Grey Wolf Optimization Applied to the 0/1 Knapsack Problem

Eman Yassien
Software Engineering Department
The World Islamic Science and Education University, Amman, Jordan

Raja Masadeh
Software Engineering Department
The World Islamic Science and Education University, Amman, Jordan

Abdullah Alzaqebah
Computer Science Department,
University of Jordan, Amman, Jordan

Ameen Shaheen
Computer Science Department
University of Jordan, Amman, Jordan

ABSTRACT
The knapsack problem (01KP) in networks is investigated in this paper. A novel algorithm is proposed in order to find the best solution that maximizes the total carried value without exceeding a known capacity using Grey Wolf Optimization (GWO) and K-means clustering algorithms. GWO is a recently established meta-heuristics for optimization, inspired by grey wolf's species. K-means clustering algorithm is used to group each 5-12 agents with each other at one cluster according to GWO constraint. The evaluated performance is satisfying. The simulation results show great compatibility between experimental and theoretical results.

General Terms
Algorithms.

Keywords
Grey Wolf Optimization (GWO), Knapsack problem, Meta-heuristic, Optimization.

1. INTRODUCTION
The 0–1 knapsack problem (01KP) is a famous NP-hard problem in combinatorial optimization introduced by [1] in the fifties. The problem then grasped lots of attention by many researchers because the knapsack has several practical applications [2][3] such as cutting stock problems, selection of projects and resource distribution. The knapsack appears as a sub-problem in many complex mathematical models of real world problems.

Many solutions were earlier provided for 01KP problem, such as the global harmony search introduced by [4], the ant colony optimization approach [5], the simplified binary version of the artificial fish swarm algorithm [6], and many others.

The 01KP problem arises when there's a need to select a subset of items from a set of items with specific values (v) and weights (w) that maximize the total profit without exceeding a known knapsack capacity (W).

The problem may be mathematically modeled as follows:

$$Maximize \sum_{i=1}^{n} x_i v_i$$

$$Subject\ to \sum_{i=1}^{n} x_i w_i \leq W, x_i \in \{0,1\}, \forall_i \in \{1,2,\dots\dots,n\}$$

where xi takes values either 1 or 0 for either including the value of the ith item or excluding it.

This paper proposes to solve the 01KP problem using the Grey Wolf Optimizer (GWO) and clustering algorithms. Grey wolf optimizer (GWO) is a new optimizer being used to solve some problems and produced good results; thus, researchers in this paper search to find the applicability of GWO on solving 01KP problem. The proposed algorithm provides global solutions rather than local ones.

The Remainder of this paper is organized as follows: Section 2 recites main researches in this regard, section 3 presents the new algorithm, section 4 provides the experimental results, and finally conclusion and future work is presented.

2. RELATED WORK
2.1 Knapsack 01 (01KP) Solutions
Several solutions were presented to solve 01KP problem; some exact algorithms search to find optimal solutions considering reasonable solution times, while others provide approximate solutions for complex algorithms.

Early researches focused mainly on exact algorithms; the general form of 01KP was solved with algorithms such as dynamic programming [7], branch and bound [8], or hybridizations of both [9].

Several updates of these algorithms also were provided to optimize the mentioned algorithms for extended versions of 01KP such as the unbounded KP, multiple KP [10], quadratic 01KP and discounted 01KP [11][12].

Anyway, exact algorithms cannot be reached for most NP-hard problems; it wouldn't be always feasible to find the optimal solution, so it would be more appropriate to find approximate solutions.

Approximate solutions can be reached through providing heuristic techniques, which are emerging quickly to solve NP-hard problems in feasible time.

Rather, meta-heuristics are becoming even more popular, because they are considered simple as they mimic simple ideas derived from nature; whereas simplicity eases application, implementation and enhancement of the algorithms. Also, meta-heuristics algorithms can be applied to many different problems.

Many meta-heuristic techniques emerged. Generally, it is classified into three main categories: evolutionary, physics-based and Swarm Intelligence algorithms. Some researchers used evolutionary meta-heuristics techniques inspired by the concepts of evolution in nature, such as the Genetic Algorithm [13][25] and Biogeography-Based Optimizer [14]. Others used physics-based techniques that mimic physical rules, such as Artificial Chemical Reaction Optimization Algorithm [15] and Gravitational Local Search [16].

And finally, some used Swarm Intelligence algorithms which mimic the social intelligent behavior of swarms and groups of creatures in nature, such as Ant colony algorithm [17], Marriage in Honey Bees Optimization Algorithm in [18] and the Bat-inspired Algorithm [19].

Recently, [20] introduced the grey wolf optimizer (GWO) that mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. GWO was used to solve many problems[26]

As the meta-heuristics started to provide good results, several trials were accomplished to solve 01KP problem using meta-heuristic.

[21] used the natural way of ants to solve problems for 01KP problems; While [22] suggested a binary particle swarm. [23] explores a monogamous pair genetic algorithm for solving 01KP. It was found to be computationally efficient, especially when dealing with high-dimensionality 01KP. Also, [24] proposed Chemical reaction optimization adopted from natural chemical reactions which showed "superior performance" in solving the 01KP compared to early mentioned algorithms.

2.2 Grey wolf optimizer (GWO):

The GWO algorithm was derived from grey wolves' social life. There are four different types of grey wolves (alpha, beta, delta, and omega). Alpha, beta and delta are the dominant wolves that employed to lead the pack (Omegas) for finding and hunting preys. [20]
The Alpha wolf is to be followed by other wolves. Beta wolves work as advisors to alpha wolves. Delta wolves are also subordinates for Beta wolves, while omega wolves are the followers and the last allowed to eat.[20]
Hunting (The optimization) is guided by set of decisions taken by the dominant wolves (α, β and δ) through a group of processes to track, chase, surround and finally attack the prey.

From algorithm's perspective: Alpha (α), beta (β) and delta (δ) represent first, second and third best solutions respectively and omega represents the rest candidate solutions.

The optimization algorithm (the hunting) is represented mathematically in 3 main stages: Encircling, hunting and attacking as follows:

2.2.1 Encircling prey

In nature grey wolves encircle prey during the hunt through several steps can be modeled mathematically as followings:

$$\vec{D} = \left|\vec{C}.\vec{X}_p(t) - \vec{X}(t)\right| \quad (1)$$

$$\vec{X}(t+1) = \vec{X}_P(t) - \vec{A}.\vec{D} \quad (2)$$

Where $\vec{D}$ illustrates the distance between the position of the prey ($\vec{X}_P$) and the position of the wolf (X) and t represents the current iteration number. A and C are coefficient vectors.

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \quad (3)$$

$$\vec{C} = 2\vec{r}_2 \quad (4)$$

Where r1 and r2 are random vectors in [0,1] and a is linearly decreased from 2 to 0 over the iteration.

2.2.2 Hunting phase:

Though in natural wild life, hunting is usually guided by the alpha with the cooperation of beta and delta as needed. But in digital world the same idea is not fully applicable because we have no idea about the location of the optimum solution (prey). Thus, mathematical simulation can be provided by considering the alpha (best candidate solution) beta and delta have better knowledge about the potential location of prey (optimal search agent). Therefore, we save the first three best solutions obtained so far and update the positions of other search agents according to the position of the -best search agent using the following equations (5, 6 and 7).

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}.\vec{X}_\alpha - \vec{X}\right|$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2}.\vec{X}_\beta - \vec{X}\right|$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3}.\vec{X}_\delta - \vec{X}\right| \quad (5)$$

$$\overrightarrow{X_1} = \vec{X}_\alpha - A_\alpha.(\overrightarrow{D_\alpha}), \overrightarrow{X_2} = \vec{X}_\beta - A_\beta.(\overrightarrow{D_\beta}),$$

$$\overrightarrow{X_3} = \vec{X}_\delta - A_\delta.(\overrightarrow{D_\delta}) \quad (6)$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1}+\overrightarrow{X_2}+\overrightarrow{X_3}}{3} \quad (7)$$

2.2.3 Attacking phase (Exploitation)

In nature, attacking the prey means the pack stops moving, so a certain constraint must be considered to keep the algorithm works globally. To model attacking mathematically it can be done by decreasing the value of (a) from 2 to 0 linearly. The random value A located between [-a, a] is also decreased. When A reaches to less than 1, suggested solutions approaches towards the optimal local solution, which leads to local stagnation.

To avoid the local stagnation, random value (A) is constrained to be greater than 1 or less than -1 to stay far from local stagnation. Another vector is used in this stage to emphasize a more random behavior for the algorithm which is C belongs to [0,2]. This component represents random weights of the prey in defining the distance. When C>1 emphasize its influence. However, when C<1 reduces its effect.

3. THE ALGORITHM 01KP-GWO

The researchers apply GWO on 01KP problem seeking for better solutions; pseudo code of the proposed algorithm is presented in figure 1.

Each cluster (k), finds first three best solutions (α, β and δ) randomly at the beginning. Other search agents update their positions around the chosen search agent (prey) according to α, β and δ positions by using equations (1) to (7). At the end, each cluster has local optimal consists of the three best solution. Local solutions are compared with each other's to find different global optimal.

Call initializations function
Call Clustering functuion
For each cluster{
For each search agent{
Calculate the fitness to find *(Maximum ratio)*
((Xα=the first best fitness
Xβ=the second best fitness
Xδ=the third best fitness))

while *(t < Max number of iterations)*
{
for *each search agent*
Update the position of the current search agent by equation (3.7)
end for

```
Update a, A, and C
Calculate the fitness of all search agents
Update Xα, Xβ, and Xδ
t=t+1   }
end while
}
Call OptimalSolution function
```

Figure 1: Pseudo-Code for " 01KP-GWO "

3.1 Initialization Function

Initialization function (as presented in Figure 2) initializes random vectors, the nodes' population, total allowed weight for the 01KP and the values and weights for each search agent. From these search agents, one of them is selected randomly to be the prey while others are considered as sources.

```
Initialize the nodes ' population Xi (i = 1, 2, ..., n)
Initialize a, A, and C  // random vectors
Initialize the Total allowed W eight (W)
for each search agent
        Initialize value (v) and w(weight) randomly
end for
```

Figure 2: initializations function

3.2 Clustering & Positioning Functions

As the GWO algorithm [20] considers maximum size of the included search agents to be 12 search agents (wolves). Figure 3 presents K-means clustering algorithm which is used to group each pack at least 5 wolves and at most 12 wolves using the following equations.

Where Xi is the wolf (i) and Cenj is the centroid in the cluster j.

Number of clusters = Ceiling ((Number of wolves) / 12)

Number of centroids=Number of clusters

Distance = min || Xi – Cenj ||

At the beginning, number of clusters is calculated by equation (8). Then, the number of centroids is calculated by equation (9). Thus, the centroids are chosen randomly. Each wolf is assigned to the nearest centroid by using equation (10) as shown in figure (4).

```
Calculate number of clusters using equation (8)
Calculate number of centroids using equation (9)
for each search agent
        Call Positioning function.
end for
```

Figure 3: Clustering function

```
// Positioning Function
Calculate the distance between each agent and
all centroids by the equation (5)
 Assign agent (i) to its closet centroid (Largest
        ratio).
```

Figure 4: Positioning Function

3.3 Optimal Solution Function

After all clusters are constructed, three best solutions are determined: α, β and δ. The α solution represents the nearest search agent to the prey in the cluster, the next closest search agent to the prey in the cluster is represented as β, and δ is the third nearest search agent to the prey in the same cluster. The maximum of all local optimal values is considered the best global solution as shown in figure 5.

```
for each Cluster
    Set Global_Optimal = Max(All Local Optimals)
end for
```

Figure 5: Optimal Solution

Below is the complexity calculation for the 01KP-GWO code, where n is the number of nodes

- Initialization function is O (n)
- Clustering function is O (n)
- Calculate Fitness function is O (n^2); it will be accomplished using 2 loops for each cluster and each search agent.
  thus,
- Total time complexity of ” 01KP-GWO " Algorithm is O (n^2)

Table 1: Run time of “01KP-GWO” for various datasets.

Network Size	Average run times/ Seconds	Network Size	Average run times/ Seconds
50	0.18	550	4.05
100	0.19	600	4.9
150	0.34	650	5.53
200	0.57	700	6.6
250	0.92	750	7.41
300	1.19	800	8.52
350	1.64	850	9.56
400	2.12	900	10.86
450	2.66	950	12.3
500	3.33	1000	13.47

4. EXPERIMENTAL RESULTS

The performance of the proposed algorithm was evaluated by simulation program tool with GUI to facilitate the use of the tool, ORACLE software was used to simulate the proposed algorithm the main load on the DBMS and it is ORACLE 11g release 2 and it used to store data and accomplishing algorithm's work by procedures and functions, ORACLE Developer 6i was used to build the GUI and connect it with the database schema.

Results of the paper show that data set (nodes, weights, values ...etc) was generated automatically from the simulator with respect to user parameters and ranges. Different sizes of networks were taken in order to evaluate the proposed algorithm to solve the 01KP problem with increasing size of

the network. In this study, each scenario was repeated 10 times to generate more accurate results. The platform specifications that used to get these experiments are: Intel (R) core (TM) 2 duo CPU with 3.00 GHz, 4 GB RAM and Windows XP Professional service pack 3. Table 1 presents the average run-time for various datasets that calculated in seconds.

It is clear from Figure 6 that the time complexity is quadratic polynomial which means it increases with increasing the number of vertices in the graph. In addition to that, it is fairly a good enough performance. Depending on Figure 6 and Figure 7, it is obvious that the experimental and theoretical results are very approximate.

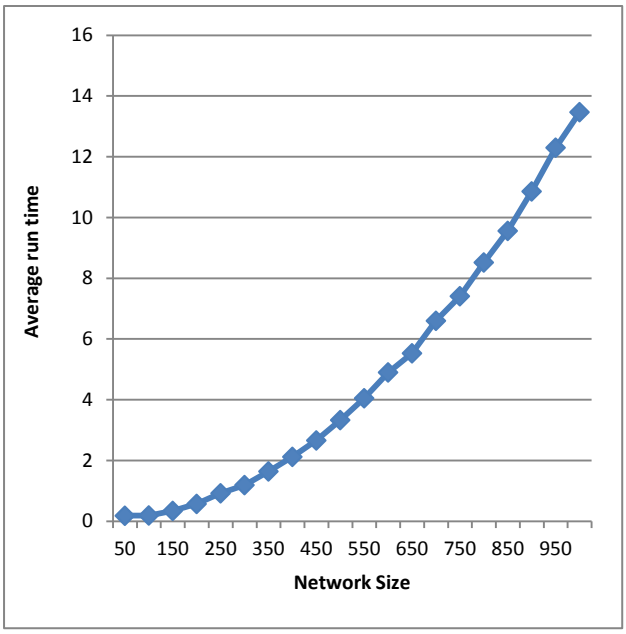

Figure 6: Average running time for experimental results for "01KP-GWO"

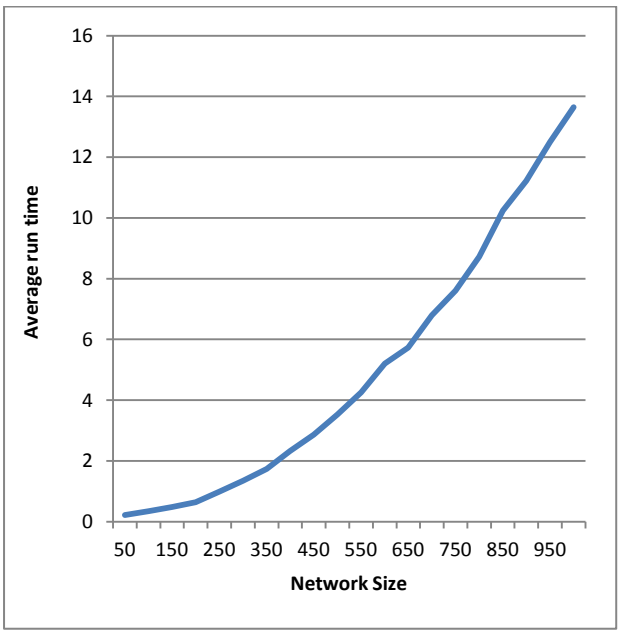

Figure 7: Theoretical runtime

5. CONCLUSION AND FUTURE WORK

This paper proposes a novel solution (01KP-GWO) to 01KP problem using Grey Wolf Optimizer (GWO) to get the optimal solution. The time complexity of 01KP-GWO algorithm is proved theoretically to be O (n2). Experimentally, time complexity is obtained to be quadratic polynomial which means it increases as number of nodes increase in the graph. Thus, there's a great convergent between theoretical and experimental results.

Future work may include investigating using different algorithms to solve 01KP problem, unifying same devices and programming languages to compare different solutions and obtain best results.

6. REFERENCES

[1] DANTZIG, George B. Discrete-variable extremum problems. Operations research, 1957, 5.2: 266-288.

[2] PISINGER, David. The quadratic knapsack problem—a survey. Discrete applied mathematics, 2007, 155.5: 623-648.

[3] Bhattacharjee, K. K., & Sarmah, S. P. (2015, March). A binary cuckoo search algorithm for knapsack problems. In Industrial Engineering and Operations Management (IEOM), 2015 International Conference on (pp. 1-5). IEEE

[4] ZOU, Dexuan, et al. Solving 0–1 knapsack problem by a novel global harmony search algorithm. Applied Soft Computing, 2011, 11.2: 1556-1564.

[5] CHANGDAR, Chiranjit; MAHAPATRA, G. S.; PAL, Rajat Kumar. An Ant colony optimization approach for binary knapsack problem under fuzziness. Applied Mathematics and Computation, 2013, 223: 243-253.

[6] AZAD, Md Abul Kalam; ROCHA, Ana Maria AC; FERNANDES, Edite MGP. A simplified binary artificial fish swarm algorithm for 0–1 quadratic knapsack problems. Journal of Computational and Applied Mathematics, 2014, 259: 897-904.

[7] TOTH, Paolo. Dynamic programming algorithms for the zero-one knapsack problem. Computing, 1980, 25.1: 29-45.

[8] KOLESAR, Peter J. A branch and bound algorithm for the knapsack problem. Management science, 1967, 13.9: 723-735.

[9] POIRRIEZ, Vincent; YANEV, Nicola; ANDONOV, Rumen. A hybrid algorithm for the unbounded knapsack problem. Discrete Optimization, 2009, 6.1: 110-124.

[10] CHEN, Yuning; HAO, Jin-Kao. A "reduce and solve" approach for the multiple-choice multidimensional knapsack problem. European Journal of Operational Research, 2014, 239.2: 313-322.

[11] RONG, Aiying; FIGUEIRA, José Rui; KLAMROTH, Kathrin. Dynamic programming based algorithms for the discounted {0–1} knapsack problem. Applied Mathematics and Computation, 2012, 218.12: 6921-6933.

[12] HE, Yi-Chao, et al. Exact and approximate algorithms for discounted {0-1} knapsack problem. Information Sciences, 2016, 369: 634-647.

[13] HOLLAND, John H. Genetic algorithms. Scientific american, 1992, 267.1: 66-72.

[14] SIMON, Dan. Biogeography-based optimization. IEEE transactions on evolutionary computation, 2008, 12.6: 702-713.

[15] ALATAS, Bilal. ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Systems with Applications, 2011, 38.10: 13170-13180.

[16] WEBSTER, Barry; BERNHARD, Philip J. A local search optimization algorithm based on natural principles of gravitation. 2003.

[17] DORIGO, Marco, et al. (ed.). Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings. Springer, 2008.

[18] ABBASS, Hussein A. MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach. In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. IEEE, 2001. p. 207-214.

[19] YANG, Xin-She. A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer Berlin Heidelberg, 2010. p. 65-74.

[20] MIRJALILI, Seyedali; MIRJALILI, Seyed Mohammad; LEWIS, Andrew. Grey wolf optimizer. Advances in Engineering Software, 2014, 69: 46-61.

[21] YANG, Jinhui, et al. An ant colony optimization method for generalized TSP problem. Progress in Natural Science, 2008, 18.11: 1417-1422.

[22] LI, Zhuangkuo; LI, Ning. A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem. In: Control and Decision Conference, 2009. CCDC'09. Chinese. IEEE, 2009. p. 3042-3047.

[23] LIM, Ting Yee; AL-BETAR, Mohammed Azmi; KHADER, Ahamad Tajudin. Taming the 0/1 knapsack problem with monogamous pairs genetic algorithm. Expert Systems with Applications, 2016, 54: 241-250.

[24] TRUONG, Tung Khac; LI, Kenli; XU, Yuming. Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem. Applied Soft Computing, 2013, 13.4: 1774-1780.

[25] SHAHEEN, Ameen; SLEIT, Azzam. Comparing between different approaches to solve the 0/1 Knapsack problem. International Journal of Computer Science and Network Security (IJCSNS), 2016, 16.7: 1.

[26] Masadeh R., Sharieh A. , Sleitn, A.. "Grey wolf optimization applied to the maximum flow problem", International Journal of ADVANCED AND APPLIED SCIENCES 4(7):95-100 · June 2017.