning and

artificial intelligence approaches. The use of such approaches may even make it possible to update production procedure to give appropriate response to fast changes in the design of product or make product customization possible [1]. To implement this on the factory floor, it is required to use robust intelligent techniques including advanced machine learning and artificial intelligence approaches to make the production line capable of implementing such varied production modifications. To avoid safety issues as well as wasting resources, a digital twin [2], a precise electronic replica of the factory floor, can be utilized. Such digital environments make it possible to design a robust system at lower cost and times to implement.

There exist different Industry 4.0 applications, in which the decision variables are binary rather than real values. Optimal task planning [3], flow shop scheduling [4] and assembly line design problems [5,6] are example applications of binary optimization algorithms in a digital manufacturing setup. The competition between different factories emerges mass production, while maintaining robustness and giving appropriate response to continuous design changes. In such an environment, it is required for the production elements, including robots and other automated machines, to perform task planning automatically to minimize the costs and time consumed, while maximizing production rates and profit. More specifically, rapid changes in products shortens the life-cycle time of production systems [7]. For certain product assembly lines, long-term shut down may be required to redesign the line and update recipes. Automated optimization of such processes are highly commercially beneficial to reduce these inefficiencies and improve production. Intelligent optimization algorithms may be beneficial for automatizing process design on the factory floor [2] which can consider energy, time and other processes. The requirement for optimization algorithms motivate us to develop a new algorithm, study its mathematical analysis and use it to perform an automated optimization process in Industry 4.0 applications.

Although classical optimization algorithms such as linear programming approaches, calculus of variations and gradient based optimization methods have proven to be efficient optimization algorithms, there are some barriers against use of these algorithms in all applications. When the function to be optimized is not explicitly defined and/or its partial derivative with respect to its parameters is hard to obtain because of high dimension and discontinuity, classical optimization processes fail. Intelligent optimization algorithms may be good candidates for high dimensional optimization problems. Intelligent optimization algorithms benefit from multiple start point, they are derivative free and are capable of finding multiple global optimums. Moreover, since these optimization algorithms are initiated from multiple points, they are less sensitive to initial conditions. Evolutionary optimization algorithms [8], swarm intelligence [9] and physics-inspired optimization methods [10] are the main categories of intelligent optimization methods applied in these cases.

Each solution in swarm intelligence approaches is defined in terms of a d-dimensional vector representing position. A velocity vector updates the positions for the next generation. Velocity vector benefits from exploration as well as exploitation terms. The third category of optimization algorithms is physics-inspired approaches in which, other than position and velocity, each particle benefits from more physical properties such as force, mass, electrical charge, etc. Examples of physics-inspired optimization algorithms are Big Bang-Big Crunch [11], central force optimization algorithm [12] and gravitational search algorithm (GSA) [13].

Among physics-inspired optimization algorithms, GSA is one of the most successful optimization algorithms [14]. Each particle in this algorithm benefits from mass, position vector, velocity vector and acceleration vector. The value of mass assigned to each particle is proportional to its merit with highest value being assigned to the best particle and the lowest value being assigned to the worst one. The gravitational force acting on these particles make the worst particle accelerated towards a low number of "best" particles, while searching the whole space for a possible "better" solution.

To apply real-valued optimization algorithms to binary optimization problems, they are required to be modified considerably. In swarm-based and physic-inspired optimization algorithms, a randomly weighted velocity vector is added to position vector to obtain new position. The well-known

modification to such problems is to change the definition of velocity to probability of change in bits of each dimension [15,16]. However, the analysis in this paper shows that in the case of binary gravitational search algorithm (BGSA), the probability vector does not always behave as expected. To alleviate this problem, the velocity vector and consequently the probability of change is modified in this paper to include an exclusive or (XOR) operator.

In this paper, the way mass values are assigned to each object, in the current BGSA, acceleration equation and formula for update of the change probability and position are fully analysed. The acceleration term is modified to use an XOR term. A repository is added as well to improve the performance of the optimization algorithm. To test the efficacy and performance of the proposed optimization algorithm, it is validated in simulation against a number of benchmark optimization problems. It is shown that the proposed algorithm is capable of optimizing such benchmark optimization problems with superior performance when it is compared to BGSA [16], improved binary particle swarm optimization (IBPSO) [17], binary particle swarm optimization (BPSO) [15], binary grey wolf optimization (BGWO) [18] and binary dragonfly optimization algorithm (BDA) [19].

Next, two Industry 4.0 applications namely assembly task planning and industrial robot path planning are considered. Assembly task planning is then validated on a digital twin first and then implemented on a physical robotic system. For the assembly task planning, a mock industrial calculator is designed and 3D printed. The studied task is to put bolts in their place using a robot manufactured by universal robot company namely UR5. Such an optimization process makes the production line robust as any new configuration for nuts and bolts can be easily given to the robot and it will optimally solve the sequence. It is shown that the proposed optimization algorithm is capable of planning this task and the length of path travelled by the robot is decreased by almost 17% in comparison to the human generated original pathway which leads to reduction in time and cost. Furthermore, path planing of the same robot in the presence of obstacles in its digital twin is considered. It is shown that the proposed approach is successful in finding an optimal path. In real-world practice the path generated in the digital twin would then be sent for human expert approval before transfer to physical robot. Such an automated optimization task reduces the need for intervention, yet keeps human elements involved as a safety measure.

This paper is organized as follows: In Section 2, real-valued GSA is discussed. The current version of BGSA investigated in this paper is presented and analysed in Section 2. In Section 3, the proposed novel XOR BGSA is introduced and analysed. Moreover, a repository is added in Section 3 to improve the performance of XOR BGSA. Full theoretical analysis of the proposed method is given in Section 4. The proposed algorithm is then compared with several existing optimization algorithms for the optimization of several benchmark examples in Section 5. Implementation of the proposed algorithm on UR5 is presented in Section 5 as well. Finally, the concluding marks are presented in Section 6.

2. Gravitational Search Algorithm

GSA is an optimization algorithm inspired by the physical forces between objects due to Newtonian forces between masses. Compared to PSO, in this optimization algorithm, each object benefits from more features such as acceleration term and the value of mass which is assigned with respect to the merit of each particle [13].

In this optimization algorithm, better particles are assigned a higher value of mass which causes other objects to be accelerated towards these better objects while still scanning the whole search space. In this case, if a particle come up with a better solution, it replaces the position of previous good solution found and attracts other objects. After a few iterations, all solutions are converged towards the heaviest mass resulting in termination of the algorithm. The algorithm is explained mathematically in the upcoming subsection [13].

2.1. Basic Gravitational Search Algorithm

Each solution in d-dimensional search space is represented by the position of an object and is represented as follows:

$$X_i = (x_i^1, x_i^2, ..., x_i^j, ...x_i^d) \;\; i = 1, ..., N \tag{1}$$

where x_i^j is the jth component of the position of object and N is the total number of objects. The mass of particles are updated and normalized at tth iteration as follows [13].

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

where $m_i(t)$ is non-normalized mass value corresponding to ith particle at iteration number t which represents the quality of solution and is defined as follows [13].

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

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. Hence, in this case $m_i(t)$ satisfies $m_i(t) \in [0, \; 1]$ and the value of mass of best solution is equal to *one* while the value of mass corresponding to worst solution is equal to *zero*. The parameters $f_{worst}(t)$ and $f_{best}(t)$ are updated at every iteration as follows:

$$\begin{aligned} f_{worst}(t) &= max\{f(X_i)\}_{i=1,...,N} \\ f_{best}(t) &= min\{f(X_i)\}_{i=1,...,N} \end{aligned} \tag{4}$$

It is to be noted that in this case the best and worst particles are defined based on minimization problem. In maximization problems, the updating equations for f_{worst} and f_{best} swap their definition. Furthermore, k_b number of best solutions are selected at each iteration as the masses which attract other masses.

The overall gravitational force acting on the ith particle is obtained as follows:

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

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.

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

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 ith mass by its value of mass as follows:

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

where $A_i(t) \in R^d$ is d-dimensional acceleration of the particles. It is further possible to update the velocity of objects using the following equation.

$$V_i(t+1) = p_i \; V_i(t) + A_i(t) \tag{8}$$

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

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{9}$$

2.2. Binary Gravitational Search Algorithm

The modification made to GSA to solve binary optimization problems is similar to BPSO [9]. In this case, the velocity vector in BGSA changes its meaning to the probability of change. Hence, a large absolute value for velocity in a dimension results in high possibility of change in such direction. Based on such a description, the function $P(V_i^j)$, the probability of change in jth bit of i th object, is considered to be a hyperbolic tangent of the velocity vector as follows [16].

$$P(V_i^j(t)) = tanh\left(\left|V_i^j(t)\right|\right), \; i = 1, ..., N, \; j = 1, ..., d \tag{10}$$

where $tanh(.)$ represents the hyperbolic tangent function as follows:

$$tanh\,(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{11}$$

Calculating the $P(V_i^j)$, the probability of change in every dimension, the next state in each dimension is calculated as follows [16].

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

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.

2.3. Analysis of Binary Gravitational Search Algorithm

An analysis of BGSA shows that in certain cases, the algorithm does not behave as expected. Focusing on acceleration term of BGSA as in (7), the following analysis is performed in different cases.

As the value assigned to parameter $X_k(t), \; k = 1, ..., N$ in each dimension may only be any binary value of *zero* and *one*, the result of subtraction $X_j(t) - X_i(t)$ in each dimension may be any of following possible conditions. In the following cases, undesirable behaviour of velocity vector, where occurred, is shown under unchanged sign of the velocity vector condition.

(1) In the case when $X_i^k(t) = X_j^k(t) = 0$ or $X_i^k(t) = X_j^k(t) = 1$. One has $X_i^k(t) - X_j^k(t) = 0$ which results in no acceleration for the dimension k which is reasonable. In this case, the value of bits in the kth dimension for the corresponding object and the best solutions are exactly the same and no change is required in that dimension.

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

(3) If $X_j^k(t) = 0, \; X_i^k(t) = 1, \; X_j^k(t) - X_i^k(t) = -1$ results in a negative acceleration term. In this case, if the velocity in kth dimension is positive, this acceleration term decreases the velocity and

consequently the probability of change. Since $X_j^k(t)$ is different from $X_i^k(t)$, this behaviour is not desirable. On the other hand, if the velocity in the kth 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 cases in which the acceleration term for BGSA does not behave as desired, this algorithm is modified using an XOR operator. It is shown that using such acceleration term, the undesirable conditions will never be met again.

3. Proposed XOR Binary Gravitational Search Algorithm

To alleviate the problems mentioned in Section 2, the BGSA is modified to have an XOR operator in its update rule for the acceleration term as follows:

$$A_i(t) = \sum_{j \in \{1,\ldots,k_b\}} r_j G(t) \frac{M_{rj}(t)(X_{rj}(t) \oplus X_i(t))}{\|X_i(t) - X_j(t)\|^{r_p} + \varepsilon} \tag{13}$$

where $X_{rj}(t)$ is the jth solution in the repository and $\oplus$ represents the XOR operator acting bit-wise on $X_{rj}(t)$ and $X_i(t)$ with its truth table being presented in Table 1. Furthermore, the equation used for $G(t)$ is as follows.

Table 1. Truth table for an exclusive or (XOR) operator.

A	B	A⊕B
0	0	−1
0	1	1
1	0	1
1	1	−1

$$G(t) = \alpha_0 \left(1 - \frac{t}{t_{max}}\right) + \alpha_1 \tag{14}$$

where α_0 and α_1 are selected as 0.8 and 0.2, respectively. The probability function $P(V_i^j)$ is modified as follows:

$$P\left(V_i^j(t)\right) = 0.5 + 0.5 \tanh\left(0.5 V_i^j(t)\right), \; i = 1, \ldots, N, j = 1, \ldots, d \tag{15}$$

where $tanh(.)$ is defined as in (11). The update equation for the velocity vector as well as position vector remains the same as (8) and (9). Moreover, the mass parameter in the proposed algorithm associated with particles in repository is updated as follows:

$$M_{ri}(t) = \frac{m_{ri}(t)}{\sum_{k=1}^{N} m_{rk}(t)} \tag{16}$$

and $m_{ri}(t)$ is non-normalized mass value corresponding to ith particle at iteration number t which represents the quality of solution and is defined using the fitness of particles in repository as follows:

$$m_{ri}(t) = \frac{f(X_i) - f_{r,worst}(t)}{f_{r,best}(t) - f_{r,worst}(t)} \tag{17}$$

where $f(X_{ri})$ is the fitness function associated with particle X_{ri} in the repository, $f_{r,worst}(t)$ and $f_{r,best}(t)$ represent the worst and the best fitness function of particles in repository.

While the original version of BGSA is memory-less, in the proposed algorithm a repository is added to the algorithm. In each iteration, a tournaments selection procedure is performed between the

current generation and old repository and the new repository is updated as a result of this selection procedure. Tournaments selection is widely known to be an algorithm to preserve the diversity in the solutions. The flowchart of the proposed XOR BGSA with repository is depicted in Figure 1, where V_{max} refers to the maximum absolute value of velocity in each dimension.

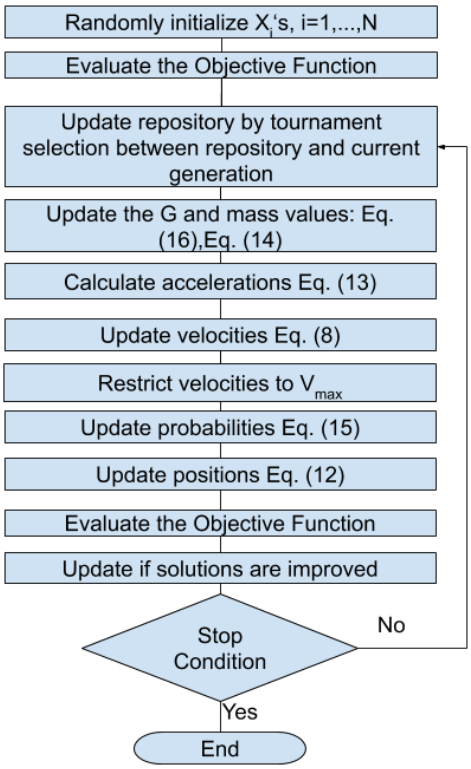


Figure 1. Flowchart of the proposed XOR binary gravitational search algorithm (BGSA).

4. Theoretical Analysis of the Proposed XOR Binary Gravitational Algorithm with Repository

4.1. General Analysis of the Proposed XOR BGSA

The proposed XOR BGSA is investigated statistically to show that it is more compliant with the expectations of acceleration and velocity vectors of BGSA. Since the parameter $X_k(t),\ k = 1, \ldots, N$ can only accept binary values of *zero* and *one*, the result of XOR operation $X_j(t) \oplus X_i(t)$ may fall in each of the following categories.

(1) If $X_i^k(t) = X_j^k(t) = 0$ or $X_i^k(t) = X_j^k(t) = 1$, in these cases $X_i^k(t) \oplus X_j^k(t) = -1$. It is expected from the acceleration term that since the corresponding bit in the best particle is the same as the bit in the current particle, the probability of change reduces. This expectation is fulfilled using (13).

(2) In the case $X_j^k(t) = 1,\ X_i^k(t) = 0$, $X_j^k(t) \oplus X_i^k(t) = 1$, the acceleration term introduces a positive value which contributes positively to the probability of change and increases its value.

(3) 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, since the corresponding bit in the better particle is different than that of the current particle, a positive acceleration term is obtained which results in higher probability of change. This is an expected behaviour from the system.

According to this analysis, the issues mentioned earlier for BGSA do not exist in the proposed XOR BGSA.

4.2. Full Statistical Analysis of the Proposed Method

To perform theoretical analysis of the proposed XOR BGSA, a single dimensional optimization problem is considered. The parameter k_b in this case is equal to *1*. Hence, in this section, we have a single particle whose behavior is studied and one best particle in the repository. The acceleration term, velocity term and the probability of change in the case of single dimensional optimization problem is obtained as follows:

$$a\left(t\right) = \frac{r_j G\left(t\right)\left(M_{rb}\left(x_{rb}\left(t\right) \oplus x\left(t\right)\right)\right)}{\left\| x_{rb}\left(t\right) - x\left(t\right)\right\|^{r_p} + \varepsilon} \tag{18}$$

$$v(t+1) = p_i v(t) + a(t) \tag{19}$$

$$P\left(v(t)\right) = 0.5 + 0.5\tanh\left(0.5v(t)\right) \tag{20}$$

where $a(t)$, *v(t)*, *P(v)* are single dimensional vectors representing the acceleration, velocity and the probability of change corresponding to the studied particle. Furthermore, $x\left(t\right)$ is the single dimensional vector representing the current position of studied particle and x_{rb} is its corresponding value in the best particle in the whole population and M_{rb} corresponds to the mass of the best particle in the repository. The minimum value of $G\left(t\right)$ is equal to α_1.

Two cases are investigated: the case when $p_i = 1$ and the case when $0 < p_i < 1$.

Case I: $p_i = 1$

Lemma 1. *If* $p_i = 1$ *and* $x = x_{rb}$, $\forall t > 0$ *we have* $a(t) \leq 0$ *and* $v(t+1) \leq v(t)$.

Proof. Considering (13), we have the following equation for the acceleration term $a(t)$:

$$a\left(t\right) = -r(t)G(t)M_{rb}D \leq 0 \tag{21}$$

where $D = \left\lfloor \frac{1}{\varepsilon} \right\rfloor$ is the dimension of the optimization algorithm. Consequently, considering (18), one obtains the following inequality:

$$v(t+1) = v(t) - r(t)G(t)M_{rb}D \leq v(t) \tag{22}$$

□

This implies that v_t is monotonically non-increasing function of time.

Lemma 2. *If* $p_i = 1$ *and* $x(t) = x_{rb}$ *then* $\forall \overline{v}$, $\exists T > 0$ *s.t.* $\forall t > T$ *we have* $E\left[v(t)\right] < \overline{v}$.

Proof. In the case when $v(t_0 - 1) < \overline{v}$, since according to Lemma 1 $v(t)$ is monotonically non-increasing function of time, we have:

$$v(t_0) \leq v(t_0 - 1) < \overline{v} \tag{23}$$

which means that $\forall t_0 <$ T Lemma 2 holds. However, in the case when $\overline{v} \leq v(t_0 - 1)$, we have:

$$E\left[v(T)\right] = v(t_0 - 1) + E\left[\sum_{t-1}^{T} \frac{r(t)G(t)M_{rb}\left(x_{rb}(t) \oplus x(t)\right)}{\left\| x_b\left(t\right) - x\left(t\right)\right\|^{r_p} + 1/D}\right] \tag{24}$$

Since $x(t) = x_{rb}$, we have the following equation:

$$E\left[v(T)\right] = v(t_0 - 1) + E\left[\sum_{t-1}^{T} -r(t)G(t)M_{rb}D\right] \tag{25}$$

Considering the fact that min$G\,(t)$ $=\alpha_1$, one obtains the following inequality:

$$v(t_0-1)-E\left[\sum_{t-1}^{T} r\,(t)\,G\,(t)\,M_{rb}D\right] \leq v(t_0-1)-\alpha_1 M_{rb}D\;E\left[\sum_{t_0-1}^{T} r(t)\right]$$

To find the appropriate T, the following inequality needs to hold.

$$v(t_0-1)-\alpha_1 M_{rb}D\;\left[\sum_{t_0-1}^{T} E[r\,(t)]\right] \leq \overline{v} \tag{26}$$

Considering the fact that $r\,(t)$ is a random variable uniformly distributed in [0, 1] , the following equation holds.

$$E\,[r\,(t)] = 0.5 \tag{27}$$

By applying expected value of $r\,(t)$ as in (27) to (26), the following equation is obtained.

$$v(t_0-1)-0.5\alpha_1 M_{rb}D\;\;(T-t_0+1) \leq \overline{v} \tag{28}$$

and finally T is obtained as follows:

$$\frac{2(v(t_0-1)-\overline{v})}{D\alpha_1}+t_0-1\leq \mathrm{T} \tag{29}$$

Since $v(t-1)\;<\;\overline{v}$, $\forall t_0>1$ from (29) we have $0\leq$T s.t. $\forall \mathrm{t}>\mathrm{T}$ one obtains $E\,[v(t)]<\overline{v}$. This concludes the proof. □

Remark 1. *Lemma 2 implies that in the case when the position in one dimensional optimization problem is the same as that of best particle, the velocity vector corresponding to that dimension grows in negative direction resulting in the probability of change to vanish to zero. This in turn results in no change in that dimension.*

Lemma 3. *If $p_i = 1$ and $x(t) \neq x_{rb}$, then $\forall \mathrm{t} > 0$, we have $a(\mathrm{t})\geq 0$ and $v(t+1)\geq v(t)$.*

Considering (17), we have the following equation for the acceleration term $a(t)$:

$$a\,(t) = \frac{r(t)G(t)M_{rb}}{1+1/D}\geq 0 \tag{30}$$

Consequently considering (30), one obtains the following inequality:

$$v(t)\leq v(t+1)=v(t)+\frac{r(t)G(t)M_{rb}}{1+1/D} \tag{31}$$

This implies that v_t is monotonically non-decreasing function of time.

Lemma 4. *If $p_i = 1$ and $x(t) \neq x_{rb}$ then $\forall \overline{v}$, $\exists \mathrm{T} > 0$ s.t. $\forall \mathrm{t} > \mathrm{T}$, we have $\overline{v} < E\,[v(t)]$.*

Proof. In the case when $\overline{v} < v(t_0-1)$ since according to Lemma 3, $v(t)$ is monotonically non-decreasing function of time we have:

$$\overline{v}\leq v(t_0-1) < v(t_0) \tag{32}$$

which agrees with the Lemma 4. However, in the case when $v(t_0-1)\leq\overline{v}$, we have:

$$E\,[v(T)] = v(t_0-1)+E\left[\sum_{t-1}^{T}\frac{r(t)G(t)M_{rb}\,(x_{rb}(t)\oplus \mathrm{x}(t))}{\|x_{rb}\,(t)-x\,(t)\,\|^{r_p}+1/D}\right] \tag{33}$$

Considering the fact that $x(t) \neq x_{rb}$, the following equation is obtained.

$$E\left[v(T)\right] = v(t_0 - 1) + E\left[\sum_{t-1}^{T} \frac{r(t)G(t)M_{rb}}{1 + 1/D}\right] \tag{34}$$

Considering the fact that min$G\left(t\right) = \alpha_1$, the following equation is obtained.

$$v(t_0 - 1) + \alpha_1 M_{rb} \frac{D}{1 + D} E\left[\sum_{t_0-1}^{T} r(t)\right] \leq v(t_0 - 1) + E\left[\sum_{t-1}^{T} \frac{r\left(t\right) G\left(t\right) M_{rb}}{1 + 1/D}\right]$$

Hence, to find T which satisfies Lemma 4, the following inequality needs to be hold.

$$\overline{v} \leq v(t_0 - 1) + \alpha_1 M_{rb} \frac{D}{1 + D} \left[\sum_{t_0-1}^{T} E[r\left(t\right)]\right] \tag{35}$$

Considering the fact that $r\left(t\right)$ is a random variable with uniform probability distribution function, the following equation holds.

$$E\left[r\left(t\right)\right] = 0.5 \tag{36}$$

By using the property of $r\left(t\right)$ as in (36), the following is obtained.

$$\overline{v} \leq v(t_0 - 1) + \frac{0.5\alpha_1 D\ M_{rb}\left(T - t_0 + 1\right)}{1 + D} \tag{37}$$

Hence, the value of T to satisfy Lemma 3 is obtained as follows:

$$\frac{2(1 + D)e(\overline{v} - v(t_0 - 1))}{D\alpha_1 M_{rb}} + t_0 - 1 \leq \mathrm{T} \tag{38}$$

Since $v(t_0 - 1) < \overline{v}$, $\forall \mathrm{t} > 1$ from (38), we have $0 \leq$ T s.t. $\forall \mathrm{t} >$ T and one has $E\left[v(t)\right] < \overline{v}$. This concludes the proof. □

Remark 2. *It immediately follows from Lemma 4 that in the case when the position in one dimensional optimization problem is different than that of the best particle, the velocity vector corresponding to that dimension grows resulting in the probability of change to increase to one which would result in increase in the probability of change.*

Theorem 1. *If* $p_i = 1$.

1. In the case when $x(t) = x_{rb}$, the acceleration term is non-positive and the velocity term is a non-increasing function of time and the probability of change converges to zero.
2. In the case when $x(t) \neq x_{rb}$, the acceleration term is non-negative and the velocity vector is a non-decreasing function of time and the probability of change converges to zero.

Proof. Using Lemmas 1–4, this theorem is concluded. □

Case II: $0 < p_i < 1$:

Lemma 5. *In the case when* $x(t) = x_{rb}$, $\forall \mathrm{t} > 0$ *the velocity vector* $v(t)$ *will never be larger than* $p_i^t v(0)$, *where* $v(0)$ *is the initial value of the velocity.*

In the case when $x(t) = x_{rb}$, *from* (17), *it follows that*

$$a\left(t\right) = -r(t)G(t)D\ M_{rb} \leq 0 \tag{39}$$

Considering (18), *the following equation is obtained for* $v(t)$

$$v(t) = p_i^t v(0) - D \sum_{\tau=1}^{t} r(\tau) G(\tau)\ M_{rb} \leq p_i^t v(0) \tag{40}$$

Hence, $p_i^t v(0)$ *is an upper bound for* $v(t)$.

Remark 3. *From Lemma 5, it is concluded that in the case when* $x(t) = x_{rb}$, *the acceleration term is non-positive acting in the direction of decreasing the probability of change. However, the impact of* $p_i^t v(0)$ *depends on the sign of* $v(0)$.

Lemma 6. *In the case when* $x(t) \neq x_{rb}$, $\forall t > 0$ *the velocity vector* $v(t)$ *will never be smaller than* $p_i^t v(0)$.

In the case when $x(t) \neq x_{rb}$, *from* (17), *it follows that*

$$a\,(t) = r(t) G(t) D\ M_{rb} \geq 0 \tag{41}$$

Considering (18), *the following equation is obtained for* $v(t)$

$$p_i^t v(0) \leq v\,(t) = p_i^t v(0) + D \sum_{\tau=1}^{t} r(\tau) G(\tau)\ M_{rb} \tag{42}$$

Hence, $p_i^t v(0)$ *is a lower bound for* $v(t)$.

Remark 4. *From Lemma 6, it is concluded that in the case when* $x(t) \neq x_b$, *the acceleration term is non-negative acting in favor of making probability of change in the corresponding bit higher. However, the impact of* $p_i^t v(0)$ *depends on the sign of* $v(0)$.

Theorem 2. *In XOR BGSA, considering sufficiently large number for maximum number of iterations, following properties hold.*

1. If $x(t) = x_{rb}$, $\exists$T s.t. $\forall t >$ T, the probability of change becomes smaller than 0.5:

$$P\,(x\,(t+1) = \overline{x}(t)) \leq 0.5$$

2. If $x(t) \neq x_{rb}$, $\exists$T s.t. $\forall t >$ T, the probability of change becomes larger than 0.5:

$$P\,(x\,(t+1) = \overline{x}(t)) \geq 0.5$$

In the case when $p_i = 1$, from Theorem 1 it follows that when $x = x_{rb}$, as mentioned earlier in Remark 1, the velocity value grows in negative direction which results in:

$$P\,(x\,(t+1) = \overline{x}(t)) = \frac{1}{1 + e^{-v(t)}} \to 0 \leq 0.5 \tag{43}$$

Moreover, in the case when $p_i = 1$, from Theorem 1 it follows that when $x(t) \neq x_{rb}$, as mentioned earlier in Remark 2, the velocity value grows larger which results in:

$$P\,(x\,(t+1) = \overline{x}(t)) = \frac{1}{1 + e^{-v(t)}} \to 1 \geq 0.5 \tag{44}$$

Hence in the case of $p_i = 1$, Theorem 2 holds. When $0 < p_i < 1$, two cases are considered. The case when $v(0) = 0$ and $v(0) \neq 0$.

If $v_0 = 0$, in the case of $x = x_{rb}$, the velocity term according to (18) is obtained as follows:

$$v(t) = -D\sum_{\tau=1}^{t} r(\tau)G(\tau)\ M_{rb} \leq 0 \tag{45}$$

Hence:

$$P\left(x\left(t+1\right) = \overline{x}(t)\right) = \frac{1}{1+e^{-v(t)}} \leq 0.5 \tag{46}$$

In the case of $x(t) \neq x_{rb}$, the velocity term according to (13) is obtained as follows:

$$v(t) = D\sum_{\tau=1}^{t} r(\tau)G(\tau)\ M_{rb} \geq 0 \tag{47}$$

Hence:

$$P\left(x\left(t+1\right) = \overline{x}(t)\right) = \frac{1}{1+e^{-v(t)}} \geq 0.5 \tag{48}$$

If $v(0) \neq 0$, in the case of $x(t) = x_{rb}$, considering the fact that

$$v\left(t\right) = p_i^t v(0) - D\sum_{\tau=1}^{t} r(\tau)G(\tau)\ M_{rb} \tag{49}$$

If $v(0) < 0$, from (49), it can be seen that $v\left(t\right)$ is the result of addition of two negative terms resulting in a negative value for $v\left(t\right)$. However, if $v(0) > 0$, considering the fact $E\left[r(t)\right] = 0.5$, we have:

$$E[v\left(t\right)] = p_i^t v(0) - DE\left[\sum_{\tau=1}^{t} r\left(\tau\right) G\left(\tau\right)\ M_{rb}\right] \leq p_i^t v(0) - \alpha_1.D\ M_{rb}\ E\left[\sum_{0}^{T} r\left(t\right)\right] = p_i^t v(0) - 0.5T\alpha_1.\ M_{rb}D \tag{50}$$

Hence, the time T after which we have $E[v\left(t\right)] < 0$, is obtained from

$$p_i^T v(0) - 0.5T\alpha_1.D\ M_{rb} < 0 \tag{51}$$

and

$$\frac{2p_i^T v(0)}{\alpha_1 D\ M_{rb}} < T\ . \tag{52}$$

Hence, after a sufficiently large value of iterations specified by T, $v\left(t\right)$ becomes negative and stays negative onwards. After that considering (19), we have the following:

$$P\left(x\left(T+1\right) = \overline{x}(T)\right) = \frac{1}{1+e^{-v(T+1)}} \leq 0.5 \tag{53}$$

However, if $v(0) \neq 0$, in the case of $x(t) \neq x_b$, considering the fact that

$$v\left(t\right) = p_i^t v(0) + D\sum_{\tau=1}^{t} r(\tau)G(\tau)\ M_{rb} \tag{54}$$

If $v(0) > 0$, from (54), it can be seen that $v\left(t\right)$ is the result of addition of two positive terms resulting in a positive value for $v\left(t\right)$. However, if $v(0) < 0$, similar to the analysis which resulted in (50) and considering the fact $E\left[r(t)\right] = 0.5$, we have:

$$E[v\left(t\right)] = p_i^t v(0) + DE\left[\sum_{\tau=1}^{t} r\left(\tau\right) G\left(\tau\right)\ M_{rb}\right] \leq p_i^t v(0) + \alpha_1.D\ M_{rb}\ E\left[\sum_{0}^{T} r\left(t\right)\right] = p_i^t v(0) + 0.5T\alpha_1.D\ M_{rb} \tag{55}$$

Hence, the time T after which we have $v(t) > 0$, is obtained from the following inequality

$$p_i^T v(0) + 0.5T\alpha_1 .D\ M_{rb} > 0 \tag{56}$$

and

$$\frac{-2p_i^T |v(0)|}{\alpha_1 D\ M_{rb}} < T\,. \tag{57}$$

Hence after sufficiently large value of iterations specified by T in (57), $v(t)$ becomes positive and stays positive on-wards. After that, considering the probability of change as in (19), it follows that:

$$P\left(x\left(T+1\right) = \overline{x}(T)\right) = \frac{1}{1+e^{-v(t)}} \geq 0.5 \tag{58}$$

This concludes the proof.

Remark 5. *It immediately follows from Theorem 2 that if the value of a particle in one dimension is different than that of the best particle, the probability of change if not immediately, after sufficiently large number of iterations becomes larger than 0.5 making change in the corresponding dimension more probable than not. Moreover, if the value of particle in a dimension is similar to that of the best particle, the probability of change in that dimension if not immediately, after sufficiently large number of iterations becomes smaller than 0.5 making change in the corresponding dimension less probable.*

Remark 6. *The results obtained in Section 4 are valid for the case when $k_b = 1$. However, these results can be easily extended to the case when k_b has a greater value in which case the value of k_b particle effect other particles with a gain proportional to their mass. Hence, the overall probability of change is influenced by each of k_b particles disproportionately.*

5. Simulation Results

5.1. Benchmark Optimization Problem

To evaluate the efficacy and performance of the proposed XOR BGSA with repository, it is tested against several benchmark problems. In optimization of benchmark problems, 20 bits are used to decode numerical values into the binary domain. The number of solutions considered (population size) is equal to 25, maximum number of generations are then selected to be equal to 1000. Two sets of comparisons are presented. In the first set of optimization, the proposed XOR BGSA is compared to BGSA [16], IBPSO [17], BPSO [15], BGWO [18] and BDA [19]. The main goal in these simulations is to demonstrate how the modifications proposed in this paper including the addition of XOR to the acceleration equation for BGSA as well as the repository effect its performance.

As in the real world, an optimization problem may be applied to a wide range of different functions including unimodal and multimodal problems, the proposed optimization algorithm is tested against both classes of optimization problems. The ability of the proposed optimization algorithm in dealing with problems with multiple local minimas is required to be investigated as well. For having such study, some of the optimization problems considered as the benchmark optimization problems include multiple local minimas. The benchmark problems considered are a collection of functions with single optimum solutions and functions with multiple local optimums. Other than these specifications, it is required to illustrate the performance of the proposed optimization algorithm in dealing with non-differentiable problems. The specifications of benchmark optimization problems are mentioned in front of them for ease of reference. The parameter N represents the dimension of these optimization problems which is considered to be equal to either 1, 2, 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 20, 40, 100 and 200 dimensional space. The mathematical formulas for the test functions as well as their specifications are as follows [20].

Sphere function [21]**:** a differentiable, separable, unimodal function without local minima

$$f_1(X) = \sum_{i=1}^{N} x_i^2 \tag{59}$$

Multiplication of squares: a differentiable, non-separable, unimodal function without local minima

$$f_2(X) = \prod_{i=1}^{N} i x_i^2 \tag{60}$$

Griewark function [22]**:** a differentiable, non-separable, multimodal function with local minimas

$$f_3(X) = \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{61}$$

Schwefel function [23]**:** a differentiable, separable, multimodal function with local minima

$$f_4(X) = -\sum_{i=1}^{N} x_i sin(\sqrt{|x_i|}) \tag{62}$$

Rastrigin Function [22]: a differentiable, separable, multimodal function with local minima

$$f_5(X) = 10.N + \sum_{i=1}^{N} \left(x_i^2 - 10cos\left(2\pi x_i\right)\right) \tag{63}$$

Styblinski-Tang [24]**:** a differentiable, separable, multimodal function with local minima

$$f_6(X) = \frac{1}{N} \sum_{i=1}^{N} \left(x_i^4 - 16x_i^2 + 5x_i\right) \tag{64}$$

Michalewicz function [22]**:** a differentiable, separable, multimodal function with local minima

$$f_7(X) = -\sum_{i=1}^{N} sin(x_i)\left(sin(ix_i^2/\pi)\right)^{20} \tag{65}$$

Ackley function [25]**:** a differentiable, non-separable, multimodal function with local minima

$$f_8(X) = -ae^{-b\sqrt{\frac{1}{d}\Sigma_{i=1}^{30} x_i^2}} - e^{\frac{1}{d}\Sigma_{i=1}^{30}\cos(cx_i)} + \mathrm{a} + e^1 \tag{66}$$

Rotated hyperellipsoid function [22]**:** a differentiable, scalable, unimodal function without local minima

$$f_9(X) = \sum_{i=1}^{N} \sum_{j=1}^{i} x_j^2 \tag{67}$$

Alpine 1 function [22]**:** a non-differentiable, separable, multimodal function with local minima

$$f_{10}(X) = \sum_{i=1}^{N} |x_i \sin(x_i) + 0.1x_i| \tag{68}$$

Cosine mixture function [22]**:** a differentiable, separable, multimodal function with local minima

$$f_{11}(X) = -0.1 \sum_{i=1}^{N} \cos\left(5\pi x_i\right) - \sum_{i=1}^{N} x_i^2 \tag{69}$$

The results of comparison between the proposed algorithm and five other existing binary optimization algorithms are presented in Table 2. Optimization process in each case is repeated for 30 times and the mean values are reported. It can be seen that the proposed optimization algorithm performs well in optimizing uni-modal as well as multi modal optimization problems. In 28 cases out of 44 cases, the proposed method outperforms other investigated optimization algorithms. It is further observed that when the proposed method is not the best optimization algorithm, it is still very close to the desired results. For instance, in the case of optimization function number 1, a uni-modal optimization problem, the proposed version of BGSA outperforms other algorithms in all different number of dimensions except for the case of $n = 1$ where it is the second best optimization algorithm after BDA. Optimization function number 11, $f_{11}(x)$, is a multi-modal optimization problem with several local minimums for which the proposed XOR BGSA outperforms the other five investigated optimization algorithms in all four dimensions. It can further be observed from the Table 2 that BDA ranks the second after the proposed optimization algorithm.

Table 2. Optimization results using various optimization algorithms for the minimization of several benchmark optimization problems.

Function Number	n	d	Proposed	BGSA [16]	IBPSO [17]	BPSO [15]	BGWO [18]	BDA [19]
$f_1(X)$	1	1	3.64E-10	6.55E-10	0.003499	2.98E-07	0.079185	**9.09E-11**
$f_1(X)$	1	2	**7.28E-10**	0.000421	0.110086	0.004851	3.173297	8.66E-08
$f_1(X)$	1	5	**1.82E-09**	0.01808	3.837217	3.492558	54.25284	0.025257
$f_1(X)$	1	10	**1.24E-07**	1.640722	23.62139	43.34931	212.2319	0.892732
$f_2(X)$	2	1	3.64E-10	3.64E-10	8.62E-05	2.73E-07	0.153546	**9.09E-11**
$f_2(X)$	2	2	**2.65E-19**	2.81E-18	0.000454	2.5E-07	0.358629	2.25E-16
$f_2(X)$	2	5	**7.65E-46**	7.08E-34	7.45E-05	7.57E-05	2.79E-05	1E-22
$f_2(X)$	2	10	**8.3E-82**	2.57E-56	12.26801	1274.831	2.09E-11	5.44E-22
$f_3(X)$	3	1	**−1**	−0.9999	−0.99705	**−1**	−0.98737	**−1**
$f_3(X)$	3	2	−0.99488	−0.99375	−0.98506	−0.99787	−0.94229	**−0.99901**
$f_3(X)$	3	5	−0.95717	−0.96513	−0.90313	−0.91313	−0.74288	**−0.96923**
$f_3(X)$	3	10	**−0.90475**	−0.89211	−0.64083	−0.5024	−0.51453	−0.88339
$f_4(X)$	4	1	**−19.4256**	−19.4253	−3.94361	−3.9453	−18.4693	−3.9453
$f_4(X)$	4	2	**−38.8511**	−38.8511	−7.8514	−7.8897	−31.8297	−7.89018
$f_4(X)$	4	5	**−97.1278**	−97.1165	−19.0216	−19.1181	−55.8132	−19.6948
$f_4(X)$	4	10	**−194.255**	−193.588	−35.5675	−31.5541	−79.5703	−39.2195
$f_5(X)$	5	1	3.64E-10	3.64E-10	7.76E-05	3.32E-07	0.372345	**9.09E-11**
$f_5(X)$	5	2	**3.64E-10**	0.006168	0.140178	0.002341	1.452548	1.08E-09
$f_5(X)$	5	5	**3.64E-10**	0.13037	9.419671	4.551582	2659.276	0.002952
$f_5(X)$	5	10	**3.64E-10**	17.81345	4293.123	2334.993	1.06E+08	4.71886
$f_6(X)$	6	1	**−78.3323**	−78.1186	−77.5076	**−78.3323**	−67.8744	**−78.3323**
$f_6(X)$	6	2	−156.094	−147.94	−151.238	−156.496	−116.85	**−156.659**
$f_6(X)$	6	5	−338.666	−327.884	−314.902	−322.303	984.8181	**−369.827**
$f_6(X)$	6	10	−641.095	−630.392	−431.271	−400.089	21937.59	**−645.258**
$f_7(X)$	7	1	**−0.99419**	−0.97691	−0.88143	−0.98787	−0.88543	−0.98792
$f_7(X)$	7	2	−1.95034	−1.95069	−1.68317	−1.89381	−1.66732	**−1.95806**
$f_7(X)$	7	5	−4.66631	**−4.67197**	−3.45559	−2.81238	−3.45303	−4.40256

Table 2. *Cont.*

Function Number	n	d	Proposed	BGSA [16]	IBPSO [17]	BPSO [15]	BGWO [18]	BDA [19]
$f_1(X)$	1	1	3.64E-10	6.55E-10	0.003499	2.98E-07	0.079185	**9.09E-11**
$f_7(X)$	7	10	−8.83999	**−9.00763**	−5.11393	−3.84276	−5.97745	−7.38148
$f_8(X)$	8	1	7.63E-05	7.63E-05	0.17688	0.001563	1.270744	**3.82E-05**
$f_8(X)$	8	2	**7.63E-05**	0.002378	0.873989	0.346521	3.054737	0.00026
$f_8(X)$	8	5	**7.63E-05**	0.497182	4.210083	4.515039	9.078786	0.311861
$f_8(X)$	8	10	**0.07325**	1.479706	6.721189	8.003571	11.92714	2.243791
$f_9(X)$	9	1	3.64E-10	3.64E-10	0.000344	2.89E-07	0.052176	**9.09E-11**
$f_9(X)$	9	2	**1.09E-09**	**1.09E-09**	0.285221	0.005263	3.989166	2.82E-08
$f_9(X)$	9	5	**5.46E-09**	0.267058	10.71721	7.755365	108.3779	0.011655
$f_9(X)$	9	10	**3.09E-07**	7.555599	125.9897	176.937	1128.728	5.058231
$f_{10}(X)$	10	1	4.63E-06	**2.32E-06**	0.000547	2.03E-05	0.008308	2.88E-06
$f_{10}(X)$	10	2	0.000948	0.000326	0.015704	0.003854	0.111948	**0.000223**
$f_{10}(X)$	10	5	0.045093	**0.021953**	0.898418	1.041691	1.199264	0.080774
$f_{10}(X)$	10	10	**0.189655**	0.304581	3.68566	6.668867	5.377988	0.748952
$f_{11}(X)$	11	1	**−400.1**	**−400.1**	−99.9787	−100.093	−398.391	−100.1
$f_{11}(X)$	11	2	**−800.2**	−799.77	−196.958	−198.57	−786.66	−200.195
$f_{11}(X)$	11	5	**−2000.5**	−1996.02	−461.362	−442.035	−1900.74	−498.794
$f_{11}(X)$	11	10	**−4000.99**	−3983.41	−839.887	−740.219	−3875.63	−974.665

5.2. Application to Knapsack

As the second class of benchmark optimization problem, the multi knapsack problem is considered. The problem formulation of 0/1 knapsack problem is discussed in summary in this part. Let there be n items with each having a positive profit $p_i > 0$ and a positive resource consumption $r_i > 0$. The main objective is to fill several knapsacks with resource capacity of C_j such that the accumulated profit of the items is maximized. Considering the constraints imposed by capacity of knapsacks, this problem is a constrained binary optimization problem. Such problems arises in various real-world applications such as in a distributed computer systems, warehouse optimization, cargo loading and stock problems [5,26]. This problem is explicitly formulated as follows:

$$\mathrm{f}(\mathbf{x}) = \max \sum_{i=1}^{n} p_i x_i \tag{70}$$

$$\text{s.t.} \sum_{i=1}^{n} r_{ij} x_i \leq C_j,\ j = 1, \ldots, m \tag{71}$$

$$x_i \in \{0,\ 1\}\ \forall i \in \{1, \ldots, n\}$$

where n represents the item number, m is the number of constrains, $p_i \geq 0$ represents the profit value for each item, resource consumption is expressed by $r_{ij} \geq 0$ and the capacity is denoted by another positive value $C_j \geq 0$. The cost function is multiplied by negative value of -1 to change maximization to minimization problem. The constraints are added as penalty terms to the cost function to obtain an unconstrained optimization problem as follows:

$$f_{uc}(\boldsymbol{x}) = -\sum_{i=1}^{n} p_i x_i + \beta \sum_{i=1}^{n} \sum_{j=1}^{m} min\left(C_j - r_{ij} x_i, 0\right) \tag{72}$$

where the parameter β is used as the penalty coefficient and assigned the value equal to 10^{10}. Several standard databases for knapsack optimization problems exist to provide means for comparison between the algorithms which provide special values for p_i's, C_j's and r_{ij}'s. Among these database of multi knapsack optimization available at the website of Queen Mary University of London (http://www.eecs.qmul.ac.uk/~jdrake/mkp.html) formerly hosted at University of Nottingham (http://www.cs.nott.ac.uk/~jqd/mkp/index.html) are selected for performance comparison of the proposed algorithms with several state-of-the-art binary optimization algorithms. The total number of populations are taken to be equal to 10 with maximum number of iterations selected to be equal to 1000.

Table 3 presents the results obtained using the proposed optimization algorithm versus other optimization problems. The solutions are the optimization solutions to the function represented by (72) which are multiplied by *minus one* to reflect the solutions to the original problem of (70) and (71). There exist several negative values in the table which show that in some cases, algorithms could not fulfil the constraints of optimization at least once. Such problem does not occur for the proposed XOR BGSA with repository. The best result obtained for each dimension is made bold faced to demonstrate its superior performance. The values used for optimization are collected from the website of Queen Mary University of London and their file names are presented for ease of reference. Problems have also been given a number for further discussions. As can be seen from Table 3. The proposed XOR BGSA with repository outperforms all other investigated optimization problems in almost all cases.

The convergence property of the proposed approach versus previous optimization methods for optimizations number 7 to 12 are demonstrated in Figure 2. As can be seen from this figure, the convergence of the proposed algorithm towards the solution is well above other investigated algorithms. Although during the first few iterations, some of the algorithms rise faster towards optimal solutions, after 100 iterations, the proposed XOR BGSA speeds up and becomes the dominant one. It can be concluded from the figure that other than quality of solutions, the speed of the proposed approach is well above other investigated optimization methods.

Table 3. Optimization results using various optimization algorithms for the maximization of the knapsack problem.

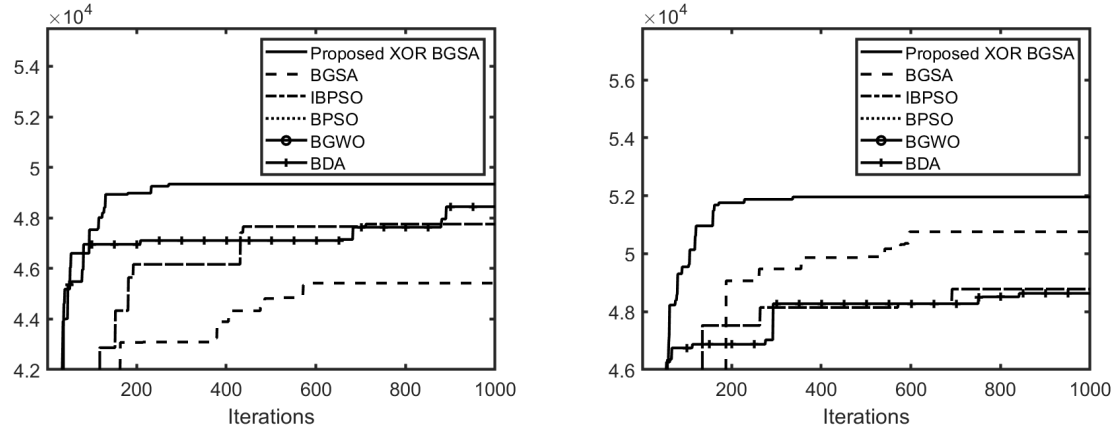
Problem Number	Dimension	Tightness Ratio	m	N	Filename	Proposed	BGSA [16]	IBPSO [17]	BPSO [15]	BGWO [18]	BDA [19]
1	100	0.25	5	10	OR5x100-0.25_1.dat	**21675.33**	20,742.73	−7E+13	−1.4E+14	−3.6E+14	20,278.77
2	100	0.25	5	10	OR5x100-0.25_2.dat	**21,223.93**	20,663	−9.2E+13	−1.4E+14	−3.4E+14	19,938.63
3	100	0.5	5	10	OR5x100-0.50_1.dat	**39,835.93**	39,302.37	37,097.47	37,573.93	37,093.93	38,813.2
4	100	0.5	5	10	OR5x100-0.50_2.dat	**39,812.4**	39,372.73	36,388.13	37,300.47	36,902.1	38,704.93
5	100	0.75	5	10	OR5x100-0.75_1.dat	**57,626**	56,945.5	52,385.83	50,954.87	55,652.37	56,107.47
6	100	0.75	5	10	OR5x100-0.75_2.dat	**59,946.2**	59,272.1	55,457.23	53,529.87	57,861.93	58,294.9
7	250	0.25	5	10	OR5x250-0.25_1.dat	**49,323.17**	46,860.27	−5.9E+14	−7.6E+14	−1.1E+15	46,805.43
8	250	0.25	5	10	OR5x250-0.25_2.dat	**51,256.37**	49,071.9	−6.1E+14	−7.7E+14	−1.2E+15	48,596.37
9	250	0.5	5	10	OR5x250-0.50_1.dat	**99,682.27**	99,393.17	94,943.5	94,993.6	94,801.07	97,535.43
10	250	0.5	5	10	OR5x250-0.50_2.dat	**99,759**	99,000.57	94,482.33	94,256.9	94,158.23	97621.53
11	250	0.75	5	10	OR5x250-0.75_1.dat	**142,598.2**	140,975.2	120,843.3	115,794.4	139,086.3	139,172.7
12	250	0.75	5	10	OR5x250-0.75_2.dat	**147,544.4**	145,975.8	124,749.9	119,751.7	143,700.6	144207.3

Table 3. *Cont.*

Problem Number	Dimension	Tightness Ratio	m	N	Filename	Proposed	BGSA [16]	IBPSO [17]	BPSO [15]	BGWO [18]	BDA [19]
13	500	0.25	5	15	OR5x500-0.25_1.dat	**98,610.83**	97,553.3	−1.5E+15	−1.9E+15	−2.4E+15	−3.2E+11
14	500	0.25	5	15	OR5x500-0.25_2.dat	**96,512.53**	96,034.53	−1.4E+15	−2E+15	−2.5E+15	92,190.13
15	500	0.5	5	15	OR5x500-0.50_1.dat	197448.2	**200,036.1**	189,901.2	188,863.1	189,116.2	194,592.8
16	500	0.5	5	15	OR5x500-0.50_2.dat	199,792.7	**202,340**	192,444.5	190,561.9	191,162.6	196,350
17	500	0.75	5	15	OR5x500-0.75_1.dat	**278,893.6**	278,751.9	228,713.8	215,389.8	274,491.7	272,973.2
18	500	0.75	5	15	OR5x500-0.75_2.dat	**289,925.9**	289,654.3	238,158.3	223,533.2	285,228	284,254.3
19	100	0.25	10	10	OR10x100-0.25_1.dat	**20,393.63**	18,793.87	−1.8E+14	−2.9E+14	−7.4E+14	18,583.23
20	100	0.25	10	10	OR10x100-0.25_2.dat	**19,867.93**	18,857.43	−1.8E+14	−2.6E+14	−7.1E+14	18,583.03
21	100	0.5	10	10	OR10x100-0.50_1.dat	**39051.77**	38,258.77	35,430.33	36,021.43	35,607.37	37669.33
22	100	0.5	10	10	OR10x100-0.50_2.dat	**40,356.13**	39,734.97	37007.3	37,450.4	36,969.97	38,926.1
23	100	0.75	10	10	OR10x100-0.75_1.dat	**55,408.4**	54,477.57	50,067.83	49,302.2	53,058.4	53,580.87
24	100	0.75	10	10	OR10x100-0.75_2.dat	**56,950.7**	56,234.7	51,602.53	50,940.77	54,119.3	55,144.23
25	250	0.25	10	10	OR10x250-0.25_1.dat	**49,901.6**	49,401.6	46,393.33	46,438.4	46,738.6	48,197.13
26	250	0.25	10	10	OR10x250-0.25_2.dat	**49,978.63**	49,191.07	46,514.7	46,100.77	46,874.6	48,557.03
27	250	0.5	10	10	OR10x250-0.50_1.dat	**101,134.7**	100,792.5	96,179.27	95,810.07	−4.8E+11	99,126.03
28	250	0.5	10	10	OR10x250-0.50_2.dat	**98,053**	97,841.1	92,281.53	92,293.1	−8.8E+11	95,604.23
29	250	0.75	10	10	OR10x250-0.75_1.dat	**144,205**	141,717.6	121,695.7	117142	139,645.4	139,894.6
30	250	0.75	10	10	OR10x250-0.75_2.dat	**141,698.4**	139,494.2	120,037.3	115,081.9	137,321.4	138,109.7
31	500	0.25	10	15	OR10x500-0.25_1.dat	**97,327.1**	96,654.03	−2.9E+15	−3.9E+15	−4.9E+15	93,571.07
32	500	0.25	10	15	OR10x500-0.25_2.dat	**96,865.7**	96,662.67	−3.1E+15	−4.1E+15	−5.1E+15	93,284.37
33	500	0.5	10	15	OR10x500-0.50_1.dat	195,540.6	**199,151.9**	188,580.9	187,265	−8.1E+11	192,963.6
34	500	0.5	10	15	OR10x500-0.50_2.dat	196,987.5	**199,922.2**	189,706.1	187,867.3	188,244	194,179
35	500	0.75	10	15	OR10x500-0.75_1.dat	**286,164.6**	285,945.3	237,022	220,772.4	281,163.6	279,953.5
36	500	0.75	10	15	OR10x500-0.75_2.dat	**284,177.3**	283,569.9	235,502.1	219,917.2	278,756.1	277,475.7

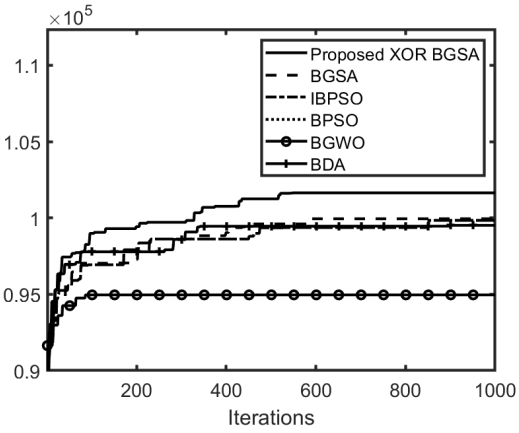
In most real-world optimization problems, the cost function evaluation requires high volume of computation. Hence, it would be highly desired for an optimization algorithm to result in superior performance after its single run. Box plot is a graphical method which gives more information about the diversity of the results rather than just its mean value. A small standard deviation leads to a smaller interquartile range which can be interpreted as its ability to avoid local minima. Outliers can also be easily spotted from box plot. In Figure 3 while the interquartile is illustrated using a blue box, the minimum and the maximum associated with data are demonstrated using the black line. There is a 50 percent chance that for a new experience, the result falls within the interquartile range. The outliers in this figure are represented by "+" sign. The median values are demonstrated with red line. The box plot obtained using different optimization algorithms for optimization problems 7 to 18 are illustrated. It can be observed from the figure that not only the median values obtained for the proposed approach in 10 cases out of 12 cases are the best, interquartile range obtained for the proposed approach in most cases is smaller meaning that it is more repeatable than in other investigated optimization algorithms. Furthermore, for optimization problems numbers 7, 8, 11 and 12, it can be observed that the interquartile range obtained for the proposed approach is well above the other optimization algorithms. This means that for every single run of the proposed optimization algorithm, there is

an appreciable chance we can obtain superior performance for the proposed optimization algorithm compared to other investigated optimization methods.

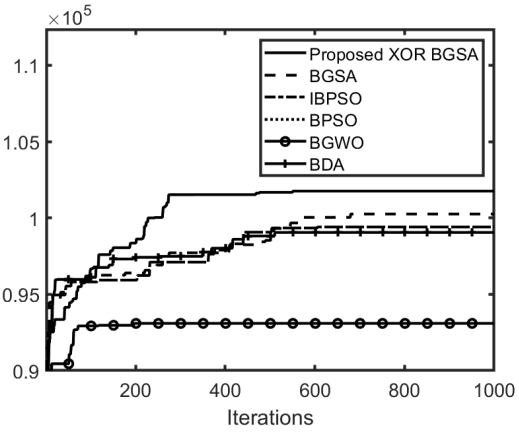


(**a**) Problem #7

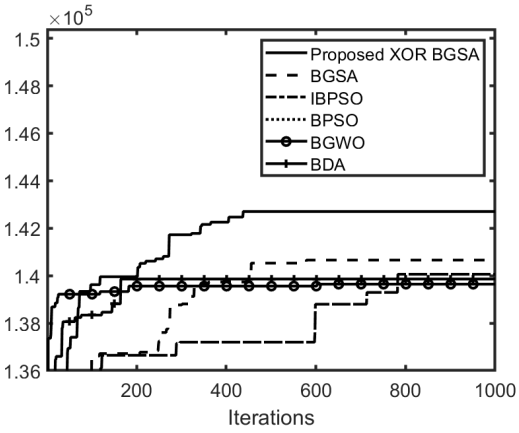
(**b**) Problem #8



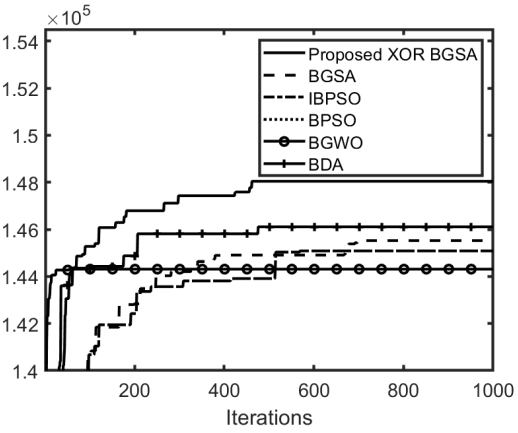
(**c**) Problem #9



(**d**) Problem #10

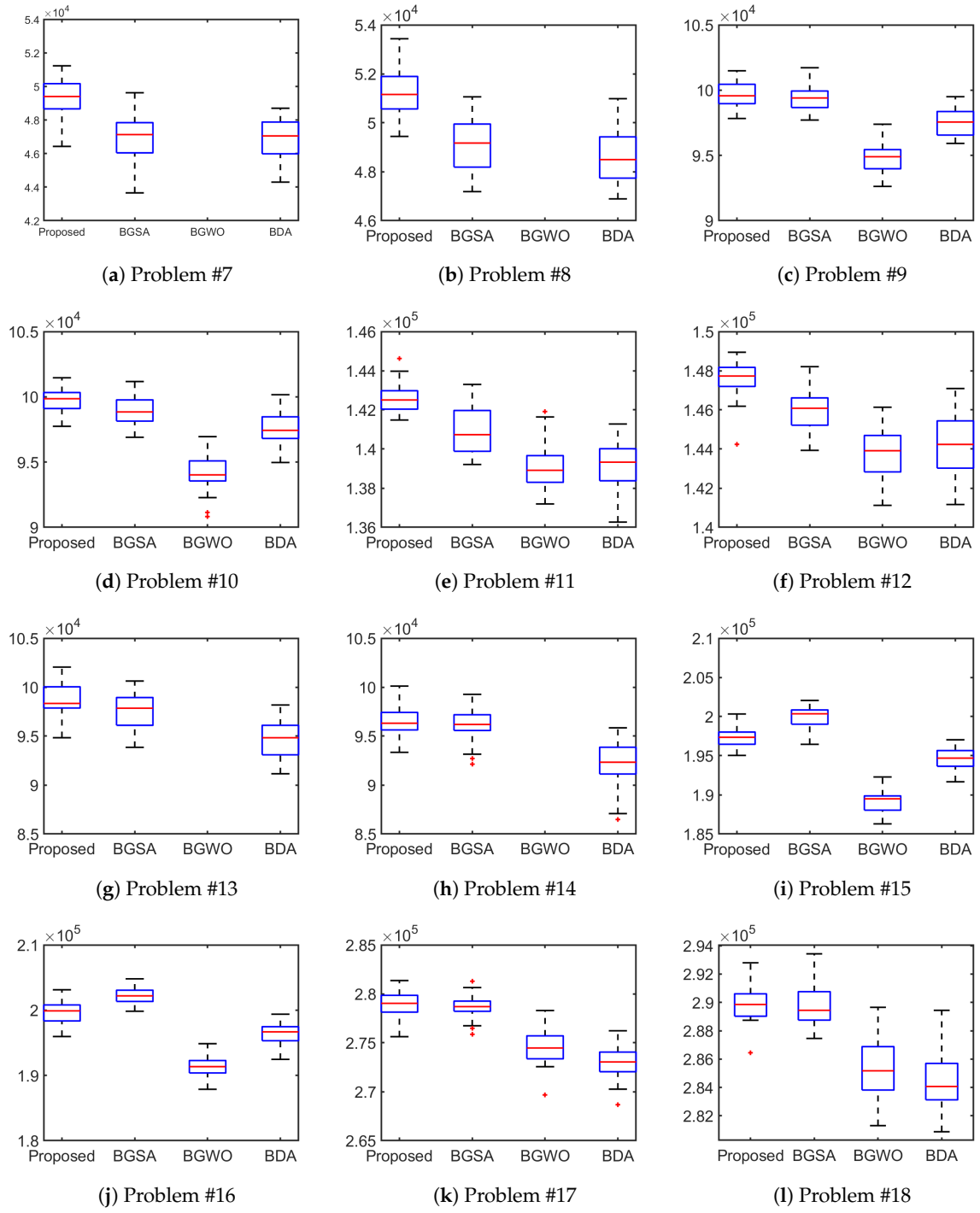


(**e**) Problem #11



(**f**) Problem #12

Figure 2. Trend of Cost function obtained from proposed XOR BGSA vs. four other algorithms namely: BGSA [16], improved binary particle swarm optimization (IBPSO) [17], binary particle swarm optimization (BPSO) [15], binary grey wolf optimization (BGWO) [18] and binary dragonfly optimization algorithm (BDA) [19]. Knapsack optimization problems numbers 7 to 12.



(**a**) Problem #7 (**b**) Problem #8 (**c**) Problem #9

(**d**) Problem #10 (**e**) Problem #11 (**f**) Problem #12

(**g**) Problem #13 (**h**) Problem #14 (**i**) Problem #15

(**j**) Problem #16 (**k**) Problem #17 (**l**) Problem #18

Figure 3. Box plot obtained for the knapsack problem using the proposed XOR BGSA vs. three other algorithms namely: BGSA [16], BGWO [18] and BDA [19]. Knapsack optimization problems 7 to 18.

6. Industry 4.0 Applications

In current manufacturing environments, modifications made to a product often require redesign and retraining/programming of production elements on the factory floor. Optimal or sub-optimal solutions to design changes in a process is the goal of decision-making processes. Recent work has explored showing that systematic optimization is possible in a well-defined detailed digital twin in which searching algorithms can perform multiple trial and errors without safety and resource restrictions. Additionally, current robot programming techniques require highly talented professionals

to perform programming, which increases response times and increases costs [27]. Real-time decision making and robot programming is therefore highly desirable and is a stated requirement of Industry 4.0 [28]. However, autonomous programming of robots will impact safety and certification requirements as well. To mediate such impacts, one solution is to tie the digital twin solutions to the real-production methods. Such a digital twin needs to be as accurate as possible to ensure compatibility of actions with real world. The simulated movements in such an environment can then be transformed to the real robot after full examination of resulting optimization solutions by a human expert. This framework makes the most of machine learning and artificial intelligence approaches. In this part, the proposed XOR BGSA with repository is used to find an optimal solution for assembly task planning and motion planning for an industrial robot namely UR5 that would then be passed to a human for certification before implementation. Previously the proposed XOR BGSA is tested against different types of optimization problems including uni-modal, multi-modal, those with single optimum solution, those which suffer from multiple local minimums and knapsack optimization problem which is a constrained one. The fact that the proposed optimization procedure obtained the best results among six binary optimization problems in most cases motivates us to use it in Industry 4.0 optimization problems.

It is required for robots to be able to optimally plan their own task autonomously otherwise massive resources may be used to redesign, reconfigure and reprogram them. To fulfil this goal, it is required to decode the planning process in the form of an optimization problem and use artificial intelligence approaches to solve the problem on a UR5 capable of handling 5 Kg loads to a precision of order of 0.1 mm .

The assembly task planning to be optimized in this application is to put bolts in their respective places while minimizing the distance. The bolts used in this study are from three different types of of $M6 \times 20$ mm, $M8 \times 20$ mm and $M10 \times 20$ mm. Hence, we have three different type of nuts which must be fastened using the bolts. The place and the position of the nuts and bolts are depicted in Figure 4 with the z component of position for nuts being on 5.7 cm and for bolts being equal to 7 cm.

Such an assembly task is similar to the benchmark problem frequently solved in computer science projects called the travelling salesperson. The cost function to be optimized in this case is the total distance travelled with main difference between the assembly task designed to test the proposed XOR BGSA and the travel salesperson being that in the current experiment, we have certain bolts which can be used to fasten certain nuts; this imposes some constrains on the optimization problem.

To encode the optimization problem, 31 bits are considered for each individual. Inspired by the results obtained by the proposed optimization procedure, it is used to minimize the total distance travelled by UR5 during the assembly task of industrial calculator (See Figure 5). In this case, there exist $2^{31} = 2.1475 \times 10^9$ different combinations for performing the assembly task. As can be seen from Figure 5, the first two bits determine which bolts are used to fasten nut #1, the bits numbers 3 and 4 are used to determine which bolts are used for nuts number 2, the fifth bit is for nut number 3. Since we have four $M10 \times 20$ mm bolts, the remaining bolt goes to nut number 4. Bit number 6 determines which $M8 \times 20$ mm is used to fasten nut number 5, the other one is used to fasten bolt number 6. The bit number 7 is used for bolts numbers 7 and 8. However, since the order of performing these tasks effects the cost function, it is required to determine the order of performing the tasks using the next 24 bits. All 3 of them making an integer value from the interval of $[0,\ 7]$. In this case, if the ith value after converting to decimal value is equal to j, the ith and jth task swap their position. This results in avoiding forbidden conditions which is difficult to avoid otherwise. The cost function to be optimized is the total distance travelled by robot with its initial conditions being at point $(10,\ 30,\ 10)$.

The evolution of total distance travelled by the robot is depicted in Figure 6. The total number of particles in this optimization is chosen to be 10 with the maximum number of iterations being equal to 200. As can be seen from this figure, the best random initialization for the total distance travelled by robot is 328.0 cm which decreases to 273.2 cm after optimization, resulting in a 16.7% improvement. The optimum solution of the problem is represented in Table 4 and the assembly process done by

UR5 is illustrated in Figure 7. The total assembly time is 116 seconds. Where for our laboratory safety reasons, the joint speed of the robot is limited to 90/s. However, as the UR5 is capable of working with joint speed equal to 180/s, the total assembly time could be reduced if it works under full speed.

Table 4. Optimal assembly solution obtained from the optimization algorithm.

Action	Bolt	Nut
Step 1	#8	#8
Step 2	#7	#7
Step 3	#4	#1
Step 4	#5	#5
Step 5	#3	#4
Step 6	#6	#6
Step 7	#2	#3
Step 8	#1	#2

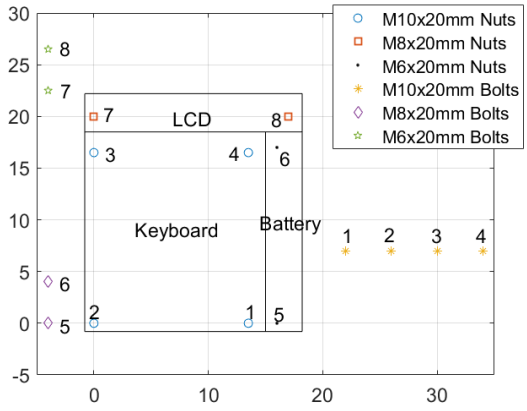


Figure 4. 2D position of nuts and bolts in assembly task.

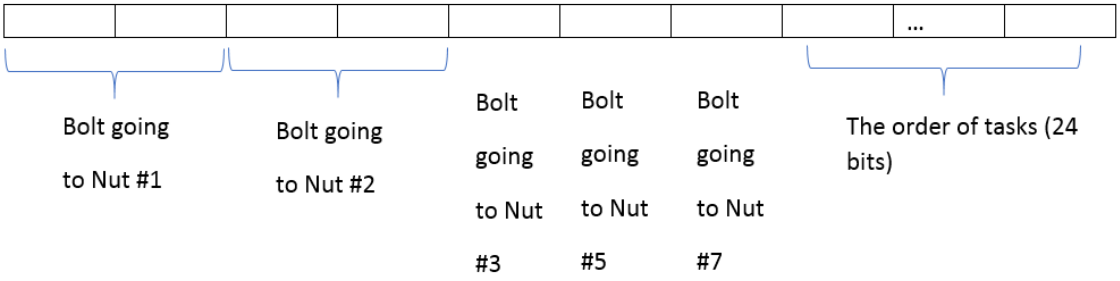


Figure 5. Encoding assembly sequence to a particle.

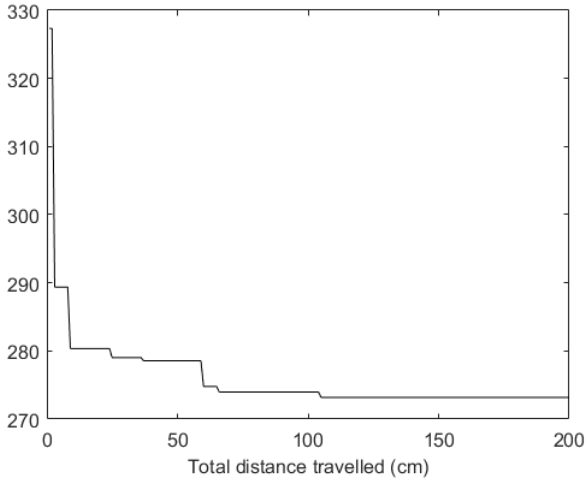


Figure 6. Evolution of total distance travelled by UR5 versus iterations.

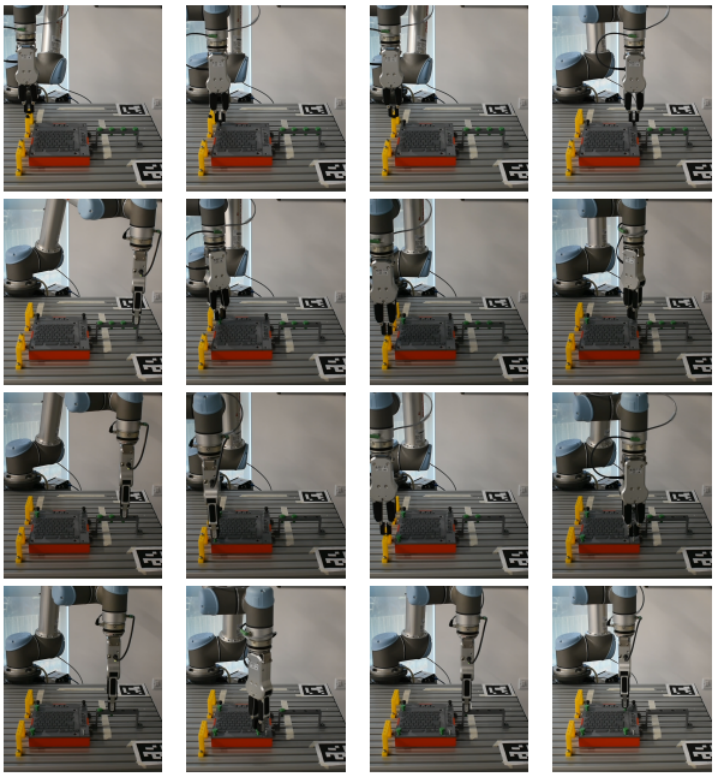


Figure 7. Movements obtained from optimization.

6.1. Motion Planning of Industrial Robots Inside a Digital Twin Created by V-REP Software

Proposed XOR BGSA with repository is used to find the optimal path for an industrial robot in its digital twin environment. The industrial robot used in this part is of a cobot type namely UR5. Cobot is a slang which refers to collaborative robot, generally accepted as any robot designed to operate in close proximity to human workers without need for a separating cage. The problem considered here is to find the shortest collision free path for the robot in 3D space while undergoing a pick and place task. Figure 7 illustrates the digital twin of the manufacturing environment simulated in V-REP software, where there exist a human worker working closely to the robot.

Simulations have been carried out using V-REP software, a user-friendly software, which works under various operating systems including Windows. This software V-REP is designed around a versatile architecture with different independently working functions. The options embedded in this

software can be easily enabled or disabled as required. The communication with this software is possible from ROS using rostopics as well as directly from Python or Matlab using respective API developed specially for this software. A wide range of robots including industrial robots from different companies including Universal Robots are available in this software. The driver included in this software for each of these robots possesses much of real robot features including stopping movement in the case of collision with an object. Real features of robot including maximum speed, maximum joint angles and maximum torque are available in the robot and may be modified at user desire making it replicate the real robot as much as possible. Another feature which exist under V-REP software is its ability to compute the inverse kinematic of robot and find the joint angles of the robot automatically which considerably facilitates robotic simulation.

The obstacles in this environment are possible obstacles in real manufacturing environments such as other machines, tools and devices. V-REP software is a robust software for simulating the physical behaviours of objects, including collision between them as well as forces such as gravity and friction that can all be included as part of the software environment. In this way, the user can easily define object positions and orientations while the V-REP software takes care of updating movement of the various other objects and collisions which may happen during motion. As the result is a game like 3D interface, examination of the movement can be easily achieved by the human expert. In this way the human expert can ascertain if there may be problems with the solution that are not specified in the programming. To perform the automated optimization, it is required that the robot movements be encoded in terms of an evolvable array which may then be translated to motions to evaluate cost function and make changes to optimize the problem. A similar approach has been previously done using Ant colony optimization in another software namely Gazebo [28]. However, the benefit of using V-REP versus Gazebo is that the former benefits from built-in inverse kinematic which makes it possible to perform the simulations in a timely manner. A comparative study has been performed previously in [29] which compared V-REP software and Gazebo. It is concluded that V-REP is a more user-friendly and intuitive software than Gazebo in which intelligent optimization can be carried out with more comfort.

While Figure 8 depicts the real robot, its digital twin developed in V-REP is depicted in Figure 9. As can be seen from these figures, digital twin matches with real robot in terms of dimensions and position of robot and obstacles, respectively. As can be seen from this figure, several objects as well as the UR5 robot exist in this digital twin. The obstacles are in the form of several cubic boxes which replicate the machines and devices which exist close to the robot.

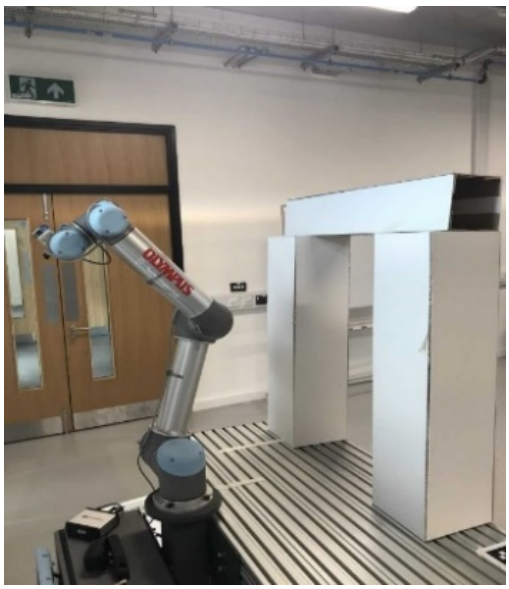


Figure 8. Setup of real UR5 robot in Nottingham Advanced Robotic Laboratory (NARLy).

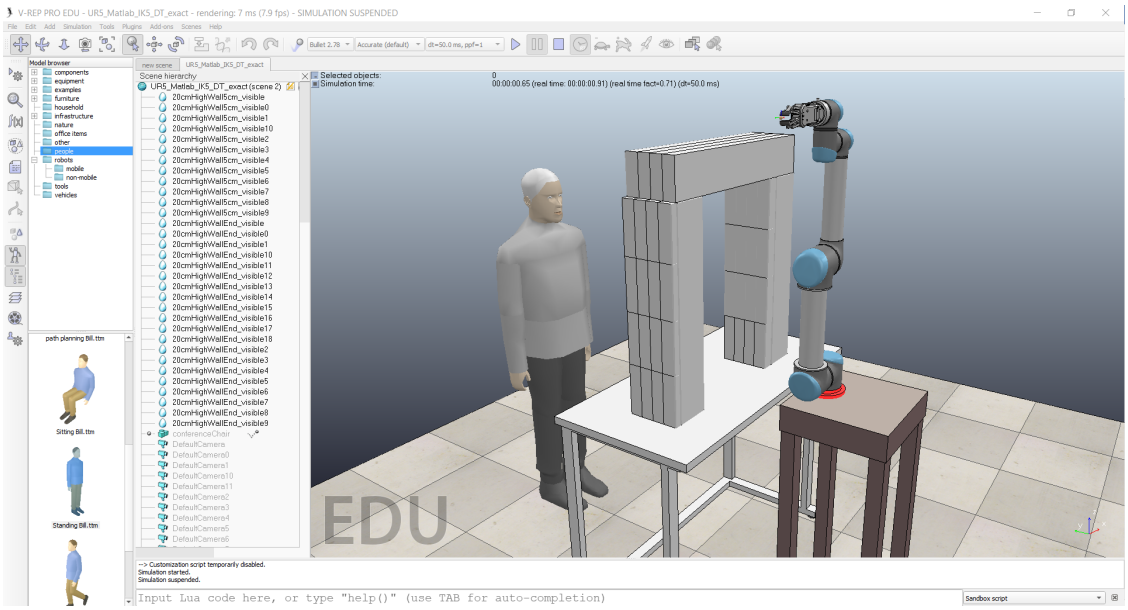


Figure 9. Digital twin of manufacturing environment developed in V-REP.

6.1.1. Encoding Movements

To optimize movements, they are encoded in terms of an evolvable array according to the proposed XOR BGSA with repository which is implemented under Matlab software. In this paper, it is assumed that the initial orientation of the robot remains the same during optimization and its position needs to be changed to perform the displacement from the start point to the end point. Considering three x, y and z dimensions and the fact that they can be increased and decreased, there would be six movements which can be represented by any integer value from the set $S = \{0, 1, \ldots, 5\}$. However, the proposed optimization algorithm would generate three bits for each movement. To avoid infeasible solutions, the set is considered to have all eight conditions and duplicate movements may be added. Moreover, during optimization duplicate coordinates may occur as a result of back and forth in the movements. To avoid such loops in robot movement, all coordinates are compared with each other and all intermediate movements which result in a loop are eliminated. The interpretation of movements are presented in Table 5.

Table 5. Interpretation of values of array generated by the proposed XOR BGSA.

Movements		**Rotations**	
Value	**Interpretation**	**Value**	**Interpretation**
0	$x = x + 0.025$	4	$z = z + 0.025$
1	$x = x - 0.025$	5	$z = z - 0.025$
2	$y = y + 0.025$	6	$x = x + 0.025$
3	$y = y - 0.025$	7	$x = x - 0.025$

6.1.2. Kinematic and Inverse Kinematic of UR5

The UR5 robot benefits from 6-degrees of freedom. Position and orientation of this robot and its end-effector in 3D work-space is represented by $x \in R^6$. While the first three dimensions of this vector present three translational variables, the next three dimensions represent its orientation. The joint angles, represented by $q = \begin{bmatrix} q_1 & q_2 & \ldots & q_6 \end{bmatrix}$, are solutions to the inverse kinematic problem.

The forward kinematic function is the function used to compute the position and orientation of the robot corresponding to its assigned joint angles.

$$x = \mathcal{FK}(\boldsymbol{q}) \tag{73}$$

The standard Denavit–Hartenberg (DH) convention to represent the forward kinematics of the robot is used in this paper. The DH parameters associated with UR5 are presented in Table 6. Using such convention, inverse kinematic of the robot can be found as follows:

$$x{=}T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 \tag{74}$$

where T_i^{i-1} presents the homogeneous transformation matrix which makes the transform from frame $i-1$ to frame i defined as follows:

$$T_i^{i-1} = \begin{bmatrix} c(q_i) & -s(q_i)c(\alpha_i) & s(q_i)s(\alpha_i) & r_i c(q_i) \\ s(q_i) & c(q_i)c(\alpha_i) & -c(q_i)s(\alpha_i) & r_i s(q_i) \\ 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{75}$$

where $s(.)$ and $c(.)$ represent the sinus and cosine of their arguments. The values obtained from the movement array need to be translated in terms of robot movements. To enable this a transformation matrix that represents position and orientation of robot is obtained based on work by [30,31]. The transformation matrix for frame 6 with respect to frame 0 is as follows:

$${}_0^6T\,(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) = \begin{bmatrix} {}_6^0R & {}_6^0P \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}_6^0\hat{X}_x & {}_6^0\hat{Y}_x & {}_6^0\hat{Z}_x & {}_6^0P_x \\ {}_6^0\hat{X}_y & {}_6^0\hat{Y}_y & {}_6^0\hat{Z}_y & {}_6^0P_y \\ {}_6^0\hat{X}_z & {}_6^0\hat{Y}_z & {}_6^0\hat{Z}_z & {}_6^0P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{76}$$

where ${}_6^0\hat{X}$, ${}_6^0\hat{Y}$ and ${}_6^0\hat{Z}$ are unit vectors defining the axis of frame 6 in relation to frame 0 and:

$${}_6^0R = R_x\,(a)R_y(b)R_z(c)$$

in which the rotational matrices are obtained as follows:

$$R_x{=}\begin{bmatrix} 1 & 0 & 0 \\ 0 & cosa & -sina \\ 0 & sina & cosa \end{bmatrix} \tag{77}$$

$$R_y{=}\begin{bmatrix} cosb & 0 & sinb \\ 0 & 1 & 0 \\ -\,sinb & 0 & cosb \end{bmatrix} \tag{78}$$

$$R_z{=}\begin{bmatrix} cosc & -sinc & 0 \\ sinc & cosc & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{79}$$

Inverse kinematics are then used to find joint angles of robot to move to a desired position given by proposed XOR BGSA. The angles of the robot are then obtained as [30,31]:

$$\begin{aligned}
\theta_1 &= atan2\left({}^{0}_{5y}P, {}^{0}_{5x}P\right) + \text{acos}\left(\frac{d_4}{\sqrt{{}^{0}_{5x}P^2 + {}^{0}_{5y}P^2}}\right) + \frac{\pi}{2} \\
\theta_2 &= atan2\left(-{}^{1}_{4z}P, -{}^{1}_{4x}P\right) - asin\left(\frac{-a_3\, sin\theta_3}{|{}^{1}_{4xz}P|}\right) \\
\theta_3 &= \pm acos\left(\frac{|{}^{1}_{4xz}P|^2 - {a_2}^2 - {a_3}^2}{2a_2a_3}\right) \\
\theta_4 &= atan2\left({}^{3}_{4y}\hat{X}, {}^{3}_{4x}\hat{X}\right) \\
\theta_5 &= \pm acos\left(\frac{{}^{0}_{6x}Psin\theta_1 - {}^{0}_{6y}Pcos\theta_1 - d_4}{d_6}\right) \\
\theta_6 &= atan2\left(M_1, M_2\right)
\end{aligned}$$

where:

$$M_1 = \frac{-{}^{6}_{0y}\hat{X}sin\theta_1 + {}^{6}_{0y}\hat{Y}cos\theta_1}{sin\theta_5} \tag{80}$$

$$M_2 = \frac{{}^{6}_{0y}\hat{X}sin\theta_1 - {}^{6}_{0y}\hat{Y}cos\theta_1}{sin\theta_5} \tag{81}$$

Table 6. Denavit–Hartenberg (DH) parameters representing the forward kinematics.

i	d_i(mm)	θ	a_i(mm)	α_i(rad)
0	–	–	0	0
1	*89.16*	q_1	0	$\frac{\pi}{2}$
2	*0*	q_2	−425	0
3	*0*	q_3	−392.25	0
4	*109.15*	q_4	0	$\frac{\pi}{2}$
5	*94.65*	q_5	0	$-\frac{\pi}{2}$
6	*82.3*	q_6	0	0

6.1.3. Communication between Matlab and V-REP

Communication between Matlab and V-REP is done using API developed by Coppelia robotics. Figure 10 demonstrates the overall communication requirement between Matlab and V-REP. As can be seen from this figure, proposed XOR BGSA with repository generates the path in terms of a binary sequence. This binary sequence is then translated to real movement with required resolution. Such movements are transmitted to V-REP in terms of positions and orientations of robot which is translated to joint angles of robot using built in inverse kinematic software. When a collision occurs, the target position of robot needs to be modified to the last possible point before collision in the system. After performing simulation, the results including total simulation time and distances are reported back as the real-world perception of the binary code in terms of a cost function. After all particles are evaluated, the proposed XOR BGSA iterates for another iteration to generate the next solutions.

The parameters x_i, y_i, z_i, a_i, b_i and c_i are the position and orientation of robot after every movement. Since obstacles are static objects in V-REP, in case of a collision, the robot stops, and a collision flag is returned. Furthermore, the desired position given to the robot is not met because of collision. Hence,

the difference between desired and obtained position of robot is a measure of collision. Such position difference can be used to penalize the cost function. The increment in the position suggested by the proposed XOR BGSA would be the actual position of robot plus the decoded increment from Table 5.

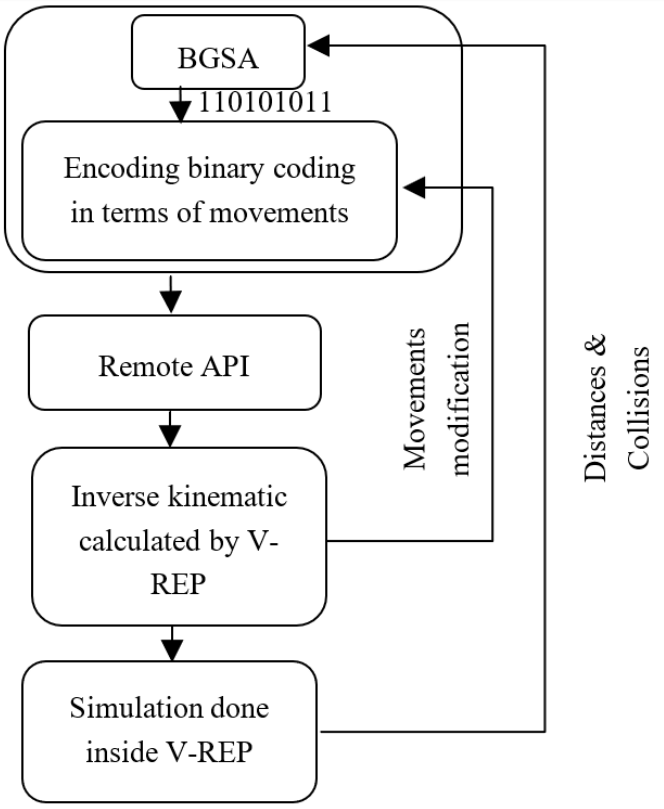


Figure 10. Communication between Matlab and V-REP.

6.1.4. Cost Function to Be Optimized

As the goal of the optimization is to obtain a desired position and orientation while ensuring that no collision happens, total distance between the desired position and the actual position of the robot is as follows:

$$D_T = \sum_i (x_i - x_d)^2 + (y_i - y_d)^2 + (z_i - z_d)^2 \quad (82)$$

where x_d, y_d and z_d are coordinates of P_D the desired position of the robot. Other than distance to the desired position, total time is used as another term in the cost function.

$$f(x) = K_D D_T + K_T T + D_C \quad (83)$$

where T represents the total time, D_C represents the distance between the actual position of robot and its desired position in every single movement due to collision with obstacles. The parameters K_D and K_T are two gains used for optimization which are selected as equal to 10,000 and 1, respectively.

For each individual particle, after performing the movements given by the particle, P_D is given as the final position of the robot. Meanwhile if there exist no obstacles between the last position obtained from the proposed optimization algorithm and P_D, robot moves easily to the final position. Otherwise, obstacle stops robot from movement. It is further assumed that if the end effector is close to the desired point robot does not need to continue with the movements given by the proposed XOR BGSA and total time is considered to this point. When end effector is within obstacle free range of the desired position, the other movements given by the proposed XOR BGSA are bypassed and the desired position is used as the next step for robot.

6.1.5. Simulation Results

The proposed XOR BGSA with repository is iterated for 30 times in Matlab software connected to V-REP through API using 30 particles. The evolution of cost function during optimization is depicted in Figure 11 on a logarithmic scale. As can be seen from this figure, the best cost function value among the 30 particles starts from a high value of 3446 due to collisions and is reduced during optimization to 6.77, which is a collision free, time optimal solution to the problem which hits the desired point. This indicates that for the final collision free solution, the total time is equal to 6.77 s.

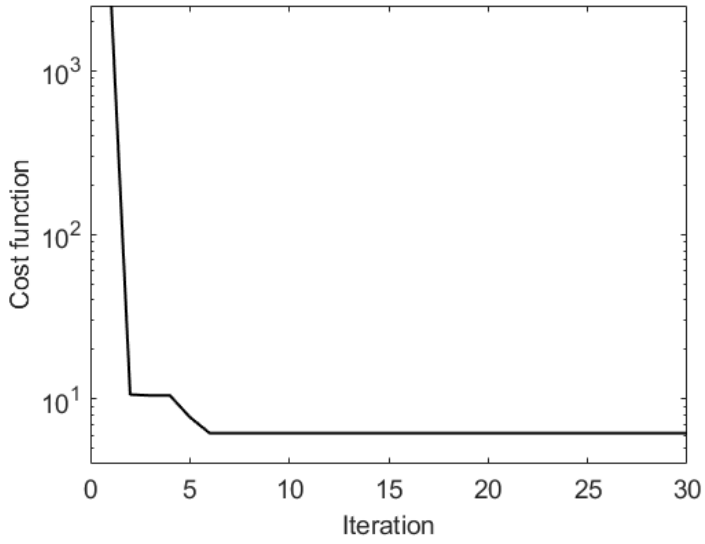


Figure 11. Cost function values associated with UR5 during path planning application.

The resulting movements are depicted with a red line in Figure 12, which can be inspected by human expert before final implementation on a real robot.

The 3D environment provided by simulation software provides complete visual demonstration of movements, which makes it is easy to perform the inspection and ensure suitability of the solution to real-world implementation before that happens. This can be an important safety feature within any working environment and such linking between the human and automated elements of production is a key factor in Industry 4.0 practice.

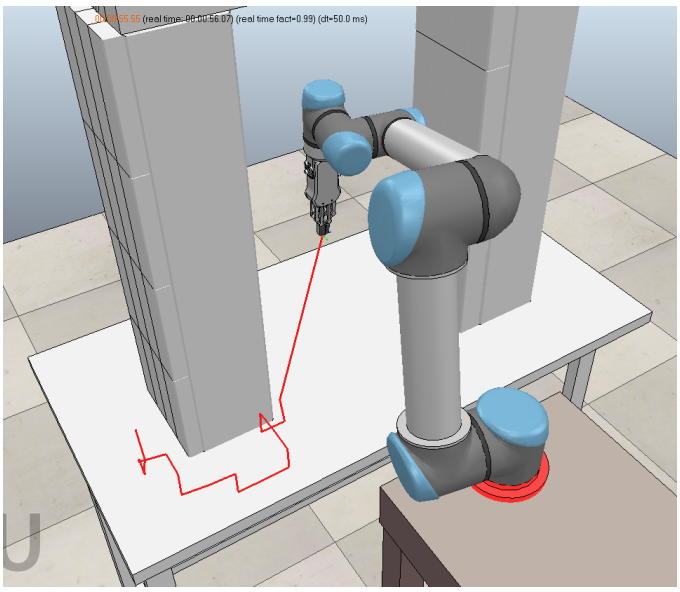


Figure 12. Optimal path obtained after performing optimization.

7. Conclusions

In this paper, an analysis of BGSA has been performed on mathematical update formulation for its acceleration term. Using such analysis, we show the fact that the existing acceleration and velocity term is not behaving as expected under certain conditions, and therefore conclude that it is required to modify the acceleration term of BGSA. We then propose the use of an XOR BGSA which benefits from repository. Analysis of this algorithm illustrates that the behaviour of acceleration and velocity terms are as expected. A repository is added to maintain best particle and provide a guideline for the movements of other particles. Complete statistical analysis is provided which demonstrate how the expected value of particles in each dimension converge to the particle selected from repository using the proposed version of XOR BGSA. To test the efficacy of the proposed optimization algorithm, a number of simulation studies are performed against different uni-modal and multimodal optimization problems. Simulation results indicate that the proposed modification outperforms other binary optimization algorithms namely BGSA [16], IBPSO [17], BPSO [15], BGWO [18] and BDA [19]. The proposed algorithm is used to solve a constrained maximization problem namely multi-knapsack problem. It is observed that the proposed algorithm outperforms BGSA [16], IBPSO [17], BPSO [15], BGWO [18] and BDA [19] for this optimization problem as well. Not only the mean value of results obtained during 30 times of optimization is better than other five optimization algorithms, but also the proposed algorithm benefits from a fast convergence rate. Moreover, repeatability of the proposed algorithm is better in most cases than in other investigated algorithms.

The conclusion made from the first part of simulation is that the proposed XOR BGSA with repository is an effective optimization algorithm to deal with binary optimization problems. Motivated by such results the algorithm is used for task planning of robot performing an assembly task.

Industry 4.0 emerges the need for robust machine learning and artificial intelligence approaches to be used on the factory floor to determine responses to production changes resulting from continuous design changes. Assembly task planning and path planning in an industrial environment as two frequent tasks are considered. The assembly task planning problem scenario considered in this paper is to find the optimum order of bolts to be fastened to minimize the total distance travelled by robot. Using the proposed optimization algorithm the distance travelled by robot decreased by 16.7%. This is a practical application of the proposed optimization algorithm in flexible manufacturing. In addition, the proposed XOR BGSA with repository is utilized to assist in path planning in a manufacturing environment. Such application would allow the robot to find time optimal path while avoiding obstacles and moving to a desired position. This demonstrated the ability to successfully perform such an operation, while enabling human feedback on results for a final implementation of a real robot.

Author Contributions: Conceptualization, M.A.K. and D.T.B.; Data curation: M.A.K.; methodology, M.A.K.; software, M.A.K.; validation, M.A.K., formal analysis, M.A.K.; investigation, M.A.K.; resources, M.A.K.; writing—original draft preparation, M.A.K.; R.B., D.T.B. Writing—review and editing, M.A.K.; R.B.; G.M.-A., D.T.B.; visualization, G.M.-A. supervision, D.T.B.; project administration, D.T.B.; funding acquisition, D.T.B. All authors have read and agreed to the published version of the manuscript.

Funding: 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.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Burnap, P.; Branson, D.; Murray-Rust, D.; Preston, J.; Richards, D.; Burnett, D.; Edwards, N.; Firth, R.; Gorkovenko, K.; Khanesar, M. Chatty factories: A vision for the future of product design and manufacture with IoT. In *Living in the Internet of Things (IoT 2019)*; IET: London, UK, 2019.
2. Zhang, H.; Liu, Q.; Chen, X.; Zhang, D.; Leng, J. A digital twin-based approach for designing and multi-objective optimization of hollow glass production line. *IEEE Access* **2017**, *5*, 26901–26911. [CrossRef]
3. Kongchuenjai, J.; Prombanpong, S. Binary Integer Programming to Solve the Process Planning Problem for Prismatic Mixed Parts. *Int. J. Mech. Prod. Eng. IJMPE* **2018**, *6*, 107–110.

4. Liang, J.; Wang, P.; Guo, L.; Qu, B.; Yue, C.; Yu, K.; Wang, Y. Multi-objective flow shop scheduling with limited buffers using hybrid self-adaptive differential evolution. *Memetic Comp.* **2019**, *11*, 407–422.
5. Özmen, Ö.; Batbat, T.; Özen, T.; Sinanoğlu, C.; Güven, A. Optimum assembly sequence planning system using discrete artificial bee colony algorithm. *Math. Probl. Eng.* **2018**, *2018*, 3407646 . [CrossRef]
6. Oesterle, J.; Amodeo, L. Efficient multi-objective optimization method for the mixed-model-line assembly line design problem. *Procedia CIRP* **2014**, *17*, 82–87. [CrossRef]
7. Rekiek, B.; Dolgui, A.; Delchambre, A.; Bratcu, A. State of art of optimization methods for assembly line design. *Annu. Rev. Control.* **2002**, *26*, 163–174.
8. Jin, Y.; Branke, J. Evolutionary optimization in uncertain environments-a survey. *IEEE Trans. Evol. Comput.* **2005**, *9*, 303–317. [CrossRef]
9. Engelbretch, A. *Fundamentals of Computational Swarm Intelligence*; John Wiley & Sons Ltd.: Hoboken, NJ, USA, 2005; pp. 5–129.
10. Biswas, A.; Mishra, K.; Tiwari, S.; Misra, A. Physics-inspired optimization algorithms: A survey. *J. Optim.* **2013**, *2013*, 438152. [CrossRef]
11. Genc, H.M.; Eksin, I.; Erol, O.K. Big bang-big crunch optimization algorithm with local directional moves. *Turk. J. Electr. Eng. Comput. Sci.* **2013**, *21*, 1359–1375.
12. Formato, R.A. Central force optimization. *Prog. Electromagn. Res.* **2007**, *77*, 425–491.
13. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
14. Rashedi, E.; Rashedi, E.; Nezamabadi-pour, H. A comprehensive survey on gravitational search algorithm. *Swarm Evol. Comput.* **2018**, *41*, 141–158. [CrossRef]
15. Kennedy, J.; Eberhart, R.C. A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997; IEEE: Piscataway, NJ, USA, 1997; Volume 5, pp. 4104–4108.
16. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. BGSA: Binary gravitational search algorithm. *Nat. Comput.* **2010**, *9*, 727–745. [CrossRef]
17. Yuan, X.; Nie, H.; Su, A.; Wang, L.; Yuan, Y. An improved binary particle swarm optimization for unit commitment problem. *Expert Syst. Appl.* **2009**, *36*, 8049–8055. [CrossRef]
18. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [CrossRef]
19. Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput. Appl.* **2016**, *27*, 1053–1073. [CrossRef]
20. Jamil, M.; Yang, X.S. A literature survey of benchmark functions for global optimisation problems. *Int. J. Math. Model. Numer. Optim.* **2013**, *4*, 150–194. [CrossRef]
21. Schumer, M.; Steiglitz, K. Adaptive step size random search. *IEEE Trans. Autom. Control.* **1968**, *13*, 270–276. [CrossRef]
22. Surjanovic, S.; Bingham, D. *Virtual Library of Simulation Experiments: Test Functions and Datasets*; Simon Fraser University: Burnaby, BC, Canada, 2013; Volume 13, p. 2015.
23. Surjanovic, D.B.S. Evolution and Optimum Seeking. 1995. Available online: https://www.sfu.ca/~ssurjano/optimization.html (accessed on 6 July 2020).
24. Silagadze, Z. Finding two-dimensional peaks. *Phys. Part. Nucl. Lett.* **2007**, *4*, 73–80. [CrossRef]
25. Bäck, T.; Schwefel, H.P. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.* **1993**, *1*, 1–23. [CrossRef]
26. Liu, J.; Mei, Y.; Li, X. An analysis of the inertia weight parameter for binary particle swarm optimization. *IEEE Trans. Evol. Comput.* **2015**, *20*, 666–681.
27. Alcácer, V.; Cruz-Machado, V. Scanning the industry 4.0: A literature review on technologies for manufacturing systems. *Eng. Sci. Technol. Int. J.* **2019**, *22*, 899–919.
28. Bansal, R.; Khanesar, M.A.; Branson, D. Ant Colony Optimization Algorithm for Industrial Robot Programming in a Digital Twin. In Proceedings of the 2019 25th International Conference on Automation and Computing (ICAC), Lancaster, UK, 5–7 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–5.
29. Nogueira, L. Comparative analysis between gazebo and v-rep robotic simulators. *Semin. Interno Cognicao Artif. SICA* **2014**. [CrossRef]

30. Hawkins, K.P. *Analytic Inverse Kinematics for the Universal Robots*; Georgia Institute of Technology: Atlanta, GA, USA, 2013.
31. Andersen, R.S. *Kinematics of a UR5*; Aalborg University: Aalborg, Denmark, 2018.