AI Searching Techniques

Problem Solving with Genetic Algorithm

Dr.Sherif Saad

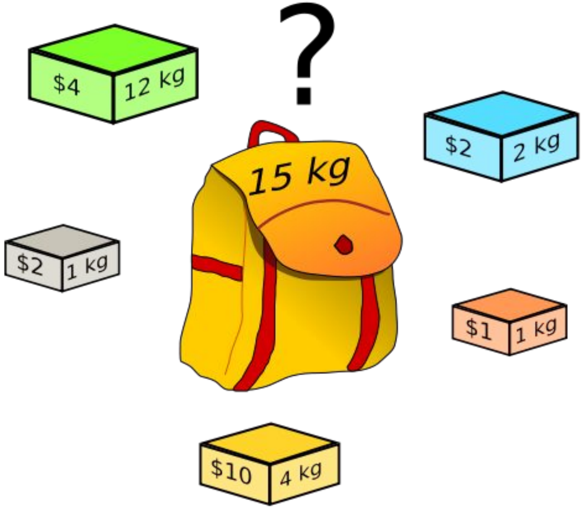
Outlines

- Problem Encoding
- Initialize Population
- Parents Selection
- Crossover
- Mutation
- Termination

0-1 Knapsack Problem: Definition

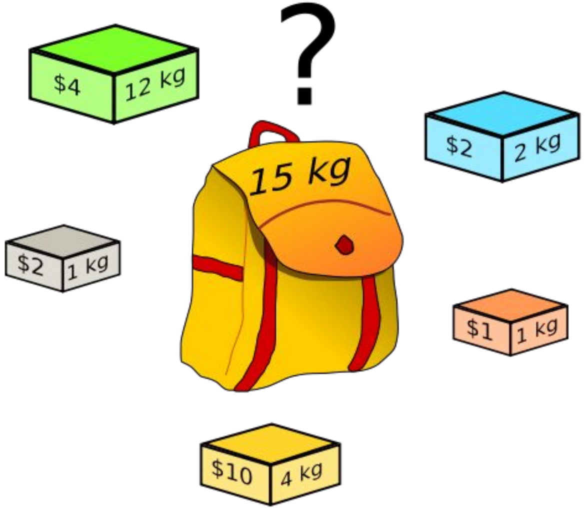
Given weights and values of ***n*** items, put these items in a knapsack of capacity ***W*** to get the maximum total value in the knapsack.

This an example of combinatorial optimization problem.



0-1 Knapsack Problem: Example

Assume we have the following set of items and a knapsack of capacity = 9. Find the items that maximize the profit with the capacity limit



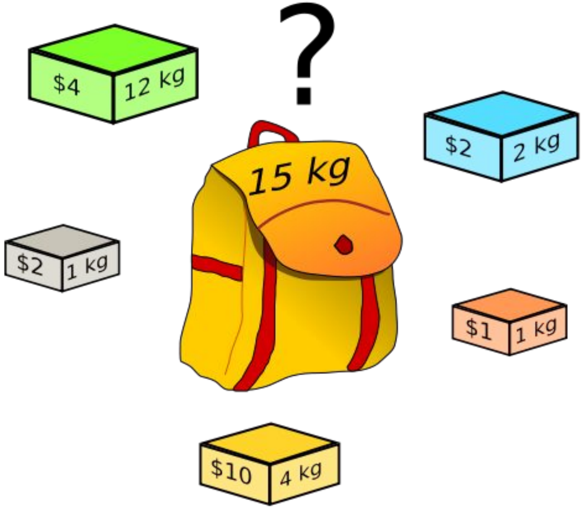
Item	A	B	C	D	E	F	G
Profit	6	5	8	9	6	7	3
Weight	2	3	6	7	5	9	4

How can we solve this problem using greedy algorithm?

0-1 Knapsack Problem: Greedy Solution

Calculate the ratio of profit to weight

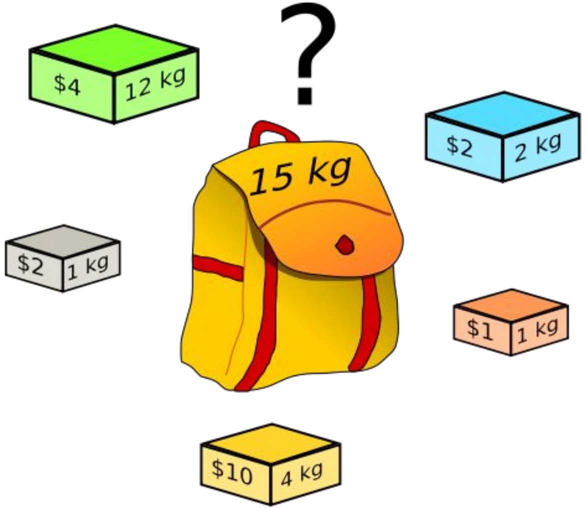
Item	A	B	C	D	E	F	G
Profit	6	5	8	9	6	7	3
Weight	2	3	6	7	5	9	4
P/W	3	1.67	1.33	1.29	1.2	0.78	0.75



The solution using greedy method is [A, B, G] , where the profit = 14 and the weight = 9 (Can you come up with a better solution?)

0-1 Knapsack Problem: Genetic Algorithm Solution

Item	A	B	C	D	E	F	G
Profit	6	5	8	9	6	7	3
Weight	2	3	6	7	5	9	4
P/W	3	1.67	1.33	1.29	1.2	0.78	0.75



How big is the solution (search) space?

0-1 Knapsack Problem: Engineering a GA solution

Problem Encoding:

- We can use a binary encoding where each chromosome is a binary string of length n (n is the number of items)
- Each bit from this binary string denotes whether an item is included in the knapsack 1 or not 0

A	B	C	D	E	F	G
1	0	0	1	0	1	1

0-1 Knapsack Problem: Engineering a GA solution

Population Initialization:

```
''' Let P an empty population list'''
P = list()
while P.size() < n:
    ''' empty chromosome of size m, where m is the number of items '''
    p = list()
    randomly assign 0 or 1 for each itme in 'p'
    add the chromosome 'p' into the population 'P'

return 'P'
```

0-1 Knapsack Problem: Engineering a GA solution

Fitness Function:

- The sum of the profits of all the items included in the knapsack.
- The fitness function is subject to one constraint, which is the capacity of the knapsack.

$$maximize\{f(x)\} = maximize\left\{\sum_{i=1}^{n} x_i p_i\right\}$$

$$subject \quad to = \sum_{i=1}^{n} x_i w_i \leq W \quad x_i \in \{0,1\}, i = 1,2,3,\ldots,n$$

0-1 Knapsack Problem: Fitness Function Algorithm

```
for each chromosome 'p' in the population 'P':
    fitness = sum of all profits for each gene in 'p' equal 1
    capacity = sum of all weights for each gene in 'p' equal 1

    while capacity > knapsack.capacity:
        ''' Try to fix the chromosome '''
        Randomly choose a gene with value 1 and flip it to 0
        update capacity of 'p'
```

0-1 Knapsack Problem: Engineering a GA solution

Parents Selection

- Using Roulette-Wheel Selection as a fitness-proportionate selection method.
- Hybrid Fitness-Rank selection (design problem specific selection method). That consider both the fitness and the rank of each chromosome

0-1 Knapsack Problem: Engineering a GA solution

Roulette-Wheel Selection

```
 Wheel = sum of the fitness values of all chromosome in the population
FixedPoint = random value betweeb 0 and 'Wheel'
rotate = 0
for each chromosome 'p' in the population 'P':
    rotate = rotate + fitness('p')
    if rotate > = FixedPoint:
        return 'p'
```

0-1 Knapsack Problem: Engineering a GA solution

Hybrid Fitness-Rank selection:

- Sort the individuals in the population by fitness in descending order.
- Divide the sorted list into equal groups (e.g. 3 or 4 groups)
- Randomly select an individual form the first group with probability 60% and from the second group with 30% probability and from the third group with 10% probability

Write an algorithm to implement this method?

0-1 Knapsack Problem: Engineering a GA solution

1. Which selection method is better roulette-wheel or the hybrid fitness-rank selection?
2. How could we compare them?
3. When do you think they will have the same effect of the GA performance?
4. What are the other factors that affect the parents selection method?

0-1 Knapsack Problem: Engineering a GA solution

Crossover

- Use either single point crossover of multi-point crossover
- Generate random SPC value between 0 and M.
- Select a crossover rate between (0,100], example 75% crossover rate means 75% of the individual of the new generation are the result of crossover operator and 25% are copied from the current generation.

What is the good crossover rate?

0-1 Knapsack Problem: Engineering a GA solution

Mutation

- As we mentioned before mutation is important to prevent the GA from getting stuck into local optimum
- Select a a mutation ratio either for the population:
  - How many offsprings will go through mutation
  - How many genes of each mutated offspring will be updated
- What is the a good mutation ratio?

0-1 Knapsack Problem: Engineering a GA solution

Termination

- C1: We put limits on the number of generations
- C2: We put limits on the fitness-change for last ***g*** number of generation

Example:

  - Does **85%** of the individual in the population have the same (close) fitness values ***and|or*** is the number of generations greater than the maximum allowed number of generations?

0-1 Knapsack Problem: Engineering a GA solution

Termination

- Maximal Generations
- Hitting a predefined fitness value or range
- Fitness improvement or change:
  - Running Mean: the difference between the average fitness of the current generation and the last g generation is less than or equal a given threshold
  - Min-Max: the difference between the best and the worst fitness values of the current generation is less than or equal a given threshold
  - Standard Deviation: the std of the current generation fitness is less than or equal a given threshold

Questions