

Mechanism Design: A Mathematical Programming Framework for Principal-Agent Systems*

I Nyoman Sutapa¹, Laksmi Prita Wardhani², Magdalena Wullur³, and Mohammad Sukri Bin Mustapa⁴

Abstract—Mechanism design is often introduced as a reverse form of game-theoretic analysis because the designer starts from a desired objective and then constructs rules under which strategic agents choose outcomes consistent with that objective. This paper develops a mathematical programming framework for principal-agent mechanism design under asymmetric information. The framework treats allocation and transfer rules as decision variables and imposes incentive compatibility, individual rationality, and feasibility as explicit constraints, consistent with the optimization-based view of mechanism design. The proposed mathematical formulation is intended to be directly implementable as a finite-dimensional optimization model when types and actions are discretized. The paper makes three contributions. First, it clarifies the relationship between the reverse game-theoretic interpretation and classical mechanism design, and inverse game-theoretic learning. Second, it formulates a direct-mechanism optimization model and derives its finite-type LP and MILP specialization. Third, it discusses continuous type spaces through finite-grid approximation and provides a general sensitivity interpretation for finite networks. A two-agent binary-type example is used not as a new auction result, but as a benchmark recovery test showing that the optimization template recovers a second-price auction with a reserve price from incentive and participation constraints. The framework is relevant for systems in which incentives and outcomes must be designed jointly.

Index Terms – Mechanism Design, Revers Game Theory, Mathematical Programming

I. INTRODUCTION

IN a standard non-cooperative game, the analyst begins with players, strategies, information, and payoff functions, and then studies equilibrium behavior [1], [2]. Mechanism design reverses this analytical direction: the designer begins with a desired objective, such as welfare, revenue, compliance, or efficient allocation, and constructs a mechanism that induces strategic agents to reveal information or choose actions consistent with that objective [3], [4], [5]. For this reason, mechanism design is often described informally as reverse game theory.

The phrase is pedagogically useful, but it is not a separate established theory; it should be understood as a modeling

orientation within the broader mechanism design literature [3], [2].

The present paper adopts that orientation without treating it as a rebranding of classical mechanism design. The contribution is computational and methodological. We formulate the principal-agent design problem as a mathematical program in which allocation and transfer rules are decision variables, while incentive compatibility (IC), individual rationality (IR), feasibility, and implementation constraints appear as explicit inequalities [3], [6], [7]. The resulting structure is useful for applied computing contexts because it can be converted into a linear program (LP) or mixed-integer linear program (MILP) when type spaces and allocation alternatives are finite [6], [8], [9].

Classical mechanism design provides the theoretical foundation for this paper. The revelation principle allows the analysis to focus on direct mechanisms in which agents truthfully report their private types when the mechanism is incentive compatible [5], [7]. Optimal auction theory, implementation theory, and principal-agent analysis then provide conditions under which desired social choice rules can be implemented [4], [2]. However, many classical treatments are analytical and problem-specific. They often emphasize existence, monotonicity, envelope conditions, or equilibrium implementation, while the computational form of the designer's problem remains implicit or is presented only for specific auction environments [3], [6].

At the same time, algorithmic mechanism design and automated mechanism design have made computation central to incentive design [10], [8], [9]. These fields study mechanisms under algorithmic, informational, and computational constraints, but their models are often specialized to auctions, combinatorial allocation, or network resource allocation [8], [10]. Inverse game theory and inverse reinforcement learning move in a different direction by inferring latent payoffs, utilities, or reward parameters from observed behavior or demonstrations [11], [12], [13]. The framework proposed here is positioned between these literatures. It keeps the IC/IR structure of mechanism design, writes the design task as an optimization problem, and leaves room for parameters to be estimated from data in future learning-based extensions [6], [11], [12].

The paper addresses the concern that a purely finite formulation may appear too narrow. The revised version therefore distinguishes three levels of analysis. The first is the general direct-mechanism problem over allocation and transfer functions [3], [5], [7]. The second is a finite-type specialization suitable for LP or MILP implementation [6], [8], [9]. The

*This work was not supported by any organization.

¹I N. Sutapa is with the Industrial Engineering Department, Petra Christian University, Surabaya 60236, Indonesia. mantapa@petra.ac.id

²L. P. Wardhani is with the Mathematics Department, Sepuluh Nopember Institute of Technology, Surabaya 60111, Indonesia.

³M. Wullur is with the Management Department, Sam Ratulangi University, Manado 95115, Indonesia.

⁴M. S. B. Mustapa is with the Faculty of Mechanical and Manufacturing Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Johor Darul Ta'zim, Malaysia.

Corresponding author: I Nyoman Sutapa (mantapa@petra.ac.id).

Manuscript received January 19, 2026; accepted May 29, 2026.

third is a continuous-type interpretation in which finite grids approximate an infinite-dimensional problem under regularity assumptions [14]. This hierarchy clarifies that the finite model is not the full theory, but a computational representation of a broader mechanism design problem.

The remainder of the paper is organized as follows. Section 2 positions the work relative to classical and computational mechanism design. Section 3 presents the mathematical programming framework, including the direct mechanism formulation, the finite-type LP/MILP form, and the continuous-type approximation. Section 4 presents a benchmark numerical example and develops sensitivity analysis. Section 5 concludes with implications, limitations, and future research directions.

II. RELATED WORK AND CONTRIBUTION

Theoretical mechanism design begins from the problem of implementing social objectives when agents possess private information [4], [5]. Myerson's auction model, Maskin's implementation framework, and subsequent principal-agent models show that allocation and payment rules must be designed together because strategic agents respond to incentives rather than to the designer's objective directly [2], [4]. In this tradition, IC and IR are not optional modeling details; they are the mathematical representation of strategic truthfulness and voluntary participation [3], [6], [7].

Algorithmic mechanism design extends this tradition by incorporating computational complexity and algorithmic feasibility into mechanism design [10]. It asks how welfare, revenue, or allocation objectives can be achieved when the optimal mechanism is computationally hard, when communication is limited, or when allocation constraints are combinatorial [8], [10]. Automated mechanism design further treats the mechanism itself as an output of an optimization procedure [9]. These approaches are close to the spirit of the present paper, but the present paper emphasizes a compact principal-agent template that is accessible to applied mathematics readers and can be specialized to several domains without committing to one auction format [6], [15].

Inverse game theory and inverse reinforcement learning are also relevant because they reverse the conventional direction of game analysis. Rather than predicting behavior from known payoffs, they infer utilities, cost functions, or reward parameters from observed behavior [11], [12], [13]. The framework in this paper does not solve an inverse estimation problem. Instead, it designs a mechanism after the designer specifies the objective, feasible allocations, and informational constraints. Nevertheless, the optimization template can be coupled with inverse learning methods when type distributions, utility parameters, or cost coefficients are estimated from data [11], [12].

The contribution of this paper can be stated precisely. First, it clarifies the meaning of reverse game-theoretic mechanism design as a descriptive orientation rather than a new field [2], [3], [5]. Second, it presents a principal-agent mathematical program in which allocation and transfer variables, IC constraints, IR constraints, and feasibility constraints are visible in one model [6], [7]. Third, it shows how the finite-type

version reduces to LP or MILP form and how continuous-type problems can be approximated through finite grids [6], [14]. Fourth, it reframes a familiar two-agent auction result as a validation test for the optimization template and extends the sensitivity discussion beyond the binary example [16].

III. METHODS

A. Principal-Agent Environment

Consider a principal interacting with a finite set of agents $N = 1, \dots, n$. Each agent i has a private type θ_i in a type space Θ_i . The type profile is $\theta = (\theta_1, \dots, \theta_n) \in \Theta = \Theta_1 \times \dots \times \Theta_n$. The principal knows a prior distribution F over Θ , while each agent observes its own type. A mechanism asks agents to send messages and maps reported messages into allocations and transfers [3], [2].

An allocation is denoted by $x \in X$, where X may represent assignment decisions, procurement quantities, audit intensity, service access, platform exposure, or other controllable outcomes. Transfers are denoted by $t = (t_1, \dots, t_n)$, where t_i is the payment received by agent i . A negative transfer can represent a payment from the agent to the principal. The utility of agent i is assumed to be quasi-linear: [3], [6].

$$u_i(x, t_i, \theta_i) = v_i(x, \theta_i) + t_i \quad (1)$$

The principal's payoff is represented by

$$\Pi(x, t, \theta) = R(x, \theta) - C(x, \theta) - \sum_{i \in N} t_i \quad (2)$$

where R denotes revenue or social benefit and C denotes implementation cost. The specific interpretation depends on the application. In a procurement problem, t_i may be a supplier payment; in an audit problem, x may represent the audit allocation and t_i an incentive or penalty; in a digital platform, x may represent access, visibility, or matching decisions [6], [15].

B. Direct Mechanisms, IC, and IR

By the revelation principle, if an outcome can be implemented by some mechanism under suitable equilibrium conditions, then it can be represented by a direct mechanism in which agents report their types truthfully. A direct mechanism is a pair $(x(\bullet), t(\bullet))$ consisting of an allocation rule $x : \Theta \rightarrow X$ and a transfer rule $t : \Theta \rightarrow R^n$ [5], [7].

For Bayesian incentive compatibility, truthful reporting must maximize each agent's expected utility given the distribution of other agents' types. For each agent i and for every true type θ_i and possible misreport θ'_i , the IC condition is [3], [6], [7].

$$\begin{aligned} E_{\theta_{-i}} [v_i(x(\theta_i, \theta_{-i}), \theta_i) + t_i(\theta_i, \theta_{-i})] \\ \geq E_{\theta_{-i}} [v_i(x(\theta'_i, \theta_{-i}), \theta_i) + t_i(\theta'_i, \theta_{-i})] \end{aligned} \quad (3)$$

Individual rationality requires that truthful participation provide at least the reservation utility $r_i(\theta_i)$: [2], [7].

$$E_{\theta_{-i}} [v_i(x(\theta_i, \theta_{-i}), \theta_i) + t_i(\theta_i, \theta_{-i})] \geq r_i(\theta_i) \quad (4)$$

Equations (3) and (4) are introduced once and used throughout the paper. They are the strategic and participation constraints of the mathematical program [6], [7].

C. Reverse Design as a Mathematical Program

In the reverse design formulation, the allocation rule and the transfer rule are treated as decision variables. The designer chooses them to maximize expected principal payoff subject to feasibility, IC, and IR. The general problem is [6].

$$\max_{x(\bullet), t(\bullet)} E_{\theta} [\Pi(x(\theta), t(\theta), \theta)] \quad (5)$$

subject to $x(\theta) \in X(\theta)$ for all θ , and subject to the IC and IR constraints in (3) and (4). The feasible set $X(\theta)$ may include capacity limits, assignment constraints, budget restrictions, technology constraints, or policy restrictions [6], [8], [9].

Two variants are useful. In the first, both $x(\bullet)$ and $t(\bullet)$ are optimized. This is the full mechanism design problem. In the second, the designer fixes a target allocation rule $x^*(\bullet)$ and optimizes only the transfer rule $t(\bullet)$ to implement it. This second variant is particularly useful when a regulator, platform, or organization has already specified a desired allocation principle, such as efficiency, fairness, compliance priority, or sustainability, and needs payments or incentives that make the target implementable [3], [6], [7].

This formulation is not intended to replace classical analytical mechanism design. Instead, it turns the same theoretical ingredients into an optimization model. This computational representation is useful because feasibility, incentive constraints, and design objectives can be passed directly to optimization solvers in finite settings [6].

D. Finite-Type Specialization

Suppose each type set Θ_i is finite. Let $P(\theta)$ denote the probability of type profile θ . For each θ , define allocation variables $x(\theta)$ and transfer variables $t_i(\theta)$. If the feasible allocation set is discrete, binary variables may be introduced. For example, $y_{ij}(\theta) = 1$ may indicate that agent i receives object or task j under type profile θ [6], [8], [9].

The objective becomes

$$\max \sum_{\theta \in \Theta} P(\theta) \left[R(x(\theta), \theta) - C(x(\theta), \theta) - \sum_i t_i(\theta) \right] \quad (6)$$

The IC constraints become finite systems of linear inequalities whenever utilities are linear in allocation and transfers. For each i , θ_i , and θ'_i , [6].

$$\begin{aligned} & \sum_{\theta_{-i}} P(\theta_{-i} | \theta_i) [v_i(x(\theta_i, \theta_{-i}), \theta_i) + t_i(\theta_i, \theta_{-i})] \\ & \geq \sum_{\theta_{-i}} P(\theta_{-i} | \theta_i) [v_i(x(\theta'_i, \theta_{-i}), \theta_i) + t_i(\theta'_i, \theta_{-i})] \end{aligned} \quad (7)$$

The IR constraints are similarly finite. If x variables are continuous and $X(\theta)$ is polyhedral, the model is an LP. If some

allocation decisions are binary or integer-valued, the model becomes a MILP. This finite specialization is the primary computational object in the paper [6], [8], [9].

E. Continuous Type Spaces and Finite Approximation

Continuous type spaces are central in mechanism design. If θ_i belongs to a compact interval or multidimensional compact set, the problem in (5) becomes an infinite-dimensional optimization problem over functions $x(\bullet)$ and $t(\bullet)$. Classical approaches often use monotonicity, envelope theorems, and virtual valuation arguments to characterize implementable mechanisms. The mathematical programming approach can still be used, but the functions must be approximated numerically [6], [14].

Let G_k be a finite grid over Θ with mesh size h_k , where $h_k \rightarrow 0$. A finite mechanism on G_k assigns allocations and transfers to grid points. The continuous distribution F is approximated by a discrete distribution F_k over the grid. IC is imposed on grid types, and approximate IC can be extended to off-grid types by interpolation. Under compactness of Θ , bounded and Lipschitz utilities, continuity of feasible allocations, and regularity of the target allocation rule, the error in expected payoff and IC violation can be bounded by a term proportional to the grid mesh. In this sense, the finite program can be interpreted as a discretized approximation of the continuous design problem [14].

This statement should be read as a regularity-based approximation principle rather than a universal convergence theorem. If utilities are discontinuous, if feasibility changes abruptly, or if the continuous problem has nonunique or unstable optima, finite approximations may converge slowly or may require additional constraints. Therefore, the paper treats finite discretization as an implementable computational method and identifies formal convergence analysis as a future research direction [14].

F. Implementation Steps for LP/MILP Models

The finite framework can be implemented through the following steps. First, enumerate all type profiles and assign probabilities. Second, define allocation and transfer variables for each profile. Third, add feasibility constraints reflecting physical, budgetary, or institutional limits. Fourth, generate IC constraints for every agent, true type, and misreport. Fifth, generate IR constraints for every agent and type. Sixth, define the expected payoff objective. Seventh, solve the resulting LP or MILP and evaluate whether the solution is robust to parameter changes [6], [8], [9].

The number of IC constraints grows with the number of agents and type alternatives. If agent i has m_i types, then agent i contributes $m_i(m_i - 1)$ Bayesian IC comparisons. The number of type profiles is the product of the type-set sizes. This scaling explains why decomposition, constraint generation, and approximation methods become important in larger networks [8], [9], [10].

IV. RESULTS AND DISCUSSION

A. Benchmark Recovery Example: Two Agents and Binary Types

The numerical example (Table 1) on benchmark recovery with two agents and binary types. Its purpose is not to introduce a new auction result, but to validate the optimization template by showing that it recovers a well-known mechanism from IC, IR, feasibility, and objective constraints. Consider one principal selling one indivisible object to at most one of two agents. Each agent has type L or H , with valuations $0 < v_L < v_H$. Types are independent, and the probability of a high type is p . The principal incurs cost c when the object is allocated [6], [16].

The target allocation is efficient: if one agent reports H and the other L , the high type receives the object; if both report the same type, the object is assigned randomly with equal probability. Transfers are chosen to maximize expected principal profit subject to IC and IR. The loser pays zero [16].

Let $q_i(\theta)$ be the probability that agent i receives the object under profile θ , and let $p_i(\theta)$ be the payment made by agent i to the principal. Using payments to the principal rather than transfers to agents, agent i 's utility is $q_i(\theta)v_i - p_i(\theta)$. The finite program chooses payments so that truthful reporting is optimal. Solving the IC and IR system gives a payment rule equivalent to a second-price auction with reserve price v_L : the winner pays the maximum of the other reported valuation and the reserve price [6], [16].

This recovery is important for validation. If the optimization template failed to reproduce a canonical mechanism in this benchmark case, it would not be credible for more complex principal-agent networks. The example therefore serves the same role as a calibration test in computational modeling [16].

B. Expected Profit and Local Sensitivity

The expected revenue in the benchmark example is

$$\begin{aligned} E[\text{Revenue}] &= v_L [(1-p)^2 + 2p(1-p)] + v_H p^2 \\ &= v_L (1-p^2) + v_H p^2 \end{aligned} \quad (8)$$

Since the object is always allocated under the efficient rule, expected cost is c . Expected profit is

$$E[\text{Profit}] = v_L (1-p^2) + v_H p^2 - c \quad (9)$$

The local derivative with respect to p is

$$\frac{dE(\text{Profit})}{dp} = 2p(v_H - v_L) \quad (10)$$

which is nonnegative when $v_H \geq v_L$. The derivative with respect to cost is -1 because the object is allocated in every state. These expressions are useful pedagogically, but they are only the simplest case of a more general parametric optimization structure [6].

C. General Sensitivity in Finite Networks

In a larger finite-type network, the optimized objective can be written as the value function of a parametric LP or MILP. Let $z(\theta; \alpha)$ be the principal's payoff contribution under type profile θ and parameter vector α , which may include costs, valuations, capacity levels, or policy weights. If the optimal allocation and active constraint regime remain unchanged locally, the derivative of expected payoff with respect to the probability $P(\theta)$ is simply the optimized payoff contribution $z(\theta; \alpha)$. This gives a direct interpretation: type profiles with high payoff contributions become more valuable as their probability increases [6].

For cost parameters, the derivative is the negative expected use of the costly resource. If c_j is the unit cost of resource j and $a_j(\theta)$ is the amount used under type profile θ , then locally

$$\partial V / \partial c_j = - \sum_{\theta \in \Theta} P(\theta) a_j(\theta) \quad (11)$$

When the active constraint set changes, the value function is generally piecewise linear for LPs and piecewise constant or piecewise linear over feasible regimes for MILPs. In such cases, global comparative statics require re-solving the model over a parameter grid or using parametric programming. The important point is that the optimization formulation makes sensitivity analysis computationally accessible, whereas a purely verbal mechanism design description does not [6], [8], [9].

For continuous distributions, the analogous statement is expressed through the expected payoff integral. A perturbation of the distribution changes the objective by changing the weight assigned to each type profile. Under differentiability and regularity assumptions, envelope-type arguments can be used to evaluate local sensitivity while holding the optimal policy fixed to first order. Formal global comparative statics in continuous spaces, however, require additional assumptions and are left for future work [14].

D. Methodological Value Beyond the Benchmark Auction

The benchmark example reproduces a second-price auction with a reserve price, which is a well-known result in optimal auction design [3]. The value of the framework lies in the way the result is obtained. The model begins with feasibility, IC, IR, and an objective, and then recovers the payment rule as the solution of an optimization problem [6], [7]. This matters when the designer's environment is no longer a textbook auction.

For example, in procurement, suppliers may differ in cost, quality reliability, emission intensity, and capacity. Allocation may be constrained by service-level requirements or sustainability targets. In auditing, auditees may possess private information about process quality or compliance effort, and the principal may need to design audit intensity and incentives to elicit truthful disclosure. In digital platforms, the allocation may represent visibility, matching, or access, while transfers may represent fees, rewards, or penalties. In these settings, the mechanism is shaped by multiple constraints that are difficult to solve analytically but natural to represent as an LP or MILP [6], [8], [15].

TABLE I
TYPE PROFILES, PROBABILITIES, ALLOCATIONS, AND BENCHMARK PAYMENTS

Type profile	Probability	Efficient allocation	Winner payment
(L, L)	$(1 - p)^2$	Random assignment, probability 1/2 each	v_L
(H, L)	$p(1 - p)$	Agent 1 receives object	v_L
(L, H)	$(1 - p)p$	Agent 2 receives object	v_L
(H, H)	p^2	Random assignment, probability 1/2 each	v_H

The optimization framework also facilitates extensions. Fairness constraints, budget balance, risk aversion, robustness, and data-driven estimation can be added as additional constraints or objective terms [15], [17]. This provides a bridge between classical mechanism design and modern computational decision models [6], [8], [9], [10].

E. Limitations

Several limitations should be stated explicitly. First, the finite-type MILP formulation can become large as the number of agents and type levels increases; therefore, constraint generation or decomposition may be needed for scalability [8], [9], [10], [15]. Second, the continuous-type approximation discussion provides a regularity-based computational pathway but not a universal convergence theorem [14]. Third, the benchmark example validates the formulation but does not by itself prove superiority over existing auction theory [16]. Fourth, the paper assumes quasi-linear utilities; settings with risk aversion, robustness concerns, budget constraints, or behavioral preferences require additional modeling [17]. These limitations define a research agenda rather than a weakness of the basic framework.

V. CONCLUSIONS

This paper has clarified a mathematical programming framework for mechanism design in principal-agent systems under asymmetric information. The term reverse game-theoretic mechanism design is used only as a descriptive orientation: the designer starts from a desired objective and constructs allocation and transfer rules that satisfy strategic and participation constraints. The paper does not claim that reverse game theory is a new field separate from mechanism design.

The main contribution is a compact optimization representation of the design problem. Allocation and transfer rules are modeled as decision variables, while IC, IR, and feasibility are imposed as constraints. For finite type spaces, the model becomes an LP or MILP, making it suitable for computational implementation. For continuous type spaces, finite-grid approximation provides a practical route under appropriate regularity conditions, although formal convergence guarantees require further study.

The two-agent binary-type example was reframed as a benchmark recovery test. It shows that the optimization template can recover the second-price auction with a reserve price from the constraint system. The paper also extends the sensitivity analysis beyond the example by explaining how finite

network models can be studied through parametric LP/MILP value functions and expected resource-use derivatives.

Practically, the framework can support mechanism design in procurement, auditing, reverse logistics, digital platform governance, and other systems where incentives and allocations must be co-designed. Future research should develop scalable algorithms, formal convergence results for continuous approximations, robust and risk-averse variants, and empirical or simulation-based case studies in applied domains.

REFERENCES

- [1] E. E. C. Damme, "Game theory: Noncooperative games," in *International Encyclopedia of the Social & Behavioral Sciences*, Elsevier, 2015, pp. 582–591. doi: 10.1016/B978-0-08-097086-8.71048-8.
- [2] Y. Narahari, *Game Theory and Mechanism Design*, vol. 4. World Scientific, 2014. doi: 10.1142/8902.
- [3] T. Börgers, *An introduction to the theory of mechanism design*. Oxford University Press, 2015. doi: 10.1093/acprof:oso/9780198731535.001.0001.
- [4] E. S. Maskin, "Mechanism design: How to implement social goals," *Am. Econ. Rev.*, vol. 98, no. 3, pp. 567–576, 2008, doi: 10.1257/aer.98.3.567.
- [5] R. B. Myerson, "Mechanism Design," in *The New Palgrave Dictionary of Economics*, London: Palgrave Macmillan, 2008, pp. 1–13. doi: 10.1057/978-1-349-95121-5_2675-1.
- [6] R. V. Vohra, *Mechanism design: a linear programming approach*. Cambridge University Press, 2011. doi: https://doi.org/10.1017/CBO9780511979243.
- [7] R. B. Myerson, "Revelation principle," in *The New Palgrave Dictionary of Economics*, London: Palgrave Macmillan, 2018, pp. 11646–11652. doi: 10.1057/978-1-349-95121-5_2358-1.
- [8] V. Conitzer and T. Sandholm, "Complexity of mechanism design," in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, San Francisco, 2002. doi: 10.48550/arXiv.1408.1486.
- [9] V. Conitzer and T. Sandholm, "Automated mechanism design: Complexity results stemming from the single-agent setting," in *Proceedings of the 5th ACM Conference on Electronic Commerce*, 2004, pp. 17–24. doi: 10.1145/988772.988776.
- [10] N. Nisan and A. Ronen, "Algorithmic mechanism design," *Games Econ. Behav.*, vol. 35, no. 1–2, pp. 166–196, 2001, doi: 10.1006/game.1999.0790.
- [11] V. Kuleshov and O. Schrijvers, "Inverse game theory: Learning utilities in succinct games," in *International Conference on Web and Internet Economics*, Berlin, Heidelberg: Springer, 2015, pp. 413–427. doi: 10.1007/978-3-662-48995-6_30.
- [12] K. Cao and L. Xie, "Game-theoretic inverse reinforcement learning: A differential Pontryagin's maximum principle approach," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 34, no. 11, pp. 9506–9513, 2022, doi: 10.1109/TNNLS.2022.3152313.
- [13] L. Cui, H. Yu, P. Pinson, and D. Paccagnan, "Inverse game theory: an incentive-based approach," in *34th International Joint Conference on Artificial Intelligence, IJCAI 2025*, 2025, pp. 3805–3813. doi: 10.24963/ijcai.2025/423.
- [14] Y. Tao and H. Xu, "Generalized Bayesian Nash equilibrium with continuous type and action spaces," *SIAM J. Optim.*, vol. 35, no. 2, pp. 789–817, 2025, doi: 10.1137/24M1653859.
- [15] A. Sinha and A. Anastasopoulos, "Incentive mechanisms for fairness among strategic agents," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 288–301, 2017, doi: 10.1109/JSAC.2017.2659061.

- [16] R. B. Myerson, "Perspectives on mechanism design in economic theory," *Am. Econ. Rev.*, vol. 98, no. 3, pp. 586–603, 2008, doi: 10.1257/aer.98.3.586.
- [17] G. Carroll, "Robustness in mechanism design and contracting," *Annu. Rev. Econom.*, vol. 11, no. 1, pp. 139–166, 2019.