2023-2024

MOHAMED-HOUSSEM REZGUI
YAHYA EL OUDOUNI
SASCHA CAUCHON

ESILV
ENGINEERING SCHOOL DE VINCI PARIS

AI ALGORITHMS

The problem and its challenges

After following an AI algorithms course rich in content, we students at ESILV were given the task of developing an effective solution to a problem. A biscuit manufacturing comapny came to us to help them produce a series for christmas. Using what we learned in the course we had the objective of maximizing the biscuit production from a single roll of dough all while ensuring the highest possible profit. The company had a specific length for the roll of dough and wanted to produce 4 types of biscuits. The task at hand was not simple as we had to respect a certain number of constraints. Indeed, as the roll of dough contained irregularities, referred to as defects, there was a specific threshold for the maximum number of defects each biscuit could contain. Otherwise it would not be marketable. We also needed to ensure that there was no overlap in biscuit placement and see that the assigned biscuits would not exceed the length of the roll. The main challenges of this problem were:

- Find an algorithm that would be suitable for solving the problem
- Finding the optimal assignement of biscuits to maximize profit all while considering the constraints of size, defect threshold, and overlapping biscuits.

Formulate and implement the problem

From the information that was provided by the company, we thought it would be best to take the following approach to solve the problem at hand. We first selected two algorithms that we thought would be relevant for solving this problem. For each algorithm, we took different approaches when it came to implementing them. The two algorithms that we used were **Greedy Search** and **Genetic Algorithm**. The reasons why we choose these two algorithms will be explained in a later section.

Concerning the **Genetic Algorithm (GA)** we started by creating 2 classes. The first class was "Dough" and the second one "Biscuit". We believed that this would help in the structure and organization of our code. You can find more detail below:

1. The **Dough** class which will represent the roll of dough has the following attributes and methods:
    - Attributes:
        - length: the length of the roll of dough which in this case is 500 units.
        - defects: a dictionary data structure that will contain the defects with the class type and position on the roll
    - Methods:
        - __str__(self): dunder method to show the representation of the class
        - load_defects(self, file_path): to load the defects from the given file "*defects.csv"* and add them to the defects attribute
2. The **Biscuit** class can represent the types of biscuit given by the company. It will contain the following attributes and methods:
    - Attributes:
        - id: id representing the type of the biscuit (0 to 3)
        - size: size of a biscuit
        - value: price of the biscuit
        - max_defects: dictionary data structure to store the maximum allowed defects for each class
    - Methods:
        - __str__(self): dunder method to show the representation of the class
        - get_max_defects(self, defect_class): get the maximum allowed defects for a specific class type
        - check_defect_threshold(self, defects): to check if the biscuit that is placed on the roll contains a defect number inferior to the threshold specified by the manufacturer

By using these classes we have a more object-oriented approach which can lead to cleaner and more maintanable code. The steps that we will take to solve the problem using the **Genetic Algorithm (GA)** will be the following:

The solution that will be found using the the help of the **Genetic Algorithm) GA** will represent a valid assignement of biscuits on the roll. The genotype representation will be a list where each element represents a position on the roll, and the value of the element indicates the biscuit type placed at that position. The length of the data structure will be equal yo the length of the roll of dough. The algorithm will then do its job to manipulate and evolve the genotype to explore the best combination of biscuit placements all while considering the different biscuit types and their constraints.

For the first step we will generate an initial population of potential solutions randomly where the potential solution is the assignement of the biscuits on the roll. Since each individual is a solution it must be valid and we will check at the same time if it respects the constraints.

Then we define the fitness function which will guide the **Genetic Algorithm (GA)** towards a better solution. In our case the fitness function should consider the total value of the assigned biscuits.

$$max \quad f(x) \quad with \quad f(x) = \sum_{i=1}^{n} x_i \quad and \quad x_i \quad the \quad value/price \quad of \quad each \quad biscuit \quad type.$$

After calculating the fitness, we will go to the selection phase consisting of selecting individuals for the population so that they can be parents for the next generation based on fitness. We will use a combination of elitism and roulette wheel selection. Individuals with higher fitness will be more likely to get selected.

Performing crossover between selected individuals will create children. The crossover's operation will combine genetic information from the 2 first parents to generate new individuals with a mix of their characteristics.

Finally we have the mutation phase. In this part we introduce random changes in the children's genetic information to explore new regions of the solution. Then in the genetic algorithm we initialize the population and go through an evolution loop so that at the end we come out with the best solution.

Two alternative problem-solving approaches

As it was stated before we chose to use the **Genetic Algorithm (GA)** and the **Greedy Search** which are both local search algorithms. In this section, we will give a little definition of both algorithms, explain why this algorithm was chosen, and how it is relevant for solving the problem.

Genetic Algorithm

A **Genetic Algorithm (GA)** is based on the principles of natural evolution. It performs different operations such as selection or crossover to find a solution to a specific problem. The potential solutions evolve over multiple iterations. The solution of each individual is evaluated based on their fitness. This influences its chances of being selected for the next generation.

The reason why we chose the **Genetic Algorithm (GA)** for this problem is they are commonly used to generate high-quality solutions to optimization problems. Another reason why we chose **Genetic Algorithm (GA)** in addressing the biscuit manufacturing is because of its effectiveness to navigate and optimize the problem's complicated constraints. It adapts to different distributions of defects and varying biscuit characteristics to perceive the most optimal arrangement. This method is suitable for the problem, where a couple of variables and constraints ought to be balanced to maximize profit. The GA's iterative refinement process guarantees the evolution toward an ideal answer, adeptly handling the complexity and variability inherent on this manufacturing situation.

Greedy Search

A **Greedy Search Algorithm** is a methodical approach to problem-solving that makes the local optimal choice at each step with the hope of finding a global optimum. In the context of biscuit manufacturing, this algorithm selects the placement of biscuits on the roll of dough based on immediate profit.

Why a **Greedy Search Algorithm** ?

- **Simplicity and speed** : The greedy algorithm is relatively simple to implement and quick to execute. For complex problems like this one, a fast and understandable algorithm offers a practical advantage.
- **Relevance to the problem** : The greedy approach is directly aligned with the goal of maximizing the total value. By systematically choosing the option that seems best at each step (here, the biscuit with the best value-to-size ratio), the greedy algorithm aims to find a solution close to the optimum.
- **Handling constraints**: The biscuit manufacturing problem includes constraints like defect positions, sizes, and defect thresholds for each biscuit. A Greedy Algorithm can sequentially address these constraints by evaluating the best biscuit to place at each step, thus efficiently utilizing the dough roll.
- **Suitability for one-dimensional problem**s: Since the biscuit problem is essentially one-dimensional, the Greedy Algorithm is particularly suited for it. Its approach of iteratively choosing the best local option can efficiently exploit this linear structure.

To Sum up :

The greedy approach is chosen for its ability to quickly provide solution for a complex problem of maximization under constraints. While it may not always guarantee an optimal solution, it offers a balance between efficiency, simplicity, and performance, which is often desirable in practical situations like industrial production.

Conclusion and reflections

This project was a good opportunity to apply the different algorithms that we learned in a real-world optimization problem. It highlights the importance of understanding the specific constraints and requirements of a given scenario. By doing such a project we needed to pay close attention and go through the benchmark over and over again to not miss any important information. We also got to see the power of AI and how it could be used to solve problems. It also showed us that planning what we needed to do before writing the code was very important. Through the project we also encountered some challenges. It was difficult to manage the multiple constraints and implement them in the algorithms. The presence of defects added complexity and the algorithms needed to adapt during the optimization process.

Moreover, this project provided us a rich learning opportunity in algorithm selection, problem-solving, and dealing with real-world constraints. The challenges encountered have been valuable in understanding the practical applications and limitations of algorithms.

In summary, this biscuit project illustrated the importance of AI in an optimization problem, and even though it was challenging in the end, we learned a lot.