Automated Mechanism Design: A Survey

MICHAEL J. CURRY
University of Illinois Chicago
and
ZHOU FAN, YANCHEN JIANG, SAI SRIVATSA RAVINDRANATH, TONGHAN WANG, DAVID C. PARKES
Harvard University

In this note, we survey automated mechanism design (AMD): the use of computational techniques to solve mechanism design problems. We describe three distinct but overlapping threads of research: an optimization-based paradigm that formulates mechanism design as linear programming, a line of work on sample complexity and learning theory, and the recent trend of *differentiable economics*, which has produced state-of-the-art results on a range of problems by borrowing tools and techniques from modern deep learning.

1. INTRODUCTION

Automated mechanism design (AMD), construed broadly, is the use of computational techniques to find solutions to specific instances of mechanism design problems. The term was introduced in a paper by Conitzer and Sandholm [2003a] just over two decades ago. Since then, AMD has developed into a rich interdisciplinary field integrating methods from across computer science, including optimization, learning theory, and, most recently, deep learning.

There are practical reasons to care about automated mechanism design. Many organizations already use a data-driven approach for pricing and bundling decisions, and even minor improvements can have a large financial impact. For instance, recent work at Yahoo shows a revenue increase of millions of dollars from optimizing a single parameter of their ad auctions [Alcobendas et al. 2024]. Automated mechanism design is a generalization of such optimization: while to our knowledge very general automated mechanism design techniques have not yet been used to field such important mechanisms, the potential seems large.

Researchers can also make use automated mechanism design as a tool for theory. As we will discuss further below, there are many natural and interesting mechanism design problems (most notably revenue-maximizing auction design) that have resisted clean theoretical characterization. Solutions computed using automated mechanism design can act as conjectures and offer insight into problem structure.

In the following sections, as we survey the last two decades of AMD research, we will focus on three main threads. First, we will discuss the earliest approach to AMD, which frames mechanism design as an optimization problem, typically represented as a linear program (LP) (see Section 3). This approach is reasonably general, although limited to finite type spaces, and benefits from decades of advances in LP solvers. However, many mechanism design problems of interest require an unreasonably large LP.

The second thread of research connects automated mechanism design to *learn-*

ing theory (Section 4). Given a class of mechanisms, and information about the distribution of participants in the form of samples, this approach aims to select a mechanism with high estimated performance. Ideally, there is some guarantee of generalization from the set of samples to the true distribution. This is exactly the type of problem learning theory tries to tackle. Conveniently, many interesting classes of mechanisms have structural properties that can be connected back to learning-theoretic properties such as pseudodimension and Rademacher complexity.

The most recent thread of research, known as *differentiable economics* (Section 5), also treats mechanism design as a learning problem—specifically, a *deep learning* problem. In using deep learning as a tool for discovering economic mechanisms, differentiable economics is similar to efforts in the natural sciences to use deep learning for scientific discovery [Wang et al. 2023]. This approach borrows the computational tools of modern deep learning (such as composable, differentiable function approximators and gradient-descent-based optimization) and embraces a deep learning sensibility: it is relatively pragmatic and empirical, willing to relax hard guarantees in order to get better results. Despite somewhat less theoretical grounding, differentiable economics has in practice been remarkably successful, producing state-of-the-art mechanisms for a number of interesting problems.

2. MECHANISM DESIGN BACKGROUND

In this section, we give a brief and fairly standard definition of mechanism design. The purpose of this description is to provide some concreteness and to introduce notation—it is not meant to be an exhaustive or canonical formulation of mechanism design. Then, we discuss a smörgåsbord of basic theoretical results that are used in automated mechanism design.

Appealing to the *revelation principle*, we focus on *direct-revelation mechanisms*, which accept reports from each agent of their preferences over the outcome to be chosen by the mechanism (known as the *type* of the agent). We suppose agent types lie in some space $\mathcal{V}$, with outcomes in space $\mathcal{O}$. We index individual agent types as v_i and, in multi-agent settings, refer to *type profiles* $v = (v_1, \cdots, v_i, \cdots, v_n) = (v_i, v_{-i})$. We will treat types and outcomes as dual, so the welfare enjoyed for a bidder i with type v_i who receives outcome o can be thought of as computing an inner product $v_i \cdot o$. (This is fully general – following Frongillo and Kash [2021], we can consider $\mathcal{O}$ to be the space of distributions over all base outcomes, $\mathcal{V}$ to be functions on this space, and the inner product to be integrating v_i against an o.)

We focus on bidders with *quasilinear utilities*, so that bidder i's utility with type v_i for receiving outcome o and paying a monetary transfer of $p_i \in \mathbb{R}$ is $u(v_i) = v_i \cdot o - p_i$.

Definition 2.1 *Typical Mechanism Design Problem.* A mechanism design problem instance is defined by a distribution P over participant type profiles $v \in \mathcal{V}^n$ for a set of n agents, and a *performance goal* $\mathcal{L}$. The mechanism designer chooses an *allocation rule* $a : \mathcal{V}^n \to \mathcal{O}$ which maps type profiles onto outcomes, and a *payment rule* $p : \mathcal{V}^n \to \mathbb{R}^n$.

When seeking a dominant strategy equilibrium, the mechanism designer must

solve

$$\max_{a,p} \mathbb{E}_{v\sim P}\left[\mathcal{L}(a,p,v)\right] \text{ s.t.}$$
$$a(v_i, v_{-i}) \cdot v_i - p_i(v_i, v_{-i}) \geq a(b_i, v_{-i}) \cdot v_i - p_i(b_i, v_{-i}), \quad \forall v, b_i, i \quad \text{(DSIC)}$$
$$a(v_i, v_{-i}) \cdot v_i - p_i(v_i, v_{-i}) \geq 0, \quad \forall v, i \quad \text{(ex-post individual rationality)}.$$

A mechanism that satisfies the *dominant-strategy incentive compatibility (DSIC)* constraints is also called *truthful* or *strategyproof*. The DSIC constraints are sometimes weakened to *Bayesian incentive compatibility (BIC)*:

$$\mathbb{E}_{v_{-i}\sim P(\cdot|v_i)}\left[a(v_i, v_{-i}) \cdot v_i - p_i(v_i, v_{-i})\right] \geq \mathbb{E}_{v_{-i}\sim P(\cdot|v_i)}\left[a(b_i, v_{-i}) \cdot v_i - p_i(b_i, v_{-i})\right] \quad \forall v_i, b_i, i$$

In Bayesian mechanism design problems, it is common to consider the *interim mechanism* for each agent i defined by the average allocation and payment rules $\mathbb{E}_{v_{-i}\sim P(\cdot|v_i)}\left[a(v_i, v_{-i})\right]$ and $\mathbb{E}_{v_{-i}\sim P(\cdot|v_i)}\left[o_i(v_i, v_{-i})\right]$.

Remark 2.2 *Revenue-maximizing DSIC auction design.* An important special case of Definition 2.1 is revenue-maximizing DSIC auction design. Here, there are m items being sold to n bidders. The mechanism chooses either a deterministic assignment $o \in \{0,1\}^{n\times m}$ or a lottery over such assignments. In full generality, bidders may have types $v_i \in \mathbb{R}^{2^m}$ expressing their value for every bundle. In an important special case, *additive* bidders value each item individually, with their types represented by $v_i \in \mathbb{R}^m$. For additive bidders, the value of an assignment is the sum of the values for the assigned items. The auctioneer's performance goal is simply $\mathcal{L}(a,p,v) = \sum_i p_i(v_i, v_{-i})$, the total revenue.

Revenue-maximizing multi-bidder auction design is a natural problem to study and has immediate practical importance—many real-world auctions are run with the goal of maximizing revenue. Yet almost nothing analytical is known about optimal DSIC solutions to this problem, even in the seemingly simple case of additive valuations, beyond the single-item result of Myerson [1981], and the result of Yao [2017], which assumes valuation distributions with bivalued support.

This combination—little theoretical progress on a very natural and important problem—has meant that revenue-maximizing auction design has become a model problem for AMD. Since auction design has been such a major focus, we will focus on it heavily in this survey, and move somewhat freely between talk about agents (as in general mechanism design problems) and bidders (as in auction design). Of course, automated mechanism design can be used to design auctions with other goals, or to design other types of mechanisms entirely, and we will mention these other types of work where appropriate.

2.1 Characterizations of truthfulness

Here, we summarize some useful results about strategyproof mechanisms which tend to show up throughout the AMD literature.

2.1.1 *Convexity and truthfulness.* For agents with quasilinear utilities, there is a direct connection between truthfulness and convexity. The key connection is the concept of a cyclically monotone function. In convex analysis, a function is cyclically

monotone if and only if it is the (sub)gradient of some convex function (§24 of Rockafellar [1970]). It can also be shown [Rochet 1987] that the condition of cyclic monotonicity is equivalent to the DSIC constraints in Definition 2.1.[1]

The upshot of this for mechanism design is that every truthful mechanism induces a convex utility function for the mechanism participant, whose (sub)gradient is the allocation rule. Conversely, if and only if an allocation rule is cyclically monotone, it can be paired with a payment rule that will result in a truthful mechanism and a convex utility.

This relationship is defined for single-agent mechanisms. For multi-agent DSIC problems, it holds true for the mechanism faced by each bidder i holding v_{-i} fixed. For Bayesian problems, it holds true for each bidder's *interim* mechanism.

Many automated mechanism design methods ensure truthfulness by enforcing cyclic monotonicity of the allocation, or convexity of the utility.

2.1.2 *Convex conjugates and menu representations.* Recall that types v and outcomes o are dual to each other. Given a truthful mechanism with convex utility function $u(v)$, we can define the convex conjugate (§ 12 of Rockafellar [1970]) as $f^*(o) = \sup_{v \in \mathcal{V}} v \cdot o - f(v)$.

The conjugate has many nice properties (most importantly, it is always convex). In the mechanism design context, taking the conjugate $u^*(o)$ of a truthful mechanism's utility function has a natural interpretation as summarizing a *menu*: for each outcome o, $u^*(o)$ is the agent's payment, so the agent receives utility $o \cdot v_i - u^*(o)$ for the outcome. Equivalently, a truthful direct-revelation mechanism with allocation rule $a = \nabla u$ will pick the correct menu element on behalf of the agent.

2.1.3 *Affine maximizers and Roberts' theorem.* The above gives a very general characterization of truthful mechanisms. A specific class of truthful mechanisms of great importance is known as the class of *affine maximizers*. These can be seen as a generalized version of the VCG mechanism [Vickrey 1961; Clarke 1971; Groves 1973], and they inherit its nice properties.

Affine maximizers have a well-defined set of parameters that can be arbitrarily varied to optimize the mechanism design goal while always remaining within the constraints of Definition 2.1. This is why many AMD techniques restrict their search to affine maximizers.

What is lost by restricting to affine maximizers? In one sense, nothing is lost, if one must choose a mechanism that is strategyproof on an arbitrary type space. The reason is *Roberts' theorem* [Roberts 1979], which states that for a mechanism design problem on an unrestricted domain (so each agent could have any value for any outcome), with three or more outcomes, *all* DSIC mechanisms must be affine maximizers.

On the other hand, the unrestricted domain is not the right model for most mechanism design problems, so non-affine-maximizers may be truthful. For example, in an auction setting, the assumptions in Roberts' theorem would require the strange situation that bidders may have unbounded positive or negative values for receiving certain items, and moreover that the same holds for the items their opponents

[1]There is a particularly clear explanation of the connection between cyclic monotonicity and truthfulness in Börgers [2015], Chapter 5.

receive (i.e., there is the possibility of spite or altruism). Limiting mechanism design to affine maximizers therefore may mean giving up on finding the true optimal mechanism for realistic problems.

3. MATHEMATICAL OPTIMIZATION APPROACHES

The mechanism design problem (Definition 2.1) is a constrained optimization problem. In fact, if one considers a type distribution with finite support and known density, then it becomes a linear program: the objective is linear and each of the DSIC constraints is a linear constraint. This requires allowing for allocations in a continuous space; i.e., either divisible goods or randomized (so-called "lottery") allocations.

3.1 Linear programming as a computational tool

The papers that introduced the problem of automated mechanism design framed the problem in exactly this way [Conitzer and Sandholm 2003a; 2003b; 2004; Sandholm et al. 2007]. They transform the mechanism design problem into a linear program, and solve it using standard solvers. They also establish that deterministic AMD, without the randomized allocations that allow for linear programming, is NP-hard. With randomized allocations, linear programming remains a powerful tool, and work has continued using this basic approach on new problems [Guo and Conitzer 2010; Zhang and Conitzer 2021; Albert et al. 2015; Conitzer and Sandholm 2004].

3.1.1 *Discretizing the type space.* Linear programming approaches to AMD typically operate on an explicit description of a discrete type distribution, with decision variables indexed by each possible type or type profile, and translating mechanism design into linear programs poses some problems.

Large support of type distributions. It is common for type distributions to have an extremely large support. For combinatorial valuation functions (even assuming discrete possible values for each item), the number of possible types is doubly-exponential in the number of items. Even for very simple valuation structures such as additive valuations, if the type distribution has a continuous support, then approximating it on a grid will require a support whose size is exponential in the number of items. Moreover, even though the size grows "only" polynomially in the fineness of the grid, sufficiently-accurate grids even for small numbers of items create very difficult problem instances in practice [Dütting et al. 2024; Sandholm and Likhodedov 2015].

Approximation error due to discretization. Discretizing a continuous type distribution, or coarsening a discrete type distribution for tractability, unavoidably introduces error. The linear program outputs a mechanism that is defined only on the discrete or coarsened space. One can then consider rounding to the nearest discrete or coarse point to recover a mechanism in the original space, but this introduces violations of the incentive compatibility constraint. In the Bayesian mechanism design setting, there are recipes to transform approximate-BIC mechanisms into (exact) BIC mechanisms, while bounding the loss in revenue, welfare, etc. [Cai et al. 2012a; Cai et al. 2021; Conitzer et al. 2022; Daskalakis and Weinberg 2012]. Such techniques can be used to correct for the IC approximation that

is introduced by making use of discretization.

3.2 Mechanism design and duality

Of course, linear programs have duals, and thinking about duality has been theoretically fruitful in mechanism design. A line of several papers [Cai et al. 2012b; 2012a] lays out a reduction from multi-bidder *Bayesian* mechanism design to a single-bidder problem, culminating in a duality framework for finite-support type distributions [Cai et al. 2019]. Giannakopoulos and Koutsoupias [2018] also describe a duality framework for mechanism design and derive an auction format they call the *Straight-Jacket Auction*, which they can prove optimal for some cases.

Daskalakis et al. [2017], and some followup work [Kash and Frongillo 2016], formulate *single-bidder* mechanism design as dual to an optimal transport problem, which, as explicated by Kleiner and Manelli [2019], is equivalent to an infinite-dimensional linear programming problem. The primal problem is a search over a subset of convex functions corresponding to feasible mechanisms (as discussed briefly in Section 2.1.1). The structure of the dual provides useful information about optimal solutions, and in some cases dual solutions can certify optimality of mechanisms. This is theoretically useful and also motivates some of the differentiable economics approaches discussed in Section 5.

Kolesnikov et al. [2022] generalize this optimal transport approach to multi-bidder *Bayesian* mechanism design. Using their duality result combined with the idea of many-to-one bidder reduction and a bag of tricks from other works [Cai et al. 2012a; Alaei et al. 2019; Kleiner et al. 2021], they are able to formulate and explicitly solve a linear program to find an optimal interim auction for multiple bidders and multiple items. Many of the aforementioned papers reduced mechanism design to linear programming to prove theoretical results about tractability; Kolesnikov et al. [2022] uses many of those advances to numerically solve actual linear programs.

4. AUTOMATED MECHANISM DESIGN AND LEARNING THEORY

If one only has sample access to the type distribution, then the mechanism design problem (Definition 2.1) becomes a learning problem: the goal is to choose a function from some parameterized class (the class of feasible mechanisms) to optimize some loss (the mechanism design objective), hopefully generalizing from the training samples to the true distribution.

There are several advantages to this perspective. Requiring only samples from the distribution avoids the problems of large support mentioned in Section 3. Further, it is perhaps more natural to imagine that a mechanism designer has access to historical data from the population of bidders, rather than perfect knowledge of the population distribution. Moreover, there is already an extremely rich body of learning techniques and one can seek to apply these to mechanism design. In this section, we focus on a line of work motivated by "classical" machine learning and using techniques which come with strong generalization guarantees.

4.1 Single-parameter settings

Much work has focused on *single-parameter* (e.g., selling one item) auction design settings, where the Myerson auction is known to be optimal and strategyproof. The learning problem is to choose a function from the family of feasible Myerson

auctions. In the case of regular distributions, this just means choosing the Myerson reserve price; for more general distributions it may be more involved (e.g., Roughgarden and Schrijvers [2016]). This has been an enormously fruitful line of research and we cannot do it justice here, but we recommend the article of Guo et al. [2020] from a previous issue of SIGEcom Exchanges for a comprehensive overview.

4.2 Multi-parameter settings

In multi-parameter settings, there are currently no general characterizations of strategyproof mechanisms that lead to sample-efficient learning, but it is usually possible to restrict the learning problem to some nicer class of multi-parameter strategyproof mechanisms.

Learning simple auctions. A long line of work focuses on learning within classes of so-called *simple* auctions (e.g., combinations of Myerson auctions). It can sometimes be established that the best auction in some class of simple auctions gives a constant-factor approximation of the optimal revenue [Chawla et al. 2010; Hajiaghayi et al. 2007]. Moreover, it is sometimes possible to learn from samples efficiently over such classes [Morgenstern and Roughgarden 2016; 2015; Balcan et al. 2008; Feldman et al. 2014; Hsu et al. 2016].

Learning affine maximizers. As mentioned above, for completely general type spaces, only affine maximizers are strategyproof due to Roberts' theorem [Roberts 1979], and moreover affine maximizers form a parameterized class of auctions in a very convenient way. This motivates work on learning affine maximizers. The general idea is that functions—such as the revenue function—induced by affine maximizers have very nice structural properties that allow bounding their pseudodimension or Rademacher complexity—complexity measures in learning theory that can be used to bound the generalization error of learning from training samples. Sample complexity results are even better for certain restricted yet interesting subclasses of affine maximizers. The techniques can also be applied to classes of simple auctions [Balcan et al. 2016; 2018; Balcan et al. 2018].

5. DIFFERENTIABLE ECONOMICS

Differentiable economics uses tools from modern deep learning to solve problems in mechanism design. Like the aforementioned works in Section 4, it frames the mechanism design problem as a learning problem over samples from the type space; however, it focuses more on practical performance and flexibility rather than strictly on computational complexity and generalization guarantees. While some works do present sample complexity results [Dütting et al. 2024; Kuo et al. 2020], the main goal of differentiable economics is to use data-driven optimization to explore mechanism design in practice, assuming sufficient sample access for training complex mechanisms effectively.

At the broadest level, differentiable economics refers to using a gradient-based search process to optimize within some parameterized, differentiable class of mechanisms. Why might one want to take this approach to automated mechanism design? Modern function approximators for deep learning are both flexible and composable. If part of a problem has poorly-understood structure, one can simply use a universal

neural network of some kind; on the other hand, if information on special structure is available, or if there are particular problem constraints, this information can often be baked into the architecture. Due to the use of end-to-end gradient-based optimization, it is easy to freely combine and compose different architectural components.

The computational tools— first-order optimizers, autograd libraries such as TensorFlow [Abadi et al. 2015], Jax [Bradbury et al. 2018], and PyTorch [Ansel et al. 2024], and so on—are well-maintained and pleasant to use due to the enormous resources poured into their development by tech companies. And, as has been a surprise to the larger research community, even though in principle they are non-convex and should be intractable, solving deep learning problems using stochastic first-order methods just turns out to work really well, and this is equally true when the application happens to be mechanism design.

5.1 Introducing differentiable economics

Dütting et al. [2024] tackled the problem of optimal DSIC auction design and introduced the term "differentiable economics." One can extract a core recipe which remains useful for a very wide range of mechanism design problems (Definition 2.1):

(1) Use a flexible neural network to model the allocation (and payment) rules of the mechanism a, p as a function of type profiles; train the network on samples $v = (v_1, \ldots, v_i, \ldots, v_n) \sim P$ from the type distribution to maximize the performance goal $\mathcal{L}$ (e.g., revenue).
(2) Wherever possible, enforce problem constraints by careful design of the network architecture. In the world of deep learning, this is known as imposing an *inductive bias*.
(3) For other constraints where this is not possible, search for constraint violations (often by optimizing the network inputs) and use these violations to calculate a penalty term in the loss function.

Given that a direct-revelation mechanism is a function taking type profiles as inputs and outputting an outcome, differentiable function approximators can be used to represent mechanisms for many problem settings, and their flexibility often means that natural constraints of the problem can be designed into the architecture. This is therefore a very general-purpose recipe.

Dütting et al. [2024] introduce multiple concrete methods for learning auctions. One, known as RegretNet, works for auctions with any number of items and bidders, and imposes less inductive bias. Another, known as RochetNet, specializes to the single-bidder setting and is therefore able to impose more inductive bias to enforce IC. (There is also a third, MyersonNet, for single-item auctions, which we do not focus on here.)

5.1.1 *Multi-bidder auctions: RegretNet.* RegretNet (Figure 1) uses feedforward neural networks to represent the allocation and payment rules.

The input is the type profile. The output of the allocation rule is a matrix, with activation functions that ensure no item will ever be over-sold. The output of the payment network is a number between zero and one expressing the *fraction* of their received welfare to recover from each player as a payment; this, the allocation,

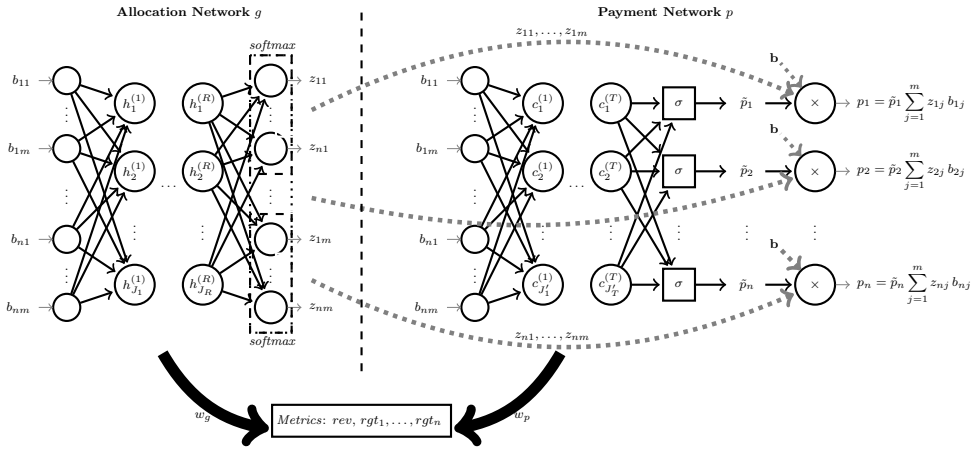

Fig. 1. A schematic of the RegretNet architecture, adapted from Dütting et al. [2024], for a setting with n buyers and m items. The neural network takes in bids as inputs and outputs both allocation and payment. Relevant metrics, such as regret and revenue, are then computed from these outputs and optimized by adjusting the network parameters through gradient descent.

and the types suffice to calculate the actual payment, which will never exceed a bidder's value for the items they get. These architectural design choices introduce an inductive bias, ensuring that individual rationality and item supply constraints are inherently satisfied.

DSIC, however, is not explicitly hardcoded into the architecture. Instead, it is enforced as a constraint during training, by penalizing *regret*: the expected maximum increase in utility that a bidder can achieve by misreporting their true value (in other words, the expected magnitude of the DSIC constraint violation). For a given truthful type profile, regret can be estimated by keeping the weights of the partially trained RegretNet fixed, and using gradient ascent on the bids themselves to maximize each bidder's utility from an untruthful report. Including the regret penalty in the training loss gradually drives the bidders' regret for truthful bidding to near zero.

The RegretNet approach is quite powerful and can be used to find high-performing mechanisms that are empirically almost perfectly strategyproof, and where there is often good reason to believe they are very close to the true optimal mechanisms. There is a problem, however. If an approximately DSIC mechanism is deployed, even a small (in terms of regret/utility) violation in DSIC may cause a larger shift in bidding behavior. Even worse, and as discussed in Curry et al. [2020], there may be very large (in terms of regret/utility) DSIC violations that only show up for a tiny proportion of types and may not be observed by the measurement methodology in Dütting et al. [2024]. Despite the empirical evidence that the mechanisms learned via differentiable economics tend to be near to truly strategyproof mechanisms, many mechanism designers—including those with both theoretical and practically-motivated interests—would prefer stronger guarantees.

5.1.2 *Exact DSIC for single bidders – RochetNet.* Fortunately, Dütting et al. [2024] also present an approach for a special case. *Single-bidder* DSIC mechanisms are well-characterized: as described in Section 2.1, they can be identified with convex, utility functions.

Dütting et al. [2024] use this characterization to design a single-bidder architecture that they call *RochetNet*, a schematic of which is shown in Figure 2. The utility function being convex (DSIC constraint), and non-decreasing and 1-Lipschitz (constrained to feasible allocations), is enforced at the architectural level.

Concretely, the learnable parameters of the network consist of a large number of menu elements, with each element specifying a bundle with the associated price. A null option with all-zero allocation and zero payment is added as a fixed menu element to ensure individual ratonality.

Given a bidder type as the input, the utility of each menu element can be evaluated, and a max operation is applied over all menu elements to select the highest-value item for the bidder, therefore ensuring exact IC. At training time, the non-differentiable max operation is approximated by a *softmax*. The menu elements, all being learnable parameters, are optimized by gradient descent. Because DSIC constraints are enforced via inductive bias in the architecture, the optimization proceeds in a straightforward way with no penalties or search for misreports.

Shen et al. [2019] give an architecture similar in practice to RochetNet, but they focus more on the conjugate interpretation (also discussed in Section 2.1). Dütting et al. [2024] and Shen et al. [2019] are both further able to build on the optimal transport formulation of mechanism design (Section 3.2) to formally prove the optimality of some mechanisms learned by their techniques.

5.2 Work related to Dütting et al. [2024]

RegretNet and RochetNet are two prototypical models which can help to think about related work — does a method enforce all constraints by inductive bias, like RochetNet, or does it have some penalty terms?

5.2.1 *Precursors to differentiable economics.* Conitzer and Sandholm [2007] design mechanisms by starting from a non-strategyproof mechanism, searching for counterexamples to strategyproofness, and repeatedly modifying the mechanism in response – similar in spirit to RegretNet. Narasimhan et al. [2016] and Dütting et al. [2015] use non-deep-learning ML techniques to find good mechanisms for problem settings with and without money. Sandholm and Likhodedov [2015] use a hill-climbing approach to optimize parameters of affine maximizer auctions.

5.2.2 *Further progress on auction design.* Some works have followed Dütting et al. [2024], focusing on the same auction problems and using essentially the same penalty-based training paradigm. Some involve improvements to the training algorithm [Rahme et al. 2021]. Others involve improvements to the architecture [Ivanov et al. 2022; Duan et al. 2022], both for the sake of regret/revenue performance and to allow for imposing additional inductive biases such as symmetry (which corresponds to bidder-anonymity).

Still others extend to auctions with new types of constraints: Feng et al. [2018] considers budget-constrained bidders, Tacchetti et al. [2022] optimizes for total participant welfare instead of revenue, Kuo et al. [2020] imposes a fairness constraint

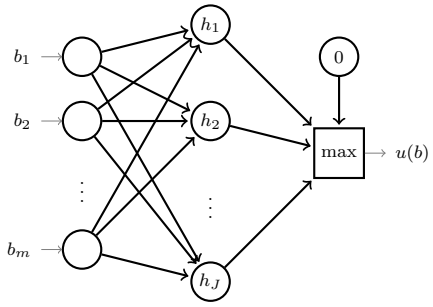

Fig. 2. A schematic of the RochetNet Architecture reproduced from Dütting et al. [2024]. RochetNet encodes the convex utility function associated with a truthful mechanism.

on the learned mechanisms (via training penalty), and Curry et al. [2022] extends to no-free-disposal settings where the number of items allocated to each bidder is enforced (via the architecture). Ravindranath et al. [2024] extends the RochetNet approach to sequential posted-menu mechanisms through the use of deep reinforcement learning.

5.2.3 *Problems beyond auctions.* Ravindranath et al. [2021] applies a RegretNet-like approach to two-sided matching, in particular to explore the space of mechanisms which trade off between the incompatible goals of stability and strategyproofness. Golowich et al. [2018] studies multi-facility location mechanism design, another mechanism design problem without payments. Ravindranath et al. [2023] studies the problem of data market design in economic theory, where the goal is to find a set of signaling schemes (statistical experiments) to maximize expected revenue to the information seller. Curry et al. [2024] extend the RochetNet approach to find optimal automated market makers, and extend the duality theory to prove them optimal. Wang et al. [2024] study contract design, and use specially-designed deep nets to model the utility function of the principal. A little further afield from these methods, Hossain et al. [2024] study the problem of multi-sender Bayesian persuasion.

5.2.4 *Affine maximizer auctions (again): a multi-bidder extension of RochetNet.* Affine maximizers are guaranteed to provide truthful, multi-bidder mechanisms, and as mentioned in Section 4, they have been used successfully for automated mechanism design and have well-understood learning-theoretic properties. They are also a good fit for a differentiable-economics inspired approach. In fact, if one considers the special case of an auction with one bidder, an affine maximizer is essentially just a menu, and in this sense this is one natural, multi-bidder generalization of RochetNet.

Curry et al. [2022] introduce this approach and find it works well. A particular advantage is that, by learning end-to-end, one can consider outcomes in the space of lotteries in addition to deterministic outcomes. A followup work takes advantage of the power of differentiable economics by adding attention layers everywhere, and finds increased performance [Duan et al. 2023]. Curry et al. [2024] further extend the approach to dynamic mechanism design problems, where multiple allocative decisions must be made over time.

Dütting et al. [2024] and Curry et al. [2022] observed that *overparameterization*

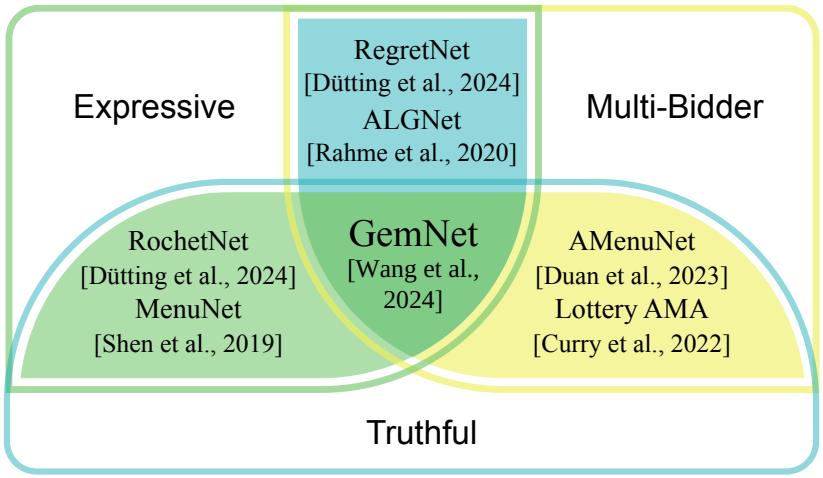

Fig. 3. The "trilemma" of current automated mechanism design reproduced from Wang et al. [2024]. Previously, no architecture had been exactly strategyproof, fully expressive, and multi-agent. GemNet is the first approach to achieve all three goals.

was useful during training of both RochetNet and affine maximizers, even though the learned mechanisms (in some cases, provably optimal) use very few menu elements by the end. By appealing to a concept from deep learning theory called *mode connectivity*, Hertrich et al. [2024] give a partial explanation for this phenomenon.

5.3 The current state of the art – GemNet

The aforementioned approaches demonstrate that perfectly strategyproof methods are achievable within differentiable economics. However, in the realm of multi-bidder auctions, which are particularly central to automated mechanism design, these methods must compromise either on full expressiveness or on exact strategyproofness. Wang et al. [2024] overcome this dilemma (see Figure 3) and give the current state-of-the-art method for differentiable economics: *GEneral Menu-based NETwork (GemNet)*.

GemNet extends the RochetNet architecture to learn a multi-bidder menu network that maps, for each bidder i, opponent bids v_{-i} onto a menu of allocations and payments for bidder i. As discussed in Section 2, as long as bidder i always chooses their favorite menu element, the mechanism will be strategyproof.

The problem is that if each bidder freely chooses their favorite menu element, the result might be that some items are over-sold. Some obvious ways present themselves to fix this (take into account other bidders' choices when choosing the menu, or adding a RegretNet-like activation to the allocation), but these turn out to destroy strategyproofness.

The main innovation in GemNet is to accommodate multi-bidder and multi-item settings by overcoming this problem, ensuring *menu compatibility*: even though bidders independently and freely choose the elements from their respective independent menus, the menus are designed such that their combined choices will not result in over-allocation of any item.

As a first step, the GemNet menu network is trained to optimize a combination of revenue and an *incompatibility loss*, which penalizes menus that could result in the over-allocation of items when bidders independently select their utility-maximizing menu elements. Although the incompatibility loss does not entirely eliminate menu incompatibility, it significantly reduces the likelihood of over-allocation.

Since any strategyproof mechanism can be represented by self-bid independent menus with bidder-optimization [Hammond 1979], for a network with sufficient capacity (i.e., enough hidden layers and nodes), GemNet's menu-based computational framework is without loss of generality, in that it can in principle learn the revenue-optimal auction with access to sufficient training data.

Exact menu compatibility in practice is achieved in a second step through a price adjustment algorithm involving mixed-integer linear programming (MILP), which fine-tunes menu prices post-training and pre-deployment to enforce compatibility across the entire continuous value domain. The approach leverages Lipschitz continuity of neural networks to extend compatibility from discrete grid points to continuous spaces, ensuring menu compatibility over the full domain and strategyproofness.

Empirically, GemNet demonstrates improved performance in various auction settings, outperforming existing methods such as AMAs in terms of revenue while achieving exact strategyproofness. Moreover, GemNet exhibits better interpretability through clear decision boundaries and agrees with known optimal solutions in specific scenarios. Figure 4 gives a nice illustration of this latter point.

Relationship with LP-based AMD methods. GemNet requires solving a series of mixed-integer linear programs (MILPs), which can be computationally demanding. Fortunately, because the trained networks typically exhibit only a minimal rate of over-allocation after training, it is often possible to preserve bidders' original choices in the trained menu (prior to any adjustments) for the majority of grid points. This strategy, among others, significantly reduces both the number of binary variables and the computational time needed to solve the MILPs (a decrease in running time of more than 99.99% in some settings). The size of the MILPs are substantially smaller than the size of the LPs that would be required in solving the same problem of AMD through earlier methods (as per Section 3). Moreover, although GemNet's price-adjustment method can in principle be adopted to a MILP-only framework for strategyproof mechanisms in continuous value domains, the use of deep learning is crucial in enabling local, "single-bidder focused" formulations.

6. CONCLUSION

6.1 Fruitful directions for future work

An even better characterization of strategyproof mechanisms. GemNet is fully expressive (i.e., it can represent any feasible mechanism, by appeal to universal approximation theorems for neural networks) and always exactly DSIC. But some parameter settings can represent infeasible mechanisms, hence the requirement of the post-training transformation. The post-training transformation is computationally costly, but designing an architecture that does not require it would probably require new theoretical breakthroughs in characterizing strategyproof mechanisms on restricted domains. With this improved understanding of feasible mechanisms,

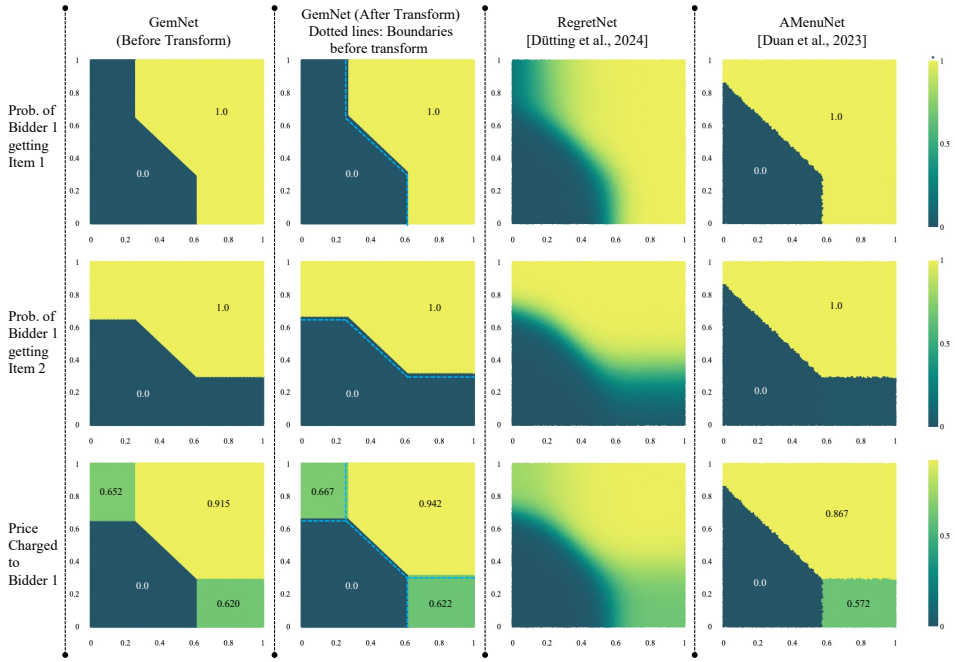

Fig. 4. This figure reproduced from Wang et al. [2024] shows the mechanism learned by several differentiable economics techniques — GemNet itself both before and after the price adjustments, RegretNet [Dütting et al. 2024], and the AMenuNet affine maximizer architecture [Duan et al. 2023]. The problem setting is selling 2 items to 2 additive agents with uniform type distributions on $[0, 1]$ — a problem setting for which the optimal auction is not known. The plots show the allocation rule for agent 1, holding agent 2's bid fixed. One can see that GemNet learns a mechanism with clear decision boundaries, while RegretNet appears to learn a smoother approximation of the same mechanism. The affine maximizer, meanwhile, learns a simpler mechanism, likely due to the reduced expressiveness of affine maximizers.

a differentiable economics pipeline could directly generate likely candidates for optimal and strategyproof mechanisms.

Sample-efficient differentiable economics. There are some generalization bounds provided in the work discussed in Section 5; e.g., in Dütting et al. [2024]. But those papers do not really treat sample efficiency as a major concern: most papers assume that it is easy to get at least hundreds of thousands of samples from the valuation distribution. This will not always be true, especially if these techniques are to be applied on real-world data.

Getting meaningful generalization bounds for deep learning is not easy; learning theory tends be too pessimistic. However, the architectures used for differentiable economics can be relatively small and can have special structure, so bridging the gap with the learning-theoretic work discussed in Section 4 could be possible and advantageous.

Automatically proving optimality. Differentiable economics has been used to find some optimal mechanisms [Dütting et al. 2024; Shen et al. 2019; Curry et al. 2024].

The idea is to search for a conjectured mechanism and then find a solution to a corresponding dual problem to certify optimality. Right now, these dual problems are solved by hand in an *ad hoc* way, which is both challenging and unreliable. An algorithm to attack the dual problem, paired with the existing techniques for searching through the primal space, could be extremely useful. And access to bounds from the dual during the optimization process could prove helpful for solving the primal problem more quickly.

Putting automated mechanism design to work for theorists. On the whole, these tools are not yet easy-to-use or reliable enough that a theoretical economist or mechanism designer would reach for them when studying a new problem. But progress on this front continues, and when automated mechanism design is truly mature, it should be a common part of the toolkit even for people who have no intrinsic interest in the computational techniques and would prefer to do as much as possible on a blackboard. One can imagine many scenarios: using automated mechanism design to generate conjectures, to explore the space of feasible solutions, to try to *falsify* conjectures, and much more. Progress here would involve both improvements in reliable training, easy-to-use software packages, and, crucially, visualization or other means of understanding learned mechanisms. There has been some partial progress on this front—moving from totally black-box neural networks to architectures that output interpretable menus—but the problem remains challenging, especially for mechanisms whose outputs live in higher-dimensional spaces that are intrinsically hard to visualize.

6.2 Wrapping up

There has been slow but steady progress on automated mechanism design in the decades since it was first introduced. The central idea shared by all automated mechanism design work is that mechanism design problems can be made to look a lot like other familiar optimization or learning problems from computer science, for which powerful computational tools already exist that can be brought to bear on mechanism design. The earliest work observed that mechanism design is a constrained optimization problem and so took an optimization-based approach. This is a very natural formulation and remains useful, but although solving LPs is efficient, the size of the LP formulations tend to scale badly (and require discrete type spaces). Other work has observed that mechanism design looks like a machine learning problem—choosing a function from some class, evaluating its performance on samples, and giving high-probability bounds on the estimation error. Most recently, work in *differentiable economics* brings to bear the techniques and pragmatic sensibilities of modern deep learning. This means giving up on some guarantees, but has also resulted in state-of-the-art results, including the discovery of new optimal mechanisms and interesting insights on very challenging problems.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Alaei, S., Fu, H., Haghpanah, N., Hartline, J., and Malekian, A. 2019. Efficient Computation of Optimal Auctions via Reduced Forms. *Mathematics of Operations Research 44,* 3 (Aug.), 1058–1086.

Albert, M., Conitzer, V., and Lopomo, G. 2015. Assessing the Robustness of Cremer-McLean with Automated Mechanism Design. *Proceedings of the AAAI Conference on Artificial Intelligence 29,* 1 (Feb.).

Alcobendas, M., Ji, J., Gokulakannan, H., Wami, D., Kapchits, B., Duteil, E. P., Satow, K., Roman, M. R. L., Diaz, O., Jr, A. A. D., and Kavoori, R. 2024. Optimizing Floors in First Price Auctions: An Empirical Study of Yahoo Advertising.

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Suo, M., Tillet, P., Wang, E., Wang, X., Wen, W., Zhang, S., Zhao, X., Zhou, K., Zou, R., Mathews, A., Chanan, G., Wu, P., and Chintala, S. 2024. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM.

Balcan, M.-F., Blum, A., Hartline, J. D., and Mansour, Y. 2008. Reducing mechanism design to algorithm design via machine learning. *Journal of Computer and System Sciences 74,* 8, 1245–1270.

Balcan, M.-F., Dick, T., and Vitercik, E. 2018. Dispersion for Data-Driven Algorithm Design, Online Learning, and Private Optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. 603–614.

Balcan, M.-F., Sandholm, T., and Vitercik, E. 2018. A General Theory of Sample Complexity for Multi-Item Profit Maximization. In *Proceedings of the 2018 ACM Conference on Economics and Computation*. ACM, Ithaca NY USA, 173–174.

Balcan, M.-F. F., Sandholm, T., and Vitercik, E. 2016. Sample complexity of automated mechanism design. *Advances in Neural Information Processing Systems 29*.

Börgers, T. 2015. *An introduction to the theory of mechanism design*. Oxford University Press, USA.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. 2018. JAX: composable transformations of Python+NumPy programs.

Cai, Y., Daskalakis, C., and Weinberg, S. M. 2012a. An algorithmic characterization of multi-dimensional mechanisms. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*. STOC '12. Association for Computing Machinery, New York, NY, USA, 459–478.

Cai, Y., Daskalakis, C., and Weinberg, S. M. 2012b. Optimal Multi-dimensional Mechanism Design: Reducing Revenue to Welfare Maximization. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. 130–139.

Cai, Y., Devanur, N. R., and Weinberg, S. M. 2019. A duality-based unified approach to bayesian mechanism design. *SIAM Journal on Computing 50,* 3, STOC16–160.

Cai, Y., Oikonomou, A., Velegkas, G., and Zhao, M. 2021. An efficient ϵ-bic to bic transformation and its application to black-box reduction in revenue maximization. In *Proceedings of*

the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA). Proceedings. Society for Industrial and Applied Mathematics, 1337–1356.

Chawla, S., Hartline, J. D., Malec, D. L., and Sivan, B. 2010. Multi-parameter mechanism design and sequential posted pricing. In *Proceedings of the forty-second ACM symposium on Theory of computing*. 311–320.

Clarke, E. H. 1971. Multipart pricing of public goods. *Public choice*, 17–33.

Conitzer, V., Feng, Z., Parkes, D. C., and Sodomka, E. 2022. Welfare-preserving ε-bic to bic transformation with negligible revenue loss. In *Web and Internet Economics*, M. Feldman, H. Fu, and I. Talgam-Cohen, Eds. Springer International Publishing, Cham, 76–94.

Conitzer, V. and Sandholm, T. 2003a. Automated mechanism design: Complexity results stemming from the single-agent setting. In *Proceedings of the 5th International Conference on Electronic Commerce*. ICEC '03. Association for Computing Machinery, New York, NY, USA, 17–24.

Conitzer, V. and Sandholm, T. 2003b. Automated mechanism design for a self-interested designer. In *Proceedings of the 4th ACM Conference on Electronic Commerce*. EC '03. Association for Computing Machinery, New York, NY, USA, 232–233.

Conitzer, V. and Sandholm, T. 2004. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the 5th ACM Conference on Electronic Commerce*. EC '04. Association for Computing Machinery, New York, NY, USA, 132–141.

Conitzer, V. and Sandholm, T. 2007. Incremental Mechanism Design. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Curry, M., Sandholm, T., and Dickerson, J. 2022. Differentiable Economics for Randomized Affine Maximizer Auctions. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Curry, M., Thoma, V., Chakrabarti, D., McAleer, S., Kroer, C., Sandholm, T., He, N., and Seuken, S. 2024. Automated Design of Affine Maximizer Mechanisms in Dynamic Settings. *Proceedings of the AAAI Conference on Artificial Intelligence 38,* 9 (Mar.), 9626–9635.

Curry, M. J., Chiang, P.-Y., Goldstein, T., and Dickerson, J. 2020. Certifying Strategyproof Auction Networks. In *Neural Information Processing Systems*.

Curry, M. J., Fan, Z., and Parkes, D. C. 2024. Optimal automated market makers: Differentiable economics and strong duality. *arXiv preprint arXiv:2402.09129*.

Curry, M. J., Lyi, U., Goldstein, T., and Dickerson, J. P. 2022. Learning Revenue-Maximizing Auctions With Differentiable Matching. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 6062–6073.

Daskalakis, C., Deckelbaum, A., and Tzamos, C. 2017. Strong duality for a multiple-good monopolist. *Econometrica 85,* 3.

Daskalakis, C. and Weinberg, S. M. 2012. Symmetries and optimal multi-dimensional mechanism design. In *Proceedings of the 13th ACM Conference on Electronic Commerce*. ACM, Valencia Spain, 370–387.

Duan, Z., Sun, H., Chen, Y., and Deng, X. 2023. A Scalable Neural Network for DSIC Affine Maximizer Auction Design. *Advances in Neural Information Processing Systems 36*, 56169–56185.

Duan, Z., Tang, J., Yin, Y., Feng, Z., Yan, X., Zaheer, M., and Deng, X. 2022. A context-integrated transformer-based neural network for auction design. In *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. PMLR, 5609–5626.

Dütting, P., Feng, Z., Narasimhan, H., Parkes, D. C., and Ravindranath, S. S. 2024. Optimal auctions through deep learning: Advances in differentiable economics. *Journal of the ACM 71,* 1, 1–53.

Dütting, P., Fischer, F., Jirapinyo, P., Lai, J. K., Lubin, B., and Parkes, D. C. 2015. Payment Rules through Discriminant-Based Classifiers. *ACM Trans. Econ. Comput. 3,* 1 (Mar.), 5:1–5:41.

Feldman, M., Gravin, N., and Lucier, B. 2014. Combinatorial auctions via posted prices. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 123–135.

Feng, Z., Narasimhan, H., and Parkes, D. C. 2018. Deep learning for revenue-optimal auctions with budgets. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. 354–362.

Frongillo, R. M. and Kash, I. A. 2021. General truthfulness characterizations via convex analysis. *Games and Economic Behavior 130*, 636–662.

Giannakopoulos, Y. and Koutsoupias, E. 2018. Duality and optimality of auctions for uniform distributions. *SIAM Journal on Computing 47,* 1, 121–165.

Golowich, N., Narasimhan, H., and Parkes, D. C. 2018. Deep learning for multi-facility location mechanism design. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 261–267.

Groves, T. 1973. Incentives in teams. *Econometrica: Journal of the Econometric Society*, 617–631.

Guo, C., Huang, Z., and Zhang, X. 2020. Sample complexity of single-parameter revenue maximization. *ACM SIGecom Exchanges 17,* 2 (Jan.), 62–70.

Guo, M. and Conitzer, V. 2010. Computationally Feasible Automated Mechanism Design: General Approach and Case Studies. *Proceedings of the AAAI Conference on Artificial Intelligence 24,* 1 (July), 1676–1679.

Hajiaghayi, M. T., Kleinberg, R., and Sandholm, T. 2007. Automated online mechanism design and prophet inequalities. In *AAAI*. Vol. 7. 58–65.

Hammond, P. J. 1979. Straightforward individual incentive compatibility in large economies. *The Review of Economic Studies 46,* 2, 263–282.

Hertrich, C., Tao, Y., and Végh, L. A. 2024. Mode connectivity in auction design. *Advances in Neural Information Processing Systems 36*.

Hossain, S., Wang, T., Lin, T., Chen, Y., Parkes, D. C., and Xu, H. 2024. Multi-sender persuasion–a computational perspective. *arXiv preprint arXiv:2402.04971*.

Hsu, J., Morgenstern, J., Rogers, R., Roth, A., and Vohra, R. 2016. Do prices coordinate markets? In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 440–453.

Ivanov, D., Safiulin, I., Filippov, I., and Balabaeva, K. 2022. Optimal-er Auctions through Attention. *Advances in Neural Information Processing Systems 35*, 34734–34747.

Kash, I. A. and Frongillo, R. M. 2016. Optimal auctions with restricted allocations. In *Proceedings of the 2016 ACM Conference on Economics and Computation*. 215–232.

Kleiner, A. and Manelli, A. 2019. Strong Duality in Monopoly Pricing. *Econometrica 87,* 4, 1391–1396.

Kleiner, A., Moldovanu, B., and Strack, P. 2021. Extreme points and majorization: Economic applications. *Econometrica 89,* 4, 1557–1593.

Kolesnikov, A. V., Sandomirskiy, F., Tsyvinski, A., and Zimin, A. P. 2022. Beckmann's approach to multi-item multi-bidder auctions. arXiv:2203.06837.

Kuo, K., Ostuni, A., Horishny, E., Curry, M. J., Dooley, S., Chiang, P.-y., Goldstein, T., and Dickerson, J. P. 2020. Proportionnet: Balancing fairness and revenue for auction design with deep learning. *arXiv preprint arXiv:2010.06398*.

Morgenstern, J. and Roughgarden, T. 2016. Learning Simple Auctions. In *Conference on Learning Theory*. PMLR, 1298–1318.

Morgenstern, J. H. and Roughgarden, T. 2015. On the pseudo-dimension of nearly optimal auctions. In *Advances in Neural Information Processing Systems*.

Myerson, R. B. 1981. Optimal Auction Design. *Mathematics of Operations Research 6,* 1 (Feb.).

Narasimhan, H., Agarwal, S. B., and Parkes, D. C. 2016. Automated mechanism design without money via machine learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*.

Rahme, J., Jelassi, S., and Weinberg, S. M. 2021. Auction learning as a two-player game. In *International Conference on Learning Representations*.

Ravindranath, S. S., Feng, Z., Li, S., Ma, J., Kominers, S. D., and Parkes, D. C. 2021. Deep Learning for Two-Sided Matching. arXiv:2107.03427.

Ravindranath, S. S., Feng, Z., Wang, D., Zaheer, M., Mehta, A., and Parkes, D. C. 2024. Deep reinforcement learning for sequential combinatorial auctions. arXiv:2407.08022.

Ravindranath, S. S., Jiang, Y., and Parkes, D. C. 2023. Data market design through deep learning. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds. Vol. 36. Curran Associates, Inc., 6662–6689.

Roberts, K. 1979. The characterization of implementable choice rules. *Aggregation and revelation of preferences 12,* 2, 321–348.

Rochet, J.-C. 1987. A necessary and sufficient condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics 16,* 2 (Jan.), 191–200.

Rockafellar, R. T. 1970. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J.

Roughgarden, T. and Schrijvers, O. 2016. Ironing in the Dark. In *Proceedings of the 2016 ACM Conference on Economics and Computation*. EC '16. Association for Computing Machinery, New York, NY, USA, 1–18.

Sandholm, T. and Likhodedov, A. 2015. Automated Design of Revenue-Maximizing Combinatorial Auctions. *Operations Research 63,* 5 (Oct.), 1000–1025.

Sandholm, T. W., Conitzer, V., and Boutilier, C. 2007. Automated Design of Multistage Mechanisms. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Shen, W., Tang, P., and Zuo, S. 2019. Automated Mechanism Design via Neural Networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

Tacchetti, A., Strouse, D., Garnelo, M., Graepel, T., and Bachrach, Y. 2022. Learning truthful, efficient, and welfare maximizing auction rules. In *ICLR 2022 Workshop on Gamification and Multiagent Solutions*.

Vickrey, W. 1961. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance 16,* 1, 8–37.

Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., Deac, A., et al. 2023. Scientific discovery in the age of artificial intelligence. *Nature 620,* 7972, 47–60.

Wang, T., Duetting, P., Ivanov, D., Talgam-Cohen, I., and Parkes, D. C. 2024. Deep contract design via discontinuous networks. *Advances in Neural Information Processing Systems 36*.

Wang, T., Jiang, Y., and Parkes, D. C. 2024. GemNet: Menu-Based, Strategy-Proof Multi-Bidder Auctions Through Deep Learning. In *Proceedings of the 2024 ACM Conference on Economics and Computation*.

Yao, A. C.-C. 2017. Dominant-Strategy versus Bayesian Multi-item Auctions: Maximum Revenue Determination and Comparison. In *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM, Cambridge Massachusetts USA, 3–20.

Zhang, H. and Conitzer, V. 2021. Automated Dynamic Mechanism Design. In *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 27785–27797.