# Knapsack problem - solved with SOO and MOO with GA

## Problem definition

Given a set of $n$ items numbered from 1 up to $n$, each with a weight $w_i$ and a value $v_i$, along with a maximum weight capacity $W$,

## Versions:

### 1. 0-1 knapsack problem

Each item can be included (1) or not (0) in the knapsack. The individuals are represented as bit strings.

Given a set of $n$ items numbered from 1 up to $n$, each with a weight $w_i$ and a value $v_i$, along with a maximum weight capacity $W$:

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

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\}.$$

### 2. Bounded knapsack problem (BKP)

Each element can be included in the knapsack a certain number of times. The individuals are represented as natural numbers.

The **bounded knapsack problem** (**BKP**) removes the restriction that there is only one of each item, but restricts the number $x_i$ of copies of each kind of item to a maximum non-negative integer value $c$:

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

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } 0 \leq x_i \leq c$$

### 3. Unbounded knapsack problem (UKP)

Each element can be included in the knapsack an unlimited number of times. The individuals are represented as natural numbers.

The **unbounded knapsack problem** (**UKP**) places no upper bound on the number of copies of each kind of item and can be formulated as above except for that the only restriction on $x_i$ is that it is a non-negative integer.

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

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \geq 0$$

## 0-1 (binary) knapsack problem solved with GA SOO

Objective: maximize total value

In the script *define_values.m,* fill in the values for the maximum weight and the matrix that contains the weight and value for each item.
The function *fitness_kp_binary.m* computes the total value of the items in the knapsack.
Analyze and run the script *run_kp_binary.m.*
What type of representation is used for the individuals of the population? How many generations does the optimization take?
Comment on the results.
For the final solution, check the weight and the value of the knapsack.

## 0-1 (binary) knapsack problem solved with GA MOO

Objectives: maximize total value, minimize number of items

In the script *define_values.m*, fill in the values for the maximum weight and the matrix that contains the weight and value for each item.
The function *fitness_kp_binary_multi.m* computes the total value of the items in the knapsack, as well as the number of items.
Analyze and run the script *run_kp_binary_multi.m.*
What type of representation is used for the individuals of the population? How many generations does the optimization take?
Comment on the results.
Visualize and comment the Pareto front.

# Bounded knpasack problem solved with GA SOO with linear inequality constraints and boundary constraints

Objective: maximize total value

In the script *define_values.m*, fill in the values for the maximum weight and the matrix that contains the weight and value for each item.
The function *fitness_bkp.m* computes the total value of the items in the knapsack.
Analyze and run the script *run_bkp.m*.
What is the maximum allowed weight?
What are the inequality constraints?
What are the lower and upper boundaries for each item?
What type of representation is used for the individuals of the population?
How many generations does the optimization take?
Comment on the results.