COURSE PROJECT

Algorithmic Methods for Mathematical Models

Alba Soldevilla & Thomas Aubertier

Universitat Politècnica de Catalunya

Table of contents

Problem statement & ILP model

Problem Statement

Given

- A set of street crossings $i = 1, \ldots, N$.
- Camera models $k = 1, \ldots, K$ with:
  - price P_k, range R_k, autonomy A_k, daily cost E_k.
- Coverage matrix M_{ij}.
- Weekly schedule (7 days).

Goal: Help Batman **minimize the total cost** while covering all crossings during the seven days of the week.

Example (simplified)

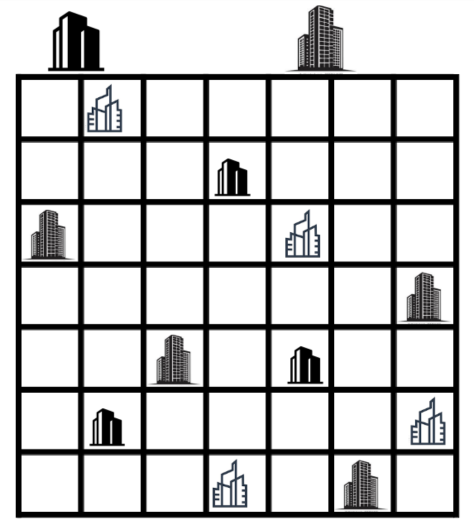


Figure 1: Example with $K = 5$, $N = 64$, blank

Example (simplified)

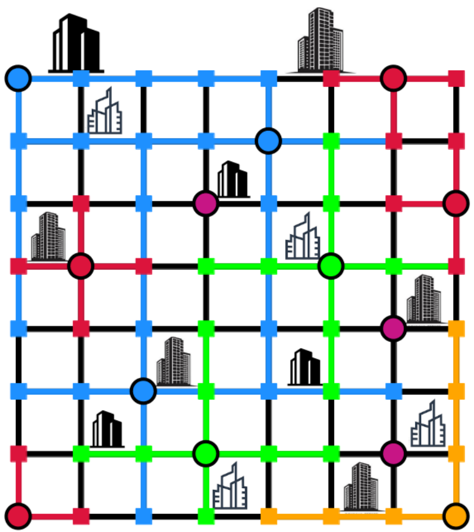


Figure 2: Example with $K = 5, N = 64$, complete

ILP Model

Decision variables

- Installation variable:

$$x_{ik} = \begin{cases} 1, & \text{if we install a camera } k \text{ at } i, \\ 0, & \text{otherwise.} \end{cases}$$

- Operation variable:

$$y_{ikd} = \begin{cases} 1, & \text{if camera } (i, k) \text{ is ON on day } d, \\ 0, & \text{otherwise.} \end{cases}$$

Objective function

$$z = \min \left(\sum_{i=1}^{N} \sum_{k=1}^{K} P_k \, x_{ik} + \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{d=1}^{7} E_k \, y_{ikd} \right)$$

Constraints

- Placement

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

- Coverage

$$\sum_{\substack{i=1..N \\ k:M_{ij} \leq R_k}} y_{ikd} \geq 1 \quad \forall j, d$$

- Consistency

$$y_{ikd} \leq x_{ik} \quad \forall i, k, d$$

- Autonomy

$$\sum_{h=0}^{A_k} y_{ik,\ ((d+h-1) \bmod 7)+1} \ \leq \ A_k$$

$$y_{ikd} = 1 \Rightarrow y_{ik,d+1} = 1$$

Heuristics

Greedy algorithm

Algorithm 1 Greedy constructive heuristic

Require: (K, P, R, A, E, N, M)

$C \leftarrow \textsc{InitCandidates}()$
$S \leftarrow \emptyset$
$covered \leftarrow \emptyset$
while $\neg\textsc{IsSolution}(covered, N)$ **do**
    $(c^\star, q^\star) \leftarrow \textsc{EvaluateQuality}(C, covered)$
    if $c^\star =$ **null then break**
    end if
    Add $c^\star$ to S
    Update $covered$
    $C \leftarrow \textsc{Feasibility}(C, c^\star)$
end while
return S

Quality function

Trade-off between **gain** (newly covered crossings) and **cost**.

$$q(c) = \frac{\left|covers(c) \setminus covered\right|}{P_{k(c)} + E_{k(c)} \cdot |days(c)|}$$

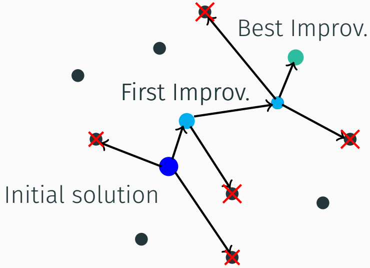
Tie-breaking rule: first candidate in the list.

Local Search (LS)

Algorithm 2 LocalSearch (simplified)

Require: S_{initial} from Greedy and policy FI/BI

$S \leftarrow S_{\text{initial}}$
$S_{\text{best}} \leftarrow S$
while there exist neighbors to evaluate **do**
  Evaluate new neighbor S_{new}
  if S_{new} is better than S **then**
    if policy = FI **then**
      $S \leftarrow S_{\text{new}}$
      break
    else ▷ Best Improvement
      if S_{new} is better than S_{best} **then**
        $S_{\text{best}} \leftarrow S_{\text{new}}$
      end if
    end if
  end if
end while
if policy = BI **then**
  $S \leftarrow S_{\text{best}}$
end if
return S ▷ Local optimum



GRASP

Algorithm 3 GRASP constructive phase (simplified)

Require: $(K, P, R, A, E, N, M, \alpha)$

$C \leftarrow$ InitCandidates()
$S \leftarrow \emptyset, \quad covered \leftarrow \emptyset,$
while $\neg$IsSolution($covered, N$) **do**
    $(q_{\max}, q_{\min}) \leftarrow$ EvaluateQuality($C, covered$)
    RCL $\leftarrow$ BuildRCL($C, covered, P, E, q_{\max}, q_{\min}, \alpha$)
    $c^{\text{rand}} \leftarrow$ RandomChoice(RCL)
    $S \leftarrow S \cup \{c^{\text{rand}}\}$
    Update $covered$
    $C \leftarrow$ Feasibility(C, c^{rand})
end while
return S

Restricted Candidate List construction (RCL):

$$\text{RCL} = \{\, c \in C \mid q(c) \geq q_{\max} - \alpha(q_{\max} - q_{\min}) \,\}.$$

Tuning and instances

GRASP tuning

Instance	N	K	$N \cdot K$
small	4	2	8
medium	10	5	50
large	20	10	200

Table 1: Set of generated instances used for performance evaluation.

GRASP tuning: Maximum Iterations (I)

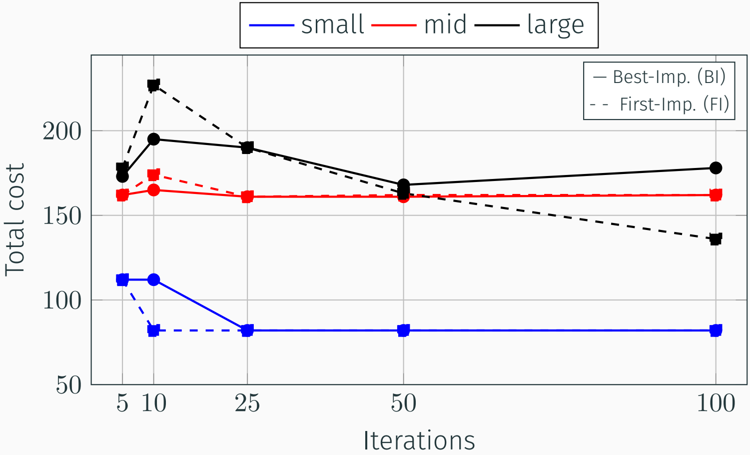


Figure 3: Total cost as a function of the maximum number of iterations ($\alpha = 0.5$).

GRASP tuning: Maximum Iterations (II)

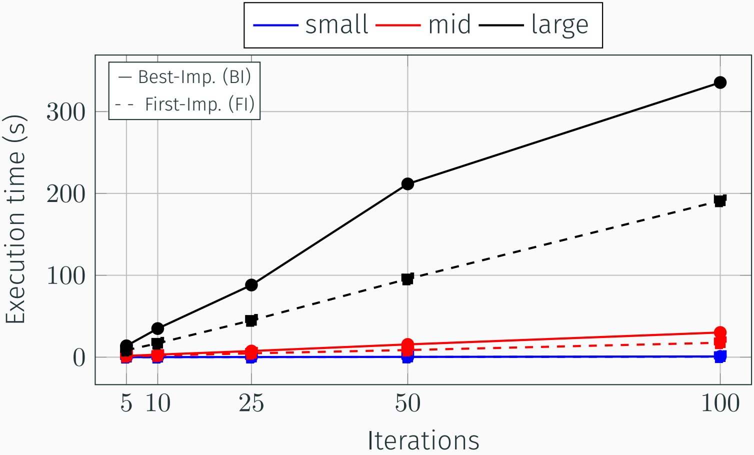


Figure 4: Execution time as a function of the maximum number of iterations ($\alpha = 0.5$).

GRASP tuning: Alpha (α)

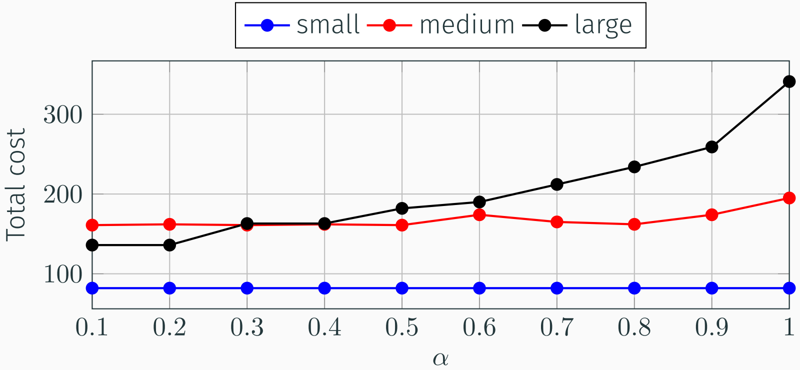


Figure 5: Effect of α on total cost (50 iterations, First Improvement policy).

Instance Generation (list)

Instance	N	K	$N \cdot K$
1	5	5	25
2	7	7	49
3	10	10	100
4	20	20	400
5	30	30	900
6	50	50	2500
7	75	75	5625
8	2	20	40
10	10	20	200
11	40	2	80
12	40	20	800

Table 2: Set of generated instances used for performance evaluation.

Experimental results

Methods used

- ILP (CPLEX)
- Greedy
  - Greedy constructive (simple)
  - Greedy constructive + LS with First improvement policy
  - Greedy constructive + LS with Best improvement policy
- GRASP
  - GRASP constructive (simple)
  - GRASP constructive + LS with First improvement policy
  - GRASP constructive + LS with Best improvement policy

Procedure : find the best setting for Greedy / GRASP, and compare them with ILP.

Results: Greedy

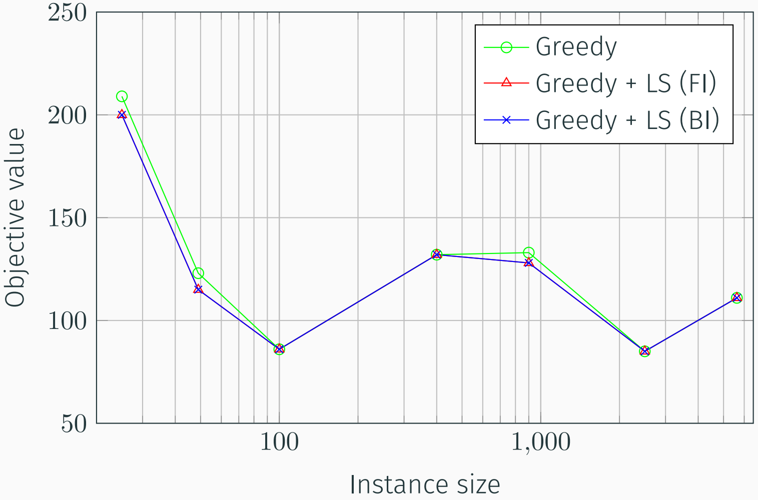


Figure 6: Objective value as a function of the instance size for Greedy.

Results: Greedy

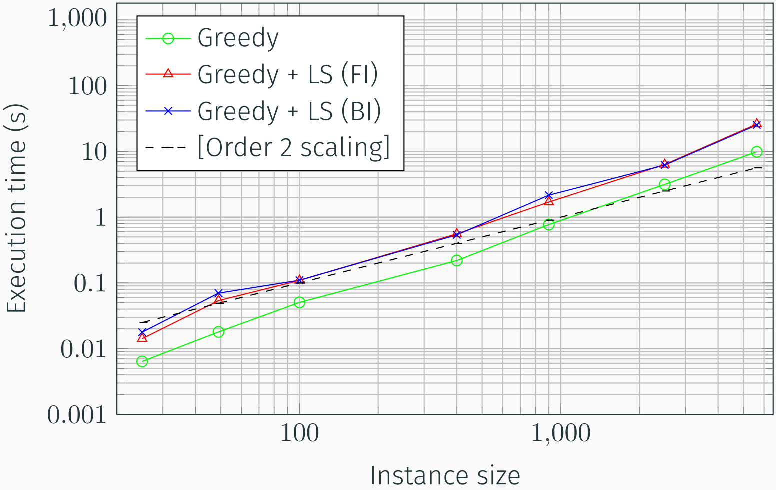


Figure 7: Execution time as a function of the instance size for Greedy.

Results: Greedy

- **Objective function:** FI = BI for all instances; simple Greedy stays close (especially for large instances).
- **Execution time** : simple Greedy is significantly faster; FI little faster than BI.

Choice: Computation is fast in all cases. Accuracy should be prioritised. Between FI and BI, FI is faster with similar performance.

$\Rightarrow$ Greedy + LS (FI).

Results: GRASP

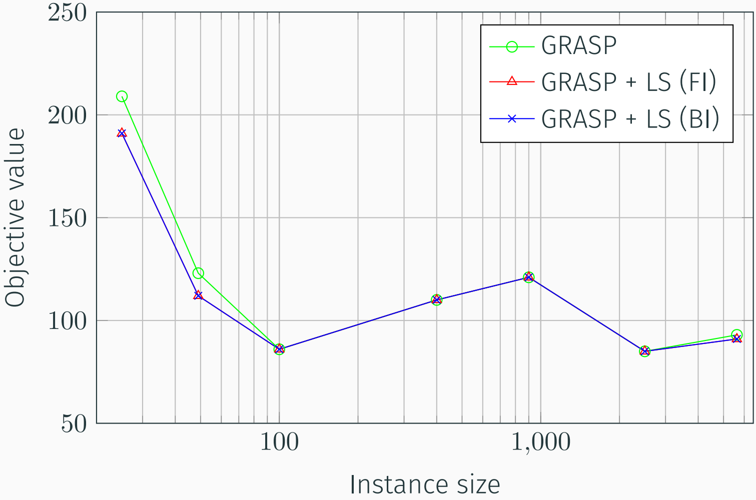


Figure 8: Objective value as a function of the instance size for GRASP.

Results: GRASP

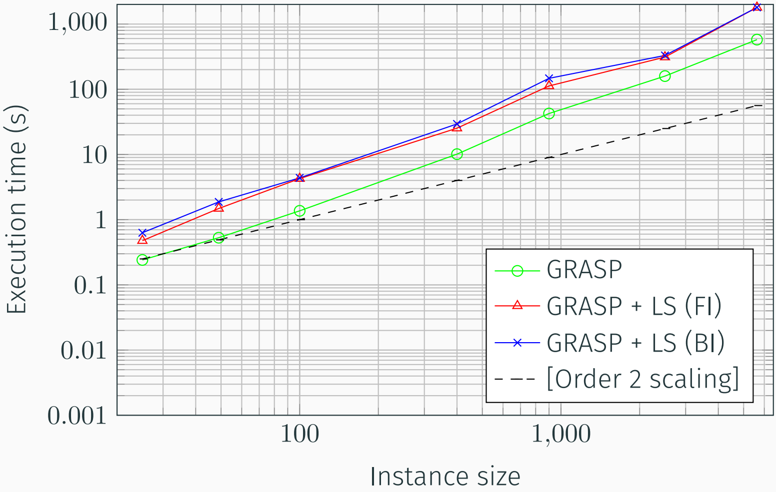


Figure 9: Execution time as a function of the instance size for GRASP.

Results: GRASP

- **Objective function** : FI = BI for all instances, simple GRASP stays close (especially for big instances)
- **Execution time** : simple GRASP is significantly faster, FI is close second above BI.

Choice: As with the Greedy case, FI outperforms BI. However, here the computation time is no longer negligible: simple GRASP is 2–3$\times$ faster, with only a small loss in accuracy.

$\Rightarrow$ GRASP + LS (FI) (simple GRASP is also a valid option).

Final comparison

- ILP (CPLEX)
- Greedy
  - ~~Greedy constructive (simple)~~
  - ~~Greedy constructive + Local search with first improvement policy~~
  - Greedy constructive + Local search with best improvement policy
- GRASP
  - ~~GRASP constructive (simple)~~
  - ~~GRASP constructive + Local search with first improvement policy~~
  - GRASP constructive + Local search with best improvement policy

Final comparison

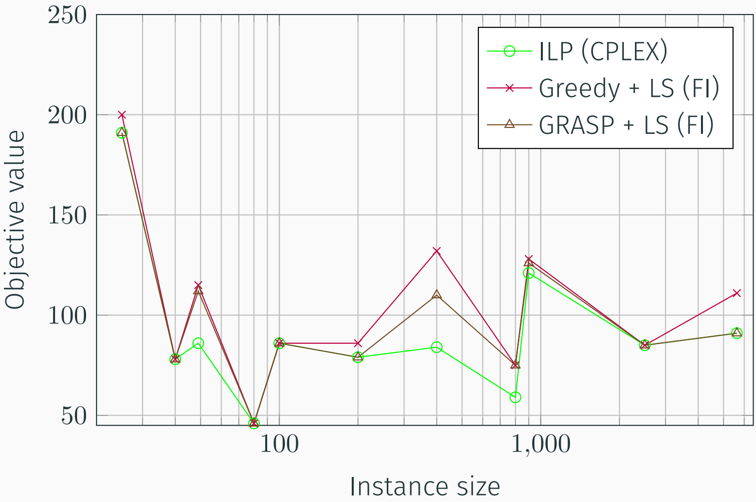


Figure 10: Objective value as a function of the instance size.

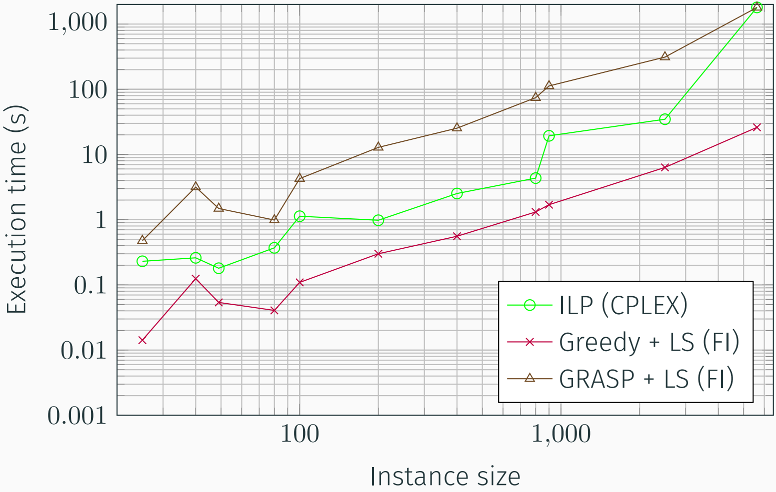


Figure 11: Execution time as a function of the instance size

Final comparison

Rankings :

- **Objective function** : ILP > GRASP > Greedy
- **Execution time** : Greedy > ILP > GRASP

Choice : ILP outperforms GRASP in all aspects, therefore GRASP is not viable.

Greedy is 5–10$\times$ faster than ILP, scales stably, and maintains acceptable accuracy even on large instances.

ILP execution time increases sharply beyond size 50×50 and is computationally expensive (high CPU usage).

Final choice

⇒ Greedy + LS (FI) : Suitable when computing resources are limited or when dealing with very large instances (particularly with many crossings).

⇒ ILP : If computing resources are not an issue and instances are moderate; best for long-term planning where operational savings justify longer solving times.

Thank you for listening !

Figure 12: Batman on his way to place all the cameras