International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 3 Issue: 8
ISSN: 2454-4248
203 – 207

New Hybrid Approach to Solve the 0/1, Bounded Knapsack Problem

Saher Manaseer/ (Corresponding Author)
Computer Science Dept. / King Abdullah II, Information Technology Faculty
The University of Jordan, Amman, Jordan
saher@ju.edu.jo

Huthaifa Almogdady *(Author)*
Computer Science Dept. / King Abdullah II, Information Technology Faculty
The University of Jordan, Amman, Jordan
Huthaifa.cis@gmail.com

***Abstract*—**Knapsack problem is one of the combinatorial optimization problems, consisting of selecting a subset of given items in such a way that the total profit of the selected items doesn't exceed the maximum of knapsack capacity. As well known that most of the famous algorithms published nowadays work on either dynamic or greedy system as greedy is a fast technique and dynamic systems can produce an optimal solution for the problem, however, as greedy function can solve the problem with high speed it cannot give an optimal solution and, sometimes, answers are relatively far from optimality. Therefore, the dynamic approach may solve the problem with optimal solution but it consumes much time and space. In order to merge advantages of these two approaches, in this proposed study we introduced an idea that generates a hybrid algorithm that is based on the two methods, the dynamic and the greedy, to get the best performance for finding the best proximity to the optimal solutions.

***Keywords*-** *Knapsack; Greedy; Dynamic, Hybrid, Utilization*

*****

I. INTRODUCTION

Combinatorial streamlining issue is a Knapsack issue that accomplish. Boosting the advantage of items in a rucksack without surpassing the limit [1]. An outstanding advancement issue is considered as NP-hard family issues [2].The backpack issue or rucksack is an issue in combinatorial improvement: Given an arrangement of objects, each object with a mass and esteem. The decision must be made about the quantity of objects to incorporate into a collection so that the aggregated weight is not equivalent to a given point of confinement and the aggregated esteem is as high as possible. The mechanism gets its name from the issue confronted by a person who is compelled by a settled size rucksack and must fill it with the most profitable items.

Backpack issue can be divided into two basic classes. Limited rucksack issue, where the quantity of items is constrained. In addition, unbounded rucksack issue, where the quantity of items is not restricted. Moreover, these two types have two adaptations of the issue. In 0/1 backpack issue. The selection of items is binary. This means that the object is either selected as a unit or not selected at all As for fragmentary rucksack issue, items are distinct; objects are not necessary selected as unit and can be partially selected. The proposed algorithm addresses limited, 0/1 backpack issue.

Given two n-tuples of positive numbers Values as {v1, v2, … ,vn} and Weights as {w1,w2,… ,wn} And W > 0, we wish to decide subset that have the most important things with the limitation of W.

$$\text{Maximize} \quad \sum_{i=1}^{n} v_i x_i \quad \text{subject to}$$

$$\sum_{i=1}^{n} w_i x_i \leqslant W, \qquad x_i \in \{0,1\}$$

Boost subject to The backpack issue has been contemplated for over a century, with early works dating back as far as 1897.It is not known how the name "rucksack issue" started. However, the issue was alluded to thusly in the early works of mathematician Tobias Dantzig (1884–1956), proposing that the name could have existed in old stories before a numerical issue had been completely characterized [3].

This paper is sorted out as following: Section II for the related work while Section III presents the approach more in subtle elements for the proposed calculation in this study. Segment IV talks about the reenactment setup utilized for assessment purposes. Simulation results are introduced in Section V. Lastly, Section VI finishes up the paper with the conclusion and the future works.

II. RELATED STUDIES

The following studies consider as most recent studies that cover the all phases of any recognition system, a lot of approaches have been made to solve such important problem as knapsack some of them are interested in the optimality. Other approaches are in satisfy-ability. Next, this section gives examples for these studies.

In [4], a new exact approach for the 0–1 Collapsing Knapsack Problem was introduced; the authors proposed a novel ILP formulation and a problem reduction procedure together with an exact approach.

In [5], an Approximation scheme for the parametric knapsack problem has been proposed. The authors, provided the first (parametric) polynomial time approximation scheme (PTAS) for the knapsack. They also made use of the connection between the parametric problem and bi-criteria problem in order to demonstrate that the parametric 0-1-knapsack problem concedes a parametric FPTAS when the parameter is restricted to the positive real line and slopes and intercepts of the affine-linear profit functions of the items are non-negative.

 ISSN: 2454-4248
203 – 207

In [6] they presented a cost-optimal parallel algorithm for the 0–1 knapsack problem and its performance on multicore CPU and GP implementations, they suggested the cost-optimal algorithm (COPA) on an EREW PRAM model with shared memory to solve this problem. They also implemented COPA on to cases one is multicore CPU bases architectures using open MP and GPU based configurations using CUDA. Their approach reduced the speedups up to 10.26 on multi core CPU and 17.53 on GPU implementations.

In [7], they proposed an improved genetic algorithm based approach to solve constrained knapsack problem in fuzzy environment, were they brought forth an improved GA to solve the problem, where by introducing "refining" and "repairing" operations the genetic algorithm was improved. The proposed GA improved the profit by an average of 19.06.

In [8], a Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem was presented; he introduced the SACRO algorithm, which dynamically and automatically changes the alternative pseudo-utility ratio as the PSO algorithm runs. It was proved as a result of the simulations and evaluations that the SACRO is more competitive and robust than the simple or improved CROs.

In [9], proposed Solving 0–1 Knapsack Problem using Cohort Intelligence

Algorithm, the impact of different parameters on, the advantages and limitations of the CI methodology are also discussed and tested by the solving NP-hard combinatorial problem such as the Knapsack problem (KP).

In [10], introduced The multidimensional 0–1 knapsack problem: An overview, his paper surveyed the main results published in the literature, its main focus is on the theoretical properties as well as an approximate or exact solutions of this special 0-1 program.

In [11] A hybrid Approach for the 0-1 multidimensional Knapsack problem was introduced. Their proposed approach combines the linear programming and tabu search. They showed that their algorithm improves significantly on the best known results of a set of more than 150 benchmark instances.

In [12] they presented a hybrid Genetic Algorithm (GA) for solving the Multi-constrained 0–1 Knapsack Problem (MKP). Based on the solution of the LP-relaxed MKP, the proposed GA uses sophisticated repair and local improvement operators which are applied to each newly generated solution. Care has been taken to define these new operators in a way avoiding problems with the loss of population diversity. The new algorithm has been empirically compared to other previous approaches by using a standard set of "large-sized" test data. Results show that most of the time the new GA converges much faster to better solutions.

And in [13] they proposed analyze several algorithm design paradigms applied to a single problem the 0/1 Knapsack Problem. The main goal of proposed paper is to present a comparative study of the brute force, dynamic programming, memory functions, branch and bound, greedy, and genetic algorithms. The paper discusses the complexity of each algorithm in terms of time and memory requirements, and in terms of required programming efforts. Their experimental results show that the most promising approaches are dynamic programming and genetic algorithms. The paper examines in more details the specifics and the limitations of these two paradigms.

Also in [14] they proposed a method to work in problems in combinatorial optimization. In this presented study they consider their multi-objective extension (MOKP and MOMKP), for which the aim is to obtain or to approximate the set of efficient solutions. There methodology start as, they classify and describe briefly the existing works that are essentially based on the use of meta-heuristics. In a second step, they propose the adaptation of the two phases Pareto local search (2PPLS) to the resolution of the MOMKP. With this aim, they use a very large scale neighborhood (VLSN) in the second phase of the method that is the Pareto local search. They compare our results to state of the art results and they show that we obtain results never reached before by heuristics, for the objective instances.

III. METHODOLOGY

The methodology of our proposed work contains of three phases; Phase one is the initiative phase that creates working area for preparing comparison values, and the second phase works on the greedy approach, then the third phase deals with dynamic. The following graph shows the methodology and the working approach algorithm.

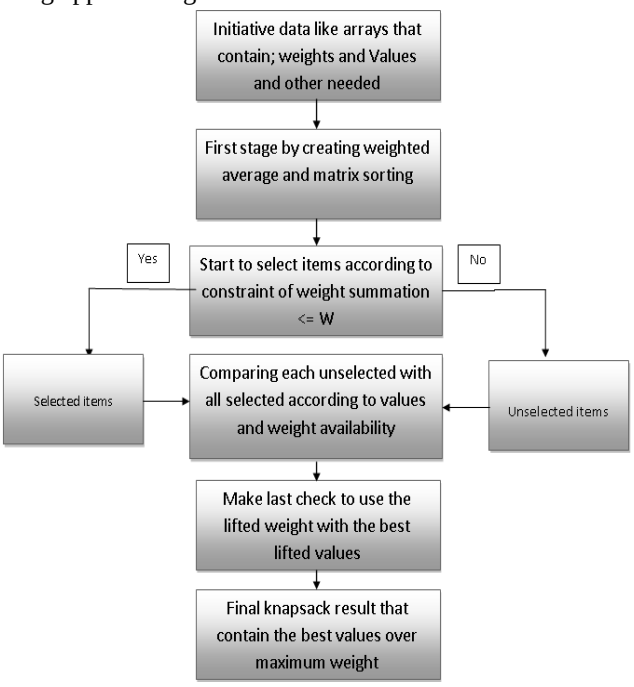

Figure 1: our methodology diagram

 ISSN: 2454-4248
203 – 207

We introduced a new approach mixing both dynamic and greedy approach; our methodology will split into 4 main steps as following:

1. Using the weighted average that used on greedy to have a descending table for these values; as will see on next two steps this table will be used for both selecting and comparing.

Calculate weighted average.
WA[i] = v[i] / w[i] for i=1, 2… n items; v: value; w:wight
Sort the items by WAi in decreased order.

2. Start selecting from up to down till we reach a threshold - in our case the threshold should be the maximum allowed weight- then will move to next step.

```
Remain = W
Sum = 0
For i = 1 to n
If sum < W
B[i] = items[i]
Sum = sum + B(weights [i])
Remain = W - Sum
End If
End For
```

Part 1 matrix "C" contain remaining unselected items.
Part 2 matrix "Z "contain selected items.

3. Using a recurrence dynamic approach to deal with lifted item and that shall happened after applying some constraints to make replacements or adding as necessary

```
If C =! 0
For j = 1 to n, where n is the length of C array
If selcted (L) == 0 &&selcted (O) == 1 &&newsorted (1,L) + summation - newsorted (1,O) <= W &&newsorted (2,L) > newsorted (2,O) ;
For z = y to 1, where y is the length of B array
If C(wa [j]) > B(wa [Z]) and Remain + B (weight [z]) =< W
B[z] = C[j]
End if
End for
End if
End for
```

4. Sorting remaining items according to values

```
For I from 1 to remaining items
        If item.weight <= the remaining weight
        Items.selected = true
        End if
End for
```

Selecting the best from remaining items that can be put in the remaining weights.

Our methodology is based on reverse searching on already obtained results from already exist dynamic and greedy approaches to find a relation between that data and the table that we obtained from step number 1 then find the applying constraint to add them to the rest of item to make the last selection and replacement step.

IV. IMPLEMENTATION AND RESULTS

Our implementation has been carried on HP laptop with:
8 G Ram, Core I5 processer, Windows 8 operating system and Matlab as implementing and testing tool.
We made the simulation and implementation using MATLAB, the generated code is as follow and down here result of time and space complexity will be shown.

a. Time and Space Complexity

i. **Space Complexity**: we will use a*N+c for calculation While:

a = number of arrays
N= number of element
c= number of used variable

The Space complexity for our proposed algorithm = O(n)
So our approach is much better in space complexity comparing with normal dynamic knapsack
That need O(m*n) which could be huge amount of space.

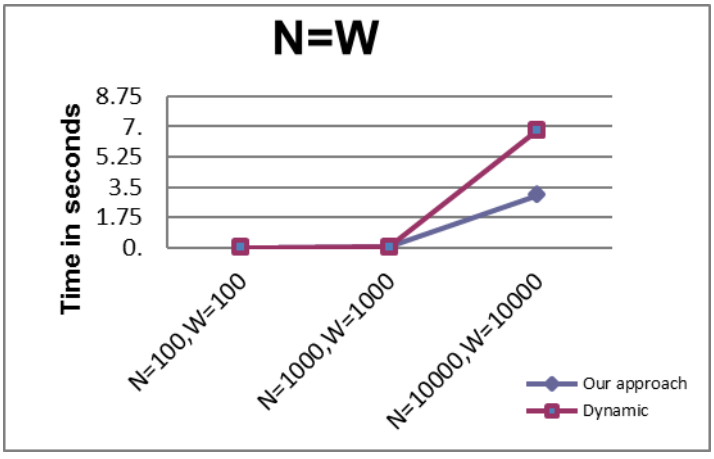

Figure 2: time comparison where Number of items is equal to the max weight

Case 2 as shown in figure 3 when N is bigger than W.

ii. **Time Complexity**:

 ISSN: 2454-4248
203 – 207

The main different between our approach and normal dynamic that when maximum weight is larger than items number we have beat time complexity in worst , best and average case

Best Case: O (n log n) and its equal the time of sort
Average Case: calculation in process
Worst Case: O (n2) and its equal to time of comparing selected results

Table 1 show both time and space complexity for the most used algorithm approach for solving knapsack problem and if the approaches provide optimal solutions or not.

Table 1: Comparison between approaches

	Time complexity	**Space complexity**	**Optimality in 0/1**
Dynamic	O(n*W)	O(n*W)	yes
Greedy	O(n*log n)	O(n)	no
Backtracking	O(2n)	O(2n)	yes
heuristic			no
Our hybrid	O(n^2)	O(n)	no

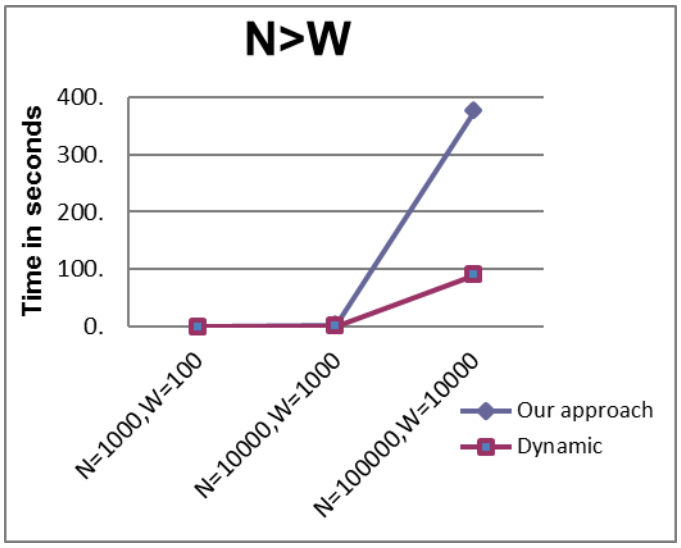

Figure 3: time comparison where Number of items is bigger than max weight

Case 3 as shown in figure 3 when W is bigger than N.

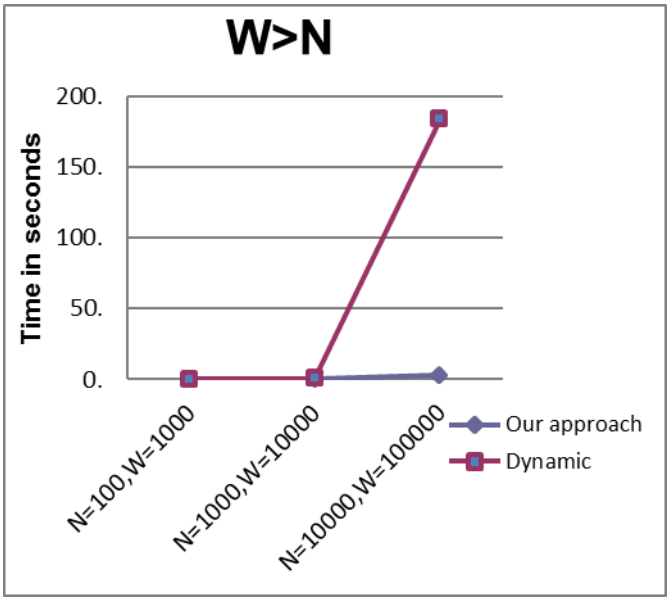

Figure 4: time comparison where max weight is bigger than Number of items

As we can notice that our approach is faster than dynamic in both cases 1 and much more faster in 3, in case 1 it take almost the half time, and in case 3 it toke less than 2.7% of dynamic time, but in case 2 our system was 4 time slower than dynamic , these differences in time caused by the dependency of our system in reducing the used space and time on the number of items more than max Wight, so we can notice that when N=W and in N>W we do better.

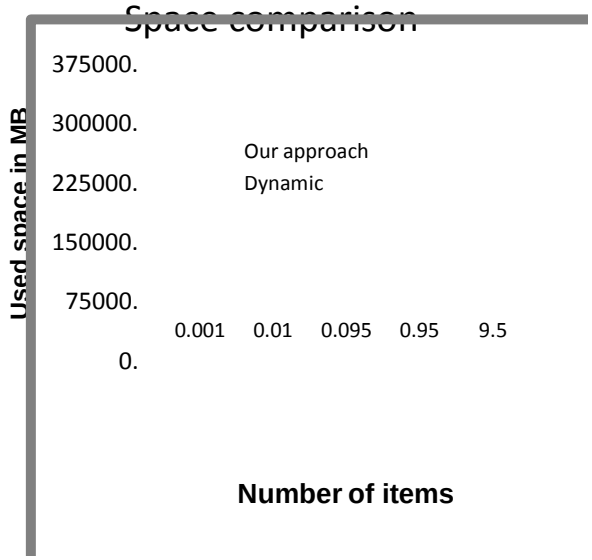

Figure 5: Space comparison between ours and Dynamic

As a results of our hybrid approach comparing to the dynamic one, these results shows on table 2 bellow, as it appear from results our system gives 99.7% of optimal solution as an average, where the rest of 0.3 % of none optimal solutions, our solution is near by almost 99.95% near optimality, which is very good results for a system that can

 ISSN: 2454-4248
203 – 207

save up to thousands of times less in space as shown in figure 5 and much faster in the most cases.

Table 2: Results comparison between approaches

Approach	N=W		N>W		W>N	
	OPT*	DFO**	OPT	DFO	OPT	DFO
Dynamic	100%	0	100%	0	100%	0
Ours	99.8%	0.02%	99.6%	0.11%	99.7%	0.01%

*OPT: Ratio of Optimal Results over 1000 iterations.

**DFO: Difference from Optimal which is a ratio between the difference between optimal solution and our results divided by the optimal solutions.

V. Conclusion And Future Work

Knapsack problem is one of the combinatorial optimization problems, consisting of selecting a subset of given items in such a way that the total profit of the selected items doesn't exceed the maximum of knapsack capacity, our approach is better than greedy and heuristic because it gives a better solution and better than heuristic in time and space wise.

Our approach is better than dynamic in space wise and whenever weight is larger than number of items our approaches have advantage in time complexity either, and as the best case in our approach is better than another approaches TIMES with O (n log n), and this time complexity came from the time of sorting so any better sorting time algorithm will enhance our method best case and make the approach much better.

References

[1] Mahajan, R., & Chopra, S. "Analysis of 0/1 Knapsack Problem Using Deterministic and Probabilistic Techniques". In Proceedings of the 2012 Second International Conference on Advanced Computing & Communication Technologies (2012, January). (pp. 150-155). IEEE Computer Society.

[2] Xiao-hua, X., Ai-bing, N., Ma, L., & An-bao, W. "Competitive decision algorithm for multidimensional knapsack problem". Proceedings of Management Science and Engineering, 2009.ICMSE 2009. International Conference, (2009, September) (pp. 161-167). IEEE.

[3] Information taken from http://en.wikipedia.org/wiki/Knapsack_problem Wikipedia, accessed date 27/5/2014.

[4] Della Croce, F., Salassa, F., & Scatamacchia, R. "A new exact approach for the 0–1 Collapsing Knapsack Problem". European Journal of Operational Research, (2017). 260(1), 56-69.

[5] Giudici, A., Halffmann, P., Ruzika, S., & Thielen, C. "Approximation schemes for the parametric knapsack problem". Information Processing Letters, (2017). 120, 11-15.

[6] Li, K., Liu, J., Wan, L., Yin, S., & Li, K.. "A cost-optimal parallel algorithm for the 0–1 knapsack problem and its performance on multicore CPU and GPU implementations". Parallel Computing, (2015), 43, 27-42.

[7] Changdar, C., Mahapatra, G. S., & Pal, R. K.. "An improved genetic algorithm based approach to solve constrained knapsack problem in fuzzy environment". Expert Systems with Applications, (2015), 42(4), 2276-2286.

[8] Chih, M. "Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem". Applied Soft Computing, (2015), 26, 378-389.

[9] Kulkarni, A. J., & Shabir, H. "Solving 0–1 knapsack problem using cohort intelligence algorithm". International Journal of Machine Learning and Cybernetics, (2016). 7(3), 427-441.

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

[11] Vasquez, M., & Hao, J. K.. "A hybrid approach for the 0-1 multidimensional knapsack problem". In IJCAI , (2001, August), (pp. 328-333).

[12] Raidl, G. R. "An improved genetic algorithm for the multiconstrained 0-1 knapsack problem". In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference, (1998, May), p. 207-211. IEEE.

[13] Hristakeva, M., & Shrestha, D. "Different Approaches to Solve the 0/1 Knapsack Problem." (2004), Retrieved November, 3, 2012.Online:[http://www.micsymposium.org/mics_2005/papers/paper102.pdf]

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