Exercise: Write an Optimization Problem

Viviana Arrigoni

March 10, 2025

Consider N sensors $s_1, \dots, s_N$ sensing an environment. Each sensor s_i sends some data to an IoT gateway every second. Processing of data produced by sensor s_i requires r_i resources (e.g., number of operations). In addition, each sensor has a different priority p_i, which defines the importance of the data produced by s_i, $\forall i = 1, \dots, N$. The gateway has limited resources g (i.e., g is the number of operations that it can perform in one second). The gateway collects the data coming from the sensors and processes it, limited to its resources. If the resources required for processing the data of all sensors are greater than the available resources g, the gateway processes the data coming from selected sensors in such a way that the overall priority is maximized.
Write an optimization problem that models this system. Is the problem hard to solve? What are the possible ways to solve it?
(Hint: you need N binary decision variables $x_1, \dots, x_N$.)

Solution.
We introduce $x = (x_1, \dots, x_N)$ such that $x_i = 1$ if the gateway processes data of sensor s_i, and 0 otherwise. We can formulate the problem as follows:

$$
\begin{aligned}
\max_{x \in \{0,1\}^N} \quad & \sum_{i=1}^{N} x_i p_i \quad \text{maximise overall priority} \\
s.t \quad & \sum_{i=1}^{N} x_i r_i \leq g \quad \text{IoT gateway cannot process more than its available resources allow it to} \\
& x_i \in \{0,1\} \; \forall i = 1, \dots, N \quad \text{decision variables are binary}
\end{aligned}
\tag{1}
$$

It is trivial to see that this is equivalent to the 0-1 Knapsack Problem (hence, it is NP-complete). To solve Problem 1, we can use any approach that works for the 0-1 Knapsack problem. We could write a program using Gurobi to find the optimal solution. We could opt for a greedy approach, but it does not provide any optimal approximation (it provides valid solutions to the problem, sometimes they are optimal, sometimes they are close to optimal, some other times they are very far from the optimal value). Another possible way to solve it is through dynamic programming. In fact, Problem 1 satisfies both optimal substructure (the optimal solution can be discovered in a recursive way, i.e., searching for optimal solutions to sub-problems) and overlapping subproblems properties (when searching for optimal solutions recursively, we end up evaluating the same sub-solutions multiple times, similarly to what happens with recursive Fibonacci), which makes it suitable for dynamic programming. Wikipedia `https://en.wikipedia.org/wiki/Knapsack_problem#0-1_knapsack_problem` shows the dynamic programming implementation for the 0-1 Knapsack problem, which works also for Problem 1, considering that the knapsack capacity is g, the items are the sensors, their weights are their required resources r_i, and their values are their priorities p_i. The idea is the following: consider the sequence of sensors $s_1, \dots, s_N$, and the relative sequences of priorities $p_1, \dots, p_N$ and of required resources $r_1, \dots, r_N$. Define the function $P(i, r)$ as the maximum overall priority that we can get from executing tasks coming from the first i sensors using resources $\leq r$. $\forall r = 0, 1, \dots, g$, $P(i, r)$ can be defined recursively as follows:

- $P(0, r) = 0$
- $P(i, r) = P(i-1, r)$ if $r_i > r$ (Not enough resources left, we cannot satisfy task i).

- $P(i,r) = \max\{P(i-1,r); P(i-1,r-r_i)+p_i\}$ if $r_i \leq r$ (we have enough resources left. We can either not satisfy task i and, in this case, $P(i,r) = P(i-1,r)$, or satisfy it. In such a case, we update the remaining resources with $r - r_i$ and the achieved overall priority by adding the term p_i. We choose the option that maximizes the achieved priority).

We can then use tabulation for filling in a table of size $(N+1) \times (g+1)$, fill it with the values of P, and retrieve the optimal solution.

Simple example: $N = 3$, $g = 5$, $p_1 = 2, p_2 = 2, p_3 = 3$, $r_1 = 2, r_2 = 3, r_3 = 2$.

$i \downarrow r \rightarrow$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	2	2	2	2
2	0	0	2	2	2	4
3	0	0	3	3	5	5

Optimal value is 5. The optimal solution is $x^* = (1, 0, 1)$.