Mean Response-Time Minimization of a Soft-Cascade Detector

Francisco Rodolfo Barbosa-Anda[1,2], Cyril Briand[1,2], Frédéric Lerasle[1,2] and Alhayat Ali Mekonnen[1]
[1] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[2] *Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France*

Keywords: Mathematical Programming, Soft-Cascade, Machine Learning, Object Detection.

Abstract: In this paper, the problem of minimizing the mean response-time of a soft-cascade detector is addressed. A soft-cascade detector is a machine learning tool used in applications that need to recognize the presence of certain types of object instances in images. Classical soft-cascade learning methods select the weak classifiers that compose the cascade, as well as the classification thresholds applied at each cascade level, so that a desired detection performance is reached. They usually do not take into account its mean response-time, which is also of importance in time-constrained applications. To overcome that, we consider the threshold selection problem aiming to minimize the computation time needed to detect a target object in an image (i.e., by classifying a set of samples). We prove the NP-hardness of the problem and propose a mathematical model that takes benefit from several dominance properties, which are put into evidence. On the basis of computational experiments, we show that we can provide a faster cascade detector, while maintaining the same detection performances.

1 INTRODUCTION

Visual object detection is of utmost importance in the Computer Vision community with applications ranging from target tracking in video surveillance (Breitenstein et al., 2011), image indexing and retrieval (Zhang and Alhajj, 2009), intelligent robotic systems (Ess et al., 2010), Advanced Driver Assistance Systems (ADAS) (Gerónimo et al., 2010), etc. Recently, the interest has significantly increased owing to the improvements in computational resources as attested by the number of published works, e.g., (Zhang et al., 2013; Dollár et al., 2012; Gerónimo et al., 2010), proposed challenges, e.g., ImageNet (Russakovsky et al., 2015) and the Pascal Challenge (Everingham et al., 2010), and the dynamics of contributions.

The objective in visual object detection is to detect and localize the presence and position (with correct scale) of target objects in the image. The objects can be anywhere in the image. In the literature, the most successful visual object detection approach is based on what is known as a sliding-window technique (Viola and Jones, 2004). This technique searches for an object on all possible positions and scales in the image using a trained classifier – see illustration and 1% of the total sampled windows in Figure 1. This is and has been the most successful approach without any prior (context) information (Dollár et al., 2012). Unfortunately, this is computationally demanding and creates a bottleneck for real-time implementations. Given the proportion of positive and negative samples in an image, which is typically a staggering one to thousands (Figure 1), researchers have investigated several classifier architectures that employ a cascade structure to discard as much negative samples as early as possible. This boosts the computation gain as there will be less samples to test further along the cascade. Examples include, AdaBoost variants (Dollár et al., 2014), ad-hoc manually constructed cascades (Pan et al., 2013), and tree ensembles (e.g., Random Forest) (Tang et al., 2012).

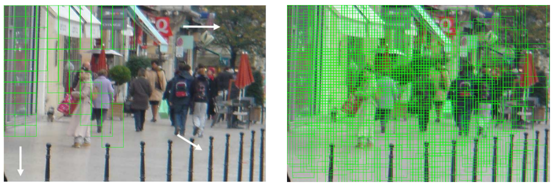
Figure 1: Sliding-window illustration (left) and randomly sampled 1% of all windows (right) on a 769 × 516 sample image taken from the public INRIA person dataset.

Of all the classifiers, cascade classifiers constructed using AdaBoost have received significant attention and have been widely used for several object detection tasks (Viola and Jones, 2004). AdaBoost

Barbosa-Anda, F., Briand, C., Lerasle, F. and Mekonnen, A.
Mean Response-Time Minimization of a Soft-Cascade Detector.
DOI: 10.5220/0005700702520260
In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, pages 252-260
ISBN: 978-989-758-171-7

learns a classifier model of an object as a linear sum of weighted rudimentary weak classifiers in a supervised manner given labeled positive and negative training samples. Ideally, each weak classifier, e.g., a decision tree, is associated with a unique feature, e.g., difference of gradient distribution in a specific local patch within the candidate window (hence, it is common to find the terms weak classifiers and features used interchangeably). Usually, several features, thus weak classifiers, are extracted/trained and AdaBoost iteratively selects and builds a strong classifier using only a handful of these weak classifiers. Classically, AdaBoost has been used in a cascade arrangement composed of several stages, each stage containing a single strong classifier trained with AdaBoost (Viola and Jones, 2004; Zhu et al., 2006). The main interest of the cascade arrangement is to reject as much of the negative windows as early as possible, thereby (1) decreasing the computation time, and (2) decreasing the False Positive Rate (FPR). Since each cascade stage aggregates the weighted score of the constituent weak classifiers and thresholds the aggregate to label each sample as positive (which is passed to the next level) or as negative (which is rejected), it is referred as a *hard-cascade*. Post-training refinements to further tune the performance of the classifier to meet detection performance requirements are possible by adjusting the thresholds used at each stage of the cascade level. Recently, new variants called *soft-cascades* have burgeoned (Zhang and Viola, 2008; Bourdev and Brandt, 2005). The main idea of soft-cascade is instead of having separated cascade stages, to have one stage with a single strong classifier and then to threshold each sample response after each weighted weak classifier evaluation. These thresholds are learned after the complete training of the strong classifier in a kind of calibration phase. In both cases, AdaBoost trains a classifier solely to fulfill detection performance requirements without any computation time consideration. However, in real-time systems using a sliding window detection approach, it is imperative to consider computation time aspects explicitly.

Computation time of a cascade classifier can be predominantly decreased in two ways: (1) By using a feature selection mechanism with computation time consideration so that cheap features are used in the initial stages of the cascade and costly, but more discriminatory, ones at later stages; for example, the Binary Integer Programming (BIP) based feature selection framework proposed in (Mekonnen et al., 2014) and the ad-hoc weighted computation time based feature selection approach in (Jourdheuil et al., 2012). And, (2) by maximizing cascade stage rejection ratios, in which the rejection thresholds on each cascade stage are set to reject as much negative windows as early as possible, notably the soft-cascade paradigm proposed in (Zhang and Viola, 2008; Bourdev and Brandt, 2005). The first approach is suitable when considering different classes of features with varying computation time and detection performance, whereas the later is suitable for similar classes of features having the same computation time but different detection performance. The work presented in this paper makes its contributions in the vein of the second approach.

As highlighted, soft-cascades use classical AdaBoost and adjust the rejection thresholds post-facto (after the classifier is trained) to improve the overall computation time without (possible) loss of detection performance. In line with this, in this paper, we propose a novel optimization framework based on Binary Integer Programming (BIP) to determine optimal rejection threshold values for a trained AdaBoost classifier that minimizes the overall computation time without worsening its detection performance. The proposed framework achieves as much as a 22% relative computation time gain over (Zhang and Viola, 2008) under the same TPR conditions with a pedestrian detector trained on the INRIA person dataset (Dalal and Triggs, 2005)[1]. Eventually, this work makes three important contributions: (1) it proves that learning a soft-cascade explicitly minimizing the incurred computation time is NP-hard; (2) it proposes a novel BIP based optimization framework for solving this problem; and (3) it demonstrates experimentally and comparatively the viability of the framework on a relevant application, namely pedestrian detection, using publicly available real life dataset.

The rest of this paper is structured as follows: section 2 briefly presents the AdaBoost classifier and notions of soft-cascade; then, the problem of mean response time minimization is highlighted in section 3 and its complexity is studied. Section 4 focuses on problem modeling and linear programming formulations. Relevant experimental evaluations, results, and associated discussions are detailed in section 5. Finally, concluding remarks are provided in section 6.

2 ADABOOST AND SOFT-CASCADE

This work deals with the construction of a cascade classifier used in object detection applications. This section provides a brief overview of discrete AdaBoost and soft-cascade. The presentation on soft-

[1]http://pascal.inrialpes.fr/data/human/

cascade focuses on the Direct Backward Pruning (DBP) algorithm (Zhang and Viola, 2008), which is the most prevalent technique used to learn one.

Discrete AdaBoost, one instance of the Boosting classifier variants, builds a strong classifier as a linear combination (weighted voting) of a set of weak classifiers. Suppose we have a labeled training set $\{(x_n,y_n)\}_{\{n=1,...,N\}}$ where $x_n \in X$, $y_n \in Y = \{0,1\}$, and N denotes the number of training samples. Given a set of weak classifiers (features) $\mathcal{F} = \{h_l\}_{\{l=1,...,|\mathcal{F}|\}}$, where $|\mathcal{F}|$ denotes the total number of weak classifiers that can assign a given example a corresponding label, i.e., $h : x \rightarrow y$, discrete Adaboost constructs a strong classifier of the form $\mathcal{H}(x) = \sum_{l=1}^{L} \alpha_l h_l(x)$, with $sign(\mathcal{H}(x) - \theta_L)$ determining the class label. θ_L is a threshold value tuned to set the classifier's operating point (TPR and FPR). The l indexes connote the sequence of the weak classifiers and this specific classifier has a total of L selected weak classifiers. The specific weak classifier to use at each iteration of this boosting algorithm, h_l, and the associated weighting coefficients, α_l, are derived minimizing the exponential loss, which provides an upper bound on the actual 1/0 loss (Schapire, 2003).

As previously mentioned, instead of directly using the AdaBoost trained strong classifier, a set of rejection threshold values are learned to threshold each sample response after each weighted classifier evaluation. The strong classifier along with the rejection thresholds form a soft-cascade. The most widely used algorithm to learn the rejection thresholds is the Direct Backward Pruning (DBP) algorithm (Zhang and Viola, 2008). Considering the cumulative score of a sample x_n at the l^{th} weak classifier as $S_{n,l} = \sum_{u=1}^{l} \alpha_u h_{n,u}$ (where $h_{n,u} := h_u(x_n)$), DBP sets the threshold θ_l according to Equation (1) – i.e., to the minimum score $S_{n,l}$ registered by any of the positive samples that have a final score $S_{n,L}$ above the final threshold θ_L.

$$\theta_l = \min_{\{n|S_{n,L}>\theta_L, y_n=1\}} S_{n,l} \tag{1}$$

Although this framework has been successfully used in several detection applications, e.g., (Zhang and Viola, 2008; Dollár et al., 2012), it is not optimal in terms of minimizing incurred computation time. Hence, we propose a novel soft-cascade construction algorithm based on BIP dubbed Mean-Cascade Response-time Minimization Problem (MSCRMP) and show it leads to a faster cascade over DBP under the same TPR conditions. In the following sections, the proposed MSCRMP algorithm is extensively presented and demonstrated via a people detection application using realistic public dataset.

3 MEAN RESPONSE TIME MINIMIZATION

3.1 Problem Statement

This section defines more formally the MSCRMP, which is studied in this paper. It involves a trained cascade having L weak classifiers with a training set composed of N samples, partitioned into J positive and K negative samples (i.e., $\mathbf{N} = \mathbf{J} \cup \mathbf{K}$). The notations n, j and k will be further used for designating a sample (either positive or negative), a positive one and a negative one, respectively. The weak classifier located at level l has a positive cost c_l, which corresponds to the computation time needed to analyze one single sample. It also has a positive weight α_l, which reflects its importance in the cascade. We refer to $S_{n,l} = \sum_{u=1}^{l} \alpha_u h_{n,u}$ as the score of sample n at level l where $h_{n,l}$ is the known weak-classifier response, which equals 1 when the weak classifier sees n as positive.

The MSCRMP aims at determining, at each level l, a threshold θ_l such that if $S_{n,l} < \theta_l$ then the sample n will be rejected from the cascade at level l (it will not pass through the next cascade levels $u > l$). Conversely, if $S_{n,l} \geq \theta_l$) then n will pursue to the next level $l+1$. Obviously, when a sample is rejected at level l, a computational time saving is obtained that equals $\sum_{u=l+1}^{L} c_u$.

Provided that a minimum number TP of positive samples should have never been rejected at any cascade level, the objective is to find a threshold vector $\Theta = \{\theta_1, \ldots, \theta_L\}$ that minimizes the total computation time (or equivalently, that maximizes the total saved computation time). Note that, while this objective only considers the training set, this one is supposed to be statistically representative of any other sample set, as commonly assumed in machine learning. Therefore the minimization of the total computation time (for the training set) can be viewed as equivalent to the minimization of the mean response time (for any unknown sample set).

In the particular case where $\alpha_l = 1\ \forall l$, considering a five-levels cascade ($L = 5$), the diagram provided in Figure 2 illustrates the evolution of the score of six samples (positives ones are drawn in blue circles, negatives in red squares). The scores, which take their value in the discrete set $\{0,1,2,3,4\}$ in this example, are displayed on the vertical axis. For this particular problem instance, if one assumes a desired minimum true-positive rate $TPR = 50\%$, only two possible threshold vectors Θ will be feasible: $\Theta_a = \{1,1,1,2,3\}$ and $\Theta_b = \{0,1,2,3,4\}$. Assuming $c_l = 1\ \forall l$, the use of Θ_a gives a total computation

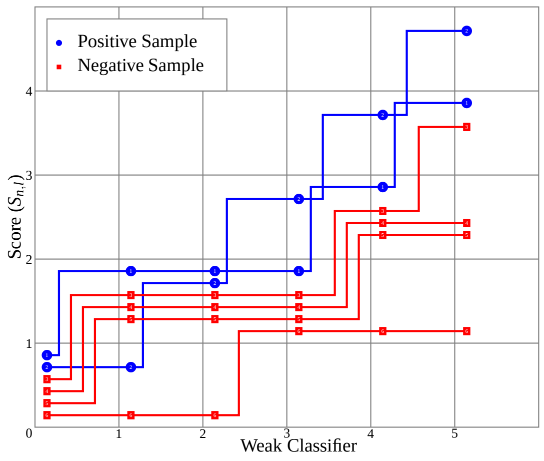

Figure 2: Example with 5 weak classifiers, all cost $c_l = 1$, all weight $\alpha_l = 1$, 2 positive samples and 4 negative samples.

time equals to $6+4+4+4+4 = 22$, while the one associated with Θ_b is $6+6+5+1+1 = 19$, which is optimal.

Let us recall that the performance of a soft cascade is not only characterized by the true-positive rate TPR but also by the false-positive rate FPR, which gives the percentage of negative sample never rejected at any cascade level. Obviously, this rate should be as low as possible and negative samples should be rejected the earliest as possible as in practical life $|\mathbf{J}| << |\mathbf{K}|$. Consequently, as soon as the decision of rejection of a positive sample j having score $S_{j,l}$ is made at one level l, one should also reject at level l any negative sample k having a score $S_{k,l} \leq S_{j,l}$. In other words, the FPR can be seen as the consequence of the chosen feasible threshold vector Θ. Moreover, minimizing the total computation time naturally tends to minimize the FPR. Coming back to our previous example, Θ_a conserves one false positive, while the optimal Θ_b threshold vector eliminates all of them.

3.2 Problem Complexity

We prove the following proposition.

Theorem 3.1. *MSCRMP is NP-hard.*

Proof. The proof is based on a reduction from Subset-Sum Problem (SSP). An instance of SSP is a pair (Σ, t), where $\Sigma = \{\sigma_1, \ldots, \sigma_R\}$ is a set of R positive integers and t (the target) is a positive integer. The decision problem consists in determining whether there exists a subset σ^* of Σ whose sum equals t. SSP is known to be NP-complete in the ordinary sense (Garey and Johnson, 1979).

First, for any MSCRMP, it is obvious that given a threshold vector Θ, one can check in polynomial time whether it is feasible and determine the resulting total computation time. So MSCRMP is in NP. Considering now any SSP instance, we build up a MSCRMP as follows. The soft-cascade is composed of $L = 2R$ levels. We set $\alpha_{2u} = \alpha_{2u-1} = 1$ (the scores are integer) and $TP = |\mathbf{J}| - t$. The costs are such that $c_L = 1$ and $c_l = 0\ \forall l < L$. The set $\mathbf{J}$ is partitioned into R subsets (i.e., $\mathbf{J} = \{\mathbf{J}_1, \ldots, \mathbf{J}_R\}$) such that: i) $|\mathbf{J}_i| = \sigma_i$ and ii) the score of a sample $j \in \mathbf{J}_i$ equals $l \div 2 - 1$ when $l = 2i - 1$ and $l \div 2$ at any other level, $\forall l = 1, \ldots, 2R$.

Under those assumptions, let us make a few observations. First, at an even cascade level $l = 2i$, all the samples have the same score value i, while for an odd level $l = 2i - 1$, only the samples belonging to $\mathbf{J}_i$ have score $i - 1$ (i for all the other samples). Moreover, as $TP = |\mathbf{J}| - t$, any feasible threshold vector can never reject any sample at an even level, as it will discard the whole set $\mathbf{J}$. Additionally, there are only two possible decisions at an odd level $l = 2i - 1$: either $\theta_l > i - 1$ and the set $\mathbf{J}_i$ is rejected, or $\theta_l \leq i - 1$ and all remaining samples are conserved. Last, if $\mathbf{J}_i$ is rejected at level $l = 2i - 1$, the time saving will equal $|\mathbf{J}_i| \sum_{u=l+1}^{L} c_u = \sigma_i$. As the total saving has to be maximized, the reduced MSCRMP aims at maximizing the total number of positive samples rejected, provided it remains not larger than t. We prove below that there exists a threshold vector Θ^* for this MSCRMP-instance if and only if the SSP instance is a YES-instance.

($\Leftarrow$) Let us consider a feasible solution $\Sigma^A \subseteq \Sigma$ of an SSP instance such that $\sum_{\sigma_i \in \Sigma^A} \sigma_i = t$. Clearly, in the corresponding reduced MSCRMP instance, if we consider a threshold vector Θ^A such that i) $\theta^A_{2i-1} = \theta^A_{2i} = i$ when $\sigma_i \in \Sigma^A$ and ii) $\theta^A_{2i-1} = \theta^A_{2i} = i - 1$ when $\sigma_i \notin \Sigma^A$, then only the positive samples in the set $\cup_{i|\sigma_i \in \Sigma^*} \mathbf{J}_i$ will be rejected, which exactly has t members. Consequently, Θ^A is a feasible and optimal solution of the MSCRMP-instance. ($\Rightarrow$) Now, consider an optimal (feasible) solution Θ^* of one reduced MSCRMP and a solution Σ^* of the initial SSP instance such that $\sigma_i \in \Sigma^*$ if and only if $\mathbf{J}_i$ is rejected (i.e., $\theta^*_{2i-1} < i - 1$). If the total number z^* of positive samples rejected equals t then Σ^* will obviously be feasible for the initial SSP, which means it is a YES-instance. Conversely, if the number z^* of positive samples rejected is strictly lower than t then Σ^* will obviously not be feasible for SSP. Moreover, as the number of positive samples is maximized, there is no way for increasing z^* while preserving the true positive rate $TP = |\mathbf{J}| - t$, which means that the initial SSP is a NO-instance. $\square$

4 PROBLEM MODELING

4.1 Sample Rejection Based Model

First let us state the following property.

Proposition 4.1. *The solution set of any MSCRMP instance can be restricted to threshold vectors* Θ *such that i) at any level* l, $\exists n : \theta_l = S_{n,l}$ *and ii)* $\theta_l \leq \theta_{l+1}$.

Proof. Assume there exists a level l that does not respect the property i). Now let consider a sample n reaching level l and having the lowest score, provided that $\theta_l < S_{n,l}$. It is obvious that θ_l can be increased up to $S_{n,l}$ without any modification of the rejected samples. Moreover, if property ii) is not met (i.e., $\theta_{l+1} < \theta_l$), then θ_{l+1} can be increased up to θ_l still without any consequence on the rejected samples. $\square$

From that straightforward property, one can issue a Binary Integer Program (BIP), which does not make use of any θ_l variables in its formulation. The binary variables $x_{n,l}$ that equal 1 if $\theta_l > S_{n,l}$ for the *first* time are introduced. The objective function (2) maximizes the compute time saving, which is linearly expressed as a function of the $x_{n,l}$ variables. Constraints of type (3) enforce a sample to be rejected only once. The desired TPR is obtained thanks to constraint (4). Eventually, constraints (5) describe the relationships between samples and scores: it states that whenever $x_{n,l} = 1$, all samples having a score lower or equal to $S_{n,l}$ at level l should have been rejected at a level $u \leq l$. From an optimal solution, the vector of thresholds Θ can be easily computed by setting $\theta_l = \min_{n \in N : \sum_{u=1}^{l} x_{n,u} = 0} S_{n,l} \ \forall l$.

First BIP Formulation (BIP1)

Maximize

$$\sum_{l=1}^{L} \sum_{n=1}^{N} \left[x_{n,l} \left(\sum_{u=l+1}^{L} c_u \right) \right] \tag{2}$$

Subject to

$$\sum_{l=1}^{L} x_{n,l} \leq 1 \ \forall n \tag{3}$$

$$|\mathbf{J}| - \sum_{l=1}^{L} \sum_{n \in \mathbf{J}} x_{n,l} \geq TP \tag{4}$$

$$x_{n,l} \leq \sum_{u=1}^{l} x_{v,u} \ \forall (n,l) \text{ and } \forall v | S_{v,l} \leq S_{n,l} \tag{5}$$

$$x_{n,l} \in \{0,1\} \ \forall (n,l) \tag{6}$$

Let us highlight that the set of constraints of type (5) being quite large, only small problem instances can be solved. In the next section, another MILP formulation is proposed that outperforms this one both in terms of computation time and instance-size capacity.

4.2 Threshold Space Analysis

From the analysis of the weight α_l of each weak classifier, a score tree with 2^L nodes can be constructed such that any node (l,s) corresponds to one combination of weights and has two adjacent nodes $(l+1,s)$ and $(l+1, s+\alpha_{l+1})$. An example of such a tree is given in Figure 3. The instance is a soft-cascade with four weak classifiers trained by AdaBoost having weights $\alpha = \{2.8498, 4.3778, 3.9534, 4.6368\}$. Any path from node $(0,0)$ to a node of level L corresponds to a possible score evolution of one sample. Of course, given a sample set, any score evolution is not possible and for the case depicted in Figure 3, score-classes containing no samples are represented in white (the others being darkened). We further refer to $\mathcal{C}_{(l,s)}$ as the set of samples n such that $S_{n,l} = s$.

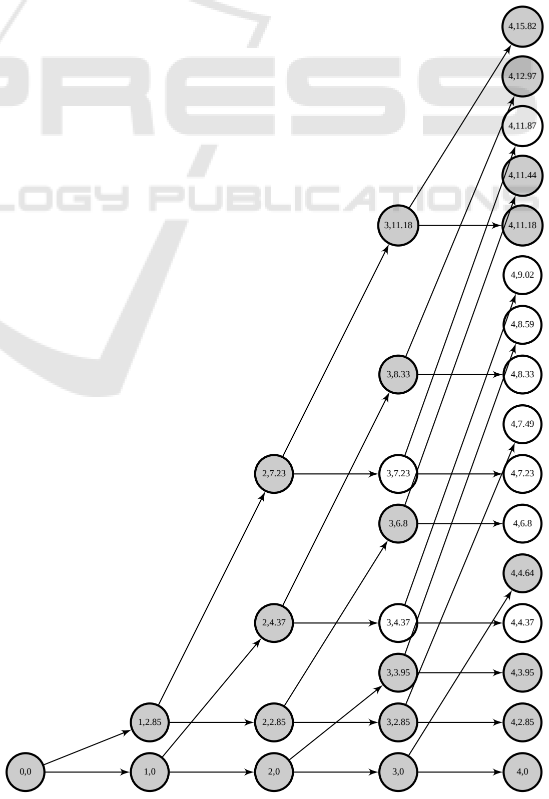

Figure 3: A score tree.

From such a score tree, a threshold tree $\mathcal{T}(V,E)$ can be constructed in its turn having the same set V of nodes than the score tree and such that any node

$(l,s) \in V$ of level l is connected by an arc $e \in E$ to a node $(l+1,s') \in V$ if and only if $s' \geq s$, as illustrated in Figure 4 for the example of Figure 3. According to Proposition 4.1), any solution of the MSCRMP (i.e., any threshold vector Θ) corresponds to a specific path from node $(0,0)$ to a node of level L in $\mathcal{T}$ such that, if node (l,s) is traversed then $\theta_l = s$. The number of different paths is clearly exponential. Anyway it is possible to drastically prune this path number using the following dominance property.

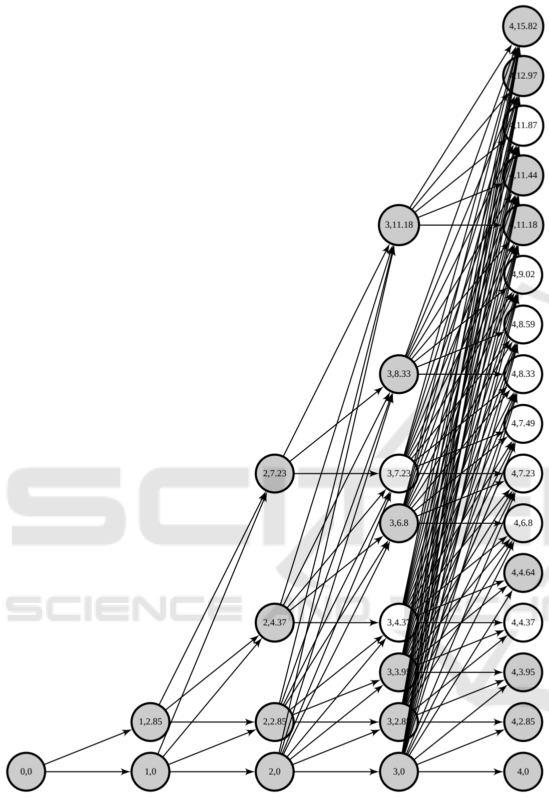

Figure 4: A threshold tree.

Proposition 4.2. *Any arc $e \in E$ of the threshold tree $\mathcal{T}$ linking node (l,s) with $(l+1,s')$ can be deleted whether $\forall n \in \mathcal{C}_{(l,s)}$ it holds $s' > S_{n,l+1}$.*

Proof. Let us consider a path passing through node (l,s). Clearly all the (remaining) samples belonging to $\mathcal{C}_{(l,s)}$ are not discarded at level l as $S_{n,l} = \theta_l$. Now if the path continues from (l,s) to $(l+1,s')$ with $s' > S_{n,l+1}$, $\forall n \in \mathcal{C}_{(l,s)}$, all the remaining samples belonging to $\mathcal{C}_{(l,s)}$ will be rejected at level $l+1$. Consequently, it should have been more profitable with respect to the objective function to have rejected them at level l. In other words, any path passing through arc $e = ((l,s),(l+1,s'))$ is dominated by at least one path passing through a node (l,s'') with $s'' > s$. $\square$

In addition to the previous dominance property, considering the TPR target, one can further prune $\mathcal{T}$ by removing the paths that necessarily lead either to poor $100\% - TPR$-solutions or to some too low TPR values. A poor $100\% - TPR$-solution is a threshold vector offering a $TPR = 100\%$ such that there still exist some negative samples k that could have been rejected by modifying the threshold vector, still preserving the $TPR = 100\%$ value. It is clear that rejecting them will improve the FPR value and the total computation time. Poor $100\% - TPR$-solutions can be easily filtered by removing from $\mathcal{T}$ any node $(l,s) \in V$ such that $s < \min_{j \in \mathbb{J}} S_{j,l}$. On the other hand, one can also remove any node $(l,s) \in V$ such that the sum of positive samples having a score $S_{j,l} \geq s$ is lower than TP. We further refer to $\mathcal{T}_p(V_p, E_p)$ as the pruned threshold tree obtained after applying the previous reduction rules.

Considering the MSCRMP example depicted in Figure 4, one obtains the pruned threshold tree represented in Figure 5, which "only" has 24 different paths.

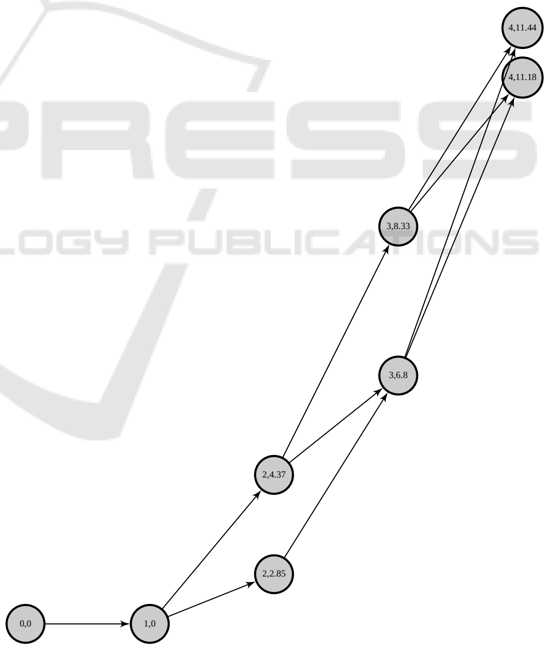

Figure 5: Pruned threshold tree.

Aiming at taking benefit from this pruned threshold-tree, we propose below another BIP (BIP2) flow formulation. As in the first BIP, we still consider the binary $x_{n,l}$ variable that equals 1 whether sample n is such that $S_{n,l} < \theta_l$ for the first time. Additionally, we now introduce the $\varphi_{n,l}$ and $\psi_{s,t,l}$ binary variables. On the one hand, $\varphi_{n,l} = 1$ whether there exists a weak classifier at level $u \leq l$ such that $S_{n,u} < \theta_u$. Clearly, variables $x_{n,l}$ and $\varphi_{n,l}$ are linked together by the relation $\varphi_{n,l} = \varphi_{n,l-1} + x_{n,l}$, which is valid for any sample n and level l. On the other hand, $\psi_{s,t,l}$ is a flow variable: $\psi_{s,t,l} = 1$ whether the arc $e \in E_p$ between node $(l,s) \in V_p$ and $(l+1,t) \in V_p$ of $\mathcal{T}_p$ is selected into the solution. The variables $\psi_{s,t,l}$ should be chosen such that they define a path in $\mathcal{T}_p$, i.e., $\sum_{(l-1,t)\in\sigma^{-1}_{(l,s)}} \psi_{t,s,l-1} = \sum_{(l+1,t)\in\sigma_{(l,s)}} \psi_{s,t,l}$, $\forall(l,s)$. The three kinds of variable are linked by the following relation: $x_{n,l} \leq \sum_{t|S_{n,l}<t} \sum_{(l+1,t)\in\sigma_{(l,s)}} \psi_{s,t,l} \leq \varphi_{n,l}$. It models that at any level l, a sample n can be rejected only if the selected score class (l,s) i such that $s > S_{n,l}$. As it can be observed, this second BIP presents a reasonable amount of constraints, although new variables have been introduced. The effectiveness of this formulation is discussed in the next section.

Second Binary Integer Program Formulation (BIP2)

Maximize

$$\sum_{l=1}^{L}\sum_{n=1}^{N}\left[x_{n,l}\left(\sum_{u=l+1}^{L} c_u\right)\right] \quad (7)$$

Subject to

$$\sum_{n\in\mathbf{J}} x_{n,L+1} \geq TP \quad (8)$$

$$\varphi_{n,l} = \varphi_{n,l-1} + x_{n,l} \;\forall(n,l) \quad (9)$$

$$\psi_{0,0,1} + \psi_{0,1,1} = 1 \quad (10)$$

$$\sum_{(l-1,t)\in\sigma^{-1}_{(l,s)}} \psi_{t,s,l-1} = \sum_{(l+1,t)\in\sigma_{(l,s)}} \psi_{s,t,l} \;\forall(s,l) \quad (11)$$

$$x_{n,l} \leq \sum_{t|S_{n,l}<t}\;\sum_{s|(l+1,t)\in\sigma_{(l,s)}} \psi_{s,t,l} \leq \varphi_{n,l} \;\forall(n,l) \quad (12)$$

$$x_{n,l} \in \{0,1\}, \varphi_{n,l} \in \{0,1\} \;\forall(n,l) \quad (13)$$

$$\psi_{s,t,l} \in \{0,1\} \;\forall(s,t,l) \quad (14)$$

5 EXPERIMENTS

5.1 Problem Instances

To validate our proposed models, a set of MSCRMP instances are created using training images taken from the public INRIA person dataset (Dalal and Triggs, 2005). The training is carried out using Piotr's Computer Vision Matlab Toolbox (Dollár, 2014). Each person detection classifier is trained with discrete AdaBoost using Histogram of Oriented Gradient (HOG) features coupled with decision trees as weak classifiers. The HOG features computed have the same computation time (along with the associated decision trees). Thus, c_l is set to 1 at any cascade level l. Once the soft-cascade is trained, every MSCRMP instance can be characterized by the triplet (L,J,K), namely the number of levels of the cascade, the number of positive and negative samples of the training set. In our benchmark, L is picked from the set {4, 8, 16, 32, 64, 128, 256} and J from {16, 32, 64, 128, 256, 512, 1024, 2048}. For K, two cases are considered either $K = J$ or $K = 3J$, further referred as (1:1) and (1:3) class of instances, respectively. Note that all the instances that do not respect $J \geq 2L$ are removed, as this condition is an AdaBoost requirement for the training. In total, 82 instances are generated (41 for each (1:1) and (1:3) class). For each instance, a soft-cascade is trained using DBP, BIP1 and BIP2 for every TPR value in {100%, 97.5%, 95%, 92.5%, 90%, 87.5%, 85%, 82.5%, 80%}. Let us highlight that, even though some of these instances can be considered huge with respect to the number of variables and constraints, an efficient MILP solver is commonly able to deal with, they remain rather academic with respect to real life training sets, which can present a number of negative samples greater than 10-100 times J with $L \geq 1000$.

5.2 Experimental Analysis

All the experimental tests are carried out on an Intel® Core$^{\text{TM}}$ i5-4670 CPU 3.4 GHz processor machine with 16GB DDR3 1600MHz RAM memory. Gurobi Optimizer version 6.0, constrained to only use a single CPU core, is used to solve all our BIP formulations. We evaluate the two proposed BIP formulations and compare their performances in terms of computation time, TPR and FPR. BIP1 solves 59% of the (1:1) instances and only 46% of the (1:3) instances for a $TPR = 95\%$. BIP2 solves 100% of the (1:1) instances and 88% of the (1:3) instances, including the ones solved by BIP1. The non-solved instances were indeed too huge to be loaded into the solver due to memory limitation. Of the instances solved optimally by both BIPs, BIP1 took 19 min in average (max = 324 min), while BIP2 took only 0.14 sec (max = 1 sec). Of the harder instances that only BIP2 managed to solve, it did so with a mean computation time of 11 min. This computational gain indicates clearly that the use of a flow formulation, combined with the dominance properties, improves the efficiency of the solving process considerably.

For every TPR setting, the results obtained by the BIP solver and that of the DBP algorithm with respect to the total computation time are compared below. Figure 6 illustrates the time gained considering a (1:3) instance with $L = 256$ and $J = 512$. It can be observed that the time saving increases for low TPR. Let us highlight that the solutions obtained for $TPR = 100\%$ are identical since, in this particular case, DBP is optimal. We also point out that in almost all the solutions obtained by both methods, $FPR = 0\%$, which can be explained by the low number K of negative samples, which does not offer a large enough diversity.

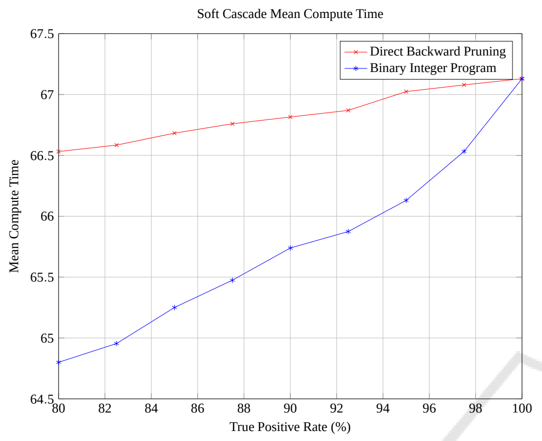

Figure 6: Example of a Mean Computation Time gap between DBP and the proposed optimal approach.

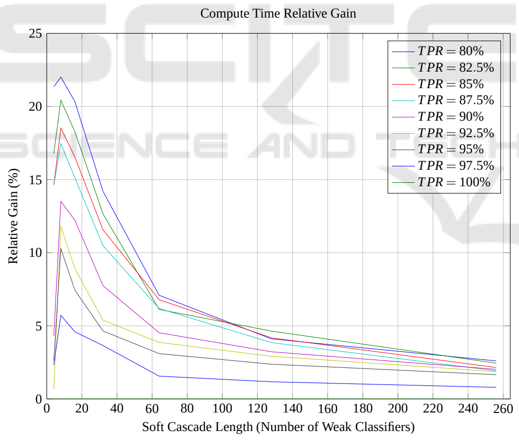

Figure 7: Mean-Computation-Time relative gain between DBP and our approach on (1:3) instances.

Figure 7 shows the mean relative time gain percentage, as a function of TPR and L, between DBP-solutions and our optimal ones for all solved (1:3) instances. The relative gain is expressed as $(C_{DBP} - C_{BIP}) \div C_{DBP}$, where C_{DBP} and C_{BIP} are the mean computation times for the DBP and optimal solutions, respectively. These results confirm that the DBP algorithm is really under optimal in terms of response time as a 22% gain can be obtained for $TPR = 80\%$. Nevertheless we also observe that a peak is reached around $L = 10$ and that the gain decreases for larger L value. As already mentioned, this is probably due to our instances that do not offer a high enough diversity of negative samples. As a consequence, an $FPR = 0$ value is reached only after around 10 levels and, in all the remaining levels, all the true positive samples are simply conserved (which means that only around 10-level cascade would have been necessary for the detection). In this situation, our optimal approach does not provide any additional profit in comparison with DBP, which explains the decreasing gain. A greater profit could have been obtained by increasing the size, hence the diversity, of the negative samples set. Unfortunately, due to the solver's limitations, our approach is not able to efficiently solve such large instances.

6 CONCLUSION

In this paper, we investigated the MSCRMP, which proved to be relevant for training time-response efficient soft-cascade detectors. We gave a formal definition of this optimization problem and proved its NP-hardness. Two BIP formulations were proposed that allow to find optimal threshold vector using MILP solvers. We showed that the problem consists in finding an optimal path inside a threshold tree. We also provided dominance properties that allow to drastically prune this threshold tree and considerably cut the search space. The second BIP formulation, based on a flow formulation, advantageously exploits the threshold tree and is actually quite efficient to solve medium-size MSCRMP instances. The results put into evidence that the classical DBP algorithm, commonly used in soft-cascade design for fixing threshold vectors, is not adapted for finding good time-response cascade. The exact approach is indeed able to improve the time response by more than 20%, keeping the TPR and FPR performances unchanged. Unfortunately, the BIP solver cannot deal easily with large-size problem instances. Nevertheless, the results obtained so far are quite encouraging and justify the need for additional research to design more advanced exact approaches for the MSCRMP (e.g., branch-and-bound procedures, dynamic programming formulations or other decomposition approaches) that can better cope with realistic large-size problem instances.

ACKNOWLEDGEMENTS

We thank the Mexican National Council of Science and Technology (CONACYT) and the French National Center for Scientific Research (CNRS) for their support.

REFERENCES

Bourdev, L. and Brandt, J. (2005). Robust object detection via soft cascade. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 236–243.

Breitenstein, M., Reichlin, F., Leibe, B., Koller-Meier, E., and Van Gool, L. (2011). Online multiperson tracking-by-detection from a single, uncalibrated camera. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(9):1820–1833.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1.

Dollár, P. (2014). Piotr's Computer Vision Matlab Toolbox (PMT).

Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545.

Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761.

Ess, A., Schindler, K., Leibe, B., and Van Gool, L. (2010). Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research*, 29(14):1707–1725.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gerónimo, D., López, A., Sappa, A., and Graf, T. (2010). Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1239–1258.

Jourdheuil, L., Allezard, N., Chateau, T., and Chesnais, T. (2012). Heterogeneous adaboost with real-time constraints - application to the detection of pedestrians by stereovision. In *Proc. VISAPP*, pages 539–546.

Mekonnen, A. A., Lerasle, F., Herbulot, A., and Briand, C. (2014). People detection with heterogeneous features and explicit optimization on computation time. In *International Conference on Pattern Recognition (ICPR'14)*, Stockholm, Sweden.

Pan, H., Zhu, Y., and Xia, L. (2013). Efficient and accurate face detection using heterogeneous feature descriptors and feature selection. *Computer Vision and Image Understanding*, 117(1):12 – 28.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42.

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. *Lecture Notes in Statistics*, pages 149–172.

Tang, D., Liu, Y., and kyun Kim, T. (2012). Fast pedestrian detection by cascaded random forest with dominant orientation templates. In *Proceedings of the British Machine Vision Conference (BMVC'12)*, pages 58.1–58.11. BMVA Press.

Viola, P. A. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.

Zhang, C. and Viola, P. A. (2008). Multiple-instance pruning for learning efficient cascade detectors. In *Advances in Neural Information Processing Systems (NIPS'08)*, pages 1681–1688.

Zhang, M. and Alhajj, R. (2009). Content-based image retrieval: From the object detection/recognition point of view. In Ma, Z., editor, *Artificial Intelligence for Maximizing Content Based Image Retrieval*, PA: Information Science Reference, pages 115–144. Hershey.

Zhang, X., Yang, Y.-H., Han, Z., Wang, H., and Gao, C. (2013). Object class detection: A survey. *ACM Comput. Surv.*, 46(1):10:1–10:53.

Zhu, Q., Yeh, M.-C., Cheng, K.-T., and Avidan, S. (2006). Fast human detection using a cascade of histograms of oriented gradients. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*, New York, NY, USA.