Combining Mixed Integer Programming and Supervised Learning for Fast Re-planning

Emmanuel Rachelson
Department of EECS
University of Liège
B-4000 Liège, Belgium
emmanuel.rachelson@ulg.ac.be

Ala Ben Abbes and Sébastien Diemer
Optimisation, Simulation, Risques et Statistiques
EDF R&D
F-92141 Clamart, France
ala.ben-abbes, sebastien.diemer@edf.fr

***Abstract*—We introduce a new plan repair method for problems cast as Mixed Integer Programs. In order to tackle the inherent complexity of these NP-hard problems, our approach relies on the use of Supervised Learning method for the offline construction of a predictor which takes the problem's parameters as input and infers values for the discrete optimization variables. This way, the online resolution time of the plan repair problem can be greatly decreased by avoiding a large part of the combinatorial search among discrete variables. This contribution was motivated by the large-scale problem of intra-daily recourse strategy computation in electrical power systems. We report and discuss results on this benchmark, illustrating the different aspects and mechanisms of this new approach which provided close-to-optimal solutions in only a fraction of the computational time necessary for existing solvers.**

***Keywords*-Mixed Integer Programming, Boosting, Power Systems Planning, Hybrid methods.**

I. Introduction

Many large-scale planning problems can be cast as Mixed Integer Programming (MIP) problems, where one tries to minimize a global linear cost function, depending on continuous and discrete variables, under a number of linear constraints. Consider, for instance, the problem of modifying the production plan of an electricity provider, in order to fit the demand in real-time during the day. This problem is subject to many constraints, involving continuous variables (the production levels, for instance) and boolean ones (Is a plant functioning or not? Has its plan been changed compared to the reference plan?) with the general goal of minimizing the production cost while satisfying the numerous operational and regulatory constraints. Many such problems exhibit significant complexity because of the interaction between continuous and discrete variables, on top of a rich formulation, often implying millions of variables and constraints. This complexity makes their exact online resolution incompatible with the operational time constraints on power networks. The approach we develop in this paper aims at making this problem tractable and solvable in a time window that is compatible with the operational requirements.

The introduction of discrete variables in linear optimization strongly changes the nature of the problems and makes their resolution NP-hard. The causes for this complexity increase are the loss of the problem's convexity properties (which makes standard continuous optimization methods inappropriate) and the appearance of a combinatorial search problem in the space of discrete variables. Traditional methods used to address this complexity issue range from local search [1] — which suffers from the absence of optimality guarantees — to global optimization through continuous relaxation [2] or heuristic search [3]. Based on our experience on power systems re-planning, we introduce a new *boolean variable assignment* method in order to predict quasi-optimal values of the boolean variables for a given MIP problem, and hence, to reduce the computation time for its resolution. The key idea we develop consists in exploiting the structuring aspect of discrete variables in order to predict the values of subsets of variables which are crucial to the optimal solution. Our approach considers an offline/online scheme that builds a boolean variable predictor offline, using Supervised Learning methods, in order to make the online resolution of similar instances of the MIP problem compatible with real-life computational constraints.

We introduce the general formulation of our method in Section II and detail a particular instance of the Boosting [4] algorithm used as a Supervised Learning method in Section III. Then, Section IV introduces the large-scale problem of computing intra-daily recourse strategies in electrical production systems. Section V reports experimental results on the previous problem, along with a thorough discussion on the various aspects, advantages and weaknesses of the proposed method. Section VI finally summarizes our contribution and suggests some perspectives to this work.

II. Algorithm

Consider a Mixed Integer Programming problem. Such problems involve continuous and integer variables, although in practice, the discrete variables often are boolean ones. For this reason, we will restrict our presentation to boolean variables (and hence, should speak of Mixed Boolean Pro-

gramming), and will discuss the straightforward extension to integer ones in the conclusion of this paper. Thus, consider the MIP problem M, with n_b boolean and n_c continuous variables, written under augmented form[1] as in Equation 1.

$$M : \begin{cases} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \in \{0;1\}^{n_b} \times \mathbb{R}^{+^{n_c}} \end{cases} \quad (1)$$

M is entirely defined by the data of the c, A and b vectors and matrix. One of the keys in solving such MIP problems lies in finding an appropriate search strategy for the combinatorial problem of assigning values to the n_b discrete variables. The main contribution of our work consists in using a Supervised Learning method in order to assign these values, in order to facilitate the online resolution of the problem described in Equation 1.

In the context where Planning [5] is cast as a MIP problem, the planning domain knowledge is encoded in the constraints of the problem through the A and b elements and the goal of the agent is specified using the c vector. When unexpected events occur, for instance when the environment's behaviour does not fit the description given by the problem's constraints, one usually triggers a *plan repair* phase, based on the reference plan. In terms of MIP solving, it consists in constructing a new MIP problem from the new initial state and the updated knowledge about the constraints, and solving it. A desirable property of plan repair methods is to run in real-time; however, for large-scale problems, this generally conflicts with the combinatorial complexity of MIP problems and common search heuristics can end up providing solutions far from the optimal solution of the updated problem. For example, in practice, local search starting from the optimal solution of the previous problem can turn out to be a very bad choice.

In a nutshell, our approach supposes two separate phases of offline and online computation. Before the problem at hand is provided, a series $(M_i)_{i\in[1,N]}$ of similar problems can be solved offline, as illustrated in Figure 1. The solution x_i^* of problem M_i is a fully-instantiated vector of $\{0;1\}^{n_b} \times \mathbb{R}^{n_c}$. The difficult part of the optimization procedure being to find correct values for the n_b boolean variables, we focus on these ones. One can extract these n_b variables from the x_i^* vectors and, for each problem M_i and boolean variable index $k \in [1, n_b]$, one can associate the pair (M_i, k) to the value $y_{i,k}$ of the kth boolean variable in x_i^*. The crucial idea here is that if one had a reliable predictor of the value associated to (M, k) — M being a new, yet unsolved problem — then the resolution of M would be a lot easier. Based on the set of $((M_i, k), y_{i,k})$ pairs, we build a *boolean variable predictor* by training a carefully chosen classification algorithm.

Then, when the new problem M becomes available, the boolean variable predictor is used to assign values to the n_b boolean variables, and a reduced problem M' is specified and solved as illustrated by Figure 2. This operation relies on the following two ideas:

- If a certain structure exists in the boolean variables assignment, across several instances of similar MIP problems, then we hope to better capture it with this Supervised Learning approach than with predefined heuristics. Note that from this point of view, our approach can be seen as an automated heuristic computation.
- In the reduced problem M', there are none or few boolean variables[2] which dramatically reduces the combinatorial search time and brings the overall resolution time close to the one of a Linear Program defined only on the n_c continuous variables.

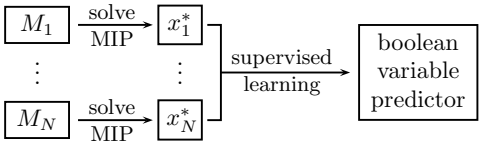

Figure 1. Construction of the boolean variable predictor

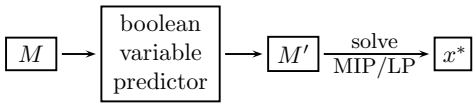

Figure 2. Online MIP simplification

III. A BOOSTING-BASED BOOLEAN VARIABLE PREDICTOR

The previous section showed how we cast the problem of selecting boolean variables in MIP problems as a supervised classification problem. A crucial element in this offline/online architecture is the classification method used. In order to compute reliable predictors with as little parametric tuning as possible and to obtain classifiers that are easy to interpret, we chose to implement the ADABOOST meta-algorithm [6].

Classification problems consist in constructing a function $h : X \to \{-1; +1\}$ which maps elements of a set X to the -1 or $+1$ labels. More generally, it consists in finding a function $h : X \to \mathbb{R}$ for which $sign(h(x))$ provides the label of element x. Such algorithms exploit the knowledge of a training set of examples $\{(x_i, y_i)\}_{i\in[1;N]}$ where $x_i \in X$ and $y_i \in \{-1; +1\}$.

The principle of a Boosting meta-algorithm [4] is to turn weak PAC (Probably Approximately Correct, [7]) learners into strong ones by assembling them in a weighted ensemble. It relies on the assumption that the original family of learners (classifiers) h are weak PAC learners, *i.e.* they always

[1] In augmented form, all inequality constraints are transformed to equalities through the introduction of positive slack variables.

[2] A direct extension of this work is to predict only a subset of the problem's boolean variables.

provide a better prediction, in probability, than a random draw. Then, boosting this family of classifiers provides a way of combining them into a strong PAC one, for which the classification error tends to zero.

ADABOOST is the most popular Boosting algorithm. It considers a distribution over training examples, learns a weak PAC classifier on this distribution, then updates the distribution so as to increase the weights of badly labeled examples and iterates, as illustrated on Algorithm 1. ADABOOST has been shown to provide efficient classifiers which are relatively insensitive to overfitting.

Algorithm 1 ADABOOST [6]

input: examples $\{(x_i, y_i)\}_{i\in[1;N]}$, iterations number T.
initialisation: $t = 1$ and $\forall i \in [1; N]\ D_1(i) = \frac{1}{N}$
repeat
  train weak learner $h_t : X \to \mathbb{R}$ on D_t
  compute the training error $\epsilon_t = \mathbb{P}_{i\sim D_t}\left[h_t\left(x_i\right) \neq y_i\right]$
  choose $\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
  update the weights distribution:

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

  where Z_t is a normalization term.
  $t \leftarrow t + 1$
until $t > T$
return $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

The method presented in Section II allows to construct offline a training set $\{(x_j, y_j)\}_{j\in[1;Nn_b]}$ where $x_{(i-1)n_b+k}$ is a pair (M_i, k) and y_j is the corresponding $y_{i,k}$ value. With this training set and a given weak PAC classifier we are able to construct a strong PAC *boolean variable predictor* which will be used afterwards for the online resolution phase. We shall discuss the choice of the weak PAC classifiers along with the experimental framework and results in Sections IV and V.

IV. INTRA-DAILY RECOURSE STRATEGIES FOR ELECTRICAL PRODUCTION

This section presents the large-scale application that motivated this research and the characteristics of the model upon which we applied the approach presented earlier.

Electricity has become a commonly used element of everyday life during the last century and is today an essential commodity of our modern societies. Optimal management of the electrical production is a major issue for power system actors, who deal permanently with the two main constraints of *stability* (insuring the safety of the whole power network) and *equilibrium* (balancing the supply and demand).

While some aspects of the electrical demand, such as the average needs at a certain time of year, are periodic and predictable, some other aspects are strongly dependent on exogenous causes that are hard to foresee. A significant cause of uncertainty in power management systems is the weather conditions: a drop of one degree Celsius during the winter time can lead to a raise in demand of more than 1800 MW, which is more than the power delivered by a third generation nuclear plant. Similarly, meteorological conditions affect the water reservoirs of hydro-electrical plants, the behaviour of individual consumers as well as industries, or even the pricing of electricity on energy exchange markets.

Such a variability is hard to predict and the daily production plans, designed to balance supply and demand, often need to be adapted during the course of execution, whenever an updated demand scenario becomes available.

Our application case deals with the French network. In France, every electricity producer is bound by a contract with the network managing company. This contract imposes that the producers should be able to mobilize extra power so that the network manager can guarantee the safety of the network in case of failure. Such power margins are called *reserves* and the producer incurs large penalties for not being able to provide them, so they have to be taken into account when planning the daily production. Consequently, every day, when the electricity producer plans its production for the T time steps of the next day, it solves a very large linear problem where the objective function (Equation 2) is the overall sum of the production costs $C_u(P_u)$ (for the power P_u generated by plant u, including the reserves), and a cost of *supply failure* $C_0(P_0)$, under the following non-exhaustive list of constraints (which we shall not present in detail for the sake of clarity):

- supply/demand balance,
- initial state of the network,
- minimal production/rest period for each plant,
- gradient constraints (increases and decreases in production are bounded),
- maximum number of starts and extreme changes per day,
- minimal times between any schedule change for power and reserve production,
- flux and volume relations on hydro-electrical plants, etc.

$$\begin{array}{ll} \underset{P_u\in X_u}{\text{minimize}} & \sum_{u=1}^{n} C_u\left(P_u\right) + C_0\left(P_0\right) \\ \text{subject to} & P_0 = D_{ref} - \sum_{u=1}^{n} P_u \\ & \text{and other constraints} \end{array} \quad (2)$$

This planning problem, cast as a linear program, contains millions of variables and constraints and already requires a significant computational time to produce an optimal solution for the next day. This problem is generally referred to as the *daily planning problem* and its solution is called the *reference plan* P^{ref}, with respect to the *reference demand* D_{ref}.

In order to face the various events that could affect the demand and render the reference plan sub-optimal, the producer is allowed to adjust its production every hour during the day, after warning the network manager in advance. This adjustment is subject to an additional contractual constraint: the producer cannot change the production plan of more than $N_{max} = 30$ power plants. This new constraint can also be written as a linear expression by introducing the intermediate variables $x^+_{u,t}$ (resp. $x^-_{u,t}$) which are equal to 1 if plant u's production program is increased (resp. decreased) at time step t, and x_u which is equal to 1 if at least one of the $x^\pm_{u,t}$ is non-zero.

$$\forall u,t \begin{cases} x^+_{u,t} & \geq & \frac{P_{u,t}-P^{ref}_{u,t}}{P_{max}} \\ 1-x^+_{u,t} & > & \frac{P^{ref}_{u,t}-P_{u,t}}{P_{max}} \end{cases}$$

$$\forall u,t \begin{cases} x^-_{u,t} & \geq & \frac{P^{ref}_{u,t}-P_{u,t}}{P_{max}} \\ 1-x^-_{u,t} & > & \frac{P_{u,t}-P^{ref}_{u,t}}{P_{max}} \end{cases}$$

$$\forall u,t: \; x_{u,t} = x^+_{u,t} + x^-_{u,t}$$

$$\forall u \begin{cases} x_u & \leq & \sum_{t=1}^{T} x_{u,t} \\ x_u & \geq & \frac{\sum_{t=1}^{T} x_{u,t}}{T} \end{cases}$$

$$\sum_u x_u \leq N_{max}$$

By adding these last boolean constraints to the daily planning problem, one obtains the so-called *intra-daily* planning problem, for which solutions are called *intra-daily recourse strategies*. As of today, the resolution of the intra-daily planning problem is hand-made by experts who take quick adjustments decisions when unexpected events happen on the power network. Our focus in this research is on making the intra-daily planning problem's resolution compatible with the time constraints of online operations. Whenever the predicted demand changes from D_{ref}, the electricity producer has a time window of around 10 minutes in order to compute a recourse strategy, declare it to the network manager for validation, and put it in practice. Currently, solving the intra-daily planning problem requires one to several hours given the size of the French network so we aim at gaining at least an order of magnitude in resolution time with as little loss in optimality as possible.

The method presented in Sections II and III is implemented as follows. Since daily plans are computed one day for the next, whenever D_{ref} and P^{ref} become available, a series of $N = 120$ historically relevant variations ΔD_i on D_{ref} are used to generate a corresponding set of N MIP problems. These problems are solved in parallel and one builds the boolean variable predictor's training set from their results. An example of these variations in demand in presented on Figure 3. Then the predictor is trained and stored to be used during the next day, as illustrated on Figure 4. Note that, in practice, predicting the boolean variables for the intra-daily planning problem boils down to selecting which plants will participate in the reorganization of the network (within the limits of N_{max} plan changes). The selection of a given plant for plan change is called *redeclaration*.

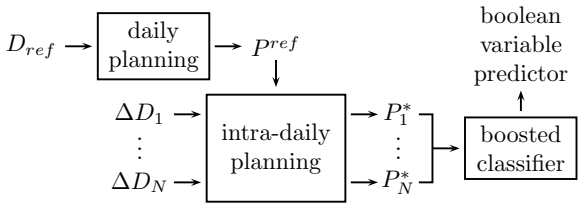

Figure 4. Offline construction of the boolean variable selector

Then, whenever a new demand $D \neq D_{ref}$ is observed during the day, the corresponding $\Delta D = D - D_{ref}$ is computed and provided to the boolean variable predictor. This one returns a selection of non-zero variables, corresponding to a selection of production plants. As illustrated by Figure 2, a reduced problem M' is then specified and solved. Note that since the output of the boolean variable predictor is a selection of power plants, the reduced problem is similar to a daily planning problem on a reduced network.

We compared several weak classifiers in order to build the boolean variable predictor: plain CART trees [8], untuned Support Vector Machines (SVMs, [9], [10]) trained only on a fraction of the input variables and a collection of static decision rules derived from the experts' knowledge and rules of thumb. In order to further study how these classifiers can be combined altogether, we introduced an extra model selection phase in ADABOOST: when classifier h_t is trained, an instance of each technique is actually trained and the one providing the best training error is kept. This allows us to compare the proportion of trees, SVMs and expert rules in the final ensemble classifier. The theoretical bound on the generalization error of Boosting is preserved as a worst-case bound on each classifier family (assuming they all are weak PAC classifiers).

The next section reports optimization results on a benchmark network composed of 27 power plants of various types. The maximum number of plants that can undergo a plan change is set to $N_{max} = 9$. This results in a MIP problem for intra-daily planning of 96219 variables and 61455 constraints.

V. EXPERIMENTAL RESULTS

A. Cross-validation of the classification phase

In order to evaluate the generalization accuracy of the boolean variable predictor, we ran an N-fold, leave-one-out, cross-validation estimation on the classification error, at each ADABOOST iteration, as reported on Figure 5. For this purpose, we separated the results of one of the MIP problems from the training set, trained the classifier against the remaining examples and tested it on the previously

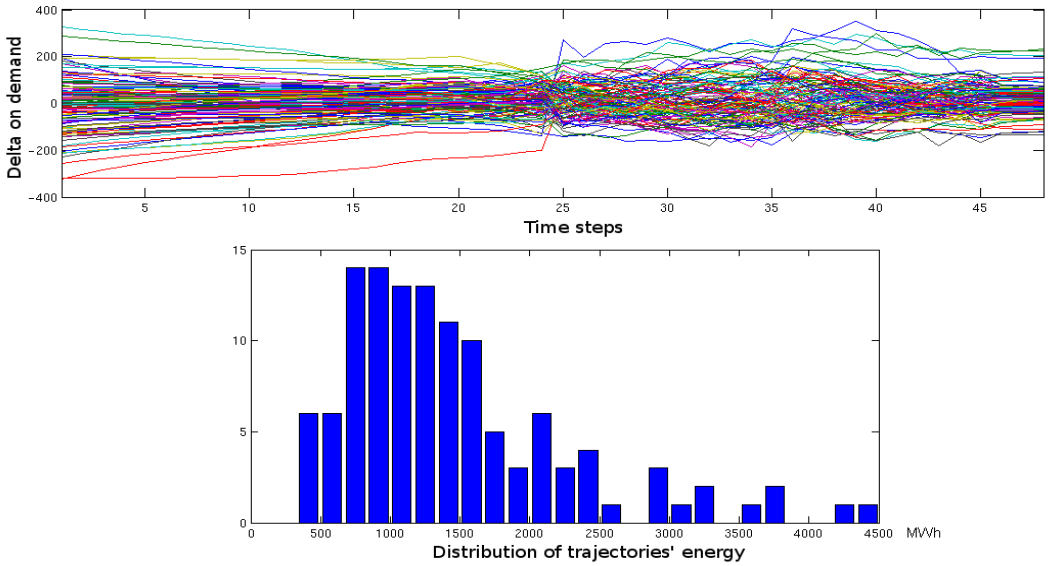

Figure 3. Variations in demand over $T = 48$ time steps

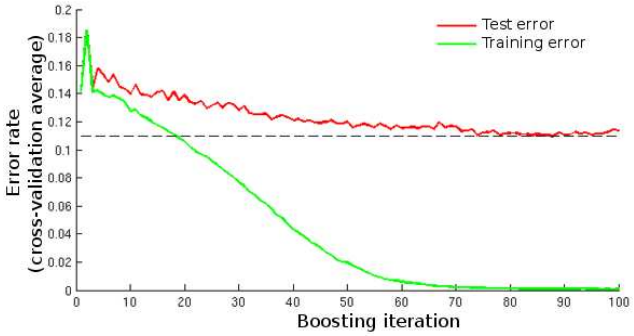

Figure 5. Training and test errors

isolated data. Since we use $N = 120$ separate MIP problems, this resulted in a 120-fold cross validation.

As expected from a Boosting algorithm, the learnt classifier does not overfit the data. However, the generalization error seems to converge around a 10% error. More specifically, the number of misclassifications tends towards 3 (out of 27 power plants) with a standard deviation of 1.35 errors.

B. Respect of the N_{max} constraint

During the boolean variable predictor construction, the N_{max} value is never explicitly imposed. Consequently, there is no strong constraint on the classifier that would lead it to respect the global N_{max} constraint. In practice however, we observed it was mostly the case, since in 55% of cases, this constraint is exactly respected (8 or 9 redeclarations), in 40% of cases, the number of reprogrammed plants is 10 and the remaining 5% cover the case of 11 redeclarations. This can be intuitively explained by considering the training set upon which this predictor is built. This training set is composed of at least twice as many negative examples than positive ones, since there can be only 9 plant redeclarations and 18 non-redeclarations in a network of 27 plants. This natural bias is recreated in the classifier's output.

Interestingly however, the boolean variable predictor tends to "over-redeclare" (redeclare too many plants) more than "under-redeclare". On average, 1.6 plants are redeclared while they do not participate in the optimal solution, while only 1.4 plants of the optimal solution are forgotten in the output of the boolean variable predictor. The latter is a lot more penalizing for our algorithm than the former since having an extra useless variable in the reduced M' problem only incurs additional computation time, while removing crucial variables can imply large losses in optimality. We further discuss this question of sub-optimality in the next paragraphs.

Figure 6 plots, for each power plant, the number of times it participated in the network's reorganization, the number of times it was forgotten from the optimal solution and the number of times it was there and should not have been.

C. Choice of classifiers

Recall that, in order to compare the flexibility and the relevance of the different families of classifiers, we introduced a model selection phase in AdaBoost by allowing a choice between different classifiers during the computation of h_t, based on the training error. On average, we witnessed a strong predominance of tree-based models (around 80%), a few untuned SVMs (20%) and almost no expert rules. This can be explained by the fact that expert rules are situation-

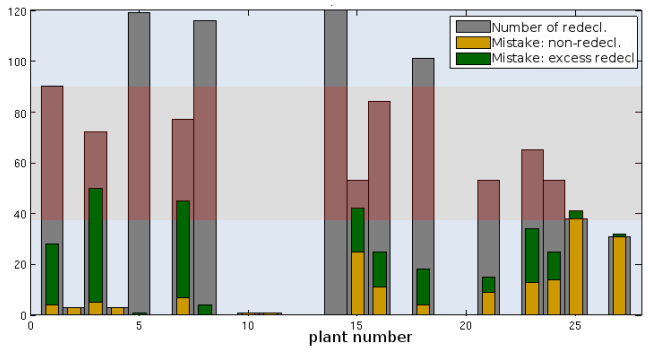

Figure 6. Number of redeclarations

specific, ill-defined heuristics. On top of this, their ability to separate the data is very dependent on the problem at hand and there is no guarantee that they are weak PAC learners whatsoever. Nevertheless, this also illustrates the potential of automated learning techniques for formalizing some general purpose expert knowledge and optimizing it.

D. Generalization error and local minima

When trying to search for the causes of misclassifications, one can try training two separate boolean variable predictors on two close data sets, as was the case during the cross-validation phase presented earlier. The purpose of this comparison is to evaluate why these two predictors might provide different results. On the one hand, the classifiers built during the two first ADABOOST iterations were very similar, comforting the idea that the global structure of the boolean variable selection function was well captured by the data set and the classifiers. On the other hand, the classifiers built at further iterations quickly diverged from each other and lead to the misclassifications. This last phenomenon can be better studied by a finer analysis of the underlying MIP problem. The intra-daily planning problem admits many quasi-optimal solutions that are quite different in nature but very close in cost. Consequently, from one training set to the other, some of these local optima could have been captured while others were not, hence resulting in the discrepancies between the classifiers, while still providing close-to-optimal cost values. This finer analysis allows to interpret the margin results provided in the example of Figure 7. In a given point x, the margin of an ensemble classifier output by ADABOOST is the absolute value of the corresponding $\sum_{t=1}^{T} \alpha_t h_t(x)$ value, measuring the confidence in the classification output. The previous analysis illustrates why, in Figure 7, the three misclassified power plants were not the ones with the smallest margins.

One of the causes of these almost equivalent local minima which affect the generalization error is that, at a given time, in the power network, several plants have almost equivalent characteristics. For instance, two thermo-electrical plants that are not functioning and have similar characteristics will

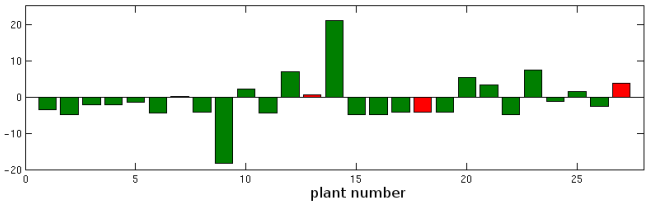

Figure 7. Example of margins per power plant

yield similar costs when started. Consequently, the crucial problem here might not be to predict exactly which one is the best plant, but to show that the other costly plants are never started instead. Hence, misclassifications might very well be errors in choosing between two very good choices, hiding the fact that the bad choices were safely discarded.

Indeed, a second consequence of this analysis is that, since several local optima exist and are close to each other in value (the associated costs of the redeclarations are similar), it might be more relevant to compare the overall costs of the recourse strategies, rather than the classification accuracy.

E. Optimality loss

As illustrated by the previous paragraph, a good global measure of our approach's efficiency is the overall optimality loss of the cost function, rather than the difference between the boolean variables of an optimal redeclaration and the ones output by the predictor. Recall that the initial motivation of this work is to leverage the computational cost of solving the large MIP problem while retaining a good strategy with respect to this overall cost.

Figure 8 compares the costs of three different strategies over the 120 scenarios considered (note the vertical price scale). The solid (red) bottom line is the optimal cost, computed offline by fully solving the MIP problem. The dashed (blue) line which almost coincides with the first solid line is the cost of the redeclarations output by the predictor. Finally, the top solid (green) line provides an estimate of what would happen if the 9 plants of the redeclaration were chosen at random and then their plans optimized.

The few cases (3 out of 120) where the dashed line notably differs from the optimal cost consist in situations where the predicted redeclaration resulted in a power supply failure, which led to large penalties. However, on average, we observe an additional cost of less than 1000 euros for using the boolean variable predictor, for a global production cost of around 3 million euros.

More precisely, Figure 9 reports the distribution of additional costs over scenarios. In 70% of cases, this additional cost was less than 100 euros. This result needs to be put in perspective with the existence of the local minima of the cost function. As the random draw of the redeclared plants illustrated, it is rather easy to reach a situation where the optimization of a subset of plants leads to a very suboptimal

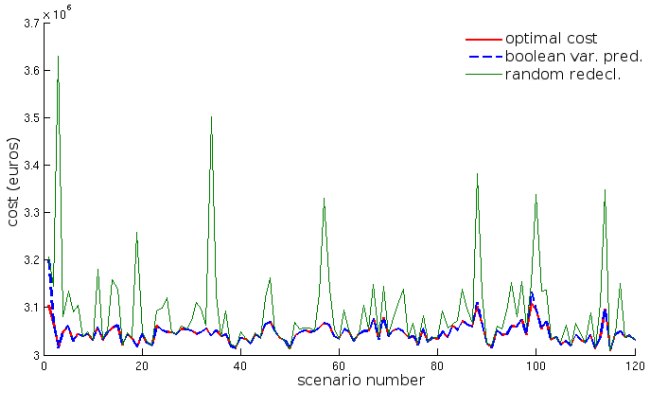

Figure 8. Costs of different plant selection strategies

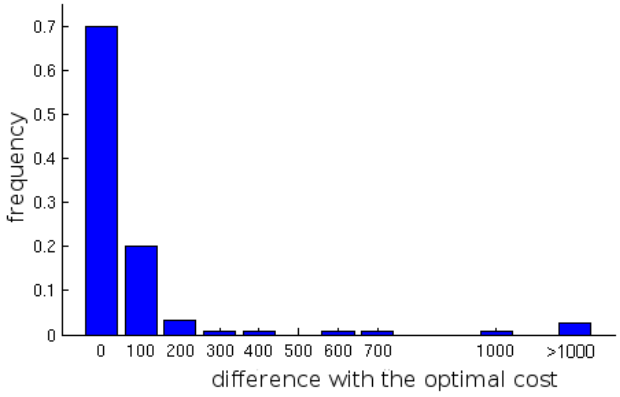

Figure 9. Optimality loss (note the difference of scale with Figure 8

solution. However, if there exists several local minima that are quasi-equivalent in cost to the global minimum, then our goal is to predict well at least one of these minima, in order for the optimization to reach a final cost that is within a short range of the optimal solution.

The existence of these quasi-equivalent local minima, in the first place, is due to the existence of quasi-equivalent plants in the network: choosing one or the other for the redeclaration makes a negligible difference on the final cost. Consequently, when the predictor learns to redeclare a given plant in a given situation, it might not exactly correspond to the optimal plant in another situation, but still provides us with a very good global cost. This is indeed another advantage compared to an *ad hoc* heuristic-based boolean variable selection strategy: by trying to capture the structure of good redeclarations via automated learning, one reduces the dependency on an expert-based heuristic, which reduces the risk of falling in a unexpected bad case of this heuristic.

F. Computation times

Finally, while we wanted to keep good quality solutions, the main goal of this study was to reduce the computation time of a quasi-optimal solution. Table I summarizes this gain. On the reduced network of 27 power plants (which is approximately one tenth of the actual French network size), the exact optimization of the intra-daily planning problem M takes 1 hour on average. On the other hand, evaluating the boolean variable predictor and computing a solution to the reduced problem M' takes a little more than 2 minutes on the same computer.

Consequently, we can report a gain in computation time of a factor almost 30, with a loss in optimality of less than 1‰. We conjecture that with larger power networks, the computational time improvement will be more than a factor 30 since the difference between N_{max} and the number of plants in the network increases.

Exact optimization of M	1h
Power plant selection	0.24s
Reduced optimization of M'	128.07s

Table I
COMPARISON OF AVERAGE COMPUTATIONAL TIMES

G. Generalization

More generally, the complexity of a standard Branch-and-Bound method for solving MIP problems is polynomial in the best case and exponential in the worse[3], in the number of discrete variables [11]. Consequently, predicting the values of some or all of these variables allows to dramatically reduce the optimization times.

While there are no more theoretical guarantees on the optimality of our approach in the general case than on heuristic search, we argue that automatically extracting the discrete variables optimal value's structure through a sound statistical learning technique might be safer than relying on precomputed heuristics. In particular, for problems such as the intra-daily recourse strategy computation, that exhibit several equivalent local minima, our approach proved to be more robust that direct heuristic optimization.

VI. CONCLUSION

We introduced a general method for the online resolution of Mixed Integer Programming problems, based on the assumption that several instances of a similar problem can be solved offline. Our contribution combines MIP optimization with a Supervised Learning approach in order to predict the values of the discrete variables of the MIP problem's optimal solution. This allows to greatly reduce the time complexity of the optimization phase. We provided a fully instantiated version of the algorithm, using the ADABOOST meta-algorithm. We then used it to illustrate our approach on a large-scale industrial problem: the intra-daily recourse strategies computation, in power systems management. This

[3]Depending on the branching and bounding functions.

allowed us to analyze the behaviour of our method, highlight its mechanisms, its strengths and potential weaknesses. The overall conclusion of this empirical study is that our method was able to find quasi-optimal solutions (less than 1‰ optimality loss) while gaining an order of magnitude in computation time (from hours to minutes). Hence these empirical results suggest this pragmatic approach is promising and deserves further analysis and experiments.

In particular, on the intra-daily planning problem, the algorithm exhibited two remarkable domain-specific properties. First, the Supervised Learning approach led to a close to optimal respect of the constraint on the maximum number of redeclarations, even on cases very different from the ones of the training set. This was never imposed in any way so it was an unexpected encouraging result for this algorithm. Secondly, the algorithm proved itself robust to generalization, in the sense that even when the predicted plants were not exactly the ones of the optimal redeclaration, their associated optimized cost was still very close to the global optimal cost for the situation at hand. More generally, the use of a simple, non-parametric method such as ADABOOST allowed us to combine around 50 simple classifiers to capture the structure of the optimal discrete variables assignment over different instances of the MIP problem. This led to an overall optimality loss of less than 0.1‰ in 70% of the scenarios considered and less than 1‰ on average.

This work's promising results open the door to several research leads. First, they confirm the idea that combining Supervised Learning and Optimization can yield large improvements to classical techniques, confirming previous similar analyses such as [12]. In particular, this work should be put in perspective with the one reported in [13] that tackles a similar optimization problem with the help of a different Supervised Learning regression approach. More generally, this contribution can be seen as an automated heuristic computation for variable selection in Optimization and hence confirms the interest of bridging the gap between Machine Learning and Optimization for this task.

In Section II, we restricted our analysis to the case of Mixed Boolean Programming using the argument of equivalence between boolean and discrete variables. Preserving the integer aspect of MIP problems translates to considering categorical variables instead of binary ones. On the classification side, it boils down to considering multi-class classification problems instead of binary ones, but our method's bottom-line remains the same.

These results also raise their share of new questions, both for the general method and for the specific case of the intra-daily planning problem. Is it possible to provide PAC optimality bounds for the results of our algorithm? Is there an interest in constructing several smaller classifiers per power plant, instead of the current large one that synthesizes all the information available? How can one extend the current version of the method to multi-class Boosting methods such as ADABOOST.MH [14]? Is it possible to reuse a classifier across several very different instances of the optimization problem (for instance reuse the boolean variable predictor of the intra-daily problem for several days)? All these perspectives (and maybe others) underline the set of possible contributions from Artificial Intelligence and Machine Learning to Optimization and emphasize the strong potential of this currently very active area of research.

ACKNOWLEDGMENTS

Emmanuel Rachelson gratefully acknowledges the support of the Belgian Network DYSCO, funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office.

REFERENCES

[1] M. Fischetti and A. Lodi, "Local branching," *Math. Program.*, vol. 98, no. 1–3, pp. 23–47, 2003.

[2] E. Danna, E. Rothberg, and C. L. Pape, "Exploring relaxation induced neighborhoods to improve MIP solutions," *Math. Program.*, vol. 102, no. 1, pp. 71–90, 2005.

[3] T. Achterberg and T. Berthold, "Improving the feasibility pump," *Discrete Optimization*, vol. 4, no. 1, pp. 77–86, 2007.

[4] R. E. Schapire, "Boosting Approach to Machine Learning: An Overview," *Nonlinear Estimation and Classification*, 2003.

[5] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[6] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," in *International Conference on Machine Learning*, 1996.

[7] L. G. Valiant, "A Theory of the Learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[8] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[9] V. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[10] J. Shawe-Taylor and N. Cristianini, *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[11] W. Zhang, "Branch-and-bound search algorithms and their computational complexity," University of California, Tech. Rep. A895413, 1996.

[12] D. Zupanic, "Value suggestion in mixed integer programming by machine learning algorithm," *Electronic Notes in discrete Mathematics*, vol. 1, pp. 74–83, 1999.

[13] B. Cornelusse, G. Vignal, B. Defourny, and L. Wehenkel, "Supervised learning of intra-daily recourse strategies for generation management under uncertainties," in *IEEE Power Tech Conference*, 2009.

[14] R. E. Schapire and Y. Singer, "Improved Boosting Algorithms Using Confidence-rated Predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.