Algoritmi cu predicţii in sisteme multiagent

Gabriel Istrate

Universitatea din Bucureşti

gabriel.istrate@unibuc.ro



UNIVERSITY OF BUCHAREST

VIRTUTE ET SAPIENTIA

Ce cautã din nou un teoretician la un Workshop de Inteligenţă Artificialã, la Diaspora Ştiinţificã ?

Los Alamos NATIONAL LABORATORY

Ex-"diaspora", ’94-’07 (Rochester, Los Alamos)

- Cariera ştiinţificã "de bazã": **algoritmi şi complexitate** (dar şi sisteme complexe, simulãri multiagent).

- Din 2019: teoria algoritmicã a jocurilor şi sisteme multiagent. **4 lucrãri (format complet/BlueSky) la AAMAS**. lucrãri @TARK, @SAT, etc.

- 2025: singura lucrare .ro in format complet la EUMAS’2025

Interesat de colaborãri: teoria algoritmicã a jocurilor, sisteme multiagent, metode (neuro)simbolice, SAT/constrângeri, etc.

Rezultatele despre care vã vorbesc

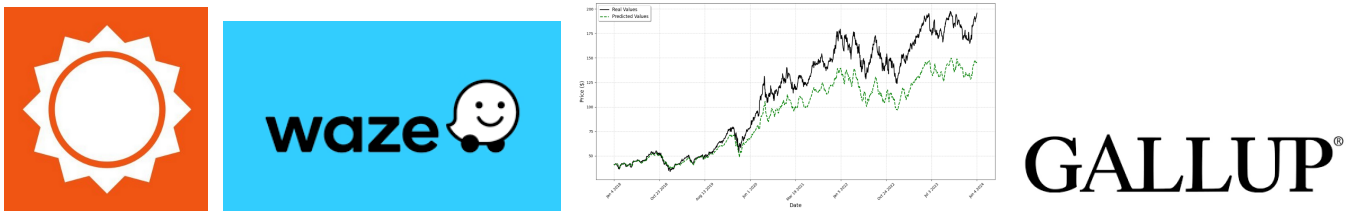
- Publicate la EUMAS'2025 + douã lucrãri in curs de recenzare la AAMAS'2026
- Deja citate in NEURIPS'2024, EC'2024, AAAI'2025, NEURIPS'2025, WWW'2025, IJCAI'2025, SODA'2025 (intre altele). Intre cei care citeazã: Toby Walsh (UNSW), Anupam Gupta (NYU)

- Prezentarea: nivel intuitiv, fãrã ~~detalii tehnice~~ matematicã, simplificând/vulgarizând.



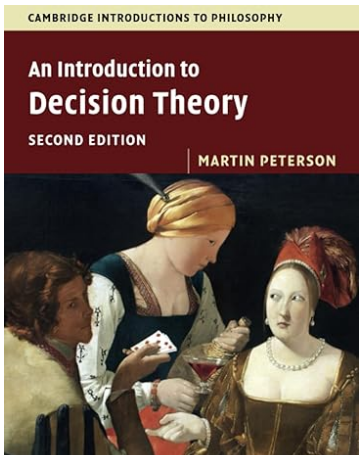

Cosmin Bonchiş şi Victor Bogdan (Univ. Vest. Timişoara)

Predictions in Everyday Life



Prediction-informed Decision:

- Predictors: not necessarily good/trusted.
- Predictor: may have its own goals/act strategically.
- How do we act optimally, given the predictor?



"Normal" Decision Theory:

- action maximizing expected utility.
- no role for (such) predictions!

Most importantly ...

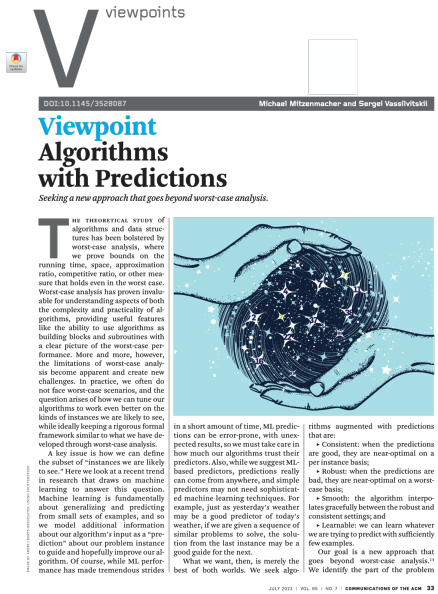


- Our students: increasingly outsource thinking to predictive technologies.
- Students' decision model: trust/follow the predictor. Let's not replicate this with software!
- **Yesterday, Cornelia Caragea:** "... deep learning training models were overconfident ..."

Ideal Scenario:

- Predictor perfect: Choice improves on the predictorless case.
- Predictor really bad: do no worse than without predictions.

Context: algorithms with predictions



viewpoints

DOI:10.1145/3528087 Michael Mitzenmacher and Sergei Vassilvitskii

Viewpoint
Algorithms with Predictions

Seeking a new approach that goes beyond worst-case analysis.

THE THEORETICAL STUDY of algorithms and data structures has been bolstered by worst-case analysis, where we prove bounds on the running time, space, approximation ratio, competitive ratio, or other measure that holds even in the worst case. Worst-case analysis has proven invaluable for understanding aspects of both the complexity and practicality of algorithms, providing useful features like the ability to use algorithms as building blocks and subroutines with a clear picture of the worst-case performance. More and more, however, the limitations of worst-case analysis become apparent and create new challenges. In practice, we often do not face worst-case scenarios, and the question arises of how we can tune our algorithms to work even better on the kinds of instances we are likely to see, while ideally keeping a rigorous formal framework similar to what we have developed through worst-case analysis.

A key issue is how we can define the subset of "instances we are likely to see." Here we look at a recent trend in research that draws on machine learning to answer this question. Machine learning is fundamentally about generalizing and predicting from small sets of examples, and so we model additional information about our algorithm's input as a "prediction" about our problem instance to guide and hopefully improve our algorithm. Of course, while ML performance has made tremendous strides in a short amount of time, ML predictions can be error-prone, with unexpected results, so we must take care in how much our algorithms trust their predictors. Also, while we suggest ML-based predictors, predictions really can come from anywhere, and simple predictors may not need sophisticated machine learning techniques. For example, just as yesterday's weather may be a good predictor of today's weather, if we are given a sequence of similar problems to solve, the solution from the last instance may be a good guide for the next.

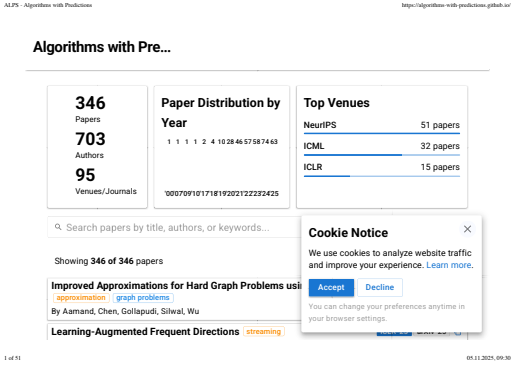
What we want, then, is merely the best of both worlds. We seek algorithms augmented with predictions that are:

- Consistent: when the predictions are good, they are near-optimal on a per instance basis;
- Robust: when the predictions are bad, they are near-optimal on a worst-case basis;
- Smooth: the algorithm interpolates gracefully between the robust and consistent settings; and
- Learnable: we can learn whatever we are trying to predict with sufficiently few examples.

Our goal is a new approach that goes beyond worst-case analysis.[1] We identify the part of the problem

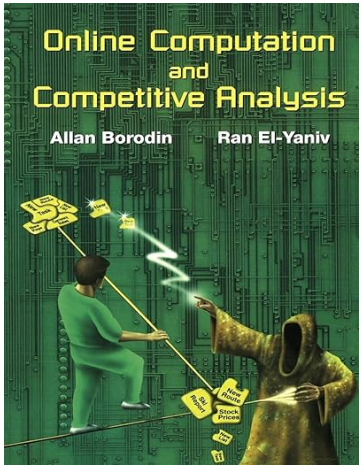
JULY 2022 | VOL. 65 | NO. 7 | COMMUNICATIONS OF THE ACM 33

- Mitzenmacher & Vassilvitskii Communications of the A.C.M. (2022)
- Assume: **predictor (ML ?)** for algorithmic problem.
- Want: algorithms that take advantage of the predictor, improve upon ordinary algorithms



Algorithms with predictions webpage:
https://algorithms-with-predictions.github.io/

This talk: Online algorithms



- Algorithm: fights not knowing the future.
- Quality measure: competitive ratio.
- Compare algorithm's performance against the optimal algorithm when knowing the future on worst-case possible input

$$c_A = sup_x \frac{c_A(x)}{c_{OPT}(x)}$$

Algorithms with Predictions: Measures



Improving Online Algorithms via ML Predictions

Ravi Kumar
Google
Mountain View, CA
ravi.k53@gmail.com

Manish Purohit
Google
Mountain View, CA
mpurohit@google.com

Zoya Svitkina
Google
Mountain View, CA
zoya@cs.cornell.edu

Abstract

In this work we study the problem of using machine-learned predictions to improve the performance of online algorithms. We consider two classical problems, ski rental and non-clairvoyant job scheduling, and obtain new online algorithms that use predictions to make their decisions. These algorithms are oblivious to the performance of the predictor, improve with better predictions, but do not degrade much if the predictions are poor.

1 Introduction

Dealing with uncertainty is one of the most challenging issues that real-world computational tasks, besides humans, face. Ranging from "will it snow next week?" to "should I rent an apartment or buy a house?", there are questions that cannot be answered reliably without some knowledge of the future. Similarly, the question of "which job should I run next?" is hard for a CPU scheduler that does not know how long this job will run and what other jobs might arrive in the future.

There are two interesting and well-studied computational paradigms aimed at tackling uncertainty. The first is in the field of machine learning where uncertainty is addressed by making predictions about the future. This is typically achieved by examining the past and building robust models based on the data. These models are then used to make predictions about the future. Humans and real-world applications can use these predictions to adapt their behavior: knowing that it is likely to snow next week can be used to plan a ski trip. The second is in the field of algorithm design. Here, the effort has to been to develop a notion of competitive ratio[1] for the goodness of an algorithm in the presence of an unknown future and develop online algorithms that make decisions heedless of the future but are provably good in the *worst-case*, i.e., even in the most pessimistic future scenario. Such online algorithms are popular and successful in real-world systems and have been used to model problems including paging, caching, job scheduling, and more (see the book by Borodin and El-Yaniv [5]).

Recently, there has been some interest in using machine-learned predictions to improve the quality of online algorithms [20, 18]. The main motivation for this line of research is two-fold. The first is to design new online algorithms that can avoid assuming a worst-case scenario and hence have better performance guarantees both in theory and practice. The second is to leverage the vast amount of modeling work in machine learning, which precisely deals with how to make predictions. Furthermore, as machine-learning models are often retrained on new data, these algorithms can naturally adapt to evolving data characteristics. When using the predictions, it is important that the online algorithm is unaware of the performance of the predictor and makes no assumptions on the types of prediction errors. Additionally, we desire two key properties of the algorithm: (i) if the predictor is good, then the online algorithm should perform close to the best offline algorithm (*consistency*) and (ii) if the predictor is bad, then the online algorithm should gracefully degrade, i.e., its performance should be close to that of the online algorithm without predictions (*robustness*).

[1]Informally, competitive ratio compares the worst-case performance of an online algorithm to the best offline algorithm that knows the future.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

- **consistency:** competitive ratio for **perfect** predictors.
- **robustness:** competitive ratio for **really bad predictions.**
- Kumar, Purohit, Svitkina, (NEURIPS 2018)
- Some followup work: Gollapudi & Panigrahi (ICML 2019), Banerjee (NEURIPS 2020), etc.

Usually: **tradeoff consistency/robustness**, algorithms that realize optimal tradeoff.

Example: ski-rental

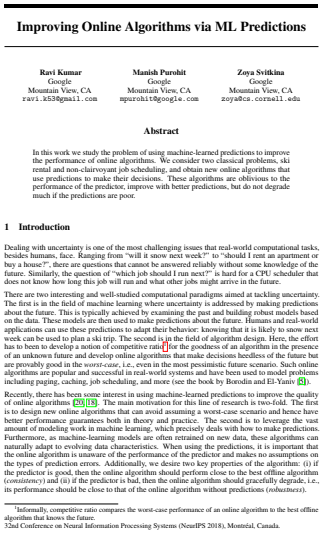
- Want to ski an unknown number of days T.
- rent: \$1/day. Buy: \$$B$.

- Optimal deterministic algorithm (with no predictions): buy on day B, competitive ratio $2 - \frac{1}{B}$
- Optimal randomized algorithm (with no predictions): competitive ratio $e/(e-1)$

Sample practical applications:

- cloud computing (A. Khanafer, M. Kodialam, K. Puttaswamy. The constrained Ski-Rental problem and its application to online cloud cost optimization, INFOCOM (2013)).
- energy scheduling Lee, R., Maghakian, J., Hajiesmaili, M., Li, J., Sitaraman, R., & Liu, Z. (2021). Online peak-aware energy scheduling with untrusted advice. ACM SIGENERGY Energy Informatics Review, 1(1), 59-77.

Single-agent ski-rental with predictions



Improving Online Algorithms via ML Predictions

Ravi Kumar
Google
Mountain View, CA
ravi.k53@gmail.com

Manish Purohit
Google
Mountain View, CA
mpurohit@google.com

Zoya Svitkina
Google
Mountain View, CA
zoya@cs.cornell.edu

Abstract

In this work we study the problem of using machine-learned predictions to improve the performance of online algorithms. We consider two classical problems, ski rental and non-clairvoyant job scheduling, and obtain new online algorithms that use predictions to make their decisions. These algorithms are oblivious to the performance of the predictor, improve with better predictions, but do not degrade much if the predictions are poor.

1 Introduction

Dealing with uncertainty is one of the most challenging issues that real-world computational tasks, besides humans, face. Ranging from "will it snow next week?" to "should I rent an apartment or buy a house?", there are questions that cannot be answered reliably without some knowledge of the future. Similarly, the question of "which job should I run next?" is hard for a CPU scheduler that does not know how long this job will run and what other jobs might arrive in the future.

There are two interesting and well-studied computational paradigms aimed at tackling uncertainty. The first is in the field of machine learning where uncertainty is addressed by making predictions about the future. This is typically achieved by examining the past and building robust models based on the data. These models are then used to make predictions about the future. Humans and real-world applications can use these predictions to adapt their behavior: knowing that it is likely to snow next week can be used to plan a ski trip. The second is in the field of algorithm design. Here, the effort has to been to develop a notion of competitive ratio[1] for the goodness of an algorithm in the presence of an unknown future and develop online algorithms that make decisions heedless of the future but are provably good in the *worst-case*, i.e., even in the most pessimistic future scenario. Such online algorithms are popular and successful in real-world systems and have been used to model problems including paging, caching, job scheduling, and more (see the book by Borodin and El-Yaniv [5]).

Recently, there has been some interest in using machine-learned predictions to improve the quality of online algorithms [20, 18]. The main motivation for this line of research is two-fold. The first is to design new online algorithms that can avoid assuming a worst-case scenario and hence have better performance guarantees both in theory and practice. The second is to leverage the vast amount of modeling work in machine learning, which precisely deals with how to make predictions. Furthermore, as machine-learning models are often retrained on new data, these algorithms can naturally adapt to evolving data characteristics. When using the predictions, it is important that the online algorithm is unaware of the performance of the predictor and makes no assumptions on the types of prediction errors. Additionally, we desire two key properties of the algorithm: (i) if the predictor is good, then the online algorithm should perform close to the best offline algorithm (*consistency*) and (ii) if the predictor is bad, then the online algorithm should gracefully degrade, i.e., its performance should be close to that of the online algorithm without predictions (*robustness*).

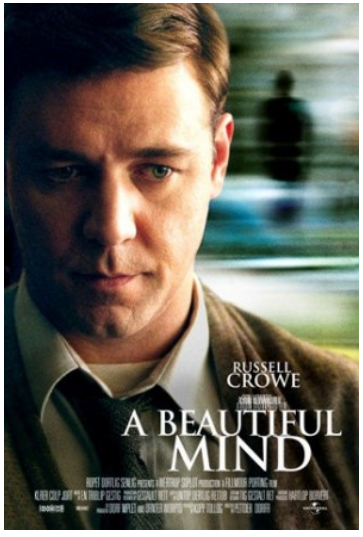
[1]Informally, competitive ratio compares the worst-case performance of an online algorithm to the best offline algorithm that knows the future.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

- Meta-algorithm with tunable parameter $\lambda \in [0, 1]$.
- $\lambda = 0$ blindly follow predictor.
- $\lambda = 1$ completely disregard predictions.

$(1 + \lambda)$-consistent, $(1 + \frac{1}{\lambda})$-robust. optimal tradeoff.

Multiagent online problems



- agents: maximize their utility

  - My payoff depends not only on what I am doing but also on what others are doing.

- Nash equilibrium: action profile so that **everyone best-responds to what others are doing.**

- Nash equilibrium: assumes perfect, trusted others predictor.
- Need notion of equilibrium with predictions.

Multiagent ski-rental

- n agents, need resource while active.
- rent: \$1/day, group license: \$$B$.
- pledging model: each day agents pledge some amount.
- Buy group license: total pledged amount $\geq B$.
- Otherwise pledges null, agents rent for the day.

- Agents optimize their competitive ratio.

Agents two predictors:

- self-predictor for their number of active days.
- others-predictor, for the amounts pledged by other agents.

The four limit cases: best possible algorithms

Results, intuitively:

- self/others-predictors bad: rent on the first day, pledge B on the second day. $(B+1)$-competitive
- both predictors perfect: 1-competitive

- others-predictor bad, self-predictor perfect: doesn't help, still $(B+1)$-competitive.
- others-predictor perfect, self-predictor bad: optimal algorithm, performance depends on what other agents do.

What about intermediate prediction error regimes?

- **Meta-algorithm,** two parameters $\lambda, \mu \in [0, 1]$.

  "Degrees to which we should trust the two predictors"
- Perfect others-predictions: variant of the meta-algorithm for ski-rental with predictions (Purohit, Kumar, Svitkina., NEURIPS 2018), same performance/tradeoffs as this algorithm.

Drawback: mathematical guarantee for the performance, but it depends on what other agents are doing.

What about intermediate prediction error regimes?

Input: (p, T)

Predictor: $\widehat{T} \geq 1$**, an integer.**

let $r_0 = r_0(p, \widehat{T}) = argmin(P_r : 1 \leq r \leq \widehat{T})$

let $r_1 = r_1(p) = argmin(\frac{P_r}{r} : 1 \leq r \leq M_*(p))$

choose $r_2 = r_2(p, \lambda)$ to be the first day $\geq \lceil (1-\lambda)(r_0 - 1) + \lambda r_1 \rceil$ such that $P_{r_2} - OPT_{r_2} = min(P_t - OPT_t :$

$t \geq \lceil (1-\lambda)(r_0 - 1) + \lambda r_1 \rceil)$

if $P_{r_2} > P_{r_1}$ let $r_2 = r_1$

choose $r_3 = r_3(\lambda, p)$ so that $\frac{P_{r_3}}{OPT_{r_3}(p)} \leq 1 + \frac{1}{\lambda}(c_{OPT}(p) - 1)$ minimizing the ratio $\frac{P_{r_3}}{r_3}$

if $(\widehat{T} \geq M_{\widehat{T}}(p))$

buy on day r_2 (possibly not buying at all if $r_2 > T$)

Else

buy on day r_3 (possibly not buying at all if $r_3 > T$)

Simplified version of meta-algorithm meta-algorithm (perfect other predictions only)

Experimental evaluation of the meta-algorithm

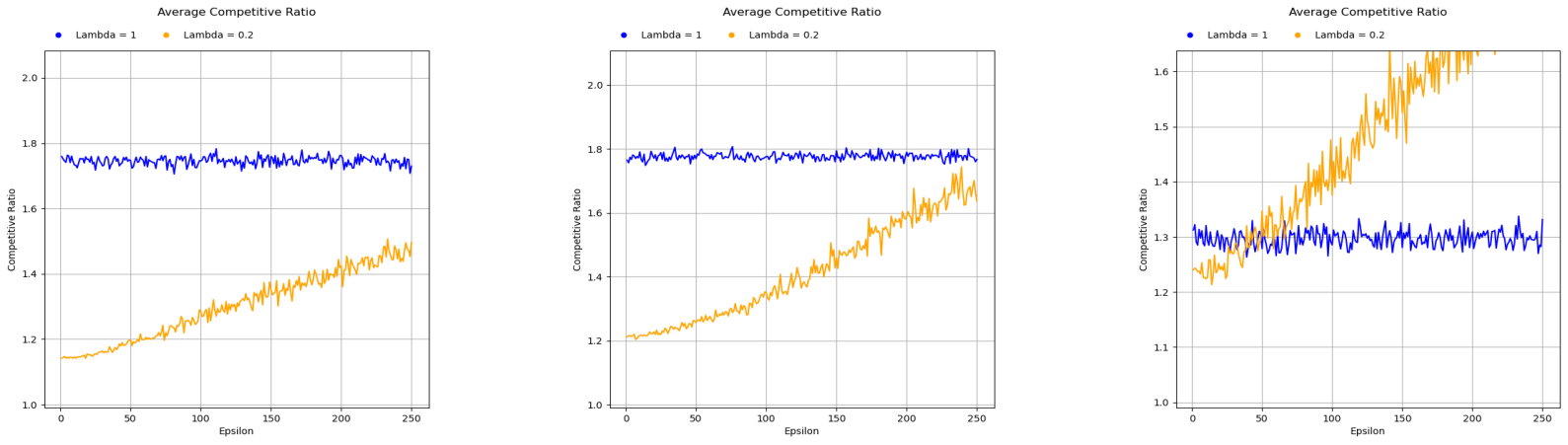


Figure 1: Average competitive ratio for the meta-algorithm, assuming random total bids of other agents.

- Small error of the self-predictor: metaalgorithm improves on the optimal algorithm without predictions.
- Robust (to some extent) to errors in others-prediction.

Equilibria with predictions

- Nash equilibria: given perfect predictions of other agents, if response is not perfect, change it.
- Approximate (ϵ-Nash) equilibria: given perfect predictions, if response is not best within ϵ, change it.

Equilibria with predictions:

- Parameterized, like approximate Nash equilibria.
- Change algorithm when current method does not meet requirements on the 4 parameters.

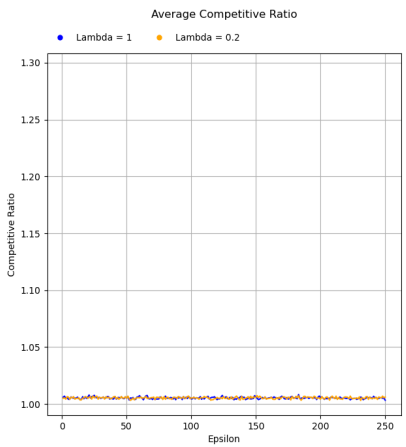
Can characterize Nash equilibria.

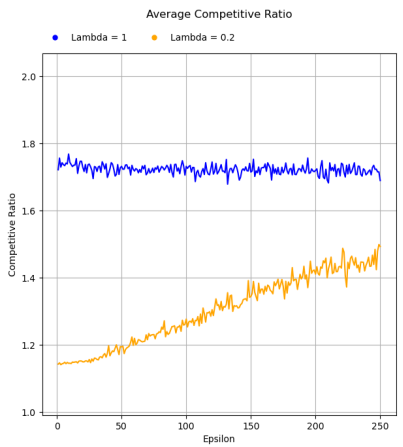
Previous four results: yield equilibria with predictions for multiagent ski-rental.

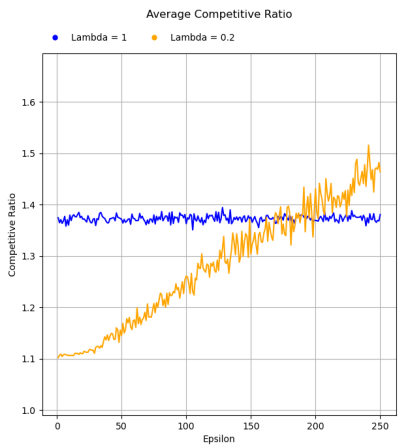
Does using predictions help?

- Nash equilibria = meta-algorithm with $\lambda = 1, \mu = 0$.
- What if one agent self-predicts ($\mu > 0$) without the other agents knowing this?

- THEOREM: in some regimes of μ the agent provably improves its competitive ratio.
- also experiments on using meta-algorithm in equillibria.





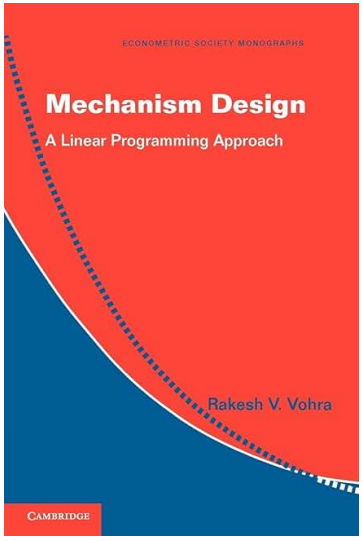


Equilibria with predictions: Conclusions

- Predictions (can) improve algorithms, but sensitive to prediction quality.
- Nash equilibria fundamental concept in multiagent systems, but not when agents use predictions

Gave an extension of Nash equilibria to settings with predictions, and concept/analysis feasible (at least for one problem).

Second Topic: Mechanism Design



- Want: protocol for collective decision for *n* agents.
- agents: private info, may lie about it to get preferred results.
- Want: mechanism to compute optimum so that no agent (coalition of agents) gains by lying.

Nobel Prize in Economics 2007: Leonid Hurwicz, Eric S. Maskin, Roger B. Myerson "for having laid the foundations of mechanism design theory".

Computer Science: Mechanism Design With Predictions

The mechanism designer has a prediction of the optimal decision.

- Chenyang Xu, Pinyan Lu, IJCAI 2022.
- P. Agrawal, E. Balkanski, V. Gkatzelis, T. Ou, & X. Tan. *Learning-Augmented Mechanism Design: Leveraging Predictions for Facility Location*, EC'2022.

EUMAS'2025 paper: study in the setting with predictions obnoxious facility location.
- *n* agents in a metric space. A single "bad" facility is to be chosen. **Agents want to be far from the facility.**

Optimal alg. (without predictions !) on segments

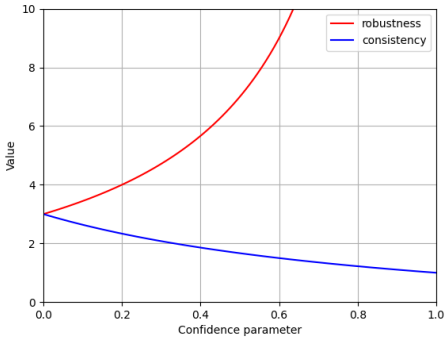
Algorithm 1 Voting mechanism.

Input $C = (x_1, x_2, \ldots, x_n) \in [0, 1]$.
let $n_1 = |\{i : 0 \leq x_i \leq 1/2\}|$.
let $n_2 = |\{i : 1/2 < x_i \leq 1\}|$.
count n_1 votes for 1, n_2 for 0.
if $n_1 \geq n_2$:
return 1;
else:
return 0

- group strategyproof, 3-approximation.
- $\lambda < 3$: no strategyproof λ-approximation.

- Cheng, Yukun, Wei Yu, and Guochuan Zhang. *"Mechanisms for obnoxious facility game on a path."* COCOA' 2011, *"Strategy-proof approximation mechanisms for an obnoxious facility game on networks."* TCS'2013.

Our results



- Tradeoff robustness/consistency: $(\frac{3+\lambda}{1-\lambda}, \frac{3-\lambda}{1+\lambda})$, $\lambda \in [0, 1)$. Same for segments, squares, circles, trees !
- Same when agents have **dual preferences** (Zou & Li, AAMAS'2015, Feigenbaum & Sethuraman, AAAI W. 2015).
- optimal for segments/circles.

Mechanism with predictions: (segment, square)

Algorithm 2 Coordinatewise voting.

Input $C = (x_1, x_2, \ldots, x_n)$, $\lambda \in [0, 1)$.

Output $\mathbf{P_C} = \mathbf{p_1 p_2 \ldots p_k} \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{k}}$.

for i:=1 to k

let $n_{1,k} = |\{i : 0 \leq x_{i,k} \leq 1/2\}|$.

let $n_{2,k} = |\{i : 1/2 < x_{i,k} \leq 1\}|$.

count $n_{1,k}$ votes for 1, $n_{2,k}$ for 0, λn for p_i.

if $votes(p_i) \geq votes(1 - p_i)$:

let $z_i = p_i$.

else let $z_i = 1 - p_i$.

return $Z_C := z_1 z_2 \ldots z_k$.

- Idea: favor the predicted point with $\lambda \cdot n$ extra "votes."
- 2-d case: independent voting on coordinates.

"Bigger picture"

- Multiagent systems have many strategic aspects.
- Predictions interact with these strategic aspects in nontrivial, interesting ways.

Both results: only the beginning!

Mulţumesc. Intrebãri ?