|  |  |  |
| --- | --- | --- |
| **EMAIL** | **ID** | **NAME** |
| **Omar\_20180368@fci.helwan.edu.eg** | **20180368** | **عمر خالد عبد الفتاح** |
| **Hussien\_20180208@fci.helwan.edu.eg** | **20180208** | **حسين وجدي محمود** |
| **Hamza\_20180213@fci.helwan.edu.eg** | **20180213** | **حمزة جمال عبد الفتاح** |
| **Hamza\_20180214@fci.helwan.edu.eg** | **20180214** | **حمزة محمود عبد العظيم** |
| **Ahmed\_20180052@fci.helwan.edu.eg** | **20180052** | **احمد عبد الستار عثمان** |

# Knapsack problem

* **Project idea in details:**

The project idea is to solve the Knapsack problem with an algorithm which is called Particle Swarm Optimization algorithm (PSO) **(which will be discussed in the 6th point in this documentation).** So, what is the Knapsack problem?

The Knapsack problem is considered an optimization problem: Given a set of objects with given weights and values for each one and a Knapsack (or a bag) with given capacity, and we tend to solve this problem by put the objects in this knapsack in which they can fit the enable capacity and to maximize the value of the summation of theses objects in the knapsack (In other words we must fill the knapsack with the most valuable objects).

There are two or three types of this problem and we will work on the following two types in this project which are:

* (0-1) knapsack 🡪 which means that there are no copies for each item and you can only either take the item in the knapsack or not.
* Unbounded knapsack 🡪 which means you can take many objects from the same item to fill the knapsack
* **Main functionalities:**

The main function of the knapsack problem is to determine the number of items which can be included in knapsack. The knapsack has weight so, the summation of all items weight should be less than or equal to the it. Through that problem should solve this issue. The issue is how to achieve the optimal solution through maximize the value (profit) and try to fill the knapsack with the maximum number of items less that its weight.

* **Similar applications in the market:**

Knapsack problem used to make decision for the variety fields. It used for less the wasteful of the cost. selection of [investments](https://en.wikipedia.org/wiki/Investment) and [portfolios](https://en.wikipedia.org/wiki/Portfolio_(finance)), selection of assets for [asset-backed securitization](https://en.wikipedia.org/wiki/Securitization). Generate keys for Merkle–Hellman and other cryptographic systems.

* **An initial literature review of Academic publications (papers):**

# A Branch and Bound Algorithm for the Knapsack Problem [Peter J. Kolesar](https://pubsonline.informs.org/action/doSearch?text1=Kolesar%2C+Peter+J&field1=Contrib) Columbia University in (1967).

# The Multiple-Choice Knapsack Problem [Prabhakant Sinha](https://pubsonline.informs.org/action/doSearch?text1=Sinha%2C+Prabhakant&field1=Contrib) [Andris A. Zoltners](https://pubsonline.informs.org/action/doSearch?text1=Zoltners%2C+Andris+A&field1=Contrib) in (Jun 1979)

## Algorithms for Knapsack Problems (1995) David Pisinger Universitetsparken 1, DK-2100 Copenhagen, Denmark.

1. Unbounded knapsack problem: Dynamic programming revisited [Vincent Poirriez](https://univ-valenciennes.academia.edu/VincentPoirriez?swp=tc-au-14017212) [Rumen Andonov](https://independent.academia.edu/RumenAndonov?swp=tc-au-14017212) Rumen Andonov S. Rajopadhye in (2000)
2. Multidimensional Knapsack Problems Hans KellererUlrich Pferschy, David Pisinger University of Graz in (2004)

* **The Dataset employed:**

We don’t use specific data set in our project. We just use two arrays one for the profits of the objects and the second for the corresponding weights to each object, and the knapsack has its own capacity.

For example:

val = [35, 85, 135, 10, 25, 2, 94]  
kg = [2, 3, 9, 0.5, 2, 0.1, 4]  
maxKg = 25

------------------------------------

val = [92, 57, 49, 68, 60, 43, 67, 84, 87, 72]  
kg = [23, 31, 29, 44, 53, 38, 63, 85, 89, 82]  
maxKg = 165

------------------------------------

val = [12, 7, 11, 8, 9]  
kg = [ 24,13, 23,15,16]  
maxKg = 26

* Examples for the input and the outputs:

Text

Description automatically generatedText

Description automatically generated

This output means that each 1 in the array corresponds to the object we want to take in the knapsack, and the profit is 309

Another example:

A picture containing text, device, gauge, meter

Description automatically generatedText

Description automatically generated

This output means that each number > 0 in the array corresponds to the object we want to take this number of instances from this object, and the profit in this case is 700, and the sum weights of taken objects is 25 which is <= the capacity of the knapsack

* **Source code URL:**

<https://drive.google.com/drive/folders/1jINRYuloqGGZtTcd90qdep4U7lQqkcrf?usp=sharing>

* **Details of the algorithm:**

Knapsack problem is solved with many algorithms such dynamic programming, greedy algorithm and backtracking.

But we use swarm intelligence to solve the KP problem. Type of swarm intelligence we use is PSO (Particle Swarm Optimization) and now we will discuss the PSO algorithm.

Steps of PSO to reach for the optimum:

**Step1:** Initialize a swarm of particles (swarm size is m), set random position and random velocity of each particle on the allowable range randomly, the position of each particle determines randomly

**Step2:** Evaluate the fitness of each particle

**Step3:** To each particle, compare the best global position undergone best p with its fitness value, if it’s better than best p, it’s the best current position best p.

**Step4:** To each particle, compare the best position undergone best g with its fitness value, if it’s better than best g , it’s the best swarm position ,the index sign of best g will be set anew.

**Step5:** Change the velocity and the position of each particle

**Step6:** Check termination condition (the condition is enough good fitness value or reaching the maximum iterations, or that optimal solution changes no longer), If it meets the above condition, stop iteration; otherwise return to Step2.

* **Development platform:**

**PYTHON - Spider IDE**

* **References:**

# [Applied Mathematical Modelling](https://www.sciencedirect.com/journal/applied-mathematical-modelling)

* Information Computing and Applications.
* Hembecker, F., Lopes, H.S.: Particle Swarm Optimization for the Multidimensional Knapsack Problem. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.)