One-Class Constraint Acquisition with Local Search

Daniel Sroka
Institute of Computing Science
Poznan University of Technology
Poznań, Poland
dnlsroka@gmail.com

Tomasz P. Pawlak*
Institute of Computing Science
Poznan University of Technology
Poznań, Poland
tpawlak@cs.put.poznan.pl

ABSTRACT

We propose One-Class Constraint Acquisition with Local Search (OCCALS), a novel method for computer-assisted acquisition of Mixed-Integer Linear Programming (MILP) models from examples. OCCALS is designed to help human experts in preparation of MILP models for their systems. It supports building MILP models from the examples of positive class only, requiring thus relatively cheap to acquire training set, e.g., by observing historical execution of a system, and effectively handles multimodal distribution of the training set that may happen in practice. We verify OCCALS experimentally and compare its performance to a state-of-the-art method, concluding superiority of OCCALS.

CCS CONCEPTS

• **Computing methodologies → Learning linear models**; **Modeling methodologies**; • **Applied computing** → *Computer-aided manufacturing*;

KEYWORDS

Model synthesis, Model learning, Linear Programming, Optimization, Classification

ACM Reference Format:
Daniel Sroka and Tomasz P. Pawlak. 2018. One-Class Constraint Acquisition with Local Search. In *Proceedings of the Genetic and Evolutionary Computation Conference 2018 (GECCO'18).* ACM, New York, NY, USA, Article ?, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Optimization is common in business and technology: companies improve efficiency of their business processes by manipulating technical parameters of their industrial systems, scheduling production and delivery, and managing personnel. No matter what is the application, effective optimization requires building a mathematical model of the entity being optimized. A common representation is Linear Programming (LP) model [25] – a formal object consisting of three parts: variables corresponding to the parameters of that entity, a linear objective function expressing the output characteristics of that entity to be optimized, and linear constraints reflecting restrictions on and relationship between the parameters. LP models are easy to inspect by humans thanks to the simple linear form and their optimization is fully automated by the solvers.

However, designing LP models is not free of challenges. Experts sharing the competencies in modeling and the entity being optimized are not common in practice. Also, the linear form of the model may be considered too restrictive in applications in which the actual characteristics are non-linear, hence requiring advanced modeling techniques. Last but not least, humans commit errors, and errors in LP models often remain unnoticed until an optimal solution happens infeasible when applied to a real system.

In this study, we attempt to mitigate these challenges by automating acquisition of LP models from data. This task can be independently solved for the objective function and the constraints. Acquisition of the objective function is a regression problem, to the date having many solutions and as such being beyond our interest. In turn, acquisition of the constraints is a classification problem with the classifier represented as a set of linear inequalities and having a limited number of solutions in the literature (cf. Sec. 2). As we do not distinguish between the 'set of constraints' and 'model' in the following, we use these terms interchangeably.

We assume that the training set consists of examples of values attained by the variables in a sample of states of the entity being modeled. An example represents a feasible state corresponding to regular operation, or an infeasible state corresponding to error or fault. The infeasible states are avoided in practice, and obtaining a representative sample of them is unlikely. For that reason, we assume absence of the examples of the infeasible states and reduce the constraint acquisition problem to a one-class classification problem, aimed at detecting the boundaries of the feasible region [11].

We expect that the distribution of the training examples is multimodal, e.g., when a production line manufactures variety of products, they correspond to different modes. This is apparent incompatibility with LP models, as these are limited to convex feasible regions and bounding a multimodal training set with even a smallest convex superset (i.e., a convex hull) may erroneously include in the feasible region large areas without examples. Avoiding such areas is extremely important from the viewpoint of optimization of an LP model, as an objective function may attain optimum there, while the solutions located there are likely infeasible. The abovementioned deficiency of LP models can be alleviated by extending to Mixed-Integer Linear Programming (MILP) models [25] with auxiliary binary variables that may model alternative of constraints, hence non-convex feasible regions.

The main contribution of this study is One-Class Constraint Acquisition with Local Search (OCCALS) algorithm which given

*Corresponding author.

GECCO'18, July 15–19, 2018, Kyoto, Japan
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

a one-class multimodal training set acquires a non-convex MILP model. Given a unimodal training set, the resulting MILP model reduces to a convex LP model. OCCALS initially partitions the given training set using x-means [22]. For each partition separately, it searches locally the space of LP constraints for an LP model representing that partition. Finally, OCCALS merges the individual LP models into a single MILP model having the interpretation of the alternative of the individual LP models.

We verify OCCALS experimentally in acquiring constraints for a set of benchmarks with multimodal and unimodal training sets, concluding considerable fidelity of the acquired MILP models to the actual MILP models and effective handling of multimodal data. The comparison to a state-of-the-art method concludes superiority of OCCALS.

2 RELATED WORK

The works on acquisition of LP constraints origin in two-class problems, where the aim is to find a set of constraints separating feasible and infeasible examples. Then, those works inspired research on one-class constraint acquisition. Hence, we first review the works on two-class acquisition, then one-class, and next we discuss the works involving different representations of constraints: Non-Linear Programming (NLP), Constraint Programming (CP), Modulo Theories (MT) and probabilistic models.

[19] formalizes the problem of two-class acquisition of LP and NLP constraints as MILP problem and solves it optimally w.r.t. a custom measure of complexity using an off-the-shelf solver. The resulting models are small and easy to inspect but generalize poorly, as the objective is not to optimize any performance measure. GeneticS [20] acquires LP and NLP models using Genetic Programming (GP). The experiment using synthetic benchmarks shows a room for improvement in terms of fidelity and size of the resulting models.

GOCCS [21] is one-class extension of [20] that estimates the distribution of feasible states prior to executing a composition of GP and NSGA-II. GOCCS given a one-class training set achieves consistently better results than GeneticS supplied with a training set consisting up to 33% of infeasible examples. GOCCS was successfully applied to constraint acquisition and optimization of a wine production process. ESOCCS [18] is a composition of Expectation Maximization and co-Evolutionary Strategy that acquires LP and NLP models from one-class training sets. ESOCCS turns out superior to GOCCS in an experimental evaluation in terms of the fidelity of the produced models to actual models, although pays for that in noticeably higher computational cost. [1] constructs a convex hull of a set of feasible examples, clusters its facets to reduce their number, and write down the facets as LP constraints. As calculating convex hull is exponential in time w.r.t. dimensionality, this method is computationally infeasible for over a dozen variables.

Conaq [5] produces CP constraints from two-class training set using Version Space Learning. An extension [6] provides support for interactive expert queries for classification of artificially-generated examples that are highly informative from the view of constraint acquisition. [23] extends Conaq further by incorporating arguments – the pieces of domain knowledge provided by an expert in support of her classifications. Conacq is limited to finite-domain variables and does not support weights nor nonlinear transformations of the variables, hence the resulting CP model is difficult to inspect and NP-complete to solve [17]. QuAcq [4] acquires CP models from examples with missing values of some variables by interactively querying an expert for classification of artificial examples. QuAcq is asymptotically optimal in the number of queries needed to converge for constraints involving = or ≠ comparisons.

Model Seeker [2] produces CP-like models in Prolog from one-class training set and a database of handcrafted and described with metadata templates of constraints. Model Seeker yields promising results on several real-world problems, but is not fully autonomous by requiring that database of background knowledge constraints. [12] is a method of learning first-order logic clauses using Inductive Logic Programming and estimating their weights using Preference Learning. Performance of this method is human-competitive when assessed on a few synthetic benchmarks.

In [14] off-the-shelf implementations of C4.5 and multilayer perceptron are learned on two-class task scheduling problems, then transformed into Mixed-Integer NLP, CP and MT models and optimized using off-the-shelf solvers. This work lacks tuning of the learning algorithms to the problem tackled, and the resulting models are oversize and difficult to inspect.

Learning Modulo Theories (LMT) [24] is a framework for learning MT models from one-class training sets and background knowledge theories. Hence, LMT is not fully autonomous and requires from an expert at least a piece of the actual model.

A related field of study is estimation of hypervolume, meant as a boundary of a given set of examples, typically represented as: a set of inequalities (like in LP and NLP models), a convex hull, or a threshold on probability density function of an empirical probability distribution. The last approach is taken in [7] that employ multivariate Kernel Density Estimation. This approach can be possibly extended to learn Stochastic Programming models, however the authors do not follow this direction of research.

3 ONE-CLASS CONSTRAINT ACQUISITION

We first formalize the problem posed, then present the algorithm solving this problem.

3.1 Constraint Acquisition Problem

Let $x_1, x_2, \ldots, x_n$ be input variables, $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ be a state, X be a set of examples of states, $b_1, b_2, \ldots, b_m \in \{0, 1\}$ be auxiliary variables such that for every $\boldsymbol{x}$ exactly one $b_j = 1$, $\boldsymbol{b} = [b_1, b_2, \ldots, b_m]$. A *constraint* $c(\boldsymbol{x}, \boldsymbol{b})$ is a logic clause in the form $\boldsymbol{w} \cdot \boldsymbol{x} + Lb_j \leq w_0 + L$, where $\boldsymbol{w} = [w_1, w_2, \ldots, w_n] \in \mathbb{R}^n$ is a vector of weights, $w_0 \in \mathbb{R}$ is a free term, $L \gg \|\boldsymbol{w}\|$ is a large constant. A *Constraint Acquisition Problem* (CAP) is to find a set of constraints C^* such that all $c \in C^*$ are met for all $\boldsymbol{x} \in X$, i.e., $\forall_{c \in C^*} \forall_{\boldsymbol{x} \in X} c(\boldsymbol{x}, \boldsymbol{b})$.

CAP is ill-posed by featuring an infinite number of solutions, including a degenerated one $C^* = \emptyset$. The root cause is unknown distribution of the feasible states and the only information on that distribution is the sample X. However, given no prior information on *type* of that distribution, as per No Free Lunch theorem [26] there is no single universally-correct way to estimate that distribution based on X. Consequently, we cannot objectively assess the fitting of that distribution with a set of constraints. However, as the primary purpose of LP models is optimization, we can formulate a rule of thumb that *the constraints should tightly but with a small margin wrap* X. This is because optimal solutions of LP models lie on the constraints and loosening the constraints increases the risk of an optimal solution being infeasible in practice. Also, very tight constraints like the facets of convex hull $co(X)$ would be too restrictive, as every vertex of $co(X)$ is actually an example from X and for all linear objective functions f no solution better than the best actually known in X belongs to $co(X)$, i.e., $\max_{\boldsymbol{x} \in X} f(\boldsymbol{x}) = \max_{\boldsymbol{x} \in co(X)} f(\boldsymbol{x})$. The algorithm presented in the next section is designed to follow that rule of thumb.

3.2 Constraint Acquisition Algorithm

One-Class Constraint Acquisition with Local Search (OCCALS) is shown in Algorithm 1. In line 2, it partitions the given training set X into $k \geq k_{min}$ partitions using x-means [22], where k is the detected number of partitions, and $k_{min} \geq 1$ is a parameter. Then, independently for each partition X_j OCCALS produces in lines 4–9 a set of LP constraints forming a closed feasible region that includes X_j. To this aim, OCCALS centers X_j and stores the result in X'_j in line 5. In line 6, it forms an initial set of constraints C_j as a bounding box of X'_j and in line 7 runs local search for c_{max} constraints wrapping X'_j more tightly, where c_{max} is a parameter. In line 8, it removes from C_j all redundant constraints that may occur in the combination of the bounding box constraints and the constraints found using local search. In line 9, OCCALS translates the constraints in C_j produced for the centered set X'_j by the vector of expected values $E(X_j)$ to obtain constraints for X_j. At this point, C_j consists of a complete set of LP constraints for X_j. Finally, OCCALS forms MILP model C as a union of the constraints in C_j extended with auxiliary binary variables b_j (line 10) and an extra constraint requiring all constraints from at least one C_j to hold (line 11).

In the following subsections, we discuss particular steps in detail.

3.2.1 *X-means partitioning of training set.*

For the reasons pointed out in Introduction, a multimodal training set X may not be accurately bounded using a convex feasible region corresponding to a set of LP constraints. To handle possible multimodality of X, OCCALS partitions X into disjoint sets $X_1, X_2, \ldots, X_k$, $k \geq k_{min}$, corresponding to individual Gaussians and thus each having unimodal distribution. In particular, if X has less than k_{min} modes, then $k = k_{min}$. The unimodal X_j are to be bounded using independent sets of LP constraints.

OCCALS uses x-means partitioning [22] shown in line 12 of Algorithm 1. It extends k-means [9] with automatic detection of the number of partitions in the given set X, subject to the lower bound k. It initializes an empty candidate partitioning $P = \emptyset$ in line 15. Then, it calls k-means in line 16 to partition X into $P_k = \{X_1, \ldots, X_k\}$. For each partition $X_j \in P_k$, XMEANS attempts to partition it further into X'_j and X''_j by calling k-means again with $k = 2$ in line 18. Whichever of X_j and $\{X'_j, X''_j\}$ has higher value of Bayesian Information Criterion (BIC) is included in P (lines 19–20). We employ BIC definition from [22], omitted here for brevity. Partitioning P replaces the best-so-far partitioning P_{best} in line 21 if the former has higher value of BIC. These steps are repeated with

Algorithm 1 OCCALS algorithm; k_{min} is minimal number of alternative LP models, c_{max} is maximum number of constraints searched for each LP model. KMEANS is k-means partitioning [9], *BIC* is Bayesian Information Criterion, $\mathcal{N}(\mu, \sigma^2)$ is Gaussian with mean μ and variance σ^2, $\mathcal{U}(A)$ is uniform distribution over set A, $\mathbf{1}(a)$ is indicator function attaining 1 if a is true, 0 otherwise, $\mathbb{D}_i$ is domain of variable x_i; red terms are new in constraints in the corresponding lines.

```
1:  function OCCALS($X, k_{min}, c_{max}$)
2:      $X_1, X_2, \ldots, X_k \leftarrow$ XMEANS($X, k_{min}$)
3:      $C \leftarrow \emptyset$
4:      for $j = 1..k$ do
5:          $X'_j \leftarrow \{\boldsymbol{x} - E(X_j) : \boldsymbol{x} \in X_j\}$            ▹ Center partition $X_j$
6:          $C_j \leftarrow \bigcup_{i=1}^{n}\{-x_i \leq -\min_{x_i \in X'_j} x_i,\ \ x_i \leq \max_{x_i \in X'_j} x_i\}$
7:          $C_j \leftarrow C_j \cup \bigcup_{l=1}^{c_{max}}\{\text{LocalSearch}(X'_j)\}$
8:          $C_j \leftarrow$ DropRedundantConstraints($C_j$)
9:          $C_j \leftarrow \{\boldsymbol{w}\cdot\boldsymbol{x} \leq w_0 + \boldsymbol{w}\cdot E(X_j) : \boldsymbol{w}, w_0 \in c \in C_j\}$   ▹ Revert centering
10:     $C \leftarrow \bigcup_{j=1}^{n}\{\boldsymbol{w}\cdot\boldsymbol{x} + Lb_j \leq w_0 + \boldsymbol{w}\cdot E(X_j) + L : \boldsymbol{w}, w_0 \in c \in C_j\}$
11:     return $C \cup \{\sum_{j=1}^{k} b_j \geq 1\}$
12: function XMEANS($X, k$)
13:     $P_{best} \leftarrow \emptyset$                                  ▹ Output partitioning
14:     repeat
15:         $P \leftarrow \emptyset$                               ▹ Candidate partitioning
16:         $P_k \leftarrow$ KMEANS($X, k$)                      ▹ k-means partitioning
17:         for $X_j \in P_k$ do                              ▹ Split each partition
18:             $\{X'_j, X''_j\} \leftarrow$ KMEANS($X_j, 2$)
19:             if $BIC(\{X_j\}) \geq BIC(\{X'_j, X''_j\})$ then $P \leftarrow P \cup \{X_j\}$
20:             else $P \leftarrow P \cup \{X'_j, X''_j\}$
21:         if $BIC(P) > BIC(P_{best})$ then $P_{best} \leftarrow P$
22:         $k \leftarrow |P|$
23:     until $P = P_k$
24:     return $P_{best}$
25: function LocalSearch($X'_j$)
26:     $\boldsymbol{w} \sim \mathcal{N}(0, 900)^n$                  ▹ Draw weights from $\mathcal{N}(0, 30)$
27:     $w_0 \leftarrow \boldsymbol{w}\cdot\boldsymbol{x},\ \ \boldsymbol{x} \sim \mathcal{U}(X'_j)$                 ▹ Calculate free term
28:     if $|\{\boldsymbol{x} \in X'_j : \boldsymbol{w}\cdot\boldsymbol{x} \leq w_0\}| < |\{\boldsymbol{x} \in X'_j : \boldsymbol{w}\cdot\boldsymbol{x} \geq w_0\}|$ then
29:         $\boldsymbol{w} \leftarrow -\boldsymbol{w}$
30:         $w_0 \leftarrow -w_0$
31:     $s_l \leftarrow s_0 \leftarrow 1$                          ▹ Initialize step coefficients
32:     while $\exists_{\boldsymbol{x} \in X'_j} \boldsymbol{w}\cdot\boldsymbol{x} > w_0$ do        ▹ A violatied example exists
33:         for i=1..n do                                    ▹ For each weight
34:             $\Delta w_i \leftarrow s_i|w_i|/\overline{|w|}$                ▹ Calculate weight change
35:             if $\frac{|\{\boldsymbol{x} \in X'_j : [w_l + \mathbf{1}(l=i)\Delta w_i]_{l=1}^{n}\cdot\boldsymbol{x} > w_0\}|}{|\{\boldsymbol{x} \in X'_j : [w_l - \mathbf{1}(l=i)\Delta w_i]_{l=1}^{n}\cdot\boldsymbol{x} > w_0\}|} > 1$ then
36:                 $\Delta w_i \leftarrow -\Delta w_i$   ▹ Flip change if it causes less violations
37:             $s_i \leftarrow 1 + \frac{|\{\boldsymbol{x} \in X'_j : [w_l + \mathbf{1}(l=i)\Delta w_i]_{l=1}^{n}\cdot\boldsymbol{x} > w_0\}|}{|\{\boldsymbol{x} \in X'_j : \boldsymbol{w}\cdot\boldsymbol{x} > w_0\}|}$
38:         $s_0 \leftarrow 1 + \frac{|\{\boldsymbol{x} \in X'_j : \boldsymbol{w}\cdot\boldsymbol{x} > w_0\}|}{|X'_j|}$
39:         $\boldsymbol{w} \leftarrow \boldsymbol{w} + \Delta\boldsymbol{w}$
40:         $w_0 \leftarrow w_0 + s_0\overline{|w|}$
41:     return $\boldsymbol{w}\cdot\boldsymbol{x} \leq w_0$
42: function DropRedundantConstraints($C_j$)
43:     $V \leftarrow \{\boldsymbol{x} \sim \mathcal{U}(\Pi_{i=1}^{n}\mathbb{D}_i)\}$ ▹ Sample Cartesian product of domains
44:     return $\{c \in C_j : \exists_{\boldsymbol{x} \in V} \forall_{c' \in C_j \setminus \{c\}} \neg c(\boldsymbol{x}) \wedge c'(\boldsymbol{x})\}$
```

$k = |P|$ until P equals P_k. Finally, the best partitioning P_{best} is returned.

3.2.2 Centering of example set and translation of constraints. OCCALS centers each partition X_j, i.e., subtracts a vector of expected values from all examples in X_j in line 5 of Algorithm 1 to facilitate search in the space of LP constraints in line 7.

We explain this without loss of generality using a single LP constraint c. Consider a local search algorithm that adds small $\epsilon > 0$ to a single weight in c, resulting in c'. Then, the angle between the hyperplanes corresponding to c and c' would be small too, however the distance between an arbitrary chosen point p on the hyperplane of c and the closest point in c' increases with the distance of p to the origin of the coordinate system. This property affects fine-tuning of weights: for a centered set of examples X'_j, c and c' are likely met for the same examples in X'_j and the difference between them may be ineffective, however for $X_j = \{\boldsymbol{x} + E(X_j), \boldsymbol{x} \in X'_j\}$ where $E(X_j)$ is an arbitrary vector, one of c and c' may be met for all examples in X_j and the other for none. In practical setting, $E(X_j)$ can be arbitrarily long and when searching for constraints for X_j one should tune ϵ for every X_j independently. By centering X_j, one can expect ϵ being more stable across sets, and thus can be tuned in advance.

After producing a set of constraints C_j for the centered partition X'_j, OCCALS translates C_j by the vector $E(X_j)$ in line 9 to bound the original partition X_j. This is done by adding $\boldsymbol{w} \cdot E(X_j)$ to the free term w_0 in each constraint $c \in C_j$ (marked in red for brevity).

3.2.3 Local search. For a centered partition X'_j, OCCALS in line 7 of Algorithm 1 locally searches the space of LP constraints for the constraints met for all $\boldsymbol{x} \in X'_j$ and close to them. Each constraint c is searched independently. It initially draws n weights from $\mathcal{N}(0, 900)$ in line 26 and calculates an initial value of the free term w_0 in line 27 as a dot product of $\boldsymbol{w}$ and a uniformly drawn $\boldsymbol{x}$ from X'_j. The variance $\sigma^2 = 900$ was determined in a preliminary experiment and allows producing a weight vector pointing at virtually any direction. In lines 28–30, OCCALS flips the direction of c if this causes c being met for more examples in X'_j. The loop in lines 32–40 gradually increases the number of examples satisfying c. In line 34, it calculates candidate Δw_i for each weight w_i independently based on step coefficient s_i and ratio $|w_i|/\overline{|w|}$, where $\overline{|w|}$ is a mean of absolute values of weights in $\mathbf{w}$. The term $|w_i|/\overline{|w|}$ is supposed to relate the magnitude of Δw_i to the current magnitude of w_i. Δw_i is to be added to w_i and other weights $w_{l,l \neq i}$ remain intact to formulate a candidate constraint c'. Δw_i is flipped in line 36 if $-\Delta w_i$ results in c' having less number of violations in X'_j. In line 37, s_i is updated based on the ratio of the numbers of examples violating c' and c and decreases if c' is violated less times than c to avoid producing a candidate constraint located far from the boundary examples. Next in line 38, step coefficient s_0 is updated based on the ratio of the number of examples violating c to the total number of examples in X'_j and decreases when the former is low to reduce magnitude of change of w_0 for c being close to the boundary examples. The vector of candidate changes $\Delta\mathbf{w}$ is added to weight vector $\mathbf{w}$ in line 39, and $s_0\overline{|w|}$ is added to the free term w_0 in line 40. The loop terminates if all examples in X'_j satisfies c or a limit on the number of iterations exceeds which we omitted from Algorithm 1 for brevity. Finally, a constraint made of $\mathbf{w}$ and w_0 is returned in line 41.

3.2.4 Drop of redundant constraints. Since C_j contains the bounding-box constraints and the constraints sought independently using local search, some of them may be redundant w.r.t. the others. A constraint $c \in C_j$ is *necessary* iff exist $\boldsymbol{x} \in \mathbb{R}^n$ such that c is violated and all other $c' \in C_j \setminus \{c\}$ are met. Otherwise, c is *redundant*. Redundant constraints do not affect the shape nor size of the feasible region but hinder readability and solving of the model. For that reason, OCCALS drops redundant constraints in line 8 of Algorithm 1.

Due to technical difficulties, OCCALS verifies constraint redundancy approximately using a validation set V sampled uniformly from the Cartesian product of variables' domains (line 43). V's cardinality is a parameter, the larger $|V|$ the better approximation it provides. In line 44 a subset of necessary constraints in C_j is identified and in line 8 replaces C_j.

3.2.5 Alternative of LP models. As the final step, OCCALS merges the LP models C_j acquired for the individual partitions X_j into a single MILP model C having the interpretation of the alternative of C_j. To this aim, OCCALS introduces auxiliary binary variables b_j, each b_j controlling whether the constraints in C_j are active ($b_j = 1$) or inactive ($b_j = 0$). In line 10 of Algorithm 1 for each C_j each constraint $c \in C_j$ is summed with inequality $Lb_j \leq L$ (marked in red in that line), where $L \gg \|\mathbf{w}\|$ is a large constant. For $b_j = 1$, L added to the right-hand part of c is neutralized by Lb_j added to the left-hand part and the resulting constraint is equivalent to c. For $b_j = 0$, L added to the right-hand part of c makes it significantly larger than the left-hand part of c and the resulting constraint effectively does not bound $\boldsymbol{x}$. In line 11, OCCALS adds an extra constraint requiring that the constraints from at least one C_j are active.

4 EXPERIMENT

We carry out four experiments: tune the OCCALS parameters, verify whether acquiring separate LP models and joining them into a MILP model is beneficial w.r.t. acquiring a single LP model for multimodal problems, compare the performance of OCCALS to GOCCS [21], and visualize the acquired MILP models.

We use benchmarks formulated as LP and NLP models:
Cube$_n^k$:

$$\begin{aligned}
\forall_{j=1}^{k}\forall_{i=1}^{n} : x_i - Lb_j &\geq ij - L \\
\forall_{j=1}^{k}\forall_{i=1}^{n} : x_i + Lb_j &\leq ij + id + L \\
\textstyle\sum_{j=1}^{k} b_j &\geq 1 \\
\forall_{i=1}^{n} : x_i &\in [i - ikd, i + 2ikd] \\
\forall_{j=1}^{k} : b_j &\in \{0, 1\}
\end{aligned}$$

Simplex$_n^k$:

$$\begin{aligned}
\forall_{j=1}^{k}\forall_{i=1}^{n}\forall_{l=i+1}^{n} : x_i \cot\tfrac{\pi}{12} - x_l \tan\tfrac{\pi}{12} - Lb_j &\geq 2j - 2 - L \\
\forall_{j=1}^{k}\forall_{i=1}^{n}\forall_{l=i+1}^{n} : x_l \cot\tfrac{\pi}{12} - x_i \tan\tfrac{\pi}{12} - Lb_j &\geq 2j - 2 - L \\
\forall_{j=1}^{k} : \textstyle\sum_{i=1}^{n} x_i + Lb_j &\leq jd + L \\
\textstyle\sum_{j=1}^{k} b_j &\geq 1 \\
\forall_{i=1}^{n} : x_i &\in [-1, 2k + d] \\
\forall_{j=1}^{k} : b_j &\in \{0, 1\}
\end{aligned}$$

Ball_n^k:

$$\forall_{j=1}^{k} \sum_{i=1}^{n} \left(x_i - i - \frac{2\sqrt{6}(j-1)d}{i\pi}\right)^2 + Lb_j \le d^2 + L$$
$$\sum_{j=1}^{k} b_j \ge 1$$
$$\forall_{i=1}^{n} : x_i \in \left[i - 2d, i + \frac{2\sqrt{6}(k-1)d}{\pi} + 2d\right]$$
$$\forall_{j=1}^{k} : b_j \in \{0, 1\}$$

Their feasible regions are sampled uniformly for training sets X of examples $\mathbf{x}$, $|X| = 1{,}000$ for all experiments and auxiliary variables b_j are not part of $\mathbf{x}$ (cf. Sec. 3.1). The benchmarks are parameterized with the number of input variables n and the number of alternative subsets of constraints k that correspond to the number of modes in X. They extend the benchmarks from [18, 21] with k, and for $k = 1$ equal to those. Each benchmark has different characteristics: Cube_n^k requires $2kn$ single-variable constraints that resemble facets of k hypercubes, Simplex_n^k requires $kn(n-1)$ constraints involving two variables each and k constraints of all variables, Ball_n^k requires a large number of linear constrains of all variables approximating the actual k quadratic constraints corresponding to k hyperballs. All benchmarks involve an auxiliary constraint $\sum_{j=1}^{k} b_j \ge 1$ implementing an alternative.

All results are calculated using test sets T, $|T| = 500{,}000$, produced for particular benchmarks by sampling the Cartesian product of variables' domains and classifying the outcome as feasible or infeasible. Validation set V, $|V| = 2{,}000$, is used for removal of redundant constraints (cf. Sec. 3.2.4). Due to stochastic characteristic of OCCALS, each experiment is repeated 30 times for different X and T, and mean results are shown.

The experimental software is open-source.[1]

4.1 Parameter tuning

We tune using grid search the minimal number of partitions $k_{min} \in \{1, 2, 3\}$ and the number of constraints sought using local search $c_{max} \in \{50, 100, 500\}$. We evaluate OCCALS on all benchmarks with $n \in \{2, 3, 4, 5, 6\}$ variables and $k \in \{1, 2\}$ subsets of constraints.

Table 1 shows the mean and 0.95-confidence interval of F_1 score [9] on test set for the acquired models. F_1 score is a harmonic mean of precision and recall and attains values from 0 (bad) to 1 (good). Generally speaking, F_1 decreases with dimensionality n, which may be a sign of the *curse of dimensionality* [3], however the absolute values and decrease rate are problem-dependent. OCCALS achieves good and over 0.89 F_1 score for Cube with $k = 1$ and seems to scale well with n. We attribute this result to wrapping of the training set X into bounding box in line 6 of Algorithm 1, as bounding box constraints are parallel and for sufficiently large X close to the actual constraints in Cube benchmark. Worse results for $k = 2$ and high n may be because partitioning of X in line 2 leads to partitions which bounding boxes are not close to the actual facets of the hypercubes. Simplex turns out the most difficult to OCCALS with F_1 below 0.92. We hypothesize that the cause is the high number of actual constraints which result in an irregular and difficult to resemble with the acquired constraints shape of the feasible region. For $k = 2$, F_1 is higher than the corresponding values for $k = 1$, which may be due to the structure of the problem imposed by $k = 2$ that is easy to decompose into independently solvable parts. In Ball, F_1 seems independent from k. This is expected, as for $k = 2$ the balls have empty intersection, and assuming that XMEANS in line 2 of Algorithm 1 correctly partitions the examples belonging to each ball, enclosing each partition into an LP model is equally difficult.

Table 1: Mean F_1 score; bars reflect 0.95-confidence intervals (cell height corresponds to 0.1); heatmap corresponds to the value: green for 1, red for 0; p-values of Wilcoxon signed rank test of the best setup vs the others.

k_{min}:	1			2			3		
c_{max}:	50	100	500	50	100	500	50	100	500
Cube_2^1	0.997	0.996	0.992	0.995	0.994	0.991	0.993	0.992	0.989
Cube_3^1	0.994	0.993	0.987	0.990	0.984	0.975	0.985	0.982	0.972
Cube_4^1	0.993	0.991	0.984	0.984	0.980	0.968	0.980	0.973	0.956
Cube_5^1	0.991	0.990	0.980	0.982	0.975	0.953	0.977	0.962	0.935
Cube_6^1	0.987	0.985	0.967	0.961	0.945	0.909	0.952	0.934	0.899
Simp_2^1	0.847	0.857	0.883	0.857	0.864	0.898	0.862	0.868	0.898
Simp_3^1	0.686	0.725	0.773	0.714	0.742	0.792	0.731	0.758	0.804
Simp_4^1	0.496	0.574	0.672	0.537	0.605	0.691	0.591	0.636	0.714
Simp_5^1	0.357	0.432	0.565	0.394	0.466	0.608	0.453	0.523	0.644
Simp_6^1	0.212	0.291	0.436	0.251	0.341	0.488	0.315	0.385	0.508
Ball_2^1	0.958	0.971	0.987	0.961	0.973	0.985	0.966	0.974	0.983
Ball_3^1	0.886	0.922	0.964	0.905	0.930	0.959	0.907	0.929	0.952
Ball_4^1	0.785	0.850	0.922	0.820	0.871	0.921	0.830	0.879	0.915
Ball_5^1	0.678	0.765	0.866	0.719	0.801	0.875	0.743	0.812	0.869
Ball_6^1	0.523	0.633	0.776	0.581	0.690	0.816	0.619	0.722	0.817
Cube_2^2	0.947	0.950	0.950	0.992	0.989	0.984	0.981	0.980	0.977
Cube_3^2	0.857	0.872	0.885	0.923	0.928	0.936	0.926	0.928	0.927
Cube_4^2	0.761	0.778	0.816	0.842	0.859	0.877	0.852	0.863	0.876
Cube_5^2	0.601	0.638	0.676	0.707	0.770	0.815	0.761	0.767	0.826
Cube_6^2	0.904	0.926	0.889	0.904	0.941	0.941	0.926	0.963	0.907
Simp_2^2	0.874	0.887	0.906	0.893	0.900	0.920	0.861	0.875	0.897
Simp_3^2	0.700	0.745	0.795	0.728	0.771	0.822	0.728	0.763	0.812
Simp_4^2	0.519	0.578	0.699	0.524	0.594	0.704	0.546	0.604	0.704
Simp_5^2	0.361	0.433	0.567	0.379	0.453	0.588	0.390	0.468	0.585
Simp_6^2	0.400	0.503	0.645	0.354	0.460	0.651	0.432	0.487	0.664
Ball_2^2	0.935	0.948	0.962	0.941	0.955	0.968	0.957	0.964	0.971
Ball_3^2	0.877	0.909	0.940	0.877	0.910	0.939	0.880	0.908	0.932
Ball_4^2	0.783	0.843	0.901	0.781	0.842	0.902	0.797	0.851	0.894
Ball_5^2	0.678	0.765	0.853	0.679	0.766	0.853	0.697	0.779	0.844
Ball_6^2	0.548	0.655	0.786	0.546	0.653	0.781	0.574	0.676	0.787
Ranks	7.433	5.600	3.567	6.733	4.633	2.867	6.167	4.567	3.400
p-value	**0.000**	**0.000**	**0.010**	**0.000**	**0.000**	n/a	**0.000**	**0.000**	0.324

The last but one row of Table 1 shows mean ranks over problems for varying k_{min} and c_{max}. The lowest rank of 2.867 is achieved for $k_{min} = 2$ and $c_{max} = 500$. The Wilcoxon signed rank test [10] concludes superiority of this setup to seven out of eight remaining setups. This result shows that even for benchmarks with $k = 1$, imposed partitioning of the training set may be beneficial, and the higher the number of constraints sought c_{max} the more likely good constraints are found. In the following experiments, $k_{min} = 2$ and $c_{max} = 500$ are used, unless stated otherwise.

4.2 Multimodal training sets

We verify whether the partitioning of the multimodal training set X and joining LP models acquired independently for each partition

[1] https://github.com/SzybkiDanny/ConstraintsSynthesis/releases/tag/v1.0.0

Table 2: Mean F_1 and MCC w.r.t. k_{max}; bars reflect 0.95-confidence intervals (cell height corresponds to 0.1); heatmap corresponds to the value: green for 1, red for 0.

	F_1		MCC	
k_{max}:	1	∞	1	∞
Cube^2_2	0.950	0.984	0.950	0.984
Cube^2_3	0.885	0.936	0.889	0.937
Cube^2_4	0.816	0.877	0.827	0.880
Cube^2_5	0.676	0.815	0.712	0.831
Cube^2_6	0.889	0.941	0.902	0.947
Simplex^2_2	0.885	0.920	0.866	0.905
Simplex^2_3	0.676	0.822	0.706	0.829
Simplex^2_4	0.414	0.704	0.507	0.731
Simplex^2_5	0.237	0.588	0.359	0.637
Simplex^2_6	0.329	0.651	0.413	0.697
Ball^2_2	0.940	0.968	0.929	0.961
Ball^2_3	0.866	0.939	0.867	0.937
Ball^2_4	0.779	0.902	0.792	0.901
Ball^2_5	0.679	0.853	0.708	0.854
Ball^2_6	0.563	0.781	0.616	0.788
Ranks	2.000	1.000	2.000	1.000

leads to better results than a simplified algorithm that acquires a single LP model for X. To this aim, we introduce an extra parameter k_{max} bounding from top the total number of partitions returned by XMEANS in line 2 of Algorithm 1. We verify $k_{max} = 1$ (and so $k_{min} = 1$), being equivalent to not running XMEANS, and $k_{max} = \infty$, being equivalent to no upper bound on the total number of partitions (like in the previous experiment). We use all benchmarks with $k = 2$, as only those feature multimodal training sets.

Table 2 shows mean F_1 score and mean Matthews Correlation Coefficient (MCC) [16]. MCC measures quality of classification for imbalanced test sets and attains values from 1 (ideal classification), through 0 (random classification), to −1 (anti-ideal classification). For all problems, both F_1 score and MCC are higher for $k_{max} = \infty$. This confirms the hypothesis under question, even without need for conducting a statistical test. Also, for all the problems the discrepancy between the statistics obtained for $k_{max} = 1$ and $k_{max} = \infty$ increases with n. This may suggest that training set partitioning may be an efficient way to handle growing number of dimensions and the curse of dimensionality.

4.3 Comparison to GOCCS

We compare OCCALS to GOCCS [21] (cf. Sec. 2), by applying them to the same benchmarks for $n \in \{2, 3, 4, 5, 6\}$ and $k \in \{1, 2\}$. For OCCALS, we use the parameters tuned in Section 4.1, and for GOCCS we use the parameters tuned in [21] and shown in Table 3.

Table 4 shows mean and 0.95-confidence interval of F_1 score and MCC on test set for OCCALS and GOCCS. OCCALS achieves higher values of both measures, with only one exception of Simplex^1_2, for which the opposite holds. Also, OCCALS scales with n much better than GOCCS: for the latter both statistics drop rapidly to 0 for $n \geq 5$, while the former achieves relatively high values even for $n = 6$. Also, GOCCS is clearly incapable of handling multimodal training sets, as both F_1 and MCC are lower for $k = 2$ than the corresponding values for $k = 1$. The Wilcoxon signed rank test in the last row of Table 4 confirms the superiority of OCCALS to GOCCS for both F_1 and MCC.

Table 3: Parameters of GOCCS.

Parameter	Value
Population size	2000
Max generations	50
Max tree height	3
Instruction set	$\{+, -, \circledast, x_1, x_2, ..., x_n, \text{ERC}\}^a$
Initialization	Ramped Half-and-Half [13] with 1 to 20 constraints per individual
Parent selection	NSGA-II binary tournament [8]
Offspring selection	NSGA-II post-selection [8]
Search operators	Constraint Swapping Crossover: 0.6, Constraint Swapping Mutation: 0.1, Constraint Tree Crossover: 0.1, Constraint Tree Mutation: 0.1, Gaussian Constant Mutation: 0.1

[a] $\circledast$ is a strongly-typed multiplication accepting a constant as the first argument and any other instruction as the second; *ERC* is Euphemeral Random Constant [13] initially drawn from $\mathcal{N}(0, 1)$.

4.4 Visualization

We assess visually the quality of the MILP models acquired by OCCALS. Figure 1 shows feasible regions of the best MILP models on F_1 score acquired in Section 4.1 for all benchmarks with $n = 3$ and $k \in \{1, 2\}$. The red regions correspond to the feasible regions of the acquired MILP models, and the green to the actual feasible regions; variables' domains are reduced for the visualization. For all problems, the feasible regions of the acquired MILP models are close to the actual regions, and discrepancies between them are visible next to the acute corners (cf. Ball and Simplex) and curves (cf. Ball) of the actual regions. The former is likely caused by lack of feasible examples lying next to the angles in the training set, and the latter by inability to represent curves using MILP constraints.

The best model acquired for Cube^1_3 is (other problems omitted due to space limit):

$$\begin{aligned}
-x_1 &\leq 1.002\\
x_1 &\leq 3.694\\
-x_2 &\leq -2.003\\
x_2 &\leq 7.400\\
-x_3 &\leq -3.002\\
x_3 &\leq 11.097\\
52.699x_1 + 9.702x_2 + 22.295x_3 &\leq 496.458\\
-20.923x_1 + 18.721x_2 + -9.425x_3 &\leq 78.491\\
-22.379x_1 + 39.951x_2 + 51.506x_3 &\leq 829.722\\
11.552x_1 - 40.673x_2 + 28.675x_3 &\leq 257.235\\
40.415x_1 - 27.575x_2 - 18.061x_3 &\leq 20.661\\
-3.841x_1 + 42.503x_2 + 31.954x_3 &\leq 649.438\\
-78.016x_1 - 55.750x_2 + 2.669x_3 &\leq -172.369\\
-28.396x_1 + 12.692x_2 - 43.350x_3 &\leq -88.575
\end{aligned}$$

Table 4: Mean F_1 and MCC for OCCALS and GOCCS; bars reflect 0.95-confidence intervals (cell height corresponds to 0.1); heatmap corresponds to the value: green for 1, red for 0; p-values of Wilcoxon signed rank test for difference between both algorithms.

	F_1		MCC	
	OCCALS	GOCCS	OCCALS	GOCCS
Cube$^1_2$	0.991	0.467	0.991	0.434
Cube$^1_3$	0.975	0.301	0.975	0.287
Cube$^1_4$	0.968	0.140	0.968	0.173
Cube$^1_5$	0.953	0.062	0.954	0.115
Cube$^1_6$	0.909	0.031	0.915	0.084
Simplex$^1_2$	0.898	0.954	0.890	0.950
Simplex$^1_3$	0.792	0.768	0.805	0.789
Simplex$^1_4$	0.691	0.401	0.722	0.491
Simplex$^1_5$	0.608	0.120	0.653	0.227
Simplex$^1_6$	0.488	0.037	0.565	0.096
Ball$^1_2$	0.985	0.861	0.982	0.832
Ball$^1_3$	0.959	0.749	0.958	0.735
Ball$^1_4$	0.921	0.464	0.921	0.502
Ball$^1_5$	0.875	0.254	0.875	0.341
Ball$^1_6$	0.816	0.099	0.819	0.196
Cube$^2_2$	0.984	0.411	0.984	0.446
Cube$^2_3$	0.936	0.175	0.937	0.267
Cube$^2_4$	0.877	0.061	0.880	0.150
Cube$^2_5$	0.815	0.017	0.831	0.075
Cube$^2_6$	0.941	0.005	0.947	0.040
Simplex$^2_2$	0.920	0.905	0.905	0.886
Simplex$^2_3$	0.822	0.640	0.829	0.662
Simplex$^2_4$	0.704	0.302	0.731	0.402
Simplex$^2_5$	0.588	0.099	0.637	0.203
Simplex$^2_6$	0.651	0.014	0.697	0.023
Ball$^2_2$	0.968	0.788	0.961	0.739
Ball$^2_3$	0.939	0.629	0.937	0.636
Ball$^2_4$	0.902	0.303	0.901	0.385
Ball$^2_5$	0.853	0.113	0.854	0.211
Ball$^2_6$	0.781	0.025	0.788	0.095
Ranks	1.033	1.967	1.033	1.967
p-value	**0.000**		**0.000**	

$$4.325x_1 - 25.003x_2 - 29.407x_3 \leq -134.066$$
$$-61.715x_1 - 24.395x_2 - 38.406x_3 \leq -263.834$$
$$37.019x_1 - 15.464x_2 + 23.845x_3 \leq 345.091$$
$$-51.377x_1 - 27.124x_2 + 4.317x_3 \leq -69.335$$

The first six constraints implement bounding box of the training set and differ from the the actual constraints on third decimal place in the free terms. The remaining 12 constraints cut corners of the cube, which is likely due to the abovementioned lack of examples.

5 DISCUSSION

MILP models with applications in optimization and simulation of systems are common in both business and academia. Designing those models, however, is challenging for human experts for the reasons elaborated in Introduction. OCCALS may reduce the burden on the expert, e.g., in a hypothetical use case in which OCCALS acquires the initial constraints from historical data and then the expert adjusts them to the domain knowledge, supplies with an objective function and optimize using an off-the-shelf solver. OCCALS's ability to acquire MILP models from one-class data is crucial in business environment, where erroneous states are rarely observed and thus underrepresented in the historical data. Also, most real systems operate in a repeatable manner and thus some their states are observed more commonly than others, resulting in likely multimodal distribution. Embracing characteristics of different modes into a single MP model may be futile. The OCCALS's approach to acquire separate LP models for the partitions detected in the training set proves effective in handling multimodal distributions (cf. Sec. 4.2).

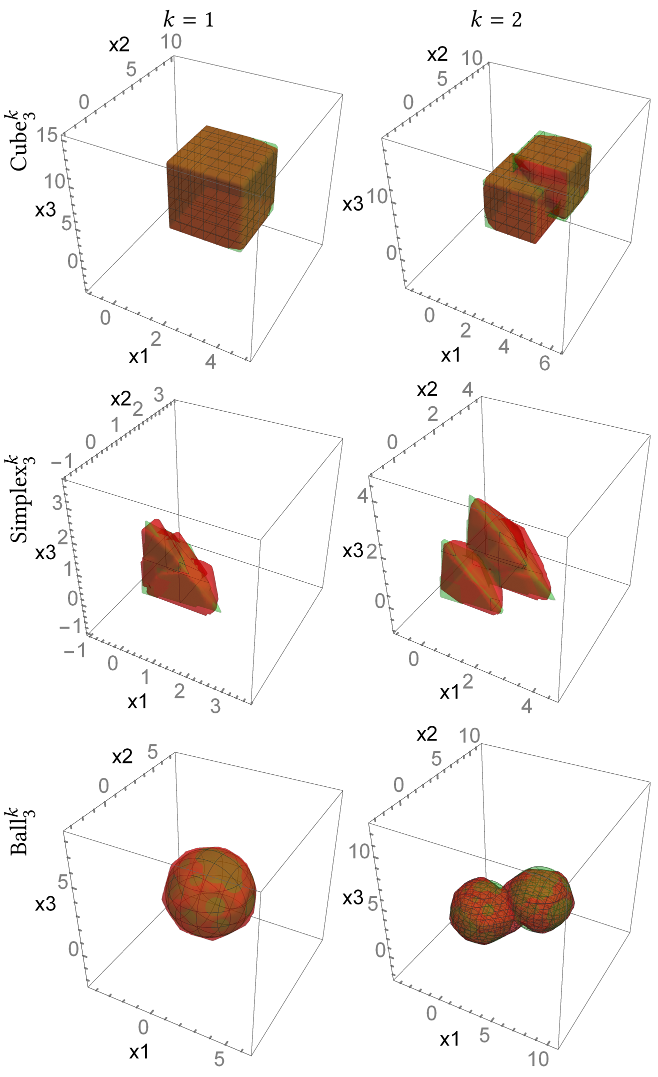

Figure 1: Feasible regions of the corresponding acquired (red) and actual (green) models.

It is worth to mention that OCCALS as a local-search-based algorithm is quick and fast, as opposed to e.g., the Genetic Programming-based GOCCS [21] that we compare to in Section 4.3. OCCALS turns out better than GOCCS on measures of model quality, e.g., F_1 score [9] and MCC [16], especially when multimodal training sets are considered.

Nevertheless, OCCALS is not free of challenges, one of which is susceptibility to the infamous curse of dimensionality [3]: the performance of OCCALS decreases with the number of problem variables. This, however, is an inherent characteristic of the constraint acquisition problem. To illustrate the influence of the curse of dimensionality, consider Ball benchmark from Section 4 with $k = 1$, which feasible region resembles the n-dimensional ball with radius d, center $\overline{\mathbf{x}} = [1, 2, \ldots, n]^T$ and volume $V \propto d^n$. For the sake of argument assume that a constraint acquisition algorithm produces a ball centered in $\overline{\mathbf{x}}$ too, but commits an error with radius by extending it with an infinitely small $\partial d > 0$. Then, the volume of the acquired ball constraint is $V' \propto (d + \partial d)^n$, and greater $V'/V \propto (1 + \partial d/d)^n$ times than V. The discrepancy between V' and V grows with n and for n going to infinity $\lim_{n\to\infty} V'/V = \infty$, i.e., infinitely small error in the radius results in the infinitely large increase of the ball volume. This small error may also dramatically influence quality measures based on test-set: large number of false positive examples occupying the infinitely large margin between the surfaces of the actual and the acquired balls may be the main factor influencing the values of those measures.

Another challenge is partitioning of the training set such that the individual LP models created for the individual partitions accurately fit data. OCCALS uses x-means [22] to detect partitions automatically. However, use of x-means brings implicit assumption that the training set distribution is a Gaussian mixture. This may not hold for all training sets observed in practice, for which the number of detected partitions would be wrong. To some extent, this challenge can be alleviated using k_{min} and k_{max} parameters which enforce the minimal and the maximal number of partitions, respectively. Given any sight on the actual distribution of the training set, one may consider substituting XMEANS call in line 2 of Algorithm 1 with another partitioning algorithm.

Also, OCCALS tends to produce oversize MILP models, see e.g. the exemplary MILP model acquired for Cube_3^1 in Section 4.4, in which the last 12 constraints can be considered unnecessary. However, facing no training examples in certain regions, the conservative approach to cut off those region with extra constraints is arguably safer than including them in the model, as the optimal solution for this model (for a linear objective function) lies on a constraint and without those extra constraints, that solution may happen infeasible in practice.

6 CONCLUSIONS AND FUTURE WORK

We proposed One-Class Constraint Acquisition with Local Search (OCCALS) method producing MILP models using examples of feasible states of the modeled entity. The key features of OCCALS are the support for acquisition of LP models from multimodal and one-class training sets. The experimental evaluation on synthetic benchmarks reveals high potential in tackling this type of acquisition problem. In particular, OCCALS is consistently better on multimodal training sets than a state-of-the-art alternative GOCCS [21]. OCCALS suffers from the curse of dimensionality [3], however this seems to be an inherent part of constraint acquisition problem.

Future research directions include integrating OCCALS with methods of dimensionality reduction to mitigate the curse of dimensionality, e.g., Principal Components Analysis [9], verification of OCCALS on real-world problems, review of alternative training set partitioning algorithms, and replacement of local search with more powerful metaheuristics, e.g., Tabu Search and Simulated Annealing [15].

Acknowledgment. This work was supported by the National Science Centre, Poland, grant no. 2016/23/D/ST6/03735 and the Foundation for Polish Science (FNP).

REFERENCES

[1] Abhilasha Aswal and G. N. Srinivasa Prasanna. 2010. Estimating Correlated Constraint Boundaries from Timeseries Data: The multi-dimensional German Tank Problem. In *EURO 2010*. http://slideplayer.com/slide/7976536/.
[2] Nicolas Beldiceanu and Helmut Simonis. 2012. A Model Seeker: Extracting Global Constraint Models From Positive Examples. In *CP'12*, Vol. LNCS 7514. Springer, Quebec City, Canada, 141–157. https://doi.org/10.1007/978-3-642-33558-7_13
[3] R. Bellman. 2013. *Dynamic Programming*. Dover Publications.
[4] Christian Bessiere, Rémi Coletta, Emmanuel Hebrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. 2013. Constraint Acquisition via Partial Queries. In *IJCAI 2013*. 475–481.
[5] Christian Bessiere, Remi Coletta, Frédéric Koriche, and Barry O'Sullivan. 2005. *A SAT-Based Version Space Algorithm for Acquiring Constraint Satisfaction Problems*. Springer Berlin Heidelberg, 23–34. https://doi.org/10.1007/11564096_8
[6] Christian Bessiere, Remi Coletta, Barry O'Sullivan, and Mathias Paulin. 2007. Query-Driven Constraint Acquisition. In *IJCAI 2007*. 50–55.
[7] Benjamin Blonder, Christine Lamanna, Cyrille Violle, and Brian J. Enquist. 2014. The n-dimensional hypervolume. *Global Ecology and Biogeography* 23 (2014), 595–609. https://doi.org/10.1111/geb.12146
[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *Trans. Evol. Comp* 6, 2 (April 2002), 182–197. https://doi.org/10.1109/4235.996017
[9] Peter Flach. 2012. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, New York, NY, USA.
[10] G.K. Kanji. 1999. *100 Statistical Tests*. SAGE Publications.
[11] Shehroz S. Khan and Michael G. Madden. 2014. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review* 29, 3 (2014), 345–374. https://doi.org/10.1017/S026988891300043X
[12] Samuel Kolb. 2016. Learning Constraints and Optimization Criteria. In *AAAI Workshops*.
[13] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.
[14] Michele Lombardi, Michela Milano, and Andrea Bartolini. 2017. Empirical decision model learning. *Artificial Intelligence* 244, Supplement C (2017), 343–367.
[15] Sean Luke. 2009. *Essentials of Metaheuristics* (first ed.). lulu.com.
[16] B.W. Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405, 2 (1975), 442 – 451. https://doi.org/10.1016/0005-2795(75)90109-9
[17] B. Mayoh, E. Tyugu, and J. Penjam. 2013. *Constraint Programming*. Springer.
[18] Tomasz P. Pawlak. 2018. Synthesis of Mathematical Programming Models with One-Class Evolutionary Strategies. *Swarm and Evolutionary Computation* (2018).
[19] Tomasz P. Pawlak and Krzysztof Krawiec. 2017. Automatic synthesis of constraints from examples using mixed integer linear programming. *European Journal of Operational Research* 261, 3 (2017), 1141 – 1157.
[20] Tomasz P. Pawlak and Krzysztof Krawiec. 2017. Synthesis of Mathematical Programming Constraints with Genetic Programming. In *EuroGP'17 (LNCS)*, Vol. 10196. Springer, 178–193. https://doi.org/doi:10.1007/978-3-319-55696-3_12
[21] Tomasz P. Pawlak and Krzysztof Krawiec. 2018. Synthesis of Constraints for Mathematical Programming with One-Class Genetic Programming. *IEEE Transactions on Evolutionary Computation* (2018).
[22] Dan Pelleg and Andrew Moore. 2000. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *ICML 2000*. Morgan Kaufmann, 727–734.
[23] Kostyantyn M. Shchekotykhin and Gerhard Friedrich. 2009. Argumentation Based Constraint Acquisition. In *ICDM 2009*. 476–482.
[24] Stefano Teso, Roberto Sebastiani, and Andrea Passerini. 2017. Structured learning modulo theories. *Artificial Intelligence* 244 (2017), 166–187.
[25] H.P. Williams. 2013. *Model Building in Mathematical Programming*. Wiley.

[26] David H. Wolpert. 1996. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation* 8, 7 (1996), 1341–1390.