Appl. Math. J. Chinese Univ.
2021, 36(4): 611-630

Gompertz PSO variants for Knapsack and Multi-Knapsack Problems

Pinkey Chauhan* Millie Pant Kusum Deep

Abstract. Particle Swarm Optimization, a potential swarm intelligence heuristic, has been recognized as a global optimizer for solving various continuous as well as discrete optimization problems. Encourged by the performance of Gompertz PSO on a set of continuous problems, this works extends the application of Gompertz PSO for solving binary optimization problems. Moreover, a new chaotic variant of Gompertz PSO namely Chaotic Gompertz Binary Particle Swarm Optimization (CGBPSO) has also been proposed. The new variant is further analysed for solving binary optimization problems. The new chaotic variant embeds the notion of chaos into GBPSO in later stages of searching process to avoid stagnation phenomena. The efficiency of both the Binary PSO variants has been tested on different sets of Knapsack Problems (KPs): 0-1 Knapsack Problem (0-1 KP) and Multidimensional Knapsack Problems (MKP). The concluding remarks have made on the basis of detailed analysis of results, which comprises the comparison of results for Knapsack and Multidimensional Knapsack problems obtained using BPSO, GBPSO and CGBPSO.

§1 Introduction

Many of the real world problems such as scheduling, assignment, ordering of discrete elements, planning and selection etc. encountered in engineering and industrial fields are often formulated as combinatorial optimization problems. In general, the combinatorial problems are formulated as NP-hard problems which due to their complex nature require extra computational efforts to obtain the desired solution. The basic PSO was designed to solve optimization problems having continuous variables but later on the research has also been extended to the discrete or binary valued problem spaces. As an initiative, [7] proposed the first discrete version of PSO for binary problems. In the basic binary PSO, the search process remains same as in the continuous version but the position update rule turns to a binary number generator in the

Received: 2021-03-29. Revised: 2021-11-04.
MR Subject Classification: 68T20, 6840, 90C27, 97R40.
Keywords: Binary PSO, Knapsack Problems, Multi Knapsack Problems, Gompertz function, chaos.
Digital Object Identifier(DOI): https://doi.org/10.1007/s11766-021-4583-y.
*Corresponding author.

case of a binary PSO. The velocity is used as a probability threshold to determine whether x_{id}, the d^{th} component of x_i, should be evaluated as 0 or as 1. To get the required value of variables as 0 or 1, a mapping rule is need to be defined between each ν_{id} to probability in the range [0, 1]. The above need is fulfilled by utilizing a function called sigmoid function 'which generates values in the range [0,1]. The position updating rule for binary PSO is given by

$$x_{id}\left(t+1\right)=\begin{cases}1, \text{ if } U(0,1) < sigm(\nu_{id}(t))\\ 0, \text{ otherwise}\end{cases} \tag{1}$$

where

$$sigm\left(\nu_{id}\right)=\frac{1}{1+exp\left(-\nu_{id}\right)}$$

is sigmoid function, $U(0,1)$ is a quasi random number in the range $[0,1]$. It is obvious from equation (1) that xid will remain 0 if $sigm(\nu_{id}) = 0$ (for convenience the time scripts are dropped). This happens when either $\nu_{id} < -10$ or $\nu_{id} > 10$. To overcome this situation it is suggested to set $\nu_{id} \in [-4, 4]$ and to use velocity clamping with $V_{max} = 4$.

Literature Review: Several studies have proposed new amendments by integrating binary PSO with different concepts for solving binary optimization problems. In the variant, suggested by [11], rather than directly encoding bit strings within the particles, every particle hold on the small range of trigonometric coefficients (angle modulation) that was then run to get bit strings. An amalgamation of Binary PSO and continuous PSO with real encoding was introduced by [12] for solving unit commitment problem. The part of unit comment problem was handled by binary PSO and the other part of problem, economic load dispatch problem, was dealt with real PSO. [8] proposed a new velocity updating rule by defining it as the probability to change in particles state from its previous state to its complement value. It was observed that the new proposed model provided better interpretation of continuous PSO into discrete version as compared to older studies. Many of the real world problems are modeled as combinatorial optimization problems which attracted the attention of researchers towards the timely modifications in Binary PSO. [6] studied and analyzed the behavior of Binary PSO with the motive of paving a path for creating a faster and simpler discrete version by making changes in representation. They disassemble the algorithm qualitatively, breaking it into its essential additives and then reinterpreting them in different approaches as new easy and reliable algorithm namely Essential Particle Swarm. [5] inserted the concept of mutation to propose a new variant namely generalized binary PSO which exploits the benefits of sigmoid as well as linear function and a logical mutation operator was also proposed to escape local optima. The concept of chaos is not new as the randomness produced by chaotic maps was observed as a good alternative to introduce randomness in the algorithm whenever required. So far, many studies have reported the implementation of chaotic maps into PSO for solving problems from various fields such as science, engineering and industries. These studies shows that the concept of chaos is helping PSO algorithm in different ways to enhance its performance. The robustness of PSO keeps motivating researchers at times to explore its applications in dealing the complexity involved in feature selection problems. [10] embedded logistic map and tent map for selecting inertia weight to develop a new chaotic BPSO variant and solving feature selection problem successfully. [14] presented a new Chaotic PSO by introducing mutation operator and chaotic sequences for overcoming the problem of stagnation when the algorithm explores local search valleys and got stuck there only causing faster convergence to a false or local optima. [2] developed a new chaotic PSO MOPSO for solving the optimization of pavement maintenance management. They had considered multiple objectives as minimizing the maintenance cost while maximizing the pavement performance. The study claimed to be the first method who combined crossover operation with velocity rule and embedded multi objective PSO algorithm with position updating rule. A study by [15] investigates chaotic PSO namely Chaotic Restart Binary Particle Swarm PSO (ChResBPSO) to the problem of feature selection. This study suggested the addition of new particles using prior information about global best particle to avoid the stagnation phenomena. Further, the best chaotic map was found by replacing the main parameters of PSO from various chaotic systems. The problem of biometric identification system was addressed by [21]. The authors incorporated chaos in binary PSO and developed a new modified chaotic binary particle swarm optimization for selecting features.

The present study analyses the efficiency of Gompertz PSO for solving binary (0-1) optimization Problems. Gompertz PSO was proposed by [4] with a new concept of replacing the regular sigmoid function by a more qualitative function namely Gompertz function. The diversified nature of Gompertz function and embodiment in PSO fascinated researchers for application of Gompertz PSO for dealing problems arising in different fields e.g. process plans scheduling [3], Automobile engineering [1] etc. Moreover, the present study also proposes a new chaotic variant of Gompertz by embedding the idea of chaos at later stages of search when the algorithm works on solution refinement and thus the chance of being stucked in the local vallies becomes higher. The organization of rest of the paper is as: Section 2 presents Gompertz PSO and Chaotic Gompertz PSO. The Numerical studies are presented in Section 3. Results are discussed and analyzed in Section 4. The paper ends with concluding remarks in Section 5.

§2 PROPOSED BINARY PSO VARIANTS

This section provides a brief introduction to Gompertz PSO and a detailed description of newly proposed chaotic variant namely Chaotic Gompertz PSO.

2.1 Gompertz Binary Particle Swarm Optimization (GBPSO)

The Gompertz PSO was developed by [4] to overcome the drawbacks of basic binary PSO. The main drawback of BPSO is the shape of the sigmoid function trajectory which serves the purpose of a probability generating function for deciding the bit change from 0 to 1 or vice-versa. Due to the S-shape of sigmoid function, for some larger velocity (ν_{id}) values the changing probability decreases for one bit and increases for the other bit. The biased probabilities result in the generation of new particles with almost same pattern. This results in the decrease of population diversity and provides low exploration for bigger velocity values. Thus for a more

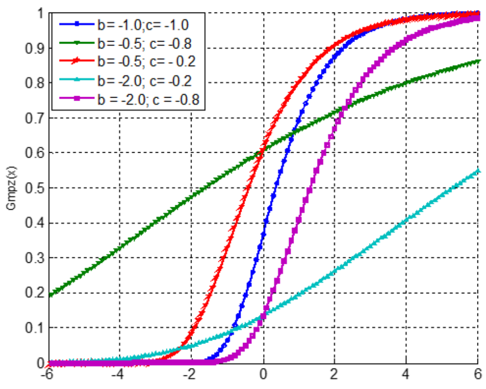


Figure 1. Behaviour of Gompertz function with $a = 1$ and varying parameters b and c.

diversified search, there is a scope for improving the searching capability of BPSO. Gompertz function is a special type of function that can be made to behave as a sigmoid function as well as a linear function by controlling its parameters. The basic sigmoid function in BPSO leads to explore the search space thoroughly in initial stages, but also has the drawback of being stagnated in later stages. This is because as the search progresses, there is a decrease in diversity which restricts its capability of finding efficient new solutions. Furthermore, the linear function provides fair chances to search the space in later stages and producing a more diversified population. It is observed that we can control the parameters of Gompertz function to exploit the benefits of sigmoid as well as linear function as per the requirement of algorithm. The basic expression of Gompertz function is as:

$$Gmpz(x) = a * exp(b * exp(cx)) \tag{2}$$

where, a : upper asymptote; b: (controls) the displacement of x; c: controls the growth rate (x scaling) or steepness of graph; a: upper asymptote. The parameters b and c are negative numbers and the a=1 (to produce values in range [0,1]). The parameters b and c are variables and fine tuned to exploit the attributes of sigmoid, semi-sigmoid and linear functions. The change in behaviour of Gompertz function with the variation of parameters b, c with $a = 1$ is demonstrated in Figure 1. *Variation of Parameter b*: In BPSO, it is observed that the probability for changing a bit from 0 to 1 or vice-versa, at the velocities near zero is 0.5. While in Gompertz function this probability can be increased or decreased in the range (0, 1) by manipulating parameter b appropriately. It is observed that by changing b while keeping other parameters fixed as: a=1 and c= -1, the scale of the function trajectories changes, which represent the change in probability of generating binary numbers 0, 1. The parameter b could be increased in the [-2.0. -0.2] or decreased in the range [-0.2, -2.0]. The value of parameter b depends on the chosen problem. The following relation controls the variation of parameter b.

$$b(t) = -\left\{0.2 + (2.0 - 0.2) * \frac{\text{Current iteration}}{\text{Maximum no of iterations}}\right\} \tag{3}$$

Variation of Parameter c: The suggested variation of parameter c is based on the effect of c on the steepness of the function *i.e.*, the shape of function changes from sigmoid to linear as c varies. The value of parameter c is changed in the range $[-0.2, -1.5]$ using the relation given below:

$$c(t) = -\left\{0.2 + (1.5 - 0.2) * \frac{\text{Current iteration}}{\text{Maximum no of iterations}}\right\} \tag{4}$$

2.2 CHAOTIC GOMPERTZ BINARY PSO (CGBPSO)

In the present study, GBPSO is further modified by exploiting the properties of chaotic dynamic systems into GBPSO. GBPSO has two parameters to handle with, which sometimes slows down the searching process and decreases the convergence rate. In order to maintain a good convergence rate, chaos is embedded into GBPSO and the resultant variant is called CGBPSO.

2.2.1 Chaos Generation

Chaotic maps are known for their unpredictable behavior and ergodic properties which can disturb any given system by producing chaotic randomness. In general terms Chaos refers to a state of disorder. A dynamical system is said to exhibit chaotic characteristics if it is sensitive to its initial conditions and parameters. Sensitivity to the initial condition signifies that each point in such a system is closely approximated by other points. Thus, an arbitrarily small perturbation in the initial conditions may lead to significantly different future behaviour. This entails that the nature of chaos is apparently random and unpredictable. Many chaotic maps have been described in literature so far, but here in the present study Logistic map is selected to produce chaos in particle positions. In the present work, different chaotic maps like logistic, tent, Henon and piecewise linear map were analysed and it was observed that logistic map gave best results for dealing with binary variables considered in this chapter.

Logistic Map: The Logistic map can be expressed as:

$$X_{n+1} = \alpha X_n * (1 - X_n) \tag{5}$$

where, $X_n \in (0, 1)$ and α is a positive number which is also known as bifurcation parameter. The system displays different behaviors such as stable, periodic, non-periodic and chaotic for distinct values of α. The chaotic behavior of logistic map for $\alpha = 4$ is of our interest.

2.2.2 Chaos Embedded Binary PSO

The chaos generated using Logistic map is embedded in GBPSO after certain number of iterations have been elapsed. The rationale of applying chaos is to prevent swarm particles from getting entrapped in a local optimizer and accelerating the convergence rate as search progresses. It is expected that perturbing the particles with the help of chaos after a certain number of iterations, will prevent the particles from getting stuck in some region (probably a local optimizer). In CGBPSO, chaos is introduced after 75% of the max number of iterations have elapsed. The modified searching procedure of CGBPSO is described Algorithm 1.

Algorithm 1 Chaotic Gompertz Binary PSO

```
BEGIN
Create and Initialize a $D$-dimensional swarm, $S$
for $t = 1$ to the maximum bound on the number of iterations do
    for $i = 1$ to $S$ do
        for $d = 1$ to $D$ do
            Apply the velocity using basic velocity update equation
            Update Position using equation as:
            Calculate $b$ iteration wise using equation (3.2)
            Calculate $c$ iteration wise using equation (3.3)
            Calculate $Gmpz(\nu_{id})$ using equation (3.1)
            if $t \leq 75\%$ of Max iter then
                Generate Binary positions as:
                $$x_{id}(t+1) = \begin{cases} 1, & \text{if } U(0,1) < Gmpz(\nu_{id}(t)) \\ 0, & \text{otherwise} \end{cases}$$
            else if no improvement is observed in global best position for 10% of left iterations then
                Create Chaos using equation (3.4)
                Add Perturbation to particles positions as
                Perturbation = $Gmpz(\nu_{id})$ * Chaos
                $$x_{id}(t+1) = \begin{cases} 1, & \text{if } U(0,1) < Perturbation \\ 0, & \text{otherwise} \end{cases}$$
            end if
        end for
        Compute fitness of updated position;
        Update historical information for $P_i$ and $P_g$;
    end for
    Terminate if $P_g$ meets problem requirements;
end for
END
```

§3 TESTING WITH BENCHMARK PROBLEMS

The proposed variants are validated on a set of 0-1 knapsack and multidimensional knapsack problems. Parameter settings taken for the algorithms and a description of knapsack problems are given in detail in the following subsections.

3.1 Performance Evaluation Criteria

Binary benchmarks problems from literature are considered and solved to analyze the performance of proposed variants. For all the considered problems the global optima are known. To test the efficiency of PSO variants for solving binary problems, the termination criteria is set to maximum number of iterations or when the known optimum is within 99% of accuracy, whichever occurs earlier. The maximum number of iterations allowed is 500 for considered continuous problems while for multi knapsack problems it varies from 500 to 1000 based on dimension of problem, for problems having dimension from up to 50, maximum iterations are taken to be 500 and for higher dimensions it is set to 1000. A run in which the algorithm finds a solution satisfying $f_{min} - f_{opt} \leq 0.01$, where f_{min} is the best solution found when the algorithm terminates and f_{opt} is the known global minimum of the problem, is considered to be successful. For each method and problem, results are recorded for following criteria:

Success Rate (SR) $= \frac{\text{Number of Suuccessful runs}}{\text{Total number of runs}} \times 100$; Average number of function evaluation (AFE)

Average Error (AE) $= \frac{\sum_n (f_{min} - f_{opt})}{n}$, Where, n is the total number of runs ; Average Computational time over n runs; Standard Deviation of Error over n runs; Minimum error over n runs.

Parameter Settings: The algorithm parameters are set as: The Swarm size S is set to 5*number of variables; The cognitive and social scaling parameters $c1$ and $c2$ are 2.8 and 1.3, respectively; Maximum velocity V_{max} is set equal to 4; Termination criteria: The algorithm will be terminated if it runs upto maximum number of iterations (1000) or minimum error is less than 0.01; The total number of Runs is set to 100. Further, the performance evaluation criteria include Average error, Minimum error, Standard deviation and Average function evaluations. The results are recorded in tabular form and also presented using box plots.

3.2 Standard Binary (0-1) Problems: Knapsack Problems

Knapsack problems (KP) belongs to the category of NP-Hard problems in the field of combinatorial optimization. These combinatorial problems have also been studied extensively due to the complexity and its immediate applications in various disciplines. KP's family needs to choose a subset from the given items that maximizes the corresponding profit while the total cost/weight of chosen items doesnt exceed the capacity of given Knapsack (s). Knapsack Problems are divided into different types based on the distribution of items and available knapsack (s).

- **0-1 Knapsack Problem:** In this category of problems, we can not choose any item more than one time. So if we choose an item then the variable will take value 1 otherwise 0.
- **Bounded Knapsack Problem:**In this category of problems, we can choose any item more than one time. So the corresponding variable will take an integer value based on the number of times an item is chosen.

- **Multiple Choice Knapsack Problem:** In this category of problems, items are subdivided into some finite number of classes and exactly one item must be taken from each class.
- **Multidimensional Knapsack Problem:** In this category of problems, there are more than one knapsacks with defined capacities and each of the knapsack has to be filled simultaneously.

3.2.1 0-1 Knapsack Problem

If there are n items, with each item i having an integer profit p_i and an integer weight w_i. The optimization problem is to choose the number of items such that the total profit is maximized and the total weight does is less than or equal to a given capacity C. The mathematical formulation of problems is given by

$$Maximize\ g(x) = \sum_{j=1}^{n} p_j x_j \tag{6}$$

subject to $\sum_{j=1}^{n} w_j x_j \leq C$; $\quad x_j \in \{0, 1\}\ j = 1, 2, ...n$

where, x_j are the binary decision variables which indicate whether item j is selected for filling the knapsack or not. Here it is assumed that all profits and weights are positive and all weights are smaller than the capacity C so each item fits into the knapsack, and the total weight of the items exceeds the capacity C to ensure a nontrivial problem.

3.2.2 Multidimensional Knapsack Problem

An important generalization of 0-1 knapsack problem is NP-Hard Multidimensional Knapsack Problem. It comprises of choosing a subset of the given articles (or items) so that the total profit (value) of the chosen objects is maximized while a set of imposed limitations are fulfilled. The mathematical model is as follows:

$$Maximize\ f(x) = \sum_{j=1}^{n} p_j x_j \tag{7}$$

subject to $\sum_{j=1}^{n} w_{j,k} x_j \leq C_k$; $\forall\ k = 1, 2, ..., m$; $\quad w_{j,k} \geq 0,\ C_k \geq 0,\ \forall\ k = 1, 2, ..., m$; $x_j \in \{0, 1\}\ j = 1, 2, ...n$

where n is the number of objects, m is the number of knapsack constraints with capacities C_k, p_j represents the benefit of the object j in the knapsack, x_j is a binary variable that indicates $x_j = 1$ if the object j has been stored in the knapsack and $x_j = 0$, if it remains out, and $w_{j,k}$ represents the entries of the knapsack's constraints matrix.

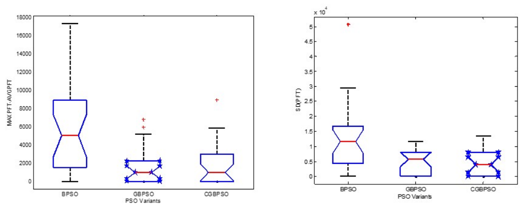
§4 RESULTS AND DISCUSSIONS

In order to verify the feasibility and effectiveness of GBPSO and CGBPSO variants for solving complex binary valued problems, both the variants are tested on standard set of 0-1 Knapsack and Multidimensional Knapsack Problems. The binary instances are picked from publicly available library (Beasley, OR Library) and other online sources. The results recorded for considered binary problem set are analyzed and discussed in this section.

4.1 Results for 0-1 Knapsack Problems

First set of 0-1 KP consists of 25 instances taken from http://www.math.mtu.edu/ kreher/cages/Data.html,. The instances are also used by [17]. In these instances, the number of items range between 8 to 24. All these instances are randomly generated so the optimum solution is not known in advance. Due to unknown global optima, the results of BPSO, GBPSO and CGBPSO are compared on the basis of maximum profit over 100 runs (MAXPFT) and average profit (AVPFT) of 100 runs. The total weight gap (in percentage) in case of maximum profit is $WHTGP = \left[\frac{\text{weight limit - weight when the maximum profit is reported}}{\text{weight limit}}\right] \times 100$
The second set of KP instances is taken from http://www.cs.colostate.edu/ cs575dl /assignments/assignment5.html and [13] with number of items ranging between 10 to 500. The above considered instances are not generated randomly so the optimum solution for these instances is known. The performance analysis of considered algorithms has been done on various performance aspects. The Average Error (AE) and the Standard deviation (SD) are computed for feasible solutions only. Numerical results recorded for considered algorithms are summarized in Table 1. It is observed that the performance in terms of finding Maximum profit (MAXPFT) over 100 runs for all the algorithms is same. Further, the maximum profit results leads to same value of the WHTGP for all the algorithms. Through AVPFT, a remarkable difference between the two algorithms can be observed. In all the instances AVPFT of GBPSO and CGBPSO is greater than that of BPSO. A statistical view in terms of box plots as shown in Figure 2(a) is more appropriate to see the improvement of GBPSO and CGBPSO over BPSO. In Figure 2(a), box plots of BPSO, GBPSO and CGBPSO are plotted for difference (MAXPFT -AVPFT). The box plot of GBPSO and CGBPSO are close to zero and has less height than that of BPSO. This shows that the minimum value, median, maximum value, quartiles and standard deviation of the difference discussed above are least for GBPSO and CGBPSO as compared to BPSO, which shows the superiority of GBPSO and CGBPSO over BPSO. But when it comes to select the best performer among the three versions the analysis shows that GBPSO performs better than CGBPSO and BPSO. The analysis of SD (PFT) for GBPSO, CGBPSO and BPSO is also performed via box plots shown in Figure 2(b). The figure shows that box plots of GBPSO and CGBPSO has less height than that of BPSO which predicts that both the proposed methods are superior over standard version. The order of performance is as: CGBPSO > GBPSO > BPSO.

The numerical results for the problem set II are given in Table 2. It is evident that GBPSO and CGBPSO has higher success rate for all considered instances which shows that CGBPSO and GBPSO are more reliable than BPSO. While averagely GBPSO performs better than CGBPSO and BPSO on reliability aspects. The average number of function evaluations (AFE) are also least for GBPSO and CGBPSO for considered instances except instance 2. The results



(a) (b)

Figure 2. Box plots of BPSO, GBPSO and CGBPSO for problem set I of 0-1 Knapsack problems.

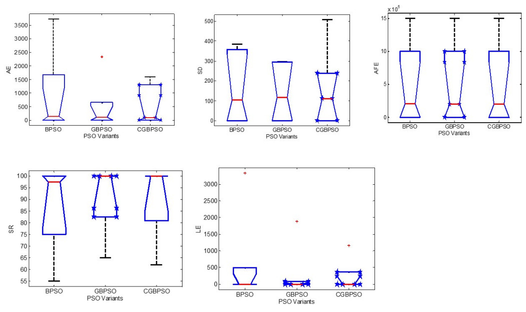


Figure 3. Box plots of BPSO, GBPSO and CGBPSO for problem set II of 0-1 Knapsack problems.

show that GBPSO performs comparatively better than CGBPSO and BPSO for these instances. It is also observed that CGBPSO performs better over GBPSO and BPSO from the standpoint of LE, AE and SD, which reflects the higher accuracy of CGBPSO than GBPSO and BPSO. A comparative analysis of BPSO, GBPSO and CGBPSO can be seen at a glance using box plots. The box plots of BPSO, GBPSO and CGBPSO for all comparison criteria are shown in Figure 3, which predict that GBPSO and CGBPSO are more efficient than BPSO in almost all criteria considered here for 0-1 KP.

4.2 Results for Multidimensional Knapsack Problems

All the three binary variants, BPSO, GBPSO and CGBPSO are tested on two groups of MKP benchmarks selected from [21]. The first group corresponds to series sento [18] and weing [17]. This group contains 10 instances and the number of items are ranging between 28 to 105. The second group corresponds to weish [20] contains 38 instances and the number of items are ranging between 20 to 90. Since MKP instances whose optimal solution is known are considered here, therefore the comparison among BPSO, GBPSO and CGBPSO is carried out on the basis of considered performance aspects. The experimental results for first set of MKP instances obtained using BPSO, GBPSO and CGBPSO are shown in Table 3. It is clear from the results that GBPSO and CGBPSO are more reliable than BPSO as the success rate

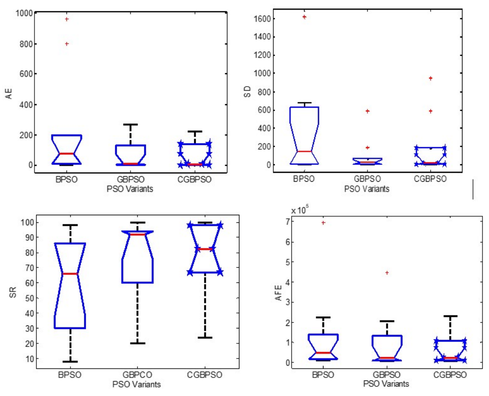


Figure 4. Box plots of BPSO, GBPSO and CGBPSO for problem set I of Multidimensional Knapsack problems.

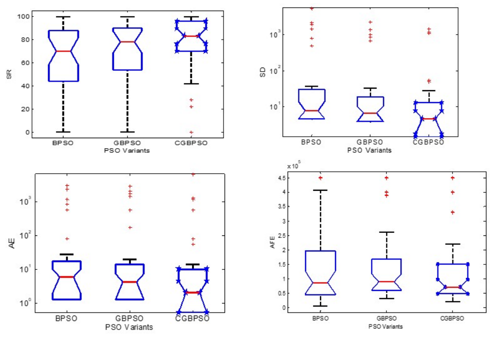


Figure 5. Box plots of BPSO, GBPSO and CGBPSO for problem set II of Multidimensional Knapsack problems.

(SR) for both the methods are higher than that of BPSO for all the 10 instances. The further analysis shows that CGBPSO performs better over GBPSO on 7 instances which indicate the reliability of CGBPSO over GBPSO and BPSO. The average number of function evaluations (AFE) for CGBPSO are also less than that of BPSO and GBPSO for 9 instances indicating capability of CGBPSO of giving solution faster. The results indicate that CGBPSO is able to provide better quality solution for MKP as AE, LE and SD is least for CGBPSO in most of the instances. An overall strength of CGBPSO with respect to GBPSO and BPSO can be seen from box plots shown in Figure 4. The box plots of BPSO, GBPSO and CGBPSO for SR, AFE, AE and SD are shown in Figure 4.

Numerical results for the second problem set of MKP problems are shown in Table 4. Due to the complexity of MKP problems, some runs are successful while others provide infeasible solution. The number of infeasible solutions are also recorded and depicted in Table 4. Further, Figure 5 shows the box plot for problem set II of MKP. It is obvious from Table 4 and Figure 5 that CGBPSO is better in reliability, accuracy and computational cost than BPSO and GBPSO for considered MKP problems.

§5 CONCLUSION AND FUTURE SCOPE

In this study, the performance of two new PSO variants namely GBPSO and CGBPSO is analyzed for solving knapsack problems and the results are compared with Basic Binary PSO. The conclusions that can be made are as follows:

1. The simulation results show that the use of Gompertz function is a useful strategy for generating random numbers. This strategy can be used in any evolutionary algorithm dealing with binary variables.
2. It is also observed that by including the component of chaos, the performance of the proposed scheme can be further improved.
3. It can be seen from the numerical results that CGBPSO outperforms GBPSO and BPSO, while GBPSO dominates BPSO in terms of reliability, cost and quality of solution.

Future Scope: In the present study, logistic map has been used to introduce randomness in the algorithm but other chaotic maps could be used and analyzed for solving knapsack /multi knapsack/ binary/combinatorial optimization problems. Apart from that we could also use chaotic maps to generate initial population or handling inertia weight. We could also analyze the performance of proposed variants for solving more complex real world problems.

Table 1. Comparative results of Problem Set I of 0-1 Knapsack problems.

Sr. No.	Method	AVPFT	SD(PFT)	MAXPFT	WHTGP
1	BPSO	3923910.4	2398.54035	3924400	1.990950
	GBPSO	3924400	0.0	3924400	1.990950
	CGBPSO	3924400	0.0	3924400	1.990950
2	BPSO	3813669	0.0	3813669	0.718926
	GBPSO	3813669	0.0	3813669	0.718926
	CGBPSO	3813669	0.0	3813669	0.718926
3	BPSO	3330096.12	50734.457	3347452	0.654025
	GBPSO	3347452	0.0	3347452	0.654025
	CGBPSO	3347452	0.0	3347452	0.654025
4	BPSO	4187707	0.0	4187707	2.9984
	GBPSO	4187707	0.0	4187707	2.9984
	CGBPSO	4187707	0.0	4187707	2.9984
5	BPSO	4945710.8	29532.688	4955555	2.050978
	GBPSO	4955555	0.0	4955555	2.050978
	CGBPSO	4955555	0.0	4955555	2.050978

6	BPSO	5687400.42	4647.9076	5688887	0.255774
	GBPSO	5688887	0.0	5688887	0.255774
	CGBPSO	5688887	0.0	5688887	0.255774
7	BPSO	6489864.92	22147.9339	6498597	0.063608
	GBPSO	6494504.52	9377.049690	6498597	0.063608
	CGBPSO	6495527.64	8311.86617	6498597	0.063608
8	BPSO	5169014.7	7356.64603	5170626	0.763310
	GBPSO	5170626	0.0	5170626	0.763310
	CGBPSO	5170626	0.0	5170626	0.763310
9	BPSO	6992144.26	1028.0805	6992404	0.587572
	GBPSO	6992404	0.0	6992404	0.587572
	CGBPSO	6992404	0.0	6992404	0.587572
10	BPSO	5329229.56	24533.539352	5337472	0.218656
	GBPSO	5336513.96	6706.280	5337472	0.218656
	CGBPSO	5337472	0.0	5337472	0.218656
11	BPSO	7844795.52	11102.9805	7850983	0.236724
	GBPSO	7849118	6443.2234	7850983	0.236724
	CGBPSO	7848738.8	6866.3346	7850983	0.236724
12	BPSO	9350438.98	12054.1785	9352998	0.015315
	GBPSO	9352787.52	1031.1372	9352998	0.015315
	CGBPSO	9352577.04	1427.54481	9352998	0.015315
13	BPSO	9138502.34	21021.3510	9151147	0.603824
	GBPSO	9149141.54	8503.477	9151147	0.603824
	CGBPSO	9149633.96	7868.7891	9151147	0.603824
14	BPSO	9338324.36	16185.4586	9348889	0.331975
	GBPSO	9344369	7825.50823	9348889	0.331975
	CGBPSO	9343060.34	11071.5220	9348889	0.331975
15	BPSO	7765397.78	7058.6312	7769117	0.201440
	GBPSO	7768447.66	3300.8555	7769117	0.201440
	CGBPSO	7768150.84	3849.0652	7769117	0.201440
16	BPSO	10722026.5	9783.64635	10727049	0.057470
	GBPSO	10725040.42	6998.4238	10727049	0.057470
	CGBPSO	10725287.36	6494.459	10727049	0.057470
17	BPSO	9811395.88	17724.2421	9818261	0.203017
	GBPSO	9815731.42	10067.8158	9818261	0.203017
	CGBPSO	9815306.14	10944.7366	9818261	0.203017
18	BPSO	10709624.44	6817.8095	10714023	0.145642
	GBPSO	10713726.22	1881.169	10714023	0.145642
	CGBPSO	10712684.2	4016.4	10714023	0.145642

19	BPSO	8925506	11617.5592	8929156	0.093826
	GBPSO	8927650.32	5676.14517	8929156	0.093826
	CGBPSO	8928605.60	2696.3983	8929156	0.093826
20	BPSO	9356884.22	3233.28548	9357969	0.132708
	GBPSO	9357137.66	2527.19298	9357969	0.132708
	CGBPSO	9357048.96	2811.7475	9357969	0.132708
21	BPSO	13532001.22	16392.2733	13549094	0.094501
	GBPSO	13542275.34	11584.7455	13549094	0.094501
	CGBPSO	13540139.16	13558.8022	13549094	0.094501
22	BPSO	12227701.92	11724.61798	12233713	0.084792
	GBPSO	12228519.34	10465.897	12233713	0.084792
	CGBPSO	12228527.08	10401.745	12233713	0.084792
23	BPSO	12439299.86	15762.910447	12448780	0.149273
	GBPSO	12446669.52	5652.8089	12448780	0.149273
	CGBPSO	12445171.5	7896.10723	12448780	0.149273
24	BPSO	11807468.06	13691.618121	11815315	0.098607
	GBPSO	11809343.62	11110.5037	11815315	0.098607
	CGBPSO	11811361.02	8936.89088	11815315	0.098607
25	BPSO	13938518.2	3624.726307	13940099	0.252759
	GBPSO	13938197.14	6802.7032	13940099	0.252759
	CGBPSO	13937847.1	6299.8126	13940099	0.252759

Table 2. Comparative results of Problem Set II of 0-1 Knapsack problems.

Example	Algorithm	AE	LE	SD	SR	AFE
1	BPSO	0.0	0.0	0.0	100	442.0
	GBPSO	0.0	0.0	0.0	100	435
	CGBPSO	0.0	0.0	0.0	100	438
2	BPSO	0.0	0.0	0.0	100	1010
	GBPSO	0.0	0.0	0.0	100	1142.0
	CGBPSO	0.0	0.0	0.0	100	1145
3	BPSO	0.1	0.0	0.303	95	15735
	GBPSO	0.0	0.0	0.0	100	15630
	CGBPSO	0.0	0.0	0.0	100	15671
4	BPSO	274.80	0.0	357.88925	55	393380
	GBPSO	220.32	0.0	295.3262	65	382850
	CGBPSO	170.20	0.0	237.761	62	382748
5	BPSO	1675.64	489	384.9666	0	1001000
	GBPSO	658.04	89	297.09	0	1001000

	CGBPSO	1300.98	362	507.2235	0	1001000
6	BPSO	3737.58	3332	207.541	0	1501500
	GBPSO	2337.76	1889	234.985	0	1501500
	CGBPSO	1610.24	1159	222.191	0	1501500

Table 3. Comparative results of Problem Set I of Multidimensional Knapsack problems.

Example	Algorithm	AE	LE	SD	SR	AFE
1	BPSO	6.78	0.0	9.95	62	130464
	GBPSO	3.70	0.0	6.97	76	95754
	CGBPSO	4.32	0.0	8.0806	75	51768
2	BPSO	8.22	0.0	6.81	30	222870.
	GBPSO	8.04	0.0	8.10	38	205518
	CGBPSO	6.68	0.0	7.8369	42	105498
3	BPSO	20.0	0.0	98.0	96	8464.4
	GBPSO	0.0	0.0	0.0	100	6706
	CGBPSO	0.0	0.0	0.0	100	8299.2
4	BPSO	73.8	0.0	267	82	17480.40
	GBPSO	9.60	0.0	38.0	94	9814
	CGBPSO	6.40	0.0	31.35	98	7142.91
5	BPSO	798.45	0.0	678.21	8	65150.40
	GBPSO	3.0	0.0	11.9	94	7560
	CGBPSO	2.0	0.0	9.797	96	6655.60
6	BPSO	198.97	0.0	633.341	86	14030.8
	GBPSO	267	0.0	47.7	94	10494.4
	CGBPSO	223.24	0.0	948.365	80	10158.75
7	BPSO	962	0.0	1621.15	70	26560.80
	GBPSO	130.26	0.0	588.92	90	13230
	CGBPSO	140.16	0.0	588.03	85	17936
8	BPSO	164	0.0	192.48	58	33373.2
	GBPSO	156.0	0.0	191.06	60	32099.20
	CGBPSO	140.4	0.0	187.20	67	30114
9	BPSO	73.0	0.0	51.482	14	694743
	GBPSO	82.78	0.0	71.87	20	447657
	CGBPSO	86.42	0.0	77.712	24	229173
10	BPSO	0.463	0.0	0.14	98	140238
	GBPSO	0.0	0.0	0.0	100	133255.5
	CGBPSO	0.0	0.0	0.0	100	121201.5

Table 4. Comparative results of Problem Set II of Multidimensional Knapsack problems.

Example	Algorithm	SR	AFE	AE	LE	SD	infeasible sol.
1	BPSO	100	5418	0	0	0	0
	GBPSO	100	32442	0	0	0	0
	CGBPSO	100	20833	0	0	0	0
2	BPSO	74	44715	1.3	0	2.19317	0
	GBPSO	78	58299	1.1	0	2.07123	0
	CGBPSO	90	33603	0.5	0	1.5	0
3	BPSO	98	12939	1.26	0	8.82	0
	GBPSO	98	39180	0.18	0	1.26	0
	CGBPSO	98	24431	0.18	0	1.26	0
4	BPSO	100	9147	0	0	0	0
	GBPSO	100	32361	0	0	0	0
	CGBPSO	100	20679	0	0	0	0
5	BPSO	100	11379	0	0	0	0
	GBPSO	100	33921	0	0	0	0
	CGBPSO	100	21897	0	0	0	0
6	BPSO	72	65744	4.64	0	7.53594	0
	GBPSO	82	84896	2.74	0	5.94242	0
	CGBPSO	80	69348	3	0	6.09918	0
7	BPSO	92	28328	1.44	0	4.88328	0
	GBPSO	94	64656	1.36	0	5.44338	0
	CGBPSO	98	41188	0.36	0	2.52	0
8	BPSO	82	45864	1.02	0	3.09509	0
	GBPSO	78	87364	0.66	0	1.93505	0
	CGBPSO	82	64140	0.36	0	0.76838	0
9	BPSO	98	15416	0.68	0	4.76	0
	GBPSO	100	52872	0	0	0	0
	CGBPSO	98	37726	0.68	0	4.72	0
10	BPSO	88	50245	0.12	0	0.32496	0
	GBPSO	88	103140	2.04	0	6.84386	0
	CGBPSO	88	82485	3.1	0	9.9844	0
11	BPSO	38	192315	842	0	819.628	14
	GBPSO	20	221995	1416	0	1378.56	13
	CGBPSO	22	219285	55.8	0	56.269	11
12	BPSO	96	53860	0.04	0	0.19596	0
	GBPSO	92	65205	2.82	0	11.3661	0
	CGBPSO	98	59285	0.02	0	0.14	0
	BPSO	90	62555	9.06	0	30.3542	0

13	GBPSO	94	60405	4.34	0	18.6275	0
	CGBPSO	98	53190	1.06	0	1.05418	0
	BPSO	66	177810	12.74	0	19.6253	0
14	GBPSO	78	91953.8	8.02	0	18.9137	0
	CGBPSO	96	52788	1.96	0	10.2547	0
	BPSO	68	105310	14.72	0	24.134	0
15	GBPSO	66	114164	13.9	0	21.033	0
	CGBPSO	90	91612	0.64	0	4.48	0
	BPSO	70	143258	1.58	0	3.48764	4
16	GBPSO	88	91716	0.6	0	2.74226	0
	CGBPSO	82	114886	0.72	0	2.58488	0
	BPSO	100	23460	0	0	0	0
17	GBPSO	96	33110	1.08	0	3.43418	0
	CGBPSO	98	28730	0.18	0	0.026	0
	BPSO	72	196344	4.3	0	7.376	0
18	GBPSO	78	205292	4.1	0	6.86222	0
	CGBPSO	88	153545	1.16	0	3.40212	0
	BPSO	48	192697	82.18	0	497.06	1
19	GBPSO	56	168159	19.6279	0	31.8624	3
	CGBPSO	74	151145	14.164	0	28.0556	0
	BPSO	80	106610	3.42	0	7.6291	0
20	GBPSO	76	152793	3.06	0	6.94956	0
	CGBPSO	94	66885	1.2	0	4.74974	0
	BPSO	75	117306	6.84	0	15.5362	0
21	GBPSO	82	113836	7	0	16.0275	0
	CGBPSO	84	112470	6.34	0	15.346	0
	BPSO	38	237208	28.081	0	37.058	0
22	GBPSO	54	190429	13.8	0	17.7854	0
	CGBPSO	60	168023	10.2	0	13.2499	0
	BPSO	10	406800	2264	0	2197.99	66
23	GBPSO	16	389333	174.714	0	694.149	42
	CGBPSO	28	330208	79.5	0	49.476	58
	BPSO	78	201229	2.3	0	4.9849	0
24	GBPSO	82	130089	2.02	0	5.76017	0
	CGBPSO	96	87360	0.54	0	3.14458	0
	BPSO	52	208440	13.51	0	18.377	0
25	GBPSO	66	193422	11.1	0	10.3252	0
	CGBPSO	70	147958	6.24	0	8.37749	0
	BPSO	0	400400	580.574	550	32.198	0
26	GBPSO	0	400400	563.46	550	19.5573	0

	CGBPSO	0	400400	571.48	550	18.111	0
27	BPSO	76	288917	17.368	0	30.925	2
	GBPSO	80	220882	16.1837	0	33.6213	1
	CGBPSO	90	179640	2.375	0	14.0396	1
28	BPSO	13	366156	2340	0	1948.84	16
	GBPSO	28	260550	2001	0	2279.28	10
	CGBPSO	42	220163	1294	0	1197.14	12
29	BPSO	0	450450	2983	609	5467.37	88
	GBPSO	0	450450	1727	699	879.02	63
	CGBPSO	0	450450	6344	459	1493.03	74
30	BPSO	18	226450	5.94	0	6.05445	0
	GBPSO	46	162470	4.18	0	6.38965	0
	CGBPSO	78	159606	0.88	0	1.65699	0
31	BPSO	46	69782.3	7.76	0	7.5301	0
	GBPSO	48	43065	14.16	0	6.011	0
	CGBPSO	46	42909	13.74	0	4.68	0
32	BPSO	50	93666.6	7.54	0	7.97047	0
	GBPSO	90	49711.7	1.3	0	3.9	0
	CGBPSO	82	58690.8	2.44	0	5.2503	0
33	BPSO	52	81559.6	1142.16	0	1453.69	0
	GBPSO	62	78285	2818	0	1025.19	0
	CGBPSO	70	74240	1141.18	0	1101.36	0
34	BPSO	28	74060	21.5	0	17.9179	0
	GBPSO	56	66728	11.5	0	15.8016	0
	CGBPSO	72	48498	2.06	0	1.107	0
35	BPSO	70	61052	5.16	0	8.75068	0
	GBPSO	64	75444	6.74	0	10.081	0
	CGBPSO	80	50462.5	2.78	0	5.449	0
36	BPSO	44	114904	4.48	0	5.11953	0
	GBPSO	50	72986.2	5.58	0	6.46866	0
	CGBPSO	54	110819	4.86	0	6.1838	0
37	BPSO	6	133898	14.5	0	5.67186	2
	GBPSO	6	135820	14.1	0	5.20865	2
	CGBPSO	48	80124.8	10.26	0	13.055	1
38	BPSO	60	81690	6.28	0	7.99009	0
	GBPSO	78	81140	3.24	0	6.201	0
	CGBPSO	84	57974	2.08	0	4.765	0

References

[1] E O Resndiz Flores, M E Lpez Quintero. *Optimal identification of impact variables in a welding process for automobile seats mechanism by MTS-GBPSO approach*, Int J Adv Manuf Syst, 2017, 90: 437-443.

[2] K Ahmed, B Al-Khateeb, M Mahmood. *Application of chaos discrete particle swarm optimization algorithm on pavement maintenance scheduling problem*, Clust Comput, 2019, 22: S4647-S4657.

[3] K Deep, P Chauhan, M Pant. *Multi task selection including part mix, tool allocation and process plans in CNC machining centers using new binary PSO*, Evolutionary Computation (CEC), IEEE Congress on, 2012, DOI: 10.1109/CEC.2012.6256439.

[4] P Chauhan, M Pant, K Deep. *Novel Binary PSO for Continuous Global Optimization Problems*, In: Deep K, Nagar A, Pant M, Bansal J(eds), Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS2011), 2011, Advances in Intelligent and Soft Computing, Springer, India, 2012, 130.

[5] Y Singh, P Chauhan. *New Mutation Embedded Generalized Binary PSO*, In: Sathiyamoorthy S, Caroline B, Jayanthi J (eds), Emerging Trends in Science, Engineering and Technology, Lecture Notes in Mechanical Engineering, Springer, India, 2012.

[6] E Chen, J Li, X Liu. *In search of the essential binary discrete particle swarm*, Appl Soft Comput, 2011, 11(3) : 3260-3269.

[7] J Kennedy, R C Eberhart. *A discrete binary version of the particle swarm algorithm*, Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, IEEE International Conference, 1997, 5(12-15) : 4104-4108.

[8] M A Khanesar, M Teshnehlab, M A Shoorehdeli. *A novel binary particle swarm optimization*,15th IEEE Med. Conf Control Automation, Athens, Greece, 2007: 1-6.

[9] L Zhao, F Qian. *Tuning the structure and parameters of a neural network using cooperative binary-real particle swarm optimization*, Expert Syst, 2011, 38(5) : 4972-4977.

[10] L Y Chuang, C H Yang, J C Li. *Chaotic maps based on binary particle swarm optimization for feature selection*, Appl Soft Computing, 2011, 11(1) : 239-248.

[11] G Pampara, N Franken, A P Engelbrecht. *Combining particle swarm optimisation with angle modulation to solve binary problems*, Evolutionary Computation, IEEE Congress, 2005, 1(5-5): 89-96.

[12] T O Ting, M V C Rao, C K Loo. *A novel approach for unit commitment problem via an effective hybrid particle swarm optimization*, IEEE Trans Power Syst, 2006, 21(1) : 411- 418.

[13] P Zhao, X Zhang. *A new ant colony optimization for the knapsack problem*, Proceedings of 7th International Conference on Computer-Aided Industrial Design and Conceptual Design, 2006: 1-3.

[14] Z Assarzadeh, A R Naghsh-Nilchi. *Chaotic particle swarm optimization with mutation for classification.* J Med Signals and Sens, 2015, 5(1): 12-20.

[15] H Djellali, N Dendani. *Chaotic Binary Particle Swarm with Anti Stagnation Strategy on Feature Selection*, In Senouci M R, Boudaren M E Y, Sebbak F, Mataoui M (eds), Advances in Computing Systems and Applications, CSA 2020, Lecture Notes in Networks and Systems, Springer, 2020, 199.

[16] Q Xiong, X Zhang, X Xu, S He. *A Modified Chaotic Binary Particle Swarm Optimization Scheme and Its Application in Face-Iris Multimodal Biometric Identification*, Electronics, 2021, 10(2):217

[17] C Y Lee, Z J Lee, S F Su. *A new approach for solving 0/1 knapsack problem*, 2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, 2006: 3138-3143.

[18] S Senyu, Y Toyoda. *An approach to linear programming with 0C1 variables*, Management Science, 1968, 15(4): B-196-B-207.

[19] H M Weingartner, D N Ness. *Methods for the solution of the multidimensional 0/1 knapsack problem*, Operations Research, 1967, 15(1): 83-103.

[20] W Shih. *A branch and bound method for the multiconstraint zero-one knapsack problem*, J Oper Res Soc, 1979, 30: 369-378.

[21] J E Beasley. *ORLib-Operations Research Library*, http://people.brunel.ac.uk/ mastjjb/jeb/ orlib/mknapinfo.html, 2005.

Department of Mathematics, Jaypee Institute of Information Technology, Noida 201304, India.
Email: pinkeychauhan030@gmail.com