

**IE343 - Heuristic Optimization Methods**

**Term Project**

**by**

**Ece Memişoğlu**

**Aslı Mehli**

**Supervised by: İhsan Yanıkoğlu**

**ÖZYEĞİN UNIVERSITY**

**FACULTY OF ENGINEERING**

Firstly, we have an initial temperature and cooling rate related to the optimization process. We need to find the best combination of values and weights while considering a total capacity of 300. For instance, if we choose 68 as the value, it will affect our capacity 21.

We refer to the current solution as the initial solution. We don’t choose any of them at first. The best solution and the current solution both start with the value 0. We generate a random index using a random library and set the current temperature as the initial temperature.

As long as the temperature value is greater than 1, we tried to get closer to the optimization, and we did this by sending it to the current solution and random library. The generated random solution clones the current solution, it puts all values there and then chooses a random index which we call the index. For example, should I choose the 25th item or not? It chooses a random index and makes that index true if the previous solution is false and makes false if the previous solution is true. This helps us evaluate the potential impact of including or excluding that particular item. Then we returned the decision’s solution, which is changed, and we compared the modified solution with the real solution to determine whether the newly generated random item should be included or not. Then return the whole solution and if the solution has this index, add its value and add its weight. If the weight is more than the capacity, the total value is set to 0, and we can’t accept it, but if not, the return the calculated value. If the selected value does not exceed the capacity, we send the current value, neighbor value, and temperature. If the neighbor value (new solution) is better than the current solution, we accept it without hesitation. However, if the neighbor value is worse, we return the other value, which is calculated by mathematical formulas in the question. Then we pull it from a random library, so we look at it whether we accept it or not. According to this probability, we have a possibility to accept it. If we accept, we change the current value, and if the value we accept is greater than the best value, then we accept it as the best solution. Then we specify how much we will lower the initial temperature as we approach each solution, and we are constantly optimizing until the temperature drops below 1. We tried to get closer to the best solution. In the end, we find the best solution and value.

QUESTIONS:

**1- How does the execution time change when the cooling rate increases? Report it by running the same algorithm for the given dataset.**

The cooling rate is used to decrease the temperature. For example, if we set the cooling rate to 1, the temperature will reach 0 in a single iteration. Thus, ending the temperature reduction process more quickly. This way, we can decrease the temperature faster. We also observed that the best value changes when we change the cooling rate value. For instance, when we set the cooling rate as 0.03, we saw that the best value was 1439, and when we subtracted the cooling rate by 0.15, it gave the result of 1284. To be more sure, when we set the cooling rate value to 0.70 this time, we observed that the best value was 454. That is, we realized that the best value decreases as the cooling rate increases.

**2- How does the execution time and solution quality change when the difference between starting temperature and stopping temperature is increased? Report it by running the same algorithm for the given datasets.**

If the difference between starting temperature and stopping temperature is increased, execution time is increased, and solution quality is decreased