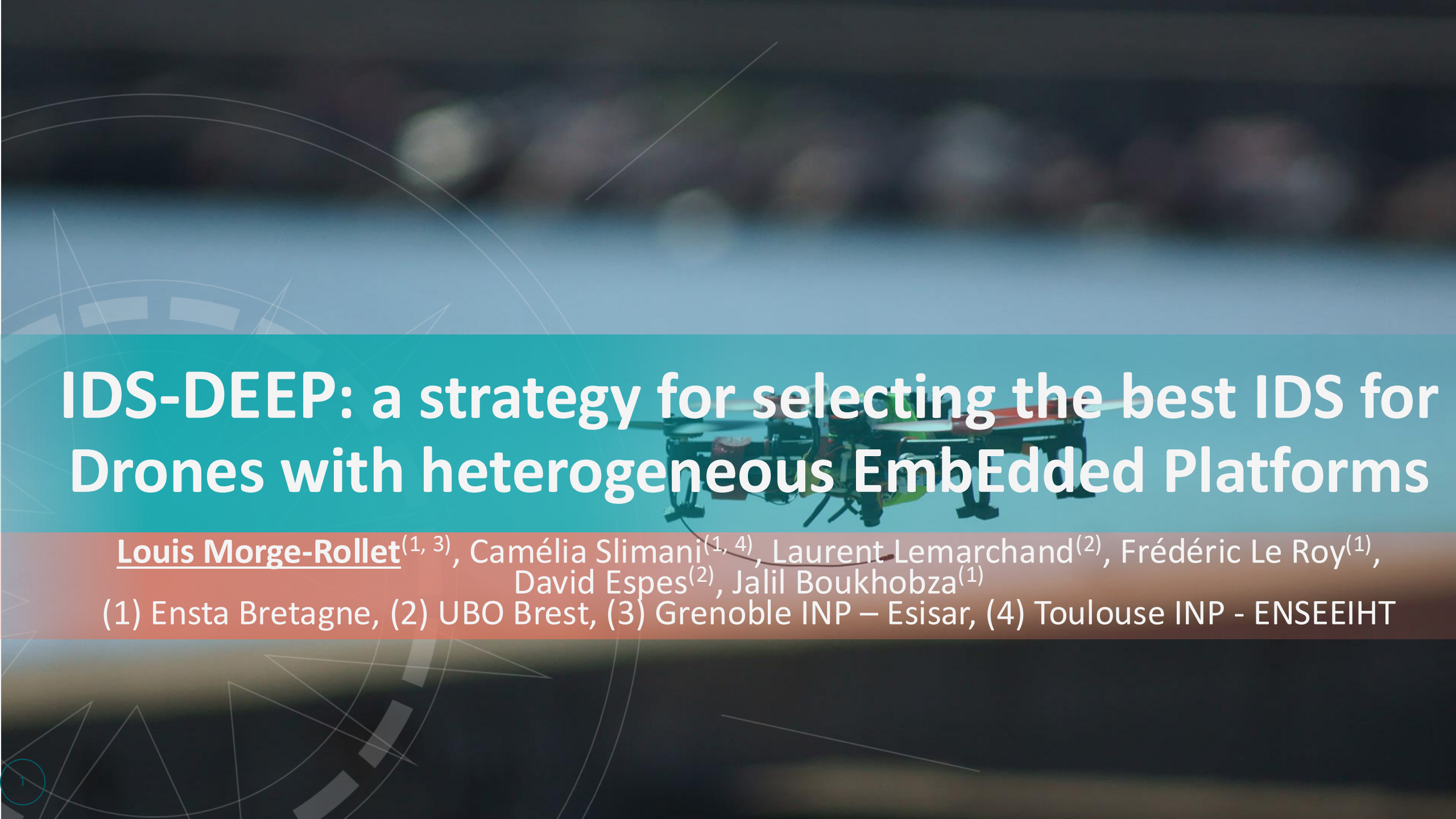


IDS-DEEP: a strategy for selecting the best IDS for Drones with heterogeneous EmbEdded Platforms

Louis Morge-Rollet[1, 3], Camélia Slimani[1, 4], Laurent Lemarchand[2], Frédéric Le Roy[1], David Espes[2], Jalil Boukhobza[1]

(1) Ensta Bretagne, (2) UBO Brest, (3) Grenoble INP – Esisar, (4) Toulouse INP - ENSEEIHT

Outline

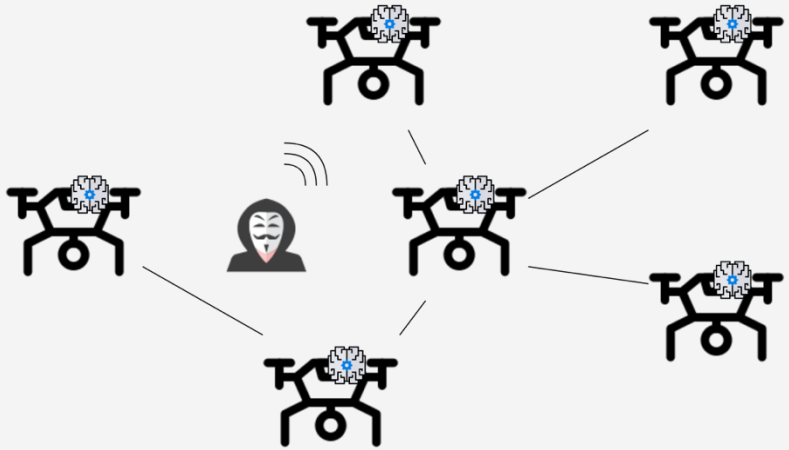
- Context and motivation: DISPEED project
- Contribution: IDS-DEEP
- Experimental case study: Experiments
- Conclusion and perspectives

DISPEED project

Context and motivation

DISPEED project
Context

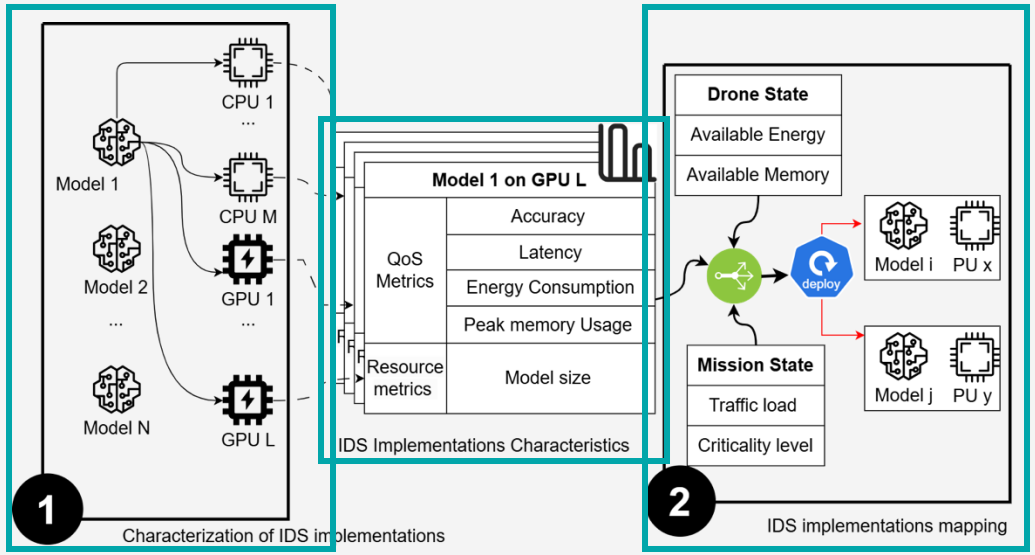
- Swarms of drones advent
  - Swarms of drones are gain more autonomy and efficiency during their mission [1].
  - Security threats and low energy can disrupt mission progression.
- (Network) Intrusion Detection Systems (IDS)
  - Software or hardware system that identifies suspicious actions on network traffic [2].
  - Modern IDS rely on Machine Learning techniques that are resource hungry [3].



Drone swarm under attack and IDS defense

DISPEED project
Overview of encompassing project

- Goal of the project: Leverage on the heterogeneity (hardware, memory) of the drones composing a swarm, to deploy Intrusion Detection Systems (IDS)



Model type	CPU	GPU	FPGA
RandomForest	Binary file	TVM file	-
Neural Netwrok	tflite file	ONNX file	Bitstream

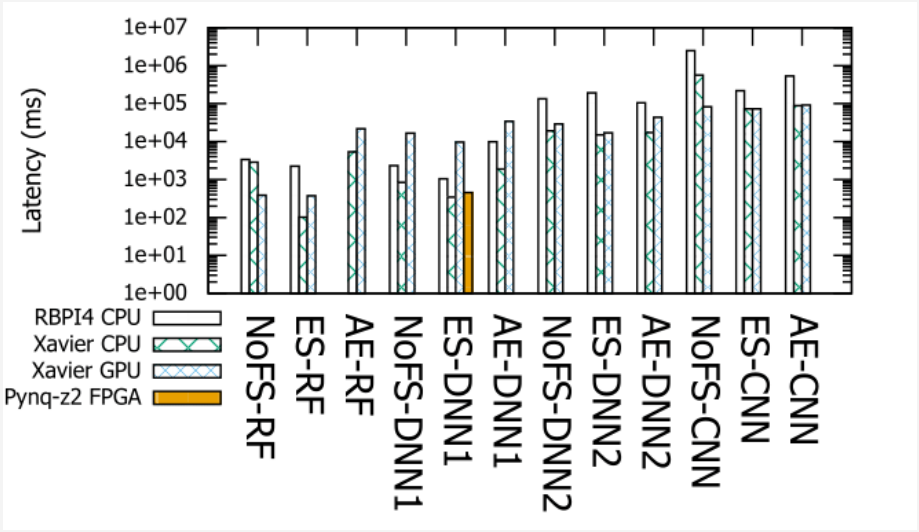
Implementation Types

Implement Model → Characterize Model → Map Model → Selection → Deployement

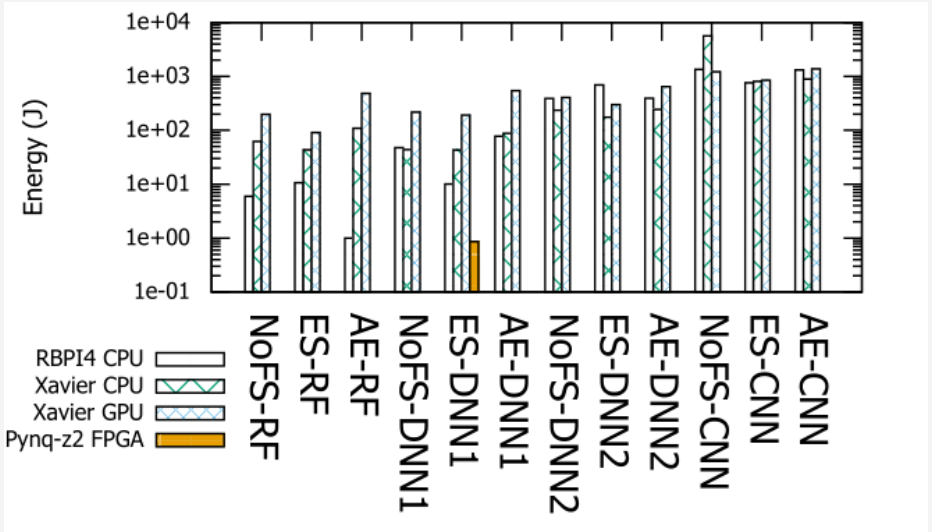
Overview of DISPEED project

DISPEED project
Characterization results

- 36 IDS implementations on Raspberry Pi 4 B, Jetson Xavier, and Pynq-Z2 [4-5]
- Metrics considered:
  - Ressource metrics: Storage size
  - Quality of Service (QoS) metrics: Accuracy, latency, energy consumption, memory peak usage
- Conclusion: There is a great diversity of IDS implementations.



Latency consumption characterization [6]



Energy consumption characterization [6]

IDS-DEEP

Contribution

IDS-DEEP
Research questions

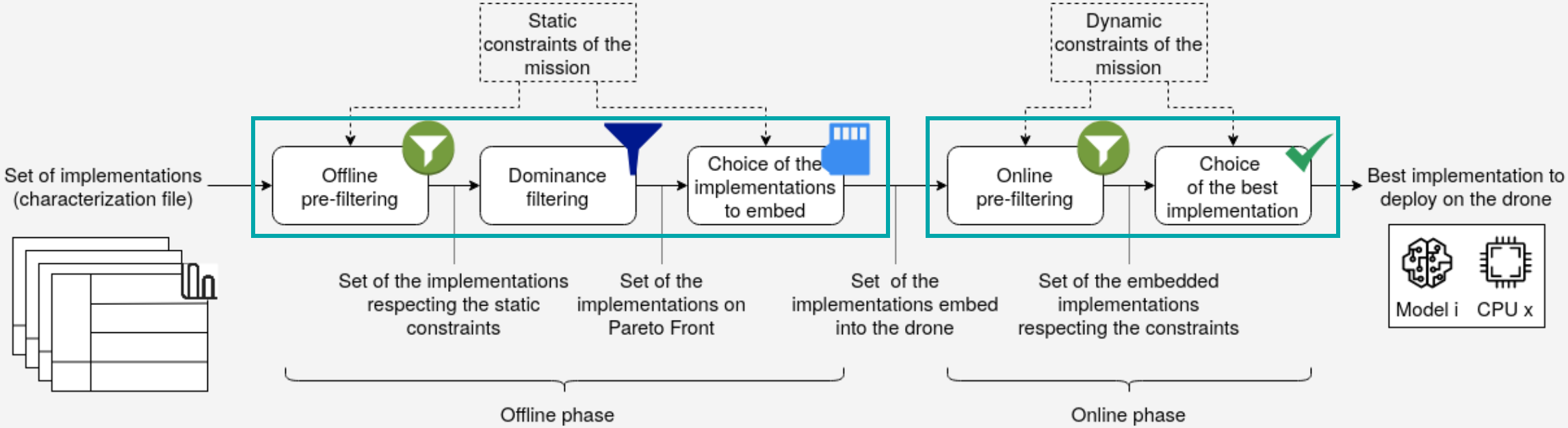
1. How to select a diversity of IDS implementations to embed into a single drone, while respecting a storage size dedicated to it ?
   - Diversity
   - Storage size
2. How to select the best IDS implementation to deploy from this embedded set, depending on the context ?
   - Adaptability
   - Low time cost

IDS-DEEP
Static and dynamic constraints

- Static constraints: Known before the mission
  - Available storage space: Storage space dedicated to IDS implementations
  - Available processing units: Embedded platform present into the drone
- Dynamic constraints: Evolving during the mission
  - Criticality level: Minimum acceptable accuracy
  - Throughput: Maximum acceptable latency
  - Energy consumption: Maximum acceptable energy consumption
  - Memory peak usage: Maximum acceptable memory peak usage

IDS-DEEP Overview

1. Offline phase: Select the IDS implementations to embed, according to the static constraints
2. Online phase: Select the best IDS implementation to deploy, according to the dynamic constraints



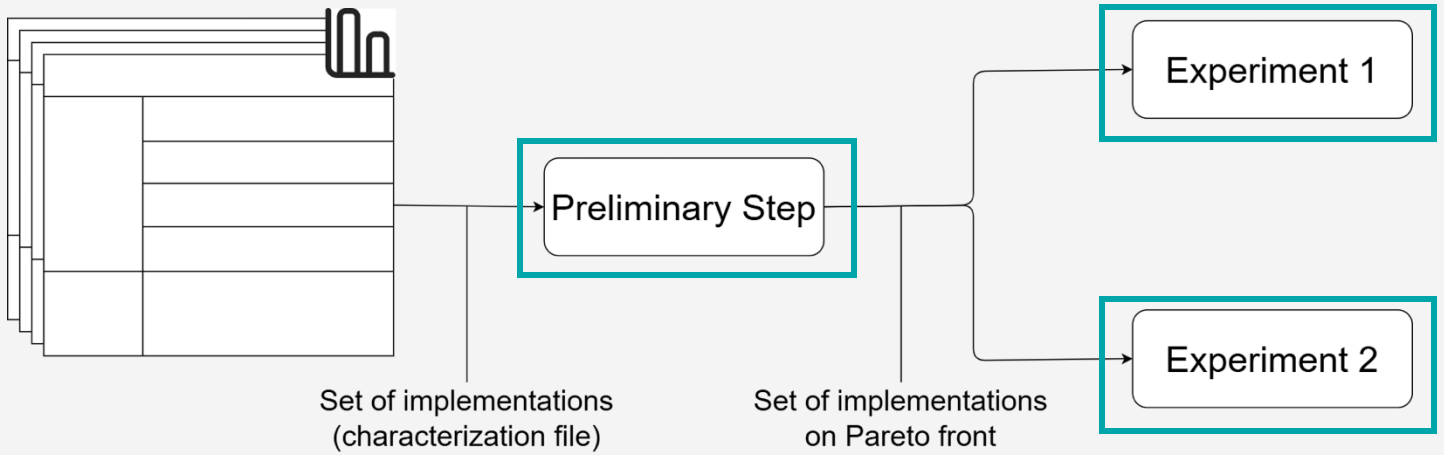
Overview of IDS-DEEP

Experiments

Experimental case study

Experiments Overview

- Input: 36 IDS implementations from [5]
- A priori hypothesis: No static constraints about available platforms
- Experiments:
  - Preliminary step: Pareto Front Computation
  - Experiment 1: IDS implementations to embed
  - Experiment 2: Online phase evaluation



Experimental methodology

Experiments
Preliminary Step: Pareto Front Computation

AGENCE INNOVATION DÉFENSE

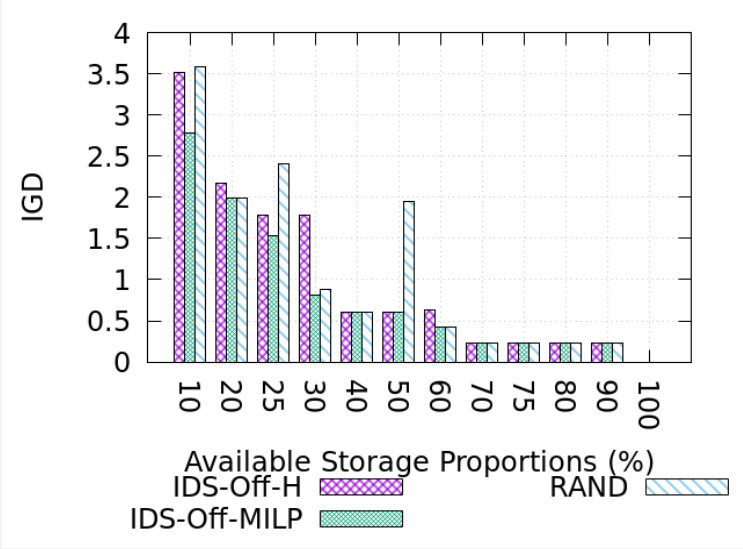
- Goal: Evaluate the dominance filtering step (offline phase)
- Results: 10 IDS implementations are on the Pareto front (~27 %)

Platform	Model	Throughput (flow/s)	Energy (µJ)	Memory Peak (MB)	Accuracy (%)
Raspberry Pi 4	NoFS-RF	23675.64	74.875	0.033	81.59
	ES-RF	35714.29	133	0.011	77.28
	ES-DNN1	76263.11	125.375	0.066	70.34
Nvidia Xavier GPU	NoFS-RF	201005.03	2462.5	1.500	81.59
Nvidia Xavier CPU	NoFS-RF	27932.96	775	0.090	81.59
	ES-RF	784313.73	542.125	0.019	77.28
	ES-DNN1	233918.13	537.5	0.015	70.34
	AE-DNN1	42553.19	1097.875	0.018	74.9
	AE-DNN2	4631.50	3037.5	0.021	79.03
Pynq-Z2 FPGA	ES-DNN1	176678.45	10.7875	0.455	65.73

Results of the preliminary step

Experiments
Experiment 1: IDS implementations to embed

- Goal: Evaluate the step that selects the IDS implementations to embed (offline phase)
- Evaluated methods:
  - **RAND**: A greedy random choice method
  - **IDS-Off-H**: An Heuristic based on Pareto Archive Evolution Strategy (PAES, [7])
  - **IDS-Off-MILP**: The Mixed Integer Linear Programming (MILP) formulation



Comparison between methods according to Inverted Generational Distance (IGD, [8])

Experiments
Experiment 2: Online phase evaluation

- Goal: Evaluate the online phase, with a set of 907400 combinations of dynamic constraints
- Evaluated methods:
  - **IDS-On-RC**: A Random Choice method
  - **IDS-On-MX**: A MaXimization method of Manhattan distance (L_1-norm)
  - **IDS-On-TP**: Technique for Order Preference by Similarity to Ideal Solution (TOPSIS, [9])

Selection methods	IDS-On-RC	IDS-On-MX	IDS-On-TP
Latency (s)	4.04e-05	4.12e-05	**2.52e-05**
Energy (J)	5.87e+01	**7.25e+00**	1.51e+01
Memory Peak (MB)	1.21e-01	3.57e-02	**2.80e-02**
Accuracy (%)	7.82e+01	**8.15e+01**	7.74e+01
Mean execution time (µs)	**66.93**	201.78	1022.00

Comparaison between methods to select the best IDS implementation to deploy

Conclusion and Perspectives

Conclusion and perspectives
Conclusion

AGENCE INNOVATION DÉFENSE

- Research questions:
  1. How to select a diversity of IDS implementations to embed into a single drone, while respecting a storage size dedicated to it ?
  2. How to select the best IDS implementation to deploy from this embedded set, depending on the context ?
- Solution:
  1. Offline phase: A phase executed before the mission, to select the IDS implementations to embed, according to the static constraints
  2. Online phase: A phase executed during the mission, to select the best IDS implementation to deploy, according to the dynamic constraints
- Results:
  1. Offline phase: IDS-Off-MILP performed better than other methods for IGD
  2. Online phase: IDS-On-MX make a good comprise between QoS metrics and temporal execution

Conclusion and perspectives
Perspectives

1. Consider more QoS metrics, such as per-class accuracies
2. Consider an early-exit IDS implementation to remove the offline phase
3. Study the possibility of distribution of IDS processing among a swarm of drones

Bibliography

[1] P. Svec et al. - Automated synthesis of action selection policies for unmanned vehicles operating in adverse environments (Autonomous Robots, 2012)

[2] A. Khraisa et al. - Survey of intrusion detection systems: techniques, datasets and challenges (Cybersecurity, 2019)

[3] M. Aiss et al. - Autonomous solar usv with an automated launch and recoverysystem for uav: State of the art and design (ICECOCS, 2020)

[4] C. Slimani et al. - Characterizing Intrusion Detection Systems On Heterogeneous Embedded Platforms (DSD, 2023)

[5] C. Slimani et al. - A study on characterizing energy, latency and security for Intrusion Detection Systems on heterogeneous embedded platforms (FGCS, 2024)

[6] V. Lannurien et al. - HeROcache: Storage-Aware Scheduling in Heterogeneous Serverless Edge – The Case of IDS (CCGrid, 2024)

[7] J. D. Knowles et al. - The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimization (CEC, 1999)

[8] C. A. Coello et al. - A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm (MICAI, 2024)

[9] S.-J. Chen et al. - Fuzzy Multiple Attribute Decision Making Methods (Springer, 1992)

[10] J. J. Thakkar - Technique for Order Preference and Similarity to Ideal Solution (TOPSIS) (Multi-Criteria Decision Making, 2021)

Appendix 1: Offline phase (1/3): Pre-filtering

- Goal: Select the IDS implementations that respect the static constraints, such as the available processing units

```
Input: Implementation Characterized (IC), Static Constraints (SC)
Output: Offline Pre-filtered Implementations (OffPI)
OffPI = {}
for implementation in IC do
    if implementation respects SC then
        add implementation to OffPI
```

Pseudo code of offline pre-filtering

Appendix 1:
Offline phase (2/3): Dominance filtering (1)

- Pareto Front: The concept of Pareto front or **set of optimal solutions in the space of objective functions** in multi-objective optimization problems (MOOPs) stands for a set of solutions **that are non-dominated to each other** but are superior to the rest of solutions in the search space (ScienceDirect).

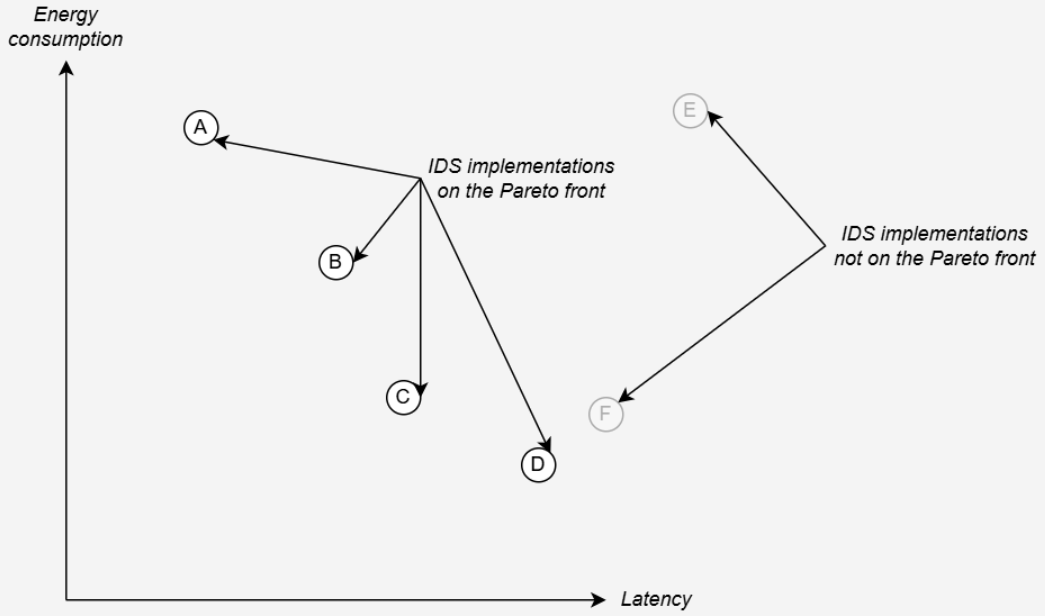


Illustration of Pareto front

Appendix 1: Offline phase (2/3): Dominance filtering (2)

AGENCE INNOVATION DÉFENSE

/ Goal: Select the IDS implementations that are not dominated

```
Inputs: Offline Pre-filtered Implementations (OffPI)
Output: Pareto front implementations (PS)
PS = {}
for implementation1 in OffPI do
   dominated = False
   for implementation2 in OffPI do
      if implementation1 != implementation2 then
         if implementation1 is dominated by implementation2
then
            dominated = True
  if dominated = False then
     add implementation1 to PS
```

Pseudo code of offline pre-filtering

Appendix 1:
Offline phase (3/3): MILP formulation (1)

- Goal: Select the IDS implementations to embed that maximize the diversity, while respecting the storage size dedicated to the IDS implementations
- **IDS-Off-MILP**: A Mixed Integer Linear Programming (MILP) formulation to find the approximate Pareto front (A) that minimize the Inverted Generational Distance (IGD) [6], while respecting the storage space constraint.

Appendix 1: Offline phase (3/3): MILP formulation (2)

AGENCE INNOVATION DÉFENSE

/ The Inverted Generational Distance (IGP) [6] maximize the diversity of the IDS implementation to embed.

$$IGD = \frac{1}{|PS|} \sum_{i \in PS} \min_{j \in A} d_{ij}$$

where d_{ij}: the Euclidian distance between point $i \in PS$, and $j \in A \subset$PS.

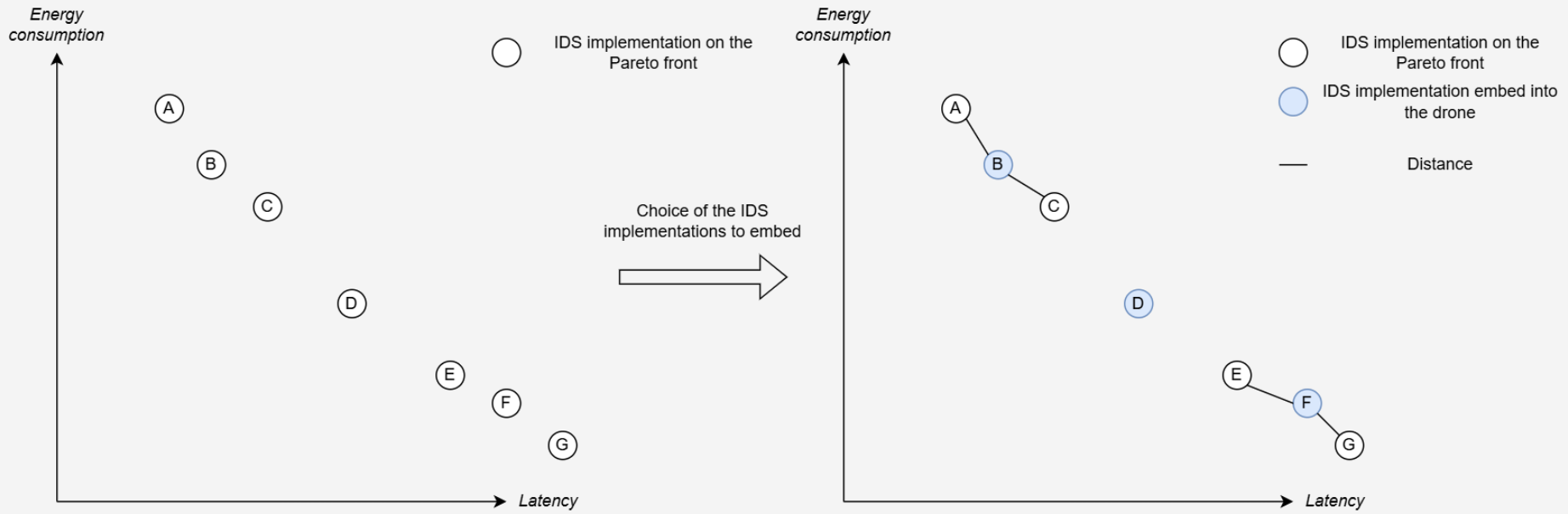


Illustration of IGD for selecting the IDS implementation to embed

An example of IGD computation

Université de Bretagne Occidentale

Lab-STICC

ENSTA BRETAGNE

Appendix 1: Offline phase (3/3): MILP formulation (3)

/ **IDS-Off-MILP** formulation:

$$A = \arg\min_{A \subset PS} \sum_{i \in PS} \sum_{j \in A} d_{ij} z_{ij}$$

Where A, the approximate Pareto front implementations

/ Decision variables:

- / x_i: A decision variable such as $x_i = 1$ if i is selected ($i \in PS$), 0 otherwise
- / z_{ij}: An auxiliary decision variable such as $z_{ij} = 1$ if $j \in A$ is the nearest point to $i \in PS$

/ Constraints:

- / $\sum_{i \in PS} s_i x_i < C$: The storage constraint is respected, with C: the static constraint about storage space
- / $\forall\, i \in PS,\ \sum_{j \in A} z_{ij} = 1$: There is a single point $j \in A$ that is nearest from i
- / $\forall\, i, j \in PS,\ z_{ij} < x_i : (z_{ij} = 1) \Rightarrow (x_i = 1)$
- / $\forall\, i, j \in PS,\ x_i \in \{0,1\}\ and\ z_{ij} \in \{0,1\}$: The decision variables are binary

Appendix 2: Online phase (1/2): Pre-filtering

- Goal: Select the IDS implementations that respect the dynamic constraints, such as the maximum acceptable latency

```
Inputs: Approximate Pareto front implementations (A), Dynamic Constraints (DC)
Output: Online Pre-filtered Implementations (OnPI)
OnPI = {}
for implementation in PS do
    if implementation respects DC then
        add implementation to OnPI
```

Pseudo code of online pre-filtering

Appendix 2
Online phase (2/2): IDS-On-MX (1)

- Goal: Find the best implementation i^* among the set of IDS implementations resulting from the previous step (OnPi).
- **IDS-On-MX**: Choose the implementation that is the farthest from the dynamics constraints according to the (normalized) Manhattan distance (L_1 norm)

$$i^* = \arg \max_{i \in Onpi} |A_i - A_c| + |E_c - E_i| + |L_c - L_i| + |M_c - M_i|$$

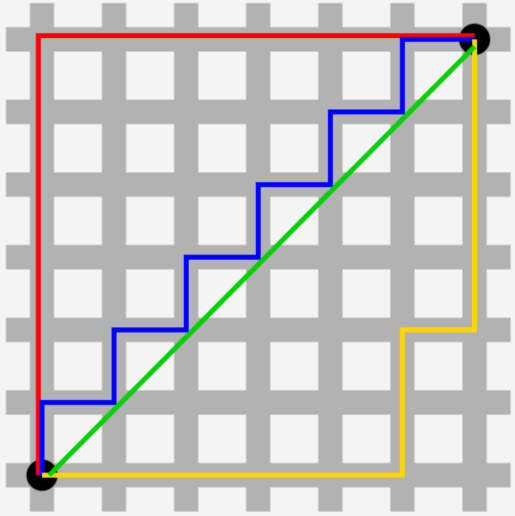
Where A_i, E_i, L_i and M_i are the (normalized) QoS metrics of implementation i, and A_c, E_c, L_c and M_c the (normalized) dynamic constraints

Appendix 2
Online phase (2/2): IDS-On-MX (2)

- **IDS-On-MX**: Choose the implementation that is the farthest from the dynamics constraints according to the (normalize) Manhattan distance (L_1 norm)

$$i^* = \arg \max_{i \in Onpi} |A_i - A_c| + |E_c - E_i| + |L_c - L_i| + |M_c - M_i|$$

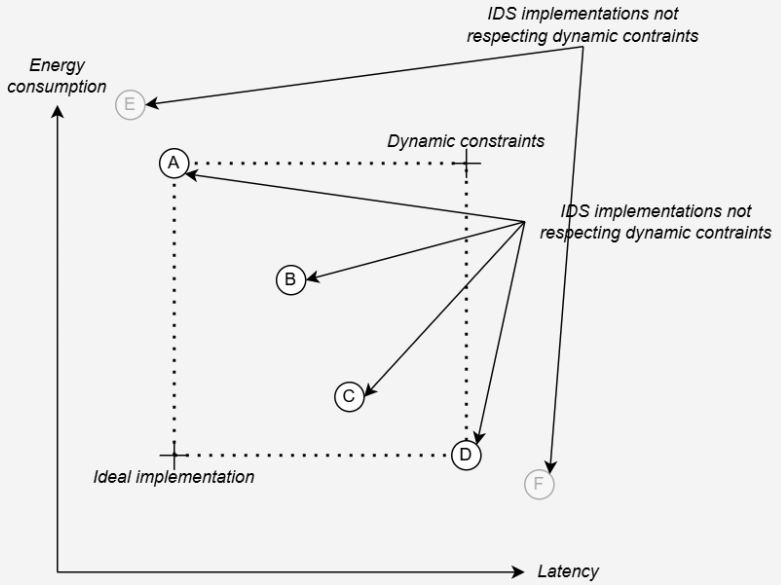
Where A_i, E_i, L_i and M_i are the QoS metrics of implementation i



Manhattan distance (Wikipédia)

Appendix 2
Online phase (2/2): IDS-On-TP (1)

- Goal: Find the best implementation i^* among the set of IDS implementations resulting from the previous step (OnPi).
- Problem: According to Euclidian distance point C is the nearest from ideal implementation, but points A and D are the farthest from anti-ideal implementation. Which one to choose ?



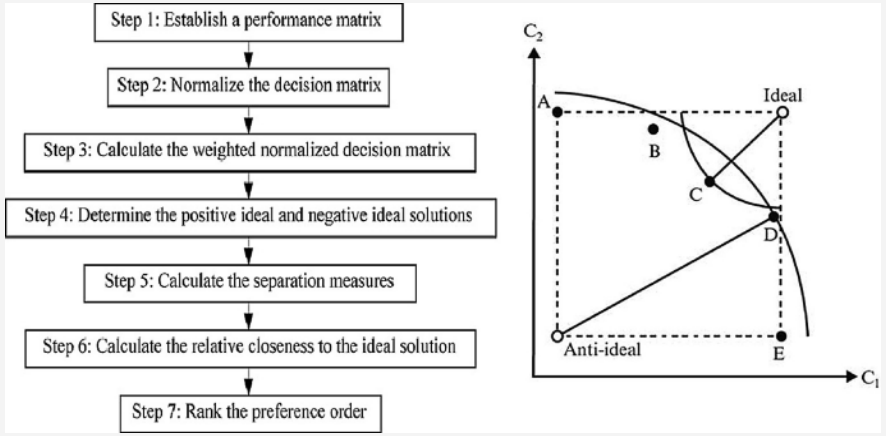
Selection problem overview

Appendix 2
Online phase (2/2): IDS-On-TP (2)

- **IDS-On-TP** using Technique for Order Preference by Similarity to Ideal Solution (TOPSIS, [7]): Choose the implementation that make the better compromise between an ideal solution and an anti-ideal solution

$$i^* = \arg \max_{i \in Onpi} \frac{E_i^-}{E_i^- + E_i^+}$$

Where E_i^-: the distance between the implementation i and an anti-ideal solution, and E_i^+: the distance between the implementation i and an ideal solution



TOPSIS overview [9]