Zijie (Frank) Wan

2/6/2022

Artificial Intelligence (Spring 2022)

Prof. Blossom

**Application of Genetic Algorithm and Simulated Annealing in Knapsack Problem**

**Summary**

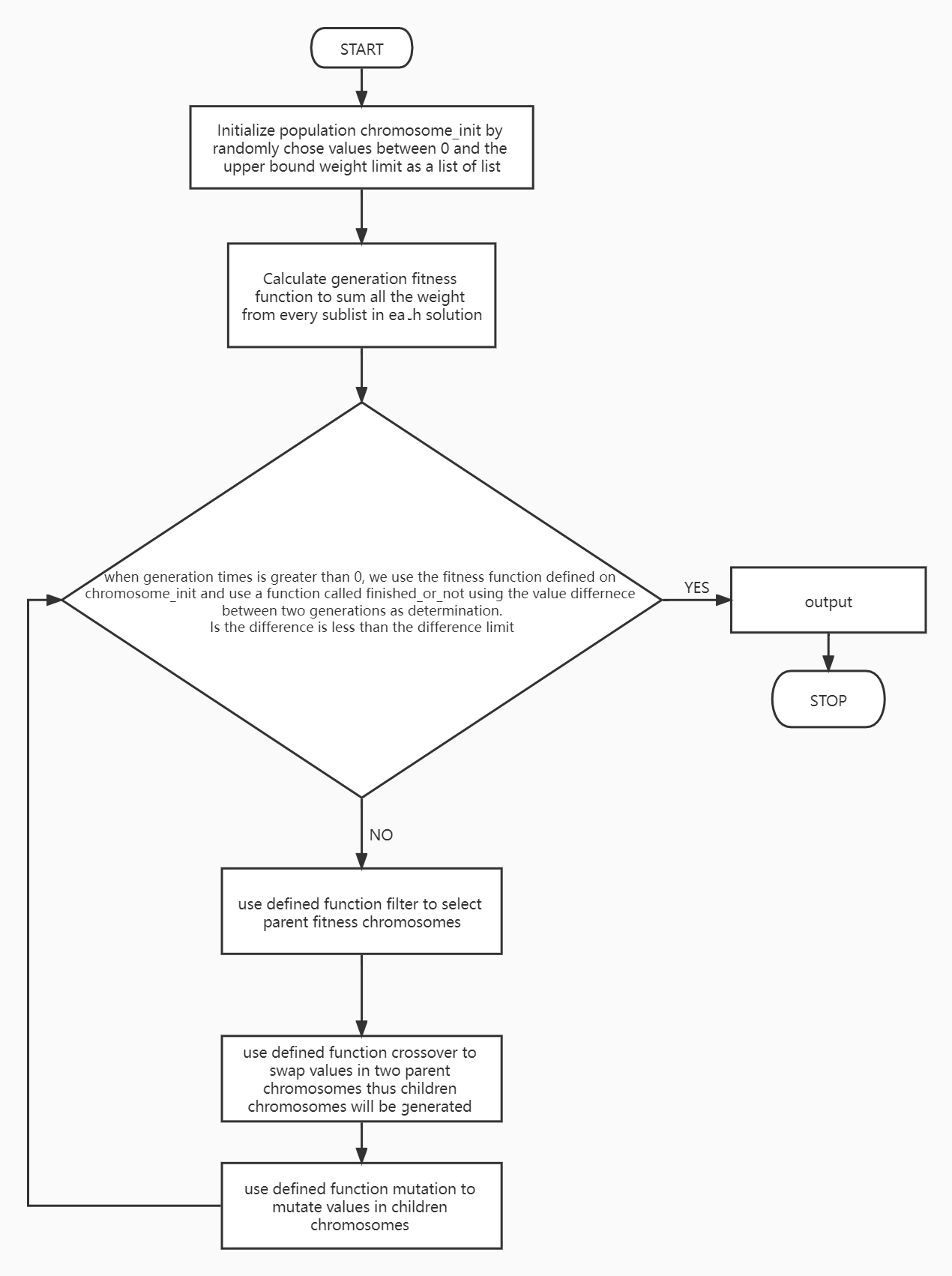
Genetic algorithm and simulated annealing are popular optimization method nowadays when people want to find optimal solutions either out from the entire population or from a sample population. In the knapsack problem, both methods are applicable and have their own processes to deal with the problem. Simulated annealing algorithm accepts a solution worse than the current solution with a certain probability, so it may jump out of the local optimal solution and reach the global optimal solution. Genetic algorithm takes all individuals in a population as the object, and uses randomization technology to guide the efficient search of a coded parameter space. The main difference of the two methods is that simulated annealing uses a single individual to evolve, and genetic algorithm uses a population to evolve. In this paper, we will create flowcharts and pseudocodes to simply illustrate the processes of how each method work on the knapsack problem.

**Introduction**

Knapsack problem is a NP (non-deterministic polynomial) complete problem of combinatorial optimization which means any NP problem can be transformed into a NP problem by a polynomial time algorithm. Given a group of items, each type of items has its own weight and value. Within the limited total weight and bounded amount of items, how can we choose to make the total value of the items the highest. With the variables, constrains and objective given in this problem, we can use genetic algorithm and simulated annealing to get optimal solutions.

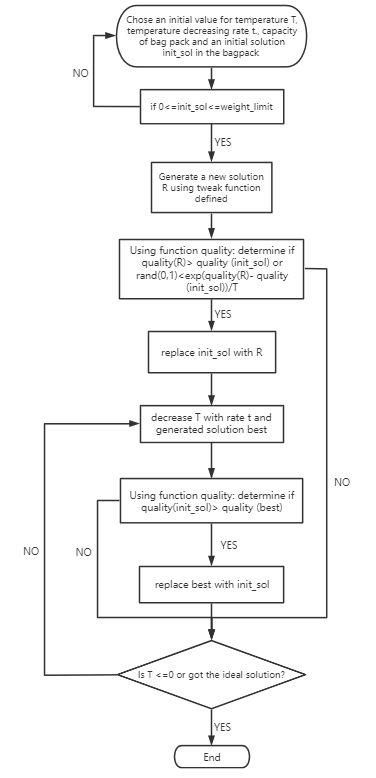
**Flowcharts + pseudocodes**

***Genetic Algorithm Flowchart of Knapsack Problem***



When we are using genetic algorithm to solve knapsack problem, we need to initialize population by randomly chose 0 to upper bound weight limit to fit in each chromosome. We created a function called initialize\_population to create the initial chromosome. Then we defined the fitness function by adding every value in each sub-list in chromosomes. We also created a function called finished\_or\_not as a condition for the entire work flow process (by comparing the difference of weight difference between the current chromosome weight and the last chromosome weight and the finished\_limit we set). If chromosomes cannot pass the condition, then we are using filter function to select parents from the existing chromosomes by comparing with the weight limit. When we get children chromosomes from filter, we are going to apply crossover and mutation functions on the children chromosomes. Then we are using finished\_or\_not function to determine the solution is optimal or not.

***Simulated Annealing Flowchart of Knapsack Problem***



In the process of simulated annealing, we have to set an initial temperature T, weight limit, temperature decreasing rate, generation times per decrease and a random solution init\_sol. Firstly, we will generate a new solution by randomly chose numbers as weights for every type item from 0 to the upper bound limit. Then we need to create another solution using tweak function we created. To compare weights for two solutions, we created a function called quality. If quality of solution R is greater than the quality of solution init\_sol or the exp(quality(R)-quality(init\_sol))/T is greater than a random value from 0 to 1, then we replace init\_sol with R. Now, we need to decrease the temperature with rate t, thenwe compare the quality of solution init\_sol with solution best (solution best is equal to the initial solution). If the quality of init\_sol is greater and the value of this solution is less than the weight limit, then we replace solution best with solution init\_sol. Run this process until T<0, then return the best solution.

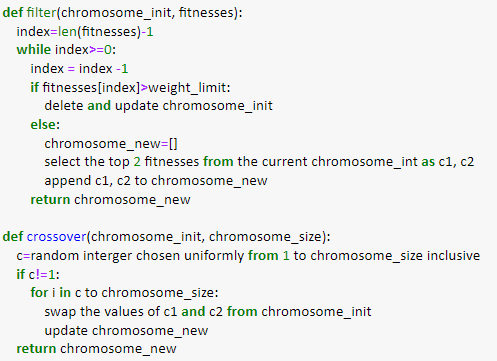
**Conclusion**

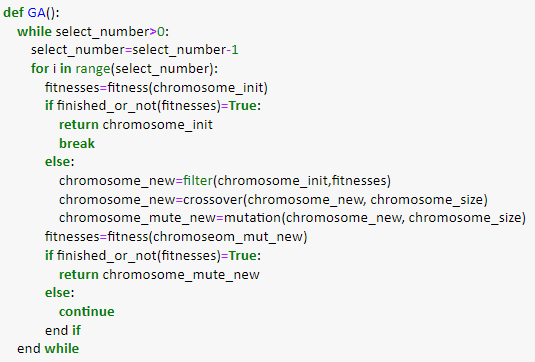
As we can see from pseudocodes, both methods have variables and constraints in their process to get the ideal result. However, in simulated annealing, we have parameters T and t to search from all the possible results. Genetic algorithm has a straighter forward concept to get optimal solution by crossover and mutation. One thing we need to pay attention is that, this problem is bounded, so it was complicated when initialize population for both methods. The simulated annealing algorithm can finally get the best solution, but its speed is affected. The speed of the algorithm process is likely to be very slow due to the annealing process and probability problems. Genetic algorithm may not get the best optimal solution all the time, but it can get a broad space of approximate optimal solution, which makes the solution more selective, and will explore the solution set infinitely close to the optimal solution when the optimal solution cannot be obtained.

**Appendices**

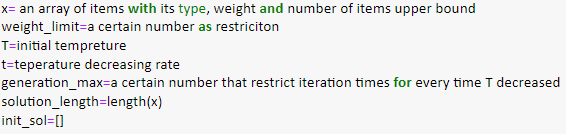
***Genetic Algorithm Pseudocode***

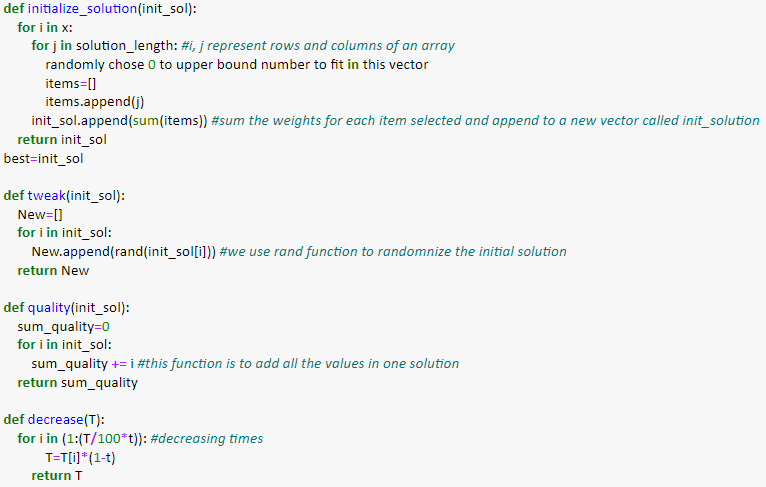


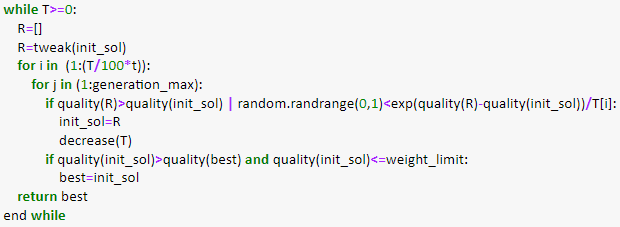




***Simulated Annealing Pseudocode***







***Reference links & book:***

*Solving the knapsack problem with a simple genetic algorithm*. DataMiningApps. (n.d.). Retrieved February 8, 2022, from <https://www.dataminingapps.com/2017/03/solving-the-knapsack-problem-with-a-simple-genetic-algorithm/>

*limit exceeded - CITESEERX*. (n.d.). Retrieved February 8, 2022, from <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.2222&rep=rep1&type=pdf>

Luke, S. (2015). Essentials of metaheuristics: A set of Undergraduate Lecture notes. editor no identificat.