Synthesis of Mathematical Programming Models with One-Class Evolutionary Strategies

Tomasz P. Pawlak*

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

Abstract

We propose an Evolutionary Strategy-based One Class Constraint Synthesis (ESOCCS), a novel method for computer-assisted synthesis of Mathematical Programming models. ESOCCS synthesizes constraints of Linear Programming and Non-Linear Programming types using examples of solely feasible states of the modeled entity. This is a crucial feature from the viewpoint of modeling real-world business processes from data, as acquisition of examples of feasible states is straightforward for a normally operating process, while infeasible states corresponding to errors and faults are avoided in practice and thus uncommon. We verify ESOCCS on a suite of synthetic benchmarks having known model representation and compare the synthesized models with the actual models in terms of syntax and semantics, concluding noticeable fidelity. We employ ESOCCS and an off-shelf solver in a fully automated setup of modeling and optimization of a real-world business process of rice production. We validate the resulting optimal parameters of production in contexts of three exemplary rice farms and conclude practical feasibility of those parameters and likely increase of profit thanks to applying them.

Keywords: constraint acquisition, model induction, linear programming, quadratic programming, set cover, distribution

1. Introduction

Simulation and optimization of business processes are important tasks in industry aimed at e.g., prediction of processes' outcomes in changing environment and maximization of their operating revenue, respectively. An essential part of both these tasks is building a mathematical model of the business process. Manual model building, albeit common in practice, is trial-and-error procedure in which model is iteratively verified for conformance with really and improved. Building a conformant model requires insight into details of the modeled process

*Corresponding author
Email address: tpawlak@cs.put.poznan.pl (Tomasz P. Pawlak)

and experience in modeling techniques. Experts combining these two competencies are uncommon in practice. Also, some relationships between process parameters may be unknown even to an experienced expert or incompatible with assumed model type (e.g., linear), requiring thus advanced modeling techniques. All these challenges cause manual model building expensive and sometimes not affordable for institutions with limited budget.

We propose to automatize model building to alleviate the abovementioned challenges. Model building can be conveniently decomposed into two separate problems: modeling of relationships between process variables and modeling of process outcome. In this study, we tackle the former problem, as the latter is in the essence a regression problem with many existing solving methods.

We propose an Evolutionary Strategy-based One Class Constraint Synthesis (ESOCCS) method that synthesizes constraints for Mathematical Programming (MP) models automatically from examples of states of a modeled process. An MP model is commonly employed formalism that embraces definition of variables' domains, constraints expressing relationships between these variables, and an objective function representing the process outcome. Use of MP models is completely automated by solvers – software tools that given an MP model finds an assignment of variables optimal w.r.t. the objective function and satisfying all constraints. ESOCCS is parameterized to synthesize either Linear Programming (LP) or Non-Linear Programming (NLP) models and NLP models can be restricted to a specific class, depending on technical requirements of e.g., available solver.

ESOCCS builds models using examples of feasible states of the process, i.e., states reached by the process during normal execution. This stays in contrast to many methods known from literature (see review in Section 3) that employ also examples of infeasible states: errors, faults and other undesirables states. Infeasible states are rare in practice and acquisition of their examples is expensive, as the process likely run into faults to reach them. In the effect, the set of examples of infeasible states is often underestimated and location of the boundary between feasible and infeasible states is difficult to estimate. ESOCCS drops requirement for examples of infeasible states by estimating distribution of the feasible states.

From Machine Learning perspective, we solve one-class classification problem, where the objective is to describe specific features of the feasible states rather than separating them from infeasible states. Using MP model representation, it is crucial to properly locate boundary of the feasible region of the model. If this boundary wraps examples of feasible states too loosely, an optimal solution of the model that usually lies at this boundary may turn out to be inapplicable in practice. In contrast, for boundary tightly wrapping the examples, the optimal solution may be actually far from optimality. However, in practice tightly wrapping boundary should be preferred over loosely wrapping one, as an suboptimal solution is often better than an infeasible one. ESOCCS strives to synthesize tight boundary by locating it where probability of a state being feasible falls down below an adaptive threshold.

We verify ESOCCS on a set of synthetic benchmarks of parameterized com-

plexity and known representation in terms of MP models. This enables us to measure syntactic and semantic fidelity of the synthesized models to the actual ones and verify how characteristics of a problem influence characteristics of the synthesized models. Overall conclusions are promising and thus, we apply ESOCCS to a real-world problem of modeling a business process of rice production. We synthesize a Quadratic Programming model in which constraints represent characteristics of rice production and an objective function represents production profit. We solve it using an off-shelf solver in contexts of three rice farms, concluding practical applicability of the optimal solutions and likely increase of production profit.

2. Constraint synthesis

We split discussion on constraint synthesis into two sections: statement of the constraint synthesis problem and description of ESOCCS algorithm that solves this problem.

2.1. Constraint synthesis problem

Let $x_1, x_2, ..., x_n$ be variables, $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ be a state, and X be a set of examples of feasible states. A term $t_i(\boldsymbol{x})$ is a function of $\boldsymbol{x}$ and T is a set of terms. A constraint $c(\boldsymbol{x})$ is an expression $\sum_{i=1}^{m} w_i t_i(\boldsymbol{x}) \leq w_0$ where $w_i \in \mathbb{R}$ is a weight of term t_i and $w_0 \in \mathbb{R}$ is a constant. A constraint synthesis problem is a tuple (X, T) and a set of constraints C such that $\forall_{\boldsymbol{x} \in X} \forall_{c \in C}\, c(\boldsymbol{x})$ is a solution to this problem.

A constraint synthesis problem is *ill-posed* as it may have many solutions varying in characteristics. They depend on the set of available terms T, application, user's preference and possible other criteria not included in the above definition. The method described in Section 2.2 is designed to maintain beneficial trade-off between wrapping tightly the examples in X using constraints and low model complexity.

A Mathematical Programming model is typically formed of a set of constraints C being a solution to the above problem and an objective function. In the following, we assume that the objective function is given or synthesized using an other method which is out of the scope of this study. For this reason, we use the terms 'set of constraints' and 'model' interchangeably.

2.2. Constraint synthesis algorithm

Evolutionary Strategy-based One Class Constraint Synthesis (ESOCCS) is shown in Algorithm 1. It is composed of three steps:

1. Estimating distribution of the feasible states using the set of feasible examples X and sampling that distribution for examples of likely infeasible states (line 2).
2. Evolving population of constraints using Evolutionary Strategy (line 3).
3. Simplification of constraints in the best-of-run model (lines $4 - 8$).

In the following subsections, we describe these steps in detail.

Algorithm 1 ESOCCS algorithm. ExpectationMaximization(X) estimates parameters of Gaussian Mixture Model of X, BuildModel(P, X, U) selects subset of P covering sets X and U, Truncate(C, a) returns at most a elements of set C, Zeros(c) calculates number of zero weights in constraint c.

function ESOCCS(X)
  $U \leftarrow$ SampleUnlikelyStates(X)
  $C \leftarrow$ EvolutionaryStrategy(X, U)
  $C' \leftarrow \emptyset$ ▷ Constraint simplification
  for $c \in C$ **do**
    $c' \leftarrow$ZeroWeights($c, X, U, 0$)
    $c' \leftarrow$Normalize(c')
    $C' \leftarrow C' \cup \{c'\}$
  return C'
function SampleUnlikelyStates(X)
  $\boldsymbol{\theta} \leftarrow$ ExpectationMaximization(X) ▷ Estimate parameters of GMM
  $t \leftarrow \rho$th percentile of $\{\mathrm{p}(\boldsymbol{x}|\boldsymbol{\theta}) : \boldsymbol{x} \in X\}$ ▷ Low probability threshold
  $U \leftarrow \{\boldsymbol{x} : \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\theta}) \wedge \mathrm{p}(\boldsymbol{x}|\boldsymbol{\theta}) < t\} \wedge |U| = |X|$ ▷ Low probable states
  return U
function EvolutionaryStrategy(X, U)
  $P_1 \leftarrow$InitializePopulation
  $C^* \leftarrow P_1$ ▷ Best so far model
  for $g = 1..G$ **do** ▷ G generations
    $P' \leftarrow P_g$ ▷ Initialize $(\mu + \lambda)$ constraint pool
    for $c \in P$ **do** ▷ Produce λ offspring using CM
      for $i = 1..\lambda/\mu$ **do**
        $P' \leftarrow P' \cup \{\mathrm{CM}(c)\}$
    $P_{g+1} \leftarrow \emptyset$ ▷ Initialize next generation population
    while $|P_{g+1}| < \mu$ **do** ▷ Fill P_{g+1} with μ best constraints
      $C \leftarrow$ BuildModel(P', X, U)
      $f(C) < f(C^*) \implies C^* \leftarrow C$ ▷ Store best so far model
      $P' \leftarrow P' \setminus C$ ▷ Remove constraints from pool
      $P_{g+1} \leftarrow P_{g+1} \cup$ Truncate($C, \mu - |P_{g+1}|$)
  return C^*
function ZeroWeights(c, X, U, m_0)
  $X_c \leftarrow \{\boldsymbol{x} : \boldsymbol{x} \in X \wedge c(\boldsymbol{x})\}$ ▷ Feasible examples satisfying c
  $U_{\overline{c}} \leftarrow \{\boldsymbol{x} : \boldsymbol{x} \in U \wedge \neg c(\boldsymbol{x})\}$ ▷ Unlabeled examples violating c
  $c^* \leftarrow c$ ▷ Initialize best constraint
  for $j = m_0..m$ **do** ▷ For each weight w_j
    $c' \leftarrow c$
    $c'[w_j] \leftarrow 0$ ▷ Set $w_j = 0$ in c'
    if $\forall_{\boldsymbol{x} \in X_c} c'(\boldsymbol{x}) \wedge \forall_{\boldsymbol{x} \in U_{\overline{c}}} \neg c'(\boldsymbol{x})$ **then** ▷ If c' does not worsen model
      $c' \leftarrow$ ZeroWeights($c', X, U, m+1$) ▷ Try zero next weights
      Zeros(c') > Zeros(c^*) $\implies c^* \leftarrow c'$ ▷ Store best constraint
  return c^*
function Normalize(c)
  $w \leftarrow \arg\min_{j=0..m, c[w_j] \neq 0} ||c[w_j]| - 1|$ ▷ Find weight closest to 1 or -1
  $\forall w_j : c[w_j] \leftarrow c[w_j]/|w|$ ▷ Divide all weights by $|w|$
  return c

2.2.1. Estimating distribution of feasible states

In this step, we estimate parameters of Gaussian Mixture Model (GMM) of feasible states using the set of feasible examples X given in the problem statement. The number of multivariate Gaussian distributions employed in GMM should increase with the number of variables n, since we expect that complexity of data distribution increases with n. However, the exact formula for this number depends on many factors. In a preliminary experiment we formulated a rule of thumb to use n Gaussians, as in most cases this maintains good trade-off between fidelity to actual probability distribution and model complexity.

We estimate parameters of GMM in line 11 of Algorithm 1 using Expectation-Maximization (EM) method by Dempster et al. [8] and its implementation by Souza [26]. EM finds maximum likelihood of parameters of a statistical model by repeating two steps:

- Expectation step: Calculate the expected value of the log likelihood function $Q(\boldsymbol{\theta}|\boldsymbol{\theta}_i) = E_{Z|X,\boldsymbol{\theta}_i}[\log p(X, Z|\boldsymbol{\theta})]$, where Z is a set of unobserved data of latent variables, $\boldsymbol{\theta}$ is a vector of the parameters, $\boldsymbol{\theta}_i$ is current estimate of the parameters, and p is probability density function,
- Maximization step: Find the parameters maximizing that function: $\boldsymbol{\theta}_{i+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}_i)$.

Once parameters $\boldsymbol{\theta}$ of GMM are estimated, we sample GMM for examples of states having low probability of feasibility, i.e., unlabeled but likely infeasible. To do this, in line 12 we calculate threshold t on p as the ρth percentile of values of p for all $\boldsymbol{x} \in X$. Use of small ρ is intended to filter out outlying examples having extremely low values of p that otherwise would underestimate t. Then, in line 13 we sample GMM for $|X|$ unlabeled examples $\boldsymbol{x}$ having $p(\boldsymbol{x}|\boldsymbol{\theta}) < t$ and store them in set U. The sets X and U are involved in assessment of models in Evolutionary Strategy.

2.2.2. Evolutionary Strategy

The main part of ESOCCS is evolution of constraints using $(\mu+\lambda)$-Evolutionary Strategy (Beyer and Schwefel [7]). We evolve a population P of μ constraints, each represented as a vector consisting of

- Weights $w_i, i = 1..m$,
- Constant w_0,
- Standard deviations $\sigma_i, i = 0..m$ corresponding to w_is,
- Rotation angles $\alpha_{ij}, i = 1..m, j = 0..m, i > j$, corresponding to pairs of w_i and w_j.

Where w_is are parameters of constraint $c(\boldsymbol{x})$ as defined in Section 2.1, while σ_i and α_{ij} are parameters of probability distribution involved in mutation of w_is.

The weights w_i and constant w_0 are initially drawn from $\mathcal{N}(0,1)$, standard deviations σ_i are set to 1 and angles α_{ij} to 0 for each new constraint in P, which is exemplified by line 16 of Algorithm 1.

We employ Correlated Mutation (CM) operator as described by Eiben and Smith [10, Ch4]. It mutates the given parent constraint using multivariate Gaussian distribution parameterized by σ_i and α_{ij}. Those parameters are different for each constraint and adapted by CM. The algorithm of CM is given by equations:

$$\forall_i : \sigma_i' \sim \sigma_i e^{\tau' N} e^{\tau \mathcal{N}(0,1)} \qquad \text{Mutation of standard deviations}$$

$$\forall_{ij} : \alpha_{ij}' \sim \beta \mathcal{N}(\alpha_{ij}, 1) \qquad \text{Mutation of angles}$$

$$c_{ij} = \begin{cases} {\sigma_i'}^2 & i = j \\ \frac{1}{2}({\sigma_i'}^2 - {\sigma_j'}^2)\tan(2\alpha_{ij}') & i > j \\ \frac{1}{2}({\sigma_j'}^2 - {\sigma_i'}^2)\tan(2\alpha_{ji}') & i < j \end{cases}$$

$$C = \begin{bmatrix} c_{00} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mm} \end{bmatrix} \qquad \text{Covariance matrix}$$

$$\boldsymbol{w}' \sim \boldsymbol{\mathcal{N}}(\boldsymbol{w}, C) \qquad \text{Multivariate Gaussian mutation of weights}$$

where $N \sim \mathcal{N}(0,1)$ (the same number for all i), $\tau = 1/\sqrt{2\sqrt{m+1}}$, $\tau' = 1/\sqrt{2(m+1)}$, β is a parameter of strength of angle mutation, $\boldsymbol{w} = [w_0 \dots w_m]^T$, and symbols with apostrophe refer to values in offspring. To maintain σ_i' and α_{ij}' in correct range we employ two boundary rules: $\sigma_i' < 0.001 \Rightarrow \sigma_i' = 0.001$ and $|\alpha_{ij}'| > \pi \Rightarrow \alpha_{ij}' = \alpha_{ij}' - 2\pi \text{sign}(\alpha_{ij}')$.

For each generation population P_g, we run CM for all μ parents in P_g producing so λ offspring, i.e., λ/μ offspring from each parent (lines $20-22$). μ parents and λ offspring form constraint pool P' that we use to iteratively build new models (line 25) as described in Section 2.2.3. Constraints are accepted to the next generation population P_{g+1} using $(\mu+\lambda)$-survival selection that accepts all constraints from successively built models until $|P_{g+1}| = \mu$ (line 24). Models are built in descending quality, thus only constraints included in the best models are accepted and the remaining λ constraints are dropped. If the number of constraints in a model is greater than the number of free slots in P_{g+1}, then an arbitrarily chosen subset of constraints from a model is accepted (line 28).

2.2.3. Construction of models and fitness assessment

We build a model using constraints from P' in line 25 of Algorithm 1 by selecting a set of constraints $C \subseteq P'$ such that it minimizes the sum of (a) number of times a feasible example in X violates a constraint in C and (b) number of unlabeled examples in U satisfying all constraints in C and (c) cardinality of C. To do this, we solve Mixed-Integer Linear Programming problem using Gurobi

Optimization, Inc. [12] solver:

$$\min \overbrace{\sum_{c_j \in P'} |X_{\overline{c_j}}| b_j}^{(a)} + \overbrace{\sum_{\boldsymbol{x}_k \in U} v_k}^{(b)} + 0.0001 \overbrace{\sum_{c_j \in P'} b_j}^{(c)} \tag{1}$$

subject to

$$\forall \boldsymbol{x}_k \in U : \sum_{\substack{c_j \in P', \\ \neg c_j(\boldsymbol{x}_k)}} b_j + v_k \geq 1 \tag{2}$$

$$\forall b_j, \forall v_j \in \{0, 1\}$$

where $X_{\overline{c_j}} = \{\boldsymbol{x} : \boldsymbol{x} \in X \wedge \neg c_j(\boldsymbol{x})\}$, $b_j = 1$ indicates that $c_j \in C$, $v_k = 1$ indicates that no constraint in C is violated for $\boldsymbol{x}_k \in U$. The term (a) penalizes use of constraints violating feasible examples, as they cause the conjunction of constraints in the model to be violated for these examples too. The term (b) penalizes the model for satisfying unlabeled examples, as they are unlikely feasible (cf. Section 2.2.1). Note that feasible example violating a constraint is equally costly like unlabeled example satisfying the model and errors of both types may compensate each other. The term (c) is to prefer the smallest model out of all models optimal w.r.t. sum of (a) and (b) as small models are often easier to interpret. The constant 0.0001 is to achieve lexicographic order of criteria (a) + (b) and (c).

This problem is related to classic Set Cover problem (Karp [17]), where a minimal set of sets of unlabeled examples violating corresponding constraints in P' is to be determined such that the union of these sets equals to U. In above problem, we actually soften this requirement by allowing some unlabeled examples to be absent in this union due to no guarantee that such union exists. Also, some sets have extra costs of use resulting from numbers of feasible examples violated by the corresponding constraints.

Eq. (1) is minimized fitness function $f(C)$ of model C and the best-so-far model is stored in line 26 of Algorithm 1.

The constraints used in model C are removed from pool P' (line 27) and model building repeats. Since the above problem is solved optimally, successively built models are not better w.r.t. $f(C)$. We use this fact to promote constraints to the next generation in the order of use in successive models as this reflects usefulness of constraints in building high quality models.

2.2.4. Simplification of constraints

When evolution ends, we simplify the produced model C by applying two steps to each constraint $c \in C$.

Given a constraint c, we zero all weights in c which zeroing do not worsen the model C. We employ branch & bound search for such weights in lines $30 - 40$ of Algorithm 1. The search works by iteratively zeroing weight w_j (line 36) and verifying whether all feasible examples previously satisfying c still satisfy and all unlabeled examples previously violating c still violate (line 37), i.e., whether

it does not worsen model C. If true, the search repeats recursively for the remaining weights. The constraint with the highest number of weights zeroed is stored in line 39.

We normalize scale of each constraint such that at least one w_i equals to 1 or -1. In line 42, we search for $w \neq 0$ which absolute value is closest to 1. Then, in line 43 we divide all w_j by $|w|$. We conduct normalization of scale because every constraint is equivalent to itself multiplied by a non-zero factor and we want to produce the same, i.e., canonical in the above sense, form of all these equivalent constraints. Also, constraints with integral weights are easier to read and interpret by humans. Our choice of w is dictated by the fact that division by a value close to 1 is almost neutral, in sense that may not significantly increase or decrease other w_j and/or introduce substantial numerical errors.

3. Related work

Related works can be split into those on one-class constraint synthesis, where the constraints are supposed to characterize a set of feasible examples like in the problem posed in Section 2.1, and two-class constraint synthesis, where the constraints are supposed to separate sets of feasible and infeasible examples. Also, representations of constraints vary across works, several works use Linear/Non-Linear Programming (LP/NLP) representation, others use Constraint Programming (CP, Mayoh et al. [20]) representation and some others use representations incompatible with Mathematical Programming paradigm.

Pawlak and Krawiec [24] propose *Genetic One-Class Constraint Synthesis* (GOCCS) system based on strongly-typed Genetic Programming (Montana [21]). GOCCS fed by a set of feasible examples constructs unlabeled examples using a custom adaptive technique and produces either LP or NLP model that separates the feasible examples from the unlabeled ones. Experimental assessment of GOCCS reveals susceptibility to the curse of dimensionality (Bellman [3]) and tendency to building oversize constraints when targeting NLP models. GOCCS was successfully applied to constraint synthesis for real-world wine production process.

Aswal and Prasanna [1] propose to build LP model by computing convex hull of a set of feasible examples and then clustering of constraints corresponding to hull's facets using k-means. Since in the worst-case scenario number of convex hull facets grows exponentially with number of variables, this method becomes computationally infeasible for over a dozen of variables.

Beldiceanu and Simonis [2] developed *Model Seeker* system that synthesizes CP-like constraints from a set of feasible examples using a handcrafted catalog of constraints annotated with metadata. The system consists of multiple layers responsible for example transformation, constraint generation, constraint simplification and translation of constraints to Prolog. Model Seeker yields promising results on several real-world problems.

Kolb [18] proposes another method that learns first-order logic clauses using inductive logic programming and estimates their weights using preference

learning. Performance of the method is human-competitive when assessed on a few synthetic benchmarks.

Alternative representations of models include e.g., modulo theories that Teso et al. [28] synthesizes from feasible examples and background knowledge constraints, and Bayesian models for constraints and objective function that Hernández-Lobato et al. [13] learn from examples.

Concerning two-class constraint synthesis, Pawlak and Krawiec [22] propose to transform constraint synthesis problem to a Mixed-Integer Linear Programming (MILP) problem and solve it optimally w.r.t. custom measure of model complexity using an off-shelf solver. Solving MILP problem is NP-hard and in practice solver must be interrupted prematurely returning thus suboptimal solution.

Pawlak and Krawiec [23] also propose *Genetic Constraint Synthesis* (GenetiCS) approach based on strongly-typed Genetic Programming (Montana [21]) like GOCCS. GenetiCS differs from GOCCS mainly in its requirement for infeasible examples in the training set.

Bessiere et al. [5] propose *Conacq* system using version space learning to build CP model from feasible and infeasible examples. The system is limited to finite-domain variables and does not support weights nor nonlinear transformations of variables. Bessiere et al. [6] extend Conacq to interactively query an expert for classification of highly informative examples from viewpoint of synthesis. Experiment shows that Conacq requires less than 100 queries to build a correct model.

Bessiere et al. [4] also propose *QuAcq* system that synthesizes constraints from examples with missing values of some variables, requiring so less knowledge from an expert. It iteratively requests the expert for classification of a new example, and depending on the answer removes constraints from a set of candidate constraints or moves a constraint from that set to a model. QuAcq is asymptotically optimal in the number of queries needed to converge for constraints involving only $=$ or $\neq$ comparisons.

The constraint synthesis problem as posed in Section 2.1 is a kind of one-class classification problem. Many classification models can be transformed into LP/NLP models, however this transformation may introduce undesirable properties to models. For instance, Support Vector Data Description (SVDD) by Tax [27] wraps feasible examples in a multidimensional sphere or with help of kernel functions in an arbitrary other shape. SVDD builds thus a single nonlinear constraint which precludes use of many conventional solvers. POSC4.5 algorithm by Denis et al. [9] given feasible examples samples unlabeled examples and then builds C4.5 decision tree (Quinlan [25]) using feasible and unlabeled examples as separate classes. A decision tree is a disjunction of conjunctions of linear constraints involving one variable each. Disjunction of constraints is not directly representable in LP models and requires auxiliary binary variables. This causes LP model solving to be NP-hard.

Table 1: Benchmarks formulated as MP models, d is a parameter used to bound variables' values, set to 2.7 to bound in non-integral values and make the benchmarks more realistic.

Ball$n$	Cube$n$
$d^2 \geq \sum_{i=1}^{n}(x_i - i)^2$	$\forall_{i=1}^{n} : x_i \geq i$
$\forall_{i=1}^{n} : x_i \in [i - 2d, i + 2d]$	$\forall_{i=1}^{n} : x_i \leq i + id$
	$\forall_{i=1}^{n} : x_i \in [i - id, i + 2id]$

Simplex$n$

$$\forall_{i=1}^{n}\forall_{j=i+1}^{n} : x_i \cot\tfrac{\pi}{12} - x_j \tan\tfrac{\pi}{12} \geq 0$$
$$\forall_{i=1}^{n}\forall_{j=i+1}^{n} : x_j \cot\tfrac{\pi}{12} - x_i \tan\tfrac{\pi}{12} \geq 0$$
$$\sum_{i=1}^{n} x_i \leq d$$
$$\forall_{i=1}^{n} : x_i \in [-1, 2 + d]$$

Table 2: Parameters of ESOCCS.

Part	Parameter	Value
Expectation Maximization	Convergence tolerance	10^{-11}
	Random initialization number	100
	Iteration limit	10^5
Sampling of unlikely states	Outlier cut-off percentile ρ	0.01
Evolutionary Strategy	Strength of angle mutation β	$5°$
Gurobi solver	Simplex iteration limit	1.5×10^6
	Visited MIP nodes limit	60,000

4. Experiment

4.1. Setup

We conduct two experiments. In the first one, we find parameters of ESOCCS that minimize mean fitness of the synthesized models (Eq. (1)). Then, we assess in detail fidelity of the models synthesized by the best ESOCCS setup to benchmark models.

In all experiments, we employ two sets of terms:

- Linear Programming set: $LP = \{t_i : \forall_{i=1}^{n} : t_i(\boldsymbol{x}) = x_i\}$,
- Quadratically Constrained Quadratic Programming set: $QCQP = LP \cup \{t_i : \forall_{i=1}^{n} t_i(\boldsymbol{x}) = x_i^2\}$.

We narrowed the scope of experiment to the term sets for synthesis of LP and QCQP models, as LP model is arguably the most common one, and QCQP model is natural extension of the former supported by all major solvers.

We use three benchmark problems represented as MP models in Table 1 and parameterized by number of variables n. The benchmarks have contrasting characteristics allowing us to assess ESOCCS's performance in synthesizing models of different types. Ball$n$ requires one quadratic constraint of n variables, Cube$n$ requires $2n$ linear constraints of one variable each, and Simplex$n$ requires $n(n-1)$ linear constraints of two variables each and one linear constraints of n variables.

Table 3: Mean best-of-run fitness, best in bold. Bars reflect .95-confidence intervals (cell height corresponds to 5, higher values truncated).

LP models

Problem	400x50x3	200x100x3	200x50x6	100x100x6	400x50x3S	200x100x3S	200x50x6S	100x100x6S
Ball3	3.57	3.43	4.00	3.40	3.07	**2.87**	3.10	3.10
Ball4	2.80	2.94	2.30	2.70	2.23	**2.00**	2.23	2.33
Ball5	4.40	4.04	4.44	4.14	2.44	**2.44**	2.54	2.70
Ball6	8.97	6.60	6.94	6.97	**2.37**	2.54	2.54	2.47
Ball7	16.50	16.47	15.07	17.17	**2.47**	2.70	2.80	3.34
Simplex3	2.90	2.73	3.17	3.07	**2.40**	2.57	2.40	2.50
Simplex4	2.33	2.20	2.73	2.47	**1.77**	2.20	1.87	2.23
Simplex5	2.73	2.13	2.20	2.23	1.80	**1.80**	2.03	1.93
Simplex6	4.43	3.10	2.83	2.53	2.07	**1.90**	1.90	2.00
Simplex7	6.43	4.80	5.03	5.13	2.23	**1.57**	1.93	1.87
Cube3	1.07	1.30	1.23	1.13	**0.87**	1.00	0.97	1.07
Cube4	2.47	1.43	1.40	1.83	**0.93**	1.03	0.97	0.93
Cube5	7.14	5.47	5.84	4.50	**1.07**	1.17	1.20	1.10
Cube6	20.44	14.54	15.14	12.84	**1.63**	1.80	2.03	1.83
Cube7	38.14	28.84	25.27	25.97	**2.50**	2.77	2.70	2.67
Rank:	7.27	5.93	6.40	6.27	1.67	2.27	3.07	3.13

QCQP models

Problem	400x50x3	200x100x3	200x50x6	100x100x6	400x50x3S	200x100x3S	200x50x6S	100x100x6S
Ball3	3.53	3.77	3.20	2.93	**1.80**	1.93	1.90	2.23
Ball4	4.50	3.67	3.83	3.73	2.53	2.57	**2.23**	2.60
Ball5	13.37	12.24	13.34	11.57	3.63	3.73	**3.23**	3.87
Ball6	19.60	17.50	18.34	18.54	5.80	**4.87**	6.10	4.87
Ball7	24.80	22.54	21.47	24.40	6.57	**5.73**	6.44	6.03
Simplex3	12.23	6.30	5.20	5.37	0.33	0.57	**0.30**	0.33
Simplex4	16.57	11.07	13.60	9.80	0.73	1.23	**0.53**	0.80
Simplex5	16.13	14.30	13.70	13.70	4.87	4.74	**4.14**	5.07
Simplex6	13.70	12.60	13.73	12.07	6.87	7.14	**5.34**	9.50
Simplex7	10.77	11.13	10.93	10.97	8.17	8.47	**7.70**	12.94
Cube3	2.17	2.33	1.93	2.40	0.53	**0.37**	0.53	0.40
Cube4	2.73	2.73	2.90	3.20	0.23	**0.23**	0.23	0.27
Cube5	5.57	5.33	4.80	5.63	0.50	0.77	0.60	**0.50**
Cube6	14.10	10.14	9.77	7.60	0.93	**0.90**	1.00	1.13
Cube7	19.04	14.10	15.04	15.00	**1.33**	1.60	1.57	1.87
Rank:	7.27	6.20	6.13	6.13	2.33	2.47	2.00	3.47

Table 2 shows parameters of ESOCCS fixed in all experiments. Parameters not shown there are either set to their defaults in Accord.NET [26] and Gurobi [12], or set in per-experiment basis. To draw statistically sound conclusions, we run ESOCCS 30 times for each problem and combination of parameters in all experiments.

Experimental software is available at [to be released after paper acceptance].

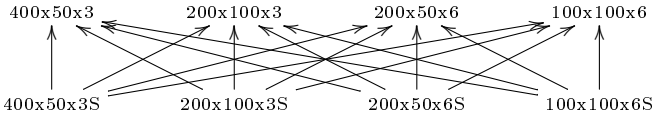
4.2. Parameter tuning

We employ all benchmarks from Table 1 for $n = 3..7$ and sample their feasible regions uniformly for $m = 300$ examples that form training sets X. We tune four parameters of ESOCCS: population size μ, number of generations G, offspring to parent ratio λ/μ and whether to standardize training set before running ESOCCS. Product of the first three parameters constitutes the total number of constraints evolved and we fix this number to 60,000 to maintain equal computational budget for each combination of parameters and make results comparable. We hypothesize that ESOCCS may benefit from standardization of the training set X, i.e., substituting each x_i in all examples $\boldsymbol{x} \in X$ with $x'_i =$

Table 4: The p-values of Friedman's tests and their post-hoc analyses: the p-values of incorrectly judging a setup in a row as outranking a setup in a column; $p \leq 0.05$ visualized as arcs in the graphs.

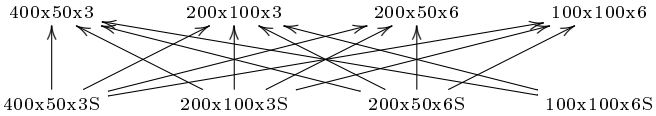
LP models; Friedman's p-value: 7.56×10^{-9}

	400x50x3	200x100x3	200x50x6	100x100x6	400x50x3S	200x100x3S	200x50x6S	100x100x6S
400x50x3								
200x100x3	0.812		1.000	1.000				
200x50x6	0.979							
100x100x6	0.953		1.000					
400x50x3S	**0.000**	**0.000**	**0.000**	**0.000**		0.998	0.792	0.702
200x100x3S	**0.000**	**0.001**	**0.000**	**0.000**			0.990	0.974
200x50x6S	**0.000**	**0.026**	**0.004**	**0.007**				1.000
100x100x6S	**0.000**	**0.041**	**0.007**	**0.012**				



QCQP models; Friedman's p-value: 6.27×10^{-8}

	400x50x3	200x100x3	200x50x6	100x100x6	400x50x3S	200x100x3S	200x50x6S	100x100x6S
400x50x3								
200x100x3	0.934							
200x50x6	0.911	1.000						
100x100x6	0.911	1.000	1.000					
400x50x3S	**0.000**	**0.000**	**0.001**	**0.001**		1.000		0.911
200x100x3S	**0.000**	**0.001**	**0.001**	**0.001**				0.953
200x50x6S	**0.000**	**0.000**	**0.000**	**0.000**	1.000	1.000		0.726
100x100x6S	**0.001**	**0.047**	0.058	0.058				



$(x_i - \overline{x_i})/\sigma_i$, where $\overline{x_i}$ and σ_i are mean and standard deviation of variable x_i in X, respectively. Transformation of a model synthesized using the standardized training set to a model for the original training set is done by substituting each occurrence of each x_i in the former with $(x_i - \overline{x_i})/\sigma_i$ in the latter. The costs of standardization and model transformation are negligible w.r.t. the evolutionary costs and we do not include them in the computational budget. To sum up, we employ eight setups:

Name	μ	G	λ/μ	Standardization
400x50x3	400	50	3	
200x100x3	200	100	3	
200x50x6	200	50	6	
100x100x6	100	100	6	
400x50x3S	400	50	3	✓
200x100x3S	200	100	3	✓
200x50x6S	200	50	6	✓
100x100x6S	100	100	6	✓

Table 3 shows mean and .95-confidence interval of the best-of-run fitness achieved by each setup. Unquestionably, training set standardization improves performance for all combinations of the other parameters. 400x50x3S achieves the lowest rank in synthesis of LP models and 200x50x6S in QCQP models.

To assess significance of superiority of these setups to the others, we conduct Friedman's tests (Kanji [16]) separately for syntheses of LP and QCQP models and show their outcomes and post-hoc analyses using symmetry test (Hothorn et al. [14]) in Table 4. The tests confirm that training set standardization leads to significant improvement of ESOCCS's performance. Also, each of 400x50x3S and 200x50x6S is significantly better than four other setups in synthesis of both LP and QCQP models.

4.3. Evaluation of the synthesized models

In this section, we assess fidelity of the synthesized models to the actual benchmark models. We run the best setups from the previous section, 400x50x3S for LP models and 200x50x6S for QCQP models, for all benchmarks from Table 1 for $n = 3..7$ and training sets X of $m \in \{100, 200, 300, 400, 500\}$ feasible examples. First, we measure fidelity of constraints' syntax, then fidelity of reconstruction of boundary of the feasible region.

Table 5 shows mean absolute difference in the numbers of constraints and terms of the synthesized and actual models. The numbers of terms are calculated as total numbers of summands in the constraints transformed to expanded polynomial form. In the synthesized models these numbers equal to the total numbers of non-zero weights and constants. ESOCCS produces more constraints than required for Ball$n$ problem. Although when targeting LP models this is explained by quadratic form of the actual constraint, when targeting QCQP models synthesis of just one constraint correctly characterizing the training set seems to be difficult for ESOCCS. LP models synthesized for Simplex$n$

Table 5: Mean difference in the numbers of (a) constraints, (b) terms in the synthesized and actual models. Positive values refer to the former being greater, negative otherwise. Bars reflect confidence intervals (cell height corresponds to 3, higher values truncated). Underlining marks means equal to 0 w.r.t. two-tailed t-test at $\alpha = 0.05$.

(a)

Ball$n$

	$m\backslash n$	3	4	5	6	7
LP models	100	6.80	8.33	9.73	10.50	10.87
	200	9.23	11.83	14.03	16.47	17.33
	300	11.20	15.13	18.43	21.37	24.73
	400	12.47	18.30	21.40	25.60	29.47
	500	14.10	20.67	25.23	31.13	34.30
QCQP models	100	2.80	4.90	7.47	8.57	9.67
	200	3.63	5.17	7.73	11.70	13.30
	300	4.13	6.40	9.30	11.13	16.80
	400	4.07	7.53	9.83	14.23	18.87
	500	5.07	8.33	10.70	15.77	20.53

Simplex$n$

	$m\backslash n$	3	4	5	6	7
LP models	100	-1.97	-6.83	-13.27	-21.87	-32.47
	200	-1.47	-6.33	-12.40	-20.37	-30.70
	300	-0.73	-5.23	-11.37	-18.93	-29.33
	400	-0.17	-4.53	-10.53	-17.73	-26.80
	500	0.37	-4.07	-10.07	-17.23	-25.90
QCQP models	100	-1.07	-5.03	-10.00	-18.03	-29.30
	200	0.27	-1.07	-5.10	-12.13	-22.07
	300	1.73	2.43	0.90	-5.17	-13.77
	400	3.30	4.20	6.10	1.80	-9.20
	500	4.40	8.37	9.57	8.17	-2.80

Cube$n$

	$m\backslash n$	3	4	5	6	7
LP models	100	0.43	0.43	-0.07	-1.40	-1.73
	200	1.07	1.30	2.40	2.07	2.03
	300	2.00	2.03	2.60	3.30	5.07
	400	2.40	2.67	3.70	3.50	6.00
	500	3.03	3.30	4.50	4.53	7.43
QCQP models	100	-2.40	-2.67	-3.17	-4.13	-5.07
	200	-1.50	-2.33	-2.70	-3.03	-3.73
	300	-1.60	-1.67	-2.33	-2.27	-2.20
	400	-1.03	-1.27	-1.37	-2.43	-1.40
	500	-0.57	-0.93	-0.80	-1.77	-1.33

(b)

Ball$n$

	$m\backslash n$	3	4	5	6	7
LP models	100	19.80	31.77	45.60	57.00	67.00
	200	26.97	45.70	66.00	92.20	109.57
	300	32.97	58.33	87.70	120.53	159.83
	400	36.73	70.93	102.50	144.53	192.90
	500	41.67	80.13	121.40	177.23	224.53
QCQP models	100	18.47	38.27	67.80	88.70	108.30
	200	23.60	42.00	73.50	124.63	157.20
	300	27.00	51.97	89.47	124.50	203.13
	400	26.40	60.40	96.17	158.50	233.13
	500	32.40	67.43	104.10	178.73	258.20

Simplex$n$

	$m\backslash n$	3	4	5	6	7
LP models	100	-0.80	-5.67	-10.17	-16.87	-26.37
	200	0.90	-2.93	-5.43	-8.33	-14.87
	300	3.13	1.37	0.10	1.10	-3.93
	400	5.13	4.60	3.83	7.67	13.57
	500	6.70	6.23	7.23	12.33	20.97
QCQP models	100	17.33	28.97	48.30	56.43	50.50
	200	25.67	57.13	90.67	117.67	132.90
	300	34.40	85.50	144.97	188.57	231.47
	400	44.13	97.97	192.17	260.63	280.23
	500	50.03	128.83	225.70	327.07	354.80

Cube$n$

	$m\backslash n$	3	4	5	6	7
LP models	100	10.77	22.13	34.97	44.40	61.87
	200	13.00	25.47	45.17	61.80	82.57
	300	15.57	27.67	45.13	68.73	102.60
	400	16.80	30.60	50.73	69.77	108.60
	500	18.67	33.03	53.80	75.50	117.77
QCQP models	100	13.60	30.33	49.23	65.10	83.63
	200	18.37	32.10	53.07	78.63	101.53
	300	18.47	37.47	56.47	86.40	119.93
	400	22.07	41.50	66.73	84.87	135.03
	500	24.83	44.07	71.40	92.80	135.17

have no more constraints than actual and the discrepancy of the numbers of the synthesized and the actual constraints increases with n. This may be because ESOCCS from all models correctly representing the training set strives to synthesize the smallest one. For Cube$n$ ESOCCS synthesizes a few more constraints than actual when targeting LP models except for $n \geq 6$ and $m \leq 100$ and a few less when targeting QCQP models. This may be because one nonlinear constraint may approximate more than one actual linear constraint.

ESOCCS uses more terms than required for all problems except several instances of Simplex$n$, and the number of terms is greater when targeting QCQP models. This suggests that cardinality of the term set, that for the considered LP and QCQP sets is respectively n and $2n$, influences size of the synthesized models and one should prefer small term sets to synthesize concise models.

To assess syntactic fidelity of constraints more deeply, we transform constraints to the expanded polynomial form and calculate mean *angle* between weight vectors $\mathbf{w}_i^s$ and $\mathbf{w}_j^b$ of the corresponding constraints in the synthesized and the actual models, respectively. Let α_{ij} be the angle between $\mathbf{w}_i^s$ and $\mathbf{w}_j^b$:

$$\alpha_{ij} = \arccos \left| \frac{\boldsymbol{w}_i^s \cdot \boldsymbol{w}_j^b}{\|\boldsymbol{w}_i^s\| \cdot \|\boldsymbol{w}_j^b\|} \right|.$$

Then, the corresponding constraints are sought by solving an *assignment problem* in which each constraint is paired with at least one other and the mean angle of all pairs is minimal:

min	$\frac{1}{N}$	$\sum_{ij} \alpha_{ij} b_{ij}$	the mean of angles
subject to			
$\forall i: \sum_j b_{ij}$	$\geq$	1	ith synth. constraint paired with at least one actual
$\forall j: \sum_i b_{ij}$	$\geq$	1	jth actual constraint paired with at least one synth.
$\forall b_{ij}$	$\in$	{0,1}	indicator of pairing the ith and jth constraints

Table 6: Mean angle in rad of the weight vectors of the synthesized and actual constraints. Bars reflect confidence intervals (cell height corresponds to 0.1).

		Ball$n$					Simplex$n$						Cube$n$					
	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7
LP models	100	1.02	1.12	1.20	1.20	1.24	100	0.15	0.19	0.29	0.39	0.47	100	0.22	0.41	0.52	0.64	0.69
	200	1.01	1.11	1.16	1.22	1.22	200	0.15	0.15	0.20	0.25	0.29	200	0.16	0.26	0.36	0.46	0.55
	300	1.03	1.12	1.16	1.22	1.23	300	0.18	0.16	0.19	0.22	0.26	300	0.17	0.22	0.27	0.36	0.45
	400	1.02	1.12	1.17	1.21	1.23	400	0.18	0.17	0.17	0.21	0.25	400	0.17	0.20	0.27	0.30	0.39
	500	1.01	1.12	1.19	1.21	1.24	500	0.17	0.17	0.18	0.21	0.24	500	0.18	0.22	0.25	0.29	0.37
	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7
QCQP models	100	0.50	0.76	0.83	0.92	0.94	100	0.62	0.77	0.88	0.96	1.03	100	0.58	0.68	0.72	0.79	0.78
	200	0.49	0.64	0.79	0.86	0.95	200	0.57	0.74	0.83	0.90	0.95	200	0.42	0.57	0.69	0.74	0.78
	300	0.48	0.60	0.73	0.80	0.93	300	0.55	0.73	0.84	0.91	0.97	300	0.41	0.52	0.61	0.67	0.70
	400	0.47	0.61	0.68	0.80	0.90	400	0.56	0.71	0.83	0.94	0.96	400	0.40	0.49	0.58	0.60	0.70
	500	0.48	0.61	0.67	0.81	0.87	500	0.54	0.71	0.83	0.92	0.99	500	0.42	0.48	0.55	0.59	0.67

Table 7: Mean Jaccard index of the feasible regions of the synthesized and actual models. Bars reflect confidence intervals (cell height corresponds to 0.1).

		Ball$n$					Simplex$n$						Cube$n$					
	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7
LP models	100	0.70	0.49	0.35	0.21	0.08	100	0.79	0.62	0.39	0.14	0.03	100	0.78	0.52	0.34	0.15	0.07
	200	0.77	0.63	0.50	0.35	0.22	200	0.84	0.71	0.53	0.22	0.13	200	0.85	0.71	0.54	0.34	0.19
	300	0.82	0.70	0.57	0.44	0.36	300	0.85	0.74	0.62	0.36	0.03	300	0.87	0.77	0.63	0.46	0.30
	400	0.83	0.72	0.60	0.47	0.38	400	0.87	0.74	0.61	0.41	0.03	400	0.89	0.79	0.68	0.51	0.35
	500	0.85	0.74	0.63	0.53	0.42	500	0.89	0.76	0.65	0.33	0.07	500	0.90	0.80	0.70	0.55	0.41
	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7
QCQP models	100	0.85	0.73	0.60	0.51	0.40	100	0.62	0.30	0.09	0.03	0.00	100	0.75	0.61	0.49	0.29	0.14
	200	0.89	0.84	0.75	0.65	0.57	200	0.75	0.43	0.16	0.05	0.02	200	0.85	0.72	0.62	0.49	0.35
	300	0.91	0.86	0.79	0.75	0.63	300	0.77	0.52	0.22	0.09	0.00	300	0.85	0.77	0.68	0.57	0.45
	400	0.93	0.88	0.82	0.74	0.69	400	0.79	0.59	0.25	0.12	0.03	400	0.87	0.79	0.70	0.61	0.46
	500	0.93	0.88	0.83	0.77	0.71	500	0.80	0.59	0.32	0.07	0.00	500	0.87	0.81	0.70	0.64	0.55

Where N is the greater of the numbers of the synthesized and the actual constraints (number of pairs). We solve this problem using Gurobi solver [12]. The resulting mean angle is in the range $[0, \frac{\pi}{2}]$ and attains 0 if all paired $\mathbf{w}_i^s$ and $\mathbf{w}_j^b$ are pairwise parallel, and $\frac{\pi}{2}$ if orthogonal. For LP models the mean angle equals to the mean angle between the hyperplanes of the synthesized and the actual constraints in the solution space. For the QCQP models, this relationship does not hold.

Table 6 shows mean angles between the corresponding synthesized and actual constraints. For all problems and model types the angles increase with n and decrease with m. For Ball$n$ the angles are lower (better) for QCQP models. This is expected, since Ball$n$'s constraint cannot be reconstructed using an LP model. For the remaining problems, angles for LP models are low, often ≤ 0.5rad, and lower than the corresponding angles for QCQP models. This is because LP syntax is less flexible than QCQP syntax and ESOCCS is more likely to find syntactically similar LP models to the actual ones.

In the following, we abstract from syntax of the synthesized models and focus on fidelity of reconstruction of the boundary of the feasible region. To calculate below statistics, we use test-set of 100,000 examples uniformly sampled from the Cartesian product of variables' domains.

Table 7 shows overlapping of the feasible regions of the synthesized and actual models using Jaccard index [15] – a measure attaining 1 for equal regions and 0 for disjoint. For Ball$n$ indexes are consistently higher when targeting QCQP models, while for Simplex$n$ when targeting LP models. These results

Table 8: Mean test-set precision of the synthesized models. Bars reflect confidence intervals (cell height corresponds to 0.1).

	Ball$n$					Simplex$n$					Cube$n$				
LP models	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7
	100	0.79	0.57	0.43	0.24	0.09	100	0.86	0.67	0.43	0.15	0.03	100	0.85	0.58	0.38	0.16	0.07
	200	0.84	0.71	0.58	0.40	0.24	200	0.87	0.75	0.57	0.23	0.13	200	0.90	0.76	0.59	0.36	0.20
	300	0.87	0.77	0.64	0.50	0.41	300	0.87	0.76	0.66	0.36	0.03	300	0.91	0.81	0.66	0.48	0.31
	400	0.88	0.79	0.67	0.53	0.43	400	0.89	0.76	0.63	0.41	0.03	400	0.91	0.83	0.71	0.53	0.37
	500	0.89	0.80	0.70	0.59	0.47	500	0.91	0.79	0.66	0.33	0.07	500	0.92	0.83	0.72	0.57	0.43
QCQP models	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7
	100	0.93	0.88	0.79	0.66	0.50	100	0.69	0.33	0.09	0.03	0.00	100	0.82	0.71	0.58	0.33	0.15
	200	0.94	0.91	0.86	0.76	0.66	200	0.79	0.45	0.17	0.05	0.02	200	0.90	0.78	0.68	0.54	0.39
	300	0.95	0.91	0.86	0.82	0.72	300	0.80	0.55	0.23	0.09	0.00	300	0.88	0.81	0.72	0.61	0.48
	400	0.95	0.92	0.88	0.81	0.77	400	0.82	0.61	0.25	0.12	0.03	400	0.89	0.82	0.74	0.64	0.49
	500	0.95	0.92	0.87	0.82	0.77	500	0.82	0.62	0.34	0.07	0.00	500	0.89	0.83	0.73	0.66	0.57

Table 9: Mean test-set recall of the synthesized models. Bars reflect confidence intervals (cell height corresponds to 0.1).

	Ball$n$					Simplex$n$					Cube$n$				
LP models	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7
	100	0.85	0.79	0.68	0.69	0.68	100	0.91	0.90	0.85	0.43	0.07	100	0.91	0.83	0.75	0.71	0.67
	200	0.90	0.85	0.79	0.75	0.75	200	0.96	0.94	0.91	0.32	0.13	200	0.94	0.92	0.87	0.85	0.81
	300	0.93	0.87	0.83	0.79	0.76	300	0.97	0.96	0.90	0.57	0.03	300	0.96	0.94	0.92	0.90	0.86
	400	0.93	0.90	0.85	0.82	0.79	400	0.98	0.97	0.95	0.56	0.03	400	0.97	0.95	0.94	0.93	0.88
	500	0.95	0.91	0.87	0.85	0.79	500	0.98	0.97	0.97	0.50	0.07	500	0.97	0.96	0.95	0.94	0.92
QCQP models	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7	$m\backslash n$	3	4	5	6	7
	100	0.90	0.82	0.72	0.69	0.69	100	0.86	0.81	0.70	0.29	0.10	100	0.90	0.82	0.75	0.73	0.70
	200	0.94	0.92	0.86	0.83	0.80	200	0.93	0.88	0.82	0.26	0.07	200	0.95	0.92	0.88	0.85	0.79
	300	0.96	0.93	0.91	0.89	0.84	300	0.95	0.91	0.87	0.47	0.00	300	0.96	0.94	0.92	0.91	0.86
	400	0.97	0.95	0.93	0.90	0.88	400	0.95	0.93	0.88	0.51	0.03	400	0.97	0.95	0.94	0.93	0.89
	500	0.97	0.95	0.94	0.92	0.90	500	0.97	0.93	0.89	0.49	0.03	500	0.98	0.96	0.95	0.94	0.93

are explained by matching types of the synthesized and actual models. For Cube$n$ corresponding indexes in LP and QCQP models are almost equal when $n \leq 5$ and $m \geq 300$, however when $n \geq 6$ they are higher for QCQP models. We hypothesize that it may be difficult for ESOCCS to synthesize several linear constraints of just one variable each, corresponding to facets of hypercube and nonlinearity of QCQP constraints enables synthesis of rounded rectangles approximating corners and facets of this hypercube. Indexes decrease with n. This may be due to *curse of dimensionality* (Bellman [3]) stating that the density of distribution of a fixed-size training set decreases exponentially with n, underdetermining thus the shape of the feasible region. This can be alleviated to some extent by increasing the size of the training set. Training sets of $m = 500$ examples seem sufficient for ESOCCS to reconstruct feasible regions well-enough to score indexes of over 0.65 for problems having five variables or less.

Table 8 shows mean precision, i.e., an empirical probability that a state selected from the feasible region of the synthesized model belongs to the feasible region of the actual model. Precision is higher when the type of the synthesized model matches the type of the actual model except for Cube$n$ for $n \geq 6$ which we attribute to the same reason as for Jaccard index. Precision increases with m and decreases with n which we attribute to the curse of dimensionality.

Table 9 shows mean recall, i.e., an empirical probability that a state selected from the feasible region of the actual model belongs to the feasible region of the synthesized model. Recall is high and over 0.67 for all problems except Simplex$n$ for $n \geq 6$. Number of examples m seems to increase recall.

Table 10: Rice production data set variables and their statistics. Outlying maximums are likely due to noise in the data.

Variable	Meaning	Min	Q1	Median	Q3	Max
A	Cultivated area in ha	0.01	0.14	0.29	0.50	5.32
V	Varieties (0: traditional, 1: mixed, 2: high yield)	0.00	0.00	0.00	2.00	2.00
B	BIMAS intensification program (0: no, 1: mixed, 2: yes)	0.00	0.00	0.00	0.00	2.00
S	Volume of seed in $^{kg}/_{ha}$	4.00	29.48	37.50	48.08	371.43
U	Volume of urea in $^{kg}/_{ha}$	0.87	154.14	214.29	286.87	877.19
Ph	Volume of phosphate in $^{kg}/_{ha}$	0.00	42.80	75.11	128.35	877.19
Pe	Volume of pesticide in $^{kg}/_{ha}$	0.00	0.00	0.00	942.86	43,697.03
L	Total labor in $^{h}/_{ha}$	108.00	699.48	966.53	1,272.55	4,551.72
C	Total cost in $^{\text{INR}}/_{ha}$	20,287.37	69,675.35	123,180.38	223,738.79	4,857,220.17
I	Total income in $^{\text{INR}}/_{ha}$	28,000.00	164,697.18	237,968.94	427,528.69	1,746,987.95

5. Case study: Rice production in India

We apply ESOCCS to model real-world business process of rice production and optimize this process. We employ data set by Feng and Horrace [11] consisting of 1,026 observations of fifteen parameters and three outcomes of production of 171 rice farms in India. We build a Quadratic Programming (QP) model maximizing profit from hectare and constraining parameters of production.

We preprocess the data set by replacing all variables representing prices of the components with a variable representing the total cost of production, and replace the total volume of production and rice selling price with total monetary income. Variables representing production components are absolute and to embrace observations from different farms in a single model, we recalculate them relatively to farm area. We include absolute farm area A in the variable set, since we expect that some parameters of production may depend on it. We encode two ordinal variables using integers reflecting the order. Also, we drop outlying examples with ids 291, 398, 923, 1004, as they are probably wrong. The preprocessed data set of 1,022 examples is available at [to be released after paper acceptance] and its statistics are shown in Table 10.

First, we build a profit objective function as a subtraction of two regression functions $\hat{I}$ and $\hat{C}$ of variables I and C, respectively. We calculate $\hat{I}$ using least-squares quadratic regression. Next, we drop a term with the highest p-value of the t-test for significance of regression coefficient and repeat calculation until all coefficients have p-values < 0.05. This results in Eq. (3) having coefficient of determination $R^2 = 0.79$. The total cost C depends linearly on volumes of production components and labor, however due to varying costs of these components in different examples, we run least-squares linear regression that results in Eq. (4) having $R^2 = 0.98$.

Finally, we synthesize constraints by calling 30 times ESOCCS's setup 400x50x3S with LP term set and parameters from Table 2. The abovementioned objective function and the set of constraints having the lowest fitness

(Eq. (1)) of 0.0008 in all 30 runs form the following QP model:

$$\max \quad \overbrace{-21599A^2-0.6254U^2+93741A+21205V-27075B+}^{\hat{I}} \tag{3}$$
$$+731.1U+779.4Ph+8.539Pe+68.22L$$
$$-$$
$$\underbrace{(-149.5S+49.76U+258Ph+92.76Pe+77.98L)}_{\hat{C}} \tag{4}$$

subject to

$$-87.84V-11.5B+S+0.3381U+0.6441Ph-0.1722Pe-0.3423L\leq 301.3$$
$$75.72A-33.16V-21.48S+U+0.5544Ph+0.01055Pe+0.2889L\leq 495.4$$
$$-6.078V+334.4B+S+0.6648U-2.984Ph+0.002483Pe+0.1672L\leq 1007$$
$$20.87V+37.19B+0.4437S-U+0.5195Ph+0.003986Pe-0.0335L\leq 85.85$$
$$18.85A-38.95V+82.20B+0.2634S+U+0.9Ph+0.014Pe-0.8153L\leq 536.8$$
$$294.8V+71.26B+S+0.2821U-0.3932Ph+0.005128Pe+0.1388L\leq 1248$$
$$A+6.985V-172.9B+0.1739S+0.005549U+0.02464Ph-0.000835Pe+$$
$$-0.004814L\leq 69.63$$
$$-4.498A-27.02V-B+0.07315S+0.02994U+0.04390Ph+0.0006084Pe+$$
$$+0.01309L\leq 65.39$$

To put this model in the context of a specific rice farm, we fix farm area A to median in the data set, i.e., $A = 0.29$. Also, it is reasonable to put upper bound on total cost reflecting a farmer's budget using an extra constraint $\hat{C} \leq c$, where c is the budget. We use three budgets corresponding to 1st quartile, median and 3rd quartile of cost in the data set. Table 11 shows the optimal solutions.

The solutions are reasonable: use of high yield varieties of rice is beneficial, participation in BIMAS program and use of pesticide are not necessary to achieve high profit. Volume of seed decreases with budget while volume of phosphate and labor increase. This may be due to insufficient human resources or volume of phosphate when budget is low, resulting in loss of yield and thus need for more seed planted. Also, higher budget leads to higher return on investment.

We validate these results by comparing the optimal solutions with the most similar examples in the data set. We use Canberra distance (Lance and Williams [19]) to handle different domains of variables. All optimal solutions happen to have the same most similar example, shown in the last column of Table 11. It has four out of eight variables, including farm area A, equal to their values in all optimal solutions. The most striking discrepancy occurs for volume of seed S which is much smaller in the most similar example than in all optimal solutions. This may be because S in all optimal solutions is higher than in 99% of examples and as such may be overestimated. The cost of the most similar example is by 27% higher than the median cost while income is 17% less than income of the optimal solution for the median cost. This result shows that the profit from

Table 11: Optimal solutions to the rice production QP model assuming different budgets, and the most similar actual example w.r.t. Canberra distance. The last row contains distances of the optimal solutions to the most similar one.

Variable	$\hat{C} \leq$ Q1	$\hat{C} \leq$ Median	$\hat{C} \leq$ Q3	Variable	Most similar
A	0.29	0.29	0.29	A	0.29
V	2.00	2.00	2.00	V	2.00
B	0.00	0.00	0.00	B	0.00
S	277.26	264.12	239.43	S	87.41
U	411.51	411.51	411.51	U	349.65
Ph	258.08	392.83	646.08	Ph	174.83
Pe	0.00	0.00	0.00	Pe	0.00
L	308.67	523.84	928.23	L	559.44
$\hat{C}$	69,675.35	123,180.38	223,738.79	C	156,924.48
$\hat{I}$	484,598.66	604,294.20	829,252.37	I	503,496.50
$\hat{I} - \hat{C}$	414,923.31	481,113.82	605,513.58	$I - C$	346,572.02
Distance	1.08	1.00	1.37		

the farm of this example can be improved in terms of both decreasing cost and increasing income.

6. Discussion

Automatic synthesis of Mathematical Programming models is tempting alternative to manual model building. To date, only a little work has been done in automatic synthesis of Linear Programming (LP) and Non-Linear Programming (NLP) models, as reported in Section 3. This is unfortunate because LP and NLP models are very common in practical optimization. Also, one-class model synthesis methods are rare, however actual training data often acquired by monitoring operation of a modeled entity is usually unbalanced with negligible share of infeasible examples. To our knowledge, only GOCCS (Pawlak and Krawiec [24]) shares support for LP/NLP models and one-class data with ESOCCS.

However, the main advantage of ESOCCS over GOCCS is a way of tackling one-class data. Both methods sample space for unlikely feasible examples and then separate such examples from the actually feasible ones. GOCCS samples uniformly the Cartesian product of variables' domains and use adaptive threshold on Canberra distance of a newly sampled example from the closest actual feasible example to determine its class. When the actual feasible region occupies only a small fraction of the Cartesian product of variables' domains, this approach obviously leads to undersampling of the region close to the actual boundary of the feasible region, as unlabeled examples are distributed uniformly. In contrary, ESOCCS estimates the distribution of feasible states and samples this distribution for states having value of probability density function less than an adaptive threshold. This leads to much denser sampling of boundary of the

feasible region, since probability density function attains the highest but less than the threshold values there.

The other features of ESOCCS include parameterized type of synthesized models and guaranteed non-redundancy of constraints in a model. The former is achieved thanks to the representation of constraint as a vector of weights of terms in the given term set (cf. Section 2.2.2). By manipulating the term set, one can achieve virtually any type of constraint. The latter is guaranteed by solving optimally the MILP problem of constraint selection (cf. Section 2.2.3).

Evolutionary Strategy in ESOCCS can be seen as a kind of cooperative co-evolution, where individual constraints cooperate together to compose well-performing model. In this scheme, a single constraint can be used by only a single model, however because each parent constraint breeds λ/μ offspring constraints, groups of $1+\lambda/\mu$ similar constraints are available for model construction. The $(\mu + \lambda)$-survival selection strategy allows models in successive generations to employ the same well-fitted constraint while the other constraints are being augmented. In every generation, models are built in order of descending quality w.r.t. Eq. (1) and the constraints employed by each model are included in the next generation population until it reaches μ constraints. This routine implicitly assesses constraints for their usefulness in forming models and no other measure of constraint quality is required.

We tune ESOCCS on 15 instances of synthetic constraint synthesis benchmarks. The results show that standardization of training set significantly improves performance. We explain this using thought experiment. Consider linear constraints that differ only on a single weight by small ϵ. The angle between them is thus small, however the distance between them increases with distance from the origin of the coordinate system. This also holds for training examples location: for a set of standardized examples, i.e., having mean 0 and standard deviation 1 for all variables, both constraints would separate examples almost equivalently, however when the set of examples is far from the origin, the constraints would separate examples differently. This influences mutation's performance (cf. Section 2.2.2): for non-standardized training set mutation may be unable to fine-tune weights, as even a small change of a weight has substantial impact on separation of examples.

The optimal setups of ESOCCS are 400x50x3S and 200x50x6S when targeting LP and QCQP models, respectively (cf. Section 4.2). 400x50x3S can be considered as an explorative setup given that in every generation it produces three offspring from each of 400 parents and thus possibly searches the space of constraints in 400 different locations. In contrast, 200x50x6S is an exploitative setup by producing six offspring from each of 200 parents and focusing more on fine-tuning of weights. We expect that 400x50x6S works better for LP models, as they are less flexible than QCQP models and thus require less fine-tuning of weights to perform well and the effort can be spent on exploration of alternative models. QCQP models require more fine-tuning offered by 200x50x6S.

The angles between weight vectors of the corresponding synthesized and actual constraints are relatively small when types of the constraints match (cf. Table 6). This is promising result, as it suggests that ESOCCS constructs

syntactically similar constraints to the actual ones. However, the numbers of synthesized constraints and terms are significantly different from the actual for almost all benchmarks instances (cf. Table 5). Overall, ESOCCS tends to produce bigger constraints than actual and their sizes correlate with size of the term set. This is possible drawback of ESOCCS, as the user should provide a small term set to produce concise models. Comparison of these results with results of GOCCS on the same benchmarks (Pawlak and Krawiec [24]) shows that although GOCCS synthesizes models of size closer to actual, the angles between the corresponding constraints are much higher than for ESOCCS. Apart from syntax, ESOCCS is quite successful in reconstructing boundary of the feasible region in terms of Jaccard index of the feasible regions, test-set precision and recall (cf. Tables 7–9). Although these statistics decrease with the number of variables which we attribute to the curse of dimensionality (Bellman [3]), they increase with the number of training examples and large-enough training set may alleviate negative impact of dimensionality. Higher recall (Table 9) than the corresponding precision (Table 8) shows that ESOCCS may have tendency to overestimate the feasible region: for all problems and most combinations their parameters the synthesized feasible region seems to include majority of the actual feasible region (high recall) and a part of the actual infeasible region (lower precision). However, higher outlier cut-off percentile ρ may tighten the feasible region.

Precise reconstruction of boundary of the feasible region is crucial from the perspective of optimization of the synthesized model, since optimal solutions usually lie on this boundary. In Section 5, we assess ESOCCS in scenario where the synthesized model of a real-world process of rice production is optimized. Given no actual objective function, we build a one using regression, thus all parts of the model are built automatically from data. We validate this model in three cases of different budgets for production by applying an extra constraint on production cost and then compare each of the optimal solutions to the most similar example in the data set. The results show that (a) extending a synthesized model to meet changing requirements is easy, (b) the optimal solutions are quite similar to the existing examples, and (c) the modeled business process can be optimized using the automatically synthesized model. Result (a) reveals possible use case scenario of computer-assisted modeling, in which an expert runs ESOCCS to acquire a base model of the process and then augment this model manually using domain knowledge. This may reduce overall time required to build a correct model. Concerning (b), the tendency of ESOCCS to overestimate feasible region seems low enough to keep the optimal solutions close to the actual data and thus likely feasible in practice. In turn, (c) shows that entire workflow of modeling and optimization of a business process can be conducted automatically at minimal costs leading to measurable profit.

7. Conclusions and future work

We propose an Evolutionary Strategy-based One Class Constraint Synthesis (ESOCCS), a tool for computer-assisted modeling of real-world entities using

Mathematical Programming (MP) models. ESOCCS combines unique features of fully parameterized type of the produced models and support for one-class training data with competitive quality of the produced models. We analyze ESOCCS on a suite of synthetic benchmarks and observe high fidelity of syntax and semantics of the constructed models to the ground truth models. We also apply ESOCCS in a real-world scenario of fully automated optimization of rice production business process, in which ESOCCS builds an MP model from data and an off-shelf solver finds using this model the optimal parameters of production. We validate these parameters with existing examples in the data set and conclude their feasibility in practice and likely increase of production profit by applying them.

ESOCCS, however, is not free of challenges and further research on improving its characteristics is required. ESOCCS produces constraints which size depends on cardinality of the supplied term set, and a way to break this dependency has to be found, as an ideal constraint synthesis method should adapt constraints to the characteristics of the data instead of method's parameters. ESOCCS in the essence produces conjunction of constraints and adding support for alternative constraints may be beneficial in many real-world scenarios. Also, alternative ways to estimate distribution of feasible states, such as Estimation of Distribution Algorithms (EDA) deserve investigation.

Funding. This work was funded by National Science Centre, Poland, grant no. 2016/23/D/ST6/03735.

[1] Aswal, A., Prasanna, G. N. S., 2010. Estimating correlated constraint boundaries from timeseries data: The multi-dimensional german tank problem. In: EURO 2010. `http://slideplayer.com/slide/7976536/`.

[2] Beldiceanu, N., Simonis, H., Oct. 2012. A model seeker: Extracting global constraint models from positive examples. In: CP'12. Vol. LNCS 7514. Springer, Quebec City, Canada, pp. 141–157.

[3] Bellman, R., 2013. Dynamic Programming. Dover Books on Computer Science. Dover Publications.
URL `https://books.google.it/books?id=CG7CAgAAQBAJ`

[4] Bessiere, C., Coletta, R., Hebrard, E., Katsirelos, G., Lazaar, N., Narodytska, N., Quimper, C.-G., Walsh, T., 2013. Constraint acquisition via partial queries. In: IJCAI 2013. pp. 475–481.

[5] Bessiere, C., Coletta, R., Koriche, F., O'Sullivan, B., 2005. A SAT-Based Version Space Algorithm for Acquiring Constraint Satisfaction Problems. Springer Berlin Heidelberg, pp. 23–34.
URL `http://dx.doi.org/10.1007/11564096_8`

[6] Bessiere, C., Coletta, R., O'Sullivan, B., Paulin, M., 6-12 Jan. 2007. Query-driven constraint acquisition. In: IJCAI 2007. pp. 50–55.

[7] Beyer, H.-G., Schwefel, H.-P., 2002. Evolution strategies – a comprehensive introduction. Natural Computing 1 (1), 3–52.
URL http://dx.doi.org/10.1023/A:1015059928466

[8] Dempster, A. P., Laird, N. M., Rubin, D. B., 1977. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological) 39 (1), 1–38.
URL http://www.jstor.org/stable/2984875

[9] Denis, F., Gilleron, R., Letouzey, F., 2005. Learning from positive and unlabeled examples. Theoretical Computer Science 348 (1), 70 – 83.
URL http://www.sciencedirect.com/science/article/pii/S0304397505005256

[10] Eiben, A. E., Smith, J. E., 2003. Introduction to Evolutionary Computing. Springer.
URL http://www.cs.vu.nl/~gusz/ecbook/ecbook.html

[11] Feng, Q., Horrace, W. C., 2012. Alternative technical efficiency measures: Skew, bias and scale. Journal of Applied Econometrics 27 (2), 253–268.

[12] Gurobi Optimization, Inc., 2015. Gurobi optimizer reference manual.
URL http://www.gurobi.com

[13] Hernández-Lobato, J. M., Gelbart, M. A., Adams, R. P., Hoffman, M. W., Ghahramani, Z., Jan. 2016. A general framework for constrained bayesian optimization using information-based search. J. Mach. Learn. Res. 17 (1), 5549–5601.
URL http://dl.acm.org/citation.cfm?id=2946645.3053442

[14] Hothorn, T., Hornik, K., van de Wiel, M. A., Zeileis, A., 2015. Package 'coin': Conditional inference procedures in a permutation test framework.
URL http://cran.r-project.org/web/packages/coin/coin.pdf

[15] Jaccard, P., 1912. The distribution of the flora in the alpine zone. New Phytologist 11 (2), 37–50.

[16] Kanji, G., 1999. 100 Statistical Tests. SAGE Publications.

[17] Karp, R., 1972. Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (Eds.), Complexity of Computer Computations. Plenum Press, pp. 85–103.

[18] Kolb, S., 2016. Learning constraints and optimization criteria. In: AAAI Workshops.

[19] Lance, G. N., Williams, W. T., 1967. Mixed-data classificatory programs i - agglomerative systems. Australian Computer Journal 1 (1), 15–20.
URL http://dblp.uni-trier.de/db/journals/acj/acj1.html#LanceW67

[20] Mayoh, B., Tyugu, E., Penjam, J., 2013. Constraint Programming. Nato ASI Subseries F:. Springer.

[21] Montana, D. J., 1995. Strongly typed genetic programming. Evolutionary Computation 3 (2), 199–230.
URL http://vishnu.bbn.com/papers/stgp.pdf

[22] Pawlak, T. P., Krawiec, K., 2017. Automatic synthesis of constraints from examples using mixed integer linear programming. European Journal of Operational Research 261 (3), 1141 – 1157.
URL http://www.sciencedirect.com/science/article/pii/S037722171730156X

[23] Pawlak, T. P., Krawiec, K., 19-21 Apr. 2017. Synthesis of mathematical programming constraints with genetic programming. In: Castelli, M., McDermott, J., Sekanina, L. (Eds.), EuroGP 2017: Proceedings of the 20th European Conference on Genetic Programming. Vol. 10196 of LNCS. Springer Verlag, Amsterdam, pp. 178–193.

[24] Pawlak, T. P., Krawiec, K., 2017. Synthesis of mathematical programming models with one-class genetic programming. IEEE Transactions on Evolutionary Computation(in review).

[25] Quinlan, J. R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[26] Souza, C. R., May 2017. The Accord.NET Framework.
URL http://accord-framework.net

[27] Tax, D. M. J., 2001. One-class classification: Concept-learning in the absence of counter-examples. Ph.D. thesis, Delft University of Technology.

[28] Teso, S., Sebastiani, R., Passerini, A., 2017. Structured learning modulo theories. Artificial Intelligence 244, 166 – 187.