Final Project

Algorithmic Methods for Mathematical Models (AMMM)

Elena de Paz Obiols
Leidy Vanesa Vidales
December 8, 2025

Contents

1 Problem statement

Crime is soaring in Gotham City. Batman feels he is no longer able to cope with it. For this reason he has decided to update his system for monitoring the streets. More precisely, he wants to hide cameras at street crossings. He quickly realizes that he does not need to place a camera at each crossing, as a camera at one of the ends of a street can also cover the other end if it is poweful enough.

There are K models of cameras available in the market, which we will refer to with numbers $1, 2, ..., K$. The price in euros of a unit of model k (where $1 \leq k \leq K$) is P_k. Other interesting attributes are the range R_k (with $1 \leq R_k \leq 49$), the autonomy A_k (with $2 \leq A_k \leq 6$), and the power consumption C_k per day (with $0 < C_k$, in euros). The larger the range, the farther the distance the camera can cover. But to avoid malfunction, a camera cannot be on indefinitely: a camera of model k can be operating at most A_k days without interruption. On the other hand, if a camera starts operating, it must be operating at least 2 days consecutively.
There are N crossings, which are tagged $1, 2, ..., N$, respectively. For $1 \leq i, j \leq N$, the value M_{ij} is the minimum range that a camera must have if placed at i to cover j (or symmetrically, to cover i if placed at j). By convention, if $M_{ij} \geq 50$ then it is impossible to cover i with a camera placed at j (and vice versa). Moreover, $M_{ii} = 0$ for all $1 \leq i \leq N$. To ensure that cameras are properly hidden, at most one camera can be placed at a given crossing.

Batman needs to find out in which crossings of the city cameras should be placed, and of which model these should be, so that all crossings are always watched. Moreover, he also wants to have a weekly schedule that shows, for each d of the week (where $1 \leq d \leq 7$, being Monday $= 1, \dots$, Sunday$= 7$), which cameras should be on. Unfortunately Wayne Corporation is currently going through a very bad financial situation, and as a consequence costs must be minimized.

2 Formal problem statement

Gotham City wants to place cameras at street crossings to monitor the streets effectively while minimizing costs. The goal is to determine the optimal placement and scheduling of cameras such that all crossings are covered every day of the week, adhering to the constraints of camera models, their ranges, autonomies, and costs.
We are given the following inputs:

- K: number of camera models
- N: number of crossings
- P_k: price of camera model k, for $1 \leq k \leq K$, where $P_k > 0$
- R_k: range of camera model k, for $1 \leq k \leq K$, where $1 \leq R_k \leq 49$
- A_k: autonomy of camera model k, for $1 \leq k \leq K$, where $2 \leq A_k \leq 6$
- C_k: daily cost of camera model k, for $1 \leq k \leq K$, where $C_k > 0$
- M_{ij}: minimum range required to cover crossing j from crossing i, for $1 \leq i, j \leq N$

We consider the auxiliary sets:

- $Nset = \{1, 2, \ldots, N\}$: set of crossings
- $Kset = \{1, 2, \ldots, K\}$: set of camera models
- $D = \{1, 2, \ldots, 7\}$: set of days in a week

And we want to determine:

- The placement of cameras at crossings
- The scheduling of cameras to be on for each day of the week (7 days)

We need to ensure that:

- Each crossing is covered every day of the week
- At most one camera is placed at each crossing
- Cameras adhere to their autonomy constraints
- Cameras are turned on for at least 2 consecutive days if they are activated

Therefore, we need to minimize the total cost, which includes both the purchase cost of the cameras and their operational costs over the week.

3 Integer Linear Programming Model

Variables:

- x_{ik}: binary variable, 1 if a camera of model $k \in Kset$ is placed at crossing $i \in Nset$.
- v_{ikd}: binary variable, 1 if the camera of model $k \in Kset$ at crossing $i \in Nset$ is on on day $d \in D$.

Since our weekly schedule is cyclic, we define:

$$f(t) = ((t-1) \pmod 7) + 1$$

such that Monday=1 comes after Sunday=7.
Constraints:

1. Each crossing must be covered every day: For each crossing $j \in Nset$ and each day $d \in D$, there must exist a crossing $i \in Nset$ and a camera model $k \in Kset$ such that the camera at crossing i covers crossing j on day d. This can be expressed as:

$$R_k \geq M_{ij} \quad \forall j \in Nset \text{ and } d \in D \implies \boxed{\sum_{i=1}^{N} \sum_{k=1}^{K} v_{ikd} \geq 1} \quad \forall j \in Nset, \forall d \in D$$

2. At most one camera can be placed at each crossing:

$$\boxed{\sum_{k=1}^{K} x_{ik} \leq 1} \quad \forall i \in Nset$$

3. Autonomy constraint: For each camera placed at crossing i of model k, the number of consecutive days it is on must not exceed its autonomy A_k:

$$\boxed{\sum_{t=0}^{A_k} v_{ikf(d+t)} \leq A_k} \quad \forall i \in Nset, \forall k \in Kset, \forall d \in D$$

4. Minimum operation constraint: For each camera placed at crossing i of model k, if it is turned on, it must be on for at least 2 consecutive days:

$$v_{ikd} \wedge \neg v_{ikf(d-1)} \implies v_{ikf(d+1)}, \quad \forall i \in Nset, \forall k \in Kset, \forall d \in D.$$

By implication law, this can be rewritten as:

$$\neg(v_{ikd} \wedge \neg v_{ikf(d-1)}) \vee v_{ikf(d+1)} \quad \forall i \in Nset, \forall k \in Kset, \forall d \in D.$$

We know that

If $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m$ are binary variables, then:

$$\neg(a_1 \wedge a_2 \wedge \cdots \wedge a_n) \vee (b_1 \wedge b_2 \wedge \cdots \wedge b_m) \Leftrightarrow \sum_{i=1}^{n}(1-a_i) + \sum_{j=1}^{m} b_j \geq 1$$

Thus, we can express the minimum operation constraint as:

$$(1 - v_{ikd}) + v_{ikf(d-1)} + v_{ikf(d+1)} \geq 1, \quad \forall i \in Nset, \forall k \in Kset, \forall d \in \{1, \ldots, 7\}$$

And simplifying,

$$\boxed{-v_{ikd} + v_{ikf(d-1)} + v_{ikf(d+1)} \geq 0} \quad \forall i \in Nset, \forall k \in Kset, \forall d \in \{1, \ldots, 7\}$$

5. Scheduling constraint only for placed cameras: A camera can only be turned on if it has been placed at the crossing:

$$\boxed{v_{ikd} \leq x_{ik}} \quad \forall i \in Nset, \forall k \in Kset, \forall d \in D$$

Objective function: We want to minimize the total cost, which includes the purchase cost of the cameras and the operational cost over the week:

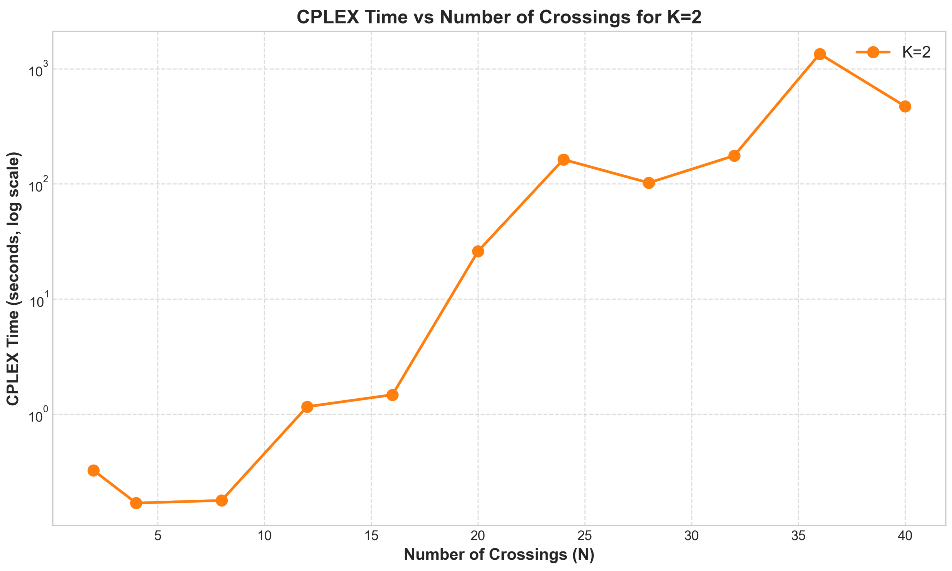
- Purchase cost:
$$\sum_{i=1}^{N}\sum_{k=1}^{K} P_k x_{ik}$$
- Operational cost:
$$\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{d=1}^{7} C_k v_{ikd}$$

Therefore, the overall objective function can be expressed as:

$$\boxed{\text{Minimize } Z = \sum_{i=1}^{N}\sum_{k=1}^{K} P_k x_{ik} + \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{d=1}^{7} C_k v_{ikd}}$$

3.1 Results of the implementation

Since we consider the size of a instance to be $N \cdot K$, and we want to see the behavior of CPLEX when increasing the size of the instance, we have generated different instances with $K = 2, 4, 8$ camera models and varying the number of crossings N from 4 to 40.



This graphic shows that the time increases with N, indicating that larger instances require more computational effort.
Moreover, we can see that for $K = 2$, CPLEX is able to solve instances up to $N = 40$ within a reasonable time frame. (The highest time is for $N = 36$, which is around 1300 seconds (around 20 minutes)).
Also, remark that we have used the same P_k, C_k, R_k, A_k for all the instances, changing only the distance matrix M, which has been randomly generated for each instance. This has been done to have a fair comparison between the different instances.

4 Meta Heuristics

4.1 Greedy algorithm

Consider the full set of elements that must be covered,

$$M = \{(i,d) \mid i \in Nset,\ d \in D\},$$

i.e., each crossing must be covered on each day of the week.
Let $R \subseteq M$ denote the subset of elements already covered at a given iteration of the algorithm.

A candidate is defined as a triple

$$p = (i, k, \text{pattern})$$

where:

- $i \in N$ is the crossing where the camera is installed,
- k is the selected camera model,
- pattern $\in \{0,1\}^7$ is a feasible weekly ON/OFF pattern satisfying the autonomy constraints of model k.

Each candidate $p = (i, k, \text{pattern})$ induces a coverage set

$$\text{cov}(p) = \{(j,d) \in M \mid M_{ij} \le R_k \text{ and } \text{pattern}[d] = 1\}.$$

Each candidate p has an associated cost

$$c(p) = P_k + C_k \cdot \#\{d \in D : \text{camera is ON on day } d\},$$

which includes both the purchase cost and the weekly operating cost.

The greedy evaluation of a candidate p is defined as

$$\boxed{q(p,R) = \frac{c(p)}{|\text{cov}(p) \cap (M \setminus R)|}}$$

where $c(p)$ is the total cost of p and $R \subseteq M$ is the set of elements already covered.

At each iteration, the greedy algorithm chooses

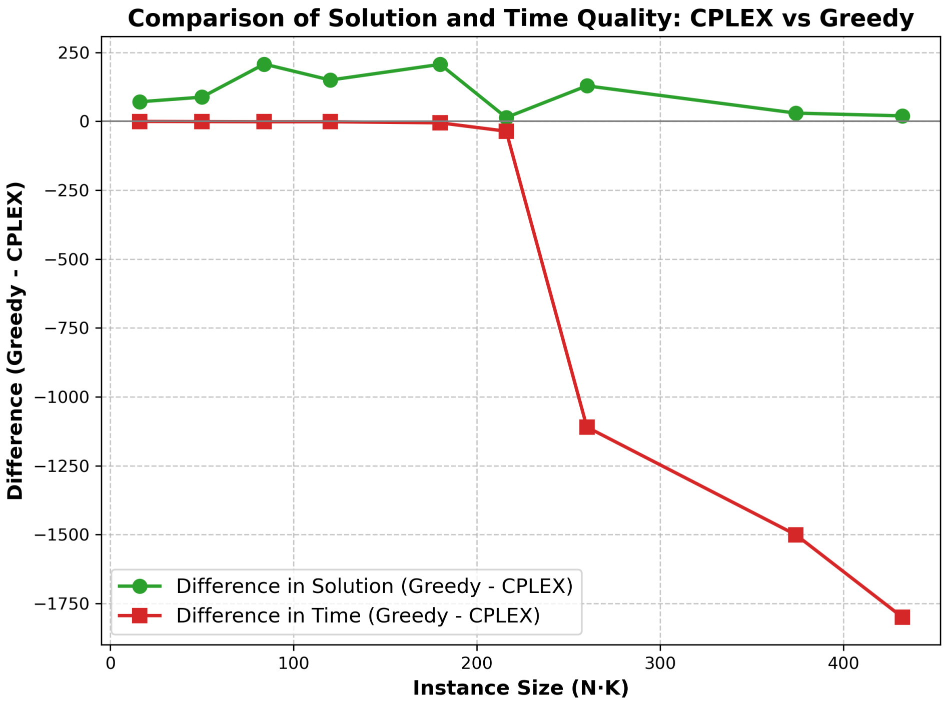
$$p^\star \;=\; \arg\min_{p}\; q(p,R),$$

breaking ties arbitrarily.
Now, the greedy constructive algorithm can be described as follows:

Algorithm 1 Greedy Constructive Algorithm

```
**Input:** crossings $N$, models $K$, matrix $M_{ij}$, ranges $R_k$, autonomies $A_k$, prices $P_k$, consumptions $C_k$
**Output:** selected cameras and weekly ON/OFF patterns covering all crossings

**Step 1: Generate valid weekly patterns**
**for** $k = 1$ to $K$ **do**
    $\mathcal{P}_k \leftarrow \{pattern \in \{0,1\}^7 :$ all ON-runs satisfy $2 \le \ell \le A_k$ (circular)$\}$
**end for**

**Step 2: Build candidate covering sets**
$\mathcal{C} \leftarrow \emptyset$
**for** $i = 1$ to $N$ **do**
    **for** $k = 1$ to $K$ **do**
        **for all** $pattern \in \mathcal{P}_k$ **do**
            $covered \leftarrow \{(j,d) : pattern[d] = 1,\ M_{ij} \le R_k\}$
            **if** $covered \neq \emptyset$ **then**
                $cost \leftarrow P_k + C_k \cdot \sum_d pattern[d]$
                Add candidate $(i, k, pattern, covered, cost)$ to $\mathcal{C}$
            **end if**
        **end for**
    **end for**
**end for**

**Step 3: Greedy Set Cover**
$U \leftarrow \emptyset$, $S \leftarrow \emptyset$, $R \leftarrow \emptyset$
$M_{univ} \leftarrow \{(j,d) \mid j \in Nset,\ d \in \{1..7\}\}$
**while** $R \neq M_{univ}$ **do**
    $best \leftarrow$ null
    $q_{best} \leftarrow +\infty$
    **for all** $c = (i, k, pattern, covered, cost) \in \mathcal{C}$ **do**
        **if** $i \notin U$ **then**
            $new \leftarrow covered \setminus R$
            **if** $new \neq \emptyset$ **then**
                $q \leftarrow cost/|new|$
                **if** $q < q_{best}$ **then**
                    $q_{best} \leftarrow q$
                    $best \leftarrow c$
                **end if**
            **end if**
        **end if**
    **end for**
    **if** $best =$ null **then**
        **return** INFEASIBLE
    **end if**
    $S \leftarrow S \cup best$
    $R \leftarrow R \cup best.covered$
    $U \leftarrow U \cup \{best.i\}$
    Remove from $\mathcal{C}$ all candidates with same location $best.i$
**end while**

**return** $S$
```

4.1.1 Results


Comparison of Solution and Time Quality: CPLEX vs Greedy
Difference (Greedy - CPLEX)
Instance Size (N·K)
250
0
−250
−500
−750
−1000
−1250
−1500
−1750
0
100
200
300
400
Difference in Solution (Greedy - CPLEX)
Difference in Time (Greedy - CPLEX)

4.2 Greedy + Local Search algorithm

Algorithm 2 Greedy Constructive + Local Search (First improvement)

```
$S \leftarrow$ GreedyConstructiveAlgorithm()
$currentCost \leftarrow q(\langle i, k\rangle, S)$
$improved \leftarrow$ TRUE
while $improved =$ TRUE do
   $improved \leftarrow$ FALSE
   for each camera $c_{in} \in S$ do
      for each crossing $j \in Nset$ without a camera do
         for each model $k \in Kset$ do
            $S_{neighbor} \leftarrow (S \setminus \{c_{in}\}) \cup \{(j, k)\}$
            if IsAllCovered($S_{neighbor}, N$) is FALSE then
               continue
            end if
            $cost \leftarrow q(\langle i, k\rangle, S)$
            if $cost < currentCost$ then
               $S \leftarrow S_{neighbor}$
               $currentCost \leftarrow cost$
               $improved \leftarrow$ TRUE
               break
            end if
         end for
         if $improved =$ TRUE then
            break
         end if
      end for
      if $improved =$ TRUE then
         break
      end if
   end for
end while
return $S$
```

4.3 GRASP algorithm

Algorithm 3 GRASP Algorithm

```
 1: $COVERED \leftarrow \emptyset$
 2: $S \leftarrow \emptyset$
 3: while $|COVERED| \neq N$ do
 4:    $randomOption \leftarrow$ NULL
 5:    $C \leftarrow \emptyset$
 6:    $c_{min} \leftarrow +\infty$
 7:    $c_{max} \leftarrow -\infty$
 8:    for each crossing $i \in Nset$ without a camera do
 9:       for each model $k \in Kset$ do
10:          $new \leftarrow$ NewCovered$(i, k, S)$
11:          if $new = \emptyset$ then
12:             continue
13:          end if
14:          $cost \leftarrow q(\langle i, k \rangle, S)$
15:          Add $(i, k, cost)$ to $C$
16:          if $cost < c_{min}$ then
17:             $c_{min} \leftarrow cost$
18:          end if
19:          if $cost > c_{max}$ then
20:             $c_{max} \leftarrow cost$
21:          end if
22:       end for
23:    end for
24:    $RCL \leftarrow \{c \in Candidates \mid c.cost \leq c_{min} + \alpha \cdot (c_{max} - c_{min})\}$
25:    $randomOption \leftarrow randomSelect(RCL)$
26:    Add $randomOption$ to $S$
27:    Update $COVERED$
28: end while
29: Build a feasible weekly schedule respecting autonomy constraints
30: return $S$
```