Iteration split with Firefly Algorithm and Genetic Algorithm to Solve Multidimensional Knapsack Problems

Ravneil Nand
SCIMS, FSTE
The University of the South Pacific
Suva, Fiji
ravneil.nand@usp.ac.fj

Priynka Sharma
SCIMS, FSTE
The University of the South Pacific
Suva, Fiji
priynka.sharma@usp.ac.fj

***Abstract*—When we talk about optimization, we mean to get the best or the optimal solutions from some set of available substitutes for the problems. If constraints are introduced in the problem, the feasible range would change. As we venture further in optimization, different types of problems need different approaches. One very common problem is combinatorial optimization problems. Combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects. In simple terms, finding optimal solutions from some set of available datasets of a problem. Multi Knapsack Problem (MKP) is NP-hard combinational optimization problem better known as the multi-constraint knapsack problem. It is one of the extensively studied problems in the field as it has a variety of real world problems associated with it. In this paper, the Firefly algorithm is used with the Genetic algorithm to solve the Multidimensional Knapsack Problem (MKP). By using the properties of flashing behavior of fireflies together with genetic evolution, some benchmark problems are solved. The results are compared with some work from the literature.**

***Index Terms*—Knapsack Problem , Firefly Algorithm, Genetic Algorithm**

I. Introduction

A task is said to be perfect if its optimal region or range has been discovered. One technique involving optimal region search is optimization. Optimization is not only function depended but also there are cases where data is involved. Optimization with and without data is quite interesting as researchers are focused to get optimal solutions based on some state-of-the-art techniques. Optimization is finding global minimum or maximum based on the objectives and constraints if any. In the real world problems, there may not only exist one objective to optimize but can be multiple [7][19]. With multiple objectives there comes added difficulties as all or most of the objectives needs to be optimized and optimal region would shift.

In the multi-objective problem, one of the best algorithms to date is NSGA-II [7]. NSGA-II uses non-dominated sets in order to solve the problem at hand. Other algorithms that are equally good are Particle Swarm Optimization (PSO) [16], Genetic Algorithm (GA) [6], Firefly Algorithm (FA) [30], and Ant Colony Optimization (ACO) [9]. Though there has been modification and hybridization of all these algorithms as there is a chance of better algorithm with an added advantage. This has given the motivation to look at Firefly algorithm in much detail and modify in terms of the structure whereby another algorithm architecture would be beneficial. The reason to choose Firefly algorithm is due to its simplicity and less application when compared to its counterparts.

Be it single objective or multi-objective, an optimization area that covers all is combinatorial optimization problems [22] [25]. Combinatorial optimization involves finding a finite solution from a set of objects such as a knapsack [29][20]. It operates on the domain of those optimization problems, whereby the set of feasible solutions is discrete or can be reduced to discrete, in which the goal is to find the best solution.

There has been a lot of problems involving combinatorial problems, two extensively discussed over a decade are Traveling Salesman Problem [18] and Knapsack Problem [15] [19] [2]. The one discussed in this paper is the Knapsack Problem. It has been noticed, that to solve knapsack problem especially multi-dimensional [19] new approaches are needed where modification [2] is a must to get better results. Even in [27], the author has even modified PSO method to solve benchmark multidimensional knapsack problem instances.

The main aim involves the implementation of the FA method with other meta-heuristic algorithms such as GA to solve some combinatorial optimization problems as FA been widely used to solve the continuous problem same as PSO. The implementation needs to combine FA with GA where the other method is only applied after executing the FA method. Besides, this particular strategy, other strategies for combining both methods may be considered, for example, the combination in which the FA algorithm is executed after applying the other algorithm. However, this alternative would make hybrid of the other method, i.e., the method to generate the initial set of solutions becomes the applied algorithm. Nevertheless, we have proposed the following combination strategy FA-GA Cycles. The two methods are implemented in separate cycles, such that the resulting algorithm has two cycles: in the first, the

FA method is implemented and in the second the PSO. This strategy halves the time complicity of both the algorithm.

In this paper, hybrid of FA and GA is discussed with application to Multi-Dimensional Knapsack problem (MDKP). The algorithm is tested against benchmark test problems and later compared with Binary PSO to see if any improvement in terms of average fitness value.

Rest of the paper is arranged as follows. Section II discusses the knapsack problem while Section III is on Firefly Algorithm and proposed method. Section IV is on experimental setup and results while discussion is given in Section V. The Section VI highlights concluding remarks with the future application.

II. Knapsack Problem

Knapsack problems are classical NP-hard problems from the domain of combinatorial problems which have been extensively studied in the literature [29] since first appearance in the work by Dantzig [5]. There is numerous application of this class of problems in the industry such as economics, engineering, financial, etc. As a consequence, various individuals appreciate problem illuminating and a vast gathering of puzzlement that can be found in various literature. However, employing intelligent optimization algorithms to take care of the optimization issues is an interesting subject and has numerous feasible applications. The problem requires to maximize the corresponding profit of selecting some given items with value and weight where the capacity of the selected items is bounded by a knapsack capacity or weight [14]. There are different types of knapsack problems depend on the distribution of items in a knapsack. Two common ones are 0-1 Knapsack problem and Multiple or Multidimensional knapsack problems.

The knapsack problem (KP) to a great extent considered as a discrete programming problem that has turned out to be a standout amongst the most contemplated problem. The (KP) is a traditional combinatorial issue used to demonstrate various present day conditions. Since Balas and Zemel [1] twelve years prior presented the hypothetical core problem as a proficient method for taking care of the Knapsack Problem, all the most effective algorithms have been instituted on this thought with a place in group of NP-hard problems, implying that it is in all respects improbable that we ever can devise polynomial algorithms for these problems [21]. Since the Knapsack Problem just has one imperative, and the coefficients are numbers, it is viewed as one of the less difficult NP-Problems. It is effectively comprehended by numerous individuals who have not contemplated different problems in hypothetical software engineering.

There are a few variations and extensions to the knapsack problem. For instance, the "subset sum problem" is a Knapsack Problem in which an object's benefit is directly relative to its weight. Another variation is the "multidimensional knapsack problem". This problem considers a limitation in addition to the weight on the problem set. For instance, when you pack your knapsack, you should think about the heaviness of your items, yet in addition to their volume. A third, case of a Knapsack Problem variation is the "quadratic knapsack problem". The Knapsack Problem and huge numbers of its variations have been exhibited as an approach to model connected problems. For instance, in [10] the authors record a few uses for a variation of the 0-1 Knapsack, called the multidimensional knapsack problem. The following sub-sections are going to discuss the two common types of knapsack.

A. 01 Knapsack Problem

Knapsack problem is one of the renowned snags in combinatorial optimization [26]. The 0-1 of Knapsack problem has been considered widely amid from the previous four decades due to the circumstances in showing up with frequent genuine spaces through useful significance. In spite of the fact that it is NP-completeness, numerous algorithms have been recommended that exhibit great behavior in the mediocre cases [17]. As a consequence, various individuals appreciate problem illuminating and a vast gathering of puzzlement that can be found in various literature. However, employing intelligent optimization algorithms to take care of the optimization issues is an interesting subject and has numerous feasible applications. The optimization issues can be isolated into [13] taking care of function optimization matters where the function esteem is in the miniature estimation of a function. In some ongoing considerations, Particle Swarm Optimization - Genetic (PSO-GA) algorithm [13], counterfeit Honey Bee State algorithm (HBS) [12] and Gravitational Search Algorithm GA-GSA [17] are used independently to take care of compelled optimization issues. Consequently, the knapsack problem becomes the route of the mortar towards combinatorial optimization problem.

For 0-1 knapsack, the item can be chosen just once where knapsack is capacity bounded. The problem can be mathematically formulated as follows:

$$\begin{gathered} Maximize\ z = \sum_{i=1}^{n} P_i X_i \\ Subject\,to \\ \sum_{i=1}^{n} W_i X_i \leq C, \\ X_i \epsilon (0,1),\ i = 1,2,...,n, \end{gathered} \tag{1}$$

where n is the number of items, P_i is profit associated with each item while W_i is the weight associated with each item. Z is the total profit, C is the capacity of the knapsack and X_i is the binary decision variable.

B. Multidimensional Knapsack Problem

The multidimensional knapsack problem (MKP) is a specialization of the 01 knapsack problem and an extraordinary instance of 0-1 integer programming [4]. The objective of a Multidimensional Knapsack Problem (MKP) is to boost the entirety of estimations of the items to be chosen from a predefined set by methods for contemplating over various asset limitations. This problem has been generally contemplated over numerous decades because of both hypothetical interests as well as its wide applications. Additionally, MKP has been extensively conversed because of its theoretical importance and the wide range of applications [8]. The multidimensional knapsack problem has been acquainted with model problems including cutting stock, stacking problems, venture arrangement for the travel industry segment of a creating nation, assignment of databases and processors in a conveyed information preparing [sic], conveyance of vehicles with multiple compartments. From a computational perspective, numerous methodologies have been proposed to unravel the MKPs and the different proposed algorithms can be extensively assembled into two classes; (i) exact algorithms, and (ii) heuristic or meta-heuristic algorithms.

For multiple knapsacks, it is filling of multiple knapsacks simultaneously with n items. The objective is to maximize the total reward of the selected items. Then, the problem can be formulated as follows:

$$\begin{gathered} Maximize\ z = \sum_{j=1}^{n} c_j x_j \\ Subject\,to \\ \sum_{j=1}^{n} a_{ij} x_j \leq b_i,\ i = 1,2,...,m, \\ x_j \epsilon (0,1),\ j = 1,2,...,n \end{gathered} \tag{2}$$

where n is the set of items, x_j is the decision variable associated with item j and c_j is profit associated with each item j. While a_{ij} is the resource requirement of each item. b is the capacity of the resource while m is the number of resource constraints.

III. Firefly Algorithm

Firefly algorithm is a bio-inspired metaheuristic algorithm for optimization problems. It was introduced in 2009 by Yang [30]. The algorithm is inspired by the flashing behavior of fireflies at night. One of the three rules used to construct the algorithm is that all fireflies are unisex, which means any firefly can be attracted to any other brighter one. The second rule is that the brightness of a firefly is determined from the encoded objective function. The last rule is that attractiveness is directly proportional to brightness but decreases with distance, and a firefly will move towards the brighter one, and if there is no brighter one, it will move randomly [30]. The algorithm is stated below in Algorithm 1.

A. Proposed Algorithm: Firefly Algorithm with Genetic Algorithm

The proposed algorithm uses Firefly algorithm (FA) with the Genetic algorithm (GA). The implementation uses FA algorithm for the first half of the iteration and then GA is used in second half. FA-GA Cycles - The two methods are implemented in separate cycles, such that the resulting algorithm has two cycles: in the first, the FA method is implemented and in the second the GA. This strategy halves the time complicity of both the algorithms. During first half

Algorithm 1: Firefly Algorithm

```
Step 1: Objective function: $f(x)$
Step 2: Generate an initial population of fireflies. $x_i(i = 1, 2, ...., n)$
Step 3: Formulate light intensity I so that it is associated with $f(x)$
Step 4: Define absorption coefficient $\Upsilon$
while $t <$ MaxGeneration do
    foreach i = 1 : n (all n fireflies) do
        foreach j = 1 : i (n fireflies) do
            if $I_j > I_i$ then
                Vary attractiveness with distance $r$ via exp(-$\Upsilon$r);
                move firefly $i$ towards $j$;
                Evaluate new solutions and update light intensity;
            else
                nothing
            end
        end
    end
    Rank fireflies and find the current best;
end
Post-processing the results and visualization;
```

of the iterations, FA searches for best or optimal solutions and in the later stage the best solutions are inserted into the GA algorithm to search further. This way it allows the algorithm to avoid getting stuck at local minimum as another technique is used post half max iteration.

In Algorithm 2, the implementation of the proposed method is shown.

Algorithm 2: Proposed Algorithm

```
Step 1: Objective function: $f(x)$
Step 2: Generate an initial population of fireflies/Population.
  $x_i(i = 1, 2, ...., n)$
Step 3: Formulate light intensity $I$ so that it is associated with $f(x)$
Step 4: Define absorption coefficient $\Upsilon$ and Mutation rate
while $t <$ MaxGeneration/2 do
    foreach i = 1 : n (all n fireflies) do
        foreach j = 1 : i (n fireflies) do
            if $I_j > I_i$ then
                Vary attractiveness with distance $r$ via exp(-$\Upsilon$r);
                move firefly $i$ towards $j$;
                Evaluate new solutions and update light intensity;
            else
                nothing
            end
        end
    end
    Rank fireflies and find the current best;
end
while $y <$ MaxGeneration/2 do
    foreach Sub-population do
        foreach Depth of $n$ Generations do
            Select and create new offspring using genetic operators
            Cooperative Evaluation the new offspring
            Add new offspring's to the sub-population
        end
    end
    Rank population
end
Post-processing the results and visualization;
```

The proposed method is shown in Algorithm 2, it uses FA to find the best solution initially and later aided by GA. The important part here is to save the existing best solutions for the next instance. In Step 1 of the given algorithm, the objective function is defined which is later used to initialize the population in terms of the method used in step 2. As for step 3 and 4, the calculation of important variables are done such as light intensity and mutation coefficient for each of the methods used. At the later stage, the population is evaluated in terms of iterations based on FA and GA methods. Mutation is used to get better solution across the field to benefit the entire population. Lastly, the results are processed and captured.

IV. Experimental Setup and Results

This section reports first on the experimental setup and then on results obtained from the experimental runs and gives further insight on the results through figure.

A. Experimental Setup

The basic setup of the algorithm is discussed here. The experiment was executed 50 times (runs) with 1000 iterations. Half iteration each for the two cycles (FA and GA). That means, first 500 iterations were used by FA and rest by GA.

The number of fireflies was kept at 100, gamma was 0.1 while Attraction Coefficient Base Value and Alpha was kept at 1. The Damping Ratio was set at 0.99. All the parameter settings of the firefly are based from the literature.

For GA parameter setting, the Crossover Percentage and Mutation Percentage was kept at 0.4 and 0.8 respectively. Selection pressure was kept at 5 and the number of offspring was kept at 2. Again all the settings are based from the literature. This allows for a controlled environment for comparison.

The benchmark datasets of MKP are selected from OR-Library [3]. The first set contains eight benchmark problems and corresponds to HP and PB [11]. These instances have also been solved by BPSO in [27]. The second set corresponds to weish [24], which contains 38 instances having 30 to 90 items. The third problem set contains 10 instances which correspond to series sento [23] and weing [28] having 28, 60 and 105 number of items.

B. Results

This subsection reports on the performance of the proposed model based on the setup given previously. The Tables I-III shows the details of trial problems as well as the corresponding results and Table IV, compares some of the results with optimal values obtained in [27].

More specifically, the columns display the Example number, Problem Instance, Number of Knapsacks and number of items in each instance. Apart from this, the final four columns show the numerical results such as minimum value attained, median value, the maximum value (Optimal) and mean value of the total runs for each instance. The results are for the proposed algorithm.

In Table I, the results of HP and PB testing problem are shown. In all the 8 instances given, the proposed FAGA method manage to produce optimal fitness for each of the problems. The mean is also closer to optimal value indicating on good performance with each run.

In Table II, the results of WEISH testing problem are shown. In all the 30 instances of WEISH, the proposed FAGA method was able to produce optimal fitness, which can be seen under Max column. The results are similar as previous table. Mean of some of the problems are very close to optimal fitness while some are not especially for hard cases such as WEISH11, WEISH19, WEISH27, WEISH28 and WEISH29.

In Table III, the results of Sento and Weing testing problem are shown. In all the 10 instances shown, the proposed FAGA method was able to produce optimal fitness, which can be seen under Max column. In all the cases, the average optimal value labeled as ”Max” is very close to optimal value, which indicates more success rates in the number of runs.

Table IV, shows comparison of proposed methods performance with existing methods discussed in [27]. It can be seen that FAGA has outperformed as PBPSO has when compared to other variants of PSO namely KBPSO and MBPSO. FAGA gave best average fitness in PB4, Sent1, Sent2 and Weish20. Similarly, PBPSO has outperformed in PB5, PB6, Sent1 and Weish12, which is four datasets.

Figure 1 shows the typical fitness value of the proposed method on PB4 for 100 iterations while Figure 2 and Figure 2 shows on 1000 iterations for PB4 and SENTO dataset respectively. It clearly shows that with the proposed method, convergence is possible in less than 100 iterations.

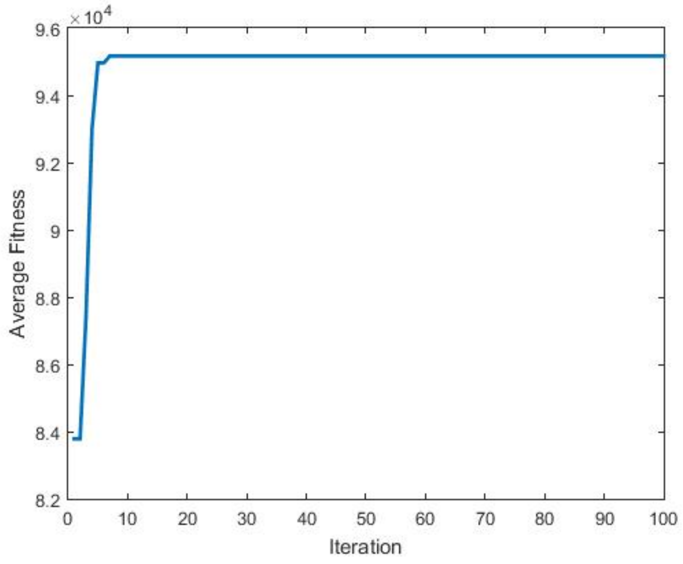

Fig. 1. The average fitness value of FAGA on PB4 dataset for 100 iterations.

V. Discussions

The performance for the proposed method with multi knapsack problem will be discussed in this section.

It has been seen in all the datasets that the proposed method FAGA was able to obtain the best known solution for all benchmark test problems. For all the datasets, the performance of the algorithm converged better as the number of iterations increased. The two methods in two cycles helped to direct the solution to the best optimal values. This allowed for the better spread of individuals as GA and FA worked together to improve the solution.

TABLE I
RESULTS OF PROBLEM SET I HP AND PB TESTING PROBLEMS OF MULTIDIMENSIONAL KNAPSACK PROBLEM.

Example	Instance	Best Known	No. Knapsacks	No. Items	Min	Median	Max	Mean
1	PB1	**3090**	4	28	3006	3060	**3090**	3061.32
2	PB2	**3186**	4	35	3084	3167	**3186**	3154.80
3	PB4	**95168**	4	27	90615	94461	**95168**	93595.24
4	PB5	**2139**	4	34	2085	2102	**2139**	2106.88
5	PB6	**776**	2	29	694	723.00	**776**	738.00
6	PB7	**1035**	10	20	1006	1024	**1035**	1024.00
7	HP1	**3418**	30	40	3354	3404	**3418**	3395.92
8	HP2	**3186**	30	37	3092	3164	**3186**	3156.40

TABLE II
RESULTS OF PROBLEM SET II WEISH TESTING PROBLEMS OF MULTIDIMENSIONAL KNAPSACK PROBLEM.

Example	Instance	Best Known	No. Knapsacks	No. Items	Min	Median	Max	Mean
1	WEISH01	**4554**	5	30	4480	4554	**4554**	4545.96
2	WEISH02	**4536**	5	30	4504	4536	**4536**	4534.12
3	WEISH03	**4115**	5	30	3987	4115	**4115**	4106.00
4	WEISH04	**4561**	5	30	4505	4561	**4561**	4558.76
5	WEISH05	**4514**	5	30	4451	4514	**4514**	4506.44
6	WEISH06	**5557**	5	40	5529	5557	**5557**	5549.28
7	WEISH07	**5567**	5	40	5365	5550	**5567**	5545.64
8	WEISH08	**5605**	5	40	5517	5603	**5605**	5594.20
9	WEISH09	**5246**	5	40	5046	5246	**5246**	5215.76
10	WEISH10	**6339**	5	50	6092	6338	**6339**	6310.24
11	WEISH11	**5643**	5	50	5396	5605	**5643**	5571.00
12	WEISH12	**6339**	5	50	6121	6338	**6339**	6301.00
13	WEISH13	**6159**	5	50	6012	6159	**6159**	6121.84
14	WEISH14	**6954**	5	60	6730	6923	**6954**	6904.36
15	WEISH15	**7486**	5	60	7376	7442	**7486**	7442.54
16	WEISH16	**7289**	5	60	7128	7272	**7289**	7253.68
17	WEISH17	**8633**	5	60	8602	8624	**8633**	8626.24
18	WEISH18	**9580**	5	70	9477	9565	**9580**	9556.20
19	WEISH19	**7698**	5	70	7333	7598	**7698**	7580.24
20	WEISH20	**9450**	5	70	9264	9410	**9450**	9400.12
21	WEISH21	**9074**	5	70	8947	9025	**9074**	9034.08
22	WEISH22	**8947**	5	80	8632	8857	**8947**	8856.72
23	WEISH23	**8344**	5	80	7989	8212	**8344**	8203.71
24	WEISH24	**10220**	5	80	10146	10215	**10220**	10204.92
25	WEISH25	**9939**	5	80	9730	9908	**9939**	9889.32
26	WEISH26	**9584**	5	90	9414	9506	**9584**	9502.18
27	WEISH27	**9819**	5	90	9473	9702	**9819**	9683.10
28	WEISH28	**9492**	5	90	6944	9360	**9492**	9163.18
29	WEISH29	**9410**	5	90	8939	9335	**9410**	9262.60
30	WEISH30	**11191**	5	90	11127	11172	**11191**	11169.72

TABLE III
RESULTS OF PROBLEM SET III SENTO AND WEING TESTING PROBLEMS OF MULTIDIMENSIONAL KNAPSACK PROBLEM.

Example	Instance	Best Known	m	n	Min	Median	Max	Mean
1	SENTO1	**7772**	30	60	7235	7680	**7772**	7695.90
2	SENTO2	**8722**	30	60	8660	8699	**8722**	8698.60
3	WEING1	**141278**	2	28	140543	141278	**141278**	141115.60
4	WEING2	**130883**	2	28	129183	130883	**130883**	130759.40
5	WEING3	**95677**	2	28	93188	94908	**95677**	94578.16
6	WEING4	**119337**	2	28	119088	119337	**119337**	119297.16
7	WEING5	**98796**	2	28	94628	98796	**98796**	97229.00
8	WEING6	**130623**	2	28	129283	130233	**130623**	130362.20
9	WEING7	**1095445**	2	105	1090738	1094945	**1095445**	1094250.16
10	WEING8	**624319**	2	105	623118	624319	**624319**	624117.16

TABLE IV
RESULTS FOR COMPARING PROPOSED METHOD TO ALGORITHMS REPORTED IN [27] ON SOME TESTING KNAPSACK PROBLEMS

Instance	Best Known	Algorithm	Best Fitness	Average Fitness
PB4	95168	KBPSO	95168	91879.15
		MBPSO	95168	92419
		PBPSO	95168	93114.1
		FAGA	95168	**93595.24**
PB5	2139	KBPSO	2139	2131.1
		MBPSO	2139	2110.9
		PBPSO	2139	**2134.45**
		FAGA	2139	2106.88
PB6	776	KBPSO	776	746.95
		MBPSO	776	708.60
		PBPSO	776	**752.85**
		FAGA	776	738.00
Sent1	7772	KBPSO	7676	77562.4
		MBPSO	7762	7683.55
		PBPSO	**7772**	**7695.90**
		FAGA	**7772**	**7695.90**
Sent2	8722	KBPSO	8655	8603.5
		MBPSO	8711	8651
		PBPSO	**8722**	8671.1
		FAGA	**8722**	**8698.60**
Weish12	6339	KBPSO	6339	6295.1
		MBPSO	6339	6317.05
		PBPSO	6339	**6331.75**
		FAGA	6339	6301.00
Weish20	9450	KBPSO	9146	9092.05
		MBPSO	9445	9352.95
		PBPSO	**9450**	9362.05
		FAGA	**9450**	**9400.12**

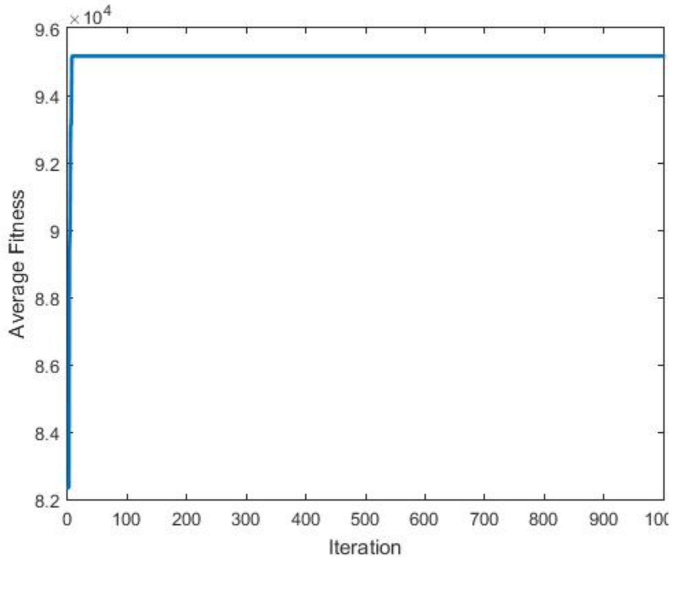

Fig. 2. The average fitness value of FAGA on PB4 dataset for 1000 iterations.

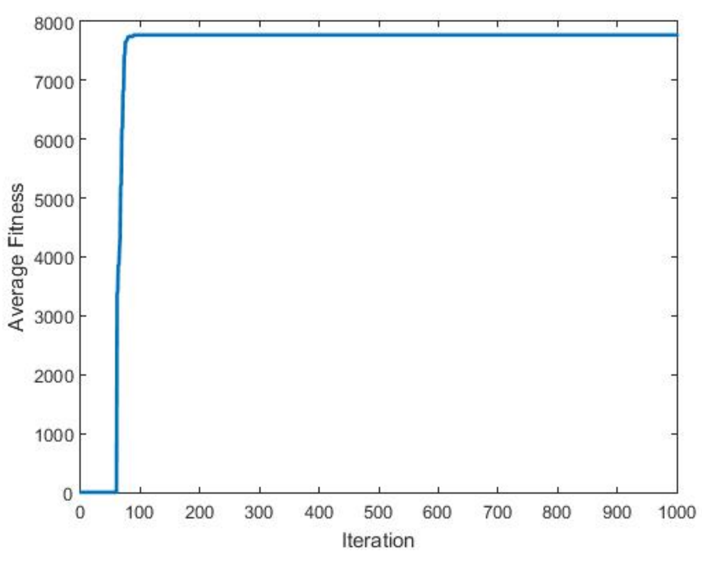

Fig. 3. The average fitness value of FAGA on Sento dataset for 1000 iterations.

In Problem Set I, II and III, it was seen that the FAGA method was able to obtain the best known optimal solutions for the HP and PB, Weish and Sento and Weing testing problems. Even the minimum fitness value, median and mean are closer to the maximum fitness value. The GA method has helped the FA to perform better in terms of solution convergence and to avoid local optimum.

Some of the results of the benchmark testing problems were compared with the existing method proposed in [27]. Proposed method outperformed in the three of the cases while PBPSO method outperformed in three as well and had the same results in Sent1. The performance indicates FAGA or

PBPSO method is a good solution for this time of problems. To further investigate the algorithm, the algorithm outperformed in all the datasets in terms of the other two methods mentioned namely KBPSO and MBPSO. This shows that splitting two algorithms into two different cycles had been very beneficial.

By observing Figure 1, further analysis can be done on the performance of the proposed methods. The figure clearly shows how fast the algorithm converged to the best known fitness for PB4 dataset. It was just less than 10 runs that the optimal value was found, noting that the number of iteration was kept at 100.

If we tend to look at the results again, the proposed hybrid of FA and GA has done well in terms of the maximum fitness function and average fitness function. These results are proof of concept and at the preliminary level as more in-depth analysis needs to be done such as success rate, mean percentage error, etc. to allow the algorithm to solve such problems in the nip of time.

VI. Conclusion

This research was intended to combine FA with GA algorithm to solve Multi Knapsack problems where two cycles are used to solve the given problem. After a successful run on some benchmark testing problems, it can be said that the proposed FAGA model works well in terms of obtaining the optimal values of a given dataset. Even the problems with datasets are difficult to solve when it comes to multi-objective, by proposing an algorithm based on hybridization and state-of-the-art technique we would be able to give insights in evolutionary algorithms for new researchers. For future work, the algorithm will be changed in terms of the two variants. The first variant would combine FA-GA algorithm. Both methods are implemented in a common cycle, more precisely, the GA method is incorporated in the cycle corresponding to the FA method. The idea is to use both methods for a reciprocal improvement of the results produced individually. In other words, after the construction of solutions by every firefly (in each cycle), the GA method is used to improve and/or to find new solutions, which will provide new clues to the fireflies in the following cycles. Secondly, the GA and FA combination would be used where it corresponds to an extension of the FA-GA combination, where the GA is used for the construction of the solution and FA would be used to improve the solution. Lastly, the method would be applied to some real world problems associated with Knapsack problems.

References

[1] Balas, E., Zemel, E.: Facets of the knapsack polytope from minimal covers. SIAM Journal on Applied Mathematics **34**(1), 119–148 (1978). https://doi.org/10.1137/0134010, https://doi.org/10.1137/0134010

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

[3] Beasley, J.E.: Orlib operations research library (2005), http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html

[4] Beaujon, G.J., Marin, S.P., McDonald, G.C.: Balancing and optimizing a portfolio of r&d projects. Naval Research Logistics (NRL) **48**(1), 18–40. https://doi.org/10.1002/1520-6750(200102)48:1¡18::AID-NAV2¿3.0.CO;2-7, https://onlinelibrary.wiley.com/doi/abs/10.1002/1520-6750%28200102%2948%3A1%3C18%3A%3AAID-NAV2%3E3.0.CO%3B2-7

[5] Dantzig, G.B.: Discrete-variable extremum problems. Operations research **5**(2), 266–288 (1957)

[6] Davis, L.: Handbook of genetic algorithms (1991)

[7] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., Fast, A.: Nsga-ii. IEEE transactions on evolutionary computation **6**(2), 182–197 (2002)

[8] Djannaty, F., Doostdar, S.: A hybrid genetic algorithm for the multidimensional knapsack problem. International Journal of Contemporary Mathematical Sciences **3**(9), 443–456 (2008)

[9] Dorigo, M., Birattari, M.: Ant colony optimization. Springer (2010)

[10] Fréville, A.: The multidimensional 0-1 knapsack problem: An overview. European Journal of Operational Research **155**, 1–21 (2004)

[11] Frville, A., Plateau, G.: Hard 0-1 multiknapsack test problems for size reduction methods. Investigation Operativa **1**(3), 251–270 (1990), www.scopus.com, cited By :40

[12] Garg, H.: Optimization problems using an artificial bee colony algorithm (2013)

[13] Garg, H.: A hybrid pso-ga algorithm for constrained optimization problems. Appl. Math. Comput. **274**(C), 292–305 (Feb 2016). https://doi.org/10.1016/j.amc.2015.11.001, http://dx.doi.org/10.1016/j.amc.2015.11.001

[14] Glover, F., Laguna, M.: Tabu search. In: Handbook of Combinatorial Optimization, pp. 3261–3362. Springer (2013)

[15] Kellerer, H., Pferschy, U., Pisinger, D.: Multidimensional knapsack problems. In: Knapsack problems, pp. 235–283. Springer (2004)

[16] Kennedy, J.: Particle swarm optimization. Encyclopedia of machine learning pp. 760–766 (2010)

[17] Lagoudakis, M.G.: The 0-1 knapsack problem – an introductory survey. Tech. rep. (1996)

[18] Lawler, E.L., Lenstra, J.K., Kan, A.R., Shmoys, D.B., et al.: The traveling salesman problem: a guided tour of combinatorial optimization, vol. 3. Wiley New York (1985)

[19] Lust, T., Teghem, J.: The multiobjective multidimensional knapsack problem: a survey and a new approach. International Transactions in Operational Research **19**(4), 495–520 (2012)

[20] Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Courier Corporation (1998)

[21] Pisinger, D.: Core problems in knapsack algorithms. Operations Research **47**, 570–575 (1994)

[22] Reeves, C.R.: Modern heuristic techniques for combinatorial problems. advanced topics in computer science. Modern Heuristic Techniques for Combinatorial Problems: Advanced Topics in Computer Science (1995)

[23] Senju, S., Toyoda, Y.: An approach to linear programming with 0-1 variables. Management Science pp. B196–B207 (1968)

[24] Shih, W.: A branch and bound method for the multiconstraint zero-one knapsack problem. Journal of the Operational Research Society **30**(4), 369–378 (Apr 1979). https://doi.org/10.1057/jors.1979.78, https://doi.org/10.1057/jors.1979.78

[25] Smith-Miles, K., Lopes, L.: Measuring instance difficulty for combinatorial optimization problems. Computers & Operations Research **39**(5), 875–889 (2012)

[26] Song, Y., Zhang, C., Fang, Y.: Multiple multidimensional knapsack problem and its applications in cognitive radio networks. In: MILCOM 2008 - 2008 IEEE Military Communications Conference. pp. 1–7 (Nov 2008). https://doi.org/10.1109/MILCOM.2008.4753629

[27] Wang, L., Wang, X., Fu, J., Zhen, L.: A novel probability binary particle swarm optimization algorithm and its application. Journal of software **3**(9), 28–35 (2008)

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

[29] Wolsey, L.A., Nemhauser, G.L.: Integer and combinatorial optimization. John Wiley & Sons (2014)

[30] Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. arXiv preprint arXiv:1003.1409 (2010)