Applied Soft Computing 68 (2018) 1–12

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

One-class synthesis of constraints for Mixed-Integer Linear Programming with C4.5 decision trees

Patryk Kudła, Tomasz P. Pawlak*

Institute of Computing Science, Poznan University of Technology, Poznań, Poland

ARTICLE INFO

Article history:
Received 15 September 2017
Received in revised form 15 February 2018
Accepted 15 March 2018
Available online 23 March 2018

Keywords:
Model acquisition
Constraint synthesis
Mathematical programming
One-class classification
Business process

ABSTRACT

We propose Constraint Synthesis with C4.5 (CSC4.5), a novel method for automated construction of constraints for Mixed-Integer Linear Programming (MILP) models from data. Given a sample of feasible states of a modeled entity, e.g., a business process or a system, CSC4.5 synthesizes a well-formed MILP model of that entity, suitable for simulation and optimization using an off-the-shelf solver. CSC4.5 operates by estimating the distribution of the feasible states, bounding that distribution with C4.5 decision tree and transforming that tree into a MILP model. We verify CSC4.5 experimentally using parameterized synthetic benchmarks, and conclude considerable fidelity of the synthesized constraints to the actual constraints in the benchmarks. Next, we apply CSC4.5 to synthesize from past observations two MILP models of a real-world business process of wine production, optimize the MILP models using an external solver and validate the optimal solutions with use of a competing modeling method.

1. Introduction

Optimization of business processes and systems is an essential part of every business, as reducing consumed resources or increasing volume of production may bring substantial advantage in a competitive business environment. To optimize a business process, one needs to build a mathematical model for it and optimize this model instead of manipulating directly the parameters of the process. This allows the process to operate uninterruptedly until the optimal parameters are found. For instance, a clothing company may reduce fabric loss by rearranging pieces of fabric to cut from a fabric roll such that the loss is minimal. To do this, one needs to build a model resembling shapes of pieces to cut and technical restrictions of the cutting machine, and then optimize this model using an off-the-shelf solver. The optimal cutting pattern may be applied during e.g., maintenance break.

In this study, we employ Mixed-Integer Linear Programming (MILP) [1] models. A well-formed MILP model consists of linear objective function, set of linear constraints and domains of variables thereof. Linearity facilitates interpretation of the objective function and the constraints while being enough expressive to model many real-world processes. The constraints in a MILP model essentially form a conjunction, however thanks to the support of binary variables, an alternative of constraints can be formed using the auxiliary binary variables. Integer (binary) variables cause optimization of a MILP model to be NP-hard [2], however modern solvers can efficiently handle hundreds of them.

MILP models are often built manually by an expert, requiring thus from she competencies in the modeled process and techniques of modeling. Combining these competencies is uncommon in practice. Also, reflecting non-linear real-world relationships using linear equations requires advanced modeling techniques. All these issues cause model building to be error-prone, laborious and expensive task. In contrast, optimization of a MILP model is fully automated by the solvers.

An appealing alternative to manual model building is synthesis of a model from data. The model synthesis problem can be conveniently decomposed into the synthesis of an objective function and the synthesis of constraints. The former can be done using e.g., least-squares regression which is beyond our interest, and the latter is investigated in this study.

The main contribution of this study is *Constraint Synthesis with C4.5* (CSC4.5), a novel method for construction of MILP constraints from examples of snapshots of feasible states of the modeled process. The feasible states represent normal operating conditions of the process and can be conveniently acquired by observing process execution. Observation can sometimes spot an infeasible state too – an error, a fault or other undesirable conditions. The infeasible states, however, are avoided in practice and thus uncommon. This

* Corresponding author.
E-mail address: tpawlak@cs.put.poznan.pl (T.P. Pawlak).
URL: http://www.cs.put.poznan.pl/tpawlak/ (T.P. Pawlak).

https://doi.org/10.1016/j.asoc.2018.03.025
1568-4946/

is the reason, why we assume their absence and consider *one-class* examples only (like one-class classification in Machine Learning [3]). CSC4.5 copes with one-class examples by estimating the distribution of the feasible states using Expectation Maximization [4] and sampling this distribution for artificial likely infeasible states that act as the second class examples. A training set made of the feasible and the likely infeasible examples is fed to the C4.5 algorithm [5] for decision tree learning. Finally, the resultant decision tree is transformed into a well-formed MILP model. CSC4.5 supports alternative of constraints by automatically introducing auxiliary binary variables. The synthesized MILP model, when supplemented with an objective function, may be fed to a ready-to-use solver for optimization.

We experimentally tune parameters of CSC4.5 using synthetic benchmarks and then evaluate properties of the synthesized models. The general conclusions are promising, however leave room for improvements. Next, we apply CSC4.5 to synthesize a MILP model for a real-world process of wine production and optimize the resulting model using an off-the-shelf solver. The comparison of this model to a model constructed by a naive algorithm based on calculating a convex hull of a training set reveals superiority of CSC4.5 in terms of consumed computational resources and higher estimated quality of the optimal solution.

Section 2 formalizes the constraint synthesis problem, Section 3 compares this work to the past works, and Section 4 describes the details of CSC4.5. Experimental evaluation is conducted in Section 5, and the abovementioned application to wine production process in Section 6. Section 7 discusses the results and the properties of CSC4.5, and Section 8 concludes this work.

2. Problem statement

This section formalizes the constraint synthesis problem, and Section 4 shows an algorithm solving this problem.

Let $x_1, x_2, \ldots, x_n \in \mathbb{R}$ be input variables, let $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ be an example of values of input variables, let X be a set of examples, and let $b_1, b_2, \ldots, b_m \in \{0, 1\}$ be auxiliary variables, $\boldsymbol{b} = [b_1, b_2, \ldots, b_m]$, i.e., $\boldsymbol{b}$ is not a part of an example. Then, constraint $c(\boldsymbol{x}, \boldsymbol{b})$ is a function in form of $\sum_{i=1}^{n} w_i x_i + \sum_{k=1}^{m} v_k b_k \leq a$, where $w_i, v_k \in \mathbb{R}$ are weights and $a \in \mathbb{R}$ is a constant. The aim of *constraint synthesis problem* is to find a set of constraints C that maximizes the number of examples $\mathbf{x}$ in X such that every constraint c in C is satisfied for $\mathbf{x}$ and some $\mathbf{b}$, i.e., $\left|\{\mathbf{x} \in X : \forall_{c \in C} \exists_{\boldsymbol{b}} c(\boldsymbol{x}, \boldsymbol{b}) = \text{true}\}\right|$.

Albeit not explicitly stated, examples in X correspond to the feasible states of a modeled process, and as such delineate location of the feasible region of C. We call them the *feasible examples* to distinguish from the unlabeled examples that are introduced later in this paper. Therefore, the constraint synthesis problem is a kind of one-class classification problem [3] with the aim to learn the characteristics of the given examples instead of learning the boundary between examples of different classes.

The constraint synthesis problem is ill-posed, as it has indefinitely many solutions, including a degenerated solution $C = \emptyset$, i.e., an empty set of constraints satisfied for every $\boldsymbol{x}$. The choice of a particular solution depends on user's preferences, technical requirements and/or other factors. In this study, from the syntactic perspective, C is preferred such that the individual constraints $c \in C$ form facets of hypercubes in $\mathbb{R}^n$, and the hypercubes correspond to alternative subsets of constraints in C, and the feasible region of C in $\mathbb{R}^n$ is a union of these hypercubes, modeled by assigning distinct values for $\boldsymbol{b}$ for each hypercube. From the semantic perspective, C is expected to maximize the number of true positives in X and minimize the number of false negatives in X. Note that measures based on true negatives and false positives cannot be calculated, as negative examples do not exist in X.

Although the set of constraints C is formally a part of a MILP model, synthesis of other parts of the model is beyond the scope of this study and we use the terms 'set of constraints' and 'model' interchangeably.

3. Related work

The work on constraint synthesis can be classified w.r.t. two axes: the type of the constraints: Linear Programming (LP), Non-Linear Programming (NLP), Constraint Programming (CP), Modulo Theories (MT) and other types, and the type of the synthesis problem: one-class, where only feasible examples are available like in the problem posed in Section 2, and two-class, where feasible and infeasible examples are available.

CSC4.5 introduced in this work belongs to the category of *one-class* synthesis of *LP constraints* and involves state-of-the-art Machine Learning (ML) algorithms: Expectation Maximization (EM) [4] and C4.5 [5]. Therefore, we first review the alternative approaches to constraint synthesis, beginning from the works on LP constraints and then advance to other types. Next, we discuss the ML perspective and pros and cons for possible alternatives to EM and C4.5.

Regarding the synthesis of LP constraints, GenetiCS [6] uses strongly-typed Genetic Programming [7] to tackle two-class synthesis. Given considerably successful results, the same authors develop GOCCS [8], an extension of GenetiCS, supporting one-class synthesis. GOCCS employs an abstract syntax tree (AST) representation that may express LP and NLP constraints. GOCCS is verified on synthetic benchmarks and in modeling of a real-world business process of wine production with good results. GOCCS, however, suffers from the curse of dimensionality [9] and the quality of the synthesized models drops rapidly for over half a dozen variables. Evolutionary-Strategy [10] based One-Class Constraint Synthesis (ESOCCS) [11] synthesizes constraints in the type parameterized within LP and NLP classes. ESOCCS turns out superior to GOCCS when assessed on several synthetic benchmarks. CSC4.5 differs conceptually from GOCCS and ESOCCS in its support for the synthesis of alternative constraints.

The work [12] proposes to encode the two-class synthesis problem of LP-like models using a MILP problem and solve it optimally w.r.t. a custom measure of model complexity. Although this method is exact, it becomes computationally infeasible if the number of variables is over a dozen due to NP-hardness of solving the MILP problem [2]. Also, this method basically features no bias, in terms of the bias-variance dilemma [13], by fitting the constraints to the *training* data such that the separation of all feasible examples from all infeasible examples is guaranteed, even if this leads to excessive variance of errors on the *test* data. In contrast, CSC4.5 uses a softer approach that allows some training examples (e.g., outliers) to remain on the wrong side of the constraints if the constraints reflect general trends in data, likely leading to lower variance of errors on the test data.

The work [14] proposes to handle the one-class synthesis problem by building a convex hull of a set of feasible examples, cluster facets of this hull using k-means [15], and write the facets down as LP constraints. Complexity of this method is exponential in the number of input variables, and as Section 6 shows calculating a convex hull not only lasts longer than running CSC4.5 for the same data, but also results in bloated models.

In [16] ready-to-use implementations of C4.5 decision tree and artificial neural network are trained using two-class data, and then transformed to either Mixed-Integer NLP, CP or MT models. Ref. [16] differs from this work by that Ref. [16] tackles two-class problem, and this work one-class problem, and Ref. [16] uses default param-