Volume 65, Issue 3, 2021
Journal of Scientific Research
Institute of Science,
Banaras Hindu University, Varanasi, India.

Solving 01 Knapsack Problem with variations of Flower Pollination Algorithm

Aboli Sundarkar and Amol C. Adamuthe*

Dept. of CS&IT, RIT, Rajaramnagar, Sangli, MS, India.
abolisundarkar5@gmail.com, amol.admuthe@gmail.com*

Abstract: **The 01 knapsack problem is a combinatorial optimization problem where objective is to maximum profit. It can be considered as a resource allocation problem where the decision is to pick the most important items. Knapsack problem has a large range of applications in many domains. This paper presents three variations of FPA influenced by concept of crossover and mutation operator in genetic algorithms. The three variations, named FPA with crossover and mutation (Version 1), FPA with crossover (Version 2), FPA with mutation (Version 3) for solving 01 knapsack problem are presented. Comparison of variations is done with basic FPA algorithms and other algorithms for three datasets of a single dimension of 01 knapsack problem. The results show that improved FPA has better results than basic FPA. The results are optimal for smaller instances and with increase in number of objects results are closer to optimal but not optimal.**

Index Terms: **01 knapsack problem, flower pollination algorithm, hybrid algorithms, optimization problems.**

I. INTRODUCTION

Knapsack problem falls in the optimization category and it is a dynamic programming problem. In the field of operation research knapsack problem is considered as conventional non polynomial problem (Bansal & Deep, 2012). In literature many papers represented applications of 01 knapsack problem such as optimal load shedding (Choi et al., 2011) and cryptosystem (Boas & Barros, 2017). In a 01 knapsack problem, there is a set of items with fixed weight and respective profit value. The main objective is to increase the profit value by selecting best suitable items with available fixed capacity. Each item can be selected only once. A set of '*n'* items, each with weights W_i and profit P_i and knapsack capacity as '*M'* are given.

$$\text{Maximize: } \sum P_i X_i \qquad \text{for } i \text{ from 1 to } n$$

$$\text{Subject to } \sum W_i X_i \leq M \text{ and } X_i \text{ is } \{0, 1\}$$

Here X_i has value '1' and '0' which indicates selected and discarded respectively. The objective of the problem is to determine the maximum profit by selecting only those items whose sum of weight is less than knapsack capacity.

The optimization problems of today are becoming largely very complex. When we solve those by conventional methods, either they fail to solve the problem or they become too time-consuming. By time-consuming, it means probably even in years. So basically we are trying to explore all the state spaces of a problem, and when it matches the given state then we are stopping the search. The problem here is that the given problem solution grows exponentially. The results can be a huge number which increases time complexity. Such problems are called NP i.e. non-polynomial problems. We can reduce the time by using the solution whose heuristic value is best. A very simple meaning of a heuristic algorithm is to try to guess or find a quick answer to a problem. When the exact solution of a problem is costly and the approximate solution is sufficient then meta-heuristic algorithms are used. Meta-heuristics do not guarantee an optimal solution, it can give an optimal solution sometimes and near-optimal the other times. Many heuristic/metaheuristics algorithms are nature/bio inspired algorithms. Traveling salesman problem, knapsack problem, searching and sorting problem and virus scanning problem are some example problems solved using meta-heuristic algorithms (Kenny et al.,

* Corresponding Author
DOI: 10.37398/JSR.2021.650314

2014). Meta-heuristics are found to be suitable and vigorous in many different industrial applications from industrial design optimization to robotics (Wong & Ming, 2019).

Knapsack problem is solved by many different types of algorithms. Some examples are given below.

- Shuffled frog leaping algorithm (Bhattacharjee & Sarmah, 2014)
- Monkey algorithm (Zhou et al., 2015)
- Migrating birds optimization algorithm (Ulker et al., 2017)
- Evolution algorithm (Ali et al., 2020)
- Swarm optimization-based search algorithm (Dahmani, et al., 2020)

01 knapsack problem is NP-complete as it cannot give an exact solution for a large number of inputs. A problem has many perspectives by which it can be solved. A comparative study of dynamic programming, greedy algorithm, brute force, memory function, branch and bound, genetic algorithms is given by (Hristakeva & Shrestha, 2005) and results are compared to find the best one. Dynamic programming technique and genetic algorithm are proved to be best among others. Gradually, nowadays the problems that are to be optimized are becoming complicated along with that the evolutionary algorithms are also becoming complex. Thus we need to improvise the algorithms to solve complex problems as for any optimization problem there is no generic algorithm. Flower pollination algorithm is a heuristic/metaheuristic algorithm that is inspired by the pollination process. Nowadays, the FPA is more popular as by the time it is better than many other algorithms in solving a given problem (Basset & Shawky, 2018). This paper presents a flower pollination algorithm to solve the 01 knapsack problem. The objective of the paper is to design and develop a flower pollination algorithm for solving the 01 knapsack problem. The basic FPA algorithm has faced premature convergence for benchmarks datasets. We have tested the variations of FPA by incorporating the concepts of crossover operator and mutation operator from Genetic Algorithms.

The next section describes the details of the flower pollination algorithm. Section III describes proposed flower pollination algorithm and variations. Section IV describes dataset, result and performance evaluation followed by section V which states the conclusion at the end.

II. FLOWER POLLINATION ALGORITHM

Like many bio-inspired algorithms, the flower pollination algorithm is also inspired by nature and is used for a large number of optimization problems. The algorithm is proposed by (Yang, 2012). It is a type of evolutionary algorithm in which the individuals that are best among first-generation are combined to create a new generation with better capabilities. There are millions of different types of flowering plants. Out of which 80% are of flowering species. The main objective behind the pollination process is the natural selection of those pollens that are fit to survive. Pollination is performed by transporting the pollen from the male anther to the female stigma. Pollination is of two types biotic and abiotic. In biotic pollination, pollens are transferred by the means of living creatures while abiotic pollination is done by wind, water, or rain. 90% of pollination is done in a biotic way while only 10% follows abiotic in which there are no pollinators. Pollination in plants is also determined as self-pollination and cross-pollination. When the transfer of pollen happens from the same flower or different flower from the same plant it is called self-pollination. There are no dependable pollinators. In cross-pollination the transfer of pollen from one plant to the flower of another plant. Ali (2014) has explained the history of the flower pollination algorithm briefly in his paper. Yang (2012) came up with four rules stating the pollination behavior and flower consistency. It is given by (Salgora & Singh, 2017) in their paper.

1. Global pollination: biotic and cross-pollination. Done via Levy distribution.
2. Local pollination: abiotic and self-pollination.
3. Flower consistency which can also be considered as reproduction probability is proportional to the similarity of two flowers involved.
4. Switch probability: Controls local and global pollination.

The main steps involved in the flower pollination algorithm are given below:

Step 1: Initialize the parameters of FPA.
Step 2: Initialize switch probability.
Step 3: Find the initial best solution.
Step 4: For each iteration
If rand < P^{switch} then perform global pollination, else local pollination.
Step 5: Evaluate the new best solution.
Step 6: If the new best solution > initial best solution then updates the solution.
Step 7: Else repeat 4, 5, and 6 till termination condition is released.

Flower pollination algorithm applies to numerous real-world implementation in various fields of research. There are many domains where FPA is been used such as wireless sensor networking, image processing, gaming, classifications and clustering problems, engineering problems (Alyasseri et al., 2018) such as mechanical, electrical and power, chemical, civil, computer science, and many more. A huge amount of research is done in the electrical and power domain which solve problems

of automatic generation control (Madasu et al., 2018), voltage stability (Pravalika et al., 2016) and load frequency (Jagatheensan et al., 2017). In wireless sensor networking, FPA is used for problems of layouts of nodes (Nguyen et al., 2019) and antenna design problems (Salgotra et al., 2020). FPA shows supremacy in the domain of image processing and a sensor for problems such as coloring (Lei et al., 2020), medical image segmentation (Wang et al., 2015), person identification based on EEG (Alyasseri et al., 2018) and many more. FPA is used to find solutions for hard levels of games such as Sudoku (Raouf et al., 2014). In the clustering and classification domain, FPA investigated for different problems like image segmentation (Dhal et al., 2019) and data clustering (Senthilnath et al., 2019).

Dubey et al., (2015) proposed a modified flower pollination algorithm (MFPA) to solve economic dispatch problems in power systems in two phases. In the first phase, the local pollination of FPA is controlled by using the scaling factor and in the second phase, an intensified exploitation step is added for tuning up the best FPA solution. MFPA was tested on many mathematical benchmarks along with test cases of four huge power systems and it outperforms compared to other methods.

Dahi et al., (2016) in their paper have proposed four binary variants of FPA (BFPA) by conducting a study about common mapping techniques and effectiveness of FPA. The algorithm was compared with two other algorithms named, differential evolution algorithm (DF) and population-based incremental learning (PBIL). The results show that BFPA outperforms 4 out of 13 instances of PBIL and 6 out of 13 instances of DF with zero statistical difference among other instances.

FPA is hybridized with other algorithms like PSO, GA, etc. To get optimal solutions in a short stretch of time hybridization method is used for FPA. Raouf et al., (2014) proposed a hybrid of SFPA and Chaotic Harmony Search (FPCHS) to increase the searching accuracy of SFPA. SFPA was also hybridized with PSO to improve the performance of SFPA which was used to solve the global optimization problem. Salgotra & Singh, (2017) in their paper reported that FPA as a standard algorithm has many drawbacks so they propose new variations of FPA containing the mutation, dynamic switching, and improving local search. The best found among them was the adaptive-levy flower pollination algorithm. The algorithm when compared to other algorithms gives better performance. Many other variations of FPA are developed by modification in local and global search, parameter-tuning, and hybridization to cope up with difficult optimization problems (Alyasseri et al., 2018).

Sayed et al., (2016) introduced a hybrid algorithm BCFA combination of clonal selection algorithm (CSA) with a flower pollination algorithm (FPA) for the problem of feature selection. For optimal function, the author used an optimum path forest classifier. The algorithm was compared with another hybrid meta-heuristic algorithm on three datasets and found that BCFA gave better results in comparison. Alweshah et al., (2020) has hybridized FPA with PNN for tuning the neural network weights in-order to expand the accuracy of the classification and speed convergence. PNN randomly produced the initial solution and then FPA is used for adjusting the weights. The hybridized algorithm was tested on 11 benchmarks and it is observed that it outperforms the basic PNN algorithm for all instances. Al-qaness et al., (2020) has proposed modified FPA by using a salp-swarm algorithm (SSA) and named it as FPASSA. The modified version is applied to an adaptive neuro-fuzzy inference system (ANFIS) to improve its performance by identifying the parameters of ANFIS. The improved model FPASSA-ANFIS is used as a forecasting technique to check the count of total confirmed cases of novel coronavirus that outbreak in Wuhan china in December 2019. The proposed algorithm has a high capacity to predict the number of confirmed cases within the time-span of 10 days. Also to evaluate the model two datasets of weekly confirmed cases of the USA and China were taken and the performance was quite good.

III. PROPOSED FPA VARIATIONS FOR 01 KNAPSACK PROBLEM

In this section variations of FPA are presented to solve the 01 knapsack problem. The FPA variations are proposed using the concepts of crossover and mutation.

Solution representation: 01 knapsack problem is solved by using 1D representation as shown in figure. The size of the array is indicated by the total number of objects. The value '1' indicates that the object is selected and '0' indicates that it is rejected. Figure 1 shows a sample solution for 10 objects, in which 3, 4, 7, 8, 10 are selected.

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
0	0	1	1	0	0	1	1	0	1

Fig. 1. Memory representation (Adamuthe et al., 2020)

The paper presents three variations of FPA.

1. Version 1: FPA with crossover and mutation
2. Version 2: FPA with crossover
3. Version 3: FPA with mutation

In the crossover, we combine parts of the parent's solution to produce new ones. Two solutions are selected from the population. The crossover operator is taken to generate new solutions from the existing. There are many different kinds of crossover operators presented in literature, we have used a single point crossover. For a single point crossover, there is a need to select a crossover point from 1 to l in which l is the length of crossover. In mutation, we are randomly change one or more

genes in solution. It creates one offspring from each parent. There are various methods to perform a mutation, here we have used swap mutation.

In version 1, the crossover and mutation is performed after global and local pollination. The pseudocode of this is given below.

Algorithm: Pseudocode for proposed FAP algorithm

```
Initialize parameters such as population size (n), switch probability
  (p).
Initialize the population of flowers randomly.
Find the best solution g* from the initial solution.
N_iter ← number of iterations.
Initialize weight (w), profit (p), number of objects (d), knapsack
  capacity (m) from excel sheet.
Initialize answer in excel sheet.
for t in 1 to N_iter do
  for i in 1 to n do
    if rand < p then
        Draw a d-dimensional step vector L which obeys a levy
        distribution
        Do global pollination
    else
        Draw epsilon from uniform distribution in [0,1]
        Randomly choose JK1 and JK2 from population
        Do local pollination
    end if
    Initialize crossoverIndex← randi(d to 1)
    Initialize parent1, parent2 with two random solutions
    Apply crossover operator on current population
    Evaluate newly generated solution x_i^{t+1}
    If new solution is better then replace x_i^t by x_i^{t+1}
    Update the current global best solution g*.
  end for
  answer(t,1) ←N_iter;
  answer(t,2) ←current global best;
end for

Initialize two random solution for mutation
Apply mutation operator on population
Again update the current global best g* if the solution obtained is
  better than the previous one.
```

In version 2, the only crossover is done after global and local pollination. Version 2 works more like a basic FPA. In version 3, the only mutation is performed after global and local pollination.

IV. Dataset, Results and Discussion

This section presents dataset details, results obtained and discussion. The proposed flower pollination algorithm for solving the 01 knapsack problem is implemented by using 'Matlab' programming language on the Matlab tool 2015a. The programs are executed on a machine with Intel Core i3-6006U CPU 2GHz and 4GB RAM. The overall performance of the proposed variations are examined by using a benchmark instances of the 01 knapsack problem. The instances of knapsack problems are taken from various sources. The paper presents the optimal solution for proposed variations of flower pollination algorithms using profit, convergence graph and success rate. The success rate is calculated by the number of times best obtained divided by the number of times the program is executed. For every dataset, FPA was executed 5 times.

Dataset 1: The dataset consists of eight instances. These instances are taken from (Knapsack_01 Data for the 01 knapsack problem) which are shown in table I. Knapsack capacity, the weight of objects, respective profits and dimensions are given in the table.

The obtained solutions are compared with the harmony search algorithm (Adamuthe et al., 2020) and the optimal solution. Table II shows the optimal profit, profit obtained by the harmony search algorithm, profit obtained by the flower pollination algorithm and its variations which include FPA with crossover and mutation (Version 1), FPA with crossover (Version 2) & FPA with mutation (Version 3). Table III shows the success rate of all the variations.

Table I. Weight, profit, capacity and dimensions of dataset 1

Dataset	Dimension	Parameter (capacity, weight, profit)
P1	10	Capacity: 165 Weights: 23 31 29 44 53 38 69 85 89 8 Profits: 92 57 49 68 60 43 67 84 87 72
P2	5	Capacity: 26 Weights: 12 7 11 8 9 Profits: 24 13 23 15 16
P3	6	Capacity: 190 Weights: 56 59 80 64 75 17 Profits: 50 50 64 46 50 5
P4	7	Capacity: 50 Weights: 31 10 20 19 4 3 6 Profits: 70 20 39 37 7 5 10
P5	8	Capacity: 104 Weights: 25 35 45 5 25 3 2 2 Profits: 350 400 450 20 70 8 5 5
P6	7	Capacity: 170 Weights: 41 50 49 59 55 57 60 Profits: 442 525 511 593 546 564 617
P7	15	Capacity: 750 Weights: 70 73 77 80 82 87 90 94 98 106 110 113 115 118 120 Profits: 135 139 149 150 156 163 173 184 192 201 210 214 221 229 240
P8	24	Capacity: 6404180 Weights: 382745 799601 909247 729069 467902 44328 34610 698150 823460 903959 853665 551830 610856 670702 488960 951111 323046 446298 931161 31385 496951 264724 224916 169684 Profits: 825594 1677009 1676628 1523970

		943972 97426 69666 1296457 1679693 1902996 1844992 1049289 1252836 1319836 953277 2067538 675367 853655 1826027 65731 901489 577243 466257 369261

Table II. Comparison of all variations of FPA for dataset 1

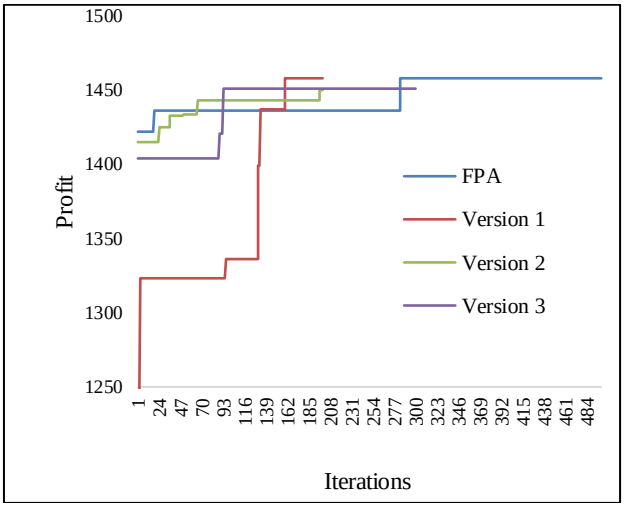
Dataset	Optimal	HS (Adamu the et al., 2020)	FPA	Version 1	Version 2	Version 3
P1	309	**309**	**309**	**309**	**309**	**309**
P2	51	**51**	**51**	**51**	**51**	**51**
P3	150	**150**	**150**	**150**	**150**	**150**
P4	107	**107**	**107**	**107**	**107**	**107**
P5	900	**900**	**900**	**900**	**900**	**900**
P6	1735	**1735**	**1735**	**1735**	**1735**	**1735**
P7	1458	**1458**	**1458**	**1458**	1450	1451
P8	13549094	**13549094**	133697167	13453154	13389908	13369336

Table III. Comparison of success rate for dataset 1

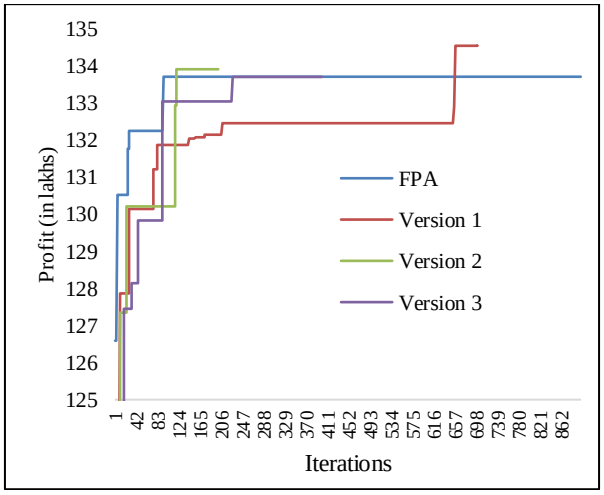
Dataset	Success rate in %			
	FPA	Version 1	Version 2	Version 3
P1	20	**60**	20	20
P2	80	**100**	40	20
P3	**60**	**60**	40	20
P4	60	**100**	40	20
P5	20	**40**	20	20
P6	**40**	**40**	20	20
P7	20	**40**	0	0
P8	0	0	0	0

Figure 2 shows the performance of variations of FPA in the form of a convergence graph. The Y-axis is best obtained profit and X-axis is number of iteration. The results show that Version 1 gives optimal solutions in less iterations compared to other variations.

FPA gives optimal results for first seven instances. For instance P8 the optimal value is not obtained. The result obtained is close to the optimal solution. FPA works well for a smaller number of objects but as the objects increase the performance of the algorithm is reduced. Version 1 gives better solutions compared to FPA. Performance of Version 2 and version 3 is similar to FPA. Among all the variations of modified FPA, version 1 works best. Harmony search algorithm given 100% results for all the instances of dataset 1. The performance of basic FPA and version 1 of modified FPA have given best results for 87.5%. The Version 2 and Version 3 of modified FPA have given similar results for 75% instances.



(a)



(b)

Fig. 2. Graph of convergence for dataset 1 (a) 15 objects (b) 24 objects

The success rate of Version 1 is more than that of basic FPA, Version 2 and Version 3. It is observed that version 3 of modified FPA which consists of only mutation fails to show any improvement.

Dataset 2: Table IV shows the dataset which is taken from Bhattacharjee & Samrah (2014). It is tested for flower pollination algorithm and its variations. The dataset contains ten instances.

Table IV. Dataset 2 (from Bhattacharjee & Samrah, 2014)

Dataset	Dimension	Parameter(w, p, b)
P2_1	10	w = {95, 4, 60, 32, 23, 72, 80, 62, 65, 46}; p = {55, 10, 47, 5, 4, 50, 8, 61, 85, 87}; b =

		269.
P2_2	20	w = {92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58}; p = {44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63}; b = 878.
P2_3	4	w = {6, 5, 9, 7}; p = {9, 11, 13, 15}; b = 20.
P2_4	4	w = {2, 4, 6, 7}; p = {6, 10, 12, 13}; b = 11.
P2_5	15	w={56.358531, 80.87405, 47.987304, 89.59624, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575}; p={0.125126, 19.330424, 58.500931, 35.029145, 82.284005, 17.41081, 71.050142, 30.399487, 9.140294, 14.731285, 98.852504, 11.908322, 0.89114, 53.166295, 60.176397}; b = 375.
P2_6	10	w = {30, 25, 20, 18, 17, 11, 5, 2, 1, 1}; p = {20, 18, 17, 15, 15, 10, 5, 3, 1, 1}; b = 60.
P2_7	7	w = {31, 10, 20, 19, 4, 3, 6}; p = {70, 20, 39, 37, 7, 5, 10}; b = 50.
P2_8	23	w = {983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959} ; p = {81, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857} ; b = 10000.
P2_9	5	w = {15, 20, 17, 8, 31}; p = {33, 24, 36, 37, 12}; b =80.
P2_10	20	w = {84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92}; p = {91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44}; b = 879.

Table V. Comparison of results for dataset 2

Data set	**Optimal** (Bhattacharjee & Sarmah, 2015)	**HS (**A. C. Adamuthe et al., 2020)	**Shuffled frog** (Bhattacharjee & Sarmah, 2014)	**FPA**	**Version 1**	**Version 2**	**Version 3**
p2_1	295	**295**	295	**295**	**295**	**295**	**295**
p2_2	1024	945	955	**1024**	**1024**	**1024**	987
p2_3	35	**35**	35	**35**	**35**	**35**	**35**
p2_4	23	**23**	23	**23**	**23**	**23**	**23**
p2_5	481.06	**481.06**	481.07	**481.06**	**481.06**	**481.06**	**481.06**
p2_6	52	**52**	52	**52**	**52**	**52**	**52**
p2_7	107	**107**	107	**107**	**107**	**107**	**107**
p2_8	9767	9731	9759	9712	**9767**	9752	9747
p2_9	130	**130**	130	**130**	**130**	**130**	**130**
p2_10	1025	889	1010	**1025**	950	**1025**	1014

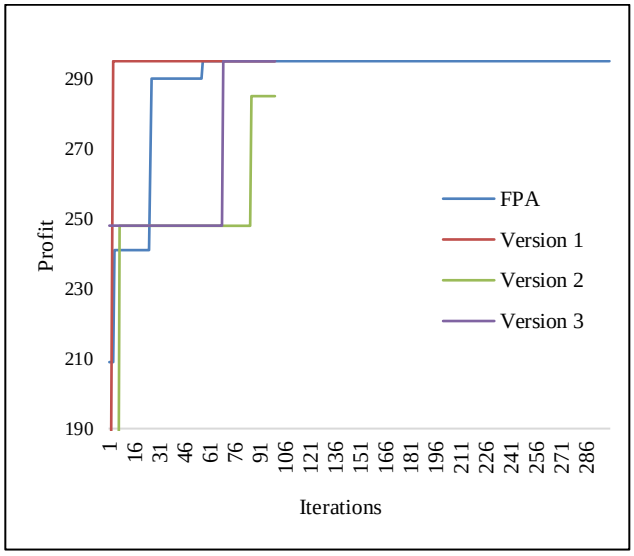
Table VI. Success rate of all the variations of FPA for dataset 2

Dataset	**Success rate in %**			
	FPA	**Version 1**	**Version 2**	**Version 3**
p2_1	40	**60**	40	20
p2_2	20	**60**	20	0
p2_3	**100**	**100**	**100**	40
p2_4	**100**	**100**	**100**	40
p2_5	20	**60**	20	20
p2_6	**80**	**80**	60	20
p2_7	**60**	**60**	40	20
p2_8	**20**	**20**	0	0
p2_9	**60**	**60**	40	20
p2_10	**40**	0	20	0

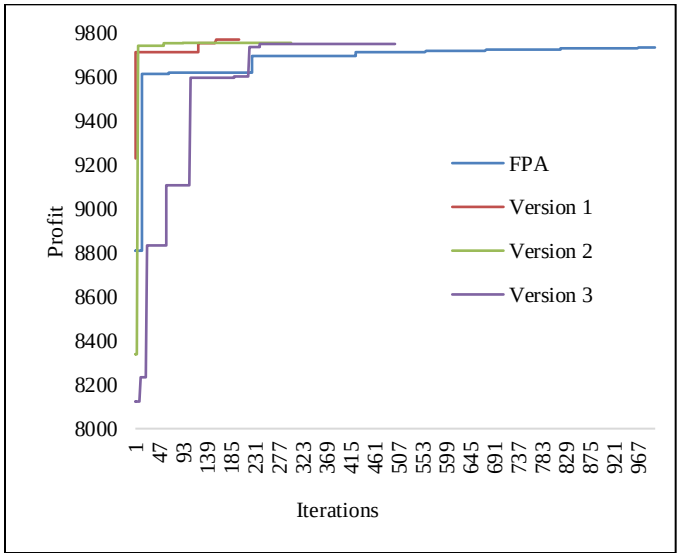
The results obtained are shown in table V and the success rate in table VI. It is observed that the harmony search algorithm gives the optimal solutions for seven instances whereas FPA gives the optimal solutions for nine instances. The result shows that for the dataset 2, version 2 works better than version 1. Harmony search algorithm given optimal results for 70% instances. The basic FPA and version 2 of the modified FPA have given results for 90% instances. Version 1 and Version 3 have given results for 80% and 70% instances respectively. Version 2 gave best for optimal solution, but the success rate of Version 1 is more than FPA followed by Version 2 and then version 3. For dataset 2, Version 3 shows the marginal improvement among all.

Figure 3 shows the convergence graph of obtained best profit per iterations for dataset 2. For dataset 2, Version 2 works better for optimal solutions. But Version 1 shows faster convergence compared to version 2 and version 3.

Dataset 3: The third dataset is taken from (http://www.math.mtu.edu/~kreher/cages/Data.html). It contains 25 instances varying from 8 objects to 24 objects shown in table VII below. The results obtained are shown in table VIII.



(a)



(b)

Fig. 3. Graph of convergence for dataset 2 (a) 10 objects (b) 20 objects

Harmony search gives optimal solutions for all the instances. Harmony search algorithms have given best solutions for 100% instances. Performance of FPA reduced with increase in number of objects. Basic FPA and all variations gives optimal solutions to instances upto 12 objects. But, FPA and variations fails to give optimal solutions when object size becomes 16 and above. The results obtained are close to best but not optimal. Basic FPA and Version 1 gives results for 48% and 44% instances respectively. Version 2 and Version 3 gives results for 40% instances. The optimal results and success rate of Version 1 are better than FPA. Version 2 and Version 3 shows similar performance to FPA. For dataset 3, performance of Version 3 is not satisfactory.

Table VII. Comparison of all variations of FPA for dataset 3

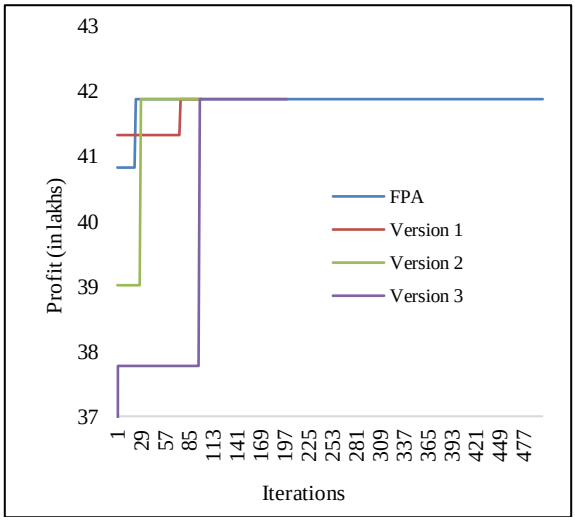
Dataset	**Optimal**	**HS** (Adamuthe et al., 2020)	**FPA**	**Version 1**	**Version 2**	**Version 3**
8a	3924400	**3924400**	**3924400**	**3924400**	**3924400**	**3924400**
b	3813669	**3813669**	**3813669**	**3813669**	**3813669**	**3813669**
8c	3347452	**3347452**	**3347452**	**3347452**	**3347452**	**3347452**
8d	4187707	**4187707**	**4187707**	**4187707**	**4187707**	**4187707**
8e	4955555	**4955555**	**4955555**	**4955555**	**4955555**	**4955555**
12a	5688887	**5688887**	**5688887**	**5688887**	**5688887**	**5688887**
12b	6498597	**6498597**	**6498597**	**6498597**	**6498597**	**6498597**
12c	5170626	**5170626**	**5170626**	**5170626**	**5170626**	**5170626**
12d	6992404	**6992404**	**6992404**	**6992404**	**6992404**	**6992404**
12e	5337472	**5337472**	**5337472**	**5337472**	**5337472**	**5337472**
16a	7850983	**7850983**	**7850983**	7821255	7823318	7823318
16b	9352998	**9352998**	9132010	9352982	9200221	9138620
16c	9151147	**9151147**	9111941	9141408	9126949	9051752
16d	9348889	**9348889**	**9348889**	9336691	9288795	9266369
16e	7769117	**7769117**	7769114	**7769117**	7754276	7725775
20a	10727049	**10727049**	10567407	10727003	10674314	10581901
20b	9818261	**9818261**	9638584	9818160	9652251	9673100
20c	10714023	**10714023**	10531375	10713021	10598176	10503619
20d	8929156	**8929156**	8837509	8929024	8893733	8839607
20e	9357969	**9357969**	9300514	9332165	8893733	9218972
24a	13549094	**13549094**	13284741	13443574	9253870	9218972
B	12233713	**12233713**	12121436	12141536	11729542	12121436
24c	12448780	**12448780**	12285798	12393818	12211862	12216778
24d	11815315	**11815315**	11660115	11740165	11610659	11522381
24e	13940099	**13940099**	13733826	13843812	13682122	13381600

Table VIII. Success rate of all the variations of FPA for dataset 3

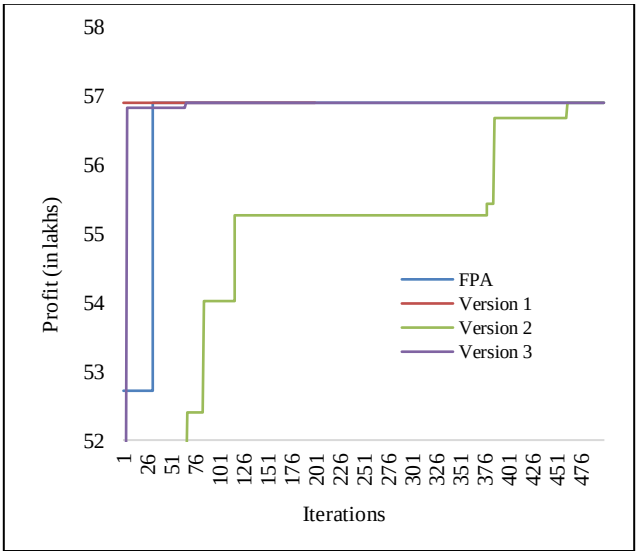
Dataset	**Success rate in %**			
	FPA	**Version 1**	**Version 2**	**Version 3**
8a	20	**40**	20	20
8b	20	**40**	20	20
8c	20	**40**	20	20
8d	20	**40**	20	20
8e	20	**40**	20	20

12a	20	**40**	20	20
12b	20	**40**	20	20
12c	20	**40**	20	20
12d	20	**40**	20	20
12e	20	**40**	20	20
16a	**20**	0	0	0
16b to 24e	0	0	0	0

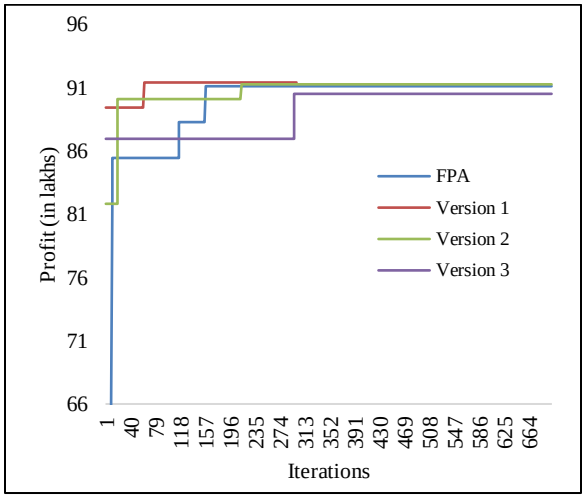
Figure 4 shows the convergence graphs. The convergence graphs shows that Version 1 takes less iteration than all other variations of FPA.



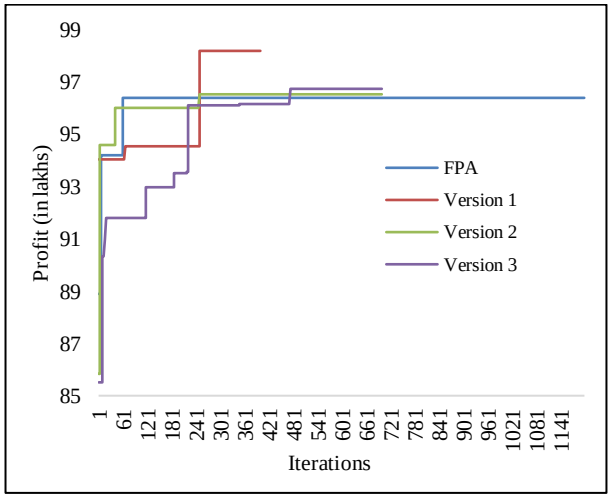
(a)



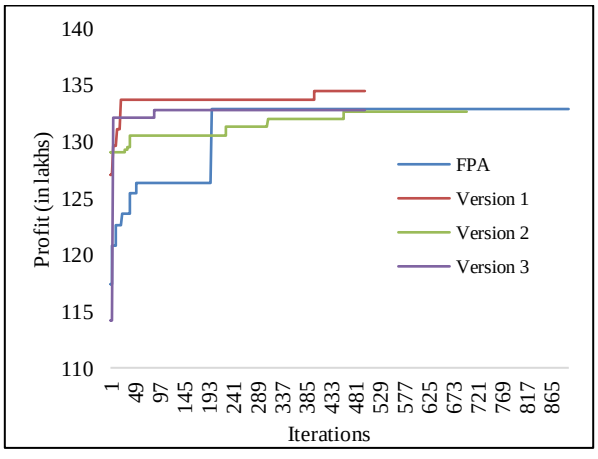
(b)



(c)



(d)



(e)

Fig. 4. Graph of convergence for dataset 3 for instance (a) 8 objects (b) 12 objects (c) 16 objects (d) 20 objects (e) 24 objects

CONCLUSIONS

01 knapsack problem comes under NP category and finding optimal solution becomes difficult when problem size increases. In literature, large number of applications of 01 knapsack problem are presented. Different heuristic algorithms are investigated to optimize the solutions. This paper presents a flower pollination algorithm to solve the 01 knapsack problem. It is an optimization algorithm encouraged by the pollination behavior of flowering plants. Methodologies of crossover and mutation is applied to basic FPA with the purpose of improving the performance of basic FPA algorithm. Three variations of FPA are done using crossover and mutation operators. First variation (Version 1) consists of both crossover and mutation with FPA. Version 1 found the most optimal compared to other two variations. The second variation (Version 2) includes crossover with FPA which gives results similar to basic FPA. The third variation (Version 3) includes mutation with FPA. All three proposed variations of FPA are shows a slight improvement in the results. For dataset 1, Version 1 found best than basic FPA, Version 2 and Version 3. The success rate of Version 1 is better than others. For dataset 2, basic FPA given better solutions. But, the success rate of version 1 is better than remaining variations. For dataset 3, success rate of version 1 is better but decreased for large instances.

Future work: Performance of FPA decreased with increase in problem size which leads to increase in search space and number of combinations. There is a scope of improvement in the variations that would provide optimal results even for large instances.

REFERENCES

Abdel-Basset, M., & Shawky, L. A. (2019). Flower pollination algorithm: a comprehensive review. *Artificial Intelligence Review*, *52*(4), 2533-2557.

Abdel-Raouf, O., & Abdel-Baset, M. (2014). A new hybrid flower pollination algorithm for solving constrained global optimization problems. *International journal of applied operational research*, *4*(2), 1-13.

Abdel-Raouf, O., El-Henawy, I., & Abdel-Baset, M. (2014). A novel hybrid flower pollination algorithm with chaotic harmony search for solving sudoku puzzles. *International Journal of Modern Education and Computer Science*, *6*(3), 38.

Adamuthe, A. C., Sale, V. N., & Mane, S. U. (2020). Solving single and multi-objective 01 Knapsack Problem using Harmony Search Algorithm. *Journal of Scientific Research*, *64*(1).

Ahmed fouad Ali (2014). Flower pollination algorithm. Scientific research group in Egypt.

Ali, I. M., Essam, D., & Kasmarik, K. Novel binary differential evolution algorithm for knapsack problems. *Information Sciences*, *542*, 177-194.

Al-Qaness, M. A., Ewees, A. A., Fan, H., & Abd El Aziz, M. (2020). Optimization method for forecasting confirmed cases of COVID-19 in China. *Journal of Clinical Medicine*, *9*(3), 674.

Alweshah, M., Qadoura, M. A., Hammouri, A. I., Azmi, M. S., & AlKhalaileh, S. (2020). Flower Pollination Algorithm for Solving Classification Problems. *Int. J. Advance Soft Compu. Appl*, *12*(1).

Alyasseri, Z. A. A., Khader, A. T., Al-Betar, M. A., Awadallah, M. A., & Yang, X. S. (2018). Variants of the flower pollination algorithm: a review. In *Nature-Inspired Algorithms and Applied Optimization* (pp. 91-118). Springer, Cham.

Alyasseri, Z. A. A., Khader, A. T., Al-Betar, M. A., Papa, J. P., & ahmad Alomari, O. (2018, July). Eeg-based person authentication using multi-objective flower pollination algorithm. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-8). Ieee.

Bansal, J. C., & Deep, K. (2012). A modified binary particle swarm optimization for knapsack problems. *Applied Mathematics and Computation*, *218*(22), 11042-11061.

Bhattacharjee, K. K., & Sarmah, S. P. (2014). Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. *Applied soft computing*, *19*, 252-263.

Bhattacharjee, K. K., & Sarmah, S. P. (2015, December). A binary firefly algorithm for knapsack problems. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 73-77). IEEE.

Boas, Y. S. V., & de Barros, C. F. SRVB Cryptosystems four approaches for Knapsack-based cryptosystems.

Choi, S., Park, S., & Kim, H. M. (2011). The Application of the 0-1 Knapsack problem to the load-shedding problem in microgrid operation. In *Control and automation, and energy system engineering* (pp. 227-234). Springer, Berlin, Heidelberg.

Dahi, Z. A. E. M., Mezioud, C., & Draa, A. (2016). On the efficiency of the binary flower pollination algorithm: application on the antenna positioning problem. *Applied Soft Computing*, *47*, 395-414.

Dahmani, I., Hifi, M., Saadi, T., & Yousef, L. (2020). A swarm optimization-based search algorithm for the quadratic knapsack problem with conflict Graphs. *Expert Systems with Applications*, *148*, 113224.

Dhal, K. G., Gálvez, J., & Das, S. (2019). Toward the modification of flower pollination algorithm in clustering-based image segmentation. *Neural Computing and Applications*, 1-19.

Donald L. Kreher. Retrieved from http://www.math.mtu.edu/~kreher/cages/Data.html

Dubey, H. M., Pandit, M., & Panigrahi, B. K. (2015). A biologically inspired modified flower pollination algorithm for solving economic dispatch problems in modern power systems. *Cognitive Computation*, *7*(5), 594-608.

Hristakeva, M., & Shrestha, D. (2005, April). Different approaches to solve the 0/1 knapsack problem. In *The Midwest Instruction and Computing Symposium*.

Jagatheesan, K., Anand, B., Samanta, S., Dey, N., Santhi, V., Ashour, A. S., & Balas, V. E. (2017). Application of flower pollination algorithm in load frequency control of multi-area interconnected power system with nonlinearity. *Neural Computing and Applications*, *28*(1), 475-488.

Kenny V., Nathal M., and Saldana S. (2014). Heuristic algorithms. *ChE 345 spring*.

Lei, M., Zhou, Y. & Luo, Q. Color image quantization using flower pollination algorithm. *Multimed Tools Appl* 79, 32151–32168 (2020).

Madasu, S. D., Kumar, M. S., & Singh, A. K. (2018). A flower pollination algorithm based automatic generation control of interconnected power system. *Ain Shams Engineering Journal*, *9*(4), 1215-1224.

Nguyen, T. T., Pan, J. S., & Dao, T. K. (2019). An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network. *Ieee Access*, *7*, 75985-75998.

Pravallika, D. L., & Rao, B. V. (2016). Flower pollination algorithm based optimal setting of TCSC to minimize the transmission line losses in the power system. *Procedia Computer Science*, *92*, 30-35.

Salgotra, R., & Singh, U. (2017). Application of mutation operators to flower pollination algorithm. *Expert Systems with Applications*, *79*, 112-129.

Salgotra, R., Singh, U., Saha, S., & Nagar, A. K. (2020). Improved flower pollination algorithm for linear antenna design problems. In *Soft Computing for Problem Solving* (pp. 79-89). Springer, Singapore.

Sayed, S. A. F., Nabil, E., & Badr, A. (2016). A binary clonal flower pollination algorithm for feature selection. *Pattern Recognition Letters*, *77*, 21-27.

Senthilnath, J., Kulkarni, S., Suresh, S., Yang, X. S., & Benediktsson, J. A. (2019). FPA clust: evaluation of the flower pollination algorithm for data clustering. *Evolutionary Intelligence*, 1-11.

Ulker, E., & Tongur, V. (2017). Migrating birds optimization (MBO) algorithm to solve knapsack problem. *Procedia computer science*, *111*, 71-76.

Wang, R., Zhou, Y., Zhao, C., & Wu, H. (2015). A hybrid flower pollination algorithm based modified randomized location for multi-threshold medical image segmentation. *Bio-medical materials and engineering*, *26*(s1), S1345-S1351.

Wong, W. K., & Ming, C. I. (2019, June). A Review on Metaheuristic Algorithms: Recent Trends, Benchmarking and Applications. In *2019 7th International Conference on Smart Computing & Communications (ICSCC)* (pp. 1-5). IEEE.

Yang, X. S. (2012, September). Flower pollination algorithm for global optimization. In *International conference on unconventional computing and natural computation* (pp. 240-249). Springer, Berlin, Heidelberg.

Zhou, Y., Chen, X., & Zhou, G. (2016). An improved monkey algorithm for a 0-1 knapsack problem. *Applied Soft Computing*, *38*, 817-830.

***