



Efficient learning of decision-making models: A penalty block coordinate descent algorithm for data-driven inverse optimization

Rishabh Gupta, Qi Zhang *

Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455, USA

ARTICLE INFO

Keywords:

Data-driven inverse optimization
Statistical learning
Bilevel optimization
Block coordinate descent

ABSTRACT

Decision-making problems are commonly formulated as optimization problems, which are then solved to make optimal decisions. In this work, we consider the inverse problem where we use prior decision data to uncover the underlying decision-making process in the form of a mathematical optimization model. This statistical learning problem is referred to as data-driven inverse optimization. We focus on problems where the underlying decision-making process is modeled as a convex optimization problem whose parameters are unknown. We formulate the inverse optimization problem as a bilevel program and propose an efficient block coordinate descent-based algorithm to solve large problem instances. Numerical experiments on synthetic datasets demonstrate the computational advantage of our method compared to standard commercial solvers. Moreover, the real-world utility of the proposed approach is highlighted through two realistic case studies in which we consider estimating risk preferences and learning local constraint parameters of agents in a multiplayer Nash bargaining game.

1. Introduction

Decision making is fundamental to everyday life. As humans, we constantly make decisions that involve balancing different trade-offs to achieve the best outcome. These decisions can be as mundane as choosing what to eat for lunch, or as intricate as designing and operating a chemical plant. Also, humans are not the only ones making decisions; other biological entities, such as animals, microorganisms, and even individual cells, are often considered intelligent and autonomous agents capable of making decisions (McFarland, 2004; Balázs et al., 2011). Furthermore, with increased automation and the emergence of artificial intelligence, many engineered systems can also be viewed as decision-making agents (Steels, 1995).

A good understanding of decision-making mechanisms is crucial for predicting the behavior of autonomous agents, learning from experts, and optimizing systems involving various decision makers. But many decision-making processes are unknown or poorly understood. For example, experts in the operation of chemical plants make decisions based on many years of experience, but their decision strategies often are not well documented and, due to the complexity of the manufacturing processes, are difficult to explain even to fellow operators. As a result, the complete transfer of expert knowledge to new operators remains an unsolved problem. Likewise in microbiology, cells can be considered autonomous agents that make decisions regarding gene expression and cell metabolic function. While we can observe these

decisions in experiments, we often do not understand why cells make these choices. Answering this question would provide fundamental insights that could advance cancer treatment, immunology research, and biomanufacturing operations. In both examples, we do not know how exactly decisions are made, but we can observe those decisions. The question is: Can we use these observations to uncover the underlying decision-making process?

The above question is, of course, not new. In particular, the first motivating example may remind some readers of expert systems, a subdomain of artificial intelligence that had spurred much enthusiasm in various disciplines, including chemical engineering, in the 1980s (Bañares-Alcántara et al., 1985; Rich and Venkatasubramanian, 1987; Stephanopoulos, 1990). While research in this field has primarily focused on using human experience and domain knowledge to build expert systems, there have been a few attempts to learn an expert system from data (Rich and Venkatasubramanian, 1989; Sammut et al., 1992), most of which are based on decision tree learning. More recently, in machine learning, this problem has been referred to as apprenticeship or imitation learning (Abbeel and Ng, 2004), with popular applications in robotics and autonomous driving. In theory, any machine learning method can be applied to this problem, but it is unclear what model is the most suitable. Popular black-box machine learning models, such as artificial neural networks, often require a

* Corresponding author.

E-mail address: qizh@umn.edu (Q. Zhang).

large amount of data to train and, more importantly, are difficult to interpret. Interpretability, however, is crucial in this problem as we are chiefly interested in gaining a better understanding of the underlying decision-making process.

In this work, we present an approach that is fundamentally inspired by the *principle of optimality* (Schoemaker, 1991), which conjectures that autonomous agents generally make decisions in some optimal fashion. This basic relationship between decision making and optimization is commonly applied in many fields. For example, most economic theory relies on the assumption that rational decision makers are utility maximizers (Morgenstern and von Neumann, 1944; Hey and Orme, 1994). Evolutionary biology tells us that through adaptations over the course of millions of years, biological systems have evolved to behave optimally, albeit the measure of optimality and the decision set often being unclear (Rosen, 1967; Parker and Smith, 1990). Even the fundamental basis for thermodynamics is an optimization problem, namely Gibbs free energy minimization (Gautam and Seider, 1979; Rossi et al., 2009). Following this principle of optimality, the key idea is to model a decision-making process as a mathematical optimization problem of the following general form:

$$\begin{aligned} & \underset{x}{\text{minimize}} && f(x, u) \\ & \text{subject to} && g(x, u) \leq 0, \end{aligned}$$

where x and u denote the vectors of decision variables and input parameters, respectively. Given input parameters u , the decision maker chooses x such that their objective function $f(x, u)$ is minimized while satisfying the constraints $g(x, u) \leq 0$. We then consider the inverse problem, which is to infer the functions f and g given observations, with each data point i corresponding to an input-decision pair (u_i, x_i) . The resulting estimated model will be a natural representation of a decision-making process that is inherently interpretable. Moreover, this approach allows us to take advantage of all the modeling flexibility provided by mathematical optimization and directly incorporate domain knowledge in the form of constraints. In the operations research literature, the inverse problem described above is referred to as *inverse optimization* (IO) (Ahuja and Orlin, 2001). We present a review of the relevant IO literature in the next section.

To better understand when IO will produce a more useful model than a black-box machine learning method, consider a decision maker who is responsible for routing traffic on a road network. Our goal is to understand the decision-making strategy of this decision maker because they are considered an expert. When using IO, the constraints on the decision maker, which represent the structure of the network and its capacities, can be easily formulated. Through our approach, learning the objective function of the decision maker will uniquely allow us to understand the preferences of the expert decision maker. This information can be used to transfer the expert's knowledge to other novice operators, which can be particularly useful in situations where the expert is unavailable or when their expertise needs to be shared with a larger group of decision makers. It is generally not easy to extract such insights from black-box machine learning models.

2. Literature review

The notion of inverse optimization was first introduced by Burton and Toint (1992) who considered the problem of recovering arc costs on a directed graph using given shortest-path solutions. Early works have primarily addressed the deterministic setting in which observations are assumed to be exactly optimal solutions of the optimization model, with the majority of existing works only considering one single observation (Zhang et al., 1996; Zhang and Liu, 1996; Ahuja and Orlin, 2001; Heuberger, 2004; Iyengar and Kang, 2005).

IO received relatively little attention in the 1990s and 2000s. However, there has been renewed interest in recent years when the research focus has shifted towards the case with multiple observations, also referred to as *data-driven* IO (Mohajerin Esfahani et al., 2018), and

noisy data (Chan et al., 2014, 2019; Keshavarz et al., 2011; Bertsimas et al., 2015; Aswani et al., 2018). This new perspective has significantly broadened the settings to which IO can be applied and it has found application in a diverse range of fields, including healthcare planning (Chan et al., 2014, 2022), transportation network design (Chow et al., 2014; Rönnqvist et al., 2017), electricity markets (Saez-Gallego et al., 2016; Birge et al., 2017), biological network inference (Burgard and Maranas, 2003; Ugun et al., 2007; Zhao et al., 2015), and expert systems (Akhtar et al., 2022); a more detailed description of existing IO applications can be found in Chan et al. (2021).

While initially designed to determine an objective function that renders a single given solution optimal, the recent research in data-driven IO has helped it gain increased acceptance as a statistical learning method (Iraj and Terekhov, 2021). Nevertheless, it has received virtually no attention from the chemical engineering or, more specifically, the process systems engineering (PSE) community. We could only find a handful of articles where IO (or similar) methods have been used to infer parameters of an optimization problem in PSE (Burgard and Maranas, 2003; Ugun et al., 2007; Bollas et al., 2009; Glass et al., 2017; Glass and Mitsos, 2018). We believe that IO can potentially be very effective in PSE applications as it provides a compelling framework for the integration of optimization, machine learning, and domain knowledge. This is in the same spirit as other hybrid modeling methods that incorporate first-principles knowledge into otherwise purely data-driven approaches, which is crucial in many chemical engineering applications (Boukouvala and Floudas, 2017; Wilson and Sahinidis, 2019; Bangi and Kwon, 2020).

From a theoretical perspective, research on data-driven IO is largely limited to the case where the decision-making process is being modeled as a convex optimization problem. Here, the main distinction among the various proposed formulations is in terms of the loss function employed to fit the data. Minimization of the slack required to make the noisy data satisfy an optimality condition is considered in Boyd et al. (2011), Bertsimas et al. (2015), and Mohajerin Esfahani et al. (2018). Aswani et al. (2018) showed that this kind of loss function can lead to statistically inconsistent estimates and proposed minimizing the sum of some norm of residuals with respect to the decision variables.

A key challenge in using the statistically consistent bilevel IO problem formulation proposed by Aswani et al. (2018) is its computational intractability. This has restricted the use of IO to very specific problems for which efficient solution methods have been developed. Almost all existing literature addresses inverse linear optimization problems (Babier et al., 2021; Shahmoradi and Lee, 2022; Gupta and Zhang, 2022). Additionally, most of these studies assume the constraints to be known and only attempt to estimate the cost coefficients. There are very few existing contributions that consider joint estimation of objective and constraint parameters (Chan and Kaw, 2019; Ghobadi and Mahmoudzadeh, 2020). Hence, there is a need for more efficient algorithms that can infer all unknown parameters for high-dimensional convex nonlinear problems with large datasets. This work aims to fill this apparent deficiency in the literature.

3. Outline and contributions

In this work, we consider data-driven IO with noisy data, and we specifically focus on the case where the optimization model describing the decision-making process can be assumed to be convex. Here, one major challenge is the computational complexity of the IO problem as it gives rise to a bilevel optimization problem whose size increases with the number of data points. To solve large model instances, we develop a penalty block coordinate descent (BCD) algorithm that exploits the specific decomposable structure of the problem. The efficacy of the proposed algorithm is demonstrated through a comprehensive set of computational experiments. Moreover, importantly, the computational results show empirically that the proposed data-driven IO method is statistically consistent and data-efficient, highlighting its promise for future, more complex applications.

In summary, the key contributions of this work are as follows:

1. We propose a data-driven framework to simultaneously learn the objective function and constraint parameters of a convex nonlinear optimization problem based on multiple noisy observations of its optimal solutions for different input parameter values.
2. We formulate the data-driven IO problem as a bilevel program, which we further reformulate into a single-level optimization problem using the Karush–Kuhn–Tucker (KKT) optimality conditions. In general, the resulting problem lacks regularization. We alleviate this deficiency by proposing an exact penalty reformulation.
3. For certain important classes of functions f and g , we show that the penalty reformulation of the IO problem is a multiconvex optimization problem. In this case, we show that large instances of the reformulated IO problem can be solved very efficiently by exploiting their decomposable structure with the BCD algorithm.
4. By applying known results from the literature, we show that the proposed solution method is guaranteed to converge to a stationary point of the reformulated IO problem. We further demonstrate the effectiveness of our method through computational experiments based on a number of instances of a convex optimization problem from the wireless communication literature. Our results indicate that a BCD-based decomposition strategy is significantly faster and finds higher-quality solutions than standard nonlinear optimization solvers.
5. We demonstrate the real-world utility of IO by considering two realistic example problems. In the first problem, we use IO to estimate a decision maker's risk preference when making decisions under significant uncertainty. In the second example, we use IO to estimate customers' internal constraints under the assumption that their buying decisions are covered by a collective bargaining agreement.

The remainder of this paper is organized as follows. In Section 4, we present the mathematical formulation of the IO problem that we consider in this work. The penalty BCD algorithm is developed in Section 5, and the results from the computational case studies are presented in Section 6. Finally, we close with some concluding remarks in Section 7.

4. The inverse optimization problem

In this section, we present the mathematical formulation of the IO problem that we consider in this work. We describe its key properties and introduce a set of assumptions that will be useful in the next section where we present our solution strategy for the problem.

We start by assuming that the given decision-making process can be modeled as a convex optimization problem of the following form:

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} && f(x, u; \theta) \\ & \text{subject to} && g_k(x, u; \omega) \leq 0 \quad \forall k \in \{1, \dots, m\} \\ & && h_k(x, u; \omega) = 0 \quad \forall k \in \{1, \dots, p\}, \end{aligned} \quad (\text{FOP})$$

where x is the vector of decision variables and u is the vector of input variables. The function f is the objective function of the problem, and g_k and h_k are the constraint functions. The model given by f , g_k , and h_k , which we refer to as the forward optimization problem (FOP), is parameterized by two sets of model parameters, θ and ω . To ease the notation, we denote the constraint functions $\{g_k\}_{k=1}^m$ and $\{h_k\}_{k=1}^p$ by function vectors g and h , respectively, in the remainder of this paper.

To ensure convexity of (FOP), we require the functions g and h to be convex and affine in x , respectively, for fixed u and ω . We also make the following assumptions.

Assumption 1. The functions $f(x, u; \theta)$ and $g(x, u; \omega)$ are twice continuously differentiable in x for fixed u , θ , and ω .

Assumption 2. The function $f(x, u; \theta)$ is strictly convex in x for fixed u and θ .

Assumption 1 is fairly standard in the IO literature (Boyd et al., 2011; Bertsimas et al., 2015; Mohajerin Esfahani et al., 2018) and is required to ensure that the bilevel IO problem (introduced later) can be reformulated as a single-level problem. There are two main arguments for Assumption 2:

1. Given a dataset consisting of (u_i, x_i) observations, it is likely that there will be multiple optimization models that can explain the data. In these cases, it may be better to learn a strictly convex model that, when used for predictions, results in unique decisions for every input. This can help avoid ambiguities and ensure that the model makes clear, distinct decisions.
2. There may still be situations in which the user wants to learn parameters of a problem that, generally, does not have unique solutions. However, as shown in our previous work (Gupta and Zhang, 2022), methods developed for estimating parameters of generally convex FOPs do not provide good solutions when the FOPs can admit multiple solutions. Therefore, in this case, we suggest adding a small regularizing quadratic term to the objective function of the problem to make it strictly convex.

The goal of IO is to recover model (FOP) from observations that are assumed to be noisy estimates of the optimal solutions to (FOP) for given values of the input variables. Specifically, the problem we aim to solve is the following (Aswani et al., 2018):

$$\begin{aligned} & \underset{\theta \in \Theta, \omega \in \Omega, \hat{x}}{\text{minimize}} && \sum_{i \in I} (x_i - \hat{x}_i)^\top W (x_i - \hat{x}_i) \\ & \text{subject to} && \hat{x}_i = \arg \min_{\tilde{x} \in \mathbb{R}^n} \{f(\tilde{x}, u_i; \hat{\theta}) : g(\tilde{x}, u_i; \hat{\omega}) \leq 0, h(\tilde{x}, u_i; \hat{\omega}) = 0\} \\ & && \forall i \in I, \end{aligned} \quad (\text{IOP})$$

where I is the set of experiments, with each experiment i defined by given inputs u_i and the corresponding observed decisions x_i . The objective is to choose the parameters $\hat{\theta}$ and $\hat{\omega}$ from sets Θ and Ω , respectively, such that a weighted sum of squared residuals, where $W \in \mathbb{R}^{n \times n}$ denotes the diagonal matrix of weighting factors, is minimized.

We make the following assumption about sets Θ and Ω , which is required to ensure statistical consistency of the estimates $\hat{\theta}$ and $\hat{\omega}$ obtained from (IOP) (Aswani et al., 2018):

Assumption 3. The sets Θ and Ω are compact.

In addition, we require a regularity assumption on these sets to ensure the convergence of our solution algorithm for (IOP).

Assumption 4. The description of sets Θ and Ω satisfies necessary regularity conditions such as Mangasarian-Fromovitz constraint qualification (MFCQ).

Finally, we highlight an implicit assumption while formulating an IO problem where we assume that the set of possible objective parameters Θ disallows the trivial solution for θ that would render the objective function a constant.

Problem (IOP) is similar to how the IO problem has typically been formulated in the literature with the only difference being the weighting factors that offer the following flexibilities while formulating the IO problem:

1. Weights can be used to reflect the level of importance of each variable. In data reconciliation, the weight for each variable is the reciprocal of the variance of the corresponding measurement, which reflects the accuracy of the measurement. This can be considered using our general matrix of weighting factors.
2. Weighting factors allow us to deal with problems where different decision variables have vastly different scales.
3. It also generalizes the case with unmeasured variables, for which the weighting factors can be set to zero.

Finally, we close this section by formally stating a mild assumption under which (IOP) is always feasible.

Assumption 5. The sets $C_i := \{x : g(x, u_i; \omega) < 0, h(x, u_i; \omega) = 0\} \forall i \in I$ are nonempty for every $\omega \in \Omega$, and f remains bounded from below for every $\theta \in \Theta$.

5. Solution strategy

5.1. Single-level reformulation

Problem (IOP) is a bilevel optimization problem with convex lower-level problems. This class of problems are typically solved by reformulating them into a single-level optimization problem by replacing the lower-level problems with their optimality conditions (Dempe et al., 2015). Here, given Assumption 5, the lower-level problems of (IOP) can be replaced with their KKT conditions, resulting in the following reformulation of (IOP):

$$\begin{aligned} & \underset{\hat{\theta} \in \Theta, \hat{\omega} \in \Omega, \hat{\lambda}, \hat{\mu}}{\text{minimize}} && \sum_{i \in I} (x_i - \hat{x}_i)^T W(x_i - \hat{x}_i) \\ & \text{subject to} && \nabla f(\hat{x}_i, u_i; \hat{\theta}) + \lambda_i^T \nabla g(\hat{x}_i, u_i; \hat{\omega}) + \mu_i^T \nabla h(\hat{x}_i, u_i; \hat{\omega}) = 0 \end{aligned} \quad (1a)$$

$$\forall i \in I \quad (1b)$$

$$g(\hat{x}_i, u_i; \hat{\omega}) \leq 0 \quad \forall i \in I \quad (1c)$$

$$h(\hat{x}_i, u_i; \hat{\omega}) = 0 \quad \forall i \in I \quad (1d)$$

$$\lambda_{ik} g_k(\hat{x}_i, u_i; \hat{\omega}) = 0 \quad \forall i \in I, k \in \mathcal{K} \quad (1e)$$

$$\lambda_i \geq 0, \hat{x}_i \in \mathbb{R}^n \quad \forall i \in I, \quad (1f)$$

where λ and μ are respectively the dual variables of the inequality and equality constraints of the lower-level problems of (IOP). Constraints (1b), (1c)–(1d), and (1e) correspond to the stationarity, primal feasibility, and complementary slackness conditions, respectively.

Problem (1) is generally a nonconvex nonlinear program (NLP) that is difficult to solve to global optimality. Moreover, the size of (1) increases with the number of data points, which quickly leads to very large instances in problems of practical relevance. In the machine learning literature, given a large nonconvex optimization problem, it is typical to focus on finding a good local solution efficiently rather than trying to solve the problem to global optimality. A popular example is deep learning, which is a nonconvex optimization problem; however, several local solution methods have been found to result in very accurate predictive models (Choromanska et al., 2015; Jin et al., 2021; Danilova et al., 2022). In the following, we pursue a similar strategy in tackling the computational complexity of problem (1).

5.2. Penalty reformulation

Problem (1) lacks sufficient regularity that is generally required to solve such problems with standard NLP methods. It contains complementarity constraints (1e), which make the problem violate the MFCQ everywhere in the feasible region (Scheel and Scholtes, 2000; Anitescu, 2005). Moreover, the unknown FOP parameters θ and ω complicate the form of the other constraints in (1) that may, in some cases, also contribute towards the lack of regularization.

The aforementioned lack of regularization of (1) is known to result in degenerate and unbounded Lagrange multipliers in the vicinity of the optimal solutions, causing convergence difficulties with NLP solution algorithms. A popular strategy to overcome this difficulty employs a penalty reformulation (Bertsekas, 1997; Nocedal and Wright, 2006) of the original problem (Anitescu, 2000). In this approach, the complicating constraints, i.e., the constraints resulting in constraint qualification violation are moved to the objective function as penalty terms, leaving the problem with a “regular” feasible region. Depending on the type of penalty function, the reformulation can be exact in the sense that, for

certain finite values of the penalty parameters, every optimal solution of the original problem will also be optimal for the reformulation. Commonly employed examples of penalty functions that result in these exact reformulations are nonsmooth penalty functions, such as the one based on the ℓ_1 -norm of constraint violation (Bertsekas, 1997; Nocedal and Wright, 2006). These have proven to be very successful in dealing with the challenges associated with difficult NLPs such as mathematical programs with complementarity constraints (MPCCs) (Benson et al., 2006), which motivates us to consider the following reformulation of (1):

$$\begin{aligned} & \underset{\hat{\theta} \in \Theta, \hat{\omega} \in \Omega, \hat{\lambda}, \hat{\mu}}{\text{minimize}} && \sum_{i \in I} (x_i - \hat{x}_i)^T W(x_i - \hat{x}_i) \\ & && + c^T \underbrace{\begin{bmatrix} \sum_{i \in I} |\nabla f(\hat{x}_i, u_i; \hat{\theta}) + \lambda_i^T \nabla g(\hat{x}_i, u_i; \hat{\omega}) + \mu_i^T \nabla h(\hat{x}_i, u_i; \hat{\omega})| \\ \sum_{i \in I} \max\{0, \hat{g}(\hat{x}_i, u_i; \hat{\omega})\} \\ \sum_{i \in I} |\hat{h}(\hat{x}_i, u_i; \hat{\omega})| \\ \sum_{i \in I} |\lambda_i^T \hat{g}(\hat{x}_i, u_i; \hat{\omega})| \end{bmatrix}}_p \end{aligned} \quad (2)$$

subject to $\lambda_i \geq 0, \hat{x}_i \in \mathbb{R}^n \quad \forall i \in I$

where c are the positive penalty parameters. Next, we present our strategy to find a minimizer for (1) through (2).

Lemma 1. *The feasible region of problem (2) satisfies MFCQ.*

Proof. By Assumption 4, the constraints describing the sets Θ and Ω satisfy MFCQ. Moreover, in (2), the only remaining constraints are the non-negativity constraints on the dual variables λ which satisfy linear independence constraint qualification (LICQ) and therefore MFCQ. \square

In (2), we move all but a few simple bound constraints of (1) to the objective function. However, note that our goal here is to only eliminate the constraints that contribute to the lack of regularity of (1). Therefore, while formulating (2), one can choose to move only those constraints for which constraint qualification violation is expected. Since, in practice, it is often not trivial to check constraint qualification violation, we use the following reformulation strategy: We always penalize all the stationarity, complementarity, and primal feasibility constraints for which ω are unknown. For the other primal constraints with known ω , we leave them as hard constraints if it is possible to see that they do not violate constraint qualification. For example, if the remaining primal constraints are simple bound constraints that obviously satisfy LICQ, then we do not need to penalize them.

Lemma 1 essentially highlights the well-posedness of (2) which makes it amenable to solution via any commercial NLP solver, such as IPOPT (Wächter and Biegler, 2006). We exploit this property of (2) to design an algorithm in which the solution of (1) can be obtained by repeatedly solving (2). The strategy employed in Algorithm 1 is to start with some initial values for the penalty parameters c and successively increase them by a factor ρ after every iteration to find the point where a feasible solution to (1) is obtained. The global convergence of this approach for a general NLP has been proved by Benson et al. (2007).

Algorithm 1 Algorithm for solving (1) through (2).

- 1: initialize: $k \leftarrow 1, (\hat{\theta}, \hat{\omega}, \hat{x}, \lambda, \mu) \leftarrow (\hat{\theta}_0, \hat{\omega}_0, \hat{x}_0, \lambda_0, \mu_0)$ and $c \leftarrow c_1$
- 2: **while** $\|P\| > \epsilon$ **do**
- 3: solve (2) with $c = c_k$, warm-start with $(\hat{\theta}_{k-1}, \hat{\omega}_{k-1}, \hat{x}_{k-1}, \lambda_{k-1}, \mu_{k-1})$, obtain $(\hat{\theta}_k, \hat{\omega}_k, \hat{x}_k, \lambda_k, \mu_k)$
- 4: $c_{k+1} \leftarrow c_k + \rho c_k$
- 5: $k \leftarrow k + 1$
- 6: **end while**
- 7: **return** $(\hat{\theta}_k, \hat{\omega}_k, \hat{x}_k, \lambda_k, \mu_k)$

Assumption 6. Suppose $(\hat{\theta}_k, \hat{\omega}_k, \hat{x}_k, \lambda_k, \mu_k)$ denotes a solution of (2) with $c = c_k$. The sequence of solutions $\{\hat{\theta}_k, \hat{\omega}_k, \hat{x}_k, \lambda_k, \mu_k\}$ generated as c_k increase to ∞ is bounded.

Theorem 1 (Benson et al., 2007). Suppose Assumption 6 holds. If $(\hat{\theta}^*, \hat{\omega}^*, \hat{x}^*, \lambda^*, \mu^*)$ is the accumulation point of the sequence of solutions generated as $k \rightarrow \infty$ and c_k increase to ∞ , then $(\hat{\theta}^*, \hat{\omega}^*, \hat{x}^*, \lambda^*, \mu^*)$ minimizes (1).

5.3. Block coordinate descent

We introduced the penalty reformulation as a means to tackle the lack of regularity of (2). However, the resulting “well-posed” problem is still a nonconvex NLP whose size increases with the number of data points. In this section, we propose a decomposition scheme designed to significantly reduce the computation time when solving certain large instances of (2).

Our approach is motivated by the fact that for many FOPs, (1) becomes a *multiconvex optimization problem* (MCP) that exhibits a particular decomposable structure. MCPs are problems for which the variables can be partitioned into blocks over which it is convex when all other variables are held constant (Shen et al., 2017). Such FOPs include, for example, the broad class of quadratic programs (QPs), which are very common in engineering applications (McCarl et al., 1977). Next, we demonstrate the MCP structure of the penalty reformulation by deriving (2) for the case when (FOP) is a QP.

Example 1. Suppose that (FOP) can be posed as the following convex QP:

$$\begin{aligned} & \text{minimize}_{x \in \mathbb{R}^n} \quad \frac{1}{2} x^\top Q x + r^\top x \\ & \text{subject to} \quad A x \leq b, \end{aligned} \quad (3)$$

where $Q \succ 0$. We assume that Q and r can be parameterized with θ , and A and b can be parameterized with ω . Also, Q , r , A , and b generally change with the input parameters u . Problem (2) for this FOP can be stated as follows:

$$\begin{aligned} & \text{minimize}_{\hat{\theta} \in \Theta, \hat{\omega} \in \Omega, \hat{x}, \lambda} \quad \sum_{i \in I} (x_i - \hat{x}_i)^\top W (x_i - \hat{x}_i) \\ & \quad + c^\top \left[\sum_{i \in I} |Q(u_i; \hat{\theta}) \hat{x}_i + r(u_i; \hat{\theta}) + \lambda_i^\top A(u_i; \hat{\omega})| \right. \\ & \quad \left. + \sum_{i \in I} |A(u_i; \hat{\omega}) \hat{x}_i - b(u_i; \hat{\omega})| \right. \\ & \quad \left. + \sum_{i \in I} |\lambda_i^\top (A(u_i; \hat{\omega}) \hat{x}_i - b(u_i; \hat{\omega}))| \right] \end{aligned} \quad (4)$$

$$\text{subject to} \quad \lambda_i \geq 0, \hat{x}_i \in \mathbb{R}^n \quad \forall i \in I.$$

Assuming that Θ and Ω are convex, this problem is an MCP for the following block decomposition of its variables: $((\hat{\theta}, \lambda), \hat{\omega}, \hat{x})$.

MCPs are nonconvex problems and hence generally very hard to solve globally; however, several important problems in machine learning fall into this class and have led to the development of a number of approximate solution methods (Shen et al., 2017; Jain and Kar, 2017). Most of these methods employ some variation of the *block coordinate descent* (BCD) approach which exploits the particular mathematical structure of subproblems obtained by fixing blocks of variables.

Assume that the MCP instances of (2) can be written as follows:

$$\begin{aligned} & \text{minimize}_{x \in \mathbb{R}^n} \quad f(x_1, \dots, x_\ell) \\ & \text{subject to} \quad x_i \in \mathcal{X}_i \quad \forall i \in \{1, \dots, \ell\}, \end{aligned} \quad (5)$$

where the variables $x \in \mathbb{R}^n$ have been partitioned into ℓ mutually exclusive, collectively exhaustive sets $\{x_1, \dots, x_\ell\}$. The function f is the nonconvex objective function and \mathcal{X}_i is a closed convex set of feasible points for the variable block x_i . Note that in (5), we require different variable blocks to lie in their own independent feasible sets; this is a requirement for the BCD method as it has been shown that in the

presence of coupled feasible sets, the BCD algorithm may stagnate at a non-stationary point. Here, our penalty reformulation naturally gives us a formulation (2) for which the feasible sets for different variables are independent of each other.

Problem (5) is an MCP because the following subproblems are convex:

$$\text{minimize}_{x_i \in \mathcal{X}_i} \quad f(\bar{x}_1, \dots, \bar{x}_{i-1}, x_i, \bar{x}_{i+1}, \dots, \bar{x}_\ell) \quad \forall i \in \{1, \dots, \ell\}, \quad (6)$$

where the bar over a variable indicates a fixed variable. In BCD, one solves (5) by solving the subproblems (6) cyclically in a Gauss–Seidel fashion until the variable values converge to a limit point. While the required number of iterations for BCD to converge can be exponentially large, the problems solved at each step are convex and overall, this approach has been proven to be significantly more efficient than directly solving the full-space problem for many practical problems (Wright, 2015).

Generally, the convergence of BCD to a stationary point of an optimization problem requires rather stringent conditions on the structure of (5); typically, f needs to be smooth and all the subproblems (6) must have unique solutions (Bertsekas, 1997). In the case when a problem does not meet these criteria, a common strategy is to replace the original problem with a well-chosen approximation for which BCD is guaranteed to converge. A detailed discussion of such strategies can be found in Razaviyayn et al. (2013) and Yang et al. (2019). Here, we make use of the block successive upper minimization (BSUM) algorithm of Razaviyayn et al. (2013) to ensure convergence of BCD for (2), which neither has a smooth objective nor results in subproblems with unique solutions. Specifically, we make use of the result stated in Theorem 2.

Theorem 2 (Razaviyayn et al., 2013). Let f in (5) be nonsmooth and the subproblems (6) be convex, but not necessarily strictly convex. BCD converges to a stationary point of (5) if executed with the following set of subproblems (instead of (6)):

$$\begin{aligned} & \text{minimize}_{x_i \in \mathcal{X}_i} \quad f(\bar{x}_1, \dots, \bar{x}_{i-1}, x_i, \bar{x}_{i+1}, \dots, \bar{x}_\ell) + \frac{1}{2\gamma} \|x_i - \bar{x}_i\|_2^2 \\ & \quad \forall i \in \{1, \dots, \ell\}, \end{aligned} \quad (7)$$

where γ is a scalar parameter.

Algorithm 2 shows the BCD algorithm that we use to solve (5).

Algorithm 2 A cyclic BCD implementation on (5).

```

1: initialize:  $x \leftarrow \bar{x}$ 
2: while convergence criteria are not satisfied do
3:   for all  $i = 1, \dots, \ell$  do
4:     solve (7), obtain  $x_i^*, \bar{x}_i \leftarrow x_i^*$ 
5:   end for
6: end while
7: return  $\bar{x}$ 

```

The following corollary follows directly from Theorem 2.

Corollary 1. Suppose x^* is the accumulation point of the sequence of iterates \bar{x} generated by Algorithm 2. Then x^* is a KKT point of (5).

Initialization. Since our proposed solution method only guarantees to return a KKT point of (1), it is important to seed it with a good initial guess. This is, in general, difficult to achieve without any knowledge of the distribution of noise in the training data. Here we detail a heuristic initialization strategy which, as evident from our extensive computational results in Section 6, is quite effective in finding an estimate for parameters θ and ω that perform well in predicting optimal decisions for unseen u values.

In our approach, we first initialize \hat{x}_i with the noisy decision data x_i for all $i \in I$, i.e., $\hat{x}_0 = x$. Next, we solve the following two problems

to sequentially initialize the $\hat{\omega}$ and $(\hat{\theta}, \lambda, \mu)$ parameters, respectively:

$$\hat{\omega}_0 \in \arg \min_{\omega \in \Omega} \sum_{i \in I} \max\{0, \hat{g}(\hat{x}_{0i}, u_i; \hat{\omega})\} + \sum_{i \in I} |\hat{h}(\hat{x}_{0i}, u_i; \hat{\omega})| \quad (8)$$

$$(\hat{\theta}_0, \lambda_0, \mu_0) \in \arg \min_{\hat{\theta} \in \Theta, \lambda, \mu} \sum_{i \in I} |\nabla \hat{f}(\hat{x}_{0i}, u_i; \hat{\theta}) + \lambda_i^T \nabla \hat{g}(\hat{x}_{0i}, u_i; \hat{\omega}_0)| + \mu_i^T \nabla \hat{h}(\hat{x}_{0i}, u_i; \hat{\omega}_0) + \sum_{i \in I} |\lambda_i^T \hat{g}(\hat{x}_{0i}, u_i; \hat{\omega}_0)| \quad (9)$$

$$\text{subject to } \lambda_i \geq 0 \quad \forall i \in I$$

Hyperparameter setting. The overall algorithm involves three hyperparameters: the initial penalty parameters c_1 , the factor ρ by which c are increased after every iteration of Algorithm 1, and the regularization parameter γ for BCD subproblems (7). Out of these three, we find that c_1 and ρ have a large impact on the quality of the solution determined by our approach. While choosing c_1 and ρ to be of large magnitude helps to quickly find a solution for which $\|P\| = 0$, very often the resulting solution is of low quality. Here, assigning small values to c_1 and ρ generally solves the problem, but slows the convergence of Algorithm 1. In our computational experiments, we utilize a validation dataset to determine sufficiently large values of c_1 and ρ for which the algorithm converges quickly yet generates good solutions.

The regularization parameter γ does not affect the quality of the solution, but has an impact on BCD's rate of convergence. We find that not having the regularization term in (7), i.e., setting γ to ∞ may make BCD cycle between non-stationary points without converging to any particular solution. On the other hand, making γ small stabilizes the algorithm but makes convergence extremely slow as the regularization term becomes dominant. In our computational experiments, we observe that assigning a large value to γ such as $\gamma = 10^6$ typically works well in terms of stabilizing the BCD iterations yet ensuring fast convergence to a stationary point.

6. Computational case studies

We apply the proposed solution methods for the data-driven IO problem to three case studies. The first case study is based on the so-called water-filling problem from the field of information theory (Kalpana and Khan, 2015). We use this example to showcase the computational advantages of Algorithms 1 and 2. The next two case studies, one addressing learning the risk preferences of a decision maker and the second related to estimating parameters of a resource allocation market, demonstrate the utility of IO in systems engineering applications. All model instances presented in this section were implemented in Julia v1.6.1 using the mathematical optimization modeling environment JuMP v0.21.10 (Dunning et al., 2017). We applied Gurobi v9.1.2 to solve all convex optimization problems, and all nonconvex NLPs were solved using IPOPT v0.7.0.

6.1. The water-filling problem

We consider the following variation of the water-filling problem from the wireless communication literature that is used to determine optimal power allocation for a multidimensional communication channel:

$$\begin{aligned} & \underset{x \in \mathbb{R}_+^D}{\text{maximize}} && \sum_{d=1}^D \theta_d \log(x_d + u_d) \\ & \text{subject to} && \sum_{d=1}^D \omega_d x_d = \omega_{D+1}, \end{aligned} \quad (10)$$

where the objective function represents the total communication rate and the constraint limits the total amount of power that can be allocated to the system. Here we assume that the weighting parameters θ for the objective and ω for the constraint are unknown. The goal

is to estimate these parameters by observing changes in the optimal power allocation decisions x based on fluctuations in the input parameters u . In the context of IO, this problem has previously been considered by Aswani et al. (2018); however, their work is limited to the estimation of objective parameters assuming ω to be known.

For each instance of the IOP, the training data are generated as follows. We first create arbitrary vectors $\theta \in \mathbb{R}^D$ and $\{\omega_d\}_{d=1}^D$ with $\omega_d \in \mathbb{R}^D$ by sampling their individual elements from the uniform distribution $\mathcal{U}(1.00, 1.10)$. For all instances, we set the right-hand-side constraint parameter ω_{D+1} to the dimensionality of the problem D . We then sample a set of input vectors $u_i \in \mathbb{R}^D$ for each $i \in I$ such that $u_{id} \sim \mathcal{U}(1.00, 2.00)$ for every $d \in \{1, \dots, D\}$. Next, keeping θ and ω the same, we solve these $|I|$ instances of (10) to obtain the optimal power allocation decisions x_i^* . Finally, we distort these true optimal solutions by adding a Gaussian noise $\gamma \sim \mathcal{N}(0, \sigma^2 \mathbb{I})$ to obtain the noisy dataset as $x_i = x_i^* + \gamma$. In this study, to show the computational advantage of Algorithm 1, we consider large datasets of up to 1,000 samples with FOPs of varying dimensionality. We also consider varying levels of noise in the datasets by changing the value of σ . A specific case is hence represented by D , σ , and $|I|$, and we solve ten random instances for each case to obtain reliable performance statistics for different solution methods. While setting up the IOP, we assume $\hat{\Theta} = \{\hat{\theta} \mid 10^{-4} \leq \hat{\theta}_d \leq 10 \quad \forall d \in \{1, \dots, D\}\}$ and $\hat{\Omega} = \{\hat{\omega} \mid \hat{\omega}_d \geq 0 \quad \forall d \in \{1, \dots, D\}, \hat{\omega}_{D+1} = 1\}$. The sets $\hat{\Theta}$ and $\hat{\Omega}$ have been chosen so that trivial solutions, such as $\theta = 0$, are eliminated from the feasible solution space of the IOP.

6.1.1. Convergence criterion for Algorithm 1

In Algorithm 1, we consider the algorithm to have converged when the norm of the penalty term becomes less than a certain threshold value ϵ . However, in our computational experiments, we find convergence of the algorithm with BCD in the inner loop to be very slow. Fig. 1a shows the evolution of $\|P\|_1$ as a function of the number of iterations with iteration 0 representing the initialization solution. As can be observed, the value of $\|P\|_1$ reaches a peak before settling into a slow decay phase, after which the desired convergence threshold takes a long time to reach. This may, at first glance, make BCD-based Algorithm 1 appear an impractical approach for our problem. However, as can be seen in Fig. 1b, we find that the prediction error of the generated estimates $\hat{\theta}$ and $\hat{\omega}$ stabilizes when $\|P\|_1$ enters the slow decay phase. Running the algorithm after this point does not improve the prediction accuracy. Therefore, we modify our convergence criterion to be the stabilization of the prediction error instead of $\|P\|_1 \leq \epsilon$.

Note that the slow reduction in $\|P\|_1$ is observed only when BCD is used to solve the penalty reformulation. When solving the penalty reformulation directly with IPOPT, we find that, in most cases, the algorithm converges in just one or two iterations.

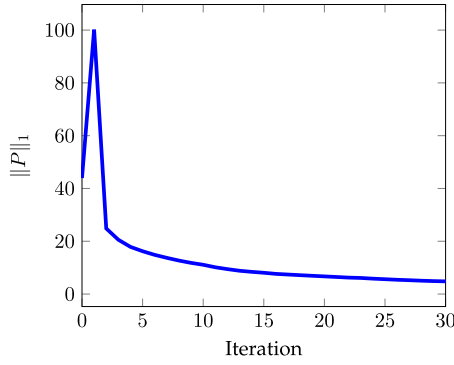
6.1.2. Computational performance

We report results comparing the computational performance of three different solution methods for (1):

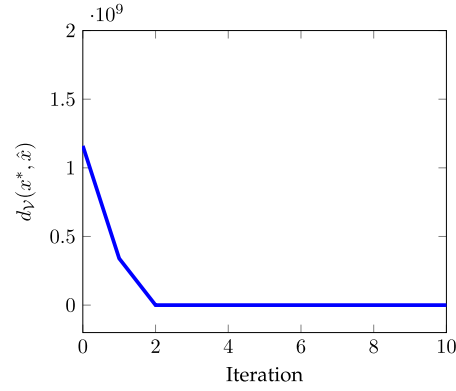
- Method A - solve the problem directly with IPOPT,
- Method B - use Algorithm 1 with IPOPT to solve the penalty reformulation, and
- Method C - use BCD (Algorithm 2) to solve the penalty reformulation in Algorithm 1.

For a fair comparison, we initialize all three methods with the same initial solution obtained using the strategy described in Section 5. All model instances are solved with a time limit of 300 s using 24 cores and 16 GB of memory on the Mesabi cluster of the Minnesota Supercomputing Institute (MSI).

Table 1 summarizes the results for Method A. For a specific D , σ , and $|I|$, we show the median value of the computation time required to solve the ten random instances. An "n/a" entry in this column indicates that IPOPT was unable to solve any of the instances in the



(a) Norm of the penalty term



(b) Prediction error on the test dataset

Fig. 1. Example progress of $\|P\|_1$ and $d_V(x^*, \hat{x})$ values as the number of iterations of Algorithm 1 increases.

Table 1

Computational performances of IPOPT (Method A) on instances of (1) designed to estimate parameters of (10).

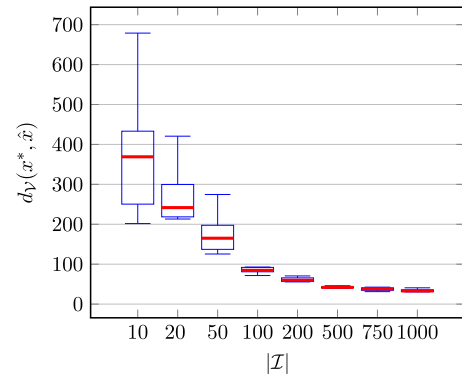
σ	I	$D = 50$				$D = 100$			
		Median computation time (s)	Prediction error			Median computation time (s)	Prediction error		
			median	min	max		median	min	max
0.01	D	36	62.96	26.90	1702.86	96	580.16	79.21	1305.69
	1000	198	5.61	4.70	11.33	206	6.26	5.56	6.54
0.05	D	64	1.97	0.29	16.62	63	3.10×10^5	118.57	2.93×10^8
	1000	n/a	–	–	–	n/a	–	–	–
0.1	D	60	2.33	0.54	7.07	64	6.34×10^6	6.13×10^4	8.11×10^7
	1000	n/a	–	–	–	n/a	–	–	–

specified time limit. Additionally, we provide the median, maximum, and minimum prediction errors of the generated estimates $\hat{\theta}$ and $\hat{\omega}$. To obtain these statistics, along with every instance of training data, we also generate a test dataset \mathcal{V} of 100 inputs. This test data consists of (u, x^*) pairs, where u values are generated in the same manner as for the training data, and x^* are the corresponding true optimal solutions obtained using the same θ and ω as the ones used to generate the noisy training data. Once $\hat{\theta}$ and $\hat{\omega}$ have been found, we use them to solve the problems in the test dataset and evaluate the prediction error as the total Manhattan distance between the true solutions and their estimated values, i.e., we use the metric $d_V(x^*, \hat{x}) = \sum_{v \in \mathcal{V}} \|x_v^* - \hat{x}_v\|_1$.

From the data in Table 1, one can observe that IPOPT can solve smaller instances very quickly but struggles with larger problems. Moreover, even for problems with smaller datasets, there is a large variability in the prediction accuracy values obtained with estimated parameters. The solved instances with $D = 50$ generally exhibit good prediction performance, in contrast to instances with $D = 100$ irrespective of the dataset size. We find that, generally, IPOPT is not a robust solution method for difficult instances of (1) with either high dimensionality or large datasets.

Next, using the same metrics as defined for Method A, we show a comparison of the performances of solution methods B and C in Table 2. While implementing Algorithm 1 on IOP instances of (10), we set $c_1 = 500e$, where e is a vector of all ones and $\rho = 1,000$. These values were obtained by trial-and-error as described in Section 5.

The computational advantage of using a BCD-based decomposition strategy is immediately apparent from the median computation times in Table 2. Even for the smaller instances where we do not expect decomposition to vastly outperform a full-space formulation, BCD outputs a solution as much as ten times faster compared to when IPOPT is used to solve the penalty reformulation of (1). For larger problems, we observe that while IPOPT starts reaching the time limit of 300 s, BCD is able to output a solution in approximately 100 s.

Fig. 2. Change in prediction error as a function of the training dataset size. The box plot shows the interquartile ranges of prediction error values for the ten random instances solved with $D = 100$ and $\sigma = 0.1$ using solution method C.

6.1.3. Solution quality

Next, we focus on the quality of the solutions generated by Methods B and C. For smaller instances, we find that both methods yield solutions of similar prediction accuracy. However, as we move to problems with larger datasets, IPOPT does not produce a good solution in any of the problem instances. In contrast, with BCD, we see that the median prediction error stays low for all cases. The only case where BCD may result in a bad solution is when the training data has a high level of noise yet a small dataset is provided to generate an estimate of the missing FOP parameters. Finally, Fig. 2 shows that the solutions obtained from BCD asymptotically converge towards the one with minimum prediction error. This empirically confirms that our algorithm generates statistically consistent solutions.

Overall, we find that Algorithm 1 is a significantly more robust solution method for problem (1) compared to a direct application of

Table 2

Comparison of computational performances of solution methods B and C on an IOP based on random instances of (10).

D	σ	I	Solved with IPOPT (Method B)				Solved with BCD (Method C)				
			Median computation time (s)		Prediction error		Median computation time ^a (s)		Prediction error		
					median	min			max	median	min
50	0.01	50	11		64.20	26.90	104.08	1	41.60	23.65	141.88
		1000	219		1.85	1.70	18.49	33	2.96	2.32	3.14
	0.05	50	24		108.21	94.65	121.49	1	116.55	89.62	255.47
		1000	300		71.83	9.25	6533.90	31	14.77	10.99	15.65
	0.1	50	55		174.99	143.54	286.04	2	166.19	134.18	1.02×10^7
		1000	300		4504.87	3994.46	4512.91	61	37.81	31.18	44.19
100	0.01	100	101		102.24	42.67	797.99	4	109.16	42.21	338.88
		1000	300		22.60	19.73	160.78	51	5.88	5.71	6.53
	0.05	100	113		155.36	135.21	1426.01	7	156.68	134.54	1.44×10^6
		1000	300		2.32×10^5	1.36×10^5	6.13×10^5	112	47.85	27.95	53.28
	0.1	100	109		195.23	91.05	265.68	10	248.60	199.15	2.18×10^8
		1000	300		4.4×10^5	8.6×10^4	6.43×10^5	107	70.60	65.86	93.36

^aBased on the modified convergence criterion described in Section 6.1.1.

standard commercial NLP solvers. Moreover, when the penalty reformulation of (1) is an MCP (as in this case), using BCD in the inner loop of the algorithm produces marked improvement in its computational efficiency without compromising (or rather enhancing) the final solution quality.

6.2. Estimating risk preferences

When people make decisions under significant uncertainty, their choices strongly depend on their individual risk preferences. Knowing these risk preferences can be very helpful to higher-level decision makers; for example, companies may be able to create better products and services for their risk-averse customers, and policymakers could develop more effective regulations and incentives for their communities. However, often these risk preferences are not explicitly known but need to be estimated from observed decisions.

In risk-averse optimization, the decision-making problem is commonly formulated as a two-stage stochastic program of the following form:

$$\text{minimize}_{x \in \mathcal{X}} f_1(x) + \mathbb{E}(Q(x, \xi)) + \lambda \rho(Q(x, \xi)), \quad (11)$$

where x and ξ denote the first-stage decisions and uncertain parameters (or random variables), respectively, \mathcal{X} is the first-stage feasible set, and $f_1(x)$ is the first-stage cost function. The recourse function is denoted by $Q(x, \xi)$ and is defined as

$$Q(x, \xi) = \min_{y \in \mathcal{Y}(x, \xi)} f_2(x, y, \xi) \quad (12)$$

with $\mathcal{Y}(x, \xi)$ being the set of feasible second-stage decisions y and $f_2(x, y, \xi)$ being the second-stage cost function. The risk function is denoted by ρ , and the objective is to minimize a weighted sum of the expected cost and risk, where the weighting factor λ is chosen by the decision maker based on their level of risk aversion.

Many common risk functions are convex, which allows us to learn them using the proposed data-driven IO framework, given that the overall FOP is also convex. Indeed, in the risk literature, there is the notion of *coherent* risk measures, which exhibit properties that are deemed desirable for the purpose of quantifying risk (Artzner et al., 1999); one of those properties is convexity. In the following example, we consider a simplified version of the risk-based electricity procurement problem introduced by Zhang et al. (2016), where the conditional value-at-risk (CVaR), one of the most popular coherent risk measures, is used as the risk function ρ .

In this problem, the decision maker needs to schedule the operation of a production plant that consumes a large amount of electricity, which can be purchased from a power contract or from the spot market. The prices for the power contract are known, but one must commit to the

purchase amount in advance, e.g. one day or one week ahead. Spot electricity prices, on the other hand, are uncertain and only become known shortly before the time of delivery. While purchasing electricity from the power contract bears no risk, the average contract price is typically higher than the average spot price, and it leaves less room for taking advantage of fluctuations in the spot price; hence, there is a trade-off between purchasing from the power contract and the spot market, which one seeks to optimize. Assuming that the uncertain spot prices follow a discrete probability distribution, the problem can be formulated as the following deterministic equivalent of the corresponding two-stage stochastic program:

$$\begin{aligned} \text{minimize}_{x, y, q, v, w} \quad & \sum_{t \in \mathcal{T}} c_t x_t^2 + \sum_{s \in S} p_s \sum_{t \in \mathcal{T}} (b_t q_{ts} + r_{ts} y_{ts}) \\ & + \lambda \left(v + \frac{1}{1 - \alpha} \sum_{s \in S} p_s w_s \right) \end{aligned} \quad (13a)$$

$$\text{subject to} \quad i^{\min} \leq i^0 + \sum_{t'=1}^t (q_{ts} - d_t) \leq i^{\max} \quad \forall t \in \mathcal{T}, s \in S \quad (13b)$$

$$\sum_{t \in \mathcal{T}} (q_{ts} - d_t) \geq 0 \quad \forall s \in S \quad (13c)$$

$$m q_{ts} = x_t + y_{ts} \quad \forall t \in \mathcal{T}, s \in S \quad (13d)$$

$$\sum_{t \in \mathcal{T}} (b_t q_{ts} + r_{ts} y_{ts}) - v \leq w_s \quad \forall s \in S \quad (13e)$$

$$x_t \in \mathbb{R}_+ \quad \forall t \in \mathcal{T} \quad (13f)$$

$$q_{ts}, y_{ts} \in \mathbb{R}_+ \quad \forall t \in \mathcal{T}, s \in S \quad (13g)$$

$$w_s \in \mathbb{R}_+ \quad \forall s \in S, \quad (13h)$$

where \mathcal{T} is the set of time periods that defines the scheduling horizon, and S is the set of possible uncertainty realizations (or scenarios). Each scenario s is defined by its probability p_s and spot price profile r_{ts} . The first-stage decisions are the amounts of electricity purchased from the power contract, denoted by x_t , across the scheduling horizon. The second-stage variables are the production amounts, q_{ts} , and the amounts of electricity purchased from the spot market, y_{ts} . The cost of electricity purchased from the power contract in time period t is $c_t x_t^2$, and the production cost (excluding the electricity cost) is assumed to be a linear function of the production amount defined by the coefficients b_t . In constraints (13b), the parameters i^0 , i^{\min} , i^{\max} , and d_t denote the initial product inventory level, the minimum inventory, the maximum inventory, and the product demand in time period t , respectively. Constraints (13c) ensure that the total amount of product produced over the course of the scheduling horizon is not less than the total demand. We assume that the electricity consumption is a linear function of the production amount defined by the coefficient m and,

per constraints (13d), must be equal to the total amount of electricity procured in each time period. The term in the objective function (13a) that is multiplied by the weighting factor λ , along with constraints (13e), represents the α -CVaR. For a given $\alpha \in (0, 1)$, the α -CVaR is defined as the expected cost greater than the α -VaR, which is the α -quantile of the cost distribution. Note the use of the auxiliary variables v and w_s , where for each scenario in which the cost is greater than v , w_s takes the value of the difference between the cost and v .

In this case study, we assume that the decision maker must decide on their electricity procurement strategy two days in advance. We discretize the 48-hour time horizon into one-hour time periods, yielding $|\mathcal{T}| = 48$. We assume that the decision maker considers 35 different spot electricity price scenarios and $\alpha = 0.9$ when making their decisions. We set up the IOP by assuming that we have access to historical electricity procurement data in which the decision maker made decisions under changing sets of spot electricity price scenarios. Our goal here is to use our knowledge of price scenarios and the corresponding procurement decisions to estimate the decision maker's risk tolerance (λ) and preferred safety stock level (i^{\min}), where the latter can also be seen as an indicator of the decision maker's level of risk aversion. All other parameters in (13) are assumed to be known. We assume that only the optimal values of (x_t, q_{ts}) for all $t \in \mathcal{T}$ and one of the $s \in \mathcal{S}$ can be observed since in practice, we are only able to see the decisions made in the scenario that actually realized. As such, this case study is an example of an IOP in which only a subset of the decision variables of the FOP can be observed.

We provide the values of all the parameters for (13) including the contract and spot electricity price profiles in the supplementary material. To generate each training data point in \mathcal{I} , we begin by selecting 35 spot price scenarios at random from the 40 included in the supplementary material. We then solve (13) with these scenarios for certain λ and i^{\min} to obtain the optimal decisions. We select one of the s in \mathcal{S} at random as the scenario that actually realizes, and we store the corresponding optimal x_t and q_{st} values for all $t \in \mathcal{T}$, as well as the r_{st} values for all $t \in \mathcal{T}$ and $s \in \mathcal{S}$, as a data point in \mathcal{I} .

Here, the FOP is a QP with 3444 variables and 8553 constraints, making the IOP a large-scale nonconvex NLP. As a result, we use BCD to solve the penalty reformulation in Algorithm 1. This algorithm was implemented with $c_1 = 0.01e$ and $\rho = 0.01$. Because we only observe a subset of the decision variables, the initialization strategy discussed in Section 5 cannot be applied here, so the algorithm is initialized with random values for the missing parameters. We solve the IOP using a training dataset of 5 partial observations (i.e., $|\mathcal{I}| = 5$), which we find is sufficient to accurately estimate the values of λ and i^{\min} . Furthermore, we solve five instances of the IOP, each with a different randomly generated training dataset, to confirm that our results are robust against price scenario selection.

The results of our computational experiments are summarized in Table 3. We consider three different cases involving decision makers with varying risk tolerances. In Case 1, where the decision maker is almost risk-neutral (with a low λ value), our algorithm predicts the same λ and i^{\min} values as the ones used to generate the five instances of training data. However, in the other two cases we observe some discrepancies between predicted and true values. In Case 2, where the decision maker has a medium risk tolerance, the predicted value of i^{\min} is greater than its true value. This is because the constraints and parameters of (13) are such that the product inventory can never fall below 50 and therefore, our algorithm "thinks" that 50 is the safety stock level for this decision maker. In Case 3, where the large λ indicates a highly risk-averse decision maker, the reason for the difference in the λ values is that for any $\lambda \gtrsim 5$, the decision maker is so risk-averse that it always chooses to buy all the electricity from the power contract. It is important to note that in Cases 2 and 3, despite the differences in the true and predicted parameter values, solving (13) with any of the predicted λ and i^{\min} values will always yield

Table 3

Results of computational experiments based on electricity procurement problem (13).

		True values	Predicted values				
			Instances				
			1	2	3	4	5
Case 1	λ	0.1	0.099	0.1	0.099	0.1	0.099
	i^{\min}	75	74.99	74.99	74.99	74.99	74.99
Case 2	λ	2	1.97	1.98	1.97	1.99	1.965
	i^{\min}	40	50	50	50	50	50
Case 3	λ	7	6.98	19.52	5.52	5.36	23.28
	i^{\min}	60	59.99	59.99	60	59.99	60

the same decisions as the ones obtained using the true FOP. In other words, in all three cases, our solution approach provides parameters that effectively minimize the decision error on both seen and unseen spot price scenarios which is what the objective of (IOP) is.

In this problem, we only had access to a subset of the decisions from the model, yet we find that even with partial observability of the decisions, we were able to correctly recover the unknown parameters. However, this is not always the case. In most situations, partial observability of the decisions will expand the solution space of the inverse problem which may lead to estimated parameter values that do not predict the decisions accurately. This highlights the need to develop conditions under which partially observable decisions can be used to accurately recover the parameters of the FOP with some theoretical guarantee.

6.3. Resource allocation market

In this case study, we consider a market with a set of buyers \mathcal{B} that enter a bargaining game to acquire a limited set of resources \mathcal{G} . Each buyer has a utility u_b as a function of resource allocations and a disagreement point d_b , which is the status quo that player b will revert to if no agreement is reached. We assume that the buyers agree to cooperate with each other such that the resource allocation is fair. Nash (1950) showed that if the utility set is compact and convex, solving the following problem achieves the unique solution that satisfies certain axioms, which represent a notion of fairness:

$$\begin{aligned} & \underset{x}{\text{maximize}} && \sum_{b \in \mathcal{B}} (u_b(x) - d_b) \\ & \text{subject to} && p(x) \leq 0, \end{aligned} \quad (14)$$

where x are the resource allocation decisions and p is a convex function encoding the constraints on the market. Problem (14) is not convex; however, a logarithmic transformation of the objective function allows it to be reformulated as a convex optimization problem.

Here, we assume that the market is constrained by two sets of constraints: first, buyers are limited in the amount of resources they can acquire, and second, the total amount of each resource available in the market is limited. A convexified reformulation of (14) for this market is as follows:

$$\begin{aligned} & \underset{x \in \mathbb{R}_+^{|\mathcal{B}| \times |\mathcal{G}|}}{\text{maximize}} && \sum_{b \in \mathcal{B}} \log \left(\sum_{g \in \mathcal{G}} x_{bg} \right) \\ & \text{subject to} && \sum_{g \in \mathcal{G}} p_{bg} x_{bg}^2 \leq m_b \quad \forall b \in \mathcal{B} \\ & && \sum_{b \in \mathcal{B}} x_{bg} \leq c_g \quad \forall g \in \mathcal{G}, \end{aligned} \quad (\text{RAMP})$$

where x_{bg} refers to the amount of a resource g that a buyer b ends up acquiring from the market. The first set of constraints place limits on how much resource a buyer can acquire and the second set of constraints model the capacity of the market. The objective function in this model seeks to determine a fair allocation of utilities, $u_b = \sum_{g \in \mathcal{G}} x_{bg}$ for each buyer $b \in \mathcal{B}$ assuming that at the disagreement point, $d_b = 0$.

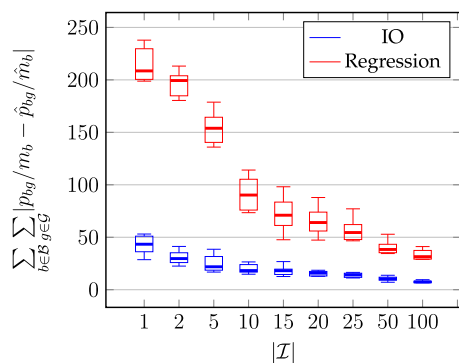


Fig. 3. Change in the error in the normalized parameter estimate values as a function of the training dataset size. Box plots show the interquartile ranges of error values for ten random instances of the problem generated by following the scheme described in the text.

For our IO set up, we consider a scenario where the resource providers cooperate among themselves to learn the constraints of the buyers, i.e., the resource providers are interested in estimating the values of parameters p_{bg} and m_b for each $b \in B$. To learn these parameters, the resource providers rely on their knowledge of historical capacity fluctuation data (changes in the values of c_g) and the corresponding fair allocations. We assume that this historical resource allocation data is only available with high noise.

To generate training data for this case study, we assume a market with 20 buyers and 5 resources. For each instance of the IOP, we sample individual elements of $p \in \mathbb{R}^{|B| \times |G|}$ and $m \in \mathbb{R}^{|B|}$ randomly from uniform distributions $\mathcal{U}(0.5, 2)$ and $\mathcal{U}(0.5, 1)$, respectively. We then sample a set of capacity vectors $c_{ig} \in \mathbb{R}^{|G|}$ for each $i \in I$ such that each of their elements follow the distribution $\mathcal{U}(0.1, 1.5)$. Next, keeping p and m the same, we solve the $|I|$ instances of (RAMP) corresponding to each of the $|I|$ c_{ig} vectors. We then distort the obtained optimal resource allocation decisions x_{ibg}^* by adding a Gaussian noise $\gamma \sim \mathcal{N}(0, \sigma^2 \mathbb{I})$ to them. Here we find that with our chosen parameter values, the mean value of x_{ibg}^* for each $i \in I$ results in being approximately 0.3, and therefore to simulate a high noise scenario, we set the value of σ to 0.05.

Note that the single-level IOP reformulation (1) for (RAMP) is not an MCP. Therefore, we are restricted to Algorithm 1 to solve this problem. Nonetheless, we find that the constraint parameters of (RAMP) can be learned with just a few data points, making the problem size amenable for Algorithm 1. While learning the parameters, we assume that the m_b values for all $b \in B$ are 1 and instead learn the normalized p_{bg} parameters, i.e., the p_{bg}/m_b values. In Fig. 3, we compare the accuracy of estimates generated through IO with simple regression-based estimates. To generate the regression estimates, we apply the least-squares method on the noisy data to find the best-fit parameters for the quadratic constraints assuming that they hold as equalities in all cases. While we agree that assuming that these constraints always hold as equalities is a rather strong assumption and may negatively affect the quality of the estimates, this is typically how this problem will be solved if no other information is available. Indeed, as we show in Fig. 3, IO can generate more accurate estimates with just 5 data points than what regression can produce with 100 data points. On the one hand, this is an obvious result because our assumption of a fair allocation provides additional information about the data. On the other hand, it also shows the power of the proposed framework in being highly data-efficient by allowing the incorporation of domain knowledge into the inverse problem.

7. Conclusions

In this work, we described the idea of discovering unknown decision-making processes from observed decisions using mathematical

optimization as a natural model for decision making. Using this concept, inferring the decision-making model is equivalent to estimating the unknown parameters of the underlying optimization problem, which is referred to as data-driven inverse optimization. We considered the case in which the underlying decision-making process can be formulated as a convex optimization problem. We formulated the IO problem as a bilevel program with as many lower-level problems as the number of available observations of prior decisions. To address the computational challenge associated with solving large instances of the IO problem, we proposed an efficient penalty-based BCD algorithm that leverages the decomposable structure of the problem.

We conducted extensive computational experiments to benchmark the performance of the proposed solution method against standard commercial solvers. In large instances, we show that using our BCD algorithm does not only reduce the computation time but also results in higher-quality solutions. Furthermore, we present two additional computational case studies based on practically relevant problems, one concerned with estimating risk preferences and the other one aimed at uncovering local constraint parameters in a multiplayer Nash bargaining formulation. Here, our results indicate that IO can recover highly accurate estimates of the parameters of interest while using only a very small number of data points.

Finally, we would like to highlight that there are many important directions to consider for future work including decomposition algorithms for the problems where BCD cannot be applied, dealing with partially observable decisions, incorporating prior beliefs, and using adaptive sampling to reduce the amount of data required to learn the parameters.

CRedit authorship contribution statement

Rishabh Gupta: Methodology, Software, Formal analysis, Investigation, Writing – original draft. **Qi Zhang:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors gratefully acknowledge the financial support from the National Science Foundation, USA under Grant #2044077 as well as the Minnesota Supercomputing Institute (MSI) at the University of Minnesota for providing resources that contributed to the research results reported in this paper. R.G. acknowledges financial support from a departmental fellowship sponsored by 3M and a Doctoral Dissertation Fellowship from the University of Minnesota, USA.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.compchemeng.2022.108123>.

References

- Abbeel, P., Ng, A.Y., 2004. Apprenticeship learning via inverse reinforcement learning. In: *Proceedings of the 21st International Conference on Machine Learning*.
- Ahuja, R.K., Orlin, J.B., 2001. Inverse optimization. *Oper. Res.* 49 (5), 771–783.
- Akhtar, S.A., Kolarijani, A.S., Esfahani, P.M., 2022. Learning for control: An inverse optimization approach. *IEEE Control Syst. Lett.* 6 (c), 187–192.
- Anitescu, M., 2000. Nonlinear programs with unbounded Lagrange multiplier sets. In: *Preprint ANL/MCS-P796-0200*, Mathematics and Computer Science Division. Argonne National Laboratory, Argonne, IL.
- Anitescu, M., 2005. On using the elastic mode in nonlinear programming approaches to mathematical programs with complementarity constraints. *SIAM J. Optim.* 15 (4), 1203–1236.
- Artzner, P., Delbaen, F., Eber, J.M., Heath, D., 1999. Coherent measures of risk. *Math. Finance* 9 (3), 203–228.
- Aswani, A., Shen, Z.-J.M., Siddiq, A., 2018. Inverse optimization with noisy data. *Oper. Res.* 66 (3), 870–892.
- Babier, A., Chan, T.C.Y., Lee, T., Mahmood, R., Terekhov, D., 2021. An ensemble learning framework for model fitting and evaluation in inverse linear optimization. *Inf. J. Optim.* 3 (2), 119–138.
- Balázsi, G., Van Oudenaarden, A., Collins, J.J., 2011. Cellular decision making and biological noise: From microbes to mammals. *Cell* 144 (6), 910–925.
- Bañares-Alcántara, R., Westerberg, A.W., Rychener, M.D., 1985. Development of an expert system for physical property predictions. *Comput. Chem. Eng.* 9 (2), 127–142.
- Bangi, M.S.F., Kwon, J.S.I., 2020. Deep hybrid modeling of chemical process: Application to hydraulic fracturing. *Comput. Chem. Eng.* 134.
- Benson, H.Y., Sen, A., Shanno, D.F., 2007. Convergence analysis of an interior-point method for nonconvex nonlinear programming. Available on Optimization Online.
- Benson, H.Y., Sen, A., Shanno, D.F., Vanderbei, R.J., 2006. Interior-point algorithms, penalty methods and equilibrium problems. *Comput. Optim. Appl.* 34 (2), 155–182.
- Bertsekas, D.P., 1997. Nonlinear programming. *J. Oper. Res. Soc.* 48 (3), 334.
- Bertsimas, D., Gupta, V., Paschalidis, I.C., 2015. Data-driven estimation in equilibrium using inverse optimization. *Math. Program.* 153 (2), 595–633.
- Birge, J.R., Hortacsu, A., Pavlin, M., 2017. Inverse optimization for the recovery of market structure from market outcomes: An application to the MISO electricity market. *Oper. Res.* 65 (4), 837–855.
- Bollas, G.M., Barton, P.I., Mitsos, A., 2009. Bilevel optimization formulation for parameter estimation in vapor-liquid(-liquid) phase equilibrium problems. *Chem. Eng. Sci.* 64 (8), 1768–1783.
- Boukouvala, F., Floudas, C.A., 2017. ARGONAUT: Algorithm for Global Optimization of constrained grey-box computational problems. *Optim. Lett.* 11 (5), 895–913.
- Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* 3 (1), 1–122.
- Burgard, A.P., Maranas, C.D., 2003. Optimization-based framework for inferring and testing hypothesized metabolic objective functions. *Biotechnol. Bioeng.* 82 (6), 670–677.
- Burton, D., Toint, P.L., 1992. On an instance of the inverse shortest paths problem. *Math. Program.* 53 (1–3), 45–61.
- Chan, T.C.Y., Craig, T., Lee, T., Sharpe, M.B., 2014. Generalized inverse multiobjective optimization with application to cancer therapy. *Oper. Res.* 62 (3), 680–695.
- Chan, T.C., Eberg, M., Forster, K., Holloway, C., Ieraci, L., Shalaby, Y., Yousefi, N., 2022. An inverse optimization approach to measuring clinical pathway concordance. *Manage. Sci.* 68 (3), 1882–1903.
- Chan, T.C., Kaw, N., 2019. Inverse optimization for the recovery of constraint parameters. *European J. Oper. Res.* 282 (2), 415–427.
- Chan, T.C., Lee, T., Terekhov, D., 2019. Inverse optimization: Closed-form solutions, geometry, and goodness of fit. *Manage. Sci.* 65 (3), 1115–1135.
- Chan, T.C., Mahmood, R., Zhu, I.Y., 2021. Inverse optimization: theory and applications. *arXiv preprint arXiv:2109.03920*.
- Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y., 2015. The loss surfaces of multilayer networks. In: *Artificial Intelligence and Statistics*. PMLR, pp. 192–204.
- Chow, J.Y., Ritchie, S.G., Jeong, K., 2014. Nonlinear inverse optimization for parameter estimation of commodity-vehicle-decoupled freight assignment. *Transp. Res. E* 67, 71–91.
- Daniilova, M., Dvurechensky, P., Gasnikov, A., Gorbunov, E., Guminov, S., Kamzolov, D., Shibaev, I., 2022. Recent theoretical advances in non-convex optimization. In: *High-Dimensional Optimization and Probability*. Springer, pp. 79–163.
- Dempe, S., Kalashnikov, V., Pérez-Valdés, G.A., Kalashnykova, N., 2015. Bilevel programming problems. In: *Energy Systems*. Springer, Berlin.
- Dunning, I., Huchette, J., Lubin, M., 2017. JuMP: A modeling language for mathematical optimization. *SIAM Rev.* 59 (2), 295–320.
- Gautam, R., Seider, W.D., 1979. Computation of phase and chemical equilibrium: Part I. Local and constrained minima in Gibbs free energy. *AIChE J.* 25 (6), 991–999.
- Ghobadi, K., Mahmoudzadeh, H., 2020. Multi-point inverse optimization of constraint parameters. *arXiv preprint, arXiv:2001*.
- Glass, M., Aigner, M., Viell, J., Jupke, A., Mitsos, A., 2017. Liquid-liquid equilibrium of 2-methyltetrahydrofuran/water over wide temperature range: Measurements and rigorous regression. *Fluid Phase Equilib.* 433, 212–225.
- Glass, M., Mitsos, A., 2018. Parameter estimation in reactive systems subject to sufficient criteria for thermodynamic stability. *Chem. Eng. Sci.* 197, 420–431.
- Gupta, R., Zhang, Q., 2022. Decomposition and adaptive sampling for data-driven inverse linear optimization. *INFORMS J. Comput.* 34 (5), 2720–2735.
- Heuberger, C., 2004. Inverse combinatorial optimization: A survey on problems, methods, and results. *J. Comb. Optim.* 8 (3), 329–361.
- Hey, J.D., Orme, C., 1994. Investigating generalizations of expected utility theory using experimental data. *Econometrica* 62 (6), 1291–1326.
- Iraj, E.H., Terekhov, D., 2021. Comparing inverse optimization and machine learning methods for imputing a convex objective function. *arXiv preprint arXiv:2102.10742*.
- Iyengar, G., Kang, W., 2005. Inverse conic programming with applications. *Oper. Res. Lett.* 33 (3), 319–330.
- Jain, P., Kar, P., 2017. Non-convex optimization for machine learning. *Found. Trends Mach. Learn.* (ISSN: 1935-8237) 10 (3-4), 142–363.
- Jin, C., Netrapalli, P., Ge, R., Kakade, S.M., Jordan, M.I., 2021. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *J. ACM* 68 (2), 1–29.
- Kalpana, N., Khan, M.Z.A., 2015. Fast computation of generalized waterfilling problems. *IEEE Signal Process. Lett.* 22 (11), 1884–1887.
- Keshavarz, A., Wang, Y., Boyd, S., 2011. Imputing a convex objective function. In: *IEEE International Symposium on Intelligent Control*. pp. 613–619.
- McCarl, B.A., Moskowitz, H., Furtan, H., 1977. Quadratic programming applications. *Omega* 5 (1), 43–55.
- McFarland, D.J., 2004. Decision making in animal. *Nature* 269 (5623), 15–21.
- Mohajerin Esfahani, P., Shafieezadeh-Abadeh, S., Hanasusanto, G.A., Kuhn, D., 2018. Data-driven inverse optimization with imperfect information. *Math. Program.* 167 (1), 191–234.
- Morgenstern, O., von Neumann, J., 1944. *Theory of Games and Economic Behavior*. Princeton University Press.
- Nash, Jr., J.F., 1950. The bargaining problem. *Econometrica* 155–162.
- Nocedal, J., Wright, S., 2006. *Numerical Optimization*. Springer Science & Business Media.
- Parker, G.A., Smith, J.M., 1990. Optimality theory in evolutionary biology. *Nature* 348 (6296), 27–33.
- Razaviyayn, M., Hong, M., Luo, Z.-Q., 2013. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM J. Optim.* 23 (2), 1126–1153.
- Rich, S.H., Venkatasubramanian, V., 1987. Model-based reasoning in diagnostic expert systems for chemical process plants. *Comput. Chem. Eng.* 11 (2), 111–122.
- Rich, S.H., Venkatasubramanian, V., 1989. Causality-based failure-driven learning in diagnostic expert systems. *AIChE J.* 35 (6), 943–950.
- Rönnqvist, M., Svenson, G., Flisberg, P., Jönsson, L.E., 2017. Calibrated route finder: Improving the safety, environmental consciousness, and cost effectiveness of truck routing in Sweden. *Interfaces* 47 (5), 372–395.
- Rosen, R., 1967. *Optimality Principles in Biology*. Springer.
- Rossi, C.C., Cardozo-Filho, L., Guirardello, R., 2009. Gibbs free energy minimization for the calculation of chemical and phase equilibrium using linear programming. *Fluid Phase Equilib.* 278 (1–2), 117–128.
- Saez-Gallego, J., Morales, J.M., Zugno, M., Madsen, H., 2016. A data-driven bidding model for a cluster of price-responsive consumers of electricity. *IEEE Trans. Power Syst.* 31 (6), 5001–5011.
- Sammut, C., Hurst, S., Kedzier, D., Michie, D., 1992. Learning to fly. *Mach. Learn. Proc.* 385–393.
- Scheel, H., Scholtes, S., 2000. Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Math. Oper. Res.* 25 (1), 1–22.
- Schoemaker, P.J.H., 1991. The quest for optimality: A positive heuristic of science? *Behav. Brain Sci.* 14, 205–245.
- Shahmoradi, Z., Lee, T., 2022. Quantile inverse optimization: improving stability in inverse linear programming. *Oper. Res.* 70 (4), 2538–2562.
- Shen, X., Diamond, S., Udell, M., Gu, Y., Boyd, S., 2017. Disciplined multi-convex programming. In: *2017 29th Chinese Control and Decision Conference (CCDC)*. IEEE, pp. 895–900.
- Steels, L., 1995. When are robots intelligent autonomous agents? *Robot. Auton. Syst.* 15 (1–2), 3–9.
- Stephanopoulos, G., 1990. Artificial intelligence in process engineering-current state and future trends. *Comput. Chem. Eng.* 14 (11), 1259–1270.
- Uygun, K., Matthew, H.W., Huang, Y., 2007. Investigation of metabolic objectives in cultured hepatocytes. *Biotechnol. Bioeng.* 97 (3), 622–637.
- Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* 106 (1), 25–57.
- Wilson, Z.T., Sahinidis, N.V., 2019. Automated learning of chemical reaction networks. *Comput. Chem. Eng.* 127, 88–98.
- Wright, S.J., 2015. Coordinate descent algorithms. *Math. Program.* 151 (1), 3–34.
- Yang, Y., Pesavento, M., Luo, Z.-Q., Ottersten, B., 2019. Inexact block coordinate descent algorithms for nonsmooth nonconvex optimization. *IEEE Trans. Signal Process.* 68, 947–961.
- Zhang, Q., Cremer, J.L., Grossmann, I.E., Sundaramoorthy, A., Pinto, J.M., 2016. Risk-based integrated production scheduling and electricity procurement for continuous power-intensive processes. *Comput. Chem. Eng.* 86, 90–105.

- Zhang, J., Liu, Z., 1996. Calculating some inverse linear programming problems. *J. Comput. Appl. Math.* 72 (2), 261–273.
- Zhang, J., Liu, Z., Ma, Z., 1996. On the inverse problem of minimum spanning tree with partition constraints. *Math. Methods Oper. Res.* 44 (2), 171–187.
- Zhao, Q., Stettner, A., Reznik, E., Segre, D., Paschalidis, I.C., 2015. Learning cellular objectives from fluxes by inverse optimization. In: *Proceedings of the IEEE Conference on Decision and Control*, 54rd IEEE (Cdc). pp. 1271–1276.