Knapsack Problem example

Important:

Turn on JCL cluster before start this example.

The knapsack problem is a combinatorial problem. The name gives up due to model a situation where you need to fill a backpack with objects of different weights and values.

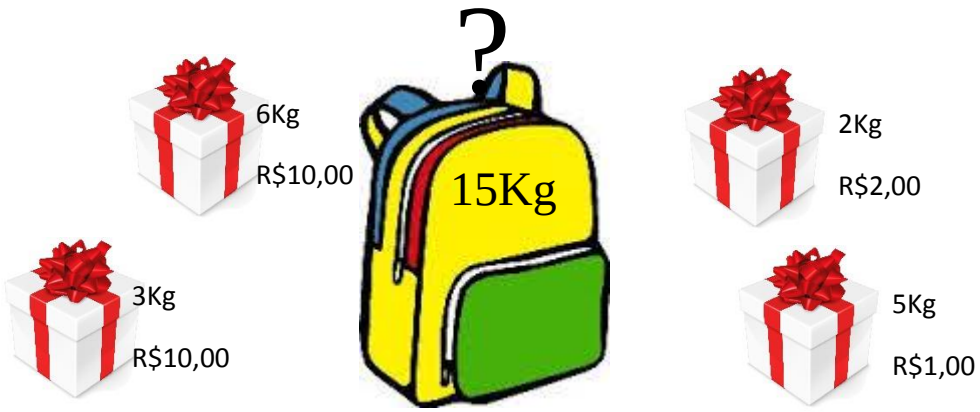

Figure 1. Knapsack problem

The big question of this problem is: what is the best way to put different item values and weights in a container that supports up to a maximum weight, so that the final value within this container is full.

The algorithm using in this example to solve this problem was the brute force where the universe of solutions was divided into n parts.

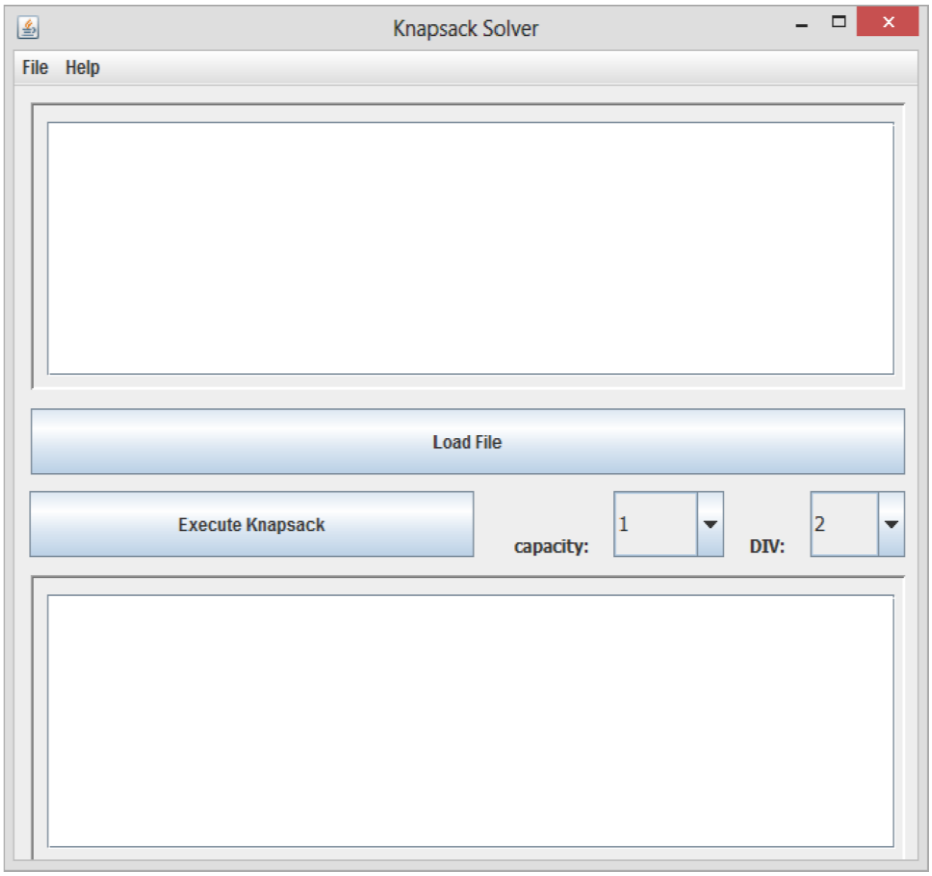

Figure 2. JCL Knapsack application – Start screen

To use the program, you must load a text file with all input data. This file must be created respecting the following restrictions:

• Each line must have the parameters of a given.

• Each data consists of three numbers separated by a space, as follows: item (number representing the item must not repeat), weight and value.

Sample files can be found in the \ Knapsack_month_year_jar \ jcl_app

With the created file, simply use the Load File button or use the menu to upload the file. After that, the data file will appear in the top text box of the program, as shown below:

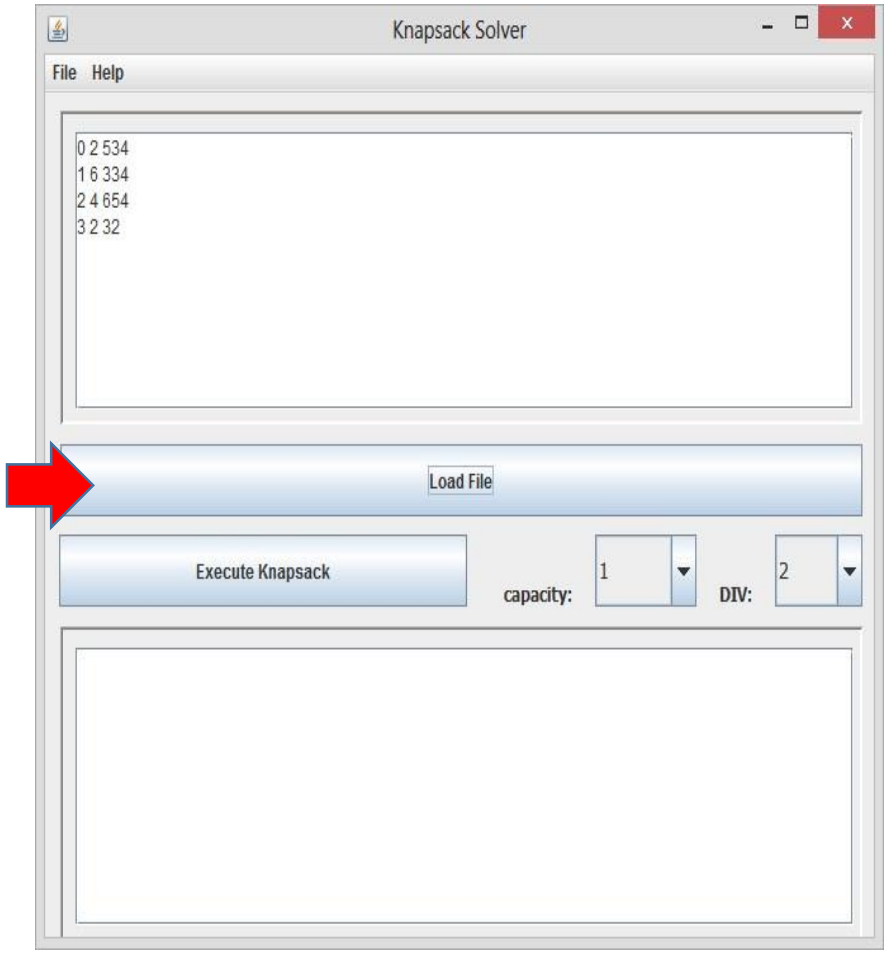

Figure 3. JCL Knapsack application – Load file

After loading the data, we must change the program execution parameters, capacity and DIV:

• Capacity refers to the capacity of the container (see definition of the problem), the maximum weight for the program.

• DIV is how many partitions the problem will be divided.

When selecting the two execution parameters, simply click in Run Knapsack button to run brute force algorithm. This will display all the items that were the solution of the problem, the maximum value and the runtime, as shown below:

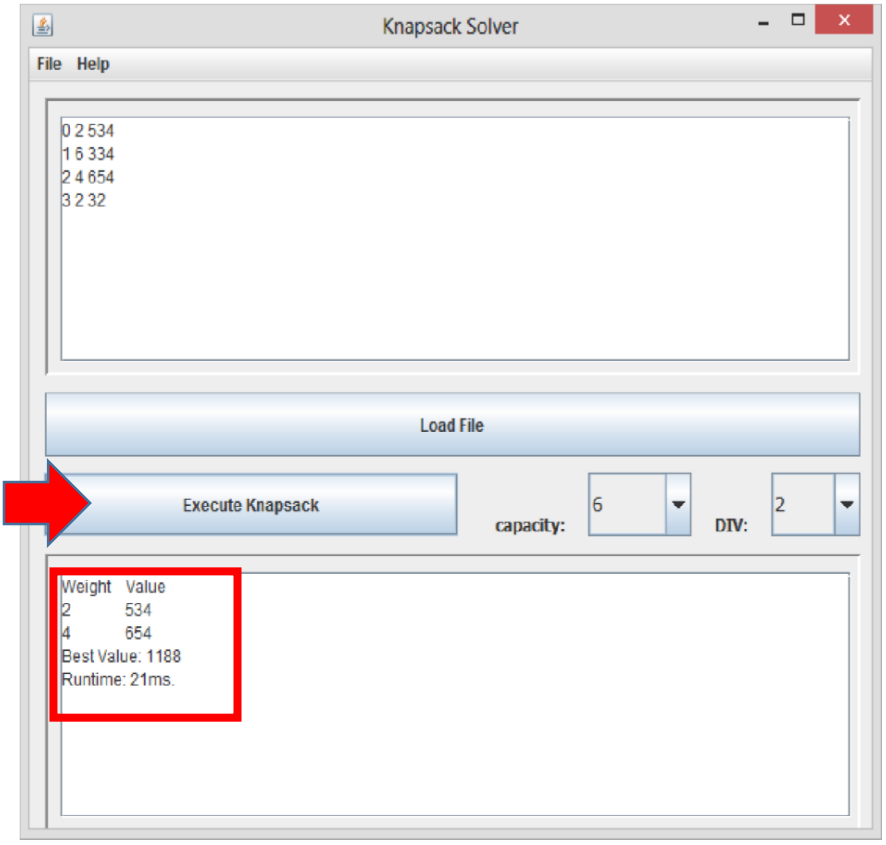

Figure 4. JCL Knapsack application – Run brute force algorithm