An Iterated Local Search to the Perfect Awareness Problem

16th Metaheuristics International Conference

Cristian Pérez-Corral (cpercor@upv.es)

Isaac Lozano-Osorio (isaac.lozano@urjc.es)

Anna Martínez-Gavara (gavara@uv.es)

Angel A. Juan (ajuanp@upv.es)

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universidad Rey Juan Carlos

VNIVERSITAT DE VALÈNCIA

MINISTERIO DE CIENCIA E INNOVACIÓN

AGENCIA ESTATAL DE INVESTIGACIÓN

Outline

Introduction

- The **Perfect Awareness Problem** (PAP) is a $\mathcal{NP}$**-hard** problem.
- It belongs to the family of problems about **Social Influence Maximization** problems.
- Real-world **applications**:
  - Marketing.
  - Fake news.
  - **Emergency alerts**.

Introduction

Problem definition

Let $G = (V, E)$ **undirected** and **unweighted** graph.

The *neighborhood* of a node v and its *degree* are defined as

$$N(v) = \{u \mid \{u, v\} \in E\}, \qquad \deg(v) = |N(v)|.$$

Each node is assigned a *threshold* $t : V \to \mathbb{Z}^+$; we adopt the **majority threshold** $t(v) = \lceil \deg(v)/2 \rceil$.

Introduction

Problem definition

At each **round** $\tau \geq 1$, letting S_τ and A_τ denote the **aware** and **spreader** sets, a node v transitions as follows:

$$v \in A_\tau \iff v \in S_0 \text{ or } |N(v) \cap S_{\tau-1}| \geq 1,$$
$$v \in S_\tau \iff v \in S_0 \text{ or } |N(v) \cap S_{\tau-1}| \geq t(v).$$

The process terminates when $S_\tau = S_{\tau-1}$. If $A_\tau = V$, the set S_0 is called a **perfect seed**.

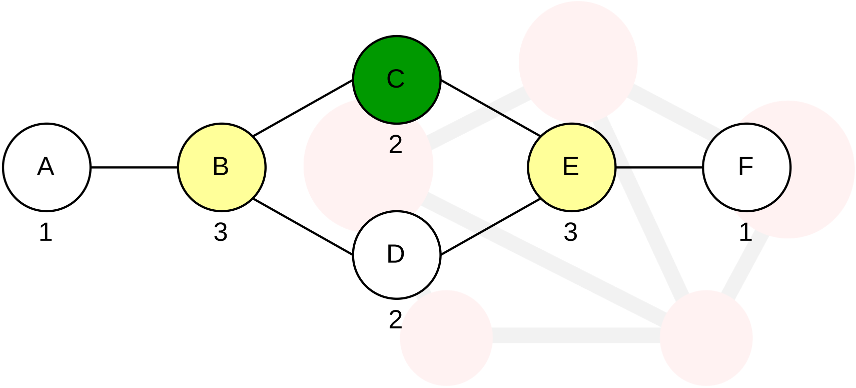
The PAP seeks:

$$\min_{S_0 \subseteq V} |S_0| \quad \text{s.t.} \quad A_\tau = V.$$

Given S_0, this condition can be verified via a **BFS** over G in $\mathcal{O}(|V| + |E|)$.

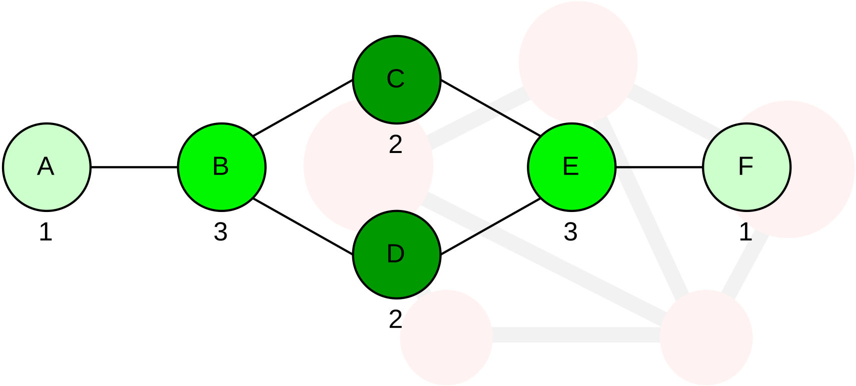
Introduction

Example



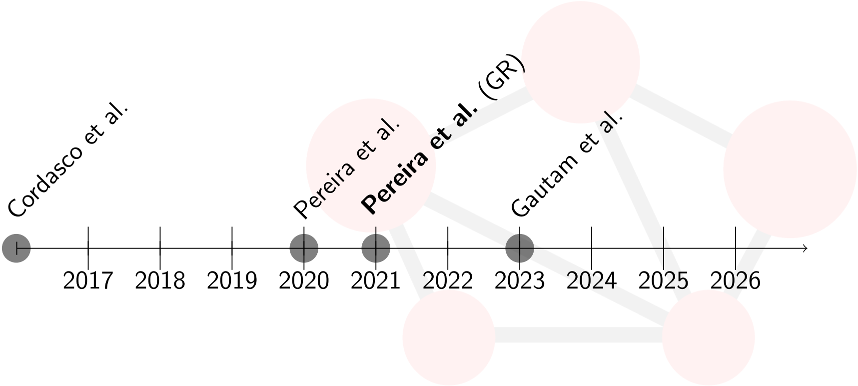
Introduction

Example



Introduction

Literature



Algorithm proposal

Iterated Local Search (ILS)

- Lourenço, H. R., Martin, O. C., & Stützle, T. (2003).
- The key features of our proposal are:
  - **ILS metaheuristic** for the PAP.
  - Reservoir-based local search using **Weighted Random Sampling with Reservoir (WRSR)** as biased sampling mechanism.
  - Computational experiments on **real-world social networks** (R1).

Algorithm proposal

Iterated Local Search (ILS)

Algorithm 1 ILS(Destruct, Reconstruct)

```
σ_0 ← RandomConstructive( )
σ* ← LocalSearch(σ_0)
σ_best ← σ*
while not StopCondition do
    σ' ← Perturb(σ*, Destruct, Reconstruct)
    σ* ← LocalSearch(σ')
    if AcceptanceCriterion(σ*, σ_best) then
        σ_best ← σ*
    end if
end while
return σ_best
```

Algorithm proposal

Constructive

Two approaches are used both for the initial solution and reconstruction during perturbation:

- **RND:** nodes are selected **uniformly at random**.
- **GRD:** nodes are selected **greedily** according to a contribution metric.

Algorithm proposal

Constructive

The greedy criterion relies on the following definitions:

Definition

Given an instance I, a partial solution $\sigma \subseteq V$, and the aware set $A(\sigma)$, the **aware neighbors** of v are:

$$A_v(\sigma) = A(\sigma) \cap N(v).$$

Definition

The **contribution** of node v with respect to σ is:

$$g(v) = \frac{\deg(v) - |A_v(\sigma)|}{\deg(v)},$$

i.e., the proportion of neighbors of v that are not yet aware.

Algorithm proposal

Perturbation Phase

The perturbation operator introduces diversification by modifying the current solution S_0:

1. **Destruction:** a fraction $\rho \in [0, 0.5]$ of the nodes in S_0 are randomly removed.
2. **Reconstruction:** the partial solution is rebuilt using the constructive phase (RND or GRD) until feasibility is recovered, i.e., $A_\tau = V$.

The parameter ρ controls the balance between **intensification** (small ρ, minor changes) and **diversification** (large ρ, major restructuring).

Algorithm proposal

Weighted Random Sampling with Reservoir (WRSR)

A **first improvement** strategy: the first candidate $u \notin S_0$ in WRSR order that finds a feasible $2 \to 1$ swap is immediately accepted, reducing $|S_0|$ by one without exhausting all possibilities.

- Candidates: all non-seed nodes $u \notin S_0$.
- Filter: top-K by degree.
- Reorder: WRSR with weight $w = d^4$.
- Accept: first feasible swap found, restart from updated S_0.

Efraimidis, P.S., Spirakis, P.G.: Weighted random sampling with a reservoir. *Information Processing Letters* 97(5), 181–185 (2006).

Algorithm proposal

Weighted Random Sampling with Reservoir (WRSR)

For each candidate $u_i \notin S_0$:

1. Compute weight $w_i = d_i^4$.
2. Draw $U_i \sim \mathrm{Unif}(0, 1)$ for each candidate.
3. Assign key $k_i = U_i^{1/w_i}$.
4. Sort candidates by k_i descending.

As w_i grows, k_i concentrates near 1:

- $d = 1 \Rightarrow w = 1 \Rightarrow \mathbb{E}[k] = 0.500$, fully random.
- $d = 2 \Rightarrow w = 16 \Rightarrow \mathbb{E}[k] = 0.941$, mild preference.
- $d = 4 \Rightarrow w = 256 \Rightarrow \mathbb{E}[k] = 0.996$, strong preference.
- $d = 8 \Rightarrow w = 4096 \Rightarrow \mathbb{E}[k] = 0.9998$, near-deterministic.

Algorithm proposal

Weighted Random Sampling with Reservoir (WRSR)

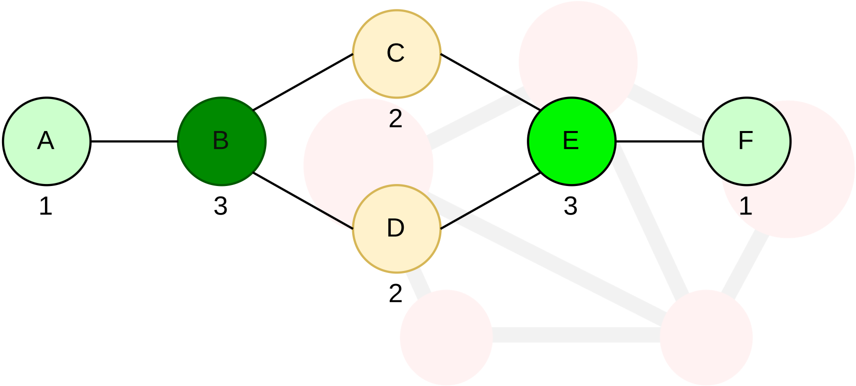
Node	Degree	$w = d^4$	U_i	$k_i = U_i^{1/w}$
B	3	81	0.491	0.943
A	1	1	0.921	0.921
E	3	81	0.210	0.901
F	1	1	0.612	0.612

Table 1: WRSR ranking with $S_0 = \{C, D\}$. Node A ($d = 1$, $w = 1$) draws $U = 0.921$ and since $k = U$ directly, ranks 2nd above E ($d = 3$, $w = 81$) which drew an unlucky $U = 0.210$. Low-degree nodes are never discarded with a fortunate draw can outrank a hub, preventing the search from stagnating on a fixed subset of candidates.

Efraimidis, P.S., Spirakis, P.G.: Weighted random sampling with a reservoir. *Information Processing Letters* 97(5), 181–185 (2006).

Algorithm proposal

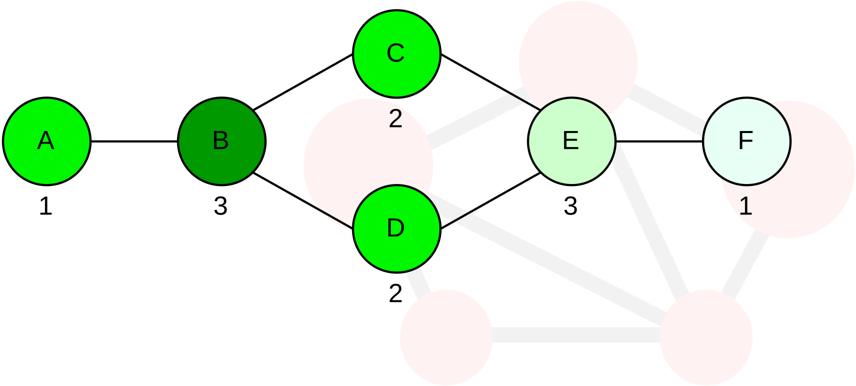
Weighted Random Sampling with Reservoir (WRSR)



Efraimidis, P.S., Spirakis, P.G.: Weighted random sampling with a reservoir. *Information Processing Letters* 97(5), 181–185 (2006).

Algorithm proposal

Weighted Random Sampling with Reservoir (WRSR)



Efraimidis, P.S., Spirakis, P.G.: Weighted random sampling with a reservoir. *Information Processing Letters* 97(5), 181–185 (2006).

Algorithm proposal

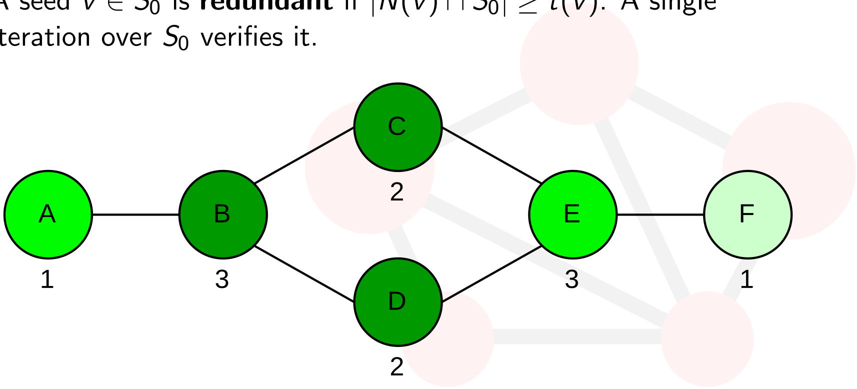
WRSR vs. GRASP Restricted Candidate List

- **RCL (GRASP)**: hard-excludes nodes below threshold $d_{\text{min}} + \alpha(d_{\text{max}} - d_{\text{min}})$. Fast, but permanently discards candidates and treats all survivors as equally likely.
- **WRSR**: no node is permanently excluded. Every $u \notin S_0$ receives a key, so a low-degree node with a high U_i can still be selected. Degree biases the ranking without making it deterministic.

Algorithm proposal

Post-process

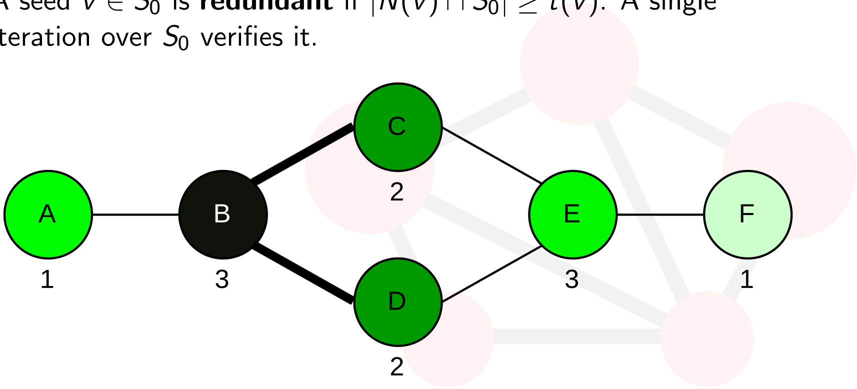
A seed $v \in S_0$ is **redundant** if $|N(v) \cap S_0| \geq t(v)$. A single iteration over S_0 verifies it.



Algorithm proposal

Post-process

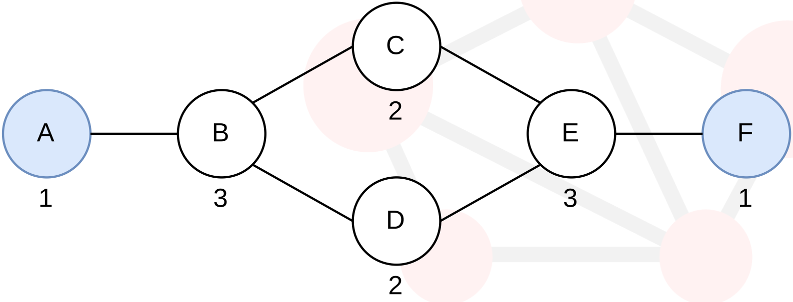
A seed $v \in S_0$ is **redundant** if $|N(v) \cap S_0| \geq t(v)$. A single iteration over S_0 verifies it.



Algorithm proposal

Network pre-processing: Leaf blindness

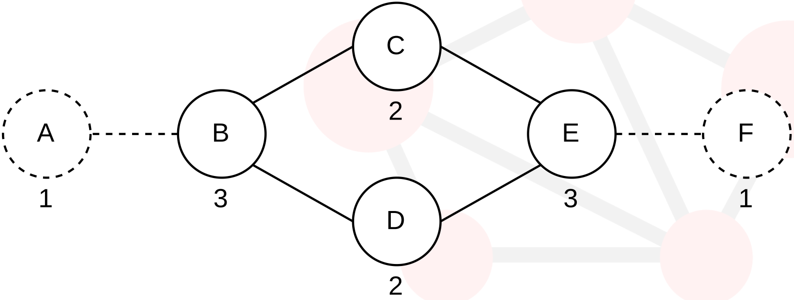
If a **node** has a **single neighbor**, both **collapse into one**, the neighbor absorbs it without losing optimal solutions.



Algorithm proposal

Network pre-processing: Leaf blindness

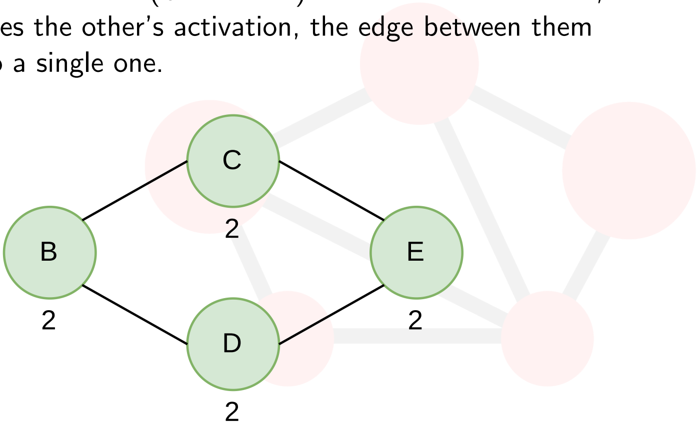
If a **node** has a **single neighbor**, both **collapse into one**, the neighbor absorbs it without losing optimal solutions.



Algorithm proposal

Network pre-processing: Edge reduction

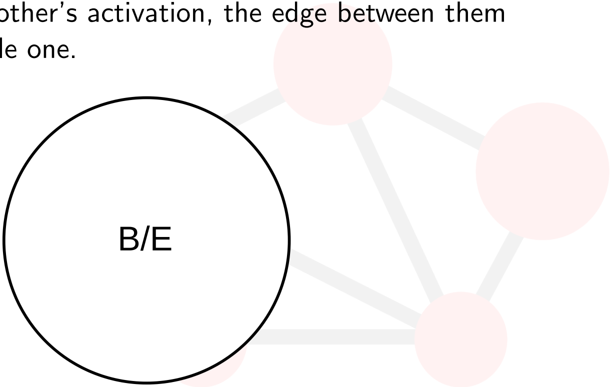
If **two connected nodes** (Union-Find) both have **threshold 1**, each guarantees the other's activation, the edge between them **collapses** into a single one.



Algorithm proposal

Network pre-processing: Edge reduction

If **two connected nodes** (Union-Find) both have **threshold 1**, each guarantees the other's activation, the edge between them **collapses** into a single one.

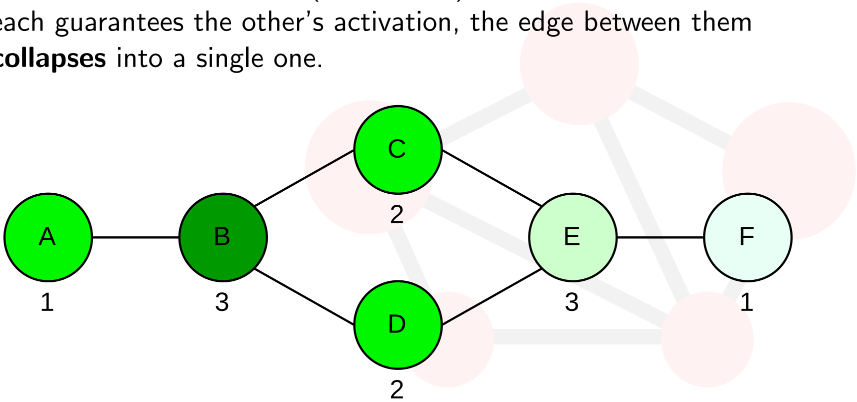


If a **single node** $v \in V$ yields a feasible solution, then $S_0 = \{v\}$ is trivially **optimal**, as no seed set of smaller size exists.

Algorithm proposal

Network pre-processing: Edge reduction

If **two connected nodes** (Union-Find) both have **threshold 1**, each guarantees the other's activation, the edge between them **collapses** into a single one.



Computational results

Experimental environment

- Programming language: **Java 25**.
- Server specifications: Ubuntu Server 20.04 en **AMD EPYC 7282**, **32 cores** y **32 GB** de RAM.
- Metrics:
  - **OF:** average objective function value across all instances.
  - **Dev. (%):** average percentage deviation from the best known value per instance.
  - **#Best:** number of instances for which the algorithm matches the best known solution.
- The **instances** are divided into **2 groups** (Pereira et al.):
  - Literature instances nodes: [10, 1000].
  - Real-world social network nodes: [34, 1.138.499].

Computational results

Preliminary experiments

We tested different algorithm configurations on a subset (20%) of literature instances. Following the experimental setup of previous works in the literature, execution time was limited to **300 seconds** for small instances and **1 hour** for large instances.

Table 2: ILS algorithms with different initial constructions.

Algorithm	**OF**	**Dev. (%)**	**#Best**
ILS(RND, RND)	**10.44**	**0.20**	**144**
ILS(RND, GRD)	10.55	2.67	133

ILS(RND, RND) achieves the **best results** across all metrics.

Computational results

Final results

Table 3: Comparison of algorithms by instance size.

$\|V\|$	Algorithm	OF	Dev. (%)	#Best
[10, 250)	*GR*	3.95	0.95	575
	ILS(RND,RND)	**3.91**	**0.00**	**600**
[250, 500)	*GR*	9.33	0.68	57
	ILS(RND,RND)	**9.27**	**0.37**	**58**
[500, 750)	*GR*	12.02	0.69	**87**
	ILS(RND,RND)	**11.99**	**0.56**	81
[750, 1000]	*GR*	**14.40**	**0.23**	**88**
	ILS(RND,RND)	14.61	1.24	74
[0, 1000]	*GR*	6.32	0.82	807
	ILS(RND,RND)	**6.30**	**0.22**	**813**

Computational results

Final results - Real-world instances (R1)

Table 4: Comparison of ILS and GR on real-world social network instances.

Instance	GR	ILS(RND,RND)
Karate	**3**	**3**
Jazz	**13**	**13**
Facebook	**10**	**10**
Power grid	602	**576**
CA-GrQc	769	**454**
CA-HepTh	1164	**757**
BlogCatalog3	208	**207**
CA-HepPh	1243	**1100**
BuzzNet	**125**	128
YouTube	**38668**	65000

Conclusions and future work

- New best-known solutions for 38 instances.
- All code and results will be publicly available to ensure reproducibility.
- Large-scale instances were added, yielding competitive results and improving 8/10 best-known solutions.

Future work:

- Increasing the computational budget would likely yield further improvements in solution quality.
- Extract structural patterns that could generate better initial solutions.
- Combining metaheuristics with machine learning.

Acknowledgments

This work was funded by:

- **Ministerio de Ciencia e Innovación** con ref. PID2021-126605NB-I00, PID2021-125709OA-C22 and PID2024-156045NB-I00 financiado por MCIN /AEI /10.13039/501100011033 and by ERDF A way of making Europe.
- **CIRMA**: "This work has been supported by Comunidad Autónoma de Madrid, CIRMA-CM Project (TEC-2024/COM-404)".
- **Cátedra ENIA**: Ministerio para la Transformación Digital y de la Función Pública (Ref. TSI-100930-2023-3).

GOBIERNO DE ESPAÑA
MINISTERIO PARA LA TRANSFORMACIÓN DIGITAL Y DE LA FUNCIÓN PÚBLICA

MINISTERIO DE CIENCIA E INNOVACIÓN
AGENCIA ESTATAL DE INVESTIGACIÓN

An Iterated Local Search to the Perfect Awareness Problem

16th Metaheuristics International Conference

Cristian Pérez-Corral (cpercor@upv.es)

Isaac Lozano-Osorio (isaac.lozano@urjc.es)

Anna Martínez-Gavara (gavara@uv.es)

Angel A. Juan (ajuanp@upv.es)

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universidad Rey Juan Carlos

VNIVERSITAT DE VALÈNCIA

MIC 2026

16th Metaheuristics International Conference

MINISTERIO DE CIENCIA E INNOVACIÓN

AGENCIA ESTATAL DE INVESTIGACIÓN