MULTI-CONSTRAINT KNAPSACK PROBLEM

Eymen Topçuoğlu & Mehmet Ali Yüksel

To solve the **0-1 Multi-constraint Knapsack problem,** we applied **Genetic Algorithm**. To improve the results we got from the genetic algorithm we applied some **Greedy Techniques** as a part of it.

A genetic algorithm (GA) is a method for solving both constrained and unconstrained optimization problems inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA).

Further explanation of our algorithm is as following:

Definitions:

Population: Collection of Individuals

Individual: Member of a population that store its genome and fitness

Genome: Bit string (i.e. for each character c in genome g, c ∈ {'0','1'}). Each genome represents a possible solution to the problem. Selected items are represented as '1' in the genome. i.e.

G: $genome\ for\ the\ individual$

$G[i]\ =\ $'1', $if\ item\ i\ is\ selected$

Fitness: The overall value of the items selected in the genome.

$fitness\ of\ item\ i,\ f_i\ =\ \Sigma w_{ij},\ j\ =\ 0,\ldots,n\ -\ 1\ where\ n\ is\ the\ number\ of\ knapsacks$

Algorithm Steps:

1) Generation of initial population

A population of individuals are generated based on randomness. Here is the sketch of the algorithm to create a genome of an individual:

G: $genome\ for\ the\ individual$

$G[i]\ =\ $'0', $i\ =\ 0$,..., $n\ -\ 1$ $//initialization$

$for\ i\ =\ 0\ to\ n\ -\ 1\ do$

$if\ (\ rand\ )\ then$

$G[i]\ =\ $'1';

$if\ (\ !\ isFeasible(G)\ )\ then$

$G[i]\ =\ $'0';

Basically, what we are doing is, we randomly flip the "bit i" from '0' to '1' if this replacement

does not violate the constraints.

2) New generation after natural selection

- **Parent Selection**

There are various techniques to choose parents and we used widely known **Tournament Selection**.

4 random individuals are selected from the population as parent nominees to generate children for the next generation. These 4 Individuals are paired to have a “tournament” in which the one with the highest fitness wins. 2 individuals from the tournament (i.e. winners), are passed to the next stage.

- **Crossover**

Crossover is mimicked from nature. Here we have 2 Individuals named as male and female. The male and female Individual’s genomes are divided into parts with the portion of CROSS_OVER_RATE and then these portions are switched to generate a mixed child. Here is the sketch of the algorithm:

$$G1{:}\ genome\ for\ parent\ 1\ ,\ G2{:}\ genome\ for\ parent\ 2$$

$$cutting\ index,\ c := n\ -\ [n\ *\ CROSS\_OVER\_RATE]$$

$$genome\ of\ first\ child,\ c1 :=\ G1[:c]\ +\ G2[c:]$$

$$genome\ of\ second\ child,\ c2 :=\ G2[:c]\ +\ G1[c:]$$

As an example let's say we have 10 items and the crossover rate is 0.5 (half).

Male Individual’s genome: **1 0 1 1 0 1 0 1 1 1**

Female Individual’s genome: **1 1 1 0 1 1 0 0 0 1**

Children Individual’s genomes: **1 0 1 1 0 1 0 0 0 1** & **1 0 1 1 1 1 1 1 0 1**

- **Mutation**

Mutation is applied to children yielded from Crossover. Each bit (each item selection) in the genome string of the Individual has a possibility to flip and this possibility is randomly determined by the program. The genome is traversed, and at each iteration of the traversal, a random number is generated by the program and if this random number is below the MUTATION_RATE constant, then a mutation occurs and the bit at the current index is flipped.

$$for\ i\ =\ 0\ to\ n\ -\ 1\ do$$

$$if\ (\ rand\ <\ MUTATION\_RATE)$$

$$flip(\ G[i]\ )$$

- **Repair**

If the mutated genome is not feasible, a repair method is employed to

re-establish the feasibility. Here are the steps:

1) Characters of the bit string are reordered with respect to profit based sorted items. Here by profit we mean:

$$v_i\ :=\ value\ of\ item\ i$$

$w_{ij} := \ weight\ of\ item\ i\ in\ knapsack\ j$

$profit\ of\ item\ i,\ p_i = v_i / \Sigma w_{ij},\ j = 0,\ldots, n-1\ where\ n\ is\ the\ number\ of\ knapsacks$

After this step, items having the most profit appear at the end of the string.

2) Traverse the genome from the beginning, replacing '1' with '0' until a feasible solution is found:

G: $genome\ for\ the\ individual$

n: $length\ of\ the\ genome$

$for\ i = 0\ to\ n-1\ do$

$if\ (G[i] == '1')\ then$

$G[i] = '0';$

$if\ (\ isFeasible(G)\ )\ then$

$break;$

3) Traverse the genome from the end, replacing '0' with '1' if it does not violate any constraint:

G: $genome\ for\ the\ individual$

n: $length\ of\ the\ genome$

$for\ i = n-1\ to\ 0\ do$

$if\ (G[i] == '0')\ then$

$G[i] = '1';$

$if\ (!\ isFeasible(G)\ )\ then$

$G[i] = '0';$

4) Reorder the characters of the bit string to their default index.

- **Replacement**

  2 individuals in the population having the lowest fitnesses are replaced with the 2 children obtained by these operations.

3) Return to step 2 if the termination condition is not satisfied

4) Return the best individual (e.g. having the largest fitness)

References: Reference1, Reference2, Reference3, Reference4, Reference5