Iraqi Journal of Statistical Science (30) 2019
pp[55-72]

Nature-inspired optimization algorithms in knapsack problem: A review

Ghalya Tawfeeq Basheer* ghalia.tawfeek@uomosul.edu.iq
Dr.Zakariya Yahya Algamal** zakariya.algamal@uomosul.edu.iq

Abstract:

Meta-heuristic algorithms have become an arising field of research in recent years. Some of these algorithms have proved to be efficient in solving combinatorial optimization problems, particularly knapsack problem. In this paper, four meta-heuristic algorithms are presented particle swarm optimization, firefly algorithm, flower pollination algorithm and monarch butterfly optimization in solving knapsack problem as example of NP-hard combinational optimization problems. Based on twenty 0-1 knapsack problem instances, the computational results demonstrated that the binary flower pollination algorithm has the ability to find the best solutions in reasonable time.

Keywords: Combinatorial optimization problem; NP-hard problem; 0-1 knapsack problem; Meta-heuristic algorithms.

خوارزميات الأَمثلية المستوحاة من الطبيعة لحل مسألة حقيبة الظهر: مراجعة مقال

الملخص:

أصبحت الخوارزميات المستوحاة من الطبيعة لها مجالٌ واسعٌ للبحث في السنوات الاخيرة. وبعض هذه الخوارزميات اثبتت كفاءتها في حل مسائل الامثلية التوافقية خاصة مسألة حقيبة الظهر وفي هذا البحث استعرضنا مفاهيم أربع خوارزميات هي : particle swarm optimization, firefly algorithm, flower pollination algorithm and monarch butterfly optimization .

لحل مسألة حقيبة الظهر كمثال على مسائل الأمثلية التوافقية الصعبة NP-hard problem. وبالاستناد الى 20 حالة من مسألة حقيبة الظهر وبأحجام مختلفة أظهرت النتائج الحسابية ان خوارزمية binary flower pollination algorithm لديها القدرة على إيجاد أفضل الحلول في وقت معقول مقارنة بالخوارزميات المستخدمة الأخرى.

*Department of operations research and intelligent techniques, University of Mosul, Mosul, Iraq.
**Department of Statistics and Informatics, University of Mosul, Mosul, Iraq

Received date:2 /4 /2019 Accepted date: 19 /5 / 2019 Published data 1/12/ 2019

1. Introduction

Combinatorial optimization "problem is a mathematical study of finding optimal solution from a finite set of objects. The popularity of combinatorial optimization problems comes from the fact that the objective function and constraints in many real-world problems have a different nature (nonlinear, nonanalytic, etc.), while the search space is finite. In such problems, exact methods are impractical in finding an optimal solution because the run time is increasing exponentially with the problem size. Therefore, interest in the application of the meta-heuristic algorithms has become a necessary to solve these problems and obtain the results" in a reasonable time [El-Ghazali 2009, Beheshti et al. 2012].

In recent years, the "nature inspired meta-heuristic algorithms have been used successfully for solving hard and complex problems in real-world problems. The meta-heuristic algorithms are stochastic algorithms inspired by the behavior of different species in nature Pirlot (1992) and Osman (1995) define meta-heuristic as follows "A meta-heuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space using learning strategies to structure information in order to find efficiently near-optimal solutions" [Osman and Kelly 1996]. The meta-heuristic algorithms use search strategies and concepts inspired from nature to explore several regions of the search space more effectively and focusing on some likely regions of the search space. Every meta-heuristic algorithm consists of a set of initial population or initial solutions, the sequence of solutions is then examined step by step based on randomization and some specified rules to reach the optimal solution. These algorithms have the ability to deal with many of optimization problems because of its simplicity and flexibility" [Yang 2014a, Yang 2014b, Siddique and Adeli 2015, Bhattacharjee and Sarmah 2016, Faris et al. 2017, Lazim et al. 2017].

The aim of this paper is to investigate the effectiveness of the nature inspired meta-heuristic algorithms when dealing with a combinatorial optimization problem such as 0-1 knapsack problem.

2. Related work

Knapsack problem is "a combinatorial optimization problem and NP-hard problem. In such problems, there is no effective algorithm to solve all their instances. These problems need alternative methods because exact methods usually cannot deal with the large size of these problems [Yang 2014a, Yang 2015]. There are many meta-heuristic algorithms applied in solving a knapsack problem such as genetic algorithm (GA), particle

swarm optimization (PSO), ant colony optimization (ACO), harmony search (HS),and whale optimization algorithm ,and so on.

Particle swarm optimization is one of the meta-heuristic algorithms that has been applied to many combinatorial optimization problems, namely knapsack problem. Bansal and Deep [2012] proposed a new modified binary particle swarm optimization for solving 0-1 knapsack problem and multidimensional knapsack problem, and introduced a new probability function which maintains the diversity in the particle swarm.

The meta-heuristic firefly algorithm was developed by Yang [2008]. Firefly algorithm mimics the behavior of fireflies which is based on flashing and attraction properties of fireflies. Zouache et al. [2015] proposed a new hybrid algorithm that combines firefly algorithm and particle swarm optimization and use the basic concepts of quantum computing to ensure a better solution diversity. The proposed algorithm has been tested on 0-1 knapsack problem and multidimensional knapsack problem. Feng et al. [2017] proposed a novel global firefly algorithm for tackling randomized time-varying knapsack problem .

The flower pollination algorithm is a meta-heuristic algorithm which mimic the pollination characteristics of flowers in plant. Flower pollination algorithm was proposed by Yang [2012] for solving single objective optimization problems. Yang et al. [2014] extended flower pollination algorithm for solving multi-objective optimization problems. Abdel-Basset et al. [2018] proposed a binary version of flower pollination algorithm for solving both small and large scale knapsack problem and the sigmoid function is used to convert continuous values into binary.

The monarch butterfly optimization algorithm is a new meta-heuristic algorithm developed by Wang et al. [2015] for solving continuous optimization problems. It mimics the migration behavior of monarch butterflies in nature . Feng et al. [2015] introduced a novel binary monarch butterfly optimization for solving knapsack problem, where the repair operator is based on greedy optimization algorithm.

Rizk-Allah and Hassanien [2017] considered a novel binary bat algorithm for tackling 0-1 knapsack problem with two phases : binary bat algorithm and local search scheme. The V-shaped transfer function is used to convert continuous values into binary values. Zhou et al. [2016] developed a binary monkey algorithm to deal with 0-1 knapsack problem. The greedy algorithm is considered to correct the infeasible solutions and to improve the feasible solutions. Changdar et al. [2013] introduced a novel ant colony optimization algorithm in fuzzy environment for tackling a knapsack problem, where the profit and weight are considered fuzzy and take as trapezoidal fuzzy number.

3. Knapsack problem

Knapsack problem is "one of the NP-hard combinatorial optimization problems which has been widely studied in operation research. Knapsack problem consists of a set of n items, each item i has a profit c_i, weight w_i and maximum weight capacity M. The objective is to maximize the total profit of the selected items in the knapsack such that the total weights of these items are achieved" ([Abdel-Basset et al. 2017b, Cao et al. 2017]) Eq.(2). Mathematically, the knapsack problem can be written as:

$$f(x) = \sum_{i=1}^{n} c_i x_i \quad (1)$$

s.t.

$$\sum_{i=1}^{n} w_i x_i \le M \quad (2)$$

where

$$x_i = \begin{cases} 1 & \text{if item i is selected} \\ 0 & \text{otherwise} \end{cases}$$

the penalty function is used to deal with the knapsack problem according to the following equation:

$$Min\phi(x) = -f(x) + \lambda Max(0, h) \quad (3)$$

where $h = \sum_{i=1}^{n} w_i x_i - M$ and λ represent the penalty coefficient. In this paper λ is set to 10^{10} for all tests. The penalty function can be described in Algorithm 1.

Penalty function

- **input** solution x_i
- Calculate total weight of x_i by $\left(\sum_{i=1}^{n} w_i x_i\right)$
- **if** $\left(\sum_{i=1}^{n} w_i x_i \le M\right)$
- $\phi(x) = -\sum_{i=1}^{n} c_i x_i$
- **else**
- $\phi(x) = -\sum_{i=1}^{n} c_i x_i + \lambda\left(\sum_{i=1}^{n} w_i x_i - M\right)$
- **end**

Algorithm 1: Penalty function

A repair operator treats "the infeasible solutions which violates the constraint in Eq. (2) by converting them into feasible solutions and also improve the feasible solutions. The repair operator algorithm can be

applied by two stage. The first stage is to convert the infeasible solution into feasible by taking out the items of the lower c_i / w_i ratio so as the constraint in Eq. (2) is not to exceed the knapsack capacity. The second stage is to improve the feasible solution by adding the items of the high c_i / w_i ratio to the knapsack with keeping" of the constraint.

4. Particle Swarm Optimization (PSO)

Particle swarm optimization is one of the meta-heuristic algorithms which was proposed by Kennedy and Eberhart [1995] for "solving continuous optimization problem. Particle swarm optimization is inspired by swarm behavior in nature like birds and fish schooling. The particle swarm optimization algorithm started with number of particles N which fly in the search space to search the best solution. Each particle i has a position $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ and velocity $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$ in D-dimensional search space where $i = 1, 2, ..., N$. Each particle is updated through each iteration based on two values: the first value is the best solution $(pbest)$ which has been obtained by the particle, and the second value is the current best value $(Gbest)$ which has been obtained in the swarm " ([Mirjalili and Lewis 2013, Chih et al. 2014, Haddar et al. 2015]). The new velocity and position vectors in each iteration are updated according to the following equations :

$$v_i^{t+1} = W\, v_i^t + c_1.r_1.\left(Pbest_i^t - x_i^t\right) + c_2.r_2.\left(Gbest^t - x_i^t\right) \quad (4)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (5)$$

where t is the current iteration in the algorithm, w refers to the inertia weight. c_1 , c_2 refer to the accelerated variables or learning factors and r_1 , r_2 are the random numbers $\in [0,1]$ obtained from uniform distribution. The pseudo code of the particle swarm optimization can be shown in Algorithm 2.

Particle Swarm Optimization Algorithm

```
Begin
    - Define the objective function max or min f (x)
    - Initialize the population with random positions and velocities
    - Evaluate each particle in the population
    - Find the (pbest)
    - while (stopping criterion)
    -   for i=1:n
    -     Update the velocity using Eq. (4)
    -     Update the position using Eq. (5)
    -     Evaluate objective function at new position
    -     Find the (Gbest)
    -   end for
    -   Find current (pbest)
    - end while
End
```

Algorithm 2: The pseudo code of the particle swarm optimization algorithm

Particle swarm optimization has been firstly "proposed for continuous optimization problems where velocity and position are real values. Therefore, it is not able to tackle a binary optimization problem such as knapsack problem. Kennedy and Eberhart [1997] developed a new version of particle swarm optimization, called binary particle swarm optimization

(BPSO) to deal with problems with binary search space. In BPSO the position of particles takes the values 0 or 1. The velocity updating remains as defined in Eq. (4), then sigmoid function is used to transform the real values to the binary values according to the following equations [Haddar et al. 2016]:

$$x_i^t = \begin{cases} 1 & if \quad S\left(v_i^t\right) > r \\ 0 & OW \end{cases} \tag{6}$$

$$S\left(v_i{}^t\right) = \frac{1}{1+e^{-v_i^t}} \tag{7}$$

Where r is a random number $\in [0,1]$, $S(v)$ is the sigmoid function.

5. Firefly Algorithm (FA)

Firefly algorithm is a meta-heuristic algorithm "developed originally by Yang [2008] for solving continuous optimization problems. Firefly algorithm is a simulated behavior of fireflies which based on flashing and attraction properties of fireflies. There are two main points in firefly algorithm: variation of light intensity and formulation of attractiveness. The brightness of firefly depends on the objective function. The attractiveness of firefly is proportional to brightness. Thus, for any two flashing fireflies the less bright one moves towards brighter one, while it will move randomly when there is no brighter one than a specific firefly". The attractiveness β of a firefly with the distance r can be defined as:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{8}$$

where r is the distance between two fireflies, γ is a light absorption coefficient and β_0 is the attractiveness at $r = 0$.

The distance between two fireflies i and j is computed using the Euclidean distance:

$$r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k} \right)^2} \tag{9}$$

where $x_{i,k}$ is the kth component of the ith firefly. The movement of a firefly i attracted by another firefly j that is brighter is computed as:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2} \left(x_j^t - x_i^t \right) + \alpha \left(rand - 0.5 \right) \tag{10}$$

where α is a parameter which controls the step and rand is a random number $\in [0,1]$. The pseudo code of the firefly algorithm can be expressed in Algorithm 3.

Firefly Algorithm

Begin

- Define the objective function max or min $f(x)$, the parameters α , γ
- Generate an initial population of n fireflies.
- Evaluate each firefly in the population
- Define light intensity I_i for each firefly based on objective function
- **while** (stopping criterion)
- **for** i=1:n
- **for** j=1:n
- **if** $I_i < I_j$
- Move firefly i towards j using Eq. (8)
- **end if**
- Vary attractiveness via $\exp(-\gamma r^2)$
- Evaluate objective function at new solution and update light intensity
- **end for** j
- **end for** i
- Rank the fireflies and find the current global best
- **end while**

End

Algorithm 3: The pseudo code of the firefly algorithm

In binary firefly algorithm we use the sigmoid function to convert the continuous values of position into binary values :

$$x_i^t = \begin{cases} 1 & if \quad S(x_i^t) > r \\ 0 & OW \end{cases} \tag{11}$$

Where r is a random number $\in [0,1]$, $S(x)$ is the sigmoid function.

$$S(x_i{}^t) = \frac{1}{1+e^{-x_i^t}} \tag{12}$$

6. Flower Pollination Algorithm (FPA)

Yang [2012] proposed "a new algorithm for global optimization called flower pollination algorithm. It is a meta-heuristic algorithm that mimics the nature, inspired of the pollination process in flowers.

The pollination in flowers can be take two forms: biotic pollination and abiotic pollination. In the first type, the pollen is transferred by a pollination like insects and animals. While the second form is based on wind and diffusion in the water.

Pollination in flowers can be divided into self-pollination and cross-pollination. Self-pollination is transferring the pollens from one flower to the same flower or different flowers in the same plant. Cross-pollination is transferring the pollens from one flower to another flower of a different plant. A flower and its pollen represent a solution of the optimization problem. In the flower pollination algorithm four basic rules are used [Yang 2012, Abdel-Basset et al. 2017a, Bozorg-Haddad 2018]:

1. The global pollination includes biotic and cross-pollination, the pollinators move in a way which follows a lévy flight distribution.
2. The local pollination includes abiotic and self-pollination.

3. Flower constancy can be considered as the reproduction probability that is proportional to the similarity of two flowers involved.
4. The interaction or switching of local pollination and global pollination can be controlled by a switch probability $p \in [0,1]$.

Rules 1 and 3 can be expressed mathematically as:

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)\left(x_i^t - g^*\right) \tag{13}$$

where x_i^t is the solution vector or the pollen i at iteration t, g^* is the current best solution that is found at the current iteration, γ is a scaling factor to control the step size, $L(\lambda)$ is the step size "in the lévy flights which is representing the strength of the pollination. Since pollinators move over a long distance with various distance steps, a lévy flight can be used to mimic this behavior. That is, $L > 0$ from a lévy distribution as

$$L \sim \frac{\lambda\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi}\left(\frac{1}{S^{1+\lambda}}\right)\left(S >> S_0 > 0\right) \tag{14}$$

Yang [2012] proposed that $\Gamma(\lambda)$ is the standard gamma function and $\lambda = 1.5$. This distribution is valid for large steps $S > 0$. In (1994) Mantegna used the Gaussian distribution for generating the step size S by generating two random numbers U and V " as follows [Abdel-Basset et al. 2017a]:

$$S = \frac{U}{|V|^{1/\lambda}} \qquad U \sim N\left(0,\sigma^2\right),\ V \sim N(0,1) \tag{15}$$

$$\sigma^2 = \left(\frac{\Gamma(1+\lambda)}{\lambda\Gamma\left[(1+\lambda)/2\right]} * \frac{\sin(\pi\lambda/2)}{2^{\frac{\lambda-1}{2}}}\right)^{1/\lambda} \tag{16}$$

For local pollination, rules 2 and 3 can be expressed as:

$$x_i^{t+1} = x_i^t + k\left(x_j^t - x_k^t\right) \tag{17}$$

where x_j and x_k are the pollens (solution vectors) from different flowers of the same plant. k is the parameter drawn from uniform distribution in $[0,1]$. To switch between common global pollination to intensive local pollination we used rule 4 [Yang ,2012] suggested that the switch probability or proximity probability is equal to $p = 0.8$ for most applications. The pseudo code of the flower pollination algorithm can be presented" in Algorithm 4.

Flower Pollination Algorithm

Begin

- Define the objective function max or min $f(x)$, and switch probability $p \in [0,1]$
- Initialize the population of n random flowers
- Evaluate each flowers in the population
- Find the best solution
- **while** (stopping criterion)
- **for** i=1:n
- **if** (r<p)
- New solution =global pollination Eq. (13)
- **else**
- New solution = local pollination Eq. (17)
- **end if**
- Evaluate new solution
- If new solution is better , update the population
- **end for**
- Find current best solution
- **end while**

End

Algorithm 4: The pseudo code of the flower pollination algorithm

In binary flower pollination algorithm, "the transfer function is used to convert the continuous values into binary values. In order to build this binary vector a transfer function in Eq. (18) can be used after Eq. (17), in which the new solution is constrained to only binary values:

$$x_i^t = \begin{cases} 1 & if \quad S\left(x_i^t\right) > r \\ 0 & OW \end{cases} \tag{18}$$

Where r is a random number $\in [0,1]$, $S(x)$ is the sigmoid" function.

$$S\left(x_i{}^t\right) = \frac{1}{1+e^{-x_i^t}} \tag{19}$$

7. Monarch Butterfly optimization (MBO)

Wang Deb et al. [2015] proposed a new meta-heuristic "algorithm for continuous optimization problems called monarch butterfly optimization. It is inspired by simulating the migration behavior of the monarch butterflies from northern USA and southern Canada to Mexico every summer.

In MBO algorithm the entire population can be divided into two subpopulations, subpopulation 1 and subpopulation 2 which lived in land 1 and land 2 respectively. The number of monarch butterflies in land 1 and land 2 are (NP1=NP*p) and (NP2=NP-NP1) respectively, where NP is the size of the entire population and p is the proportion of monarch butterflies in subpopulation 1. The monarch butterfly optimization algorithm has two main operators: the migration operator and the butterfly adjusting operator" [Wang et al. 2015, Ghanem and Jantan 2016]:

7.1 Migration Operator

The migration process can be described as follows:

$$x_{i,k}^{t+1} = x_{r_1,k}^t \tag{20}$$

where $x_{i,k}^{t+1}$ is the kth element of x_i at generation $t+1$, which "represents the position of the monarch butterfly i. $x_{r_1,k}^{t}$ is the kth element of x_{r_1} at generation t, which represents the position of the monarch butterfly r_1. Monarch butterfly r_1 is randomly selected from subpopulation 1. If $r \le p$, the element k in the newly generated monarch butterfly is generated by Eq.(20) else if $r > p$ the element k the newly generated monarch butterfly is generated by the following equation:

$$x_{i,k}^{t+1} = x_{r_2,k}^{t} \tag{21}$$

where $x_{r_2,k}^{t}$ is the kth element of x_{r_2} at generation t, that is the newly generated position of the monarch butterfly r_2. Monarch butterfly r_2 is randomly selected from subpopulation 2, where r can be computed as follows:

$$r = rand * peri \tag{22}$$

where $peri$ represents the migration period and $rand$ is a random number in $(0,1)$. Based on the above analyses, the migration operator can be expressed in" Algorithm 5 [Wang et al. 2015, Wang et al. 2016].

```
Migration Operator
Begin
    for i=1:NP1
        for k=1:d (all the elements in ith monarch butterfly)
            r = rand * peri
            if r ≤ p
                selected a monarch butterfly r1 randomly in NP1
                Generate new monarch butterfly x_{i,k}^{t+1} by Eq. (20)
            else
                selected a monarch butterfly r2 randomly in NP2
                Generate new monarch butterfly x_{i,k}^{t+1} by Eq. (21)
            end if
        end for
    end for
End
```

Algorithm 5: Migration Operator

7.2 Butterfly Adjusting Operator

In butterfly adjusting process, "the position of the monarch butterflies in subpopulation 2 is updated by updating all the elements in monarch butterfly j. If $rand \le p$ where $rand$ is a random value in $(0,1)$ then it can be updated as:

$$x_{j,k}^{t+1} = x_{best,k}^{t} \tag{23}$$

where $x_{j,k}^{t+1}$ is the kth element of x_j at generation $t+1$, which represents the position of the monarch butterfly j. $x_{best,k}^{t}$ is the kth element of x_{best} at generation t, that is the best monarch butterfly in land 1 and land 2. On the other hand if the random value $rand > p$, it can be updated as:

$$x_{j,k}^{t+1} = x_{r_3,k}^{t} \tag{24}$$

where $x_{r_3,k}^{t}$ is the k[th] element of x_{r_3} that is randomly selected in subpopulation 2. In addition to this condition, if $rand > BAR$ it can be updated as:

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha\left(dx_k - 0.5\right) \tag{25}$$

where BAR is the butterfly adjusting rate. dx is the walk step of the monarch butterfly j that can be calculated by performing Levy flight :

$$dx = Levy\left(x_j^t\right) \tag{26}$$

α is the weighting factor that can be calculated by following equation:

$$\alpha = S_{\max} / t^2 \tag{27}$$

where $S_{\max}$ is the max walk step that a monarch butterfly individual can move in one step and t is the current generation. The butterfly adjusting operator" can be described in Algorithm 6 [Wang et al. 2015, Wang et al. 2016].

Butterfly Adjusting Operator

Begin
  for j=1:NP2
    for k=1:d (all the elements in ith monarch butterfly)
      if $rand \le p$
        Generate new monarch butterfly $x_{j,k}^{t+1}$ by Eq. (23)
      else
        selected a monarch butterfly r_3 randomly in NP2
        Generate new monarch butterfly $x_{j,k}^{t+1}$ by Eq. (24)
        if $rand > BAR$
          $x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha\left(dx_k - 0.5\right)$
        end if
      end if
    end for
  end for
End

Based on the migration operator and the butterfly adjusting operator, the main steps of MBO algorithm can be described in Algorithm 7 [Wang et al. 2015, Wang et al. 2016].

Monarch Butterfly Optimization Algorithm (MBOA)

Begin
  Step 1: Initialize the population of NP monarch butterfly individuals randomly , set migration period $peri$, the migration ratio p , butterfly adjusting rat BAR , and the max step $S_{\max}$. Set the maximum generation.
  Step 2: Evaluate each monarch butterfly in the population.
  Step 3: while (stopping criterion)
    Sort the monarch butterfly individuals based on fitness.
    Divided butterfly individuals into two subpopulation (land 1 and land 2)
    for i=1:NP1
      Generate new subpopulation by migration operator Algorithm 5.
    end
    for j=1:NP2
      Generate new subpopulation by Butterfly Adjusting Operator Algorithm 6.
    end
    Combine the two newly generated subpopulations into one population .
  end
  Step 4: Find the best solution.
End

Algorithm 7: The pseudo code of the Monarch Butterfly Optimization Algorithm

The standard monarch butterfly optimization algorithm was proposed to "handle a continuous optimization problem. In discrete optimization problems the standard method cannot be applied directly to deal with such a problem. Therefore, the sigmoid function is used to convert the continuous values into binary:

$$x_i^t = \begin{cases} 1 & if \quad S\left(x_i^t\right) > r \\ 0 & OW \end{cases} \tag{28}$$

Where r is a random number $\in [0,1]$, $S(x)$ is the sigmoid" function.

$$S\left(x_i{}^t\right) = \frac{1}{1+e^{-x_i^t}} \tag{29}$$

8. Computational results

In this section twenty 0-1 knapsack problem (KP) instances (kp-1 to kp-20) "taken from [Rizk-Allah and Hassanien 2017, Abdel-Basset et al. 2018] to investigate the performance of the meta-heuristic algorithms (PSO, FA, FPA and MBO) are tested. The four meta-heuristic algorithms (PSO, FA, FPA and MBO) are run 25 independent times for each instance. Table (1) shows the comparison results among four meta-heuristic algorithms" (PSO, FA, FPA and MBO) .

Table 1: Results of 0–1 KP instances

Instance	dimension	Methods	Total profit	Total weight	Best	Mean iterations	Time	solution vector
kp-1	4	BPSO	35	18	35	1	0.006	1101
		BFA	35	18	35	1	0.005	1101
		BFPA	35	18	35	1	0.005	1101
		BMBO	35	18	35	1	0.005	1101
kp-2	4	BPSO	23	11	23	1	0.004	0101
		BFA	23	11	23	1	0.003	0101
		BFPA	23	11	23	1	0.004	0101
		BMBO	23	11	23	1	0.004	0101
kp-3	5	BPSO	130	60	130	1	0.01	11110
		BFA	130	60	130	1	0.009	11110
		BFPA	130	60	130	1	0.008	11110
		BMBO	130	60	130	1	0.008	11110
kp-4	7	BPSO	107	50	107	1.5	0.031	1001000
		BFA	107	50	107	2.01	0.035	1001000
		BFPA	107	50	107	1.08	0.022	1001000
		BMBO	107	50	107	2.15	0.029	1001000
kp-5	10	BPSO	295	269	295	2	0.055	0111000111
		BFA	295	269	295	3.41	0.052	0111000111
		BFPA	295	269	295	1.92	0.051	0111000111
		BMBO	295	269	295	3.26	0.051	0111000111
kp-6	10	BPSO	52	60	52	1	0.058	0011101000
		BFA	52	60	52	1.6	0.050	0011101000
		BFPA	52	60	52	1	0.044	0011101000
		BMBO	52	60	52	1.12	0.046	0011101000

kp-7	15	BPSO	481.0694	354.9608	481.07	1	0.09	001010110111011
		BFA	481.0694	354.9608	481.07	1	0.081	001010110111011
		BFPA	481.0694	354.9608	481.07	1	0.078	001010110111011
		BMBO	481.0694	354.9608	481.07	1	0.080	001010110111011
kp-8	20	BPSO	1025	871	1025	2.1	0.253	11111111101111010111
		BFA	1025	871	1025	3.7	0.198	11111111101111010111
		BFPA	1025	871	1025	1.68	0.106	11111111101111010111
		BMBO	1025	871	1025	2.64	0.134	11111111101111010111
kp-9	20	BPSO	1024	871	1024	2.7	0.463	11111111111110101011
		BFA	1024	871	1024	3.93	0.388	11111111111110101011
		BFPA	1024	871	1024	1.72	0.210	11111111111110101011
		BMBO	1024	871	1024	2.86	0.242	11111111111110101011
kp-10	23	BPSO	9767	9768	9767	5.17	0.805	111111110100000 11000000
		BFA	9767	9768	9767	4.91	0.719	111111110100000 11000000
		BFPA	9767	9768	9767	4.44	0.632	111111110100000 11000000
		BMBO	9767	9768	9767	5.31	0.698	111111110100000 11000000
kp-11	30	BPSO	1437	566	1437	9.78	0.498	11111011111100111 0110101111011
		BFA	1437	566	1437	8.95	0.365	11111011111100111 0110101111011
		BFPA	1437	566	1437	7.8	0.321	11111011111100111 0110101111011
		BMBO	1437	566	1437	8.15	0.374	11111011111100111 0110101111011
kp-12	35	BPSO	1689	650	1689	10.94	0.730	1101111111101011111111 01101110111111
		BFA	1689	650	1689	12.1	0.605	1101111111101011111111 01101110111111
		BFPA	1689	650	1689	7.96	0.581	1101111111101011111111 01101110111111
		BMBO	1689	650	1689	9.51	0.600	1101111111101011111111 01101110111111
kp-13	40	BPSO	1821	819	1821	45	0.793	0111110011111011111101 011111011110111110
		BFA	1821	819	1821	50.02	0.806	0111110011111011111101 011111011110111110
		BFPA	1821	819	1821	37.8	0.725	0111110011111011111101 011111011110111110
		BMBO	1821	819	1821	41.08	0.764	0111110011111011111101 011111011110111110
kp-14	45	BPSO	2033	906	2033	35.4	1.006	111101001111110 111110111111111 111010111111001
		BFA	2033	906	2033	38.16	0.925	111101001111110 111110111111111 111010111111001
		BFPA	2033	906	2033	22	0.843	111101001111110 111110111111111 111010111111001
		BMBO	2033	906	2033	29.57	0.901	111101001111110 111110111111111 111010111111001
kp-15	50	BPSO	2440	873	2440	40.08	2.380	1111100101111111011 0111111101011111111 011111111111

		BFA	2440	873	2440	42.5	2.152	1111100101111111011 0111111101011111111 011111111111
		BFPA	2440	873	2440	29.4	1.056	1111100101111111011 0111111101011111111 011111111111
		BMBO	2440	873	2440	35.19	1.809	1111100101111111011 0111111101011111111 011111111111
kp-16	55	BPSO	2651	1046	2651	867.1	4.237	1110011111011111011 1111010011111111110 01101101110101111
		BFA	2651	1046	2651	894	4.690	1110011111011111011 1111010011111111110 01101101110101111
		BFPA	2651	1046	2651	524.3	3.557	1110011111011111011 1111010011111111110 01101101110101111
		BMBO	2651	1046	2651	698.4	3.901	1110011111011111011 1111010011111111110 01101101110101111
kp-17	60	BPSO	2917	1002	2917	208.6	3.816	11111010110111110110 01111111110111111111 01111011111111101111
		BFA	2917	1002	2917	234.9	3.908	11111010110111110110 01111111110111111111 01111011111111101111
		BFPA	2917	1002	2917	81.88	2.684	11111010110111110110 01111111110111111111 01111011111111101111
		BMBO	2917	1002	2917	195.73	2.991	11111010110111110110 01111111110111111111 01111011111111101111
kp-18	65	BPSO	2818	1317	2818	945.3	4.865	111101011110111011111 011111111111100101111 110011101111111101010
		BFA	2818	1317	2818	971.9	4.947	111101011110111011111 011111111111100101111 110011101111111101010
		BFPA	2818	1317	2818	808.5	4.038	111101011110111011111 011111111111100101111 110011101111111101010
		BMBO	2818	1317	2818	894.1	4.245	111101011110111011111 011111111111100101111 110011101111111101010
kp-19	70	BPSO	3223	1426	3223	897.6	3.726	11111011101011011011111 11010111010111111111100111 11111110111111111111
		BFA	3223	1426	3223	854.4	3.451	11111011101011011011111 11010111010111111111100111 11111110111111111111
		BFPA	3223	1426	3223	885.5	2.372	11111011101011011011111 11010111010111111111100111 11111110111111111111
		BMBO	3223	1426	3223	784.5	2.905	11111011101011011011111 11010111010111111111100111 11111110111111111111
kp-20	75	BPSO	3614	1432	3614	702.2	6.937	01101111101100101111111111 10111111001111011111110111 11111001111111111101101
		BFA	3614	1432	3614	785	7.110	01101111101100101111111111 10111111001111011111110111 11111001111111111101101

		BFPA	3614	1432	3614	125.41	5.201	01101111101100101111111111 10111111001111011111110111 11111001111111111101101
		BMBO	3614	1432	3614	589.7	6.372	01101111101100101111111111 10111111001111011111110111 11111001111111111101101

The computation results for the four algorithms are presented in Table 1. The best solution, mean iteration (average number of iteration for obtaining the optimal solution for all run times of the same instance) and average executed times are used to measure the performance for these algorithms. The average executed times for all algorithms (BPSO, FA, BFPA and BMBO) are (1.53815, 1.52495, 1.1269 and 1.30795 respectively). In order to analyze the results in Table 1 and based on mean iteration and average executed times we can infer the order of the algorithms as follows: (BFPA, BMBO, FA and BPSO). Therefore, the binary flower pollination algorithm gives performance better than other algorithms (BPSO, BFA and BMBO) for solving 0-1 knapsack problem instances. All algorithms (BPSO, BFA, BFPA and BMBO) succeeded in finding the optimal solution for all 0-1 KP instances but BFPA has the least mean iteration and least average executed times compared to other algorithms".

9. Conclusion

In this paper, four meta-heuristic algorithms for solving knapsack problem have been reviewed. Table (1) shows that the BFPA has fast convergence and stability better than other used algorithms. It may be appropriate to suggest that the best algorithm for solving 0-1 knapsack problem is BFPA. Future work includes using this algorithm for solving other combinatorial optimization problems and hybridization with other algorithms.

References

1. Abdel-Basset, M., D. El-Shahat and I. El-Henawy (2018). "Solving 0–1 knapsack problem by binary flower pollination algorithm." Neural Computing and Applications.
2. Abdel-Basset, M., D. El-Shahat, I. El-Henawy and A. K. Sangaiah (2017a). "A modified flower pollination algorithm for the multidimensional knapsack problem_ human-centric decision making." Soft Computing.
3. Abdel-Basset, M., D. El-Shahat and A. K. Sangaiah (2017b). "A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem." International Journal of Machine Learning and Cybernetics **10**(3): 495-514.

4. Bansal, J. C. and K. Deep (2012). "A Modified Binary Particle Swarm Optimization for Knapsack Problems." Applied Mathematics and Computation **218**(22): 11042-11061.
5. Beheshti, Z., S. M. Shamsuddin and S. S. Yuhaniz (2012). "Binary Accelerated Particle Swarm Algorithm (BAPSA) for discrete optimization problems." Journal of Global Optimization **57**(2): 549-573.
6. Bhattacharjee, K. K. and S. P. Sarmah (2016). "Modified swarm intelligence based techniques for the knapsack problem." Applied Intelligence **46**(1): 158-179.
7. Bozorg-Haddad, O. (2018). "Advanced Optimization by Nature-Inspired Algorithms." Springer Nature Singapore Pte Ltd.
8. Cao, J., B. Yin, X. Lu, Y. Kang and X. Chen (2017). "A modified artificial bee colony approach for the 0-1 knapsack problem." Applied Intelligence **48**(6): 1582-1595.
9. Changdar, C., G. S. Mahapatra and R. K. Pal (2013). "An Ant colony optimization approach for binary knapsack problem under fuzziness." Applied Mathematics and Computation **223**: 243-253.
10. Chih, M., C.-J. Lin, M.-S. Chern and T.-Y. Ou (2014). "Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem." Applied Mathematical Modelling **38**(4): 1338-1350.
11. El-Ghazali, T. (2009). "METAHEURISTICSF ROM DESIGN TO IMPLEMENTATION." John Wiley & Sons, Inc.
12. Faris, H., I. Aljarah and S. Mirjalili (2017). "Improved monarch butterfly optimization for unconstrained global search and neural network training." Applied Intelligence **48**(2): 445-464.
13. Feng, Y., G.-G. Wang, S. Deb, M. Lu and X.-J. Zhao (2015). "Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization." Neural Computing and Applications **28**(7): 1619-1634.
14. Feng, Y., G.-G. Wang and L. Wang (2017). "Solving randomized time-varying knapsack problems by a novel global firefly algorithm." Engineering with Computers **34**(3): 621-635.
15. Ghanem, W. A. H. M. and A. Jantan (2016). "Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems." Neural Computing and Applications **30**(1): 163-181.
16. Haddar, B., M. Khemakhem, S. Hanafi and C. Wilbaut (2015). "A hybrid heuristic for the 0–1 Knapsack Sharing Problem." Expert Systems with Applications **42**(10): 4653-4666.

17. Haddar, B., M. Khemakhem, S. Hanafi and C. Wilbaut (2016). "A hybrid quantum particle swarm optimization for the Multidimensional Knapsack Problem." Engineering Applications of Artificial Intelligence **55**: 1-13.
18. Kennedy, J. and R. Eberhart (1995). "Particle swarm optimization." Proceedings of the IEEE International Conference on Neural Network , Perth ,Australia.
19. Kennedy, J. and R. Eberhart (1997). "A discrete binary version of the particle swarm algorithm." IEEE International Conferenceon Computational Cybernetics and Simulation **Vol.5**: 4104–4108.
20. Lazim, D., A. M. Zain, M. Bahari and A. H. Omar (2017). "Review of modified and hybrid flower pollination algorithms for solving optimization problems." Artificial Intelligence Review.
21. Mantegna, R. N. (1994). "Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes". Phys Rev E 49(5):4677
22. Mirjalili, S. and A. Lewis (2013). "S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization." Swarm and Evolutionary Computation **9**: 1-14.
23. Osman, I. H. and J. P. Kelly (1996). "Meta-heusirtics: theory and applications." Kluwer Academic Publishers.
24. Osman, I. H. (1995). "An introduction to Meta-heuristics, in: Operational Research Tutorial Papers Series", Annual Conference OR37-Canterbury 1995, Eds. C. Wildson and M. Lawrence (Operational Research Society Press, 1995).
25. Pirlot, M. (1992). "General local search heuristics in combinatorial optimization: a tutorial", Belgian Journal of Operations, Statistics, and Computer Science, 32, 7-67.
26. Rizk-Allah, R. M. and A. E. Hassanien (2017). "New binary bat algorithm for solving 0–1 knapsack problem." Complex & Intelligent Systems **4**(1): 31-53.
27. Siddique, N. and H. Adeli (2015). "Nature Inspired Computing: An Overview and Some Future Directions." Cognit Comput **7**(6): 706-714.
28. Wang, G.-G., S. Deb and Z. Cui (2015). "Monarch butterfly optimization." Neural Computing and Applications.
29. Wang, G.-G., S. Deb, X. Zhao and Z. Cui (2016). "A new monarch butterfly optimization with an improved crossover operator." Operational Research **18**(3): 731-755.
30. Yang, X.-S. (2008). "Nature-Inspired Metaheuristic Algorithms." Luniver Press , UK.

31. Yang, X.-S. (2012). "Flower pollination algorithm for global optimization " International conference on unconventional computting and natural computation. Springer. Berlin.
32. Yang, X.-S. (2014a). "Cuckoo Search and Firefly Algorithm_ Theory and Applications " Springer International Publishing Switzerland.
33. Yang, X.-S. (2014b). "Nature-Inspired Optimization Algorithms." Elsevier Inc.
34. Yang, X.-S. (2015). "Recent Advances in Swarm Intelligence and Evolutionary Computation." Springer International Publishing Switzerland.
35. Yang, X.-s., M. Karamanoglu and H. Xingshi (2014). "Flower pollination algorithm: A noval approach for multiobjective optimization." Engineering Optimization.
36. Zhou, Y., X. Chen and G. Zhou (2016). "An improved monkey algorithm for a 0-1 knapsack problem." Applied Soft Computing **38**: 817-830.
37. Zouache, D., F. Nouioua and A. Moussaoui (2015). "Quantum-inspired firefly algorithm with particle swarm optimization for discrete optimization problems." Soft Computing **20**(7): 2781-2799.