

Local search

AI HW2



December 16, 2022

ısmail tosun – yusuf burak peker

180316030 - 180316058

# Developing Environment

We use Python 3.10 in an Ubuntu container with Visual Studio Code and SimpleAI libraries that written for Python.

# Problem Formulation

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. We used *hill\_climbing*, *hill\_climbing\_random\_restarts*, *simulated\_annealing* and *genetic* algorithms.

## State Definitions:

Our items represent as python tuple for example : ‘(10,60)’ 10 is the weight of item and 60 is the value of item. If we added this item to knapsack mark this as 1 if we didn’t add this item to knapsack mark item as 0.

## Actions:

We have 2 main actions ‘add’ and ‘remove’, if our state graph is empty we can only use ‘add’ action.

## Result:

If the given action is ‘add’ it’s add 1 to and of the state graph, otherwise remove last element of the state graph.

## Value:

Its check items using state graph and if the item selected take sum of all selected items weight and values than compare the sum of weight with max weight of knapsack, if sum of weight is equal or less than max weight return sum of values else return zero because state is invalid in this situation.

## Generate Random State:

Its generate random a state with using Python random library.

## Crossover:

It make crossover between two state.

## Mutate:

Make a random mutation, add or remove a random item to knapsack.

## Sample cases:

It is a test function for us, its run predefined scenario and plot diagram for its.

# Results:

We show some of the test cases here check source code (*sampleTestCases()*) functions to see all test cases which we used or check the ***TestResults.txt*** files to see all results.

* Test Case: 1
  + Max Weight: 12
  + Items: [(5, 12), (3, 5), (7, 10), (2, 7)]
  + Diagram, timeline

    Description automatically generated
* Test Case: 2
  + Max Weight: 15
  + Items: [(12, 4), (2, 2), (1, 1), (1, 2), (4, 10)]
  + Timeline

    Description automatically generated
* Test Case: 3
  + Max Weight: 25
  + Items: [(24, 24), (10, 18), (10, 18), (7, 10)]
  + Timeline

    Description automatically generated
  + Same case with mutation change 0.5:
    - Diagram, timeline

      Description automatically generated

# Discussion:

We talk about 3 different example here but you can find 7 different result in ***TestResults.txt*** and you can find all diagrams for all test cases in ***graphs*** folder.

In all Cases we observed:

* Hill climbing is the fastest algorithm but its mostly not optimal
* Hill climbing random restart show us the key point for hill algorithm is the starting point, its still fast and it starts algorithm in different states so its optimality improve significantly.
* Simulated Annealing is the third fastest but sometimes it cant find any solution depends on items and initial states so its not optimal
* Genetic algorithm is significantly slow than others but it always find the optimal solution than others. Also changing the *mutation\_chance* effect the speed, mostly good but sometimes bad, it’s look like randomness is necessary in acceptable limits.
* Test Case 1 graphs:

Chart, bar chart

Description automatically generatedChart

Description automatically generated

* Test Case 2 graphs:

Chart, bar chart

Description automatically generatedChart

Description automatically generated

* Test Case 3 graphs:

Chart, bar chart

Description automatically generatedChart

Description automatically generated