Alma Mater Studiorum · Università di Bologna

SCHOOL OF SCIENCES
Master Degree in Mathematics
Curriculum Advanced Mathematics for Applications

TOWARDS QUANTUM APPROACHES FOR OBJECT DETECTION USING QUBO FORMULATIONS

Advisor:
Chiar.mo Prof.
Giacomo De Palma

Co-advisors:
Prof. Davide Pastorello
Dr. Gabriella Bettonte

Presented by:
Sandra Campacci

Session VI
Academic Year 2024-2025

Abstract

Object detection algorithms often struggle with highly redundant predictions (bounding boxes), especially when using heuristic algorithms in crowded scenarios. This thesis investigates a completely different approach, by translating the bounding box suppression task into a global mathematical optimization problem. Specifically, the chosen formulation is the Quadratic Unconstrained Binary Optimization (QUBO) model. In this way, the suppression task becomes a maximization problem: the system balances the confidence scores of each box against a set of spatial penalties. These penalties are computed by evaluating the overlap between pairs of bounding boxes, punishing the selection of redundant predictions to find the best overall combination of surviving boxes.

Since the QUBO model is mathematically equivalent to the Ising model, this optimization task is directly solvable using quantum hardware. Through an extensive analysis, this work compares the solutions and execution times on the MS COCO dataset across classical solvers (like the Gurobi Optimizer and Simulated Annealing) and the D-Wave Quantum Annealer. Our experimental results demonstrate that quantum hardware can maintain state-of-the-art detection accuracy while overcoming the exponential time complexity of classical solvers on images with a high density of bounding boxes.

Sommario

Gli algoritmi di object detection spesso faticano a gestire predizioni (bounding box) altamente ridondanti, specialmente quando si utilizzano algoritmi euristici in scenari affollati. Questa tesi esplora un approccio completamente diverso, traducendo il processo di soppressione delle bounding box in un problema di ottimizzazione globale. Nello specifico, la formulazione scelta è il modello QUBO (Quadratic Unconstrained Binary Optimization). In questo modo, il processo di soppressione diventa un problema di massimizzazione: il sistema bilancia i punteggi di confidenza di ciascuna box con una serie di penalità spaziali. Queste penalità vengono calcolate valutando la sovrapposizione tra coppie di bounding box, penalizzando la selezione di predizioni ridondanti in modo da trovare la migliore combinazione globale di box sopravvissute.

Poiché il modello QUBO è matematicamente equivalente al modello di Ising, questo problema di ottimizzazione è risolvibile direttamente utilizzando un hardware quantistico. Attraverso un'ampia analisi, questo lavoro confronta le soluzioni e i tempi di esecuzione sul dataset MS COCO tra solutori classici (come Gurobi e il Simulated Annealing) e il Quantum Annealer di D-Wave. I nostri risultati sperimentali dimostrano che l'hardware quantistico è in grado di mantenere un'accuratezza allo stato dell'arte e contemporaneamente superare la complessità esponenziale dei solutori classici sulle immagini con un'alta densità di box.

Contents

Introduction

Nowadays, object detection is one of the most active and challenging areas within computer vision. In particular, finding and tracking people in images has become relevant in everyday applications such as pedestrian detection, autonomous driving, crowd counting and video surveillance. However, person detection is notoriously difficult. Unlike rigid objects like cars or buses, human bodies change shape constantly: people can sit down, fold their arms, and are frequently hidden behind other objects. Consequently, finding exactly where subjects are located is not a trivial task, especially in complex scenarios where people can overlap or be partially occluded.

Modern deep learning architectures, such as Faster R-CNN, are very efficient at detecting people and objects. They analyze an image and return a set of predictions in the form of bounding boxes. These boxes are rectangular areas defined by spatial coordinates that theoretically contain the target. Each of these predictions comes with a confidence score that indicates how likely it is that the box contains the object. However, multiple overlapping boxes are usually predicted around a single target and suppressing these redundant boxes per object is a mandatory post-processing step that is a non-trivial task. The standard method used to filter these redundancies relies on heuristic algorithms like Non-Maximum Suppression (NMS). This approach is efficient, as it selects the box with highest confidence score and deletes all the predictions that highly overlap with it, but the algorithm easily struggles when analyzing highly crowded scenarios, frequently missing partially occluded objects.

To overcome this limitation, this thesis explores a completely different approach. We translate the bounding box suppression task into a global mathematical optimization problem. In particular, we formulate it as a Quadratic Unconstrained Binary Optimization (QUBO) model. Through this mathematical framework, the suppression task becomes a maximization problem, where each binary decision variable corresponds to a candidate bounding box. A value of 1 means the box is kept in the final output, while 0 means it is suppressed. The real strength of this formulation lies in its objective function. Our goal is to detect people within an image by maximizing the sum of the detector's confidence scores while minimizing the selection of bounding boxes with excessive spatial overlaps. To avoid multiple detections of the same individual, the selection of redundant predictions is "penalized" by spatial metrics that evaluate the overlap between pairs of bounding boxes. Consequently, the final solution of the QUBO problem represents the best overall combination of surviving boxes for that specific instance.

The true potential of this QUBO formulation is its direct compatibility with quantum computing. Since the QUBO model is mathematically equivalent to the Ising model from statistical physics, this optimization task can be solved using a Quantum Annealer. Specifically, Quantum Annealing is an approach that finds the global minimum of an objective function, particularly functional for combinatorial optimization and probabilistic sampling tasks. In practical implementations, such as D-Wave's commercially accessible Quantum Processing Units (QPUs), as the

quantum system evolves, it naturally settles into its lowest energy state, which mathematically corresponds to the optimal combination of bounding boxes.

Once we had mathematically translated the problem into a QUBO formulation, the subsequent challenge was proving it empirically. Our experimental phase focused entirely on the "person" class from the benchmarking MS COCO dataset. We wanted to test whether this rigorous formulation actually worked and how different computational paradigms handled it. First, we designed different arrangements of spatial penalties, which had to be balanced with the linear confidence scores using non-negative hyperparameters. This modeling required an extensive hyperparameter tuning phase, conducted using classical methods such as Gurobi and Simulated Annealing to find the best mathematical balance. Once the optimal parameters were obtained, we investigated the solver behaviors on single images. We started with classical solvers, like Brute Force and the Gurobi Optimizer, and probabilistic heuristics, such as Simulated Annealing, to validate the accuracy of the QUBO model. As expected, the execution time of classical solvers scaled as the number of bounding boxes in an image grew. This bottleneck motivated us to evaluate and compare these different computational paradigms with the quantum hardware. Through an extensive analysis, this work compares the accuracy and execution times of classical solvers against the D-Wave QPU. Finally, to prove that our approach was robust, we scaled our experiments to a large validation batch of 291 COCO images. To evaluate the results objectively, we relied on standard metrics, including mean Average Precision (mAP) for overall accuracy, the F1-score to balance precision and recall, and Mean Absolute Error (MAE) to track counting accuracy. These metrics specifically evaluate the overlap between the surviving bounding box and the Ground Truth provided by the COCO dataset.

The results of this study demonstrate the true potential of these QUBO formulations. In particular, when analyzing dense scenarios, the Quantum Annealer demonstrates significant computational efficiency, proving that a slight loss in accuracy is acceptable in exchange for flat scalability. These results confirm that exploring this formulation on quantum hardware offers an alternative for specific complex tasks in object detection.

This research project was carried out during a curricular internship at *E4 Computer Engineering*, an Italian company specialized in end-to-end solutions for High Performance Computing (HPC), Artificial Intelligence (AI) and Quantum Computing (QC). More recently, E4 has been a pioneer in quantum technologies, participating in several research projects at both the national and European levels.

Furthermore, it is important to acknowledge the quantum computational resources provided by CINECA, as access to these architectures today is not easily achievable. The execution of the developed Python code on actual quantum hardware was fundamental to achieving the final results of this work.

This thesis is organized into five main chapters, ranging from theoretical background to experimental results. The first chapter begins with an introduction to the QUBO problem and its physical equivalent, the Ising model, explaining how constraints are translated into penalty functions and how this framework is adapted to bounding box suppression. In the second chapter, we explore the different spatial metrics (such as Intersection over Union) used to penalize and

evaluate bounding box overlaps. Moreover, we formally describe the construction of the final coefficient matrices for the four penalty cases analyzed in this work, alongside the presentation of standard metrics for accuracy evaluation. The third chapter provides an overview of the different solution approaches. It examines the algorithmic logic and time complexities of classical methods (Brute Force and Gurobi) and probabilistic heuristics (Simulated Annealing and Quantum Annealing). In the fourth chapter, we present the experimental setup, starting from the chosen dataset (the standard MS COCO benchmark for image recognition, focusing on the "person" class) up to the Python code used to extract and process the predictions in pre-processing and post-processing steps. The last chapter presents all the experimental results. It starts with the hyperparameter tuning phase and the exact baseline validation, followed by a comparative analysis of scalability and accuracy on six selected images. Finally, it evaluates the global performance across a 291-image validation subset, comparing the results across different solvers.

Chapter 1
QUBO problems

This chapter outlines the mathematical foundation of the entire research: the Quadratic Unconstrained Binary Optimization (QUBO) model. We start by giving its formal definition, focusing on penalty terms to design the objective function. Although finding the exact global optimum of a general QUBO is an NP-hard problem, the actual difficulty strictly depends on the instance size. Small models are easily tractable, and even for large-scale problems, advanced classical heuristics can efficiently find near-optimal solutions in polynomial time. Thanks to this practical flexibility, the QUBO framework has become extremely popular for solving complex challenges across multiple applications, from logistics to machine learning.

The QUBO model is also mathematically equivalent to the Ising model from statistical physics. We will explore this connection to show how mapping binary variables to magnetic spins acts as the actual bridge between classical optimization and modern quantum computing.

We conclude this chapter by focusing on the suppression of overlapping bounding boxes in computer vision, which can be rigorously translated into a QUBO maximization task.

1.1 Definition

The QUBO model, acronym for Quadratic Unconstrained Binary Optimization, is a mathematical framework for solving combinatorial optimization problems using binary variables. Research on this model dates back to the late 1950s and early 1960s. Since that time, numerous papers have been published regarding the subject, covering various perspectives and including theoretical aspects, algorithms, and applications. The formal definition of QUBO model is expressed by the optimization problem:

$$\text{Opt}\, x^T Q x + c^T x,$$

with $x \in \{0,1\}^n, c \in \mathbb{R}^n$, and $Q \in \mathbb{R}^{n \times n}$, where x^T denotes the transpose of the binary vector x. Nevertheless, considering that $x_i^2 = x_i$ holds for a binary variable, the most used form is:

$$\text{Opt}\, x^T Q x,$$

which is easily achieved by adding c_i to q_{ii}, $\forall\, i \in \{1, 2, \ldots, n\}$.

It is also common to find the Q matrix in symmetric or upper triangular form and, if it is not, this can be realized without loss of generality:

- $\forall\, i, j,\ i < j$, replace both q_{ij} and q_{ji} with $(q_{ij} + q_{ji})/2$ to obtain a symmetric matrix;

- $\forall\, i, j,\ i < j$, replace q_{ij} with $q_{ij} + q_{ji}$ and replace q_{ji} with 0 to obtain an upper triangular matrix.

To sum up, the orientation of the optimization and the Q matrix outline uniquely the QUBO problem: (Q, min) if we have a minimization problem, (Q, max) for the maximization one. Observe that (Q, max) problem is always equivalent to ($-Q$, min).

Apart from the binary restrictions on the decision variables, QUBO functions as an unconstrained model where all the problem data is represented within the Q matrix. These features render the QUBO model especially useful as a framework for modeling combinatorial optimization issues, providing an alternative to traditional constrained representations. Constraints are not expressed explicitly; they are absorbed as penalty terms into the objective function. To ensure validity, a penalty function is constructed to equal zero when the constraint is satisfied and to assign a positive cost otherwise. By adding these quadratic penalties to the objective function, the original constrained problem is mathematically transformed into a QUBO formulation. For some constraints, the corresponding quadratic penalties are already known and are reported in Table 1.1.

Classical constraint	**Equivalent penalty**
$x_1 + x_2 \leq 1$	$P(x_1x_2)$
$x_1 + x_2 \geq 1$	$P(1 - x_1 - x_2 + x_1x_2)$
$x_1 + x_2 = 1$	$P(1 - x_1 - x_2 + 2x_1x_2)$
$x_1 \leq x_2$	$P(x_1 - x_1x_2)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x_1 = x_2$	$P(x_1 + x_2 - 2x_1x_2)$

Table 1.1: A few known constraint/penalty pairs. Source: [10].

In cases where QUBO formulation is non-trivial or where explicit penalty terms are unknown, one can use the following approach. Given a general 0/1 optimization problem $\min x^TCx$ subject to $Ax = b$ with $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$ with integer components, and $x \in \{0,1\}^n$, it is possible to transform the constraints $Ax = b$ into quadratic penalties to be added to the objective function. For a positive scalar P, the quadratic penalty is written as $P(Ax-b)^T(Ax-b)$, so the objective function becomes:

$$\begin{aligned} y &= x^TCx + P(Ax-b)^T(Ax-b) \\ &= x^TCx + P(x^TA^T - b^T)(Ax-b) \\ &= x^TCx + P(x^TA^TAx - 2b^TAx + b^Tb) \\ &= x^T(C + PA^TA)x - (2Pb^TA)x + Pb^Tb \\ &= x^TQx + k. \end{aligned}$$

Where Q is the unified matrix obtained using the property $x_i^2 = x_i$ and $k \in \mathbb{R}$ is an additive constant that does not affect the optimization problem. So the

equivalent unconstrained version of the constrained problem becomes:

$$\text{QUBO} : \min_{x \in \{0,1\}^n} x^T Q x.$$

As one can see, when reformulating the problems, the process requires the introduction of a scalar penalty P for which a positive number has to be assigned. Consequently, these penalties are not unique, so, for a particular problem, a workable value is typically set based on domain knowledge and on what are the needs. If a constraint must absolutely be satisfied, then P must be large enough to preclude a violation; conversely if the penalty is "soft" [10, 14, 21].

1.2 Applications

QUBO models belong to the class of NP-hard problems, implying that exact solutions are generally computationally difficult for large-scale instances. Fortunately, modern metaheuristic methods can find high-quality (though not always optimal) solutions in a reasonable time. These new techniques have achieved impressive results and are helping to connect classical computing with quantum computing.

Nowadays, QUBO is one of the most widely implemented optimization models in the research area, unifying a big variety of combinatorial optimization problems, which have real-world applications in both the public and private sectors. Indeed, any linear or quadratic problem with constraints and bounded integer variables can be reformulated as QUBO using penalty functions. Since various NP-Hard and NP-complete combinatorial optimization problems can be mapped to this structure, the model proves to be very general and effective when solved with modern algorithms.

Tests show that this approach challenges the traditional belief that optimization models should strictly maintain linearity to exploit specific problem structures. While linear methods are useful in many cases, experience shows that strictly following them can sometimes prevent finding the best solutions. When a problem is converted to QUBO, its original linearity and structure are lost and absorbed into the Q matrix; even so, QUBO-based methods have achieved excellent results, often matching or surpassing specialized techniques.

Consequently, the QUBO model is currently employed as a substitute modeling and solution method for an increasing number of significant issues across government, industry, and science. In operations management and logistics, applications range from job scheduling, vehicle routing, and autonomous path planning, to warehouse location, product distribution, and power system design. In the financial sector, QUBO provides robust frameworks for portfolio optimization, capital budgeting, arbitrage, and credit risk assessment. Furthermore, it offers a natural mathematical formulation for fundamental NP-hard problems in computer science and graph theory, including Constraint Satisfaction Problems, boolean SAT, large-scale set partitioning, graph coloring, maximum cut (Max-Cut), maximum independent set, the multiple knapsack problem, and the quadratic assignment problem.

Beyond classical operations research, QUBO models are deeply influencing modern artificial intelligence, cybersecurity, and applied sciences. In Machine Learning and Deep Learning, QUBO is utilized for modularity maximization, matrix factorization, and optimizing Gaussian Process Sampling to enhance training set selection. In computational biology and chemistry, it addresses complex challenges like the RNA folding problem, molecular similarity, and the accelerated discovery of new drugs and materials.

The successful computational results across many types of problems suggest that the same approach could work effectively for an even broader range of applications. Due to its flexibility, the QUBO framework encourages further exploration and application in new problem areas, which is why researchers are still working to improve x^TQx solution methods. Finally, QUBO has recently proven to be a highly effective alternative to traditional heuristics in Computer Vision, specifically for refining bounding box suppression in Object Detection tasks [14, 21].

1.3 Ising Model

The importance of QUBO model grows even more because it is mathematically equivalent to the Ising spin glass model: a fundamental framework widely used in statistical mechanics. Moving from a QUBO formulation to an Ising model is a fundamental operation in both combinatorial optimization and quantum computing.

Let $\{1, 2, \ldots, n\}$ be a set of particles and let $s_i \in \{-1, 1\}$ be the spin of i-th particle, representing either a "down" or an "up" magnetic orientation. The joint interaction field between neighboring spins i and j is indicated by a_{ij}, while b_i represents the strength of the magnetic field applied to particle i. Given a spin configuration vector $s \in \{-1, 1\}^n$, the total energy of the system is described by the associated Hamiltonian:

$$H(s) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} s_i s_j + \sum_{i=1}^{n} b_i s_i.$$

In statistical mechanics, physical systems naturally evolve toward their lowest possible energy state. Consequently, predicting the behavior of the system reduces to a combinatorial optimization problem: finding the spin configuration that minimizes the global energy state of the Hamiltonian [20, 21].

From an optimality point of view, the QUBO and Ising models are equivalent. Formally speaking, one can apply the linear QUBO-Ising transformation $x_i \mapsto \frac{s_i+1}{2}$ $\forall\, i = 1, 2, \ldots, n$ or vice versa the Ising-QUBO transformation $s_i \mapsto 2x_i - 1$ $\forall\, i = 1, 2, \ldots, n$. This bidirectional mapping implies that the QUBO model can also be utilized to solve problems from physics, making it a bridge between optimization and physical modeling.

Historically, the Ising model was developed to study magnetic systems like spin glasses. Today, however, it has found a second life connecting complex optimization with physics-inspired computing. Once an optimization problem is

translated into an Ising Hamiltonian, it can be solved using algorithms such as Simulated Annealing, which progressively reaches the lowest-energy structure by escaping local minima. Alongside this approach, traditional metaheuristics such as Tabu Search and Scatter Search have consistently proven to be highly efficient and reliable for solving large-scale QUBO and Ising instances on classical and hybrid computers.

1.4 Object detection and box suppression

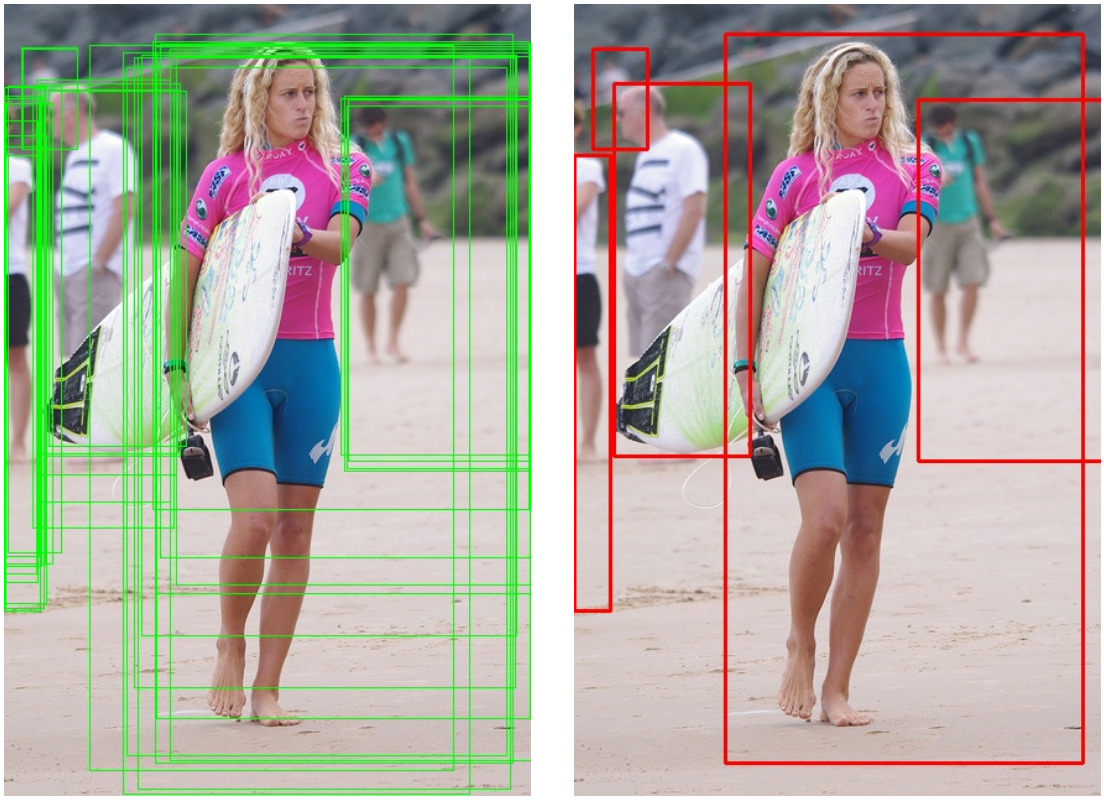
We now shift our focus to one of the most inspiring applications of the QUBO framework. Object detection, in particular person detection and tracking in images, is a significant problem applicable to various situations such as pedestrian detection, facial recognition, automatic driving, medical imaging, industrial quality control, people counting, video surveillance and security. Nonetheless, the problem is extremely complex, especially because most scenes are characterized by occlusions and overlaps. To address this, computer vision researchers have advanced different approaches based on mathematical modeling, statistical machine learning and constraint satisfaction problems.

Since a major challenge is person detection in a crowded circumstance, researchers have trained different preprocessing algorithms in order to extract a set of candidate bounding boxes. These are defined as rectangular regions characterized by four spatial coordinates or by two spatial coordinates along with width and height, that identify (theoretically) each possible individual within the frame. However, this detection typically produces a highly redundant output, generating many overlapping boxes around the same object, as illustrated in Figure 1.1a. To solve this problem, it is strictly necessary to apply a post-processing filter. This step suppresses the redundant predictions in order to identify the person, eventually refined the set of detections, as reported in Figure 1.1b.

Non-Maximum Suppression (NMS) is a heuristic-based method widely used to suppress these redundant detections. Let $\{1, 2, \ldots, n\}$ be a set of indices for the bounding boxes detected in a given image and let $c_i \in [0, 1]$ be the confidence score associated with the i-th bounding box $\forall\, i = 1, 2, \ldots, n$. The algorithm starts by sorting the bounding boxes in descending order of their confidence scores, immediately suppressing all boxes with a c_i value strictly lower than a fixed confidence threshold. To conclude, the algorithm iteratively selects the box with the highest confidence score and suppresses the others that "highly" overlap with the selected one. While several advanced variants of this basic technique have been developed through the years (such as Soft-NMS [1]), an exhaustive review of these heuristic methods falls outside the scope of this thesis.

While NMS is highly efficient for standard images, it is not so accurate in highly crowded scenarios, since it frequently suppresses valid but partially occluded objects with low confidence scores. To overcome this limitation, this work explores the replacement of heuristic NMS with a global mathematical optimization approach.

Our goal is to detect people within an image by maximizing the sum of the detector's confidence scores while minimizing the selection of bounding boxes with



(a) Redundant bounding boxes detected by the model.

(b) Final output after applying a suppression algorithm.

Figure 1.1: Example of bounding box suppression.

excessive overlaps to avoid multiple detections of the same individual. Using the previously defined notation, this task can be formulated as a QUBO problem, where each detected bounding box is represented by a binary variable $x_i \in \{0, 1\}$, indicating whether the box is accepted ($x_i = 1$) or suppressed ($x_i = 0$) as redundant [21].

The objective function for this task is conceptually defined as follows:

$$\max \sum_{i=1}^{n} c_i x_i - \text{penalties due to overlapping.}$$

Mathematically, the confidence scores are placed in a diagonal matrix L:

$$L = \begin{bmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_n \end{bmatrix}$$

Simultaneously, a symmetric penalty matrix P is constructed, where $p_{ij} \in (0, 1]$, $\forall\, i \neq j$, if boxes i and j have a significant area overlap ratio, and 0 otherwise.

Thus the matrix is written as:

$$P = \begin{bmatrix} 0 & p_{12} & \cdots & p_{1n} \\ p_{21} & 0 & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & 0 \end{bmatrix}$$

Finally, the person detection problem can be rigorously formulated as:

$$\max_{x \in \{0,1\}^n} \left(\sum_{i=1}^{n} c_i x_i - \sum_{i \neq j} p_{ij} x_i x_j \right), \; x \in \{0,1\}^n.$$

In practice, to balance confidence maximization and overlap suppression, the final QUBO matrix Q is constructed as a linear combination of the confidence matrix L and the penalty matrix P. This can be simply expressed as:

$$Q = \alpha L - \beta P,$$

where α and β are non-negative hyperparameters satisfying the condition $\alpha + \beta = 1$ [26]. The mathematical representation of QUBO matrix is:

$$Q = \begin{bmatrix} \alpha c_1 & -\beta p_{12} & \cdots & -\beta p_{1n} \\ -\beta p_{21} & \alpha c_2 & \cdots & -\beta p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\beta p_{n1} & -\beta p_{n2} & \cdots & \alpha c_n \end{bmatrix}$$

The exact definitions of different penalties and the evaluation of these hyperparameters will be extensively discussed in the next chapters.

Chapter 2
Model

In this chapter, we describe the mathematical and evaluative core of our proposed bounding box suppression model. It is rigorously defined how we mathematically penalize the selection of highly overlapping bounding boxes through three distinct spatial metrics: Intersection over Union (IoU), Intersection over Minimum (IoM) and Spatial Feature. Then, we explain in detail how these penalties are combined in the final QUBO matrix. To conclude, we present all the evaluation metrics used to assess the accuracy of our model.

2.1 Penalties

In the QUBO formulation for the bounding box suppression problem, the off-diagonal elements of the Q matrix represent the penalties applied when two overlapping boxes are selected simultaneously. To quantify this spatial overlap and assign the appropriate penalty weights, we introduce and compare three different metrics: Intersection over Union (IoU), Intersection over Minimum (IoM) and Spatial Feature.

2.1.1 Intersection over Union

Intersection over Union (IoU) measures the spatial overlap between two predicted bounding boxes. IoU is a ratio that goes from 0 to 1: a higher score means that the boxes are highly overlapped, so one of them has to be suppressed. Given two boxes A and B, we define IoU as:

$$\text{IoU}(\text{box}_A, \text{box}_B) = \frac{\text{Area}_{A \cap B}}{\text{Area}_{A \cup B}}.$$

As illustrated in Figure 2.1, the IoU score decreases rapidly when the boxes are not perfectly aligned. In the context of our QUBO formulation, this characteristic makes IoU an effective penalty measure to discourage the simultaneous selection of highly overlapping predictions.

2.1.2 Intersection over Minimum

Intersection over Minimum (IoM) is an alternative metric that calculates the ratio between the area of the intersection and the area of the smaller of the two bounding boxes. Like IoU, it ranges from 0 to 1: a score of 1 indicates that the smaller box is completely enclosed within the larger one. Given two boxes A and B, we define IoM as:

$$\text{IoM}(\text{box}_A, \text{box}_B) = \frac{\text{Area}_{A \cap B}}{\min(\text{Area}_A, \text{Area}_B)}.$$

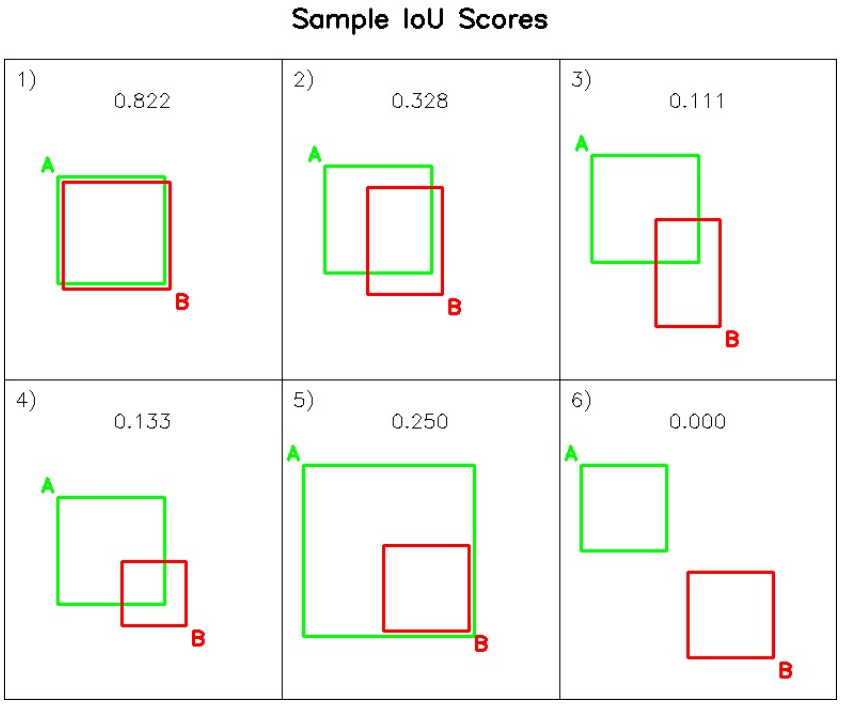


Figure 2.1: Examples of IoU scores with different overlaps.

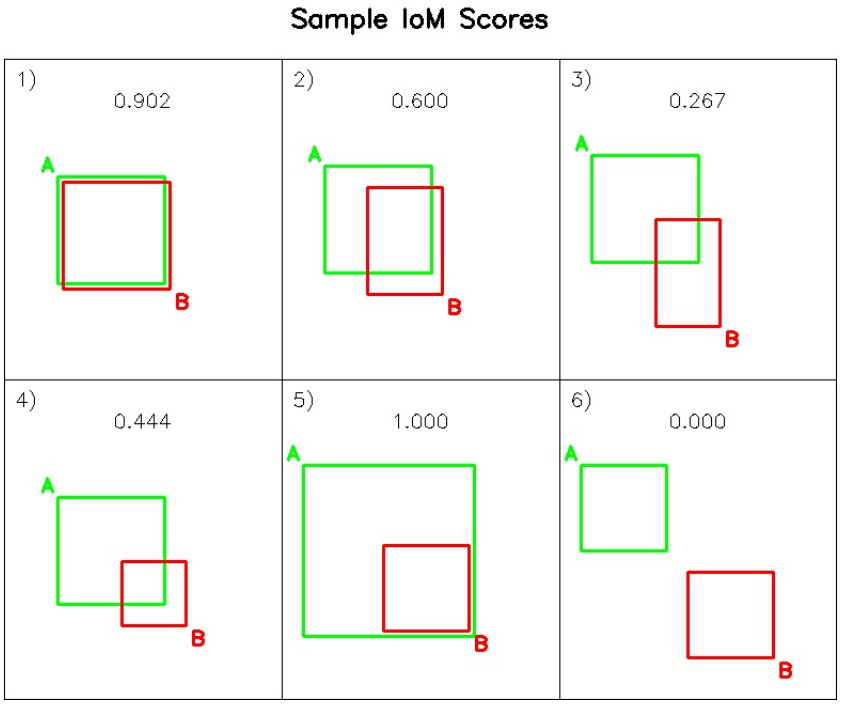


Figure 2.2: Examples of IoM scores with different overlaps.

Figure 2.2 shows examples of IoM scores calculated for different bounding box configurations, highlighting how this metric heavily penalizes a small box that is entirely contained within a larger one, regardless of the larger box's total area.

2.1.3 Spatial Feature

The Spatial Feature[1] is a metric that evaluates the overlap by using the geometric mean of the two bounding box areas in the denominator. This approach acts as a compromise between IoU and IoM, providing a smoother penalization when comparing boxes of significantly different sizes. Given two boxes A and B, we define the spatial feature as:

$$\text{Spatial Feature}(\text{box}_A, \text{box}_B) = \frac{\text{Area}_{A\cap B}}{\sqrt{\text{Area}_A \cdot \text{Area}_B}}.$$

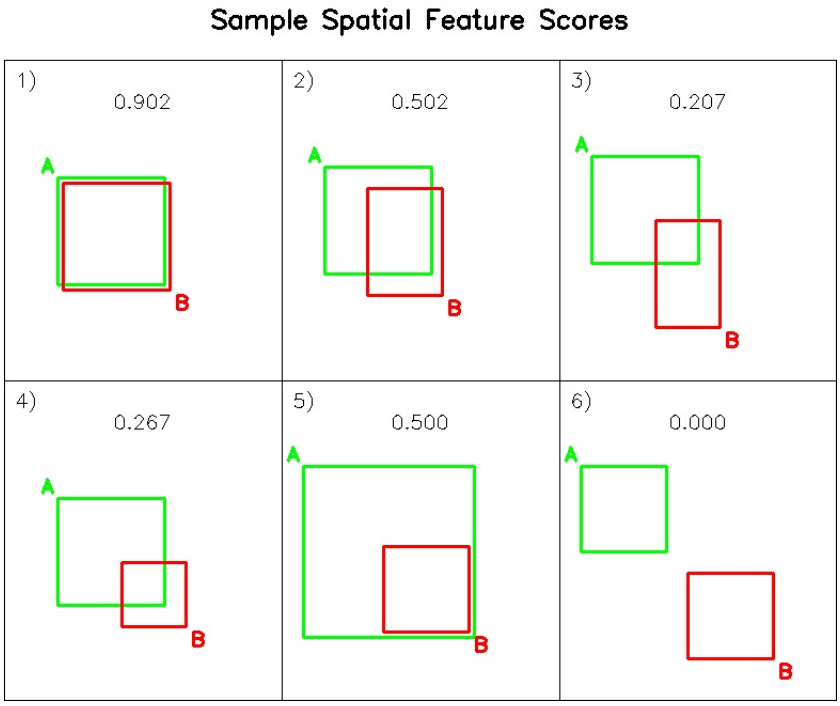


Figure 2.3: Examples of spatial feature scores with different overlaps.

Visual examples of this metric are displayed in Figure 2.3, demonstrating its intermediate behavior compared to the previous two formulations [15, 16, 26].

2.1.4 Penalties comparison

These three penalties are mathematically related to each other through a strict chain of inequalities. Specifically, for any given pair of bounding boxes, the relation IoU $\leq$ Spatial Feature $\leq$ IoM always holds.

To prove this, let us assume without loss of generality that $\text{Area}_B \leq \text{Area}_A$. This assumption implies two facts: first, that $\min(\text{Area}_A, \text{Area}_B) = \text{Area}_B$; second, that $\text{Area}_{A\cap B} \leq \text{Area}_B$.

[1]To indicate this metric, we use the same appellation as [26].

First of all, it is possible to show that Sp. Feat. $\leq$ IoM, indeed:

$$\begin{aligned}\text{Sp. Feat.} \leq \text{IoM} &\iff \frac{\text{Area}_{A\cap B}}{\sqrt{\text{Area}_A \cdot \text{Area}_B}} \leq \frac{\text{Area}_{A\cap B}}{\text{Area}_B} \\ &\iff \text{Area}_B \leq \sqrt{\text{Area}_A \cdot \text{Area}_B} \\ &\iff \sqrt{\text{Area}_B} \leq \sqrt{\text{Area}_A} \\ &\iff \text{Area}_B \leq \text{Area}_A.\end{aligned}$$

Secondly, one can prove that IoU $\leq$ Sp. Feat. using a similar approach:

$$\begin{aligned}\text{IoU} \leq \text{Sp. Feat.} &\iff \frac{\text{Area}_{A\cap B}}{\text{Area}_{A\cup B}} \leq \frac{\text{Area}_{A\cap B}}{\sqrt{\text{Area}_A \cdot \text{Area}_B}} \\ &\iff \sqrt{\text{Area}_A \cdot \text{Area}_B} \leq \text{Area}_{A\cup B}.\end{aligned}$$

This last inequality is always satisfied. Indeed, using the initial assumption ($\text{Area}_B \leq \text{Area}_A$), it follows that $\sqrt{\text{Area}_A \cdot \text{Area}_B} \leq \text{Area}_A$. Furthermore, by the definition of union, $\text{Area}_A \leq \text{Area}_{A\cup B}$. Combining these properties yields:

$$\sqrt{\text{Area}_A \cdot \text{Area}_B} \leq \text{Area}_A \leq \text{Area}_{A\cup B}.$$

This confirms that the denominator of the Spatial Feature is always smaller than or equal to the denominator of the IoU.

2.2 Construction of coefficient matrix and Hyperparameters

As presented in Section 1.4, given n bounding boxes detected in an image, the associated QUBO problem to suppress them is given by a $n \times n$ matrix linearly constructed using a confidence matrix L and a penalty matrix P. The rigorous formulation is given by:

$$\text{QUBO} : \max_{x\in\{0,1\}^n} x^T Q x,$$

where $Q = \alpha L - \beta P$. The hyperparameters α and β are non-negative by definition and always satisfy the constraint $\alpha + \beta = 1$.

While the confidence matrix L is fixed, many options can be chosen to penalize overlap between boxes by combining the penalties presented in Section 2.1 that will occur as off-diagonal terms in the P matrix. In particular, since overlapping is a symmetric feature (in the sense that "box$_i$ is overlapped to box$_j$" is equivalent to "box$_j$ is overlapped to box$_i$"), our penalty matrix is always symmetric with its main diagonal strictly composed of zeros ($p_{ii} = 0$, $\forall\, i = 1, \ldots, n$).

The first case is the most straightforward and includes only the IoU score: each element $p_{ij}\, \forall\, i = 1, \ldots, n$, $i \neq j$ is computed as $\text{IoU}(\text{box}_i, \text{box}_j)$. Starting from this basic case, three additional cases are considered by adding other penalties to IoU.

In order to build the second case, we add the "strongest" penalty, which is the IoM score. Since IoM applies a strictly severe penalty to enclosed bounding

boxes, assigning it a lower weight (0.3) compared to IoU (0.7) prevents over-suppression of valid detections, achieving a more balanced spatial penalization. So the combination $\forall\, i = 1, \ldots, n,\ i \neq j$ is $p_{ij} = 0.7 \cdot \text{IoU}(\text{box}_i, \text{box}_j) + 0.3 \cdot \text{IoM}(\text{box}_i, \text{box}_j)$.

The third case is built by considering IoU and Spatial Feature penalties. This instance is represented by splitting the P matrix into P_1 for IoU score and P_2 for Spatial Feature, together with the splitting of the hyperparameter β into β_1 and β_2. Consequently the computations become: $p^1_{ij} = \text{IoU}(\text{box}_i, \text{box}_j)$ and $p^2_{ij} = \text{Spatial Feature}(\text{box}_i, \text{box}_j)$, $\forall\, i = 1, \ldots, n,\ i \neq j$. In the end, the Q matrix is written as $\alpha L - (\beta_1 P_1 + \beta_2 P_2)$ such that $\alpha + \beta_1 + \beta_2 = 1$ [26].

Finally, we consider a more complex case that includes all the three penalties by merging case 2 and case 3: $p^1_{ij} = 0.7 \cdot \text{IoU}(\text{box}_i, \text{box}_j) + 0.3 \cdot \text{IoM}(\text{box}_i, \text{box}_j)$ and $p^2_{ij} = \text{Spatial Feature}(\text{box}_i, \text{box}_j)$, $\forall\, i = 1, \ldots, n,\ i \neq j$. The QUBO matrix is written as in case 3 with the same constraint.

Summarizing, these are the four analyzed cases:

- **Case 1 (Baseline)**: $Q = \alpha L - \beta P$, where $P = \text{IoU}$;
- **Case 2 (Mixed IoU-IoM)**: $Q = \alpha L - \beta P$, such that $P = 0.7 \cdot \text{IoU} + 0.3 \cdot \text{IoM}$;
- **Case 3 (IoU and Spatial Feature)**: $Q = \alpha L - (\beta_1 P_1 + \beta_2 P_2)$, where $P_1 = \text{IoU}$ and $P_2 = \text{Spatial Feature}$;
- **Case 4 (Comprehensive)**: $Q = \alpha L - (\beta_1 P_1 + \beta_2 P_2)$ as above, but $P_1 = 0.7 \cdot \text{IoU} + 0.3 \cdot \text{IoM}$.

For the first two cases, the hyperparameters satisfy $\alpha + \beta = 1$, whereas for the last two cases, the constraint is generalized to $\alpha + \beta_1 + \beta_2 = 1$.

As a final remark, it is worth clarifying that adding different penalty metrics does not change the sparsity of the QUBO matrix. The number of non-zero elements in Q is exactly the same across all four cases. Since IoU, IoM, and Spatial Feature all share the intersection area as their numerator, if two boxes do not overlap, their intersection is zero, making all three metrics zero as well. On the other hand, if there is an overlap, all metrics will be non-zero. Consequently, moving from the baseline Case 1 to the comprehensive Case 4 does not create new quadratic connections in the problem graph; it simply modifies the values of the existing penalties, keeping the combinatorial difficulty strictly unchanged for the solvers.

2.3 Evaluation Metrics

Before defining the evaluation metrics, it is fundamental to recall the four possible outcomes for a binary classifier in the context of object detection:

- True Positives (TP): Number of samples correctly predicted as “positive”.
- False Positives (FP): Number of samples wrongly predicted as “positive”.
- True Negatives (TN): Number of samples correctly predicted as “negative”.

- False Negatives (FN): Number of samples wrongly predicted as "negative".

In object detection tasks, classifying a prediction as a TP or FP relies heavily on the spatial overlap with the corresponding Ground Truth (GT), which is evaluated using the IoU metric presented in Section 2.1.1. Each detection is assigned to a GT object: if the IoU is above a certain threshold (commonly 0.5), the detection is considered a TP; otherwise, it is an FP. In Figure 2.4, we provide a visual example of the classification of these predictions. The red boxes are the GT and the green ones are the predicted detections. The green box on the left side correctly matches its GT with an IoU score of 0.68, so it is classified as a TP. On the other hand, the one on the right is classified as an FP, since its overlap with the GT is only 25% (IoU = 0.25). Additionally, the red boxes without a corresponding valid prediction are classified as False Negatives, representing missed targets.

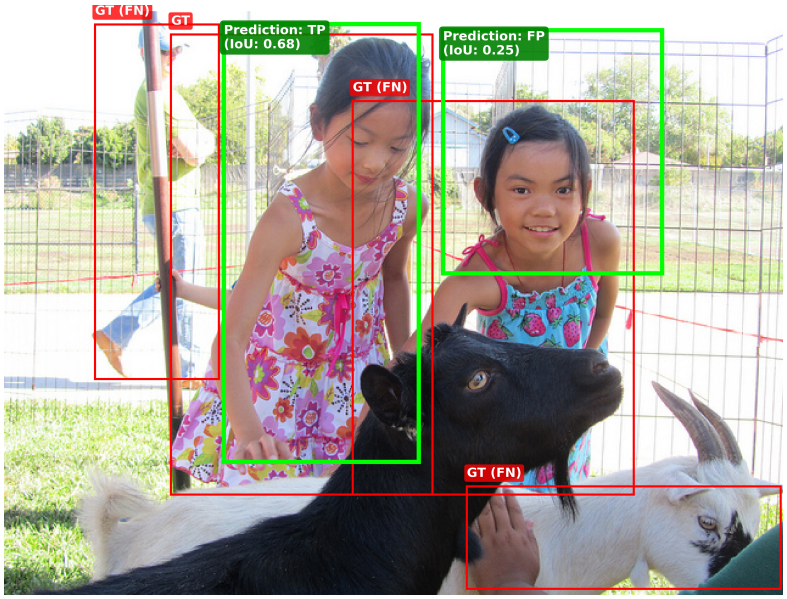


Figure 2.4: Visual representation of True Positives (TP), False Positives (FP), and False Negatives (FN) in object detection. Red boxes represent Ground Truth annotations, while green boxes indicate the model's predictions evaluated at an IoU threshold of 0.5.

Note that in Object Detection, TNs are all the possible background bounding boxes that the model correctly ignored, so they are not used in standard evaluation metrics.

Furthermore, when multiple bounding boxes cover the same GT object, only the one with the highest confidence score is counted as a TP, while all the others are classified as FPs.

Based on these definitions, we can formally introduce the evaluation metrics utilized in this work to assess the model's performance.

2.3.1 Precision, Recall and F1-Score

Precision measures the reliability of predictions, i.e., the percentage of correct predictions. High precision indicates that when the model predicts an object, it is likely to be correct, minimizing false positives. It is mathematically defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{total predictions}}.$$

Using this metric, it is possible to measure the accuracy of true positives, which indicate the model's ability to find only relevant objects without including irrelevant ones. Precision ranges from 0 to 1 and achieving a high value means there are few false alarms in people detection.

Recall indicates the model's ability to find all existing objects. A high value means that the model captures most of the objects in the scene, minimizing false negatives. It is expressed as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{GT}}.$$

It ranges from 0 to 1 and is penalized by missing detections, so its value corresponds to the percentage of people found.

F1-Score measures model's accuracy by combining Precision and Recall, leading to its widespread use in recent literature. Mathematically speaking, it is a harmonic mean because it encourages similar values for Precision and Recall. That is, the more the Precision and Recall scores deviate from each other, the worse the harmonic mean. It is computed as:

$$\text{F1} = 2\frac{\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}.$$

This metric ranges from 0 to 1 and it is the best choice between Precision and Recall if one is neutral towards false positives and false negatives because maximizing the F1 score implies simultaneously maximizing both Precision and Recall [13].

2.3.2 Mean Average Precision and Mean Average Recall

Mean Average Precision (mAP) represents the benchmark metric for evaluating the performance of computer vision models, especially in object detection and instance segmentation tasks with object detectors such as Faster R-CNN, SSD, etc. Unlike common classification accuracy, which only verifies the correctness of the label, mAP simultaneously measures the model's ability to identify the object class and locate it within bounding boxes. This metric was created to improve interpretability and to give increased visibility to performance at low recall.

Average Precision (AP) is generally defined as the area under the Precision-Recall curve. Since Precision and Recall both range from 0 to 1, AP is also within 0 and 1. Finally, the average of the APs of each class is calculated to obtain the metric that represents in a single value the prediction accuracy of the model for the entire dataset (mean Average Precision).

Mean Average Recall (mAR) is another standard evaluation metric that measures how well a model can find all objects in the scene, regardless of false positives. It highlights how well a model detects objects, often reported for small, medium, and large objects; indeed it only penalizes false negatives to avoid losing any targets. Intuitively, while Recall measures the ability to find all relevant objects, mAR narrows down how many detections are considered for each class. Mathematically speaking, this metric is computed using a fixed IoU threshold. It is commonly used in object detection tasks alongside mAP, and differs from the latter as it does not rely on the Precision-Recall curve [8, 13].

2.3.3 Mean Absolute Error and Root Mean Square Error

Mean Absolute Error (MAE) is a popular metric that measures the average absolute difference between the total amount of predicted boxes and the number of ground truths. Let N be the total number of images in the dataset, $\hat{y}_i$ be the number of predicted boxes for the i-th image and y_i be the actual number of Ground Truth boxes. MAE is calculated as:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|.$$

MAE helps assess the accuracy of a method, and the aim is to minimize it to zero, as its score indicates how many more (or fewer) objects are found on average.

Root Mean Square Error (RMSE) is one of the most commonly used measures of the differences between predicted and true values. Using the previously defined notation, it is formulated as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}.$$

This metric is highly susceptible to data outliers. Since the errors are squared before being averaged, large mistakes have an excessively negative effect on the total value. This sensitivity is really helpful for recognizing and understanding the effects of extreme predictions on the model's performance. As with MAE, the goal is to obtain a value as close as possible to 0.

Chapter 3
Solution Approaches

In the field of computer science, there exist many different algorithmic strategies, particularly for optimization problems. When dealing with a QUBO problem, the choice of an appropriate solver heavily depends on the instance size, as the computational complexity generally grows with the number of variables. Therefore, this chapter analyzes and compares four different QUBO solvers, ranging from exact approaches to advanced classical and quantum heuristics: Brute Force, Gurobi Optimizer, Simulated Annealing (SA), and Quantum Annealing (QA).

3.1 Brute Force

The Brute Force algorithm is a naive approach that evaluates every possible solution to a problem and returns the one that maximizes (or minimizes) the objective function. This algorithm always produces an exact solution and is generally very simple to implement. The procedure for "brutally" solving a maximization QUBO problem is presented in Algorithm 1.

Algorithm 1 Brute Force (Exhaustive Search) for QUBO

Input: QUBO matrix $Q \in \mathbb{R}^{n \times n}$
Output: Optimal solution x_{opt} and maximum energy E_{opt}
$E_{\text{opt}} \leftarrow 0$
$x_{\text{opt}} \leftarrow 0^n$
for each candidate solution $x \in \{0,1\}^n$ **do**
    $E \leftarrow x^T Q x$
    if $E > E_{\text{opt}}$ **then**
        $E_{\text{opt}} \leftarrow E$
        $x_{\text{opt}} \leftarrow x$
    end if
end for
return $x_{\text{opt}}, E_{\text{opt}}$

Nevertheless, the main disadvantage of Brute Force is its inefficiency. Given a QUBO problem defined by n binary variables, the solution space consists of all possible combinations of 0 and 1, so the possible solutions are exactly 2^n. Consequently, the time complexity scales exponentially and the algorithm rapidly becomes inefficient for real-world scenarios [23].

Because the evaluation of each candidate solution is entirely independent, the Brute Force approach for this specific problem, falls into the category of embarrassingly parallel problems. As a result, the algorithm may benefit from the usage of Graphics Processing Units (GPUs). Evaluating binary states at the

same time reduces the execution time for small and medium-sized problems, but this does not change the number of computations. Ultimately, the exponential time complexity is still a limiting factor.

Brute Force remains an important tool despite these scalability limits. It serves to understand the problem's behavior and can be used to compare its exact solutions to other solvers' results on small-scale tasks.

A common method to obtain faster solutions from naive algorithms is to use heuristics or mathematical observations about the problem to eliminate unnecessary computations [23], which naturally leads to the advanced approaches discussed in the following sections.

3.2 Gurobi Optimizer

The Gurobi Optimizer is a mathematical solver implemented to manage QUBO problems with a few hundred elements. The algorithm returns an output in a reasonable time frame, but scales poorly as n increases. This method classifies the QUBO model as a Mixed-Integer Quadratic Programming (MIQP) problem. In our particular case, the problem is a non-convex MIQP for two reasons. The first is because the variables are strictly binary, which implies that the solution space is discrete and non-convex. Secondly, the diagonal terms of the Q matrix are positive while the off-diagonal ones are negative, making the matrix indefinite and non-convex.

Before starting the optimization process, Gurobi applies a Presolve algorithm that aims to simplify the model. Since each quadratic term contains at least one binary variable, the non-convex MIQP is converted into an equivalent convex MIQP, which is easier to solve.

To find the optimal solution, a different algorithm is applied to each model: for MIQP, Gurobi uses the branch-and-cut technique, which is a combination of branch-and-bound and cutting planes. Branch-and-bound is a systematic tree-search approach that partitions the search space and prunes sections through rigorous bounds. Cutting planes is a method that improves the mathematical formulation and saves useless computations. Gurobi tracks two metrics during its tree-search: the Incumbent, which is the best valid integer solution found so far, and the Best Bound (BestBd), which is the theoretical upper limit of the objective function calculated from the unexplored nodes. The optimal objective value necessarily falls between these two values. When the relative difference (MIP Gap) between the Incumbent and the Best Bound falls below the default tolerance of 0.01%, the optimization process terminates [12].

For the computational complexity, we must remember that QUBO is strictly an NP-hard problem. The complete search tree contains exactly 2^n possible leaves. The pruning techniques prevent the algorithm from evaluating every single solution, but the worst-case time complexity for Gurobi remains exponential, $\mathcal{O}(2^n)$. Indeed, the solver performs well on small problems, but starts to struggle as the number of variables increases.

3.3 Simulated Annealing

Exact methods cannot solve many large-scale combinatorial optimization problems since their time complexity rapidly scales. The main solutions to this issue are approximation or heuristic approaches, which do not always guarantee the best possible solution but can provide high-quality results in a reasonable amount of time.

Simulated Annealing (SA) is a heuristic algorithm for solving complex optimization problems that relies on probability and physical inspiration. The name derives from the analogy with the physical process of annealing in metallurgy. Indeed, minimizing a cost function and the slow cooling of a solid until it reaches its low-energy ground state are significantly correlated, so Kirkpatrick et al. introduced this algorithm in 1983. Since then, the study of this approach and its applications has developed into a vast field.

The algorithm is built using randomization techniques and iterative improvement methods. In physics, when a solid is maintained in thermal equilibrium at a constant temperature T, the probability of the system being in a state with energy E is given by the Boltzmann distribution:

$$Pr\{\text{Energy} = E\} = \frac{1}{Z(T)} \cdot \exp\left(-\frac{E}{k_B T} \right),$$

where $Z(T)$ is the partition function that depends on the temperature and k_B is the Boltzmann constant. From this formulation, one can observe that as T tends to zero, the system reaches its ground state, since only the minimum energy states have a non-zero probability.

This algorithm employs a Monte Carlo method to generate a sequence of states that simulates this thermal evolution. Starting from an initial configuration, the algorithm applies a small random perturbation that causes a change in energy, denoted as $\Delta E = E_{\text{final}} - E_{\text{initial}}$. If $\Delta E \leq 0$, the new state is automatically accepted. If the difference is positive, the worse state is not strictly rejected, but it is accepted with a probability computed using the Metropolis criterion: $e^{-\Delta E/(k_B T)}$. As the method iterates, the probability distribution of the states approaches the Boltzmann distribution.

In practice, the algorithm needs three main parameters. The process starts with a random bitstring as initial state S and a chosen initial temperature T_0. This temperature must be very high and it must allow the system to explore the global search space freely. So, the temperature value is chosen using a quick trial run on the energy landscape, aiming for an initial acceptance rate of around 80-90%. Secondly, the system needs a mechanism to reduce the temperature. The most common choice is the geometric cooling schedule, where the temperature is updated using the formula $T_{k+1} = \alpha T_k$. The cooling factor α is a constant smaller than but close to 1. The annealing ends when the stopping criterion is satisfied. This happens either when the temperature falls below a minimal threshold ($T_{min} \approx 0$) or when no better energy state is found through a massive number of iterations.

Another remark is that the algorithm might get unlucky and get stuck in a

local minimum. To fix this, the user has to choose a number of independent runs, called "reads". Then, the process is iterated this number of repetitions, generating a solution, and the one with the lowest overall energy is the final solution.

The general structure of a single read is outlined in Algorithm 2.

Algorithm 2 Simulated Annealing (Single Read)

Input: Initial temperature T_0, cooling factor α, minimum temperature T_{min}
Generate a random initial state S
$T \leftarrow T_0$
$E \leftarrow \text{Cost}(S)$
while $T > T_{min}$ **do**
    for $i = 1$ to MaxStepsAtEquilibrium **do**
        Generate a random neighbor state S'
        $\Delta E \leftarrow \text{Cost}(S') - E$
        if $\Delta E \leq 0$ **then**
            $S \leftarrow S'$
            $E \leftarrow E + \Delta E$
        else
            Generate a random number $r \in [0, 1)$
            if $r < \exp(-\Delta E / k_B T)$ **then** ▷ Metropolis Criterion
                $S \leftarrow S'$
                $E \leftarrow E + \Delta E$
            end if
        end if
    end for
    $T \leftarrow \alpha \cdot T$ ▷ Geometric Cooling
end while
return S, E

From a mathematical point of view, the transition to a new state depends only on the current state and the Metropolis probability; so this iterative process can be represented by a discrete-time Markov Chain. Consequently, the algorithm executes a sequence of these Markov Chains at gradually decreasing temperatures. The convergence theorem for Simulated Annealing states that, given a proper cooling schedule, this sequence asymptotically converges to a stationary distribution. In particular, as $T \to 0$, the probability becomes entirely concentrated on the set of global optima [25].

There is, however, a critical trade-off regarding its time complexity. In practical software implementations like the `neal` sampler, the algorithm defaults to a finite cooling schedule. This means a single read executes in polynomial time, returning a solution very quickly. Nevertheless, the mathematical convergence requires an infinite number of steps. If we want to maintain a high probability of finding the true minimum as the number of variables n grows, the overall computational effort must increase exponentially. We would need either an exponentially longer cooling schedule or an exponentially larger number of reads. For a detailed mathematical proof of the asymptotic convergence and a complete theoretical background, refer

to [25].

It is also important to underline that Simulated Annealing, like Brute Force and Gurobi Optimizer, is a sequential algorithm. Indeed, SA evaluates one transition at a time using classical CPU instructions, so its execution time heavily depends on the problem size. Researchers have tried to parallelize it on multi-core architectures [9, 11]. Despite these efforts, SA still relies heavily on classical sequential computation, making the search for the absolute optimum highly dependent on the problem size.

3.4 Quantum Annealing

Since classical computers still have problems on large-scale instances and complex optimization problems, the new frontier of Quantum Computing promise to overcome this issue using a completely different computational paradigm. The journey toward quantum computing began in the early 1980s, but it has seen significant advancements in recent years. Nowadays, two main different architectures exist: the gate-based quantum computers and the analog quantum machine, such as the quantum annealer used in this work. In gate-model computing the algorithms are expressed by means of quantum gates, analogous to Boolean gates in classical circuits. Quantum gates provide a universal model for quantum computation, on the other hand there are the special purpose quantum computers like quantum annealers. This kind of machines are designed to find the global minimum of specific objective functions for combinatorial optimization and probabilistic sampling tasks.

The fundamental building blocks of quantum computing are quantum bits, or qubits. Classical bits can only be in a state of 0 or 1, while qubits are quantum objects that can exist in a superposition of both states simultaneously. Mathematically, the states of a qubits are described as the projective rays in a two-dimensional Hilbert space, say $\mathbb{C}^2$. The dimension of the Hilbert space scales exponentially in the number of qubits, so a n-qubit system is described in $\mathbb{C}^{2^n}$. Furthermore, the system exploits entanglement, a purely quantum phenomenon encoding non-local correlations among the qubits.

In practical implementations, such as D-Wave's commercially accessible Quantum Processing Units (QPUs), quantum annealing is utilized as a heuristic that minimizes Ising objective functions, presented in Section 1.3. The QPU operates in total isolation from external influences at temperatures approaching absolute zero in order to maintain quantum properties. Nowadays, this architecture counts over 5000 physical qubits and has surpassed the current capabilities of gate-model quantum computing.

Essentially, solving a problem on a D-Wave QPU means translating the mathematical formulation directly into physical interactions. The user has to program specific qubit biases (for linear weights) and couplers (to control the correlation between paired qubits) that mimic the target objective function. The annealing process starts with completely independent qubits in a uniform superposition. In a matter of microseconds the physical process evolves and the hardware gradually introduces the problem-specific biases and couplers. The

system transitions through an entangled state of numerous possible answers until every qubit collapses into a classical deterministic state. Since physical systems naturally reach their lowest energy level, this final state corresponds to the global minimum of the landscape, yielding the optimal (or a highly optimal approximate) solution to the problem [7].

3.4.1 Basic Notions on Quantum Annealing

At this point, one may ask why the research community has introduced a quantum-based architecture if classical heuristics such as Simulated Annealing (SA) already exist. The answer, supported by both theoretical proofs and empirical results, is that QA outperforms SA in most combinatorial optimization scenarios [19]. As discussed in previous sections, SA relies on thermal (classical) fluctuations to let the system hop from one state to another, while QA takes a different approach. Rather than reducing the temperature, QA introduces artificial degrees of quantum mechanical freedom to induce quantum fluctuations so that the system finally reaches the ground state. The QA procedure is very similar to the SA cooling schedule. Initially, the strength of the quantum fluctuations is set very high, enabling the system to explore the global structure of the phase space (analogous to the high-temperature phase in SA). Then, this strength is gradually decreased until it vanishes, leaving the system in its optimal lowest-energy state.

Mathematically, the state vector evolution is governed by the time-dependent Schrödinger equation. The quantum system is described by a time-dependent Hamiltonian, an operator whose eigenvalues represent the energy levels. Given the total annealing time τ, we normalize the time parameter $s = t/\tau$ so that the annealing process runs from $s = 0$ to $s = 1$. The total Hamiltonian of the system is a linear combination of two terms:

$$\mathcal{H}(s) = A(s)\mathcal{H}_{\text{initial}} + B(s)\mathcal{H}_{\text{problem}},$$

where $A(s)$ and $B(s)$ are continuous functions. The initial state ($s = 0$) is chosen to be the ground state of the initial Hamiltonian ($A(0)$ is large and $B(0)$ is exactly zero). As s approaches 1, the coefficient $A(s)$ monotonically decreases to 0, while $B(s)$ grows, and the ground state of the Hamiltonian does not degenerate. This gradually turns off the initial quantum fluctuations while turning on the problem constraints.

Before proceeding with the formulation, we briefly recall the role of Pauli matrices as quantum operators. In quantum mechanics, physical observables are simply described by linear operators, which are mathematically represented by matrices acting on the qubit's state vector. For a single qubit, the operators are the three 2×2 Pauli matrices (σ_x, σ_y, and σ_z).

The initial Hamiltonian $\mathcal{H}_{\text{initial}}$, also called tunneling Hamiltonian, is typically chosen as a time-dependent transverse field applied to all qubits. Mathematically, it is expressed using the Pauli X matrix σ_x:

$$\mathcal{H}_{\text{initial}} = -\sum_{i=1}^{n} \sigma_x^{(i)},$$

where $\sigma_x^{(i)}$ is the operator acting as the Pauli matrix σ_x on the i-th qubit and as the identity on the other qubits.

Conversely, the problem Hamiltonian $\mathcal{H}_{\text{problem}}$, also known as final Hamiltonian, is written using the classical Ising model and the Pauli Z matrix σ_z:

$$\mathcal{H}_{\text{problem}} = -\sum_{i=1}^{n} J_i \sigma_z^{(i)} - \sum_{i<j} J_{ij} \sigma_z^{(i)} \sigma_z^{(j)}.$$

In this formula, J_i represents the linear bias applied to an individual qubit, while J_{ij} is the coupling strength between two connected qubits. This final state corresponds to the problem's QUBO matrix Q, as presented in Section 1.3.

This algorithm mathematically relies on the Quantum Adiabatic Theorem. It states that if a quantum system starts in the ground state of a time-dependent Hamiltonian and changes slowly enough, the system will stay in its instantaneous ground state. As the transverse field vanishes, the quantum superposition collapses, and the system naturally settles into the lowest energy state of $\mathcal{H}_{\text{problem}}$, mathematically revealing the global optimum [19].

3.4.2 D-Wave Architecture and Topologies

The most advanced Quantum Processing Unit (QPU) currently available on D-Wave is based on the Pegasus topology, where each qubit is physically coupled to 15 other qubits, representing a significant improvement on previous generations. The Advantage QPU, which uses this topology, features a 16×16 lattice of unit cells totaling 5,760 theoretical qubits. However, due to imperfections, it is expected that about 5% of them may be inactive.

Furthermore, quantum devices are subject to computational errors induced by thermal noise, flux bias drifts and decoherence. To mitigate these issues, D-Wave relies on statistical sampling: the QPU runs multiple annealing cycles (reads) to generate a distribution of solutions, which naturally mitigates the noise introduced by faulty qubits [22].

When working with these architectures, mapping a large dense QUBO matrix onto the physical hardware is a significant challenge due to the limited connectivity between qubits. In order to solve this issue, D-Wave has introduced the next-generation Zephyr topology, which can couple each qubit to 20 other qubits. This increased connectivity aim to reduce the embedding overhead, meaning fewer physical qubits are required to form chains for a single logical variable. Consequently, the hardware is able to efficiently solve much larger and denser optimization problems [3].

3.4.3 Hardware Limitations and Digital Annealing

The progress of quantum computing, initially sparked by milestones like the IBM Q System One and Google's Sycamore processor, has transitioned from demonstrating quantum supremacy to pursuing practical quantum utility. However, it is also important to underline the limitations of quantum annealers, since their applications are restricted due to the lack of control over individual qubits.

While in gate-based quantum computers qubits can be precisely manipulated via unitary operations to run any algorithm, D-Wave's annealers are not universal computers. Nevertheless, as already presented, they are highly specialized in running optimization problems encoded in QUBO or Ising model form.

Despite the theoretical advantages of physical quantum annealers, their maintenance presents extreme engineering challenges. To keep qubit stability and avoid quantum decoherence, these machines need extremely low temperatures and vast amounts of energy. These physical requirements and the aforementioned challenges on restricted connectivity often limit their scalability for large-scale industrial tasks in the real world.

To overcome these constraints, companies like Fujitsu have built Digital Annealers, which are advanced classical computing systems that mimic the behavior of quantum annealing. These devices use much less power and can optimize large-scale QUBO problems, making them more practical for many optimization and machine learning applications [5].

Nowadays, researchers aim to transition from theoretical results to practical applications. The first step is achieving "quantum utility", meaning that quantum devices become useful tools for relevant scientific problems. The ultimate goal, however, is to show "quantum advantage", which means demonstrating that a quantum computer can outperform classical computers by solving real-world problems significantly faster, with lower costs, or more accurately. Although this universal quantum advantage has not yet been achieved for all large-scale tasks, the development of physical quantum processors (QPUs) may bring a computational advantage in the future, for specific classes of problems. The rapid progress made in recent years suggests that, in the near future, annealing-based optimization may become a fundamental tool for solving complex combinatorial problems [22].

Chapter 4

Experiments

This chapter presents the methodology and the experimental setup used to test our QUBO-based bounding box suppression. We first define the testing environment, specifying the chosen dataset and the official object detection metrics. Then we directly move into the code implementation, from extracting the initial predictions to evaluating the spatial overlaps. The last section explains exactly how we tuned the hyperparameters to find the best possible penalty configurations.

4.1 COCO Dataset

Introduced in 2014, the Microsoft Common Objects in COntext (MS COCO) dataset is a large-scale benchmark for object detection, segmentation and captioning. The dataset primarily aimed to capture everyday scenes in their natural environments, with a particular focus on objects situated in the background or partially occluded by other elements. The resulting dataset is a collection of:

- Iconic-object images, which feature a single centered object;
- Iconic-scene images, which capture multiple objects from a canonical viewpoint;
- Non-iconic images, which display different objects in a contextual scenario.

Experts had shown that datasets including a majority of non-iconic images are better suited to training models that generalize well in real-world scenarios. This inherent complexity is what makes the COCO dataset particularly adopted in modern object detection.

This work uses the COCO 2017 release, in which the dataset is split into 118,287 training, 5,000 validation, and 40,670 testing images. While COCO 2017 encompasses 80 common object categories ranging from animals (“dog”, “cow”, etc.) to vehicles (“car”, “train”, etc.), this project focuses exclusively on the “person” category. People detection is actually the most significant and recurrent task in computer vision. It is an evolution of the classical “pedestrian” detection and has many applications in security, autonomous driving and human-computer interaction [18].

For the purposes of this study, the 5,000-image validation set proved to be sufficiently large and representative to be used both for hyperparameter tuning and for evaluating the final performance of the four proposed solution approaches.

4.1.1 Mean Average Precision and Mean Average Recall in COCO

As discussed in Section 2.3.2, mAP (mean Average Precision) is typically defined as the average of the AP scores computed for each individual class. However, in the COCO benchmark, there is no structural difference between AP and mAP. As stated in the official COCO guidelines: "AP is averaged over all categories. Traditionally, this is called mean average precision (mAP). We make no distinction between AP and mAP (and likewise AR and mAR) and assume the difference is clear from context".

Furthermore, the COCO evaluation protocol calculates AP more rigorously than previous benchmarks. For instance, while the Pascal VOC[1] challenge calculated AP by interpolating the Precision-Recall curve at 11 points, the COCO metric utilizes a 101-point interpolation.

More importantly, COCO AP_{std} is not evaluated at a single Intersection over Union (IoU) threshold (typically 0.5). Instead, it is denoted as AP@[.5:.95] and corresponds to the average of multiple IoU thresholds ranging from 0.5 to 0.95 with a step size of 0.05. So for the COCO dataset, this method is much stricter and it is considered the standard COCO metric. Mathematically, it is computed as follows:

$$AP_{std} = \frac{AP_{0.5} + AP_{0.55} + \cdots + AP_{0.95}}{10}.$$

While traditional evaluation metrics prioritized simply identifying the object, this rigorous averaging heavily rewards models that can localize the target with extreme, pixel-level accuracy.

Similarly, the Average Recall (AR) metric is calculated using the same multiple IoU thresholds to reward precise localization. Moreover, COCO sets a maximum number of detections considered per image. The standard metric, AR@100, evaluates recall by considering only the 100 highest-confidence detections per image [13].

For a more comprehensive overview of all 12 evaluation metrics officially provided by the COCO benchmark, refer to Figure 4.1.

4.2 Experimental Setup

This QUBO-based research can be divided into three main steps. The preprocessing phase consists of extracting the candidate bounding boxes from the analyzed images. After that, we present the mathematical core of our work. The QUBO construction is explained in detail, showing how it adapts to the specific penalty cases and the different optimization solvers. The final phase is the accuracy evaluation of the surviving boxes through a comparison with the Ground Truths.

The Python code developed for this study is publicly available on GitHub[2].

[1] PASCAL Visual Object Classes (VOC) is a renowned dataset and benchmark suite that has significantly contributed to the advancement of computer vision research.

[2] Link to the GitHub repository.

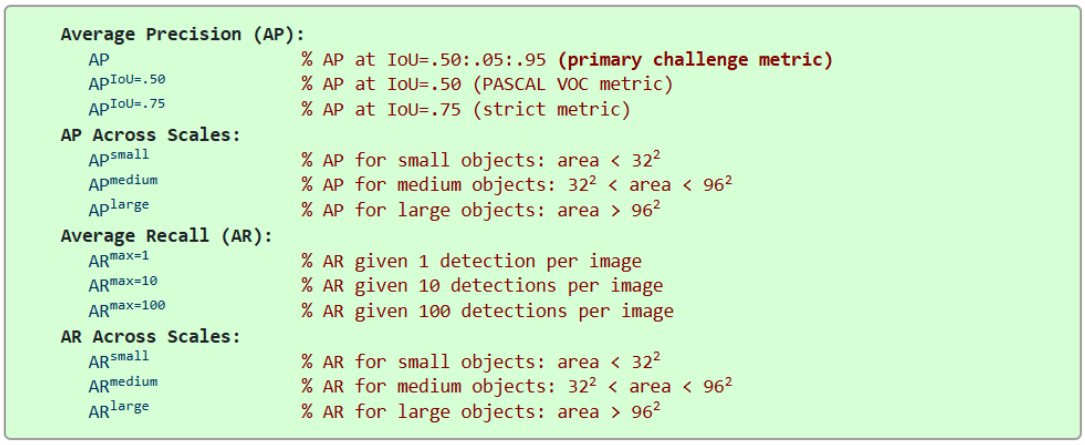


Figure 4.1: The following 12 metrics are used for characterizing the performance of an object detector on COCO. Source: [4].

4.2.1 Pre-processing: Faster R-CNN

The first step of our experimental pipeline was to extract from the target images every candidate bounding box together with the respective confidence score. To accomplish this, we selected Faster R-CNN, one of the most well-established and reliable deep learning architectures for object detection.

Earlier models, such as R-CNN and Fast R-CNN, relied on slow algorithms to generate region proposals, creating a significant computational bottleneck. Faster R-CNN [24] solved this issue by introducing a unified architecture where the entire image is processed in a single forward pass. It incorporates a fully convolutional Region Proposal Network (RPN) that works with the Fast R-CNN detector. By allowing the RPN to share convolutional features with the detection network, Faster R-CNN achieved near real-time inference speeds while maintaining high detection accuracy.

Following the recent methodology for QUBO-based suppression tasks [26], our box extraction was performed using a two-stage Faster R-CNN model pre-trained with ResNet-50 backbone and a Feature Pyramid Network (FPN) [17]. The model was instantiated via the PyTorch Torchvision library. In order to guarantee an optimal extraction, we used the default weights that were specifically pre-trained on the COCO dataset.

At this stage, a crucial modification was applied. Default Faster R-CNN employs the Non-Maximum Suppression (NMS) algorithm, already presented in

Section 1.4, to discard redundant bounding boxes. However, as the main aim of this study is to replace NMS with a QUBO-based mathematical optimization, we disabled this mechanism by applying an NMS threshold of 0.9 in order to obtain highly overlapping bounding boxes. Furthermore, since we are focusing on people detection, a minimum confidence score threshold of 0.60 was applied, ensuring that only detections with at least a 60% probability of containing a person were returned.

During the execution on the COCO validation set, the model naturally outputs bounding boxes in the coordinate format $[x_1, y_1, x_2, y_2]$, representing the top-left and bottom-right corners[3]. To ensure perfect compatibility with standard COCO ground truth annotations, we immediately converted these predictions into the $[x, y, w, h]$ format (top-left coordinates, width and height).

Finally, once we obtained all bounding boxes for each image, we applied a last filtering step to the dataset in order to keep only the images with at least 2 boxes and at most 90. The first limit was imposed because a single box requires no suppression, while the second was chosen because an excessive number of boxes would generate intractably large QUBO problems. The remaining valid images were then partitioned into three complexity classes based on the number of bounding boxes: Low (2–20 boxes), Medium (21–60 boxes) and High (61–90 boxes).

To accelerate this pre-processing phase, the Faster R-CNN architecture can be executed on a Graphics Processing Unit (GPU). This hardware parallelization reduces the bounding box extraction time by approximately 94% per single image (dropping from roughly 0.90 seconds on a standard CPU down to 0.05 seconds on a GPU).

4.2.2 Post-processing: QUBO-based suppression

Once we had extracted the candidate bounding boxes, the next task was to translate these predictions into the mathematical QUBO formulation described in Section 1.4.

Let n be the total number of detected bounding boxes for each selected image. We first initialized two $n \times n$ dense matrices using 64-bit floating-point precision to ensure high numerical stability and prevent rounding errors. These two matrices are the confidence matrix L and the penalty matrix P, required to construct the objective function. In order to optimize the overall computational time, the algorithm computed these two matrices only once per image, independently of the hyperparameters.

The i-th diagonal term of the L matrix was set to the confidence score c_i of the i-th bounding box, while other elements remained zero. Conversely, the P matrix was used to encode the spatial overlap between pairs of detections. These intersections between boxes were evaluated using the penalties (IoU, IoM and Spatial Feature) presented in Section 2.1, which, in turn, were arranged following the combinations shown in Section 2.2. Once the specific case was set,

[3]In Computer Vision, the standard convention is that the x-axis goes from left to right and the y-axis from top to bottom.

the algorithm iterated over all distinct pairs (i, j) with $j > i$. The IoU, IoM and Spatial Feature algorithms computed the intersection area, union area, and other properties directly using the box coordinates $[x, y, w, h]$. Since the overlap is symmetric, we assigned the computed penalty score to both the p_{ij} and p_{ji} terms, while keeping the main diagonal of P at zero to avoid self-penalization.

Once L and P had been populated, we assembled the final unified QUBO matrix Q by applying the non-negative hyperparameters α and β. For Case 1 and Case 2, the matrix was computed as $Q = \alpha L - \beta P$. For the comprehensive configurations (Case 3 and Case 4), the computation became $Q = \alpha L - (\beta_1 P_1 + \beta_2 P_2)$.

To prevent numerical instability and floating-point artifacts from affecting the strict tolerance criteria of the solvers, we rounded the resulting Q matrix uniformly to six decimal places.

At this point, a fundamental mathematical adjustment was required. As defined in Section 1.4, our QUBO formulation models a maximization problem. The Brute Force algorithm was explicitly programmed to search for the maximum, following the Algorithm 1, and within the Gurobi Optimizer one can chose the optimization direction to maximization. In energy-based samplers like Simulated Annealing and Quantum Annealing, the approaches are designed to minimize the system's energy. We easily solved this issue by inverting the sign of the Q matrix, since $\max(x^T Q x)$ is mathematically equivalent to $\min(-x^T Q x)$.

Furthermore, we needed to adapt the data structure of the Q matrix to satisfy the specific input requirements of the four different solvers presented in Chapter 3.

The Brute Force and Gurobi Optimizer algorithms natively support matrix-based algebraic operations. Therefore, we passed the dense NumPy matrix Q directly to the solvers without any structural modification.

For the classical heuristic approach, we used the `neal` Simulated Annealing sampler [6], which runs locally on the classical CPU. Although energy-based samplers cannot process dense matrices directly, the `neal` solver accepts symmetric inputs, so we converted the dense matrix into a dictionary mapping the variable pairs (i, j) to their respective scores.

For Quantum Annealing (QA), the requirements of D-Wave's physical QPUs are stricter than those for SA. The hardware topology strictly requires an upper-triangular dictionary, so we algorithmically converted the matrix by following the mathematical property introduced in Section 1.1. Specifically, the general QUBO formulation $x^T Q x$ became compatible with the Ising-based quantum hardware:

$$\sum_{i=1}^{n} q_{ii} x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (q_{ij} + q_{ji}) x_i x_j.$$

In practice, for every $i < j$ we summed the symmetric elements $q_{ij} + q_{ji}$ and mapped them to the (i, j) key in the dictionary, removing all lower-triangular (j, i) entries.

Once the QUBO problem had been formally defined and correctly formatted, it was passed to the respective solver. Solvers like Brute Force and Gurobi Optimizer return a NumPy array, but the heuristic and quantum samplers return a dictionary containing the lowest-energy state. To make sure that the solutions were fully

comparable across all methods, we converted these outputs into a standard binary vector $x \in \{0,1\}^n$. This unified format made the results easy to interpret: the elements equal to 1 identified the indices of the bounding boxes that were kept, effectively suppressing the redundant detections.

4.2.3 Ground Truths extraction and accuracy evaluation

Once the solver generated the solutions, the final step was to evaluate the quality of the "survived" bounding boxes using the metrics presented in Section 2.3. These boxes were compared to the Ground Truths provided by the COCO dataset.

For each image, we extracted the corresponding Ground Truth annotations from a large JSON file using the specific `image_id`. Since this study focuses entirely on human detection, we only retained annotations where the category ID corresponded to the "person" class. To ensure perfect compatibility with the subsequent evaluation functions, we stored these true boxes in the standard $[x, y, w, h]$ format and grouped them into a dictionary, using the `image_id` as the primary key.

After the Ground Truths extraction, we proceeded to the accuracy evaluation. The evaluation metrics assess the performance of the four penalty configurations (Case 1 to Case 4) across different solvers. All the metrics computed by our scripts are: Precision, Recall, F1-score, standard mAP, mAP(IoU=0.5), mAR(10), mAR(100), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE).

The first three metrics were evaluated with a custom Python script after calculating TP and FP between the Ground Truths and the predicted bounding boxes for each image.

For the official COCO standard metrics, we formatted our solver's predictions into a list of dictionaries and loaded them into a `COCOeval` object, restricting the evaluation exclusively to the "person" class. Then, following the order presented in Figure 4.1, we extracted the standard mean Average Precision (mAP_{std}) and the mean Average Precision at 50% overlap ($\text{mAP}_{0.5}$). We also saved the recall capabilities through the mAR_{10} and mAR_{100} metrics, which represent the maximum recall given 10 and 100 predictions per image.

Finally, we looked at the counting errors to see how well the algorithm guessed the actual number of people in the scene. We calculated the MAE and the RMSE by comparing the number of bounding boxes kept by our solver against the Ground Truth count.

In our scripts we analyzed both single images and groups of images. For a batch of images, the output scores for each metric represent the average computed across the entire subset. Conversely, when we analyze a single image, the metrics reflect that specific instance. In this particular case, calculating MAE and RMSE only consists of computing the absolute value of the difference, so they are mathematically equivalent.

4.3 Hyperparameter tuning

In order to achieve the best object detection results, we had to balance the linear weights and the quadratic penalties in our QUBO formulation. We recall that the hyperparameter α controls the confidence matrix, while β controls the penalty matrix.

The validation subset initially consisted of 300 images, which became 291 after the pre-processing step described in Section 4.2.1. Specifically, this final batch is partitioned into 192 Low-density, 88 Medium-density and 11 High-density instances. We simply selected the first valid images and obtained an unbalanced distribution, rather than artificially forcing an equal split (such as an unrealistic division of roughly 100 images per class). This setting was chosen in order to follow the original statistical arrangement of the COCO dataset. Since the goal of object detection is to recover all objects in a scene, mixed frames featuring only a few people alongside other objects are statistically much more frequent than environments exclusively crowded with humans. In conclusion, this natural distribution perfectly reflects the conditions encountered in real-world applications.

Once the subset was established, we implemented a grid search evaluating different α values ranging from 0.50 to 0.70 with a step size of 0.02. Since the hyperparameters must sum to 1, given α, one can trivially obtain β as $1 - \alpha$.

However, Case 3 and Case 4 introduce a second penalty matrix and the β hyperparameter splits into β_1 and β_2. To maintain the exact same overall penalty weight without arbitrarily favoring the IoU overlap over the spatial distance, we set $\beta_1 = \beta_2 = (1 - \alpha)/2$.

To find the optimal configuration, we tracked the mAP_{std} across the grid search, selecting the specific α that maximized this score for each case.

We executed this entire process on the same subset using the two classical solvers: the Gurobi Optimizer and Simulated Annealing. We chose Gurobi because it represents the state-of-the-art and is widely adopted in industrial benchmarks. On the other hand, we decided to run the same grid search using SA because its underlying heuristic mechanism is very similar to QA. Indeed, once we computed the optimal α values via SA, we obtained a highly compatible parameter baseline to test our formulation also on the quantum hardware. In fact, performing an exhaustive grid search directly on the D-Wave QPU is currently impossible, as submitting thousands of consecutive evaluation tasks to the physical quantum annealer would incur completely unsustainable financial costs.

As discussed in Section 4.2.1, the pre-processing phase benefits from GPU parallelization, particularly when analyzing a large batch such as this validation subset. Nevertheless, both Gurobi and Simulated Annealing are intrinsically sequential algorithms, so we do not obtain advantages using a GPU. If one transfers data back and forth to run the pre-processing step on a GPU and the post-processing on a CPU, a massive bottleneck is created. To avoid this, the entire code must be executed on the same hardware.

Our experiments were conducted on both CPU and GPU[4] and confirmed this

[4]All GPU experiments were executed on an NVIDIA H100 PCIe GPU equipped with 80 GB of VRAM, utilizing CUDA version 12.5.

behavior. Consequently, the results presented in Chapter 5 were all obtained using a standard CPU, except for the analysis on the limits of the Brute Force algorithm.

Once the optimal hyperparameters were chosen, the methodological setup was complete, allowing us to proceed to the final evaluation phase.

Chapter 5
Results and Discussions

This chapter presents the experimental results of our QUBO-based bounding box suppression. It is divided into three main phases. First, we explore the hyperparameter tuning process to find the optimal spatial penalty configurations for our formulations. Once we have found the best α configurations, we establish the exact mathematical baseline using the Brute Force algorithm, showing its accuracy and computational limits on both CPU and GPU architectures. Finally, we can start with an analysis of single images to evaluate the behavior of the Gurobi Optimizer, Simulated Annealing, and the Quantum Annealer, then scaling up to compare their macroscopic performance across the validation subset.

5.1 Hyperparameters Tuning and Best Configuration

As presented in Section 4.3, the initial phase was characterized by the search for optimal hyperparameters that weight penalties in our QUBO matrices. We recall that the validation subset included 291 images partitioned into three density classes based on the number of bounding boxes: 192 Low (2–20 boxes), 88 Medium (21–60 boxes) and 11 High (61–90 boxes).

An exhaustive grid search was conducted using both the Gurobi Optimizer and the Simulated Annealing solvers. The α values found by Gurobi for each case are exactly the same as those found by SA. This identical behavior allowed us to establish a highly reliable configuration, which will be used to benchmark the Quantum Annealer.

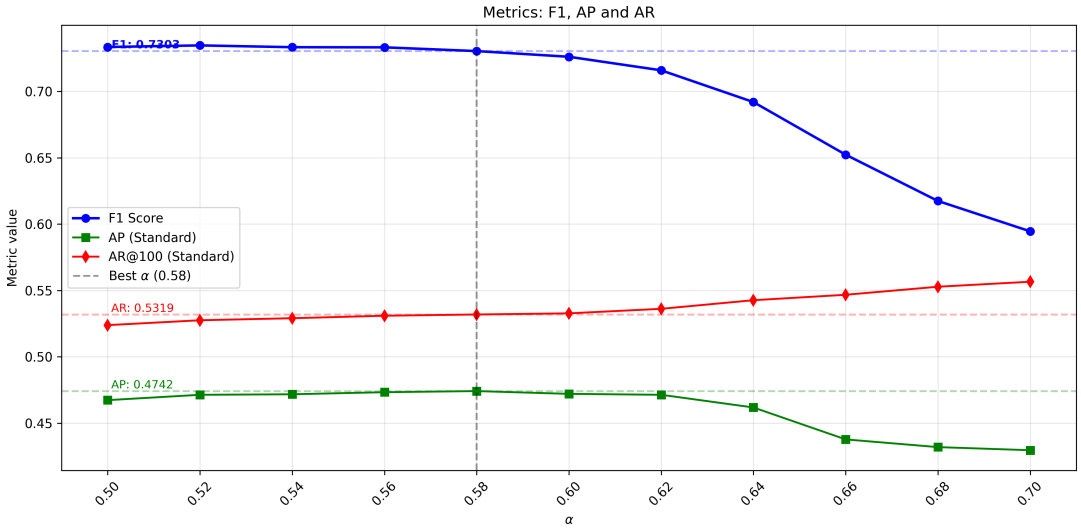
5.1.1 Metrics Evaluation across Penalty Cases

We evaluated the performance of the four spatial penalty configurations by tracking the standard mean Average Precision (mAP_{std} or AP), the mean Average Recall at 100 predictions (mAR_{100} or AR@100), and the F1-score across $\alpha \in \{0.50, 0.52, \ldots, 0.70\}$. The goal was to obtain the α (and the resulting β or β_1 and β_2) that maximizes detection accuracy for each case.

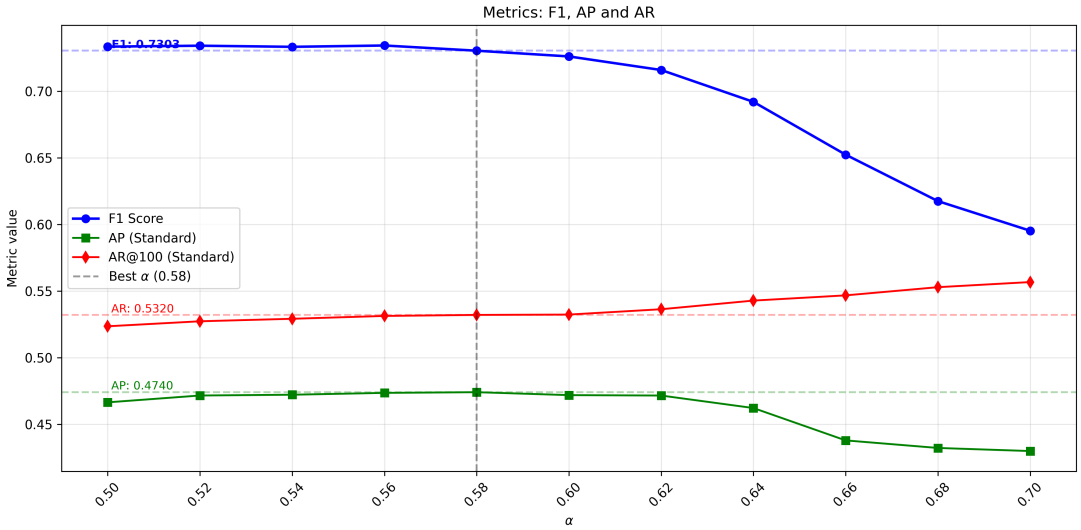
Looking at the graphs, several conclusions can be drawn. The baseline Case 1 (Figure 5.1) reaches the highest absolute AP peak among all configurations, hitting 0.4742 for Gurobi (Figure 5.1a) and 0.4740 for Simulated Annealing (Figure 5.1b) at $\alpha = 0.58$. However, this peak is misaligned with the F1-score, which reaches its maximum at $\alpha = 0.52$ for both methods. If one looks at the combined penalty configurations in Cases 2, 3, and 4 (Figures 5.2, 5.3, and 5.4), the maximum AP perfectly aligns with the maximum F1-score at $\alpha = 0.60$ (for Cases 2 and 3) and $\alpha = 0.62$ (for Case 4). This simultaneous maximization is a huge point in favor of

these combined approaches. In conclusion, we demonstrated that mixing different spatial penalties provides greater robustness and a highly credible balance between Precision and Recall for real-world scenarios.

Furthermore, we underline that the AR@100 score always increases as α grows. This metric only depends on Recall, which naturally increases when more boxes survive the suppression process. In particular, when α is high, β is small, which implies more weight for the confidence matrix rather than for the penalty matrix, thereby saving more boxes.

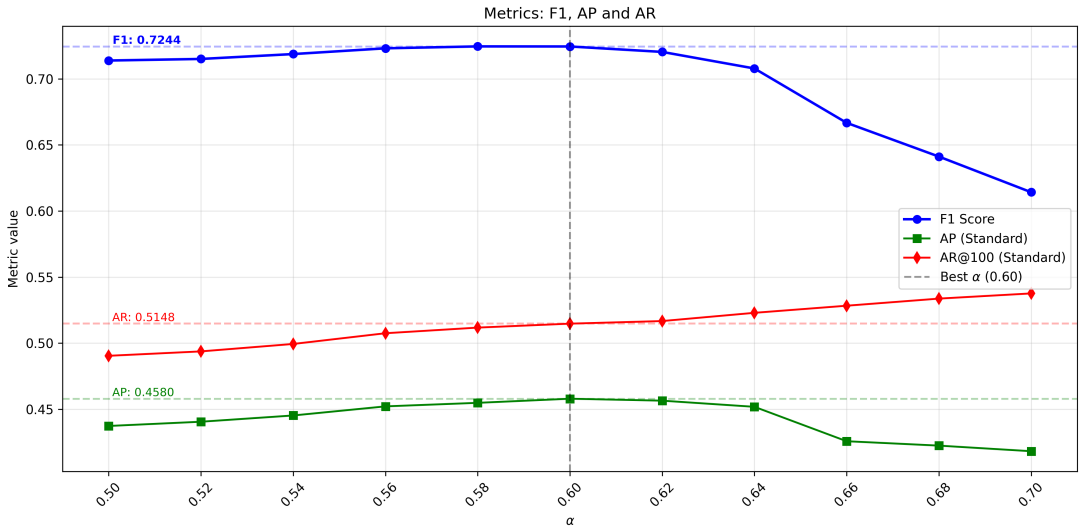


(a) Gurobi evaluation for Case 1 (Baseline) as α grows.

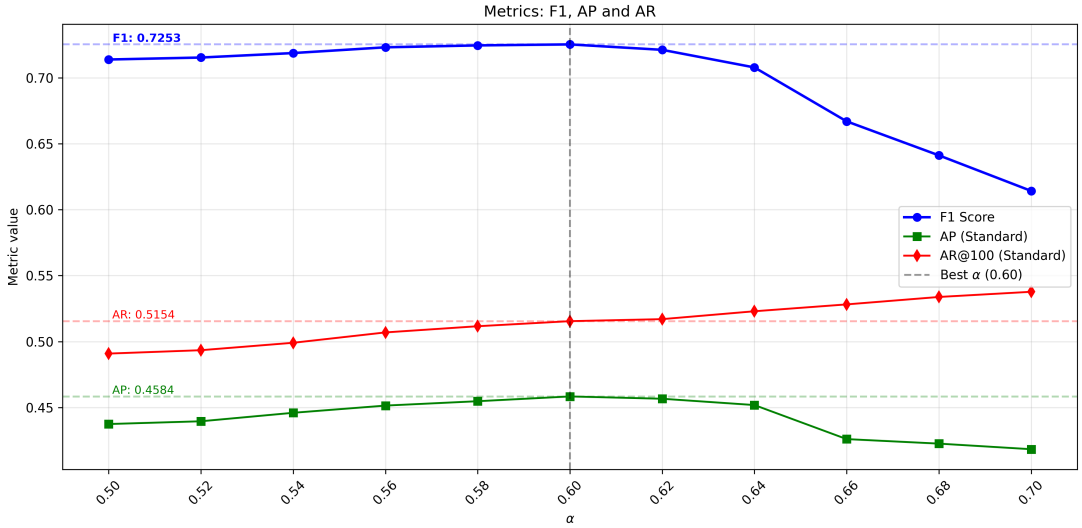


(b) Simulated Annealing evaluation for Case 1 (Baseline) as α grows.

Figure 5.1: Evaluation of detection metrics (mAP_{std}, mAR_{100}, and F1-score) for the baseline Case 1 across the hyperparameter α grid. The dashed vertical line indicates the selected $\alpha = 0.58$, which maximizes the AP score. Since this baseline relies solely on the standard Intersection over Union (IoU), we can see a clear misalignment between the AP peak and the maximum F1-score.

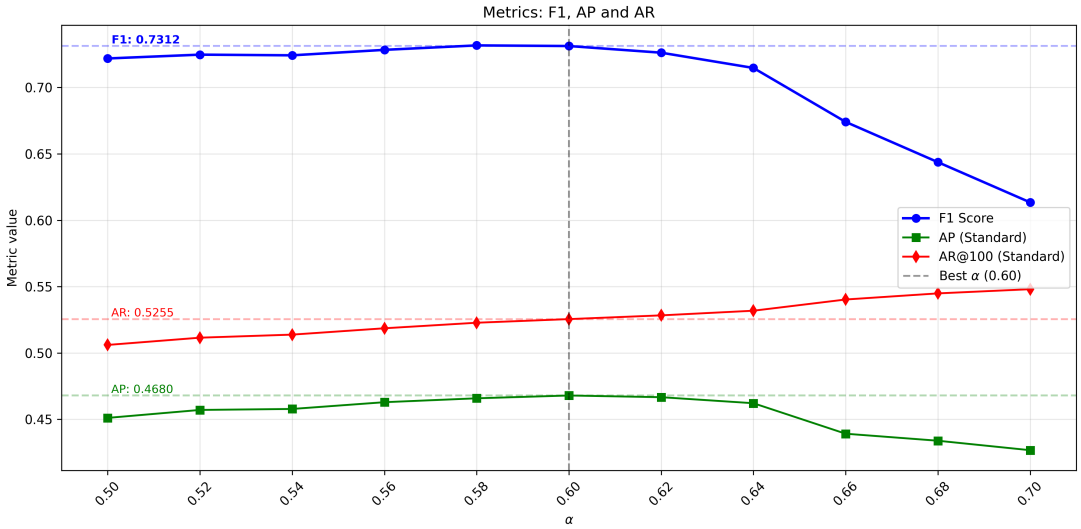


(a) Gurobi evaluation for Case 2 (Mixed IoU-IoM) as α grows.

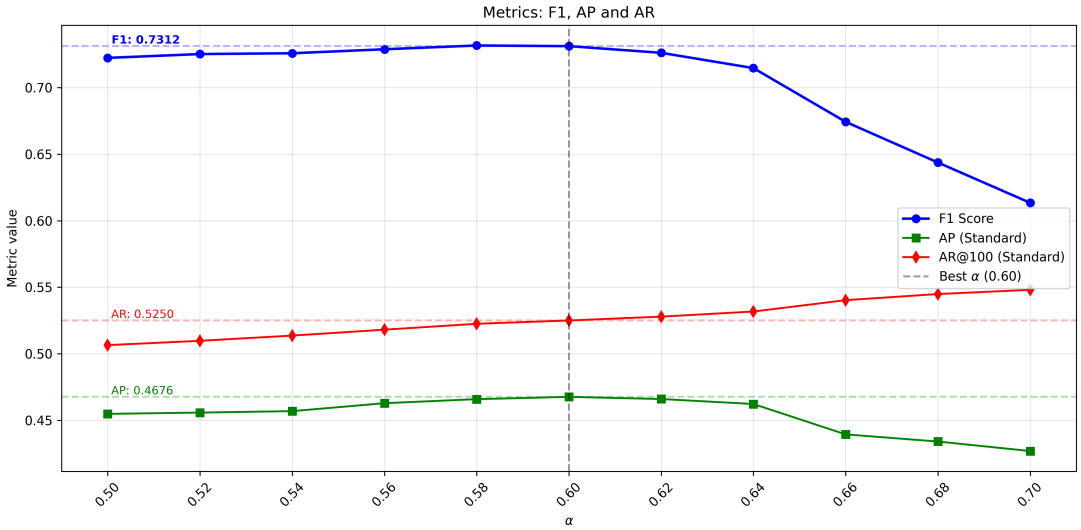


(b) Simulated Annealing evaluation for Case 2 (Mixed IoU-IoM) as α grows.

Figure 5.2: Evaluation of detection metrics (mAP_{std}, mAR_{100}, and F1-score) for Case 2 across the hyperparameter α grid. The dashed vertical line indicates the selected $\alpha = 0.60$. Compared to the baseline, the addition of the Intersection over Minimum penalty shifts the maximum AP to perfectly match the F1-score peak. This proves that the mixed formulation provides a much better balance between precision and recall.

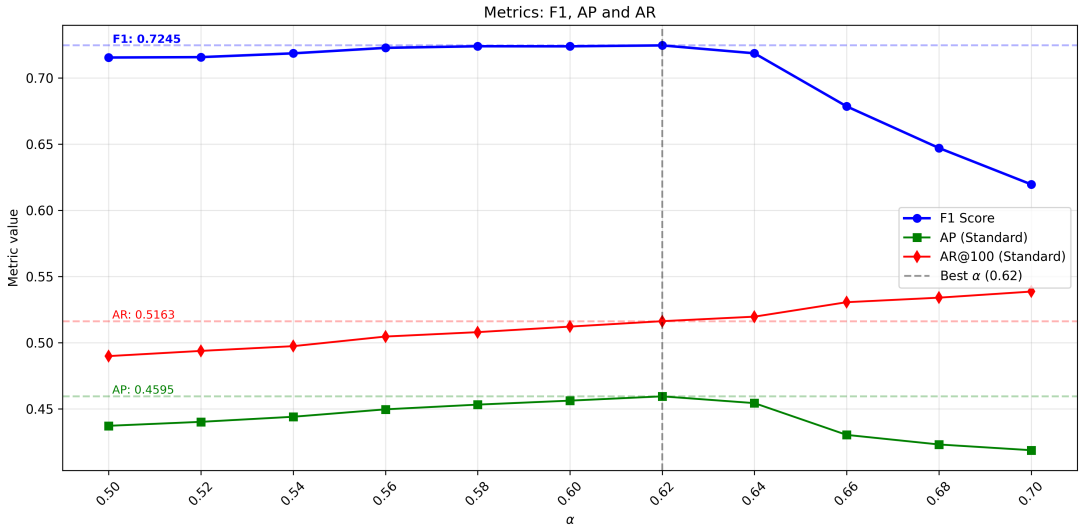


(a) Gurobi evaluation for Case 3 (IoU and Spatial Feature) as α grows.

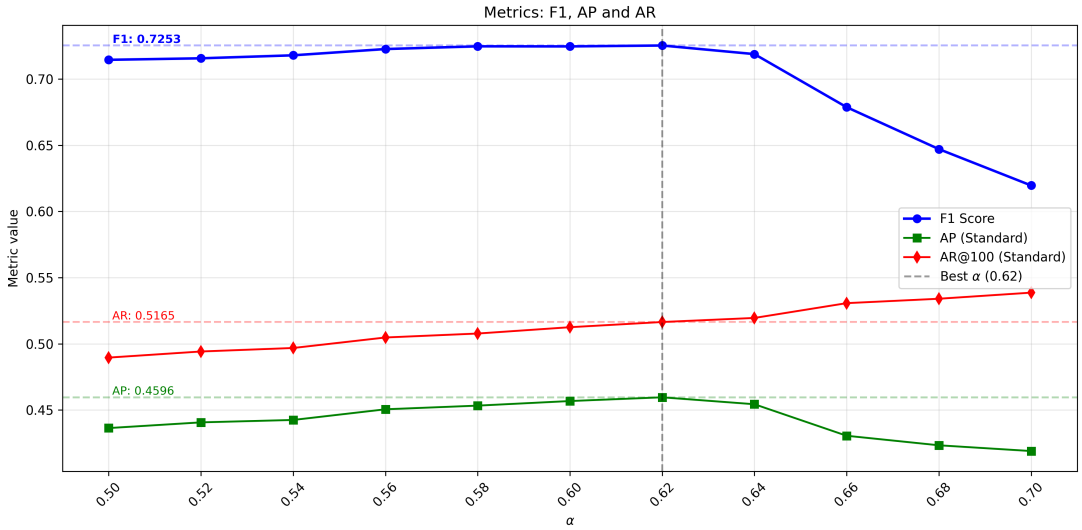


(b) Simulated Annealing evaluation for Case 3 (IoU and Spatial Feature) as α grows.

Figure 5.3: Evaluation of detection metrics (mAP_{std}, mAR_{100}, and F1-score) for Case 3 across the hyperparameter α grid. The optimal $\alpha = 0.60$ is marked by the dashed line. Just like in Case 2, combining the standard IoU with the spatial distance feature successfully aligns the highest AP score with the maximum F1-score.



(a) Gurobi evaluation for Case 4 (Comprehensive) as α grows.



(b) Simulated Annealing evaluation for Case 4 (Comprehensive) as α grows.

Figure 5.4: Evaluation of detection metrics (mAP_{std}, mAR_{100}, and F1-score) for the comprehensive Case 4 across the hyperparameter α grid. The dashed line points out the optimal $\alpha = 0.62$. Mixing all features together (IoU, IoM, and spatial distance) confirms the simultaneous maximization of AP and F1-score, proving once again the robustness of combined spatial penalties.

To summarize the outcomes of the tuning phase, the final Q matrices for each case are the following:

- **Case 1 (Baseline)**: $Q = 0.58L - 0.42P$, where $P = \text{IoU}$;
- **Case 2 (Mixed IoU-IoM)**: $Q = 0.60L - 0.40P$, such that $P = 0.7 \cdot \text{IoU} + 0.3 \cdot \text{IoM}$;
- **Case 3 (IoU and Spatial Feature)**: $Q = 0.60L - (0.20P_1 + 0.20P_2)$, where $P_1 = \text{IoU}$ and $P_2 = \text{Spatial Feature}$;
- **Case 4 (Comprehensive)**: $Q = 0.62L - (0.19P_1 + 0.19P_2)$, where $P_1 = 0.7 \cdot \text{IoU} + 0.3 \cdot \text{IoM}$ and $P_2 = \text{Spatial Feature}$.

During this process, we also evaluated mAP scores across different image densities using both solvers. As expected, images characterized by a Low-density of bounding boxes yielded better overall mAP scores compared to High-density scenes, simply because fewer overlaps imply fewer chances for suppression errors. Nevertheless, the absolute mAP score is not as high as one may expect, but peaks around 0.54. This follows from the fact that the standard metric provided by the COCO dataset is extremely restrictive: the multiple IoU thresholds heavily penalize detections if their overlap with the Ground Truth does not reach the 0.95 score.

As we move to Medium and High density images, the mAP scores naturally decrease. However, the worst overall performance was recorded in the Medium density class, showing a 2-4% drop with respect to the High density score. This behavior is counterintuitive, but it is likely dependent on the specific spatial distribution of the images. We must recall that the High-density class contains only 11 instances, while the Medium one includes 88 images. If those 11 exceptionally crowded images happen to be "simpler" cases, presenting more sparse matrices with easily separable clusters, they can easily raise the average mAP for that entire category. Conversely, if the bounding boxes in the Medium density batch happen to be exceptionally tricky or heavily overlapped, this mAP behavior makes sense. For this reason, we also conducted a deeper analysis of single images to better understand the macroscopic metric behavior. These detailed density results will be presented visually later in Section 5.3, alongside the Quantum Annealer results.

5.1.2 Density Analysis and Execution Times

During the hyperparameter tuning phase, we also tracked the execution times for both Gurobi and Simulated Annealing. Figures 5.5, 5.6, 5.7, and 5.8 illustrate these measurements for each of the four penalty configurations. The plots display both the average execution time per image divided by density class (Low, Medium, High) and the total execution time required to process the entire validation subset of 291 images.

As we expected from the background theory, execution time scales up for both methods as the number of bounding boxes increases. However, the two

classical solvers handle this complexity quite differently depending on the specific case. For a Low-density image, Gurobi takes around 0.015 seconds to output the solution, which is actually extremely fast. As the solver moves to the High-density class, the time naturally increases, peaking around 0.33-0.36 seconds per image at the optimal α configurations. Gurobi's time complexity for solving a QUBO problem is theoretically exponential, but we showed that the branch-and-cut technique manages to keep the execution time acceptable at this scale. Evidently, a maximum of 90 boxes is still below the critical threshold where the exponential growth drastically explodes. Consequently, the total execution time for the entire batch remains extremely low, ranging between 13.80 and 15.14 seconds depending on the case.

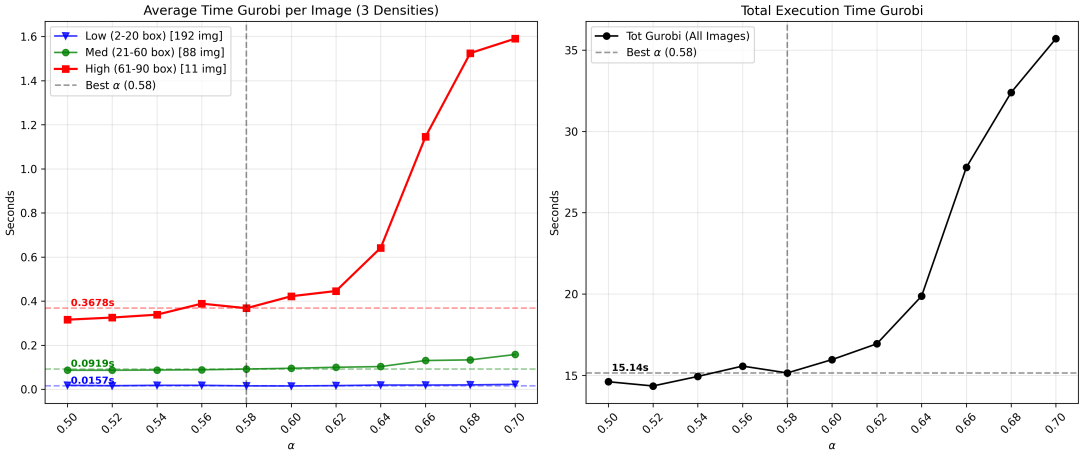
On the other hand, the Simulated Annealing heuristic is drastically slower. The average time required to process High-density images hovers around a full 1.0 second, while even Low-density scenes take about 0.16 seconds, which is ten times longer than Gurobi. This significant difference is impactful in the total execution times, which reach a peak of 89 seconds for the whole dataset at the optimal α configurations.

Beyond the solver comparison, the graphs reveal an interesting behavior depending on the α hyperparameter. For both Gurobi and SA, the execution time visibly spikes as α approaches 0.70. This happens because α and β are strictly correlated: the bigger the first, the smaller the second. Consequently, higher values of α progressively reduce the weight of the penalty matrix. If the optimization heavily discounts spatial penalties, the solver is forced to make decisions relying almost entirely on the linear confidence scores. In practical terms, the suppression logic degrades into a poorly constrained selection, effectively acting like NMS. This mathematical ambiguity forces the solvers to explore a much larger set of possible combinations before converging.

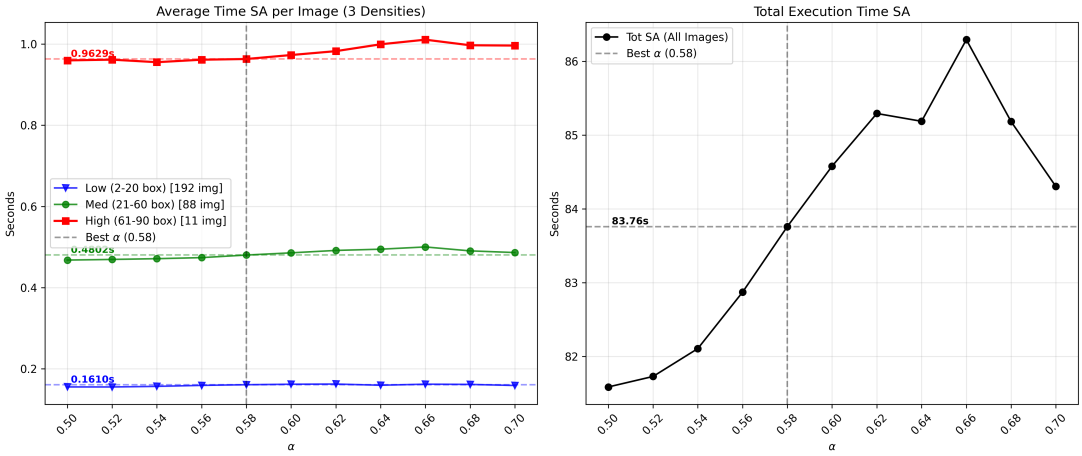
Furthermore, we must remember that the QUBO matrix is symmetric, meaning the quadratic penalties are added twice during the energy calculation. Because of this double-counting, an optimal α near 0.50 (which would theoretically imply a 50/50 split between confidence and penalty) was already unlikely. The empirical results confirm our expectations, showing that the perfect balance is when α ranges from 0.58 to 0.62.

Finally, comparing the execution times across the different penalty formulations yields an unexpected insight. For Gurobi, the baseline Case 1 (Figure 5.5a) is actually the slowest configuration, while Cases 2, 3, and 4 (Figures 5.6a, 5.7a, 5.8a) are resolved more quickly. Even though the mathematical density of the Q matrix remains identical across all scenarios, introducing the Intersection over Minimum and the Spatial Feature increases the absolute numerical values of the off-diagonal weights. This creates a more defined objective function landscape, which helps Gurobi's exact algorithm to easily prune branches during the search and to converge more rapidly. Conversely, using SA we obtained the exact opposite behavior, as Case 1 (Figure 5.5b) turns out to be the fastest run. We recall that SA relies on thermal fluctuations and probabilistic heuristic search, so different spatial metrics employed in Cases 2, 3 and 4 (Figures 5.6b, 5.7b, 5.8b) make the energy landscape more complex. This introduces deeper local minima, making

the thermal exploration slightly more time-consuming compared to the smoother landscape of the plain IoU approach.

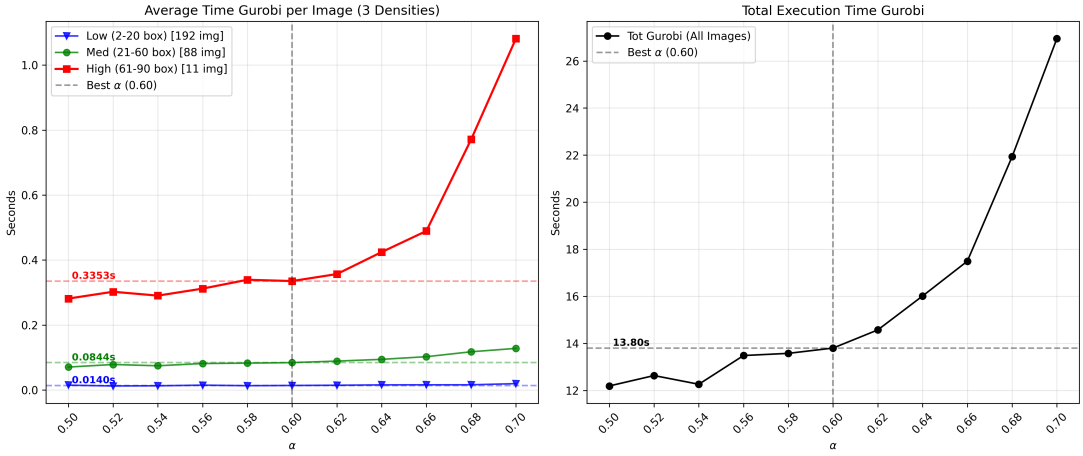


(a) Gurobi execution times for Case 1 (Baseline) as α grows.

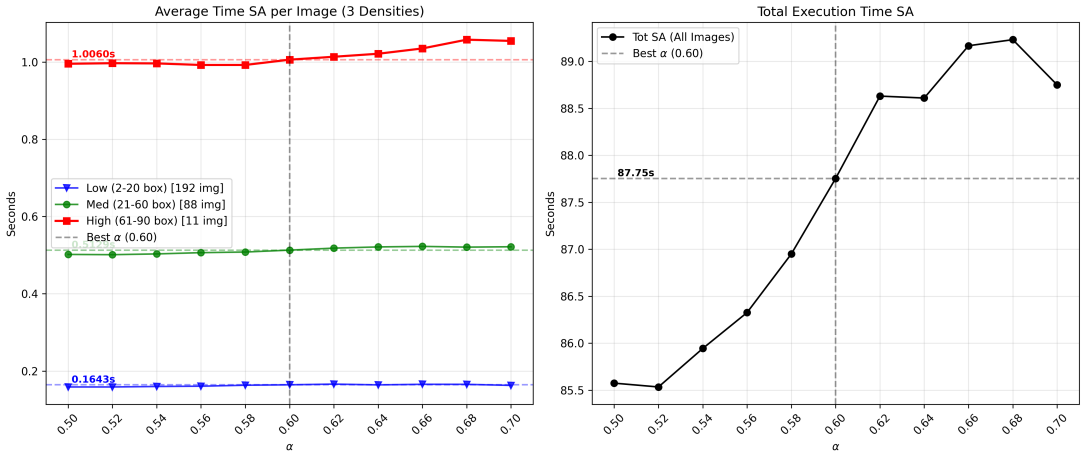


(b) Simulated Annealing execution times for Case 1 (Baseline) as α grows.

Figure 5.5: Average per-image and total execution times for Case 1 across the hyperparameter α grid. The left graphs display the average times split by bounding box density, while the right ones show the total execution time for the whole subset. As we discussed in the text, the execution time visibly spikes for both solvers when α approaches 0.70, since the penalty constraints become too weak to guide the optimization efficiently.

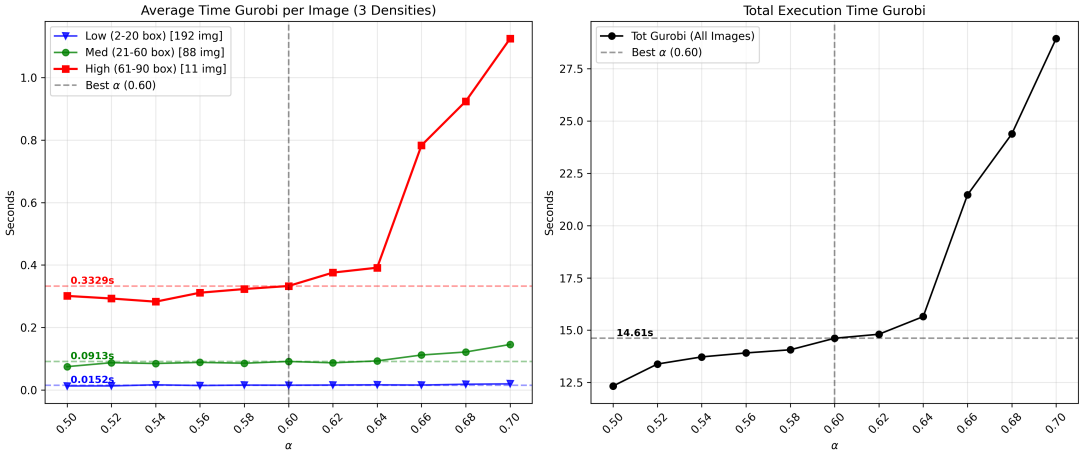


(a) Gurobi execution times for Case 2 (Mixed IoU-IoM) as α grows.

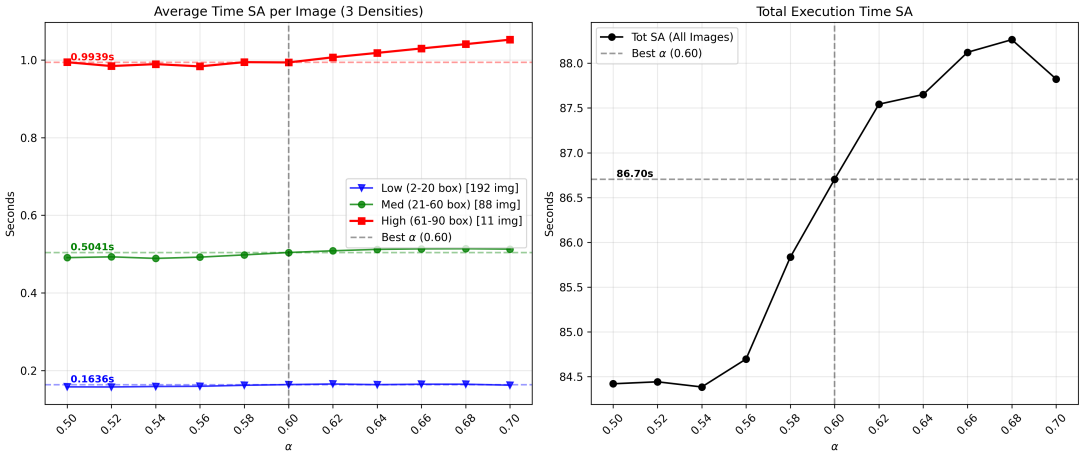


(b) Simulated Annealing execution times for Case 2 (Mixed IoU-IoM) as α grows.

Figure 5.6: Average per-image and total execution times for Case 2 across the hyperparameter α grid. By introducing the IoM penalty, we created a steeper objective function landscape. As a result, Gurobi's algorithm manages to converge faster than the baseline Case 1, whereas the more rugged energy landscape slightly slows down the SA heuristic.

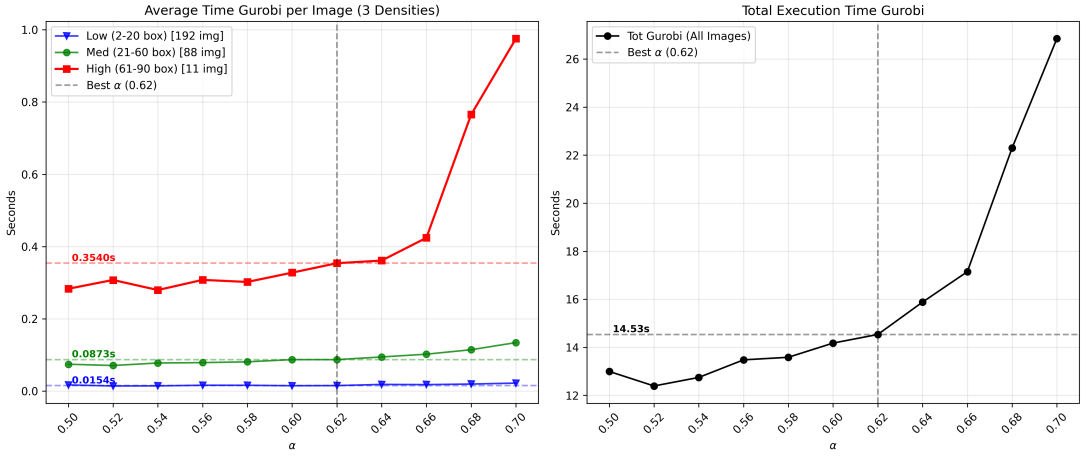


(a) Gurobi execution times for Case 3 (IoU and Spatial Feature) as α grows.

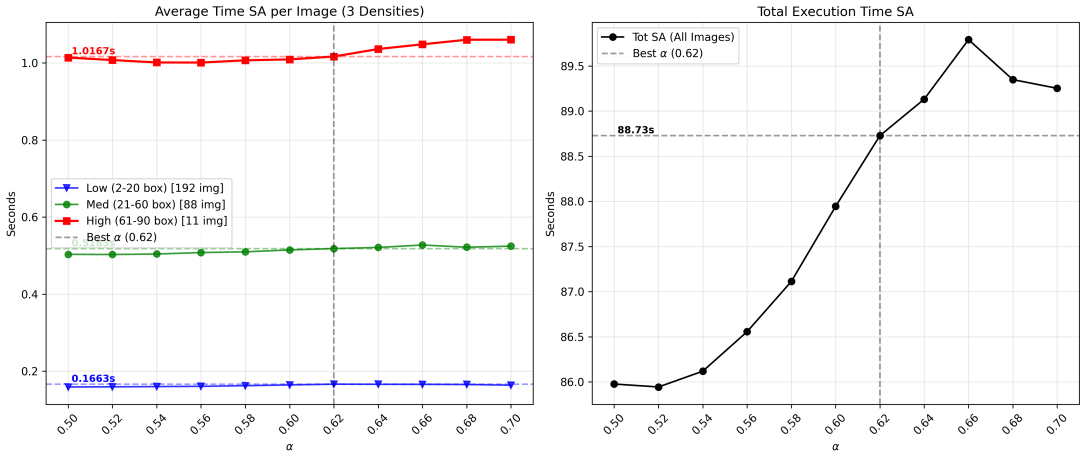


(b) Simulated Annealing execution times for Case 3 (IoU and Spatial Feature) as α grows.

Figure 5.7: Average per-image and total execution times for Case 3 across the hyperparameter α grid. We can observe a trend very similar to Case 2: the spatial feature helps Gurobi prune suboptimal branches more easily, lowering its total execution time compared to the plain IoU approach. On the flip side, SA requires a bit more thermal exploration.



(a) Gurobi execution times for Case 4 (Comprehensive) as α grows.



(b) Simulated Annealing execution times for Case 4 (Comprehensive) as α grows.

Figure 5.8: Average per-image and total execution times for Case 4 across the hyperparameter α grid. These plots confirm our hypothesis: incorporating complex spatial penalties speeds up Gurobi's branch-and-cut optimization (solving the whole batch in under 15 seconds) but slightly penalizes the probabilistic search of Simulated Annealing.

5.2 Analysis on the Limits of the Brute Force Algorithm

Once we set the hyperparameters, we evaluated the exact Brute Force algorithm on single images. The goal of this preliminary test was to validate the theoretical accuracy of our QUBO formulation and to empirically demonstrate the computational bottleneck that justifies the transition to heuristic and quantum approaches.

We tested this exact approach on small instances, such as an image with 5 bounding boxes (ID 532481), one with 14 bounding boxes (ID 270908) and one with 23 bounding boxes (ID 458755) on both CPU and GPU. To push the GPU to its limit, we also included an instance with 31 bounding boxes (ID 213035). Since Brute Force is an exact method, the recovered solution for each problem is always independent of the hardware. The resulting metrics for all images are collected in Table 5.1.

n. boxes	Config.	F1	mAP	mAP(0.5)	mAR@10	mAR@100	MAE
5	Case 1	0.6667	0.2525	0.5050	0.2500	0.2500	5.000
	Case 2	0.6667	0.2525	0.5050	0.2500	0.2500	5.000
	Case 3	0.6667	0.2525	0.5050	0.2500	0.2500	5.000
	Case 4	0.6667	0.2525	0.5050	0.2500	0.2500	5.000
14	Case 1	0.6667	0.8000	1.0000	0.8000	0.8000	0.000
	Case 2	1.0000	0.6000	1.0000	0.6000	0.6000	1.000
	Case 3	1.0000	0.6000	1.0000	0.6000	0.6000	1.000
	Case 4	0.6667	0.8000	1.0000	0.8000	0.8000	0.000
23	Case 1	0.5714	0.3030	0.5050	0.3000	0.3000	3.000
	Case 2	0.5714	0.3030	0.5050	0.3000	0.3000	3.000
	Case 3	0.5714	0.3030	0.5050	0.3000	0.3000	3.000
	Case 4	0.5714	0.3030	0.5050	0.3000	0.3000	3.000
31	Case 1	0.6667	0.3446	0.5545	0.4000	0.4000	3.000
	Case 2	0.6667	0.3446	0.5545	0.4000	0.4000	3.000
	Case 3	0.6667	0.3446	0.5545	0.4000	0.4000	3.000
	Case 4	0.6667	0.3446	0.5545	0.4000	0.4000	3.000

Table 5.1: Exact Brute Force evaluation metrics per single instance with an increasing number of bounding boxes. Finding the mathematical global optimum does not necessarily imply perfect evaluation scores. Note that RMSE is omitted since it coincides with MAE on a single image.

This test proved that even when one finds the mathematical global optimum, the standard mAP score does not reach 1.0. Indeed, some metrics remain lower than one might intuitively expect. This confirms that the accuracy ceiling is an intrinsic limitation of the object detection task itself (such as the initial Faster

R-CNN proposal or the QUBO formulation itself) and it is not caused by the solver's inability to find the best configuration.

While the exact algorithm successfully validates the accuracy of the formulation, the execution times on CPU reported in Table 5.2 highlight the computational limit.

Config.	**5-box (s)**	**14-box (s)**	**23-box (s)**
Case 1	0.0002	0.0696	35.9095
Case 2	0.0002	0.0695	36.2061
Case 3	0.0002	0.0693	36.2928
Case 4	0.0001	0.0682	36.2492

Table 5.2: Brute Force execution times on CPU. The instances feature 5, 14, and 23 bounding boxes.

As anticipated in Section 3.1, Brute Force time complexity grows exponentially as $\mathcal{O}(2^n)$, where n is the number of detected bounding boxes. On a standard CPU, processing the 23-box instance took roughly 35 seconds to check all 8.38 million possible combinations. Since waiting half a minute to process a single image is impractical for computer vision tasks, we tried running the exact same batch on a GPU.

For very small instances (5 and 14 boxes) there is no advantage. However, for the 23-box image, the hardware parallelization provided a massive speedup, as reported in Table 5.3.

Config.	**5-box (s)**	**14-box (s)**	**23-box (s)**	**31-box (s)**
Case 1	0.0626	0.0635	0.0834	39.1878
Case 2	0.0531	0.0537	0.0784	38.0650
Case 3	0.0540	0.0539	0.0787	36.7848
Case 4	0.0536	0.0535	0.0785	36.7649

Table 5.3: Brute Force execution times on GPU. We expanded the evaluation up to 31 bounding boxes.

This GPU speedup rapidly vanishes when testing an image with 31 bounding boxes, since the execution time explodes to over 36 seconds.

To conclude, we proved that, even with a parallelization hardware, exact resolution becomes useless for scenarios with more than 30 variables. These results motivate the use of other classical methods (Gurobi and Simulated Annealing) and Quantum Annealing to process highly crowded images in the following sections.

5.3 Comparative Analysis between Classical and Quantum Solvers

Now that the tuning phase is concluded and the optimal penalty configurations are found, we can finally work on the D-Wave Quantum Annealer and compare it with the classical solvers. Before testing QA on the validation batch, we first observed how scalability and accuracy evolve on six images sorted by number of boxes.

5.3.1 Scalability and Performance on Selected Instances

We chose six representative instances that span a wide range of bounding box counts to test the quantum hardware's time complexity and scalability. We tried to maintain an average step of roughly 15-20 boxes between one image and the next. The selected images, in increasing order of density, are:

1. image ID 532481 with 5 detected bounding boxes;
2. image ID 458755 with 23 detected bounding boxes;
3. image ID 147740 with 42 detected bounding boxes;
4. image ID 57597 with 62 detected bounding boxes;
5. image ID 172946 with 87 detected bounding boxes;
6. image ID 214539 with 99 detected bounding boxes.

We intentionally selected an instance with nearly 100 bounding boxes. This image is one of the most crowded scene (in terms of detected people) within the entire COCO validation dataset, and we included it specifically to "exasperate" both the classical and quantum solvers.

Our first check was on execution times, mainly to verify if the theoretical complexities discussed in Chapter 3 coincide with our empirical results. To present these results as clearly as possible, we generated two types of plots: Figures 5.9 and 5.10 show the absolute execution times for each instance, while Figures 5.11 and 5.12 show line plots that visually trace the scalability as the problem size grows proportionally on the x-axis.

Looking at the classical solvers, Gurobi solves Low-density and Medium-density instances in mere fractions of a second, but as we approach the 87 and 99-box instances, the theoretical exponential growth becomes visible (Figure 5.11a). On the other hand, Simulated Annealing performs poorly in terms of scaling, yielding much higher execution times right from the simpler images compared to Gurobi (Figure 5.9b). However, its time complexity curve approximately grows polynomially (Figure 5.11b), perfectly agreeing with heuristic theory.

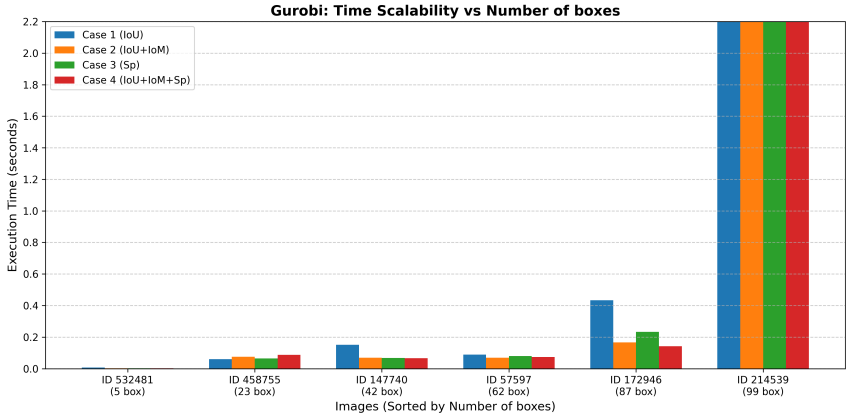
The behavior of the Quantum Annealer is entirely different. We tested the D-Wave QPU at 500, 1000 and 3000 reads per problem. Following the hardware's technical specifications, the total access time directly depends on the number of

requested reads: the execution time roughly doubles from 500 to 1000 reads and nearly triples from 1000 to 3000 (Figure 5.10).

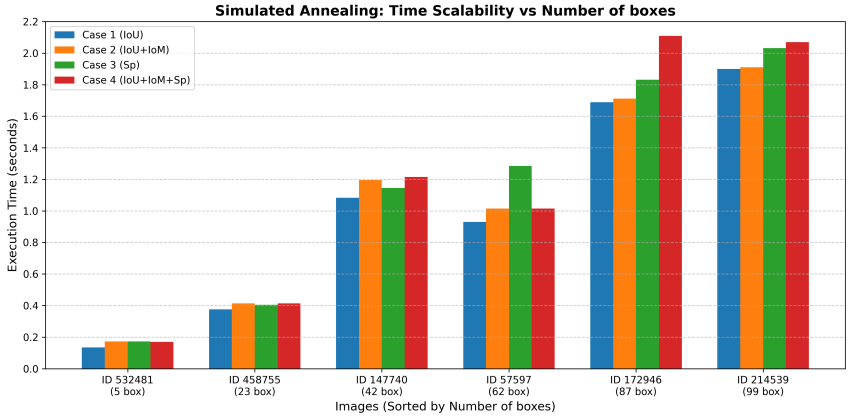
If we look at Figure 5.9, one easily understands that for low-density scenes (like the 5 or 23-box images), Gurobi easily wins in terms of time. As the number of bounding boxes grows, this gap becomes thinner, until Gurobi exponentially explodes when solving the 99-box instance. If we look at Figure 5.12, we can see how flat the QA execution curve remains as the problem scales, independently of the number of reads. This is the tangible proof that quantum hardware offers a massive speedup on dense, highly crowded scenarios.

Beyond execution times, the second aim of this analysis on selected instances was to evaluate detection accuracy. Figures 5.13 and 5.14 plot the standard mAP for the six images. Apart from the Medium-density instances, the absolute mAP scores are extremely low. We expected this behavior since mAP is a macroscopic metric that highly penalizes a single False Positive, as it is designed to evaluate multiple predictions across an entire dataset. To get a much more reliable picture of how the solvers behave on individual images, we also analyzed the F1-score and the Mean Absolute Error (MAE) for each solver. It is worth underlining that the 5-box and 23-box instances analyzed here are the same images used to test the Brute Force approach in Section 5.2. The metric scores achieved by Gurobi coincide with the exact mathematical optimum found by Brute Force, confirming the reliability of our classical baseline before comparing it with the quantum hardware.

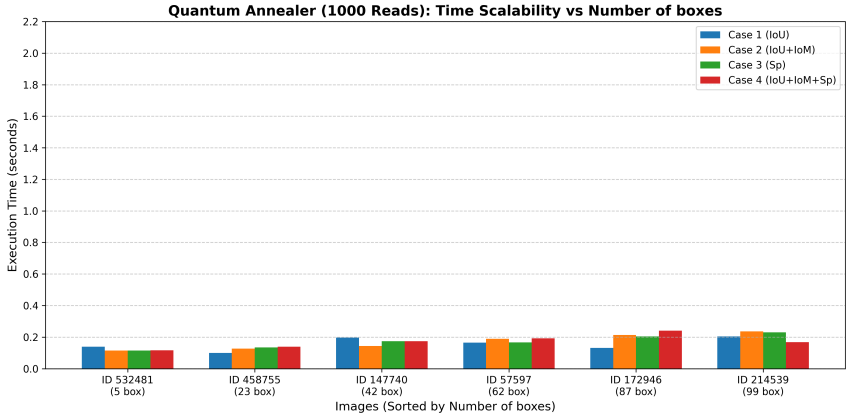
Looking at both the F1-score and the MAE, we can confidently confirm that the QA holds its ground perfectly against the classical algorithms (Figures 5.15 and 5.17). If we only request 500 reads (Figures 5.16a and 5.18a), the quantum device loses some accuracy. This happens because it likely gets stuck in sub-optimal local minima since the energy landscape is not sampled enough times. Bumping the setup to 1000 reads (Figures 5.16b and 5.18b) restores an optimal balance: the quantum accuracy completely matches, and sometimes beats, the classical ones (Figures 5.15 and 5.17). We also noticed that forcing the hardware up to 3000 reads (Figures 5.16c and 5.18c) does not bring any real benefit to the error reduction. This basically confirms that approximately 1000 reads represent a solid mathematical sweet spot to reliably find the global optimum for our QUBO framework.



(a) Gurobi single-image absolute execution times (y-axis is limited; peak time is ~ 4.9s).

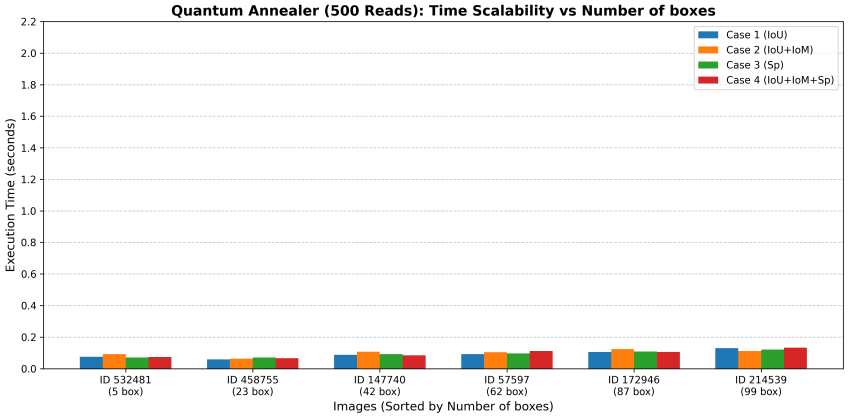


(b) Simulated Annealing single-image absolute execution times.

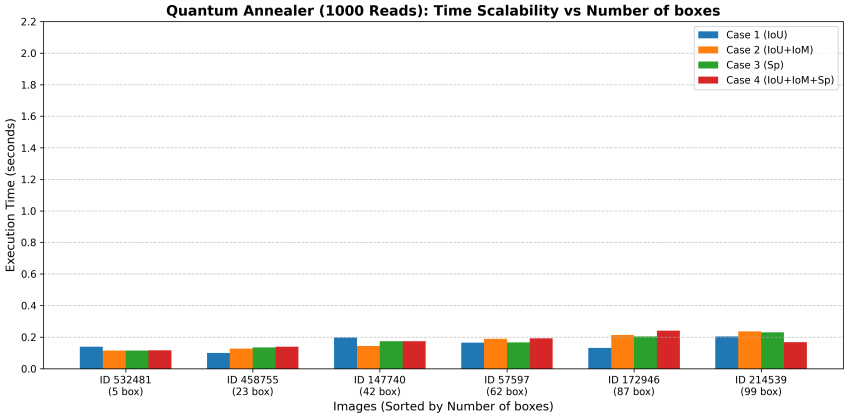


(c) Quantum Annealing (1000 reads) single-image absolute execution times.

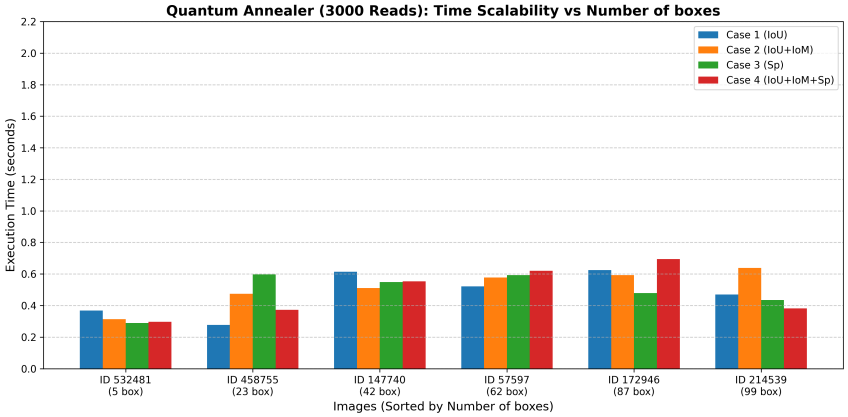
Figure 5.9: Absolute execution times across Gurobi, Simulated Annealing, and the Quantum Annealer (configured at 1000 reads) for the six selected instances. Gurobi is visibly the fastest for low-density images, but its time exponentially spikes on the 99-box instance, reaching almost 5 seconds.



(a) Quantum Annealing (500 reads) single-image absolute execution times.

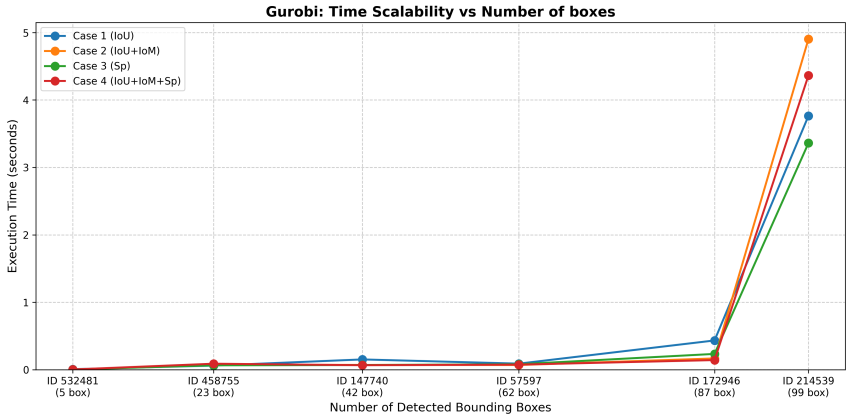


(b) Quantum Annealing (1000 reads) single-image absolute execution times.

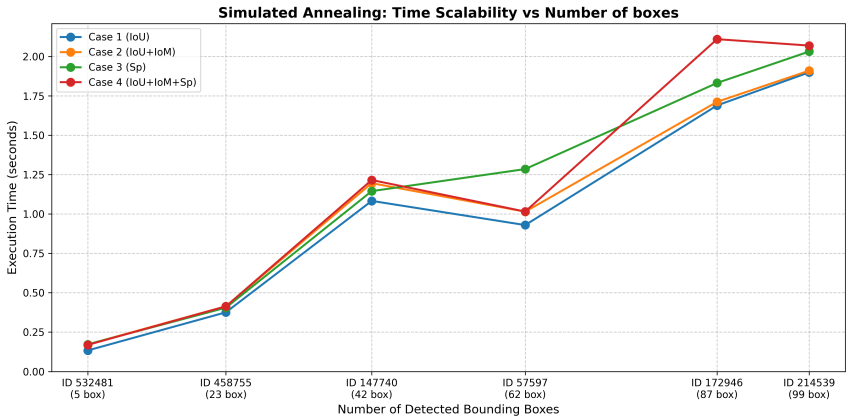


(c) Quantum Annealing (3000 reads) single-image absolute execution times.

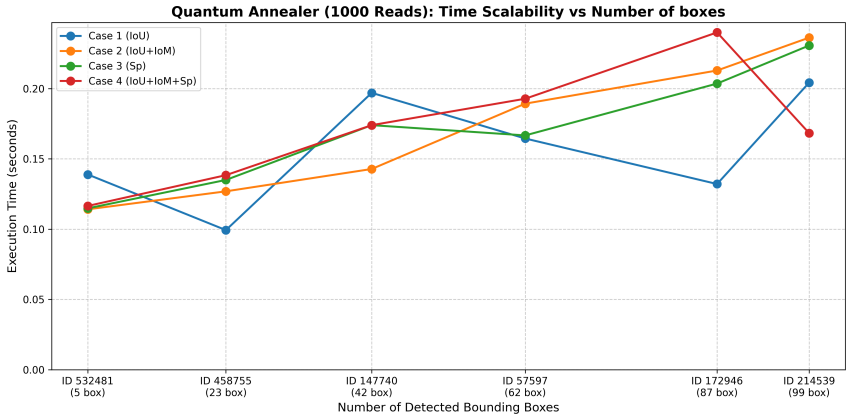
Figure 5.10: Comparison of the absolute execution times across different QA read limits for the six selected instances. The total QPU access time strictly depends on the number of requested samples: the times practically double from 500 to 1000 reads, and nearly triple from 1000 to 3000 reads.



(a) Gurobi time complexity scalability.

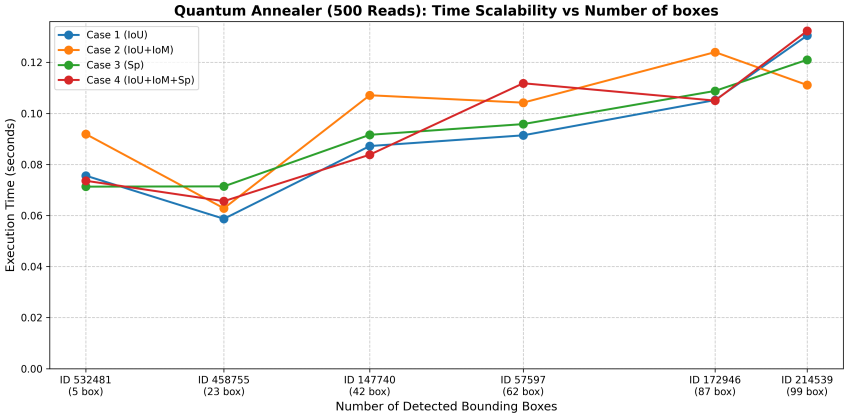


(b) Simulated Annealing time complexity scalability.

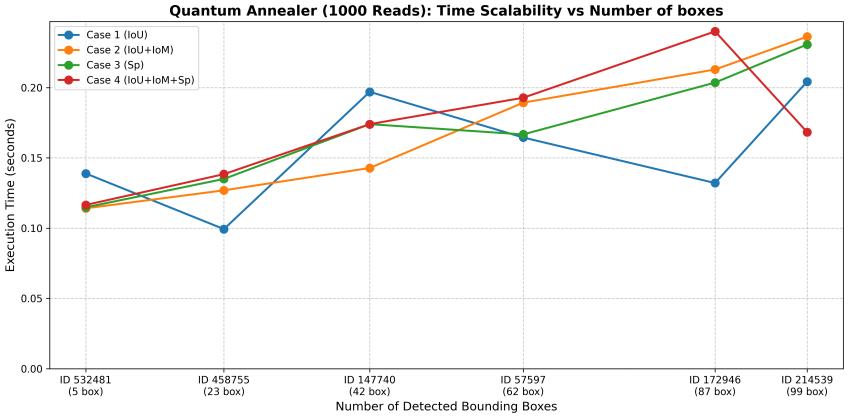


(c) Quantum Annealing (1000 reads) time complexity scalability.

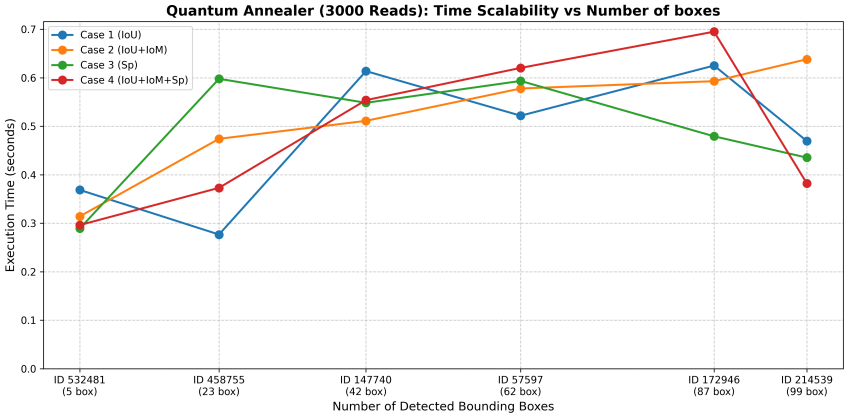
Figure 5.11: Scalability line analysis of execution times across Gurobi, Simulated Annealing, and the Quantum Annealer (configured at 1000 reads) for the six selected instances. The x-axis represents the progressive problem size in terms of initial bounding boxes. Gurobi clearly suffers from the expected exponential growth, SA shows a polynomial trend, while the QA demonstrates a remarkably flat scalability profile regardless of density.



(a) Quantum Annealing (500 reads) time complexity scalability.

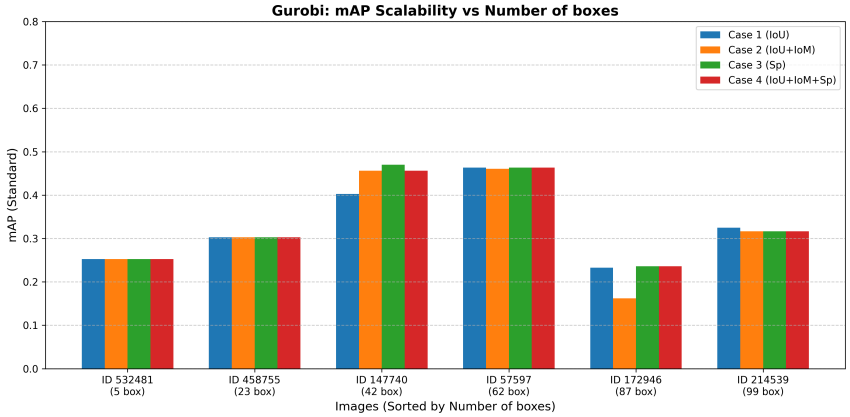


(b) Quantum Annealing (1000 reads) time complexity scalability.

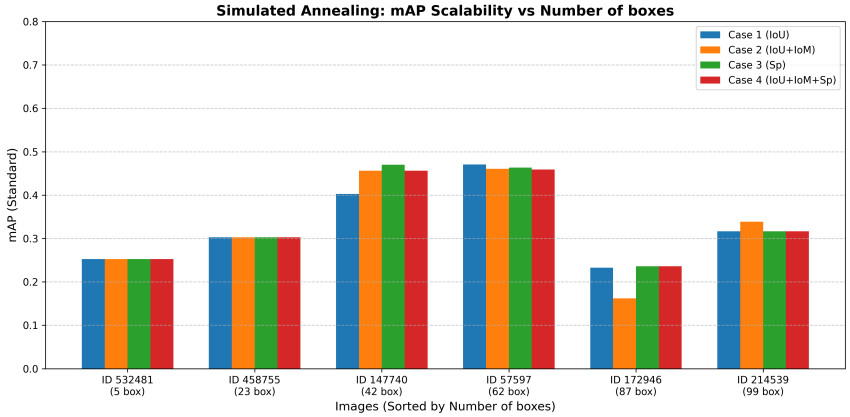


(c) Quantum Annealing (3000 reads) time complexity scalability.

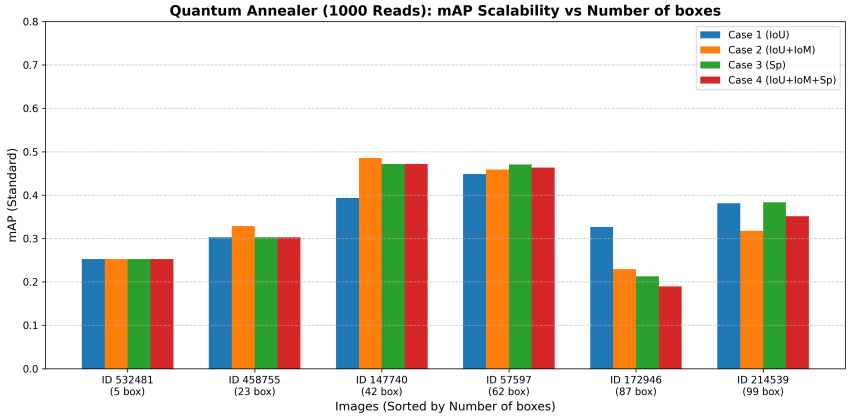
Figure 5.12: Comparison of the scalability line analysis across different QA read limits for the six selected instances. The absolute execution time shifts upward as reads increase, but the fundamental flat shape of the curve proves that the QA execution time is practically immune to the problem size growth.



(a) Gurobi single-image mAP scores.

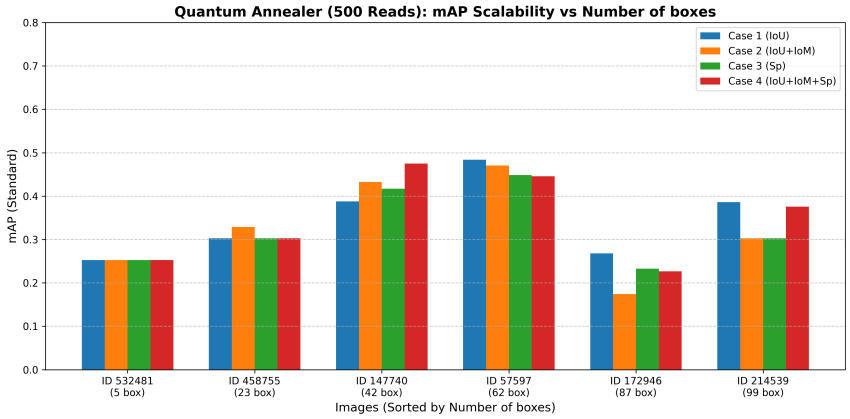


(b) Simulated Annealing single-image mAP scores.

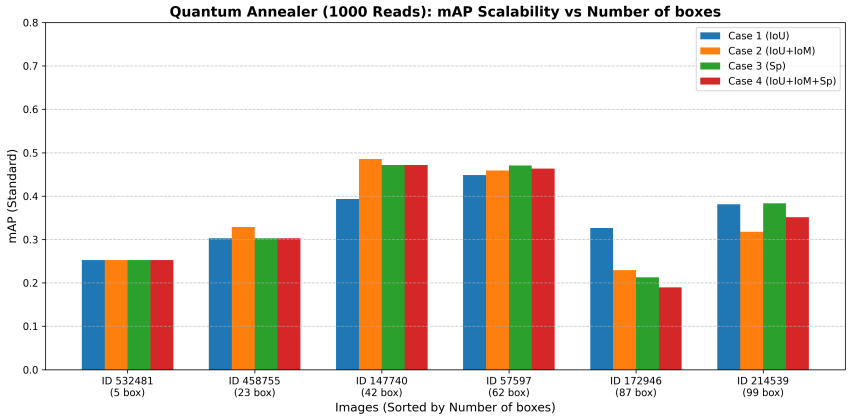


(c) Quantum Annealing (1000 reads) single-image mAP scores.

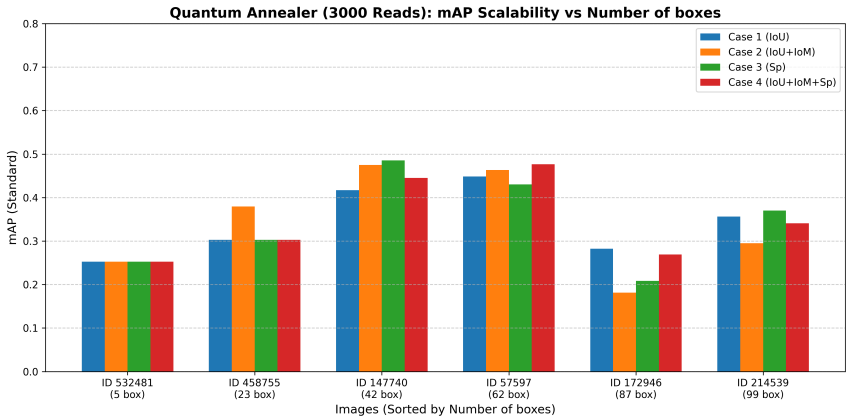
Figure 5.13: Standard mAP_{std} evaluated individually on the six selected instances across Gurobi, Simulated Annealing, and the Quantum Annealer (configured at 1000 reads). As extensively discussed, the absolute values are extremely volatile because this metric is not designed to be evaluated on a single image.



(a) Quantum Annealing (500 reads) single-image mAP scores.

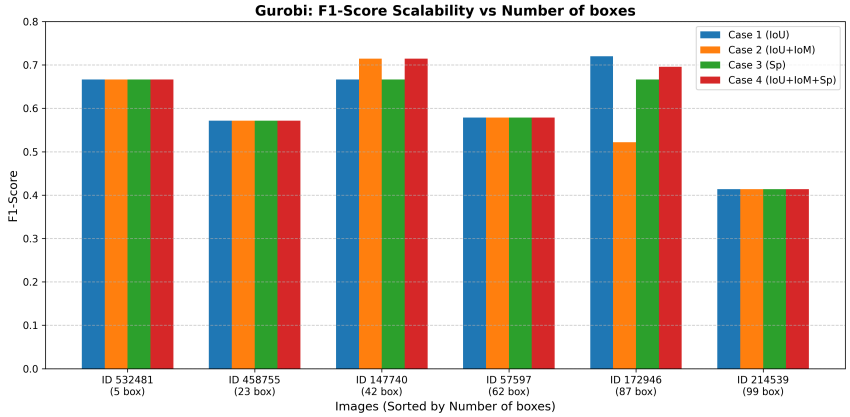


(b) Quantum Annealing (1000 reads) single-image mAP scores.

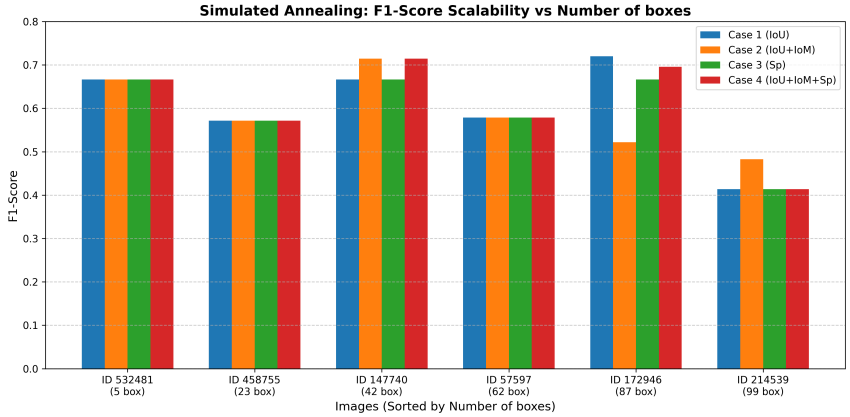


(c) Quantum Annealing (3000 reads) single-image mAP scores.

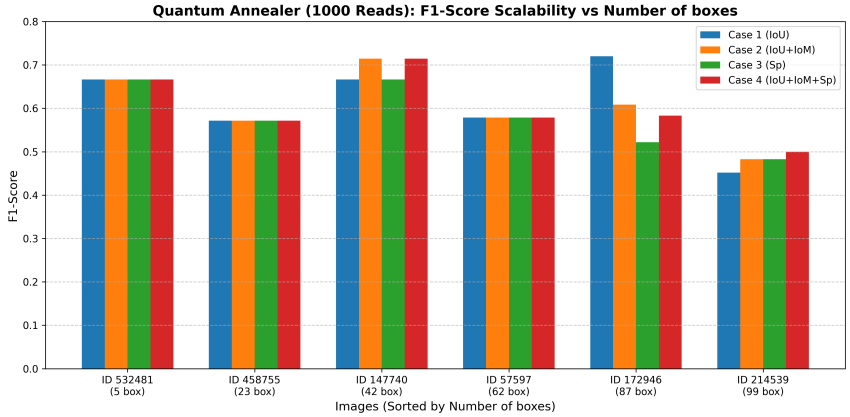
Figure 5.14: Comparison of the standard $\mathrm{mAP}_{\mathrm{std}}$ evaluated individually on the six instances across different QA read limits. Just like the classical solvers, the QA mAP scores appear very low due to the intrinsic metric sensitivity.



(a) Gurobi single-image F1-scores.

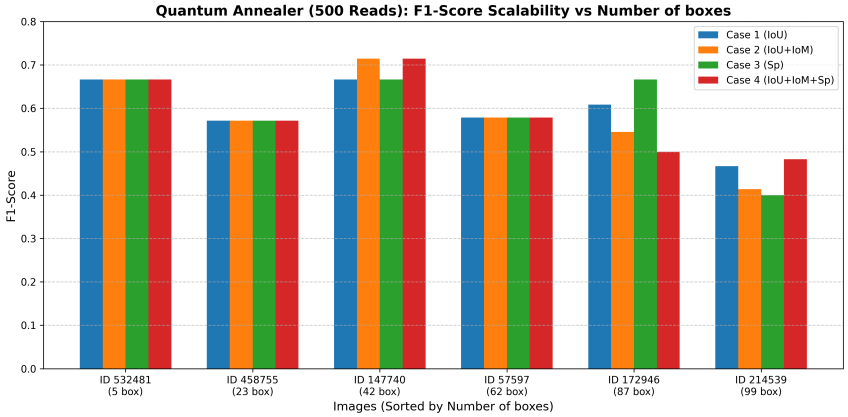


(b) Simulated Annealing single-image F1-scores.

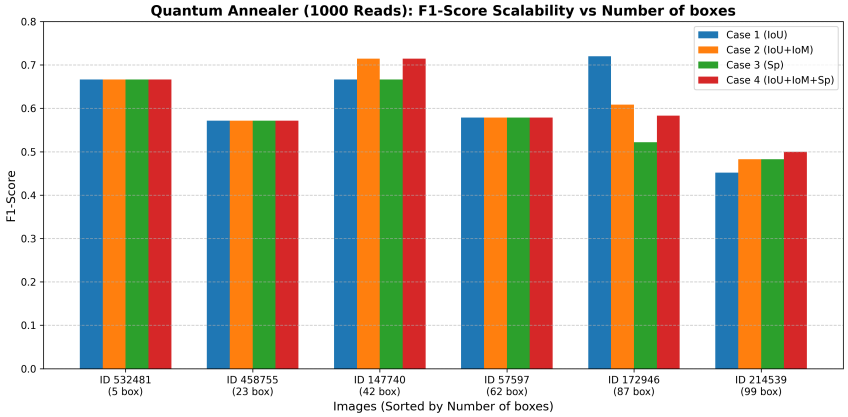


(c) Quantum Annealing (1000 reads) single-image F1-scores.

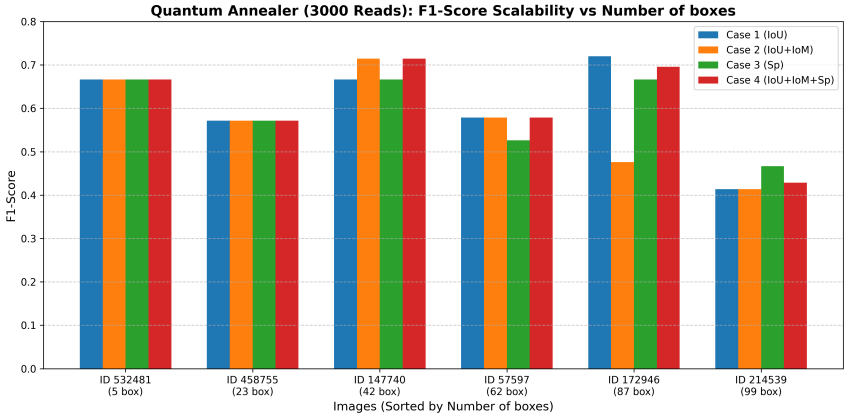
Figure 5.15: F1-score evaluation across Gurobi, Simulated Annealing, and the Quantum Annealer (configured at 1000 reads) for the six selected instances. The robust F1 metric clearly confirms that the QA running at 1000 reads holds its ground perfectly, matching the accuracy of classical algorithms.



(a) Quantum Annealing (500 reads) single-image F1-scores.

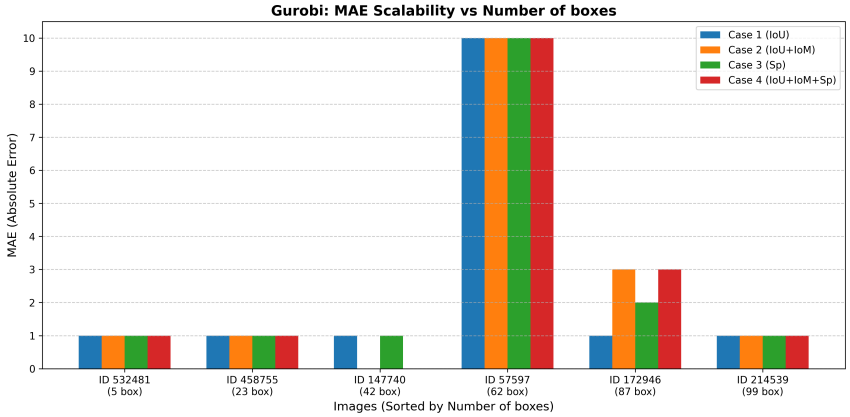


(b) Quantum Annealing (1000 reads) single-image F1-scores.

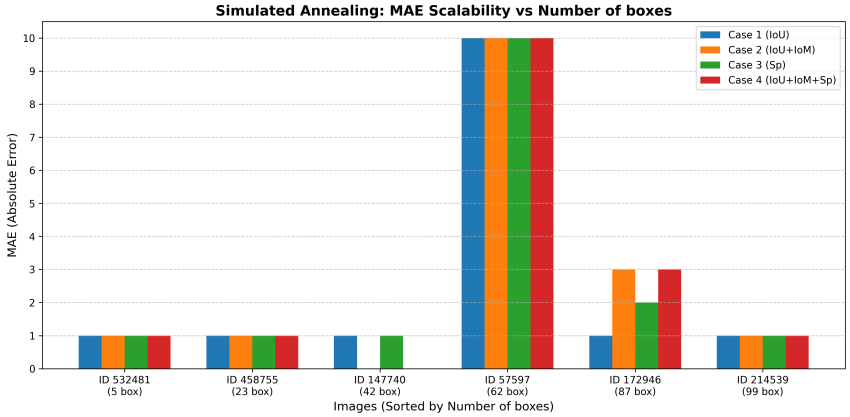


(c) Quantum Annealing (3000 reads) single-image F1-scores.

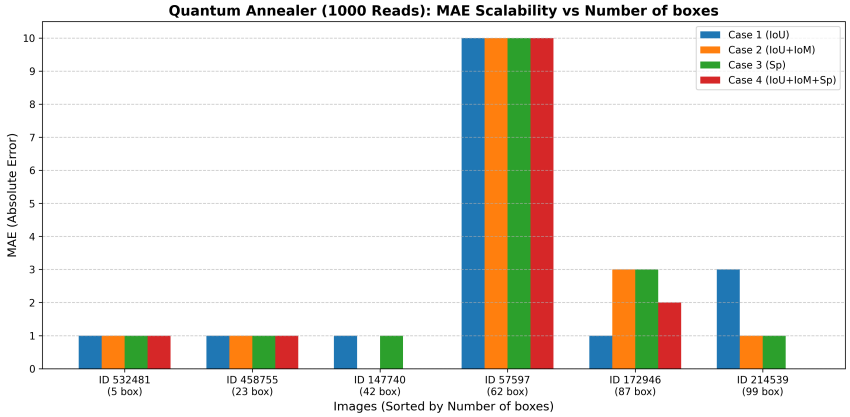
Figure 5.16: Comparison of the F1-score evaluation across different QA read limits for the six selected instances. At 500 reads, the QPU occasionally drops in accuracy. Conversely, 3000 reads provide no real benefit, proving 1000 reads to be the optimal mathematical balance.



(a) Gurobi single-image Mean Absolute Error.

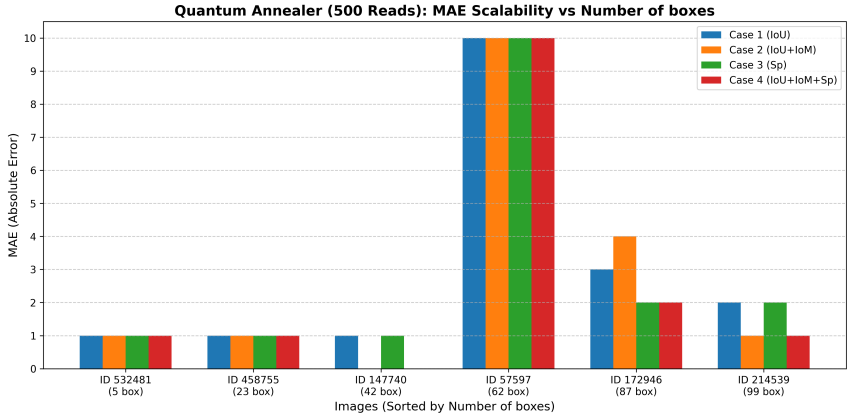


(b) Simulated Annealing single-image Mean Absolute Error.

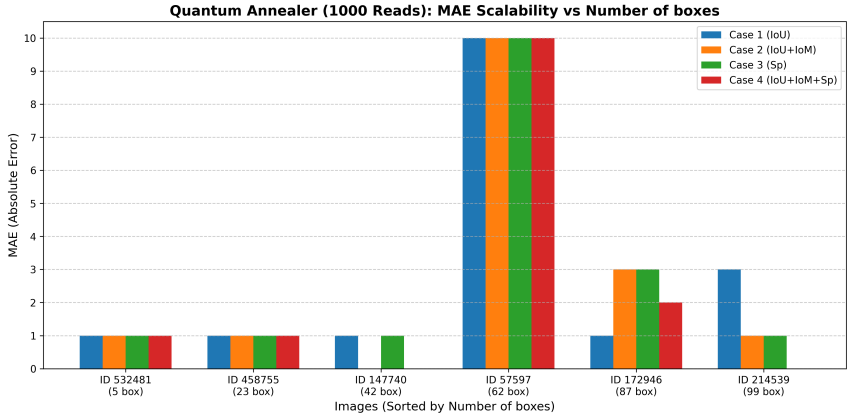


(c) Quantum Annealing (1000 reads) single-image Mean Absolute Error.

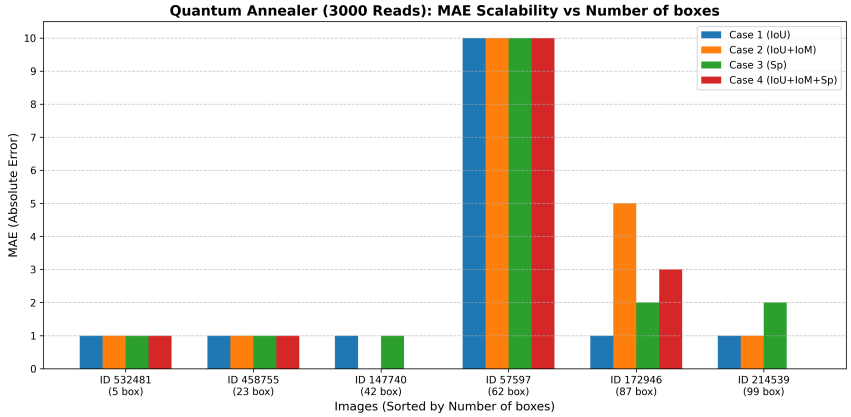
Figure 5.17: Mean Absolute Error (MAE) evaluation across Gurobi, Simulated Annealing, and the Quantum Annealer (configured at 1000 reads) for the six selected instances. Consistent with the F1-score results, the QA effectively minimizes the error gap identically to Gurobi.



(a) Quantum Annealing (500 reads) single-image Mean Absolute Error.



(b) Quantum Annealing (1000 reads) single-image Mean Absolute Error.



(c) Quantum Annealing (3000 reads) single-image Mean Absolute Error.

Figure 5.18: Comparison of the Mean Absolute Error (MAE) evaluation across different QA read limits for the six selected instances. The metric validates the hypothesis that 500 reads are insufficient and 3000 reads represent an unnecessary over-sampling for this QUBO formulation.

5.3.2 Qualitative Visual Analysis

To better understand how the different penalty terms and solver configurations affect the final output, we present a visual printing of some instances. We plotted the surviving bounding boxes resulting from the QUBO-suppression in green and the Ground Truths provided by the COCO dataset in red. The more overlaps, the more accuracy is reached.

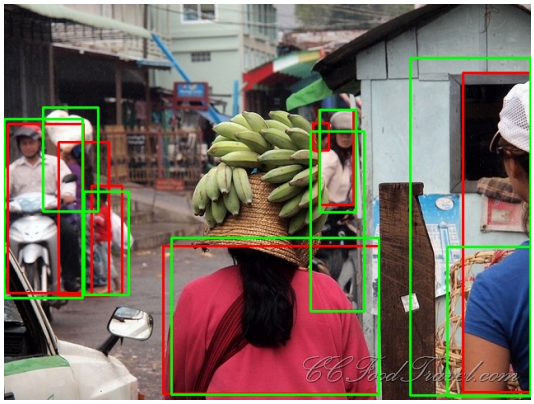
In this section, we present only the noteworthy cases obtained from the analysis on selected instances (Section 5.3.1): a Medium-density image (42 boxes) and two High-density images (87 boxes and 99 boxes). To avoid overloading, we present a comparison between the QA performances at 500 and 1000 reads (top) and Gurobi solutions (bottom). We present these comparisons across the different penalty configurations (Cases 1 to 4) to demonstrate how the formulation impacts the qualitative accuracy of the detection.

Firstly we analyze Case 1 (the baseline case), shown in Figure 5.19. With a medium density of 42 boxes, the Quantum Annealer operates in a highly favorable environment. Gurobi (Figure 5.19c) and the QA at 1000 reads (Figure 5.19b) select a different bounding box for the woman in the pink shirt, yet both are entirely valid. On the other hand, even reducing to 500 reads (Figure 5.19a), the solver finds an optimal solution, though selecting slightly different bounding boxes for the people in the background with respect to Gurobi. We can confirm that in a Medium-density instance like this, the QA at 500 reads still performs as well as classical solvers. We want to underline that this penalty configuration incorrectly gives a double detection for the person centered in the background, so additional spatial penalties are needed.

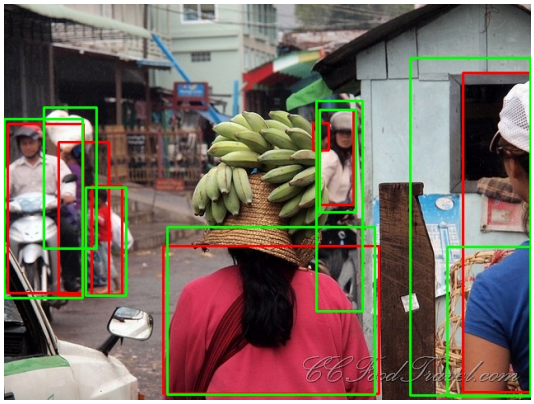
Moving to Case 2 (IoU combined with IoM), shown in Figure 5.20, both the QA at 500 reads (Figure 5.20a) and Gurobi (Figure 5.20c) return the same solution. In Figure 5.20b we can see the QA at 1000 reads selects for the woman in the pink shirt a bounding box that overlaps more accurately with the respective Ground Truth than Gurobi's box. This shows that maybe reaching the exact optima does not always guarantee the best set of bounding boxes for each Ground Truth in this case. Note that the double detection that appears in Case 1 has now disappeared.

In Figure 5.21, we present results for Case 3 (IoU and Spatial Feature). Once again, Gurobi (Figure 5.21c) and the QA at 1000 reads (Figure 5.21b) select different bounding boxes for the people in the background, yet both are entirely valid. Here, we notice that the 500-reads configuration (Figure 5.21a) performs worse, by giving a less accurate bounding box for the person on the right compared with other figures. Additionally, this setup still suffers from the same double detection as Case 1.

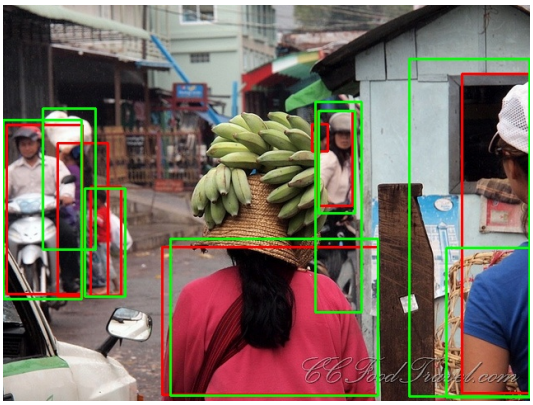
Finally, we present Case 4 (the most comprehensive formulation) in Figure 5.22. Given the relatively manageable density of 42 boxes, both the classical solver and the quantum hardware solve the problem effectively. Looking at IoU, IoM, and spatial distance all at the same time gives the algorithm the right clues to separate people even when they heavily overlap, delivering the cleanest visual output.



(a) Quantum Annealing with 500 reads (Case 1).

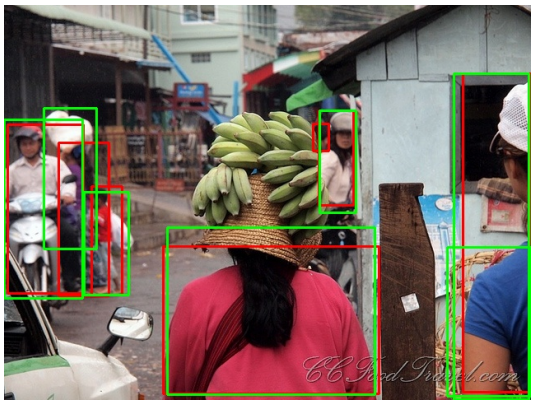


(b) Quantum Annealing with 1000 reads (Case 1).

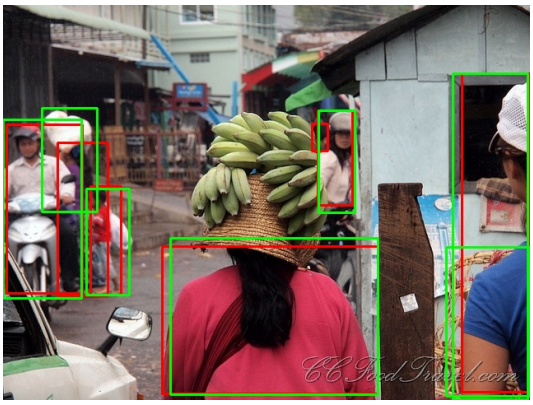


(c) Gurobi (Case 1).

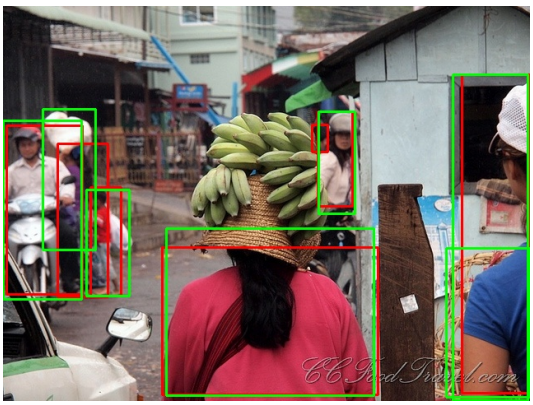
Figure 5.19: Qualitative visual comparison for the 42-box instance (ID 147740) in Case 1. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. In a Medium-density instance like this, the QA at 500 reads still performs as well as Gurobi. Both Gurobi and the QA at 1000 reads handle the scene perfectly, though selecting slightly different bounding boxes for the person in the background, demonstrating the flexibility of the QUBO energy landscape. Moreover, this penalty configuration incorrectly gives a double detection for the person centered in the background.



(a) Quantum Annealing with 500 reads (Case 2).

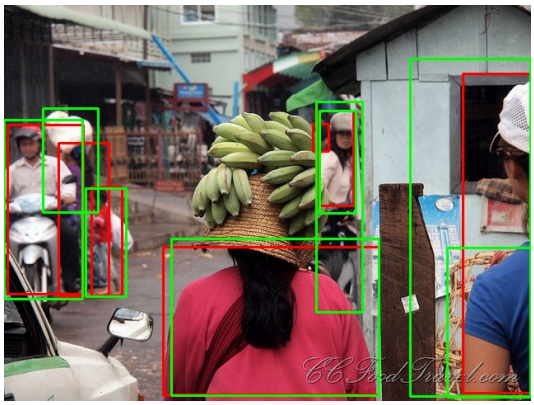


(b) Quantum Annealing with 1000 reads (Case 2).

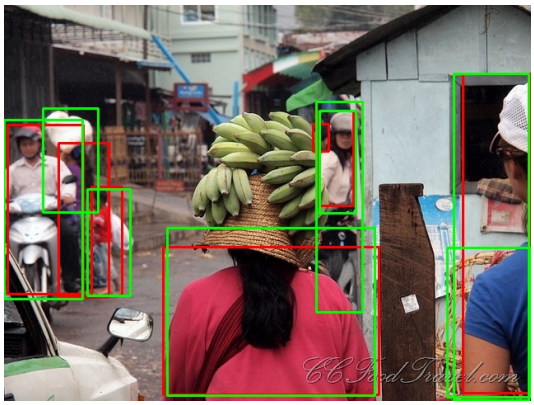


(c) Gurobi (Case 2).

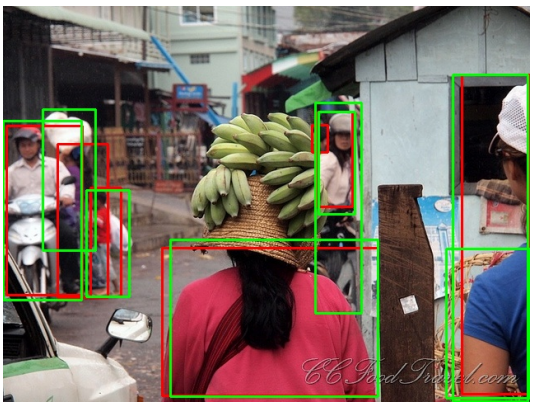
Figure 5.20: Qualitative visual comparison for the 42-box instance (ID 147740) in Case 2. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. Both Gurobi and the QA at 500 reads handle the scene perfectly. Looking at the woman in the pink shirt, we can see that the QA at 1000 reads outperforms Gurobi by choosing a box that is more accurate compared to the Ground Truth.



(a) Quantum Annealing with 500 reads (Case 3).

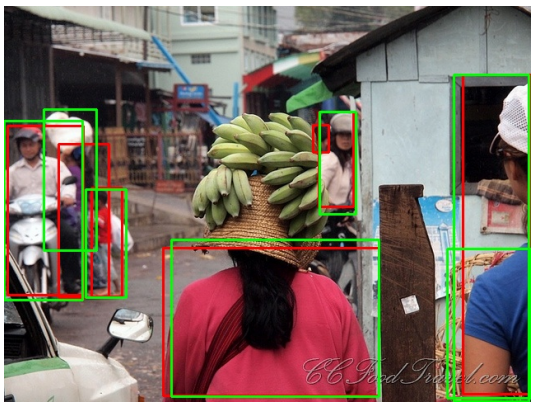


(b) Quantum Annealing with 1000 reads (Case 3).

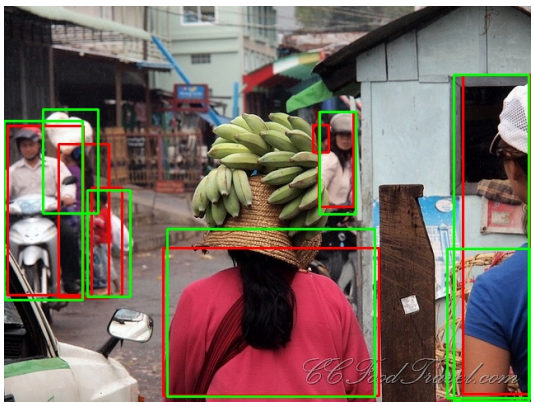


(c) Gurobi (Case 3).

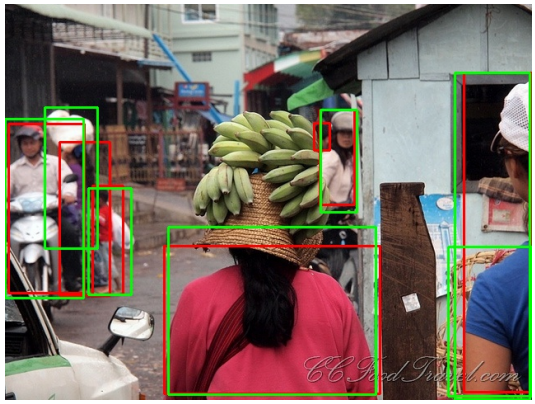
Figure 5.21: Qualitative visual comparison for the 42-box instance (ID 147740) in Case 3. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. Both Gurobi and the QA at 1000 reads handle the scene perfectly. Looking at the person on the right, we can see that the QA at 500 reads detects a different box from other solvers, losing accuracy. Moreover, this penalty configuration incorrectly gives a double detection for the person centered in the background.



(a) Quantum Annealing with 500 reads (Case 4).



(b) Quantum Annealing with 1000 reads (Case 4).



(c) Gurobi (Case 4).

Figure 5.22: Qualitative visual comparison for the 42-box instance (ID 147740) in Case 4. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. All solvers handle the scene perfectly, selecting slightly different bounding boxes. Furthermore, this comprehensive configuration successfully prevents the double-detection artifacts seen in previous cases.

Let us now move to the High-density instances, starting with the 87-box image (ID 172946). As expected, the scene is heavily cluttered, making the suppression task significantly harder. Figure 5.23 shows the results for Case 1. If we focus on the far left side of the image, the limitations of the 500-reads configuration become obvious. The QA at 500 reads (Figure 5.23a) completely misses the optimal bounding box for the partially visible person on the left edge, improperly suppressing it. This happens because the massive search space probably traps the quantum hardware in a sub-optimal local minimum. When we increase the sampling to 1000 reads (Figure 5.23b), the QPU successfully escapes this trap, recovering the missing detection just like Gurobi (Figure 5.23c). However, the person at the right edge in the red shirt is double detected in the QA (1000 reads) solution.

Moving to Case 2 (IoU combined with IoM), presented in Figure 5.24, we observe an interesting phenomenon. The addition of the IoM metric suppresses an important box that was recovered in the previous case. If we look at the woman in the black dress in the center-right, we observe that both the QA at 500 reads (Figure 5.24a) and Gurobi (Figure 5.24c) fail to recover her. Conversely, the QA at 1000 reads (Figure 5.24b) actually keeps a large bounding box that correctly detects the woman. This phenomenon also occurred in Case 2 when analyzing the 42-box instance, where the QA at 1000 reads settles in a local minimum that reveals a better suppression compared to the global optimum. This definitively proves that reaching the global optimum of the energy landscape does not always guarantee the best set of bounding boxes for our QUBO-formulation in Case 2. Moreover, both the 500-read and 1000-read quantum configurations fail to recover the far-left detection that was also missed by QA at 500 reads in Case 1.

In Figure 5.25, we present the visual outcomes for Case 3 (IoU and Spatial Feature). This scenario provides a perfect example of why qualitative analysis is essential alongside global numerical metrics. Looking at the group of women in the center, detected by the QA at 500 reads (Figure 5.25a), we can observe a massive cluster of overlapping green bounding boxes. The QPU completely failed to suppress this dense region, evidently getting trapped in a local minimum. Ironically, this failure might artificially maintain a similar F1 score to classical solvers, as failing to suppress preserves True Positives by default, even if the results are clearly worse. Conversely, the classical solution provided by Gurobi (Figure 5.25c) and the quantum solution at 1000 reads (Figure 5.25b) behave much better, both successfully cleaning the redundant boxes. They achieve similar high-quality suppression (although the F1-score evaluated from the QA solution is lower than Gurobi's) even if they select slightly different bounding boxes. For instance, the QA at 1000 reads successfully recovers the person on the far left, but it selects a larger bounding box compared to Gurobi. Furthermore, on the far right edge, the QA at 1000 reads correctly identifies the man in the red shirt without the double-detection seen in Case 1. However, it selects a small box that isolates only his head and torso, whereas Gurobi selects a box covering his full body.

Finally, Figure 5.26 illustrates Case 4, our most comprehensive formulation combining IoU, IoM, and spatial distance. Comparing Gurobi (Figure 5.26c)

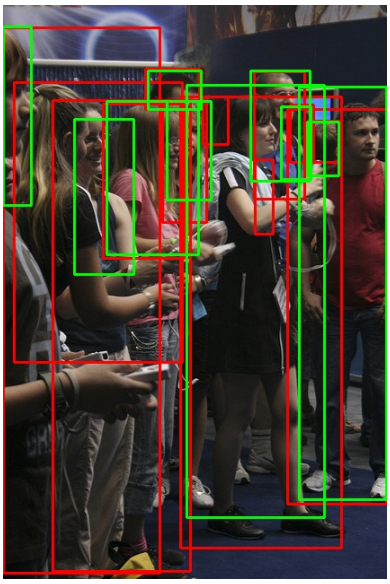
and the QA at 1000 reads (Figure 5.26b), we can see they are very similar, confirming the high quality of the quantum suppression in this complex scenario. The most notably difference is the missed detection by the quantum solver for the woman in the black dress. On the other hand, the 500-reads configuration (Figure 5.26a) struggles significantly. It creates a large bounding box on the far left that incorrectly groups multiple people together. Moreover, in the center of the image, it improperly suppresses several smaller, valid boxes that both Gurobi and the QA at 1000 reads successfully keep. In addition, we observe a double-detection artifact on the man in the red shirt on the far right. This visual output definitively confirms that as the density and the formulation complexity increase, approximately 1000 reads are strictly necessary to avoid failures.

Finally, we analyze the densest instance in the COCO validation subset: image ID 214539 with 99 bounding boxes. This image differs from the previous ones as it presents a clear spatial distinction between the people in the foreground (the football players) and the background subjects (the people on the bench). Since the visual differences across all four penalty configurations are less pronounced in this specific image, we focus our analysis on the baseline (Case 1) and the comprehensive formulation (Case 4).

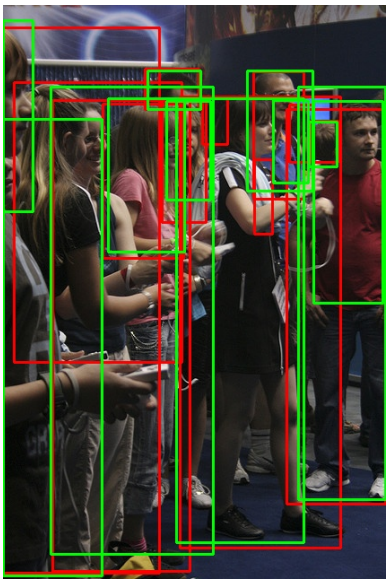
Figure 5.27 shows the results for Case 1. Despite the massive number of bounding boxes, the clear isolation of the foreground subjects makes the suppression task surprisingly straightforward. All solvers, including the QA at 500 reads (Figure 5.27a), easily identify and retain the optimal boxes for the main players. As we expected, the main differences all concern the people in the background, even if the overlapping boxes make it difficult even for the human eye to distinguish them. In this baseline scenario without spatial penalties, all approaches obtain good results.

When moving to Case 4 (Figure 5.28), the differences in the background become more evident. As expected, Gurobi (Figure 5.28c) and the QA at 1000 reads (Figure 5.28b) perform almost identically. On the other side, the QA at 500 reads (Figure 5.28a) leaves a few more redundant boxes in the background, confirming a slight worse performance. The main difference concerns the central football player, where the quantum solver selects a slightly different bounding box compared to Gurobi. Indeed, the QA slightly crops the player's left hand, but in turn, it covers his left foot much better than the classical solver's choice. In the end, the overlap between QA's bounding box and the Ground Truth is probably very similar to Gurobi's.

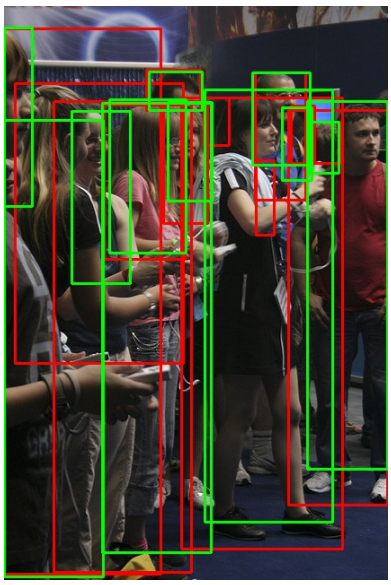
This qualitative visual analysis illustrates that different bounding box selections can yield an almost identical IoU with the Ground Truth. Consequently, we have proved the existence of equivalent, degenerate ground states within our QUBO energy landscape.



(a) Quantum Annealing with 500 reads (Case 1).

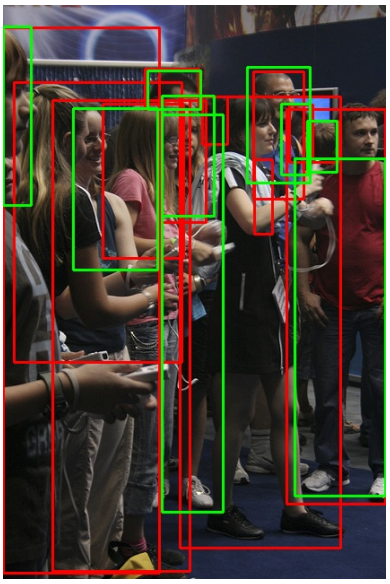


(b) Quantum Annealing with 1000 reads (Case 1).

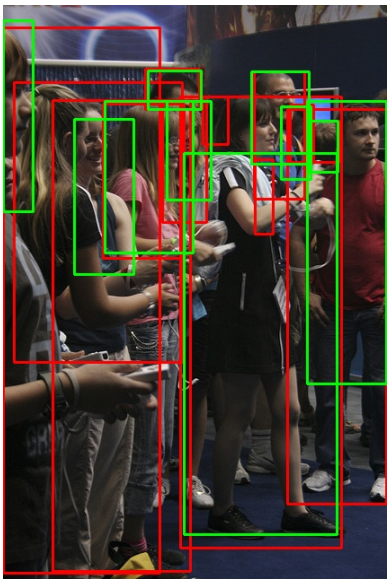


(c) Gurobi (Case 1).

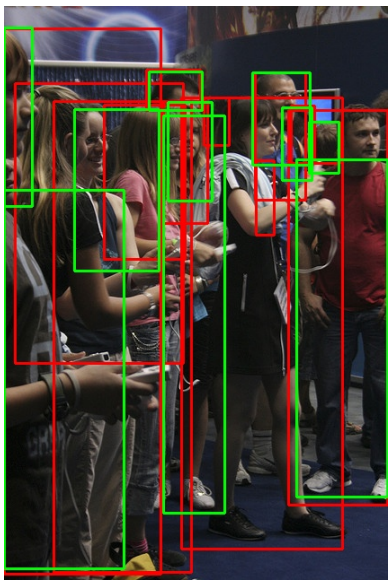
Figure 5.23: Qualitative visual comparison for the 87-box instance (ID 172946) in Case 1. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. The extreme density impacts the quantum runs: the QA at 500 reads struggles on the far left, missing a key detection. Increasing the sampling to 1000 reads allows the QPU to recover the missing bounding box, but simultaneously introduces a double-detection on the far right edge.



(a) Quantum Annealing with 500 reads (Case 2).

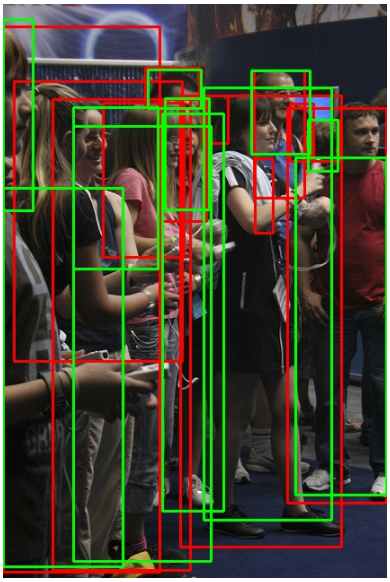


(b) Quantum Annealing with 1000 reads (Case 2).

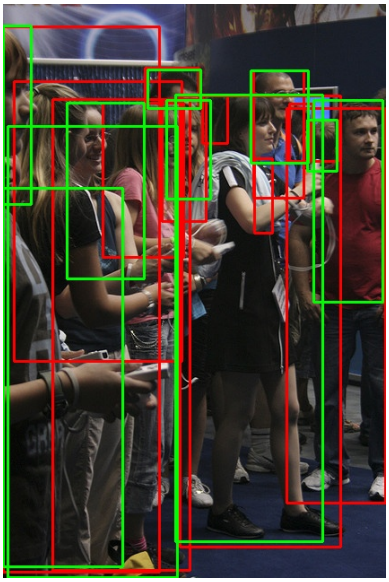


(c) Gurobi (Case 2).

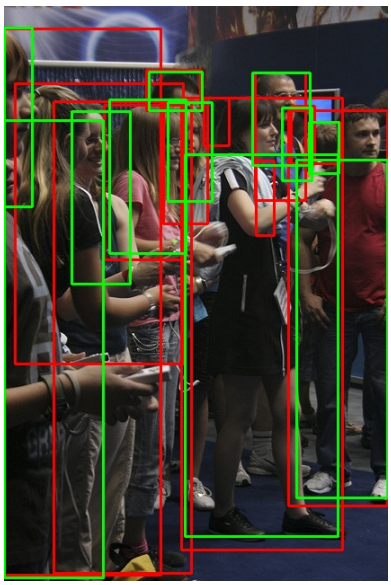
Figure 5.24: Qualitative visual comparison for the 87-box instance (ID 172946) in Case 2. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. In this scenario, Gurobi's global optimum suppresses the large bounding box on the woman in the black dress. Conversely, the QA at 1000 reads finds a sub-optimal energy state that yields a highly accurate detection for that same woman.



(a) Quantum Annealing with 500 reads (Case 3).

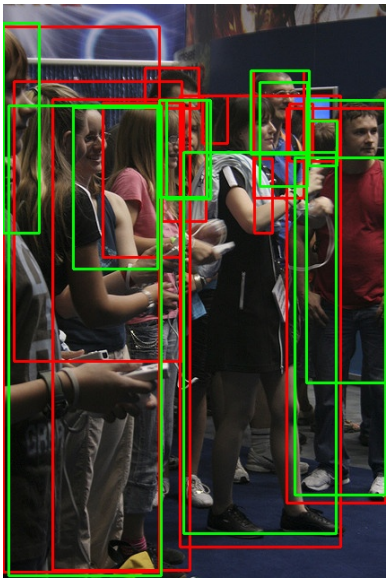


(b) Quantum Annealing with 1000 reads (Case 3).

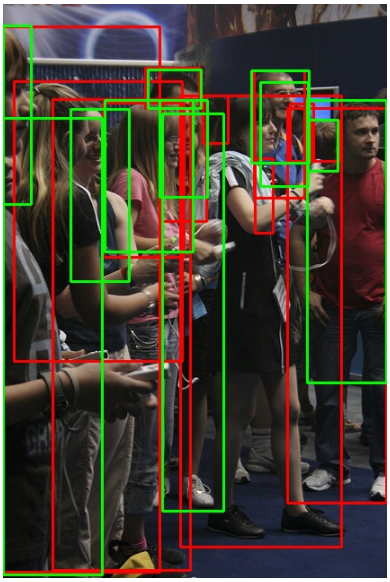


(c) Gurobi (Case 3).

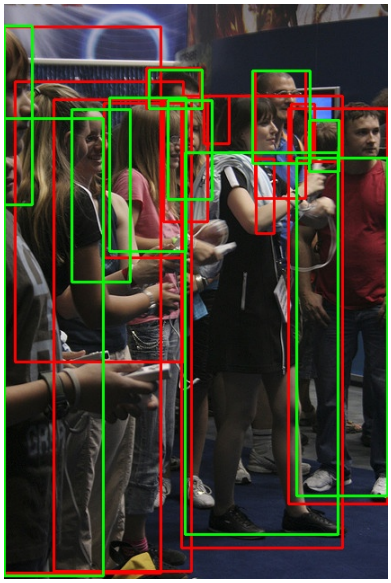
Figure 5.25: Qualitative visual comparison for the 87-box instance (ID 172946) in Case 3. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. The QA at 500 reads completely fails to suppress the dense cluster in the center, leaving a massive overlap of redundant boxes. While Gurobi and the QA at 1000 reads provide a much cleaner suppression, they select slightly different boxes: the quantum approach uses a larger box for the far-left detection and a tighter box for the man on the far right.



(a) Quantum Annealing with 500 reads (Case 4).



(b) Quantum Annealing with 1000 reads (Case 4).



(c) Gurobi (Case 4).

Figure 5.26: Qualitative visual comparison for the 87-box instance (ID 172946) in Case 4. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. Gurobi and the QA at 1000 reads provide a highly similar suppression, with the quantum approach only missing the woman in the black dress. Conversely, the QA at 500 reads completely breaks down: it keeps a massive box on the left, incorrectly suppresses valid small boxes in the center, and leaves a double detection on the right.

(a) Quantum Annealing with 500 reads (Case 1).

(b) Quantum Annealing with 1000 reads (Case 1).

(c) Gurobi (Case 1).

Figure 5.27: Qualitative visual comparison for the 99-box instance (ID 214539) in Case 1. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. Thanks to the clear foreground/background separation, all solvers easily identify the main subjects, including the under-sampled QA at 500 reads.

(a) Quantum Annealing with 500 reads (Case 4).

(b) Quantum Annealing with 1000 reads (Case 4).

(c) Gurobi (Case 4).

Figure 5.28: Qualitative visual comparison for the 99-box instance (ID 214539) in Case 4. The green boxes are the results of QUBO-suppression, the red ones are the Ground Truths. Both Gurobi and the QA at 1000 reads perform identically for people in the background, while the QA at 500 reads leaves a few more redundant boxes. Looking at the central player, Gurobi and the QA select slightly different boxes (one saves the hand, the other saves the foot), visually demonstrating the presence of equivalent optimal solutions in the QUBO formulation.

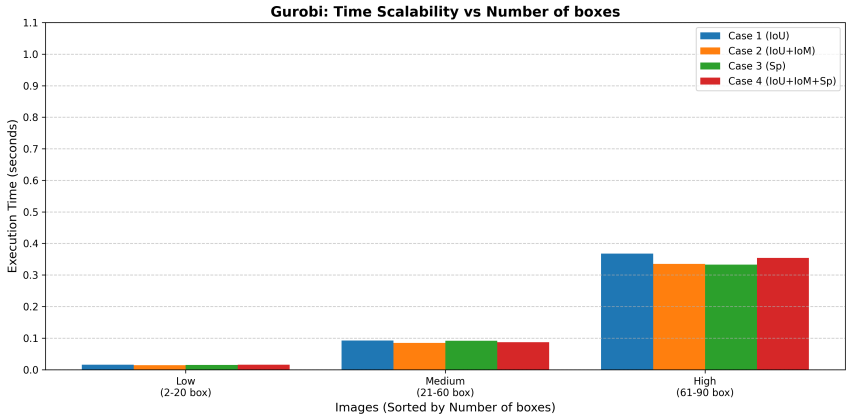
5.3.3 Scalability and Performance on Validation Subset

After analyzing the behavior of the QUBO formulations on the six selected instances, we expand our analysis to a large scale to confirm these trends. We evaluated our QUBO formulations on a validation subset consisting of 291 images from the COCO dataset, covering all density classes. The examination of 291 instances for the four penalty configurations requires over a thousand different calls to each solver. While Gurobi and Simulated Annealing results were already computed during the hyperparameters tuning (Section 5.1), we had to choose a suitable number of reads for the Quantum Annealer in order to optimize resource consumption, while maintaining high accuracy. Based on our preliminary analysis, we established that approximately 1000 reads were required to successfully navigate the complex energy landscapes, so we decided to keep 900 reads to analyze this batch.

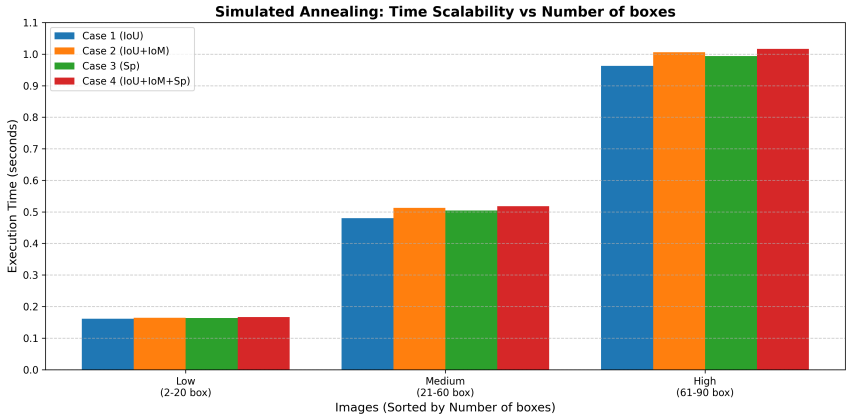
The results perfectly align with our theoretical expectations. Both Gurobi (Figure 5.29a) and Simulated Annealing (Figure 5.29b) show a clear growth in execution time as the number of bounding boxes increases. Conversely, the Quantum Annealer (Figure 5.29c) demonstrates an incredibly flat time scalability. On Low and Medium-density instances, Gurobi is the absolute winner, taking only ~ 0.0151 seconds for low-density images and ~ 0.0887 seconds for medium instances. However, if we look at the behaviors for High-density instances, we unavoidably notice that the QA is twice as fast as the Gurobi solver, with an average of ~ 0.1777 seconds versus ~ 0.3475 seconds.

To verify that this speedup does not compromise detection quality, we plotted the mean Average Precision (mAP) evaluated for the three density groups in Figure 5.30. The QA (Figure 5.30c) perfectly matches the accuracy of both classical solvers (Figures 5.30a and 5.30b), particularly for the comprehensive configuration (Case 4).

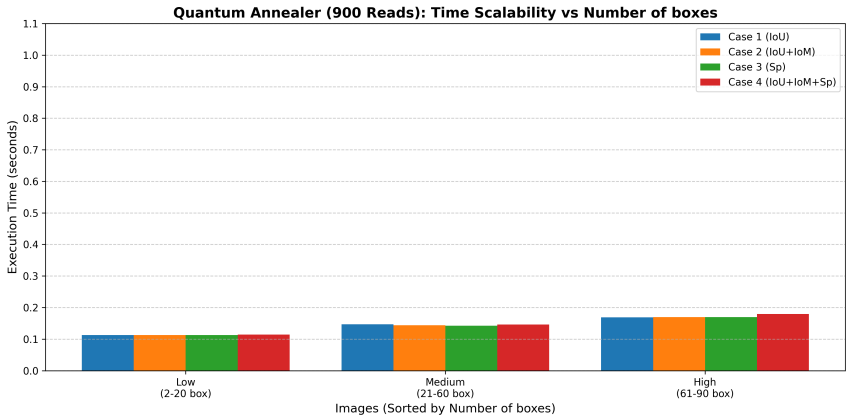
Finally, to provide a comprehensive numerical overview, Table 5.4 summarizes the global metrics evaluated across the entire 291-image subset. We can confirm that all metrics across different solvers underline the quality of our detections. Note that the QA drops the Mean Absolute Error (MAE) to approximately 1.1 and the Root Mean Square Error (RMSE) to roughly 2.3, while Gurobi and SA maintain higher error rates.



(a) Gurobi average execution times per density.

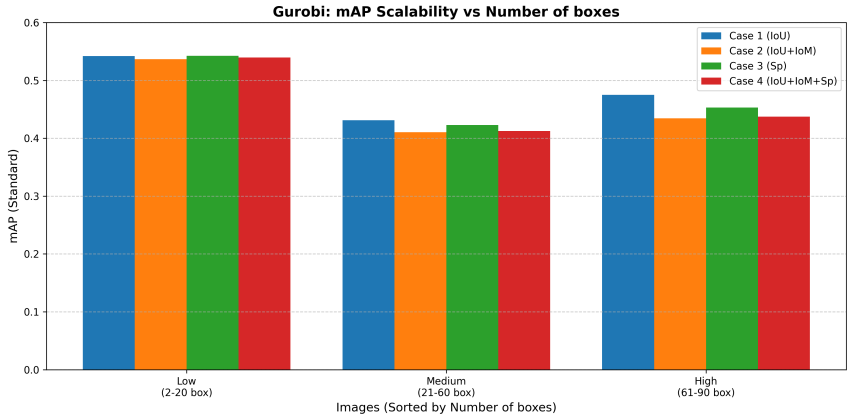


(b) Simulated Annealing average execution times per density.

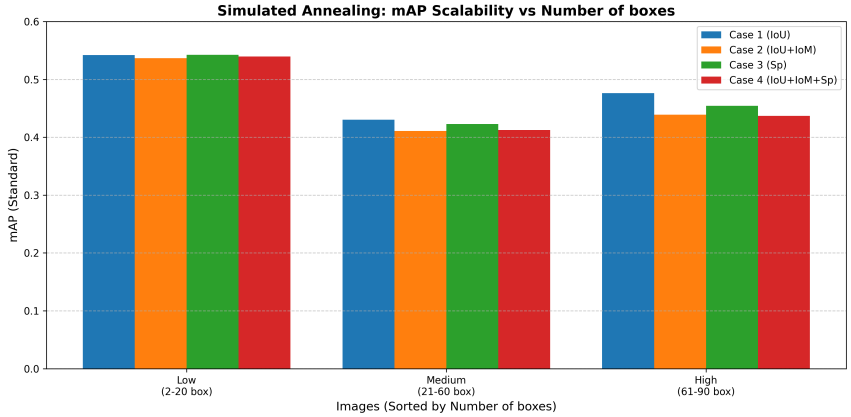


(c) Quantum Annealing (900 reads) average execution times per density.

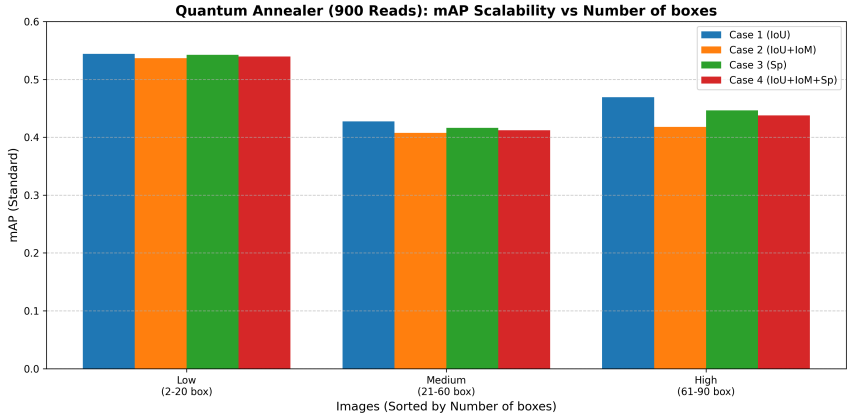
Figure 5.29: Average execution times sorted by density class (Low, Medium, High) evaluated across Gurobi, Simulated Annealing, and the Quantum Annealer (configured at 900 reads) for the validation subset (291 images). The Quantum Annealer, configured at 900 reads based on the previous sample analysis, confirms a speedup on High-density instances compared to both classical solvers.



(a) Gurobi average mAP scores per density.



(b) Simulated Annealing average mAP scores per density.



(c) Quantum Annealing (900 reads) average mAP scores per density.

Figure 5.30: Standard mAP_{std} scores sorted by density class, evaluated across Gurobi, Simulated Annealing, and the Quantum Annealer (configured at 900 reads) for the validation subset (291 images). The accuracy of the Quantum Annealer effectively matches the performance of classical solvers across all density levels.

Config.	Solver	F1	mAP	mAP(0.5)	mAR@10	mAR@100	MAE	RMSE
Case 1	Gurobi	0.7303	**0.4742**	0.7389	**0.4990**	0.5319	3.034	5.169
	SA	0.7303	0.4740	0.7388	0.4981	**0.5320**	3.034	5.169
	QA	**0.7311**	0.4716	**0.7399**	0.4968	0.5305	**1.168**	**2.479**
Case 2	Gurobi	0.7244	0.4580	0.7224	0.4846	0.5148	2.900	4.899
	SA	0.7253	**0.4584**	**0.7228**	**0.4859**	**0.5154**	2.900	4.899
	QA	**0.7261**	0.4545	0.7220	0.4846	0.5121	**1.076**	**2.274**
Case 3	Gurobi	**0.7312**	**0.4680**	0.7312	0.4934	**0.5255**	2.969	5.000
	SA	**0.7312**	0.4676	**0.7316**	**0.4936**	0.5250	2.969	5.000
	QA	0.7307	0.4643	0.7303	0.4913	0.5221	**1.100**	**2.336**
Case 4	Gurobi	0.7245	0.4595	0.7219	0.4856	0.5163	2.907	4.905
	SA	**0.7253**	**0.4596**	**0.7221**	0.4863	**0.5165**	2.907	4.905
	QA	0.7239	0.4586	0.7206	**0.4875**	0.5156	**1.083**	**2.282**

Table 5.4: Global performance metrics evaluated on the 291-image validation subset for all solvers (Gurobi, Simulated Annealing and Quantum Annealing with 900 reads) and penalty configurations. The best values for each metric within the same penalty configuration are highlighted in bold.

5.4 Comparison with State-of-the-art

In our experiments, our set up processes only the "person" class from the COCO dataset. We chose this category because it is well-known to be one of the hardest to detect in computer vision. However, since the code is highly modular, it can easily adapt to detect other classes by just changing the input filters when extracting Ground Truths and bounding boxes. To properly compare our results with the state-of-the-art, we have to consider why detecting people is so complex. Unlike rigid objects like cars or buses, which have a fixed shape and take up a predictable space, human bodies change shape constantly. People can sit down, fold their arms, or be partially occluded. Moreover, sometimes only small body parts like an arm or a hand appear in the whole photo, making the detection even more complicated. Even with these difficulties, our system reached a standard mean Average Precision (mAP) of about 0.46 on the validation subset. This score is consistent with the official benchmarks for this architecture, such as the standard baselines of Torchvision and Detectron2.

We underline that our goal was to show that the QUBO formulation can replace the classical Non-Maximum Suppression without dropping the accuracy. In the end, this comparison proves the success of our approach. We replaced the standard classical suppression with a quantum-ready mathematical model, and both localization and classification metrics remained basically similar.

Starting from the foundations of object detection [24] and recent works on QUBO-suppression [26], we kept the state-of-the-art baseline accuracy but gained a clear speedup on High-density images. This proves that quantum algorithms can effectively solve the computational limits of classical heuristics when dealing

with very crowded scenes using these QUBO-formulations.

Conclusions and Perspectives

This work explored the replacement of the standard Non-Maximum Suppression (NMS) algorithm with a QUBO-based mathematical formulation, directly evaluated on the D-Wave Quantum Annealer. The main goal was to investigate how classical and quantum architectures react to this type of problem, especially in highly crowded object detection images. The experimental results confirmed good solutions for all solvers. By transforming the suppression task into a global optimization problem, we successfully maintained the state-of-the-art accuracy of the neural network backbone. Most importantly, the Quantum Annealer demonstrated a significant advantage in scaling. While classical solvers like Gurobi and Brute Force suffer an exponential time growth as the number of predicted bounding boxes increases, the quantum execution time remains stable. On images with a high density of bounding boxes, the D-Wave hardware proved to be twice as fast as classical algorithms for this specific QUBO formulation. Furthermore, we showed that even when the QPU settles into a degenerate ground state, the resulting bounding box suppression is still comparable to classical methods.

Although these results are promising, this work represents a proof-of-concept with ample room for future developments. From an algorithmic perspective, our current pipeline was restricted to the "person" class. A natural next step is expanding the formulation to multi-class detection. Identifying people, vehicles, and objects within the same matrix will require a new penalty arrangement with different hyperparameters.

Additionally, we recall that the QUBO formulation acts as a post-processing module, so its final performance heavily depends on the quality of the initial predictions. Integrating newer, state-of-the-art neural network architectures during the pre-processing phase could yield more precise bounding box coordinates and more reliable confidence scores, ultimately improving the overall system accuracy.

However, moving to a multi-class scenario introduces hundreds of overlapping bounding boxes, increasing the problem size. These physical constraints may affect both classical and quantum solvers due to memory and qubit limitations. A "new frontier" that may be explored is the QUBO partitioning [2], which decomposes the global problem into smaller sub-problems that can be solved by current algorithms and quantum topologies.

Alongside these algorithmic strategies, the continuous physical evolution of quantum hardware will ultimately make the difference. Future generations of QPUs featuring advanced topologies and higher qubit connectivity will reduce the embedding overhead and the waste of physical qubits.

Combining these further techniques will be essential to transition this academic research into real-world applications, paving the way for quantum-assisted computer vision.

Bibliography

[1] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS – Improving Object Detection With One Line of Code. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5561–5569, 2017.

[2] Michael Booth, Steven P Reinhardt, and Aidan Roy. Partitioning Optimization Problems for Hybrid Classical/Quantum Execution. Technical report, D-Wave, 2017.

[3] Kelly Boothby, Andrew D. King, and Jack Raymond. Zephyr Topology of D-Wave Quantum Processors. Technical report, D-Wave Systems, 2021.

[4] COCO Consortium. COCO - Common Objects in Context: Detection Evaluation. `https://cocodataset.org/#detection-eval`, 2026.

[5] Philippe Codognet, Daniel Diaz, and Salvador Abreu. Quantum and Digital Annealing for the Quadratic Assignment Problem. In *IEEE International Conference on Quantum Software (QSW)*, 2022.

[6] D-Wave Systems Inc. *dwave-neal Documentation*, 2021.

[7] D-Wave Systems Inc. *Quantum Annealing Documentation*, 2024.

[8] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88:303–338, 2010.

[9] A. M. Ferreiro, J. A. García, J. G. López-Salas, and C. Vázquez. An efficient implementation of parallel simulated annealing algorithm in GPUs. *Journal of Global Optimization*, 57(3):863–890, 2024.

[10] Fred Glover, Gary Kochenberger, and Yu Du. Quantum Bridge Analytics I: A Tutorial on Formulating and Using QUBO Models. Technical report, National Academies of Sciences, Engineering, and Medicine, 2022.

[11] Daniel R. Greening. Parallel simulated annealing techniques. *Physica D: Nonlinear Phenomena*, 42(1):293–306, 1990.

[12] Gurobi Optimization LLC. Gurobi optimizer reference manual. Technical report, Gurobi Optimization, 2026.

[13] Jonathan Hui. mAP (mean Average Precision) for Object Detection. `https://jonathan-hui.medium.com/{mAP}-mean-average-precision-for-object-detection-45c121a31173`, 2018.

[14] Gary Kochenberger, Fred Glover, Haibo Wang, and Gary A Kochenberger. QUBO: The Quadratic Unconstrained Binary Optimization Problem. Technical report, ResearchGate Preprint, 2024.

[15] Donghoon Lee, Geonho Cha, Ming-Hsuan Yang, and Songhwai Oh. Individualness and Determinantal point Process for Pedestrian Detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9910 LNCS:V, 2016.

[16] Junde Li and Swaroop Ghosh. Quantum-soft QUBO Suppression for Accurate Object Detection. *arXiv preprint arXiv:2007.13992*, 2020.

[17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE/CVF conference on computer vision and pattern recognition*, 2017.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv preprint arXiv:1405.0312*, 2015.

[19] Satoshi Morita and Hidetoshi Nishimori. Mathematical Foundation of Quantum Annealing. *arXiv preprint arXiv:0806.1859*, 2008.

[20] Davide Pastorello and Enrico Blanzieri. Quantum annealing learning search for solving QUBO problems. *Quantum Information Processing*, 18(10), 2019.

[21] Abraham P. Punnen. *The Quadratic Unconstrained Binary Optimization Problem Theory, Algorithms, and Applications*. Springer International Publishing, 2022.

[22] Finley Alexander Quinton, Per Arne Sevle Myhr, Mostafa Barani, Pedro Crespo del Granado, and Hongyu Zhang. Quantum annealing applications, challenges and limitations for optimisation problems compared to classical solvers. *Scientific Reports*, 15, 2025.

[23] F N Rashadyfa. A Case Study in Optimization of Brute-Force Algorithm: Finding Longest Substring With Unique Characters. Technical report, Institut Teknologi Bandung, 2022.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv preprint arXiv:1506.01497*, 2016.

[25] Peter J. M. van Laarhoven and Emile H. L. Aarts. *Simulated Annealing: Theory and Applications*. Springer Netherlands, 1987.

[26] Keiichiro Yamamura, Toru Mitsutake, Hiroki Ishikura, Daiki Kusuhara, Akihiro Yoshida, and Katsuki Fujisawa. Enhancing Quantum-ready QUBO-based Suppression for Object Detection with Appearance and Confidence Features. *arXiv preprint arXiv:2502.02895*, 2025.

Ringraziamenti

Come prima cosa, ci tengo a ringraziare l'azienda E4 Computer Engineering per avermi fatto svolgere questo tirocinio che si è rivelato un'esperienza formativa preziosa. Una menzione speciale va a Gabriella per avermi seguita costantemente in questi mesi, insegnandomi tanto e soprattutto indirizzandomi verso un'analisi consapevole.

Desidero inoltre esprimere la mia profonda gratitudine ai Professori De Palma e Pastorello per avermi seguita in questo lavoro di tesi. Aver potuto concludere il mio percorso universitario lavorando sotto la vostra supervisione è stato per me un grande privilegio.