Optimisation over Decision Trees – A Case Study for the Design of Stable Direct-Current Electricity Networks

Daria Gutina[1], Andreas Bärmann[1], Georg Roeder[2], Martin Schellenberger[2] and Frauke Liers[1]

[1] Andreas.Baermann@fau.de
Daria.Gutina@protonmail.com
Frauke.Liers@fau.de
Department of Data Science,
Friedrich-Alexander-Universität Erlangen-Nürnberg,
Cauerstraße 11, 91058 Erlangen, Germany

[2] Georg.Roeder@iisb.fraunhofer.de
Martin.Schellenberger@iisb.fraunhofer.de
Gruppe Data Analytics, Abteilung Intelligente Energiesysteme,
Fraunhofer-Institut für Integrierte Systeme und Bauelementetechnologie IISB,
Schottkystraße 10, 91058 Erlangen, Germany

Abstract

In many real-world mixed-integer optimisation problems from engineering, the side constraints can be subdivided into two categories: constraints which describe a certain logic to model a feasible allocation of resources (such as a maximal number of available assets, working time requirements, maintenance requirements, contractual obligations, etc.), and constraints which model physical processes and the related quantities (such as current, pressure, temperature, etc.). While the first type of constraints can often easily be stated in terms of a mixed-integer program (MIP), the second part may involve the incorporation of complex non-linearities, partial differential equations or even a black-box simulation of the involved physical process. In this work, we propose the integration of a trained tree-based classifier – a decision-tree or a random forest, into a mixed-integer optimization model as a possible remedy. We assume that the classifier has been trained on data points produced by a detailed simulation of a given complex process to represent the functional relationship between the involved physical quantities. We then derive MIP-representable reformulations of the trained classifier such that the resulting model can be solved using state-of-the-art solvers. At the hand of several use cases in terms of possible optimisation goals, we show the broad applicability of our framework that is easily extendable to other tasks beyond engineering. In a detailed real-world computational study for the design of stable direct-current power networks, we demonstrate that our approach yields high-quality solutions in reasonable computation times.

Keywords: Decision Trees, Random Forests, Mixed-Integer Programming, Power Networks

Mathematics Subject Classification: 68T05, 68Q32, 90C11, 90B10

1 Introduction

In recent years, mathematical optimisation methods have been employed very successfully for the solution of many applied problems. One reason for this success is the enormous progress in algorithms for the global solution of mixed-integer programs (MIPs). Nowadays, more and more optimisation problems of the form $\min\{c^T x \mid Ax \leq b,\ x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q\}$, with vectors c and b and a matrix A of appropriate dimensions, can routinely be solved with modern available software, even for large problem instances. However, it is often difficult to develop accurate, efficiently tractable mixed-integer programming models for more general problem classes that, for example, incorporate physical processes. These processes often involve non-convex and non-linear relationships between the variables. For instance, this applies to the behaviour of electrical grids that are governed by partial differential equations (PDE). While the solution of PDEs is already complex in itself, the corresponding optimisation problems are even more challenging. In addition, it might be that the exact laws or procedural rules underlying an optimisation task are unknown, may depend on the installed hardware, or may only be given implicitly. On the other hand, typically, historical or simulated datasets are available from which approximations of these relationships can be derived. Such data can be analysed and prepared to train a machine learning classifier whose results may then be integrated into an optimisation procedure. *Training* refers to the process by which a machine learning classifier autonomously recognises relationships between the given data in the dataset and extracts the rules needed to identify them. These rules are then used to categorise new, unseen input data. There are many types of classifiers and regression methodologies. An overview of many popular machine learning methods is provided e.g. by Bhavsar and Ganatra (2012); Hastie et al. (2009); Bishop (2006). Once a trained classifier is available, the task is to integrate its decision rules into the considered optimisation model.

In this paper, we deal with tree-based classifiers, in particular random forests, which are ensemble classifiers consisting of multiple decision trees. Each of these decision trees classifies data points according to several hierarchically consecutive linear inequalities, which makes them very well suited for use in a mixed-integer linear optimization context. We exploit this to replace the implicitly given constraints in the optimisation model by the decisions of trained decision trees.

Tree-based classifiers can make very accurate predictions in the scope of the training data, but are normally not suitable for extrapolation. Their prediction performances decrease with the distance from the training data, see (Shahriari et al., 2016). In order to be independent of the training data in the optimization model and at the same time to keep the prediction accuracy constantly high in the entire parameter range considered, we use data points with maximum distance to each other to train the classification model.

At the hand of a real-world case study, we evaluate the possibilities such a modelling approach offers, namely the design of stable direct-current (DC) electricity network, *DC grids* in brief, which comprise connected DC sources, consumers and storage systems. DC grids with system voltages below 1500 V, so-called *low-voltage DC grids* (*LVDC grids or networks*) are becoming increasingly important in the context of renewable energies, see International Electrotechnical Commission (2016) for an introduction. For its safe and reliable operation, the stability of an LVDC grid must be maintained under all planned load scenarios. Network stability, in turn, is determined by the grid layout, device parameters and control methods. To assess the stability of a network for various configurations and setup parameters, we generate the respective stability information from automated circuit modelling to describe the physical grid and use the state of stability (stable or unstable) as the label for classification (cf. Kumar et al. (2020)). With increasing grid complexity, novel methods are necessary to enable continuous grid monitoring and automated adjustment in the case of grid instability. This led us to train a classifier that differentiates the "stable" physical parameter settings from the "unstable"

ones.

Incorporating multiple instead of only one decision tree is known to typically achieve a significant improvement of the prediction accuracy. Thus, in many practical applications ensemble classifiers consisting of several smaller classifiers are used, for example the random-forest approach, see Breiman (2001). Multiple decision trees are trained using equally-sized subsets of the training data, and a joint classification is determined. Random forests have already been used in many contexts, e.g. statistics, medicine and fault detection, but according to our knowledge not yet for studying the stability of electricity networks. The combination of trained classifiers with optimisation models has been used in similar applications However, to the best of our knowledge, we are the first to use the structure of trained classifiers to make inferences about input parameters, especially to find particularly stable regions in the electricity network.

Literature review The combination of machine learning classifiers with optimisation models is a very active research topic. The corresponding literature referred to in the following is divided into two categories: research articles where a classifier is trained in a globally optimal fashion using mixed-integer programming methods and works that exploit the decisions of already trained classifiers within optimisation models.

The first topic is treated e.g. in Bonfietti et al. (2015) and Bertsimas and Dunn (2017) for random-forest classifiers and in Thorbjarnarson and Yorke-Smith (2020) for neural networks. In Kumar et al. (2019), optimisation models are used to analyse trained deep neural networks with ReLU and Max Pooling Layers concerning their decision-making and to identify and visualise input data that are particularly well recognised.

In the second category, to which the present article also belongs, trained machine learning classifiers are used as input within optimisation models of different kinds. A recent overview of methods to identify the feasible regions in optimisation problems via trained machine learning classifiers can be found in Maragno et al. (2021), who present it as a more general framework for data-driven optimisation. The works (Mistry et al., 2021, 2018; Thebelt et al., 2021, 2020; Ceccon et al., 2022; Thebelt et al., 2022) deal with the integration of trained gradient-boosted regression trees into optimisation models stemming from different applications There the distribution of the training data that influences the prediction accuracy is integrated as a penalty term into the objective function to minimise risk. In contrast, in our work the training set is chosen such that it samples the whole relevant parameter space. As a consequence, such additional risk measures can be avoided. Bertsimas et al. (2016) show how regression models, more precisely random forests and support vector machines predicting clinical trial outcomes, can be incorporated into MIP models in order to determine best possible chemotherapy treatments. In Ferreira et al. (2015), the development and implementation of a pricing decision support tool used to maximise sales of novel products are described, where the price-demand ratio was predicted using random forests. Biggs et al. (2017) maximises over the predicted value of trained random forests to find the best possible input parameters. As possible applications, they consider maximizing the profitability of estate investments and determining the most appropriate jury assignment in case studies. A similar approach was taken by Mišić (2020) for applications from the fields of drug development and customised pricing. Furthermore, the latter two works consider heuristic methods and Benders decomposition to handle random-forest classifiers with many decision trees, which are used to obtain a better prediction quality. Unlike the literature cited above, we deal with random forests for classification, where we are particularly interested in the practical implementation of models incorporating many decision trees. The work by Halilbašić et al. (2018) is motivated along similar lines. They use decision trees for classification in order to extract decision rules from data. These decision rules are incorporated into an optimisation model to study redispatch measures performed by network transmission system operators in order to ensure network stability.

Our article discusses a closely related approach for larger and multiple decision trees. In ad-

dition, we investigate the possibilities our approach offers with respect to examining the areas predicted to be feasible by the classifier. This work is part of an approach to create tools and methods for designing and operating LVDC networks with the support of data analysis methods. In the design phase, different network configurations are calculated and the optimal configuration of network parameters to achieve grid stability are determined. During operation, LVDC networks are monitored continuously using a novel impedance measurement method and stabilized by optimizing feed-in characteristics or software parameters in the converter systems. For details, see (Roeder et al., 2021).

Contribution Our main contributions are as follows. We summarize how complex or unknown constraints, implicitly given via a decision tree or random forest, can be incorporated into a mathematical optimisation problem. Using three exemplary use cases corresponding to different optimisation goals, we demonstrate the versatility of a random-forest classifier for defining (part of) the feasible set of an MIP. In particular, we show how the closest feasible point and the largest feasible region represented by the classifier can be determined. In an extensive case study, we use MIP models to determine the best possible adjustment of network control or circuit parameters, e.g. capacitances, resistances and cable lengths of an LVDC network. Notably, the stability of the network is ensured by incorporating a random-forest classifier into the model. Within this case study, we find that the number of decision trees plays a significant role in the solution time of the resulting MIP model. For random forests above a certain size, the sequential solution of slightly adapted and more and more restricted MIP models is necessary. We propose efficient algorithms to solve models incorporating random forests that were previously unsolvable or only solvable with great effort. They produce high-quality solutions for the design of stable LVDC networks.

Structure This work is organized as follows. Section 2 introduces basic definitions and notation concerning decision tree and random-forest classifiers. Subsequently, Section 3 proposes algebraic reformulations of random-forest classifiers to integrate them into an MIP model. In Section 4, we discuss several practically relevant optimisation-based approaches to examine the areas predicted to be feasible by a tree-based classifier. Section 5 explains the technical background on stability requirements in LVDC networks and describes the derivation of random-forest classifiers for stability prediction. The resulting classifiers are used in Section 6 as part of an integrated optimisation approach to find the best possible stable designs of an LVDC network. We show that optimal solutions can be obtained within reasonable computation times for random-forest classifiers with more than 100 trees. We also describe a modelling trick for solving larger problem instances, which allows us to consider random-forest classifiers with up to $1,000$ decision trees and discuss the consequences for the application. We conclude in Section 7 with an outlook on possible extensions of our framework.

2 Preliminaries on Tree-Based Classifiers

We consider a data set $(X, Y) \in \mathbb{R}^{n\times p} \times \mathbb{R}^p$ consisting of n input-output data tuples $(X_i, Y_i) \in \mathbb{R}^p \times \mathbb{R}$, $i \in \{1, \dots, n\}$. In each such data tuple, the input vector $X_i \in \mathbb{R}^p$, with p-many features, is assigned a binary label $Y_i \in \{0, 1\}$. The data set (X, Y) can be used as training data to fit ("learn") a classifier, i.e. a mapping function $c\colon \mathbb{R}^p \to \{0, 1\}$ to classify data points in $\mathbb{R}^p$ into either 0 or 1. This allows us to classify unseen data points as well. Estimates on the prediction certainty of the classification can be represented by a function $f\colon \mathbb{R}^p \to [0, 1]$. We start by considering a classifier widely used in practice, namely binary decision trees, and then describe their extension to random-forest classifiers.

2.1 Binary Decision Trees

Decision trees are among the most common data classifiers used in practice. Their advantages include that they can quickly be trained to satisfactory quality using heuristics, they allow for "human-interpretable" classification rules which can be expressed in terms of the features of the input data and they can easily be visualised (cf. Bertsimas et al. (2019)).

For a detailed introduction to decision trees, we refer the reader to Edward A. Bender (2010), whose notation we adopt in the following brief recapitulation. A decision tree $T = (V, E)$ is a connected arborescence, where V is the set of vertices and E the set of edges. It possesses exactly one node without an ancestor, the *root*. Nodes without outgoing edges, and thus without descendants, are called *leaves*, all others are *intermediate nodes*. For each vertex $w \in V$ in a decision tree, there is a unique sequence of edges $(v_1, v_2), (v_2, v_3), \ldots, (v_k, w) \in E$ from the root v_1 to w, called the *path* to vertex w. The *depth* of the tree is the maximum number of edges in a path from the root to a leaf. The purpose of a decision tree is to classify data points in space by assigning each point to a leaf in the tree. It recursively subdivides the p-dimensional space into disjoint regions using logical rules. The node set V is thus partitioned into *decision vertices* V_{Dec}, comprising the root and the intermediate nodes, and the *class vertices* V_{Cl} formed by the leaves of the tree.

In *binary* decision trees, intermediate nodes have exactly two outgoing edges and thus two descendants. Further, the logical split rule in a node $v \in V$ takes the form of a linear constraint $\tilde{a}_v^T x \leq \tilde{b}_v$ with $\tilde{a}_v \in \mathbb{R}^p$ and $\tilde{b}_v \in \mathbb{R}$. The two descendants of v can be seen as the roots of two subtrees, where the "left-hand" subtree describes the subregion in space that satisfies the linear constraint, and the "right-hand" subtree the subregion that does not. In a *univariate* decision tree, $\tilde{a}_v$ is a multiple of a standard unit vector, i.e. $\tilde{a}_v = k \cdot e_i$ with $k \in \mathbb{R}$ and $i \in \{1, \ldots, p\}$. As a result, each of the decision rules splits the data points at exactly one feature $i \in \{1, \ldots, p\}$ in an axis-parallel fashion.

In this article, we restrict ourselves to binary classifiers with labels representing two different categories for ease of exposition. Our approach can, however, be extended to decision trees with more categories as they are considered in Breiman (1996, 2001) as well as Bertsimas and Dunn (2017), for example. In the following, each class vertex $v \in V_{\text{Cl}}$ thus corresponds to one of the two categories 0 and 1, depending on the labels of the data points falling into the subregion in space which fulfils all linear constraints in the decision vertices on the path from the root to v. If the majority of the points in that region have label 0, so does v, and 1 otherwise. In case of a tie, the class with the lowest label is predicted, i.e. class 0, as proposed in Breiman (1996).

2.2 Random-Forest Classifiers

One commonly-used type of multi-tree classifiers are *random forests*, introduced in Breiman (2001), which are based on Breiman's bagging (bootstrap aggregation) idea. In this approach, multiple predictive classifiers are trained on a subset of the training data. Then the predictions of all classifiers together are used to classify a given data point. Random-forest classifiers are implemented in many machine learning libraries, e.g. in *scikit-learn* (see Pedregosa et al. (2011)). Detailed explanations can be found in Breiman (1996, 2001), of which we give a short summary in the following.

The classification of a given data point produced by a random forest depends on the classifications of the individual decision trees. There are two main approaches in the literature for aggregating them into one decision: voting and averaging. The *voting* approach is the original version of the random-forest classifier used for categorical responses, and was introduced in Breiman (2001). Here, each decision tree predicts a class, and the random-forest classifier "votes" for the most popular class. The *averaging* method was previously used in Breiman (1996, 2001) to solve regression problems. Here, each decision tree predicts a class with a certain probability. The argmax of the average of these probability estimates determines the pre-

diction of the random forest. In Biau et al. (2008), however, the authors prove that both types of random forests result in consistent classifiers for categorical responses. In this context, "consistent" means that the prediction becomes more accurate as the number of data points used in training increases. In the following, we will focus on averaging random forests, the type that is also implemented in *scikit-learn*, which we will use in the application studied in Section 6.

We consider a random forest $\mathcal{T} \coloneqq \{T_t \mid T_t = (V_t, E_t),\ t \in \{1, \dots, m\}\}$, represented as a set of m decision trees T_t, the so-called *base learners*, where each of them is constructed as described in Section 2.1. The nodes V_t of each tree $T_t \in \mathcal{T}$ are partitioned into decision vertices $V_{t,\mathrm{Dec}}$ and class vertices $V_{t,\mathrm{Cl}}$. In each decision vertex $v \in V_{t,\mathrm{Dec}}$, there is a split inequality of the form $\tilde{a}_{t,v}^T x \le \tilde{b}_{t,v}$ to recursively subdivide the space $\mathbb{R}^p$ into disjoint regions. Let $\gamma_{t,v} \in [0,1]$ be the fraction of data points in v which belong to class 1. The class predicted by the random forest for a given point $x \in \mathbb{R}^p$ then results from the average over the fractions $\gamma_{t,v_{t,x}} \in [0,1]$ for class 1 of each base learner, where $v_{t,x}$ represents the leaf belonging to the point x in each tree $t \in \{1, \dots, m\}$. If the value exceeds 0.5, class 1 is predicted, otherwise class 0:

$$c^{RF}\colon \mathbb{R}^p \to \mathbb{R}, \quad c^{RF}(x) = \begin{cases} 1, & \text{if } \frac{1}{m}\sum_{t=1}^{m} \gamma_{t,v_{t,x}} > 0.5 \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

The prediction certainty for the selected class results directly from the mean value of the average over the fractions $\gamma_{t,v_{t,x}} \in [0,1]$, i.e. from the expected value:

$$f^{\mathrm{RF}}\colon \mathbb{R}^p \to [0,1], \quad f^{\mathrm{RF}}(x) = \begin{cases} \frac{1}{m}\sum_{t=1}^{m} \gamma_{t,v_{t,x}}, & \text{if } c^{RF}(x) = 1 \\ 1 - \frac{1}{m}\sum_{t=1}^{m} \gamma_{t,v_{t,x}}, & \text{otherwise} \end{cases}.$$

We remark that for $m = 1$, random forests naturally reduce to ordinary decision trees.

3 Using Tree Classifiers to Define Mixed-Integer Constraints

In this section, we model the input-output relation of tree-based classifiers algebraically. The subdivision of space into polyhedral feasible and infeasible regions inferred from a trained random forest is converted into mixed-integer linear constraints.

3.1 Problem Setting

We consider a general mixed-integer and possibly non-linear program (MINLP) with decision variables $x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ of the following form:

$$\begin{aligned} \min \quad & c^T x && \text{(2a)} \\ \text{s.t.} \quad & Dx \le d && \text{(2b)} \\ & x \in \mathcal{F} && \text{(2c)} \\ & x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q, && \text{(2d)} \end{aligned}$$

where $D \in \mathbb{R}^{q \times p}$, $d \in \mathbb{R}^w$, $c \in \mathbb{R}^p$ and $q \in \{0, \dots, p\}$. The condition $x \in \mathcal{F}$ represents partially unknown or "difficult-to-state" (i.e. algorithmically intractable) constraints. Without Constraint (2c), Model (2) is a classical mixed-integer linear problem (MIP), and thus NP-hard in general.

We now study the question of how it can be decided whether for a concrete variable assignment $x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ an implicitly given condition $x \in \mathcal{F}$ holds or not, and how this condition can be formulated as mixed-integer linear constraints. To this end, we assume in our data-driven approach that we have n concrete realisations of data points $X_i \in \mathbb{Z}^{p-q} \times \mathbb{R}^q, i \in \{1, \dots, n\}$ for which it is known whether $X_i \in \mathcal{F}$ is fulfilled or not, for example through a simulation or through explicit labelling by an expert. Each data point X_i can thus be assigned to a class

$Y_i \in \{0,1\}$, where $Y_i = 1$ if $X_i \in \mathcal{F}$ holds, and $Y_i = 0$ otherwise. This results in a training data set (X, Y) with $X \in \mathbb{Z}^{n\times(p-q)} \times \mathbb{R}^{n\times q}$ and $Y \in \{0,1\}^n$, which can be used to train a classifier. The classifier can make a prediction for any $x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ whether $x \in \mathcal{F}$ holds or not. In our exposition, we assume that the chosen trained classifier is accurate enough to trust its output.

3.2 Algebraic Reformulation of a Random-Forest Classifier

In the following, we introduce a mixed-integer algebraic reformulation of a random forest

$$\mathcal{T} = \{T_t \mid T_t = (V_t, E_t),\ t \in \{1,\dots,m\}\}$$

consisting of m individual binary decision trees, each with a maximum depth of k, as defined in Section 2.2. In a random-forest classifier, the labelling of a point $\tilde{x} \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ results from the individual predictions of the involved decision trees. In order to incorporate these predictions into a mixed-integer program, we have to model algebraically for each binary decision tree $t \in \{1,\dots,m\}$ the path

$$P_{\tilde{x}} = \{(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), \dots, (v_{i_{\tilde{k}-1}}, v_{i_{\tilde{k}}})\}, \quad \text{with } v_{i_1}, \dots, v_{i_{\tilde{k}-1}} \in V_{t,\mathrm{Dec}} \text{ and } v_{i_{\tilde{k}}} \in V_{t,\mathrm{Cl}}$$

of length $\tilde{k} \le k$ from the root $v_{i_1} \in V_{t,\mathrm{Dec}}$, along a sequence of intermediate decision nodes in $V_{t,\mathrm{Dec}}$ to one of the leaves in $V_{t,\mathrm{Cl}}$. This path depends on the results of the split inequalities along the $\tilde{k}-1$ decision vertices from $V_{t,\mathrm{Dec}}$. At each decision vertex, if $\tilde{x}$ satisfies $\tilde{a}_v^T x \le \tilde{b}_v$, the left descendant is chosen, otherwise the path continues with the right descendant, until reaching one of the leaves in $V_{t,\mathrm{Cl}}$.

In order to formulate this classification with the help of mixed-integer constraints, we need to consider the two mutually exclusive linear inequalities

$$\tilde{a}_v^T x \le \tilde{b}_v \quad \text{and} \quad \tilde{a}_v^T x > \tilde{b}_v$$

assigned to each decision vertex $v \in V_{t,\mathrm{Dec}}$ within each tree $t \in \{1,\dots,m\}$. Using a standard modelling approach, this disjunction can be incorporated into Model (2) by introducing for each decision tree $t \in \{1,\dots,m\}$ and each decision vertex $v \in V_{t,\mathrm{Dec}}$ two binary auxiliary variables $z_{t,v}^{(i)} \in \{0,1\}$, $i \in \{1,2\}$. With these it is possible to activate ($z_{t,v}^{(i)} = 1$) or to deactivate ($z_{t,v}^{(i)} = 0$) such an inequality:

$$\tilde{a}_{t,v}^T x \le \tilde{b}_{t,v} + M_{t,v}^{(1)}(1 - z_{t,v}^{(1)}), \qquad \forall v \in V_{t,\mathrm{Dec}}, \forall t \in \{1,\dots,m\}, \tag{3}$$

$$\tilde{a}_{t,v}^T x - \varepsilon_{t,v} \ge \tilde{b}_{t,v} - M_{t,v}^{(2)}(1 - z_{t,v}^{(2)}), \qquad \forall v \in V_{t,\mathrm{Dec}}, \forall t \in \{1,\dots,m\}. \tag{4}$$

In this two constraints, the constants $M_{t,v}^{(i)} \in \mathbb{R}$, for $v \in V_{t,\mathrm{Dec}}$ and $i \in \{1,2\}$ have to be sufficiently large in order not to rule out otherwise feasible solutions:

$$M_{t,v}^{(1)} \ge \max_x\{\tilde{a}_{t,v}^T x\} - \tilde{b}_v, \quad \forall v \in V_{t,\mathrm{Dec}}, \forall t \in \{1,\dots,m\}, \tag{5}$$

$$M_{t,v}^{(2)} \ge -\min_x\{\tilde{a}_{t,v}^T x\} + \tilde{b}_v, \quad \forall v \in V_{t,\mathrm{Dec}}, \forall t \in \{1,\dots,m\}. \tag{6}$$

As MIP solvers cannot work with strict inequalities, we introduce a small constant $\varepsilon_{t,v} > 0$ in Equation (4). For numerical stability, $\varepsilon_{t,v}$ should be chosen as small as necessary, but as large as possible. For example, in order to ensure, say, six valid digits in the left-hand side of the inequality, one can choose

$$\beta := -5 + \lfloor \log_{10} |\tilde{b}_{t,v}| \rfloor \qquad \text{and} \qquad \varepsilon_{t,v} \ge 10^{\beta}, \qquad \forall v \in V_{t,\mathrm{Dec}}, \forall t \in \{1,\dots,m\}. \tag{7}$$

Because of the mutually exclusive conditions, at most one auxiliary variable can be activated per decision node $v \in V_{t,\text{Dec}}$:

$$z_{t,v}^{(1)} + z_{t,v}^{(2)} \leq 1 \qquad \forall v \in V_{t,\text{Dec}}, \forall t \in \{1,\dots,m\}. \tag{8}$$

After certifying for each decision rule whether the given data point $\tilde{x}$ satisfies it or not, we need to ensure that classification of the data point follows a connected path within each decision tree $t \in \{1,\dots,m\}$. For each node $v \in V_{t,\text{Dec}}$, we thus introduce the shorthand notation $l(v)$ for the first (left-hand) and $r(v)$ for the second (right-hand) successor. Only if the first split rule of a vertex $v \in V_{t,\text{Dec}}$ is activated, i.e. $z_{t,v}^{(1)} = 1$, the two inequalities given by Equations (3) and (4) of its left-hand successor $l(v)$ can be activated, too. The same applies to the second split rule of such a node and its right-hand successor $r(v)$. Therefore, we add as additional constraints:

$$z_{t,l(v)}^{(1)}, z_{t,l(v)}^{(2)} \leq z_{t,v}^{(1)} \qquad \forall v \in V_{t,\text{Dec}}, \forall t \in \{1,\dots,m\}, \tag{9}$$

$$z_{t,r(v)}^{(1)}, z_{t,r(v)}^{(2)} \leq z_{t,v}^{(2)} \qquad \forall v \in V_{t,\text{Dec}}, \forall t \in \{1,\dots,m\}. \tag{10}$$

To avoid a case distinction in the statement of Equations (9) and (10) to cover the cases $l(v) \in V_{t,\text{Cl}}$ and $r(v) \in V_{t,\text{Cl}}$, we introduce the two binary variables $z_{t,v}^{(1)}$ and $z_{t,v}^{(2)}$ also for the class vertices $v \in V_{t,\text{Cl}}$. For each leaf node, we thus require

$$z_{t,v}^{(2)} = z_{t,v}^{(1)} \qquad \forall v \in V_{t,\text{Cl}}, \forall t \in \{1,\dots,m\}. \tag{11}$$

In each of the decision trees belonging to the random forest $\mathcal{T}$, at most one of the binary auxiliary variables belonging to a leaf must be active, i.e.

$$\sum_{v \in V_{t,\text{Cl}}} z_{t,v}^{(1)} \leq 1 \qquad \forall t \in \{1,\dots,m\}. \tag{12}$$

We call a tree with an active leaf an *active* tree. The inequalities along the associated path must then be satisfied by $\tilde{x}$. To the contrary, a tree is *inactive* if all the inequalities in its nodes are deactivated via the corresponding z-variables, such that they are valid for any $x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$. Consequently, if $z_{t,\tilde{v}}^{(1)} = 1$, the associated decision tree $t \in \{1,\dots,m\}$ is active and $\tilde{x}$ is assigned to the leaf $\tilde{v} \in V_{t,\text{Cl}}$, which is then the *active* leaf of the tree. Finally, to ensure feasibility in Equation (2c), we can only allow solutions which the random forest assigns to class 1. If the arithmetic mean of the predictions of the individual decision trees for $\tilde{x}$ fulfil $\frac{1}{m}\sum_{t=1}^{m} \gamma_{t,v_{t,\tilde{x}}} > 0.5$, the random forest assigns label 1, otherwise 0. To represent this averaging logic algebraically, we model the *tree certainty* of each individual decision tree $t \in \{1,\dots,m\}$ via a binary variable $\Gamma_t \in [0,1]$. Thanks to the auxiliary variables $z_{t,v}^{(1)}$ we know which leaf is active in the respective tree. Thus, we can determine the tree certainty by multiplying the given constant fraction $\gamma_{t,v} \in [0,1]$ at leaf $v \in V_{t,\text{Cl}}$ for class 1, described in Section 2.2, via the corresponding auxiliary variable $z_{t,v}^{(1)}$:

$$\Gamma_t = \sum_{v \in V_{t,\text{Cl}}} \gamma_{t,v} \cdot z_{t,v}^{(1)} \qquad \forall t \in \{1,\dots,m\}. \tag{13}$$

For $\tilde{x}$ to be assigned to class 1, the sum of tree probabilities must be at least half the number of decision trees:

$$\sum_{t=1}^{m} \Gamma_t \geq \lceil 0.5m \rceil. \tag{14}$$

In summary, the following MIP model emerges if we replace Equation (2c) by our MIP representation of a trained random forest with averaging as the evaluation rule:

$$\begin{aligned}
\min \quad & c^T x && && (15a)\\
\text{s.t.} \quad & Dx \le d && && (15b)\\
& (3) - (4), && && (15c)\\
& (8) - (14) && && (15d)\\
& z^{(1)}_{t,v}, z^{(2)}_{t,v} \in \{0,1\} && \forall v \in V_t, \forall t \in \{1,\dots,m\}, && (15e)\\
& \Gamma_t \in \{0,1\} && \forall t \in \{1,\dots,m\}, && (15f)\\
& x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q. && && (15g)
\end{aligned}$$

In particular, Inequalities (3)–(4) and (8)–(14) ensure that the random forest predicts label 1, i.e. feasible, for the chosen solution $\tilde{x}$.

4 Optimisation over Random-Forest Classifiers

In order to demonstrate the modelling capabilities of random-forest classifiers within MIP formulations, we will now investigate three different use cases that are relevant in the real-world application studied later. They concern in particular the choice of optimization objective, as summarized in the following:

1. Assume that we are given a point $x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ which is infeasible according to the trained classifier. This point may e.g. represent a solution candidate which was "manually" found by some expert planner. An interesting question is now which minimum adjustments need to be made to x in order to reach a solution $\tilde{x} \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ that is feasible according to the classifier. This objective can be formulated straightforwardly, see Section 4.1.

2. As a second setting, let us assume we want to ensure the "reliability" of a chosen solution $x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$, i.e. the solution should not lie close to the split between two classes of points separated into feasible and infeasible. For this reason, we determine an optimal solution x so that a sphere of maximum possible radius around x is fully contained in a feasible class of the trained classifier, see Section 4.2.

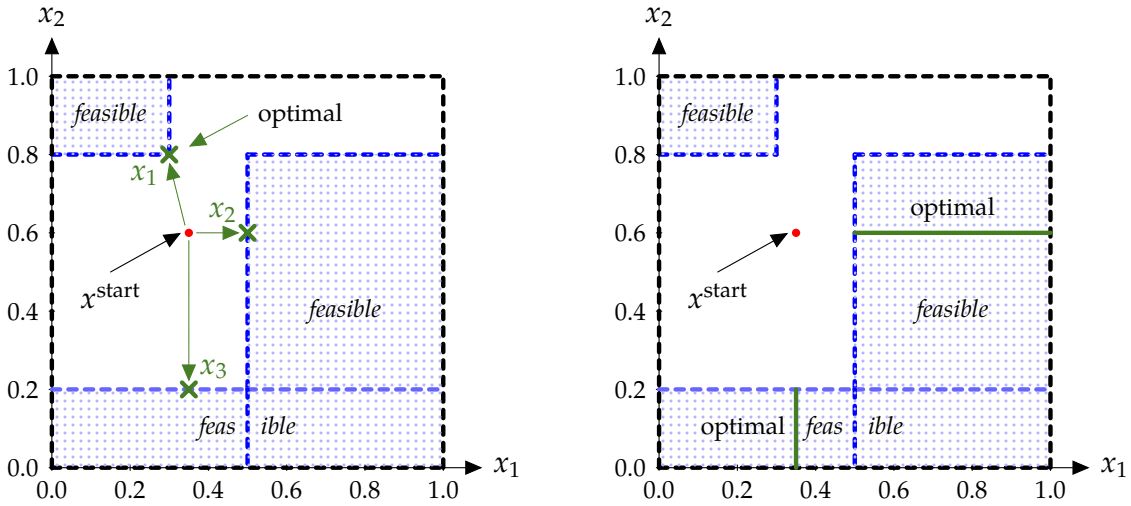
3. Finally, in Section 4.3 we show that for a random forest based on univariate decision trees we can also find the largest p-dimensional cuboid V_Q that is completely contained in a feasible class of the trained classifier.

In order to obtain meaningful results, we restrict x to the valid range of the classifier, i.e. the lower ($l \in \mathbb{R}^p$) and upper ($u \in \mathbb{R}^p$) bounds on the training data (X, Y). For the sake of simplicity, we assume that all parameter values are normalised to the interval $[0, 1]$:

$$0 = l_i \le x_i \le u_i = 1 \quad \forall i \in \{1,\dots,p\}. \tag{16}$$

4.1 Smallest Possible Adjustment to Make a Solution Candidate Feasible

In the first setting, we assume that an initial solution candidate $x^{\text{start}} \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ is given which is not recognised as feasible by the classifier. We search for a solution $x^{\text{opt}} \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ which is classified as feasible and which requires the minimum possible adjustment of the candidate x^{start}. We will exemplarily show two possibilities for such an adjustment – first minimizing the distance between x^{start} and x^{opt} and second minimizing the number of differing coordinates between the two.



(a) Smallest normalised distance.

(b) Least possible number of coordinates.

Figure 1: Starting at a given point x^{start} that is infeasible according to the random-forest classifier, find a feasible solution via a smallest possible adjustment (Figure 1a) or via adjusting the least possible number of coordinates (Figure 1b) in order to reach a feasible point.

The initial situation for both problems is shown in Figure 1a and Figure 1b, respectively, for a random-forest classifier based on univariate decision trees. The plotted green points x_1, x_2 and x_3 are potential optimal solutions for the first problem. The continuous green lines in Figure 1b represent equivalent optimal solutions for the second problem.

Finding the Closest Feasible Point. To find the closest feasible point to x^{start}, we minimize the distance

$$\min_x \|\delta\|_r,$$

where δ is the difference between x and x^{start}, i.e. $\delta = x - x^{\text{start}}$, and the r-norm is defined as $\|\delta\|_r := (\sum_{i=1}^{p} |\delta_i|^r)^{\frac{1}{r}}$. The most common norms are the norm ℓ_1, the Euclidean norm ℓ_2 and the infinity norm ℓ_∞. For instance, to calculate the absolute value of δ in the ℓ_1-norm, its entries are separated into their positive and negative parts, $\delta^+ \in \mathbb{R}_+$ and $\delta^+ \in \mathbb{R}_-$ respectively:

$$\begin{aligned} \delta_i &= \delta_i^+ - \delta_i^- && \forall i \in \{1,\dots,p\}, \\ \delta_i^+, \delta_i^- &\geq 0 && \forall i \in \{1,\dots,p\}, \\ |\delta_i| &= \delta_i^+ + \delta_i^- && \forall i \in \{1,\dots,p\}. \end{aligned}$$

Thus, to find the nearest point that is feasible according to the classifier, we can utilize the following MIP model:

$$\begin{aligned} \min \quad & \sum_{i=1}^{p} (\delta_i^+ + \delta_i^-) && && (17a) \\ \text{s.t.} \quad & Dx \leq d && && (17b) \\ & (15c) - (15f) && && (17c) \\ & \delta_i^+ - \delta_i^- = x_i - x_i^{\text{start}} && \forall i \in \{1,\dots,p\} && (17d) \\ & \delta_i^+, \delta_i^- \geq 0 && \forall i \in \{1,\dots,p\} && (17e) \\ & 0 \leq x_i \leq 1 && \forall i \in \{1,\dots,p\} && (17f) \\ & \delta_i^+, \delta_i^- \in \mathbb{R} && \forall i \in \{1,\dots,p\} && (17g) \\ & x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q, && && (17h) \end{aligned}$$

where the constraints in (17c) represent the feasibility inside the random-forest classifier.

Minimum the Number of Coordinates to be Adjusted Suppose we are looking for the smallest number of coordinates to change in the solution candidate x^{start} in order to obtain a feasible solution $x^{\text{opt}} \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$. We introduce a binary variable δ_i^{on} for each feature $i \in \{1, \dots, p\}$ that takes value 1 if $x_i \neq x_i^{\text{start}}$, and 0 otherwise:

$$M_{\delta_i}^- \cdot \delta_i^{\text{on}} \leq x - x_i^{\text{start}} \leq M_{\delta_i}^+ \cdot \delta_i^{\text{on}} \qquad \forall i \in \{1, \dots, p\}.$$

Here, $M_{\delta_i}^-$ and $M_{\delta_i}^+$ are again sufficiently large constants, which in this case, for example, can be defined as the distances between the boundary values of x and the initial value:

$$\begin{aligned} M_{\delta_i}^- &\coloneqq 0 - x_i^{\text{start}}, \qquad i \in \{1, \dots, p\}, \\ M_{\delta_i}^+ &\coloneqq 1 - x_i^{\text{start}}, \qquad i \in \{1, \dots, p\}. \end{aligned}$$

In order to minimize the number of adjusted coordinates, we sum over the δ_i^{on} in the objective function, which leads to the following optimisation problem:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{p} \delta_i^{\text{on}} && && (18a) \\
\text{s.t.} \quad & Dx \leq d && && (18b) \\
& (15c) - (15f) && && (18c) \\
& M_{\delta_i}^- \cdot \delta_i^{\text{on}} \leq x_i - x_i^{\text{start}} \leq M_{\delta_i}^+ \cdot \delta_i^{\text{on}} && \forall i \in \{1, \dots, p\} && (18d) \\
& 0 \leq x_i \leq 1 && \forall i \in \{1, \dots, p\} && (18e) \\
& \delta_i^{\text{on}} \in \{0, 1\} && \forall i \in \{1, \dots, p\} && (18f) \\
& x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q. && && (18g)
\end{aligned}$$

The constraints in (18c) again represent the trained random forest.

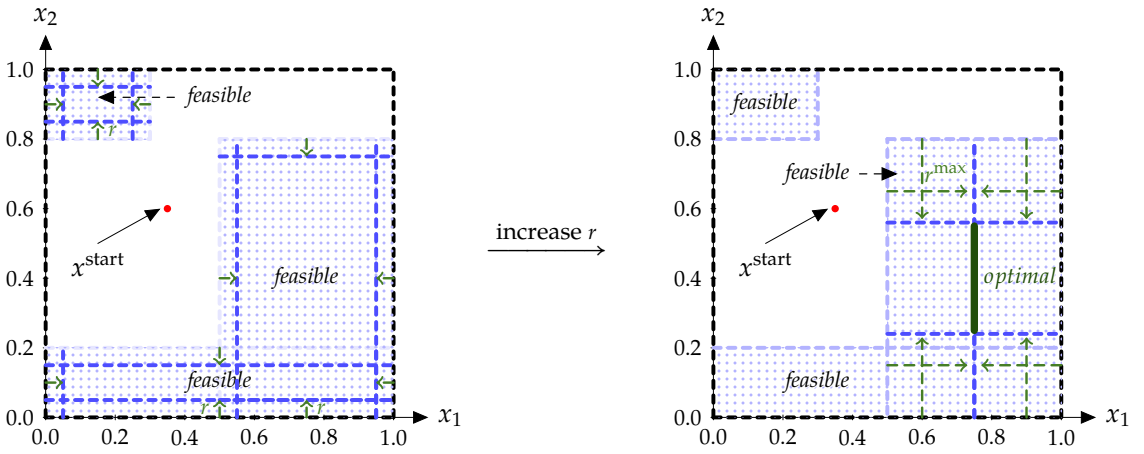
4.2 Finding Reliable Solutions

In many applications, a chosen solution $x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ is required to be feasible with a high degree of certainty. Thus, a solution on the boundary of a feasible class, as could be seen exemplarily in Figure 1a, should be avoided as it may entail infeasibilities due to parameter fluctuations or uncertainties.

For a single decision tree, all its classification areas result from the paths leading from the root to the leaves. These areas are determined as the intersection of finitely many halfspaces and are thus polyhedra. In turn, all classification areas of a random forest result from intersections of the areas implied by the individual decision trees and are therefore also polyhedra. Because of the boundary condition in Equation (16), they are actually polytopes. Thus, any solution $x^{\text{opt}} \in \mathbb{Z}^{p-q} \times \mathbb{R}^q$ lies within a polytope that depends on the classifications of the active decision trees.

In order to determine a *reliable* solution that is as far as possible from the boundaries, we introduce an optimisation variable $r \in [0, 0.5]$ which in the optimum models the smallest distance the point x^{opt} has from any of the edges of the feasible set. Due to convexity, we can shift each of its edges by r into the interior, as can be seen in Figure 2a:

$$\begin{aligned}
& \tilde{a}_{t,v}^T x \leq \tilde{b}_{t,v} + M_{t,v}^{(1)}(1 - z_{t,v}^{(1)}) - r && \forall t \in T_{\text{Dec}}, \\
& \tilde{a}_{t,v}^T x - \varepsilon_{t,v} \geq \tilde{b}_{t,v} - M_{t,v}^{(2)}(1 - z_{t,v}^{(2)}) + r && \forall t \in T_{\text{Dec}}, \\
& x_i - r \geq 0 && \forall i \in \{1, \dots, p\}, \\
& x_i + r \leq 1 && \forall i \in \{1, \dots, p\},
\end{aligned}$$



(a) Shift all inequalities by r inwards.

(b) Shift all inequalities by $r^{\max}$ inwards.

Figure 2: The feasible areas predicted by a random forest. We can reduce the size of the feasible sets by shifting all inequalities inwards by a value r in order to choose a reliable point (Figure 2a). If we choose the value of r as large as possible, we can limit ourselves to the most reliable points (Figure 2b).

where $M^{(1)}_{t,v}$ and $M^{(2)}_{t,v}$ are constants that have to be set as in Equations (5) and (6) and increased by 0.5, the maximum value of r. Figure 2b shows an exemplary result when maximising over r. We obtain a point x^{opt} that has the largest possible distance from all active edges and is thus maximally reliable.

Altogether, we can find a reliable point x^{opt} with predicted class 1, i.e. with $c^{RF}(x^{\text{opt}}) = 1$, by solving the following MIP:

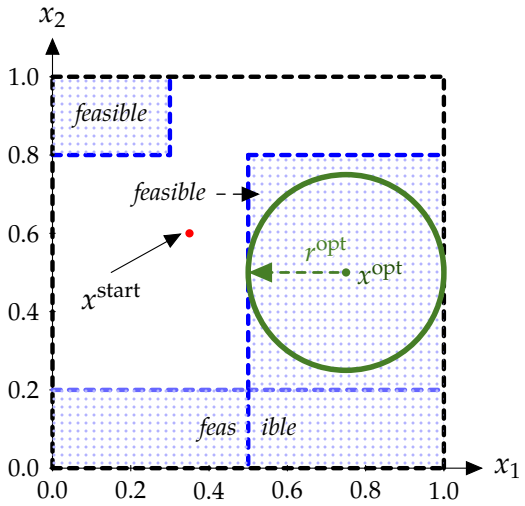
$$
\begin{aligned}
\max \quad & r && && (20a)\\
\text{s.t.} \quad & Dx \le d && && (20b)\\
& \tilde{a}^T_{t,v} x + r \le \tilde{b}_{t,v} + M^{(1)}_{t,v}(1 - z^{(1)}_{t,v}) && \forall v \in V_{t,\text{Dec}}, \forall t \in \{1,\dots,m\} && (20c)\\
& \tilde{a}^T_{t,v} x - \varepsilon_{t,v} - r \ge \tilde{b}_{t,v} - M^{(2)}_{t,v}(1 - z^{(2)}_{t,v}) && \forall v \in V_{t,\text{Dec}}, \forall t \in \{1,\dots,m\} && (20d)\\
& x_i - r \ge 0 && \forall i \in \{1,\dots,p\} && (20e)\\
& x_i + r \le 1 && \forall i \in \{1,\dots,p\} && (20f)\\
& (15\text{d}) - (15\text{f}) && && (20g)\\
& z^{(1)}_{t,v}, z^{(2)}_{t,v} \in \{0,1\} && \forall v \in V_t, \forall t \in \{1,\dots,m\} && (20h)\\
& r \in [0, 0.5] && && (20i)\\
& x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q. && && (20j)
\end{aligned}
$$

where the constraints in (20g) represent the feasibility inside the random-forest classifier.

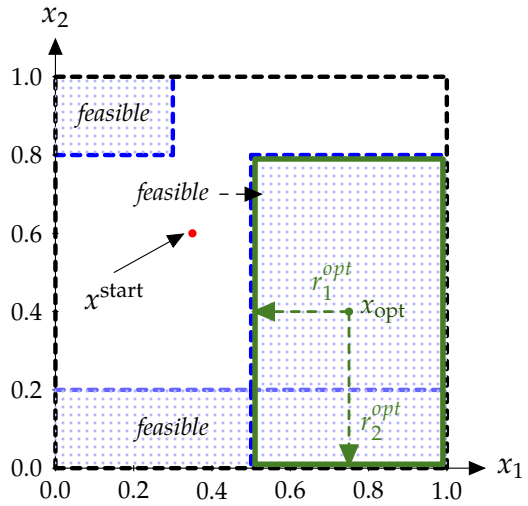
4.3 Largest Area in Univariate Classifier

The solution of the previously considered problem (20) automatically leads to an approximation of the largest feasible area with the help of a largest feasible sphere centred at x^{opt}, as can be seen in Figure 3a. An interesting related task is searching for the largest connected feasible region $\mathcal{U}^{\text{opt}}$ within the classifier. Besides the apparent motivation to identify a largest possible feasible range, we can also use this knowledge to analyse the trained classifier itself. For example, if that region turns out to be very large, it may be because only feasible data exist in it, but the reason could also be that this region is covered by too little training data. If, on the other

hand, the largest feasible area is small, this is also true for all other feasible areas and could thus indicate difficulty in explaining the feasible areas.



(a) Approximating the volume of the largest feasible area for a random forest with general decision trees via an inscribed sphere.



(b) Calcualting the volume of the largest feasible area exactly for a random forest with univariate decision trees via an inscribed cuboid.

Figure 3: Shifting all inequalities inwards is equivalent to finding the point whose maximal spherical or cuboidal environment also belongs to the feasible class.

Let us consider a general decision-tree-based classifier. To find the largest feasible region, we must first calculate the volumes of the polytopes implied by the random forest and compare them with each other. Computing the volume of a general polytope is a hard task. In practical applications, on the other hand, usually univariate decision trees are chosen, cf. the implementation of tree-based classfiers in *scikit-learn* (see Pedregosa et al. (2011)). This simplifies the arising polytopes to p-dimensional cuboids, whose volumes can by calculated by multiplying their edge lengths. To find the largest feasible cuboid within a univariate decision tree, its edge lengths can be determined by computing the centre x^{opt} using auxiliary variables $r_i \in [0, 0.5]$ storing the distance to the edges for each parameter $i \in \{1, \dots, p\}$. Similar as in Section 4.2, any inequality within a decision tree that depends on parameter i is shifted inwards by r_i. The resulting feasible region consists of points with at least r_i distance from each such inequality:

$$
\begin{aligned}
&\tilde{a}_{t,v}^T(x+r) \le \tilde{b}_{t,v} + M_{t,v}^{(1)}(1 - z_{t,v}^{(1)}) && \forall v \in V_{t,\text{Dec}}, \forall t \in \{1, \dots, m\},\\
&\tilde{a}_{t,v}^T(x-r) - \varepsilon_{t,v} \ge \tilde{b}_{t,v} - M_{t,v}^{(2)}(1 - z_{t,v}^{(2)}) && \forall v \in V_{t,\text{Dec}}, \forall t \in \{1, \dots, m\},\\
&x_i - r_i \ge 0 && \forall i \in \{1, \dots, p\},\\
&x_i + r_i \le 1 && \forall i \in \{1, \dots, p\}.
\end{aligned}
$$

For univariate decision trees, each decision rule refers to exactly one parameter. Thus, $\tilde{a}_{t,v}^T = k_{t,v} e_i$ is the $k_{t,v}$-th multiple of a unit vector e_i, $i \in \{1, \dots, p\}$, in this case. The left-hand sides of the first two inequalities above can then be simplified to

$$
\begin{aligned}
&\tilde{a}_{t,v}^T(x+r) = k_{t,v}(x_i + r_i) && k_{t,v} \in \mathbb{R}, \forall v \in V_{t,\text{Dec}}, \forall t \in \{1, \dots, m\}, && \text{(22a)}\\
&\tilde{a}_{t,v}^T(x-r) - \varepsilon_{t,v} = k_{t,v}(x_i - r_i) - \varepsilon_{t,v} && k_{t,v} \in \mathbb{R}, \forall v \in V_{t,\text{Dec}}, \forall t \in \{1, \dots, m\}. && \text{(22b)}
\end{aligned}
$$

The purpose of the constants $M_{t,v}^{(1)}$ and $M_{t,v}^{(2)}$ is to deactivate the first two constraints if $z_{t,v}^{(1)} = 0$

or $z^{(2)}_{t,v} = 0$. If $i \in \{1, \dots, p\}$ is the corresponding feature, they can be set to

$$M^{(1)}_{t,v} = k^{\max}_{t,v} u_i - k^{\min}_{t,v} l_i = k^{\max}_{t,v} \qquad \forall v \in V_{t,\mathrm{Dec}}, \forall t \in \{1, \dots, p\},$$
$$M^{(2)}_{t,v} = -k^{\min}_{t,v} l_i + k^{\max}_{t,v} u_i = k^{\max}_{t,v} \qquad \forall v \in V_{t,\mathrm{Dec}} \forall t \in \{1, \dots, m\}$$

in the univariate case, where $k^{\min}_{t,v} \coloneqq \min_{t,v} k_{t,v}$ and $k^{\min}_{t,v} \coloneqq \min_{t,v} k_{t,v}$ are the smallest or largest constant in an inequality associated with parameter i. This further simplification uses Constraint (16), which specifies that all parameter values are normalised.

Figure 3b shows an exemplary feasible point x^{opt} which is the centre of an axis-parallel cuboid. The minimum distance to each classifying inequality belonging to parameter $i \in \{1, \dots, p\}$ is given by r_i. Each edge of the cuboid has length $2r_i$, and its volume is calculated by multiplying the edge lengths. If we search for a feasible cuboid with largest possible volume, we can thus optimize with respect to the following objective function:

$$\max \prod_{i=1}^{p} r_i.$$

In order to linearise this objective, we rewrite it by using the logarithm and obtain

$$\max \log \left(\prod_{i=1}^{p} r_i \right) = \max \sum_{i=1}^{p} \log(r_i).$$

It remains to linearise the individual summands $\log(r_i)$, or more generally the functions $f_i \colon \mathbb{R}_+ \setminus \{0\} \to \mathbb{R}, r_i \mapsto \log(r_i)$, for all $i \in \{1, \dots, p\}$. For this purpose, we use the incremental method by Markowitz and Manne (1957), see e.g. Vielma (2015) for more recent results on this method. First, we define $s_i + 1$ breakpoints $(r_{i_k}, r^{\log}_{i_k})$, $i \in \{0, \dots, p\}$, where the r_{i_k} have to be ascending and logarithmically distributed in the subset $]0, 0.5]$ of the domain of f_i and the $r^{\log}_{i_k} = \log r_{i_k}$ represent the respective logarithmic value. At these breakpoints, we disjointly subdivide the domain of f_i into s_i-many subdomains. For each of these subdomains, indexed by $k \in \{1, \dots, s\}$, we further need a binary variable $z_{i_k} \in \{0, 1\}$ that registers whether it is active or inactive and a continuous variable $\delta_{i_k} \in [0, 1]$ that contains the exact proportional location of r_i in the k-th subdomain if it is active. The following mixed-integer linear inequalities can then be used to track the value of r_i:

$$r_i = r_{i_0} + \sum_{k=1}^{s_i} \left(r_{i_k} - r_{i_{k-1}} \right) \delta_{i_k} \qquad \forall i \in \{1, \dots, p\}, \tag{23a}$$
$$r^{\log}_i = r^{\log}_{i_0} + \sum_{k=1}^{s_i} \left(r^{\log}_{i_k} - r^{\log}_{i_{k-1}} \right) \delta_{i_k} \qquad \forall i \in \{1, \dots, p\}, \tag{23b}$$
$$z_{i_k} \le \delta_{i_k} \qquad \forall k \in \{1, \dots, s-1\}, \forall i \in \{1, \dots, p\}, \tag{23c}$$
$$\delta_{i_{k+1}} \le z_{i_k} \qquad \forall k \in \{1, \dots, s-1\}, \forall i \in \{1, \dots, p\}, \tag{23d}$$
$$z_{i_k} \in \{0, 1\} \qquad \forall k \in \{1, \dots, s-1\}, \forall i \in \{1, \dots, p\}, \tag{23e}$$
$$\delta_{i_1} \le 1, \delta_{i_{s_i}} \ge 0 \qquad \forall i \in \{1, \dots, p\}. \tag{23f}$$

Overall, in the case of a univariate random forest with averaging, the following MIP model

computes the largest feasible cuboid:

$$\max \quad \sum_{i=1}^{p} r_i^{\log} \tag{24a}$$

$$\text{s.t.} \quad Dx \le d \tag{24b}$$

$$(23a) - (23f) \tag{24c}$$

$$\tilde{a}_{t,v}^T(x+r) \le \tilde{b}_{t,v} + M_{t,v}^{(1)}(1 - z_{t,v}^{(1)}) \qquad \forall v \in V_{t,\text{Dec}}, \forall t \in \{1,\dots,m\} \tag{24d}$$

$$\tilde{a}_{t,v}^T(x-r) - \varepsilon_{t,v} \ge \tilde{b}_t - M_{t,v}^{(2)}(1 - z_{t,v}^{(2)}) \qquad \forall v \in V_{t,\text{Dec}}, \forall t \in \{1,\dots,m\} \tag{24e}$$

$$x_i - r_i \ge 0 \qquad \forall i \in \{1,\dots,p\} \tag{24f}$$

$$x_i + r_i \le 1 \qquad \forall i \in \{1,\dots,p\} \tag{24g}$$

$$(15d) - (15f) \tag{24h}$$

$$z_{t,v}^{(1)}, z_{t,v}^{(2)} \in \{0,1\} \qquad \forall v \in V_t, \forall t \in \{1,\dots,m\} \tag{24i}$$

$$x \in \mathbb{Z}^{p-q} \times \mathbb{R}^q, \tag{24j}$$

where the constraints in (24c) represent the incremental method and the constraints in (24h) the feasibility inside the random-forest classifier. The decision rules defining (24d) and (24e) stem from univariate decision trees, which makes it possible to simplify the left-hand sides as in Constraints (22a) and (22b).

5 Improving the Stability of LVDC Networks

We will now describe an indicative use case for the optimisation models from Section 4, namely the design of stable direct-current (DC) networks. This application is very well suited to validate the derived MIP reformulation of tree-based classifiers, since it involves a large number of network parameters to be optimised, which come from a large range of possible values – from one to several orders of magnitude. Furthermore, different strategies for parameter adjustments are required here, according to the precise problem setting.

The overall approach for stability improvement with focus on the generation of the stability classes and the generation of the random-forest classifiers is described in Roeder et al. (2021). The remainder of this section summarises the approach and elaborates on some details concerning the input data, classifiers and optimisation scenarios.

5.1 Stability of LVDC Networks

Typically, electricity distribution networks operate with alternating current (AC). AC networks have been studied intensively in the literature, e.g. in Carpentier (1962); Mary et al. (2012); Aigner et al. (2021a), also in their linearised version as DC networks, see e.g. Aigner et al. (2021b). Direct current networks with system voltages below 1500 V (so-called *low-voltage direct-current (LVDC) grids*, see Azaioud et al. (2021) for a detailed introduction and International Electrotechnical Commission (2016, Section 4.2, p. 19) for the definition of the technical standards) are of great importance for the realisation of an efficient, decentralised energy supply with an increasing share of renewable energy, as explained in Azaioud et al. (2021); Weiss et al. (2015); Gao et al. (2019). In short, the reason is that renewable energy generators, such as photovoltaic systems, storage systems such as batteries, and consumers, e.g. charging stations for e-mobility or computer systems, operate on a DC basis. The interconnection of these components in DC microgrids, which are typically operated as subsystems of AC grids, avoids unnecessary DC-AC and AC-DC conversion and enables a cost-efficient subgrid design as well as an increased self-consumption of renewable energies, see Azaioud et al. (2021). An essential

element for the reliable design and operation of LVDC microgrids is the preservation of network stability. Switching operations may generate high-frequency AC currents and can cause the LVDC grid to oscillate, which may occur in particular when networks are reconfigured, e.g. when adding or removing loads and sources, cf. Ott et al. (2015a). Figure 4 shows a simplified model of a four-terminal DC microgrid network (4-TLN), as can be realised in an experimental laboratory setup. It serves as a development platform for real-world implementations, e.g. as operated at the Fraunhofer IISB. The LVDC circuit model comprises two unidirectional sources – a DC source, which mimics the input from the AC network after conversion and a photovoltaic source, as well as two unidirectional loads, respectively. In the following, we summarise the necessary details for understanding the application. For details on stability criteria for DC power distribution systems, we refer to Riccobono and Santi (2014).

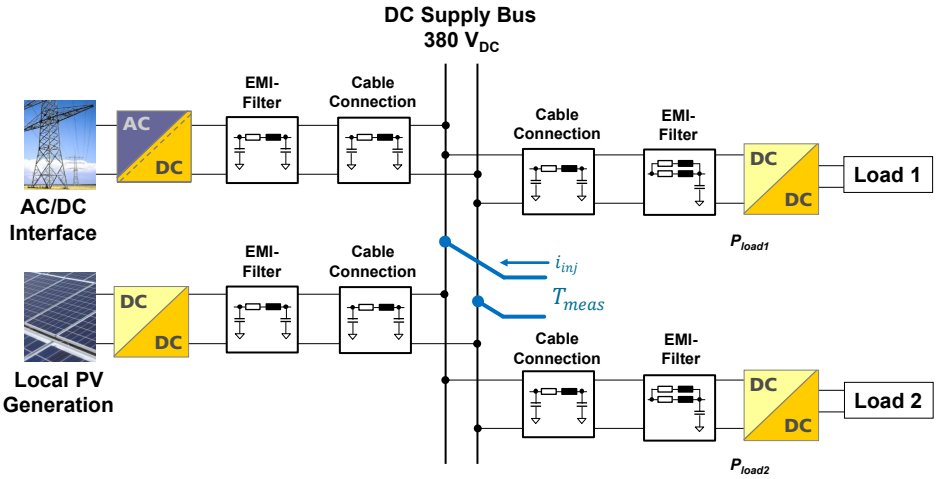


Figure 4: Model LVDC grid with bus architecture, which can be realised in an experimental laboratory setup.

The network is realised with a decentralised control structure, which exhibits inherent reliability in case of failures (Gao et al., 2019; Wunder et al., 2015). It comprises a current-mode droop control scheme as primary control (Gao et al., 2019; Wunder et al., 2015; Ott et al., 2015b) and a decentralised secondary control scheme for power-sharing as well as performance and voltage regulation (Wunder et al., 2015). The droop control scheme employs characteristic curves which control the current input from the sources into the network and keep the bus voltage at 380 V. The voltage droop concept and the characteristic curves are further discussed in Ott et al. (2015b). The impedance measurement and the subsequent stability analysis, either by simulation or by measurement, is conducted by injecting a small-signal alternating current i_{inj} with varying frequency ω at the bus. In the simulation, we calculate the impedance ratio T_{bus} from the source and load impedances Z_S, Z_l (Riccobono and Santi, 2014) according to the relationship

$$T_{\text{bus}} = \frac{Z_S}{Z_l} = \frac{V_S I_l}{I_S V_l}. \tag{25}$$

The impedances are calculated from the voltages V_S, V_l and the currents I_S, I_l, which are determined at two measuring resistors next to the small-signal source feed. The measurement determines the complex impedance T_{meas} that contains equivalent information on stability as T_{bus}, which is used for determining stability margins. The complex impedance spectrum is analysed, and the system stability is assessed by applying the gain margin and phase margin criterion, see Riccobono and Santi (2014). The gain margin and phase margin provide sufficient, but not necessary stability conditions and may be derived from the Nyquist plot, which represents the impedance ratio in the complex plane, or the Bode plot, which depicts the impedance spectrum

as amplitude given in decibel and the phase angle given in degrees, see (Riccobono and Santi, 2014). According to (Riccobono and Santi, 2014), the total input-to-output transfer function of two cascaded individually stable subsystems, such as the source and load system, may be written as

$$G_{sl} = G_s G_l \frac{1}{1 + T_{\mathrm{MLG}}}, \tag{26}$$

with the *minor loop gain* T_{MLG} of the system depicted in Figure 4 defined as

$$T_{\mathrm{MLG}} \coloneqq \frac{Z_s}{Z_l} = T_{\mathrm{bus}}. \tag{27}$$

The interconnected system is stable if the Nyquist contour of T_{MLG} does not encircle the point $(-1, 0)$ in the complex plane (Riccobono and Santi, 2014). The gain margin and phase margin criterion allow that $|Z_s| > |Z_l|$ in certain frequency ranges while ensuring respective margins such that the Nyquist criterion is satisfied (Riccobono and Santi, 2014). Maintaining a gain margin ensures that the amplitude $|T_{\mathrm{bus}}(j\omega_{\mathrm{pc}})|$ at those *phase cross-over frequencies* ω_{pc}, where $\mathrm{Im}(T_{\mathrm{bus}}(j\omega_{\mathrm{pc}}))$, i.e. where the phase angle is $-180°$, is sufficiently distant from the point $(-1, 0)$. The *gain margin* GM is defined as

$$\mathrm{GM} \coloneqq \frac{1}{|T_{\mathrm{bus}}(j\omega_{\mathrm{pc}})|}, \tag{28}$$

see Åström and Murray (2008). In the Bode plot, where the amplitude is depicted in decibels, the gain margin can be determined as

$$\mathrm{GM\ dB} = 20 \log_{10} \mathrm{GM} = 0\ \mathrm{dB} - 20 \log_{10} |T_{\mathrm{bus}}(j\omega_{\mathrm{gc}})|. \tag{29}$$

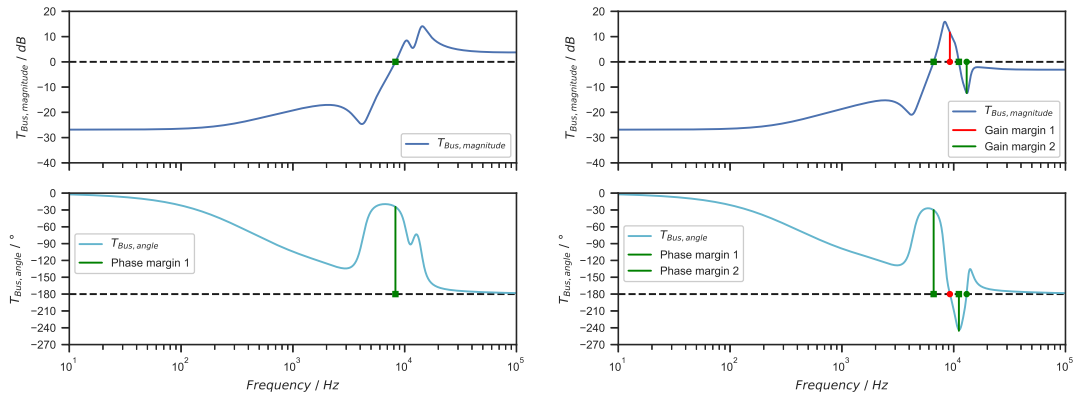
The *phase margin* at the *gain frequency* ω_{gc} is defined as

$$\mathrm{PM} \coloneqq 180° - |\arg T_{\mathrm{bus}}(j\omega_{\mathrm{gc}})|, \tag{30}$$

where $|T_{\mathrm{bus}}(j\omega_{\mathrm{gc}})| = 1$, and denotes the angle between the point $(-1, 0)$ and the intersection of $T_{\mathrm{bus}}(j\omega_{\mathrm{gc}})$ with the unit circle, see Riccobono and Santi (2014). The choice of gain and phase margins depends on the application. Typical values for gain margin are GM = 2 to 6 dB and PM = $30° - 60°$, respectively (Åström and Murray, 2008; Riccobono and Santi, 2014). For further investigations, the network was labelled to be stable whenever the determined gain margins and phase margins both exceeded critical values according to

$$\mathrm{GM} \geq 6\ \mathrm{dB}, \quad \mathrm{PM} \geq 45°. \tag{31}$$

Otherwise, the network was labelled as unstable. With these parameters, the exciting amplitude must be damped by at least $-6\ \mathrm{dB} \approx 0.5$, and if this not the case, the phase angle must be at least $45°$ distant to $180°$, the range of the opposite-phase oscillation in resonance (Riccobono and Santi, 2014). Figure 5 illustrates the identification of stable and unstable states in the Bode plot for the circuit model of Figure 4, for a stable state at the default network parameter settings in Figure 5a, and for parameter settings leading to instability in Figure 5b. With the Bode plot, the phase margins are determined at GM = 0 and the gain margins are determined at PM = $180°$. The margin exceeding the limits where the network is labelled as unstable according to Equation (31) is indicated in red.



(a) Stable network (b) Unstable network

Figure 5: Determination of the gain and phase margins from the Bode plot for a stable Figure 5a and an unstable network state Figure 5b. The margin exceeding the limits, which causes the network to be labelled as unstable, is indicated in red.

5.2 Parameter Variation in the Network

In a simulator, the small-signal DC network impedance was automatically calculated for multiple input parameter settings and the stability was analysed with the gain and phase margin limits given in Equation (31), also providing the labels for the stability class. The grid input parameters are distributed by Latin hypercube sampling (LHS, see (Lin and Tang, 2015)), autoscaling the data either on a linear or a logarithmic scale between the minimum and maximum values. The input parameters of the DC network which are varied in the LHS design and their possible values are briefly described in Table 8 in the appendix.

For the development of the random-forest surrogate classifier and for testing the optimisation models, simulations varying the parameter k_{AC}, k_{PV} and P_{load1}, P_{load2} as well as the input parameters depicted in Table 8 were conducted (Roeder et al., 2021). The summary as well as the imbalance ratio of stable and unstable states are depicted in Table 1. By automated circuit

Parameter Variation	Number of simulation results	Number of stable states	Number of unstable states	Imbalance ratio (in %)
k_{AC}, k_{PV}, P_{load1}, P_{load2}	24,770	23,162	1,608	94.0 (6.0)
all input parameters	49,018	24,685	24,333	50.4 (49.6)

Table 1: Summary and imbalance ratio of the stable and unstable states obtained from the simulations.

modelling, 24,770 resp. 49,018 input parameter combinations and the resulting stability labels were calculated, which serve as predictors and labels in the random-forest classifiers, respectively. With the parameter combinations, different ratios of stable and unstable states, i.e. imbalance ratios were obtained. As the training of accurate and reliable classifiers requires balanced input data, specific measures were taken in the preparation of the random-forest classifiers as described in the next section.

5.3 Preparation of the Random-Forest Classifiers

The random-forest classifiers were trained on the data as depicted in Table 1. The data sets were split into a training and an independent test data set with a split ratio of 70% down to 30%. Hyperparameter optimisation (HPO) of the random-forest classifiers was conducted on the training data using a grid search strategy and a 10-fold stratified cross-validation to maximise

the balanced accuracy and to minimise the *false positives*, i.e. the cases where an unstable state is predicted as stable (Roeder et al., 2021). For the investigation of the optimisation models, two random-forest classifiers were prepared. For both classifiers, the parameters P_{load1}, P_{load2} were excluded from the training to enable optimisation within their complete parameter range. One classifier, further denoted as the *14-parameter classifier*, was trained allowing all other input parameters to be varied. The second classifier, which is further denoted as the *two-parameter classifier*, was trained to allow the variation of k_{AC}, k_{PV} as an example for droop control parameter adjustment. The HPO parameters were set differently to investigate the achievement of a preferably large leaf population and to limit the tree complexity while providing stable, balanced accuracy. The 14-parameter classifier was adjusted to have a minimum number of 11 samples per leaf, the number of estimators set to 1,000 trees, and the maximum depth of the trees was set to 20. The two-parameter classifier was adjusted to have a minimum number of 100 samples per leaf, the number of estimators was limited to 2,000 trees, and no limit was provided for the tree depth, i.e. the trees may be split until the minimum number of samples per leaf is obtained. The HPO was enabled to adjust the tree splitting criterion, the maximum number of features to be sampled and the adjustment of the class weights. The balanced accuracy, the parameters of the confusion matrix, and the selected hyperparameters are given in Table 2. The classifiers provide well-balanced accuracy on the training and test data and minimise the false-positive rate. During HPO, the class weights are adjusted to compensate for the imbalance in the data sets. Entropy was selected as the splitting criterion for both classifiers. All features were selected in the case of the two-parameter classifier, whereas the number of parameters remained at the default setting for the variation of all input parameters. The balanced accuracy, the parameters of the confusion matrix, the class weights of the input classes to compensate for their imbalance as well as the splitting criterion and the maximum number of sampled features are given in Table 2. The classifiers provide high balanced accuracy, which provides a corrected accuracy measure in the presence of imbalance, on the training and test data. Additionally, the classifiers specifically minimise the false-positive rate, i.e. the prediction of an unstable network as stable. Overall, accurate and reliable input parameters to the optimisation are provided.

Data set	**Sample**	**Balanced accuracy**	**True positives:** predict stable as stable	**False negatives:** predict stable as unstable	**True negatives:** predict unstable as unstable	**False positives:** predict unstable as stable	**Class weights** class 0 | class 1	**Splitting criterion** entropy or gini	**Maximum number of features**
Variation of k_{AC}, k_{PV}	training	93%	81%	12%	7%	0%	0.95 | 0.05	entropy	N_{features}
	test	92%	82%	12%	6%	0%			
Variation of all input	training	93%	46%	5%	48%	2%	0.55 | 0.45	entropy	$\lfloor\sqrt{N_{\text{features}}}\rfloor$
	test	86%	42%	9%	45%	5%			

Table 2: Balanced accuracy, parameters of the confusion matrix and selected hyperparameters of the random-forest classifiers.

6 Case Study for Determining Stable LVDC Network Settings

In this section, we use mixed-integer optimisation models to find the best possible parameter setting for an LVDC network starting with an initial setting x^{start}. The stability of the network is assessed exclusively from the prediction of the respective random-forest classifiers presented in Section 5. In addition, from the input table 8 we know the minimum value l_p and the maximum value u_p for each parameter p in the set of parameters P. To visualise and to describe the solutions from the different mixed-integer problems, we use the smaller two-parameter classifier trained on the two-dimensional data set with 25,000 instances of k_{AC} and k_{PV}. For predicting the stability of the whole LVDC network, we use the large 14-parameter classifier trained on the dataset (X, Y) described in Section 5, which consists of 50,000 concrete network settings. A data point $X_i \in \mathbb{R}^{14}$, $i \in \{1, \ldots, 50,000\}$, represents the value for each of the 14 parameters $p \in P$, and the corresponding label $Y_i \in \{0, 1\}$ describes the resulting network status "stable" (1) or "unstable" (0). Apart from the best found random-forest classifier with 1,000 decision trees, each consisting of 920 to 1,020 paths, and a maximum depth of 20, we will also use increasing subsets of the decision trees to demonstrate and discuss the performance of the MIP models.

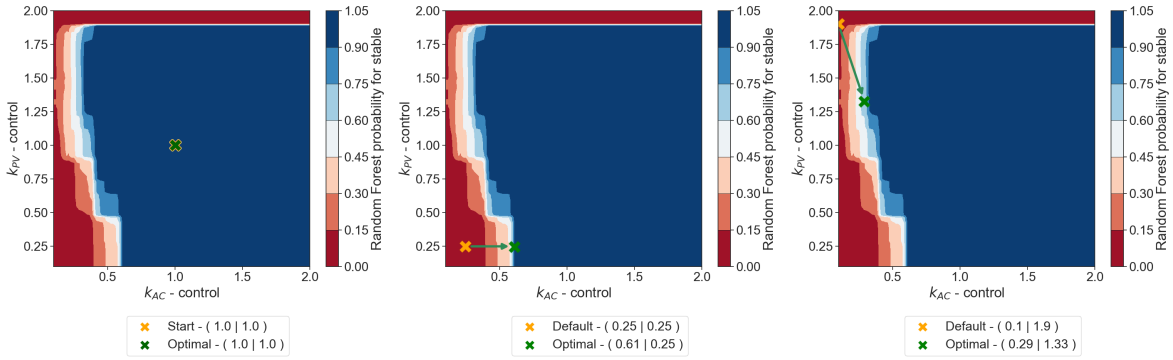
Both random-forest classifiers have been trained with the Python machine learning library *scikit-learn* (Pedregosa et al., 2011), and consist of individual univariate decision trees, each trained with an improved CART algorithm (Scikit-Learn, nd). The predictions of the random-forest classifiers are based on the averaging method. As MIP solver, we have used Gurobi (Gurobi Optimization, LLC, 2020), and all presented solution times have been obtained on Intel(R) Xeon(R) CPU E3-1240 v5 @ 3.50 GHz processors with 4 CPUs and 32 GB RAM.

6.1 The MIP Models

We assume to be given an established LVDC network and a perfect random-forest classifier representing the relation of the network input parameter values to LVDC network stability. The latter means that the classifier predicts a stable state exactly when the LVDC network is stable. Furthermore, we are given the initial input parameter setting of the network. We now consider several different tasks with respect to choosing the network parameters in order to obtain a stable network.

6.1.1 Minimal Adjustment of Parameters to Make a Network Stable (Adjust)

The first task, which henceforth will be called Adjust, is to bring the network into a stable state, using as few parameter changes as possible. Each parameter represents either a network component that needs to be replaced or a software parameter that needs to be adjusted. The fewer parameters have to be changed, the easier and less expensive the adaptation is. To solve this problem, we consider the MIP model described in Equations (18a) to (18g).



(a) Change nothing (stable state) (b) Change one parameter (c) Change two parameters

Figure 6: Solving problem Adjust using the two-parameter classifier.

Before we examine the solution times of this task, we first want to ensure the suitability of the solution. For that purpose, we consider a two-dimensional problem where the distribution of the stable and unstable areas is easy to visualize and where the parameter decisions and the correctness of the solution are well comprehensible. For that purpose, we will use the small random-forest classifier trained only with the two control parameters k_{AC} and k_{PV} that were introduced in Sections 5.1 and 5.3. After training the random-forest classifier with the given data set, it can be used to classify an arbitrary point $x \in \mathbb{R}^2$ as stable or unstable within the given minimum ($l_p = 0$) and maximum value ($u_p = 2$) for $p \in \{k_{\text{AC}}, k_{\text{PV}}\}$. The optimal assignment can be taken from the colour map in Section 6.1.1, which corresponds to the probability of the network being stable. The deeper the area is in the dark blue, the more confident the random-forest classifier is that it belongs to the stable class.

An optimal solution for Adjust changes as few parameters as possible, but the magnitude of the adjustment does not matter, which means that the objective is zero in case we start in a stable state; see Section 6.1.1. Starting in an unstable state, like in Section 6.1.1, we are returned the number of parameters to be adjusted and receive the specific changes to make to reach a stable state. For Section 6.1.1, this means, for example, that all solutions with $k_{\text{AC}} \geq k_{\text{AC}}^{\text{opt}}$ are equivalent to the point $(k_{\text{AC}}^{\text{opt}}, k_{\text{PV}}^{\text{opt}})$ found. The optimisation model is only about achieving a stable network setting. The neighbourhood of the point is unimportant to the solution, as can be seen in Section 6.1.1, where it lies in a light blue area, close to unstable regions.

We will now consider the same task using the 14-dimensional data set and the larger 14-parameter classifier. As described in Section 5, we found that a good prediction requires a random-forest classifier with 1,000 trees, each with 920 to 1,020 paths and a maximum depth of 20. This number of trees is already a challenge to the solution time of the MIP model. We verify that this is not due to incorrect modelling by comparing the solution times for differently sized subsets of the given decision trees with comparable objective values. With 100 decision trees, the MIP model finds a solution within 120 seconds, the classifier with 500 decision trees already needs 7,550 seconds, and the model incorporating the full classifier with 1,000 decision trees is not solvable within 4 hours.

The solution time of the MIP model can be reduced significantly by using the following iterative process. The integer objective value $G = \sum_{i=1}^{|P|} \delta_i^{\text{on}}$, which determines the number of parameters to be adjusted, is no longer to be chosen via the MIP model, but instead in an outer for-loop. The MIP models to be solved inside the loop are fixed to an objective value representing a certain number of parameters to be adjusted, $G_k = \sum_{i=1}^{|P|} \delta_i^{\text{on}} = k$, and are thus transformed to feasibility problems. We start with zero parameters to be adjusted and increase this number by one in each step. If the current MIP with the additional constraint $G_k = k$, $k \in \{0, \ldots, |P|\}$ is infeasible, then proceed with the fixation $G_{k+1} = k + 1$. On the other hand, if the solver finds a

solution for the current k, this value k is the optimal solution for the original MIP problem. The for-loop can then be terminated. Note that infeasibility of an integer problem can usually be determined very fast in practice, which is why we use linear search with a constantly increasing value of k here instead of binary search.

As a result of this approach, even the model incorporating 1,000 decision trees could be solved within less than 3.5 hours of computation time, see Table 3. The table shows an overview of the optimisation runs performed. Besides the number of decision trees, the required number of continuous and binary variables in the MIP model as well as the required computation time and the number of parameters to be changed (target of ADJUST) are documented.

RF classifier / number of trees	Number of MIP variables		Computation time (s)	Objective value / adjusted parameters
	continuous	binary		
10	95	48,899	4	1
100	185	487,244	132	1
500	585	2,427,989	3,200	1
1,000	1,085	4,855,969	12,223	1

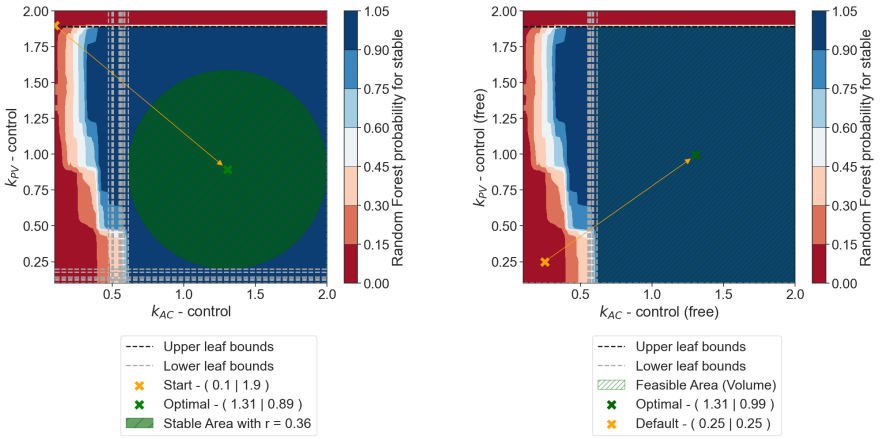
Table 3: Results for improved ADJUST procedure

6.1.2 Most Stable Point Possible Within an Already Existing Network (RELIABLE)

As a second objective, we study determining a network setting that is as stable as possible so that small fluctuations in the parameters do not affect stability. For this purpose, we determine a point in the centre of a stable region by using Equations (20a) to (20j). We call this task RELIABLE. Unlike in the simplified exposition of the optimisation model (20), the considered parameters $p \in P$ are defined on different intervals $[l_p, u_p]$. In our application, $r \in [0, 0.5]$ shall reflect the normalised safety margin. The conversion of the values is straightforward to incorporate by the linear scaling $\bar{r} = (r_p - r_p^{\min})/(r_p^{\max} - r_p^{\min})$ for all parameters $p \in P$. To understand the solution of the MIP model, we first consider the two-dimensional random-forest classifier in Figure 7a. The model finds the largest possible intersection of more than 50% of the decision trees in the classifier such that the corresponding area is predicted to be "stable". Since all decision trees are univariate, the intersection is a rectangle. For the objective function of RELIABLE, however, this is not necessary. The green centre of the optimal circle is the most reliable point w.r.t. the random-forest classifier. In both parameter dimensions, the coordinates of the center point can be changed additively by $r_p = 0.36 \cdot (u_p - l_p)$ without becoming unstable. All points within the circle have at least 50.2% certainty and are thus predicted to be stable by the classifier. The actual certainty, which is the sum of all decision tree predictions of the specific position, exceeds this value. It is easy to see in Figure 7a that the solution is not unique and that points $(k_{\mathrm{AC}}, k_{\mathrm{PV}})$, with $k_{\mathrm{PV}} = 1.31$ and $k_{\mathrm{PV}} \in [0.89, 1.87]$ are equivalently reliable solutions for the MIP model.

Considering only the colour map, one could suppose that it is possible to move the centre slightly to the upper left and thus increase the circle radius. However, this solution cannot be found, because almost every decision tree restricts the stable sphere with the inequality $k_{\mathrm{AC}} \approx 0.5$, as can be seen from the white dashed lines on the left-hand side. As a result, the total area cannot be found, even if the stable areas connect seamlessly.

Solving the RELIABLE problem with the large 14-parameter classifier requires significantly longer runtimes than ADJUST. As a result, a model with a subset of only 250 decision trees still had a high optimality gap after 4 hours of computation, where we denote by *optimality gap* the relative difference between the best solution s found and the best objective bound b obtained from linear relaxations within the branch-and-bound tree: $|b - s|/|s|$.



(a) Reliable: This figure shows the "most stable" point $(k_{AC}, k_{PV})^\top = (1.31, 0.89)^\top$ and its surrounding green circle with normalised radius $r = 0.36$. Starting with a random-forest classifier with 2,000 trees, the solution involves 1,001 active trees.

(b) Volume: The resulting largest possible stable surface is shaded in green. The area is composed of the intersection of 1,001 out of 2,000 decision trees. Consequently, these active decision trees predict it to be stable with (almost) 100% each.

Figure 7: Solution for the tasks Reliable (Figure 7a) and Volume (Figure 7b) for the two-parameter classifier. In both tasks, the optimal solution is independent of the starting point. White and black dashed lines shows the limitations of the active paths of the active trees in the optimal solution.

RF classifier / no. of trees	No. of MIP variables continuous	binary	Computation time (s)	Optimality gap (%)	Objective value / normed radius
10	152	48,885	31	0.0	0.081
100	242	487,230	2,479	0.0	0.075
250	392	1,215,180	–	985.0	0.025

Table 4: Results for Reliable with a maximal solution time of 4 hours.

The long runtime of this problem results from the high number of binary variables and constraints added to the model with each additional large decision tree. For each node in the path of a decision tree, one binary variable $z_{t,v}$, the decision rule on the node associated with that path (Constraint (20c) or (20d)), the relationship between the two binary variables of the node (Constraint (8) or (11)) as well as the inequalities needed for the connected path (Constraint (9) or (10)) are added to the MIP. For a feasible solution, only one of these paths per decision tree is activated by the associated binary variables, rendering many of the inequalities non-binding. To reduce the number of variables and constraints in the MIP model, one can try to identify as many paths as possible that will not be activated in the optimal solution already before optimisation. We again use an iterative approach to do this, taking advantage of the fact that the non-feasibility of an optimisation model can usually be decided very quickly in practice. Let r be the normalised radius of the feasible sphere. We need to consider only those paths whose decision rules in the nodes together form a polyhedron which is large enough to accommodate a sphere with a normalised radius of size r. To decide whether the feasible polyhedron associated with a given path can enclose such a sphere, we compare whether the distance in each parameter between the lower and upper bounds given by the decision rules in its nodes is at least as large as twice the radius r of the sphere. When multiple paths are considered at

once, the intersection of their associated feasible polyhedra needs to enclose the sphere. Now, in the first step of the iterative approach, we start with the largest possible sphere and thus with only a few paths that together yield a polyhedron which is large enough to accommodate it. With each iteration, we decrease the size of the sphere, so more and more paths and the feasible areas they describe can be taken into account. This is repeated until finally we consider enough paths to find a feasible region. To this end, the tree probabilities resulting exclusively from the active subset of the considered paths must sum to at least half the number of the decision trees such that Constraints (13) and (14) can be satisfied. More precisely, we introduce a lower bound $\underline{r}_k \in [0, 0.5]$ on the radius $r \in [0, 0.5]$ as the objective of the current model. In each step k, we reduce this lower bound by a fixed amount as long as the MIP remains infeasible. Starting with the highest possible lower bound, $\underline{r}_0 = 0.5$, we subtract a fixed value, here 0.05, from the lower bound in each iteration. In addition to the two constraints $r \geq \underline{r}_k$ and $\underline{r}_k = 0.5 - 0.05k$ in the k-th MIP model, we consider only those variables and constraints that belong to paths of the decision trees that form a polyhedron which is large enough to accommodate a normalised sphere with a normalised radius of size $\underline{r}_k$. If the k-th MIP is infeasible, we continue with the $k+1$-st problem; otherwise, we stop and return the solution found. It is then also the solution to the overall problem. By placing a lower bound on the normalised radius, we guarantee that the ignored paths in the optimal solution would have had to be deactivated anyway, since they do not yields a polyhedron which is large enough to accommodate a normalised sphere of radius r^{opt}.

The benefit of the iterative optimisation procedure can be seen in Table 5. We are now able to solve all instances optimally, including the largest one with 1,000 trees, in just under an hour each. In particular, we also achieved much shorter solution times for the smaller instances.

RF classifier / number of trees	Number of MIP variables		Computation		Objective value / normed radius
	continuous	binary	time (s)	gap (%)	
10	152	48,885	5	0.0	0.081
100	242	487,230	64	0.0	0.075
250	392	1,215,180	170	0.0	0.078
500	642	2,427,975	3,647	0.0	0.075
1,000	1,142	4,855,955	1,109	0.0	0.076

Table 5: Results for improved RELIABLE procedure

6.1.3 Finding the Largest Stable Cuboid VOLUME

Finally, we would like to find the largest possible region that contains only stable network settings w.r.t. the random-forest classifier. We call this optimisation problem VOLUME. We take advantage of the point that only univariate decision trees are currently considered, and thus only axis-parallel cuboids can be regarded as a possible solution. We find the largest cuboid using the slightly adapted MIP model (24). As in the previous optimisation problem in Section 6.1.2, the safety values r_p chosen individually per parameter must be normalised to $\bar{r}_p$ in the objective function, such that each feature has equal weight. To describe the relationship between these parameters, we again use linear scaling.

The solution for the two-dimensional problem is shown in Figure 7b. The largest possible rectangular area found is shaded in green. All points in the interior are thus predicted to be stable by the classifier. The centre of the rectangle is located such that it is as far away as possible from the active decision rules of the active decision trees, which are shown as dashed white lines. The safety margin in k_{AC} can increase independently of that in k_{PV}, which is particularly interesting when parameters of different importance in the classifier are considered.

In Figure 8, we see the frequency distribution of the 14-dimensional classifier, which results

from the number of times each feature occurs across all decision nodes of the decision trees. There are obviously more and less important parameters, which is also observable in Figure 9.

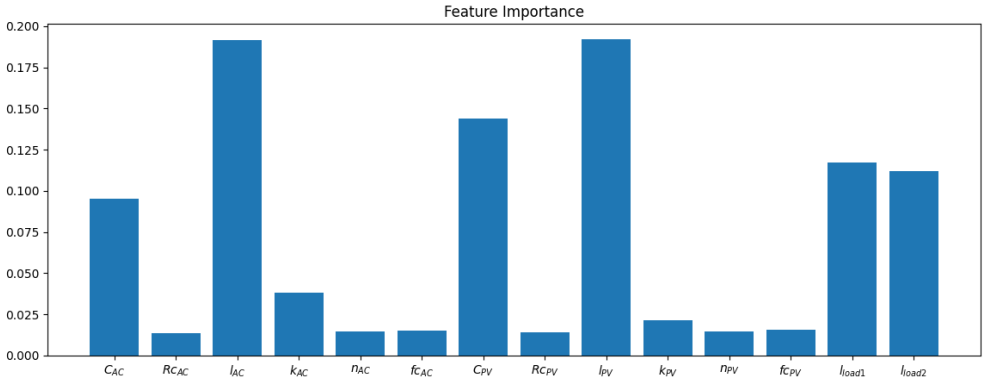


Figure 8: Distribution of parameter/feature importance for the 14-parameter classifier.

It shows the cuboid found for the 14-parameter classifier in sectional planes. The colours refer to starting from the centre of the cuboid. Within the limits of the cuboid found, all parameter combinations can be chosen without losing stability according to the classifier. The cuboid is of different size in each feature. For the rather important features l_{load1} and l_{load2}, the cuboid is very small, while for all features with low importance almost the whole area is green.

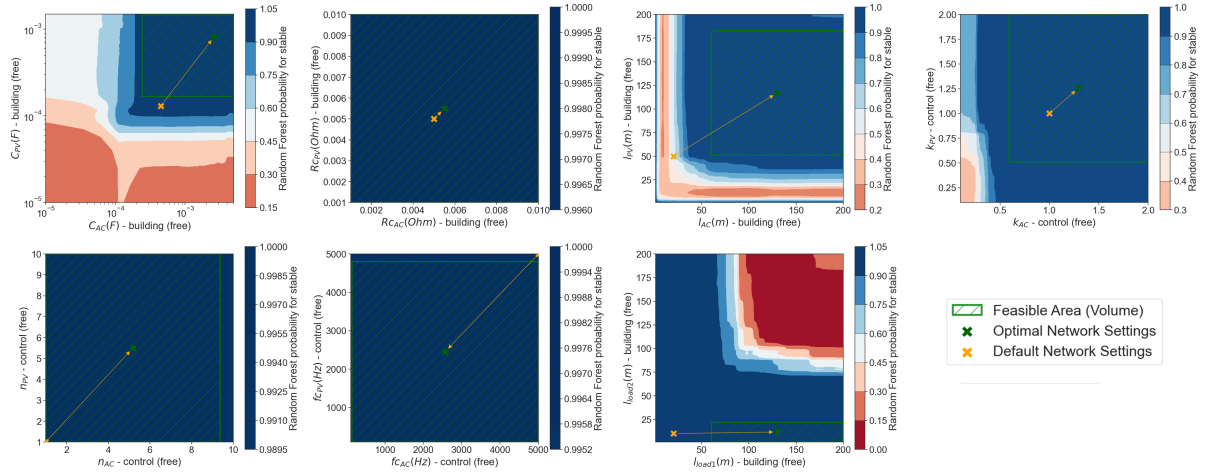


Figure 9: In these plots, we demonstrate the solution to problem Volume we found using the 14-parameter classifier. The largest stable area found by the MIP solver, limited by 501 trees, is green striped. The normalised volume has a size of 0.015, which is 1.5% of the space. Within this area, there are 475 data points, of which 474 belong to a stable network setting, which is 1.36% of the training data. The area is thus relatively well represented. The artificial selection of data points on a grid has an important role in this.

After the problem considered in Section 6.1.2 already required a very long optimisation time, it is not surprising that the runtime is also a problem for Volume. Only up to 100 trees can be solved using the MIP model within 4 hours, as shown in Table 6.

In order to solve the problem also for the random-forest classifier with 1000 decision trees, we can again follow a step-by-step approach. Because of the non-linear objective function, we need two iterative procedures.

Analogously as in the previous problem, we want to consider only variables and constraints from a subset of the possible paths in an iterative approach to save computation time. For each

RF classifier / number of trees	MIP variables		Computation		Objective value / normed volume
	continuous	binary	time (s)	gap (%)	
10	264	49,011	1,025	0.0	0.03
100	354	487,356	–	85.0	0.018

Table 6: Results for VOLUME with a maximal solution time of 4 hours.

path, we first determine the normalised volume of the axis-parallel cuboids resulting from the decision rules in the nodes. To this end, we consider all decision rules for each path and determine the largest lower bound and the smallest upper bound on each parameter. The bounds of a constraint $k_i x_i \leq b$ belonging to the parameter $i \in \{1, \dots p\}$ are obtained by dividing the right-hand side b by the multiplier k_i. This is done equivalently for the constraint $k_i x_k \geq b$. The distance between the upper and lower bound of a parameter is the length of the cuboid in that parameter. For calculating the normalised volume, we have to normalise this length to the interval $[0,1]$ at first. The normalised volume of the axis-parallel cuboid is then obtained by multiplying the normalised lengths.

We start the iterative approach with the largest possible minimum normalised volume of $W_0 \coloneqq 1$. Only variables and constraints of paths whose decision rules form an axis-parallel cuboid with at least this normalised volume are considered. If we cannot form a feasible region with the current paths, the current MIP model is infeasible, and we decrease the volume and consider more paths. Specifically, we halve the minimum normalised volume $W_k \coloneqq W_{k-1}/2$ in each iteration $k \in \mathbb{N}$. As soon as we have found a feasible region (an axis-parallel sectional cuboid) in some iteration k^*, we can calculate its normalised volume $V_{k^*}^{\text{opt}}$. Now we have achieved a first goal: we know the lower bound $V_{k^*}^{\text{opt}}$ and the upper bound W_{k^*} for the optimal axis-parallel sectional cuboid for the global problem.

Between the minimum normalised volume W_{k^*}, which constrains the set of paths considered so far, and the normalised volume $V_{k^*}^{\text{opt}}$, the optimal axis-parallel sectional cuboid found with these paths lies a (possibly very large) distance $W_{k^*} \gg V_{k^*}^{\text{opt}}$. The largest possible axis-parallel sectional cuboid has a normalised volume of V^{opt} lying between this two values. Paths that form axis-parallel cuboids with their decision rules whose normalised volumes are smaller than $V_{k^*}^{\text{opt}}$ can thus safely be ignored. The additional consideration of paths whose decision rules form an axis-parallel cuboid with a volume of at least $V_{k^*}^{\text{opt}}$ can lead to a sectional cuboid with a larger normalised volume than $V_{k^*}^{\text{opt}}$.

Because the distance between $V_{k^*}^{\text{opt}}$ and W_{k^*} is possibly very large, we proceed with another, nested iterative procedure. We start again with the largest possible minimum normalised volume of the sectional cuboid $\tilde{W}_0 \coloneqq V_{k^*}$. Only variables and constraints of paths whose decision rules form an axis-parallel cuboid with at least this normalised volume are considered in the MIP model. In this iterative process, we will always find a feasible optimal sectional cuboid. After optimisation, we can calculate its normalised volume $\tilde{V}_{\tilde{k}}^{\text{opt}}$. As long as the normalised volume of the found optimal sectional cuboid $\tilde{V}_{\tilde{k}}^{\text{opt}}$ is smaller than the lower bound $\tilde{W}_{\tilde{k}}$ on the axis-parallel cuboids of the considered paths, we will reduce it to $\tilde{W}_{\tilde{k}} = \tilde{W}_{\tilde{k}-1}$ and thus iteratively consider more paths. As soon as the volume of the optimal cuboid $\tilde{V}_{\tilde{k}^*}^{\text{opt}}$ is at least as large as the lower bound $\tilde{W}_{\tilde{k}^*}$ on the axis-parallel cuboids of the considered paths for the first time, we can stop the iterative process. The optimal volume of the global problem is that of the found intersection cuboid: $V^{\text{opt}} = \tilde{V}_{\tilde{k}^*}^{\text{opt}}$. This is because adding more paths whose decision rules form an axis-parallel sectional cuboid whose normalised volume is smaller than the found normalised volume $\tilde{V}_{\tilde{k}^*}^{\text{opt}}$ cannot lead to a larger sectional cuboid.

Using this iterative method, we were able to find a high-quality, though not optimal solution for the optimisation problem corresponding to the large 14-parameter classifier within 4 hours. The computational results can be seen in Table 7.

After 4 hours, we have reached the following intermediate state for the problem with 1000

decision trees: using paths whose decision rules have an axis-parallel cuboid with a volume of at least $\tilde{W}_{\tilde{k}} = 0.02$, we find an optimal sectional cuboid whose normalised volume is $\tilde{V}_{\tilde{k}}^{\text{opt}} = 0.014$. The best solution found is already very close to the optimal solution.

RF classifier / number of trees	MIP variables		Computation time	Solution	
	continuous	binary	time (s)	last path lower bound $\tilde{W}_{\tilde{k}}$	objective volume $\tilde{V}_{\tilde{k}}^{\text{opt}}$
10	264	49,011	5	0.03	0.03
100	354	487,356	180	0.019	0.019
500	754	2,428,101	14400	0.02	0.015
1,000	1,254	4,856,081	14400	0.02	0.014

Table 7: Improved results for VOLUME with a maximal solution time of 4 hours.

6.2 Discussion of the Optimisation Models and Practical Implications

Using various mixed-integer models, we developed three strategies to find best possible solutions to random-forest-based classification problems at the example of LVDC network stability.

In the solution strategy ADJUST, we provide a minimal adjustment of parameters to find the optimum solution. It is advantageous that for model ADJUST we can determine solutions very quickly. For the application, ADJUST enables a fast assessment of the stability situation in a given network. If the starting point is unstable, the extended parameter adjustments to reach a stable region can be estimated. Depending on the parametrisation of the optimisation, i.e. which parameters are included, adjustments in different parameters, e.g. in software parameters k_{AC}, k_{PV} may be compared to results where device parameters, e.g. c_{AC}, c_{PV} are changed. When the starting point belongs to a stable state, the visualisation provides a distance estimate to the borders, where the network reaches an unstable state (see Figure 6). As a potential drawback, the obtained solutions may be "at the border" of feasibility, such that perturbations in the data can potentially lead to infeasibilities.

The RELIABLE model initially requires more computation time than ADJUST. Using the iterative procedure, we were able to match or outperform the times for the simple ADJUST problem. For our instances, it could always produce a solution that is well contained inside a stable parameter region. As a result, solutions remain stable even under small input uncertainties. For the practical application, the strategy RELIABLE enables a reliable selection of the input parameters to obtain a stable network state. From the radius of the optimal sphere, the tolerance of the parameters to reach unstable states can be estimated. As possible future improvements, it might be worth integrating a way to rank the features. Furthermore, it is certainly interesting in the application to take certain parameters, such as the cable length, out of the optimisation and assign them fixed realistic values. Depending on these, the other parameters can be chosen reliably enough with the help of the objective function RELIABLE.

A most reliable stable point is one that has the largest possible distance to all inequalities. Parameters with high feature importance typically constrain the feasible set more often and severely. Therefore, the position of the most reliable point usually depends on these parameters. Solutions with the same level of certainty in the parameters with high feature importance and a more reliable position for features with low feature importance could be given special consideration.

In contrast to RELIABLE, problem VOLUME determines large stable regions within which the network is stable. Here, the intervals of the parameters are considered individually and independently of their feature importance. However, due to the initially non-linear objective function and its linearisation, the optimisation model typically requires significantly longer computation times. If a large number of random forests is to be considered, a stepwise approximation

to the optimal solution is required. For the practical application, the strategy Volume is the most important setting for identifying reliable, stable regions. However, care needs to be taken in the generation of random-forest classifiers, since areas may differ in the density of calculated representations, especially when linear and logarithmic scaling points are overlaid. Thanks to the training data that has been generated to represent realistic problem settings, we were able to find a representative cuboid in our solution. Starting from a LVDC circuit model, where typically the device parameters are not fixed, e.g. when identifying a potential dimensioning of the grid parameters, and are varied within several orders of magnitude, e.g. as done for the parameters c_{AC}, c_{PV}, reliable, stable areas can be determined. To avoid unequal point densities as described above, the parameter ranges may be reduced until accurate random-forest classifiers can be obtained with linear scaling of the input parameters where the point density is nearly equivalent, and the largest stable region may be more precisely determined for the refined network setting.

In order to keep the model manageable, it is important to determine a relevant set of decision trees with limited size. Random-forest classifiers can be trained very easily and quickly with the help of software frameworks, even with many trees. However, above a certain threshold, increasing the number of trees does not improve the accuracy further. The Grid_Search method of *scikit-learn* can be used to decide how many decision trees are needed for this purpose so that the effort can be estimated beforehand. For the network stability application, we have shown successfully that the optimisation models can still be solved globally even with large classifier. We have simplified the optimisation problem to an iterative solution of feasibility problems, as described for Adjust. Furthermore, we have strongly reduced the number of active inequalities in Reliable and Volume.

The work presented here concludes an initial study towards a new approach for optimizing stability in LVDC microgrids. The challenge here originates from the complexity of stability assessment including the selection of the stability criteria, their parametrisation and the capability for automated evaluation. Frequently, two parameters are evaluated independently for stability assessment as depicted in the presented examples – leading to a classification of the network as sufficiently stable or not. The new optimization approach presented here enables, for example, the design and parametrisation of new networks or the operation of existing networks targeting large stable operation regions or minimised adjustments in the latter case. In an ongoing experimental work to further validate the approach, an experimental setup was established, which mimics the network structure as depicted in Figure 4. The modular structure of the approach allows for flexible modification of the input parameter settings in the digital twin. For the experimental setup, the individual parameters of the source components need to be characterized. Specifically, the capacitances and line inductances rather than the line lengths were measured to serve as input parameters to the digital twin and subsequent surrogate modelling and optimisation. The runtime of the optimisation and the performance of the optimisation result will be further tested on the experimental setup. Concerning the computational effort, it is anticipated that for network design the computation time does not impose a restriction such that the approaches outlined here can be applied. Furthermore, in an existing setup less parameters need to be varied as several parameters are fixed, e.g. the capacitances or transmission line inductances, such that merely the factors k_{AC} and k_{PV} for changing the droop decline or the cut-off frequencies f_{CAC} and f_{CPV} of the output filter at the source converters need to be changed. Thus, the case study performed here is very relevant for our future work in network stability.

7 Conclusions

In this work, we have presented a practical optimization approach for integrating model- and data-based methods and showed its applicability in practice. For this purpose, we used a com-

bined MIP- and machine-learning-based approach. Inequalities that can easily be formulated as linear mixed-integer inequalities are explicitly modelled in the optimisation problem. To replace complex or unknown constraints, we performed two steps. First, using a representative data set, we trained a classifier, in our case a random-forest classifier, because it has several advantages to our considered settings. Then we modelled the decisions algebraically and included them as additional constraints in the MIP model.

In our case study on the stability of DC networks, we showed that this approach leads to high-quality results in practical problems. The task was to find stable network settings for a DC grid with system voltages below 1,500 V (low-voltage DC grids, LVDC grids) so that the safety and reliability of the model can be guaranteed. For this approach, a large dataset was used to train a random-forest classifier for which the resulting classifications were included in algebraic form in the MIP. As objective functions, we considered different relevant objectives, namely the closest and the most reliable stable network setting. Furthermore, we determined the largest possible stable area from the random-forest classifier.

Although modern MIP solvers can optimise large instances, the number of decision trees naturally plays a significant role in the performance of solving these optimisation problems. In this work, we have shown how the solution of the MIPs can be enhanced by sequential execution of slightly adapted MIP models that are at first strongly constrained and thus typically infeasible. Then the constraints are iteratively relaxed until the appropriate model is found from which the solution can be determined. This approach significantly reduces the runtime, because testing the infeasibility of a model is typically fast. As future work, it is interesting to analyse whether the runtime can be improved further, possibly by a better encoding of the random forests.

References

Aigner, K.-M., Burlacu, R., Liers, F., and Martin, A. (2021a). Solving AC optimal power flow with discrete decisions to global optimality. Preprint.

Aigner, K.-M., Clarner, J.-P., Liers, F., and Martin, A. (2021b). Robust approximation of chance constrained DC optimal power flow under decision-dependent uncertainty. *European Journal on Operations Research*, 301(1):318–333.

Azaioud, H., Claeys, R., Knockaert, J., Vandevelde, L., and Desmet, J. (2021). A low-voltage DC backbone with aggregated res and bess: Benefits compared to a traditional low-voltage AC system. *Energies*, 14(5).

Bertsimas, D. and Dunn, J. (2017). Optimal classification trees. *Machine Learning*, 106(7):1039–1082.

Bertsimas, D., Dunn, J., and Mundru, N. (2019). Optimal prescriptive trees. *INFORMS Journal on Optimization*, 1(2):164–183.

Bertsimas, D., O'Hair, A., Relyea, S., and Silberholz, J. (2016). An analytics approach to designing combination chemotherapy regimens for cancer. *Management Science*, 62(5):1511–1531.

Bhavsar, H. and Ganatra, A. (2012). A comparative study of training algorithms for supervised machine learning. *International Journal of Soft Computing and Engineering (IJSCE)*, 2:74–81.

Biau, G., Devroye, L., and Lugosi, G. (2008). Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9:2015–2033.

Biggs, M., Hariss, R., and Perakis, G. (2017). Optimizing objective functions determined from random forests. *SSRN Electronic Journal*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, Berlin, Heidelberg.

Bonfietti, A., Lombardi, M., and Milano, M. (2015). Embedding decision trees and random forests in constraint programming. In Michel, L., editor, *Integration of AI and OR Techniques in Constraint Programming*, pages 74–90, Cham. Springer International Publishing.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Carpentier, J. (1962). Contribution a l'etude du dispatching economique. *Bulletin de la Societe Francaise des Electriciens*, 3(1):431–447.

Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., Laird, C. D., and Misener, R. (2022). OMLT: Optimization & Machine Learning Toolkit.

Edward A. Bender, S. G. W. (2010). *Lists, Decisions and Graphs*. University of California at San Diego.

Ferreira, K., Lee, B., and Simchi-levi, D. (2015). Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, 18(1):69–88.

Gao, F., Kang, R., Cao, J., and Yang, T. (2019). Primary and secondary control in DC microgrids: a review. *Journal of Modern Power Systems and Clean Energy*, 7(2):227–242.

Gurobi Optimization, LLC (2020). Gurobi optimizer reference manual. `https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.0/refman.pdf`. (Accessed on 14.10.2020).

Halilbašić, L., Thams, F., Venzke, A., Chatzivasileiadis, S., and Pinson, P. (2018). Data-driven security-constrained AC-OPF for operations and markets. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning - Data Mining, Inference, and Prediction, Second Edition*. Springer Science & Business Media, Berlin Heidelberg.

International Electrotechnical Commission, editor (2016). *Protection against electric shock – Common aspects for installations and equipment*, volume IEC 61140:2016. VDE, Geneva, Switzerland, 4 edition.

Kumar, A., Serra, T., and Ramalingam, S. (2019). Equivalent and approximate transformations of deep neural networks. *CoRR*, abs/1905.11428.

Kumar, A. L., Indragandhi, V., and Maheswari, U. Y. (2020). *Software Tools for the Simulation of Electrical Systems*. Academic Press, San Diego, CA, USA, 1 edition.

Lin, C. D. and Tang, B. (2015). *Latin Hypercubes and Space-Filling Designs*, chapter 17, pages 593–625. CRC Press, Boca Taton, Florida, UNITED STATES.

Maragno, D., Wiberg, H. M., Bertsimas, D., Birbil, S. I., den Hertog, D., and Fajemisin, A. O. (2021). Mixed-integer optimization with constraint learning. *CoRR*, abs/2111.04469.

Markowitz, H. M. and Manne, A. S. (1957). On the solution of discrete programming problems. *Econometrica*, 25(1):84.

Mary, A., Cain, B., and O'Neill, R. (2012). History of optimal power flow and formulations. *Federal Energy Regulatory Commission*, 1:1–36.

Mistry, M., Letsios, D., Krennrich, G., Lee, R. M., and Misener, R. (2021). Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded. *INFORMS Journal on Computing*, 33(3):1103–1119.

Mistry, M., Letsios, D., Misener, R., Krennrich, G., and Lee, R. (2018). Optimization with gradient-boosted trees and risk control.

Mišić, V. V. (2020). Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624.

Ott, L., Han, Y., Stephani, O., Kaiser, J., Wunder, B., März, M., and Rykov, K. (2015a). Modelling and measuring complex impedances of power electronic converters for stability assessment of low-voltage DC grids. In *2015 IEEE First International Conference on DC Microgrids (ICDCM)*, pages 51–56.

Ott, L., Han, Y., Wunder, B., Kaiser, J., Fersterra, F., Schulz, M., and März, M. (2015b). An advanced voltage droop control concept for grid-tied and autonomous dc microgrids. In *2015 IEEE International Telecommunications Energy Conference (INTELEC)*, pages 1–6.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Riccobono, A. and Santi, E. (2014). Comprehensive review of stability criteria for dc power distribution systems. *IEEE Transactions on Industry Applications*, 50(5):3525–3535.

Roeder, G., Ott, L., Meier, A., Wunder, B., Wienzek, P., Bärmann, A., Liers, F., and Schellenberger, M. (2021). Analysis and improvement of lvdc-grid stability using circuit simulation and machine learning - a case study. *NEIS 2021; Conference on Sustainable Energy Supply and Energy Storage Systems*, pages 1–7.

Scikit-Learn (n.d.). Decision trees — scikit-learn 0.24.2 documentation. `https://scikit-learn.org/stable/modules/tree.html`. (Accessed on 08/03/2021).

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Thebelt, A., Kronqvist, J., Lee, R. M., Sudermann-Merx, N., and Misener, R. (2020). Global optimization with ensemble machine learning models. In Pierucci, S., Manenti, F., Bozzano, G. L., and Manca, D., editors, *30th European Symposium on Computer Aided Process Engineering*, volume 48 of *Computer Aided Chemical Engineering*, pages 1981–1986. Elsevier.

Thebelt, A., Kronqvist, J., Mistry, M., Lee, R. M., Sudermann-Merx, N., and Misener, R. (2021). ENTMOOT: A framework for optimization over ensemble tree models. *Computers & Chemical Engineering*, 151:107343.

Thebelt, A., Tsay, C., Lee, R. M., Sudermann-Merx, N., Walz, D., Tranter, T., and Misener, R. (2022). Multi-objective constrained optimization for energy applications via tree ensembles. *Applied Energy*, 306:118061.

Thorbjarnarson, T. and Yorke-Smith, N. (2020). On training neural networks with mixed integer programming. *CoRR*, abs/2009.03825.

Vielma, J. P. (2015). Mixed integer linear programming formulation techniques. *SIAM Review*, 57(1):3–57.

Weiss, R., Ott, L., and Boeke, U. (2015). Energy efficient low-voltage DC-grids for commercial buildings. In *2015 IEEE First International Conference on DC Microgrids (ICDCM)*, pages 154–158.

Wunder, B., Ott, L., Han, Y., Kaiser, J., and Maerz, M. (2015). Voltage control and stabilization of distributed and centralized dc micro grids. In *Proceedings of PCIM Europe 2015; International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*, pages 1–8.

Åström, K. J. and Murray, R. M. (2008). *Feedback Systems - An Introduction for Scientists and Engineers*, chapter 9, pages 278–282. Princeton University Press, New York, United States.

A List of Network Parameters

Element	Parameter				
	Name	Lower	Upper	Unit	Description
AC-DC conversion (source 1) and PV system (source 2)	C_{AC}	$1E-5$	$5E-3$	F	Output capacitance of the source converters
	C_{PV}	$1E-5$	$1.5E-3$		
	R_{CAC}	$1E-3$	$1E-2$	Ω	Equivalent series resistance of the output capacitance of the source converters
	R_{CPV}				
	k_{AC}	$1E-1$	$2E+0$		Factor for changing the droop decline at the source converters
	k_{PV}				
	f_{CAC}	$1E+2$	$5E+3$	Hz	Cut-off frequency of the output filter at the DC-DC converter
	f_{CPV}				
	n_{AC}	$1E+0$	$1E+1$		Multiplication factor of the sampling time for dead time of the DC-DC converter regulation.
	n_{PV}				
	l_{AC}	$1E+0$	$2E+2$	m	Wire length from AC source or PV system to bus node
	l_{PV}				
Load groups	l_{load1}	$1E+0$	$2E+2$	m	Wire length from the load groups to bus node.
	l_{load2}				
	P_{load1}	$0E+0$	$1E+4$	W	Load value 1 and load value 2.
	P_{load2}				

Table 8: DC grid input parameters for stability analysis.

Acknowledgements

We thank the DFG for their support within Projects B06 and Z01 in CRC TRR 154. Furthermore, we acknowledge financial support by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics – Data – Applications (ADA-Center) within the framework of "BAYERN DIGITAL II" (20-3410-2-9-8).