DUELING ALGORITHMS

CLASSICAL OPTIMIZATION THROUGH THE LENS OF COMPETITION

Talk by *Christian Kauth*

based on a paper by *N. Immorlica et. al. 2011*
within Game Theory & Algorithms lecture by *D. Pritchard*

Lausanne, EPFL, May 26th 2011

Outline

- **An introduction**
  - Snake/Tron
  - Ambitions
- **Ranking duel**
  - Example
  - Beatability
- **Bilinear duel framework**
  - Beatability of classical algorithms
  - Zero-sum => Min-max => Linear programming
- **Hiring duel**
  - Optimal single-player strategy
  - Competitive strategy
  - The price of optimality/anarchy
- **Conclusion**

Outline

- **An introduction**
  - Snake/Tron (http://www.youtube.com/watch?v=mwDKMiAfxFE)
  - Ambitions
- **Ranking duel**
  - Example
  - Beatability
- **Bilinear duel framework**
  - Beatability of classical algorithms
  - Zero-sum => Min-max => Linear programming
- **Hiring duel**
  - Optimal single-player strategy
  - Competitive strategy
  - The price of optimality/anarchy
- **Conclusion**

Ambitions

- Revisit classical optimization from perspective of competition
- Objective: ~~single-player cost minimization~~
  outperform the opponent
- Define framework for techniques to minmax-optimal strategies typically for exponentially large zero-sum games
- Case studies: ranking / compression / search / hiring
- Will players use the classic optimization solution in a dueling setting?
- **What strategies do players play at equilibrium?** *[Immorlica]*
- **Are these strategies still good at solving the optimization problem?** *[me here and now]*

Outline

- **An introduction**
  - Snake/Tron
  - Ambitions
- **Ranking duel**
  - Example
  - Beatability
- **Bilinear duel framework**
  - Beatability of classical algorithms
  - Zero-sum => Min-max => Linear programming
- **Hiring duel**
  - Optimal single-player strategy
  - Competitive strategy
  - The price of optimality/anarchy
- **Conclusion**

Ranking Duel

Problem

- Design a search engine that ranks *n* webpages
- Given a probability distribution over queries *p*
- Have the rank of the webpage lower than the opponent's one!

1-player optimal strategy

- Output greedy permutation $(\omega_1, \omega_2, \cdots, \omega_n)$ s.t.

$$p(\omega_1) \geq p(\omega_2) \geq \cdots \geq p(\omega_n)$$

What should you play to beat me for a random distribution over queries $p(i) = \frac{1}{n} + \left(i - \frac{n}{2}\right)\epsilon$?

Ranking Duel

What should you play to beat me for a random distribution over queries $\boldsymbol{p(i) = \frac{1}{n} + \left(i - \frac{n}{2}\right)\epsilon}$?

- I play $(1,2,\cdots,n-1,n)$
- You should reply with $(2,3,\cdots,n,1)$
- To win duel with probability $1 - \frac{1}{n}$

We say the 1-player optimal strategy is $\boldsymbol{\left(1 - \frac{1}{n}\right)}$**-beatable** over a random probability distribution.

Beatability

- If single player (monopolist) was solving the 1-player optimization problem

- How badly could they be beaten if a second player suddenly entered?

- Beatability of algorithm *A* over distribution p

$$E_r[v(A(p,r),p)]$$

- **Beatability** of an algorithm *A*

$$inf_p E_r[v(A(p,r),p)]$$

Outline

- **An introduction**
  - Snake/Tron
  - Ambitions
- **Ranking duel**
  - Example
  - Beatability
- **Bilinear duel framework**
  - Beatability of classical algorithms
  - Zero-sum => Min-max => Linear programming
- **Hiring duel**
  - Optimal single-player strategy
  - Competitive strategy
  - The price of optimality/anarchy
- **Conclusion**

Bounds on Beatability [Ranking]

- *Immorlica et al.* prooved the following bounds

Opt. Prob.	Opt. 1-pl.-strat	Upper bound	Lower bound
Ranking	**greedy**	**1-1/n**	**1-1/n**

Bounds on Beatability [Compression]

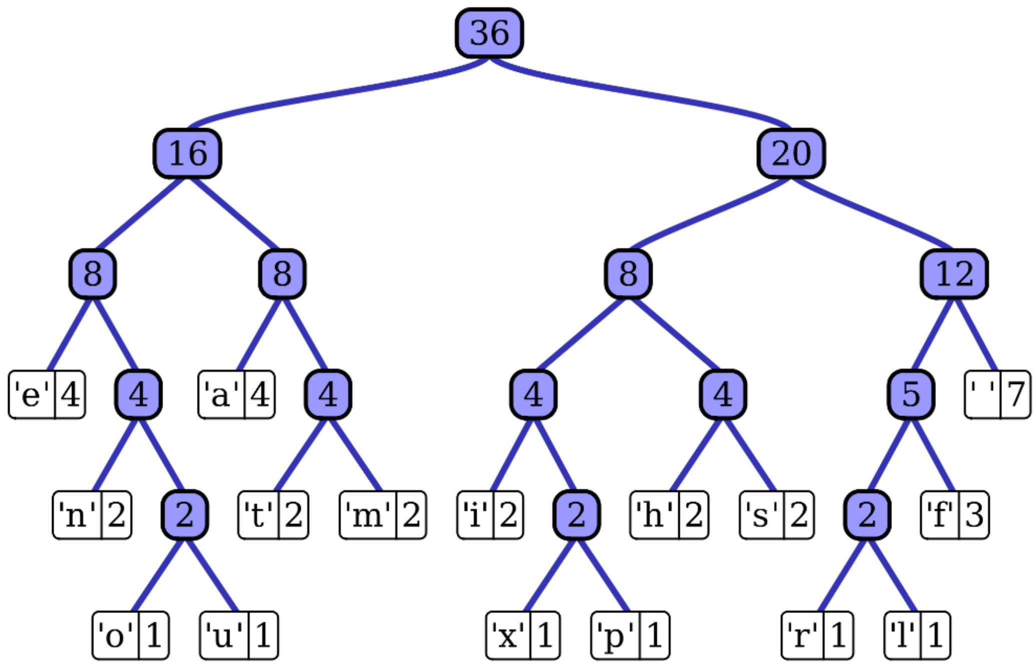
- *Immorlica et al.* prooved the following bounds

Opt. Prob.	Opt. 1-pl.-strat	Upper bound	Lower bound
Ranking	greedy	1-1/n	1-1/n
Compression	**Huffman Coding**	**3/4**	**2/3**

Informal description

- Given a set of symbols and their weights (usually proportional to probabilities).
- Find A prefix-free binary code (a set of codewords) with minimum expected codeword length (equivalently, a tree with minimum weighted path length from the root).

Bounds on Beatability [Compression]

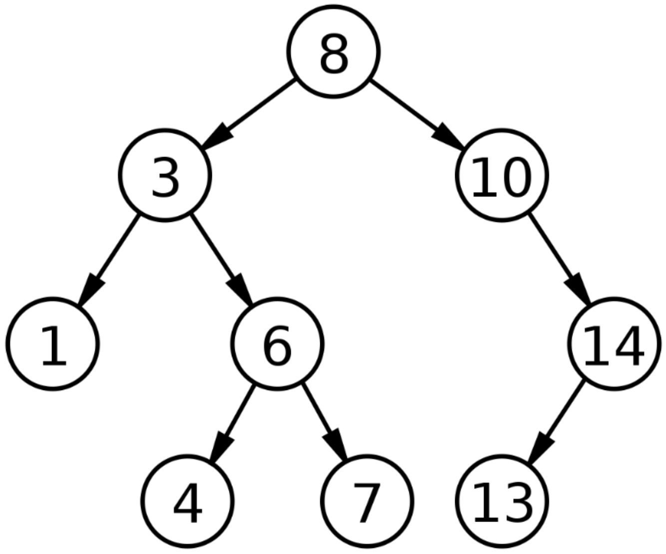


Char	Freq	Code
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

Bounds on Beatability [Search]

- *Immorlica et al.* prooved the following bounds

Opt. Prob.	Opt. 1-pl.-strat	Upper bound	Lower bound
Ranking	greedy	1-1/n	1-1/n
Compression	Huffman Coding	3/4	2/3
Search	**Binary search**	**5/8**	**5/8**



Bounds on Beatability [Hiring]

Opt. Prob.	Opt. 1-pl.-strat	Upper bound	Lower bound
Ranking	greedy	1-1/n	1-1/n
Compression	Huffman Coding	3/4	2/3
Search	Binary search	5/8	5/8
Hiring	**n/e-stopping rule**	**0.82**	**0.51**

- There is a **single secretarial position** to fill.
- There are ***n* applicants** for the position, and the value of *n* is known.
- The applicants can be **ranked from best to worst with no ties**.
- The applicants are interviewed sequentially in a **random order**, with each order being equally likely.
- After each interview, the applicant **is accepted or rejected**.
- The decision to accept or reject an applicant can be **based only on the relative ranks of the applicants interviewed so far**.
- **Rejected applicants cannot be recalled**.
- The objective is to **select the best applicant**. The payoff is 1 for the best applicant and zero otherwise.

Preliminaries

- Problem of optimization under uncertainty (X, Ω, c, p)
  - X feasible set
  - p distribution over state nature ω
  - Ω $\ni \omega$
  - c objective function $c: X \times \Omega \to \boldsymbol{R}$

- Cost of $x \in X$ $c(x) = E_{\omega \sim p}[c(x, \omega)]$
- 1-player optimum $\mathrm{opt} = \min_{x \in X} c(x)$
- Cost of algorithm A $c(A) = E_r\left[c\big(A(p, r)\big)\right]$

Preliminaries

- 2-person constant-sum duel game $D(X, \Omega, c, p)$
  - Players simultaneously choose $\mathrm{x}, \mathrm{x}' \in X$
  - Player 1's payoff

$$v(x, x', p) = Pr_{\omega\sim p}[c(x,\omega) < c(x',\omega)] + \frac{1}{2} Pr_{\omega\sim p}[c(x,\omega) = c(x',\omega)]$$

  - Value of a strategy $v(x,p) = min_{x'\in X} v(x,x',p)$
  - σ is a best response to strategy σ' if it maximizes $v(\sigma, \sigma')$
  - Set of minmax strategies $MM(D(X,\Omega,c,p)\ ) = \left\{\sigma \in \Delta(X) | v(\sigma) = \frac{1}{2}\right\}$
  - Von Neumann (1928) : For constant-sum games, the set of Nash equilibria is the cross-product of the minmax startegies for players 1 and 2.

Bilinear duels

- Feasible set of strategies are points in *n*-dimensional Euclidian space

  $X \subseteq R^n$ and $X' \subseteq R^{n'}$

- Payoff to player 1 is $v(x, x') = x^t M x'$ for some $M \in R^{n \times n'}$

- Let *K* be the convex hull of *X*
  - Every point in *K* is achievable in expectation as a mixed strategy
  - *K* is a polytope defined by the intersection of *m* half-spaces

    $K = \{x \in R^n | w_i \cdot x \geq b_i\}$ for i=1,2,..,*m*

    $K' = \{x' \in R^{n'} | w_i' \cdot x' \geq b_i'\}$ for i=1,2,..,*m'*

LP formulation

- Typical way to reduce to an LP for constant-sum games is

$$max_{v \in R, x \in R^n} v \text{ s.t. } x \in K \text{ and } x^t M x' \geq v \text{ for all } x' \in X'$$

- Exponential number of constraints $m + |X'|$

- The following LP has linear number of constraints and can be solved in polynomial time

$$max_{x \in R^n, \lambda \in R^{m'}} \sum_{i=1}^{m'} \lambda_i \cdot b_i' \text{ s.t. } x \in K \text{ and } x^t M = \sum_{i=1}^{m'} \lambda_i \cdot w_i'$$

- *Lemma (Immorlica) :* For any constant-sum game with strategies $x \in K$ and payoffs $\mathrm{x^t M x'}$, the maximum of the above LP is the value of the game to player 1, and any maximizing *x* is a minmax optimal strategy.

Reduction to bilinear duels

- Reduction of a duel $D(X, \Omega, c, p)$ to bilinear form requires

1. An efficiently computable function $\varphi: X \to K$ that maps each strategy $x \in X$ to a feasible point in $K \subseteq R^n$
2. A matrix M such that $v(x, x') = \varphi(x)^t M \varphi(x)$
3. A set of polynomially many feasible constraints that defines K

Outline

- **An introduction**
  - Snake/Tron
  - Ambitions
- **Ranking duel**
  - Example
  - Beatability
- **Bilinear duel framework**
  - Beatability of classical algorithms
  - Zero-sum => Min-max => Linear programming
- **Hiring duel**
  - Optimal single-player strategy
  - Competitive strategy
  - The price of optimality/anarchy
- **Conclusion**

Hiring – bilinear mapping

- Objective hire better candidate than opponent
- Strategies Mappings from any prefix and permutation of workers' ranks in that prefix to a binary hiring decision

The permutation of ranks in a prefix does not affect the distribution of the rank of the just interviewed worker!

WOLOG strategies are mappings from the round number and current rank to a hiring decision.

- Notation (X, Ω, c, p)
  - X are functions $h: \{1, \cdots, n\}^2 \rightarrow \{0,1\}$ indicating, for any round *i* and projected rank *j* of the current interviewee, the hiring decision $h(i, j)$
  - Ω are all permutations σ of interviewees, p a uniform distribution
  - $c(h, \sigma) = rank\ of\ hired\ candidate$

Hiring – Stopping rule (1-player)

- Payoff is 1 iff hire best candidate, else 0
- No bounds on scores known a priori => stopping rule strategy
- Accumulate knowledge by interviewing applicants and rejecting them. How many?
- *Rule 1* : never accept applicant with score lower than any previous applicant!
- *<u>Def.</u>* A *candidate* satisfies rule 1
- Strategy : @each interview :
  - Is applicant a candidate?
  - If so, compare $Pr[winning\ by\ accepting]$
    $Pr[winning\ by\ rejecting]$
- *<u>Def.</u>* a strategy *STRAT(s)* : reject first *s* applicants, then accept first candidate

Hiring – Stopping rule

- What is *P(WIN by STRAT(s))*?
  - *Hyp:* highest score at position k.
  - if k<=s, *P=0*
  - So $P(\text{WIN by STRAT(s)})$

$$= \sum_{k=s+1}^{N} P(\text{WIN by STRAT(s)} \cap \text{ maximum is at k})$$

$$= \sum_{k=s+1}^{N} P(\text{WIN by STRAT(s)} \mid \text{ maximum is at k}) \cdot P(\text{maximum is at k})$$

  - Random order implies $P(\text{maximum is at k}) = \frac{1}{N}$
  - For the other *P*, we have to ensure that applicant *k* is the first candidate after *s*. This happens only if the maximum of the first k-1 candidates lies within the first *s*, which occurs with probability *s/(k-1)*
  - Finally

$$P(WIN\ \ by\ \ STRAT(s)) = \frac{s}{N} \sum_{k=s+1}^{N} \frac{1}{k-1}$$

Hiring – Stopping rule

- The strategy *s** that maximizes *P(WIN by STRAT(s))* for given *N* can be found by *DP* in linear time or by *ODDS* algorithm in sub-linear time.

- Winning strategies for some *N*

n	1	2	3	4	5	6	7	8	9
s	0	0	1	1	2	2	2	3	3
P	1.000	0.500	0.500	0.458	0.433	0.428	0.414	0.410	0.406

- For large *N*, $s \rightarrow \frac{n}{e}$ and the winning probability converges to $\frac{1}{e} \approx 36.8\%$

Hiring duel – shared information

- Context Same set of applicants for both employers
  Each employer observes when the other hires
- Strategy strategy π is a symmetric equilibrium
  - if opponent already hired: hire anyone who beats his employee
  - else : hire as soon as the current applicant has a $\geq 50\%$ chance of being the best of the remaining ones.
- Lemma π is efficiently computable
- Algorithm let t_i a threshold such that at round *i*, π hires iff the projected rank *j* of the current candidate is at most t_i

Probability that t_ith best applicant among the *i* observed applicants is better than all remaining ones is $\binom{i}{t_i} \Big/ \binom{n}{t_i}$

Hire whenever *j*-th best so far observed on round *i* and $\binom{i}{j} \Big/ \binom{n}{j} \geq \frac{1}{2}$

Hiring duel – shared information

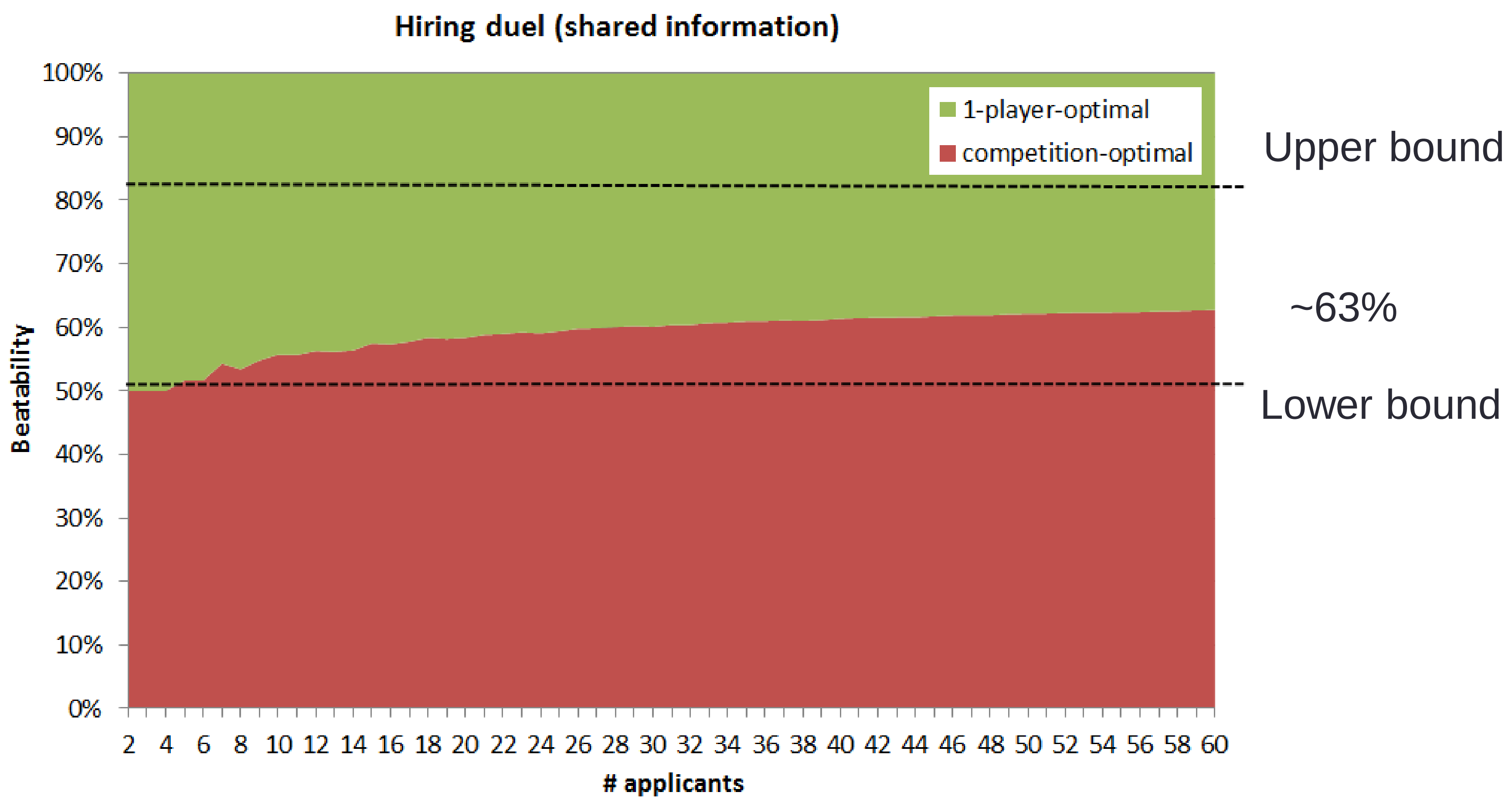
- Lower Bound Beatability of classical algorithm is at least 0.51
- Proof
  - π guarantees a payoff of at least 0.50 in any case
  - for $q > \frac{1}{e}$, consider the event that classical algorithm hires in $\left\{\frac{n}{e}, qn\right\}$.
  - This happens when best among first qn is not among first $\frac{n}{e}$, which occurs with probability $\left(1 - \frac{1}{qe}\right)$
  - Conditioned on that, π wins whenever best candidate is among last $n(1-q)$ applicants *[loose lower bound]* , which occurs with probability $(1-q)$.
  - Overall payoff $1(1-q)\left(1 - \frac{1}{qe}\right) + \frac{1}{2}\frac{1}{qe}$
  - Optimizing for q yields $q^* = \sqrt{2e}$, and payoff equal to *0.51*

Hiring duel – shared information

- Upper Bound Beatability of classical algorithm is at most 0.82
- Proof
  - Classic algorithm has probability $\frac{1}{e}$ of hiring best applicant.
  - The best an opponent could possibly do is hiring always the best applicant.

  - Payoff is then $\frac{1}{2}\frac{1}{e} + 1\left(1 - \frac{1}{e}\right)$, equal to payoff equal to *0.82*

Hiring duel – shared information



Hiring duel – shared information

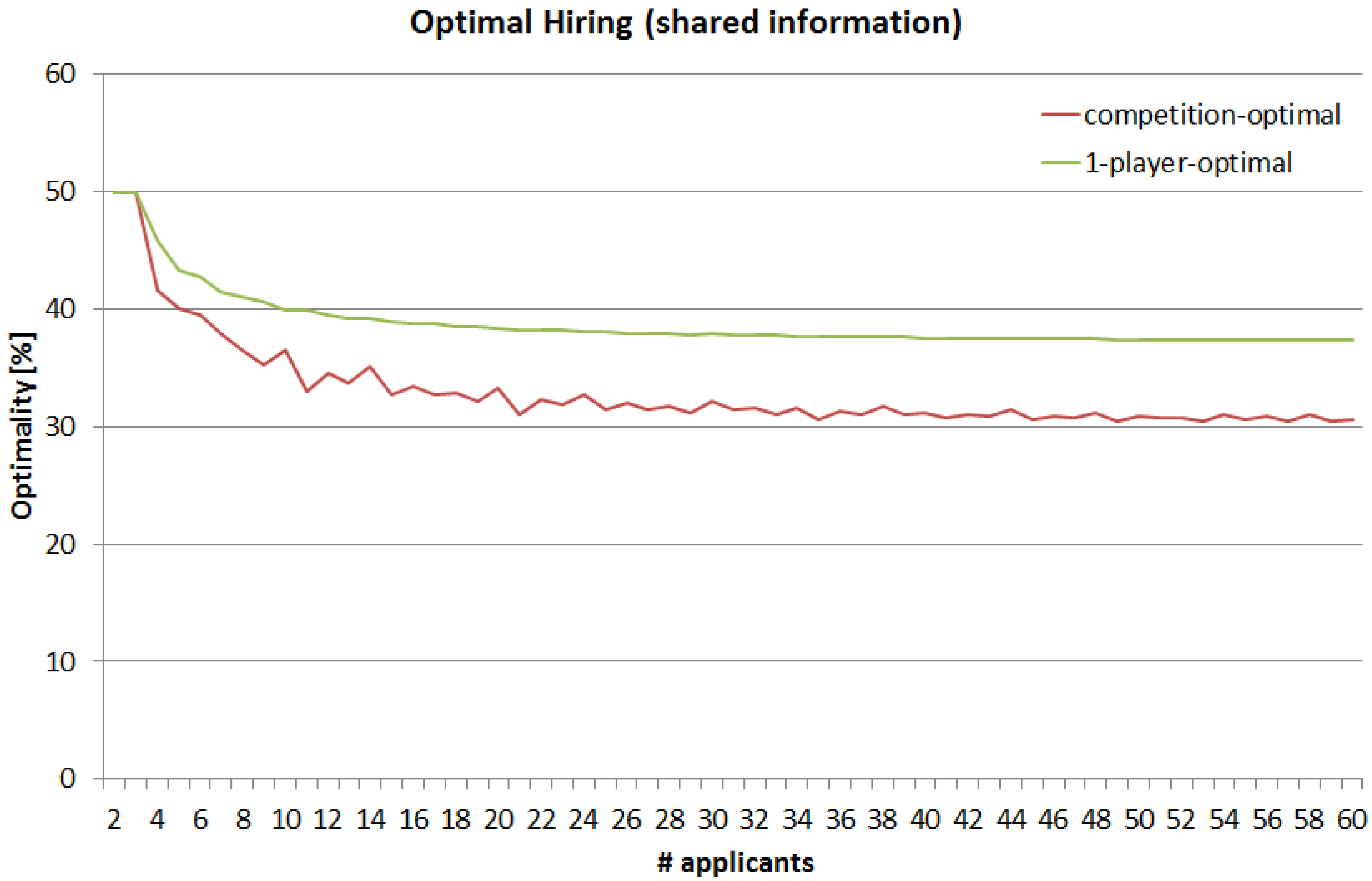
- Fairness

  The competitive algorithm uses information on when the single-player algorithm hires. The reverse is not true. Is this a fair game?

- Portability

  How could the competitive algorithm solve the classical problem?
  => Don't tell it when the opponent hires

Hiring duel – π without information



Hiring duel – no shared information

- Represent strategy π by vectors $\{p_{ij}\}$
  - p_{ij} total probability of hiring *j*-th best seen so far on round *i*

- *Let q_i be the probability of reaching round i*

$$q_1 = 1$$
$$q_{i+1} = q_i - \sum_{j=1}^{i} p_{ij}$$

- $\pi(i, j)$ is the probability of hiring *j*-th best so far on round *i*, conditioned on seeing *j*-th best at round *i.*

- Bayes' rule allows efficient bijective mapping between $\pi(i, j)$ and $\{p_{ij}\}$, our φ.

Hiring duel – no shared information

- Feassible set *K*

  - Probability of hiring *j*-th best in round *i* cannot exceed probability of reaching round *i* and seeing *j*-th best.

$$p_{ij} \leq \frac{q_i}{i}$$

  - Recursive definition of reaching round *i*

$$q_1 = 1$$

$$q_{i+1} = q_i - \sum_{j=1}^{i} p_{ij}$$

- Mapping φ

$$\underbrace{p_{ij}}_{\text{P(hire j in i)}} = \underbrace{q_i}_{\text{P(reach i)}} \underbrace{\pi(i,j)}_{\text{P(hire j in i | in i with j)}} \underbrace{1/i}_{\text{P(see j in i)}}$$

Hiring duel – no shared information

- Payoff matrix $M_{iji'j'}$
  - E_r Event that last candiate has overall rank *r*
  - F_{ij} Projected rank of last candiate in prefix of size *i* is *j*

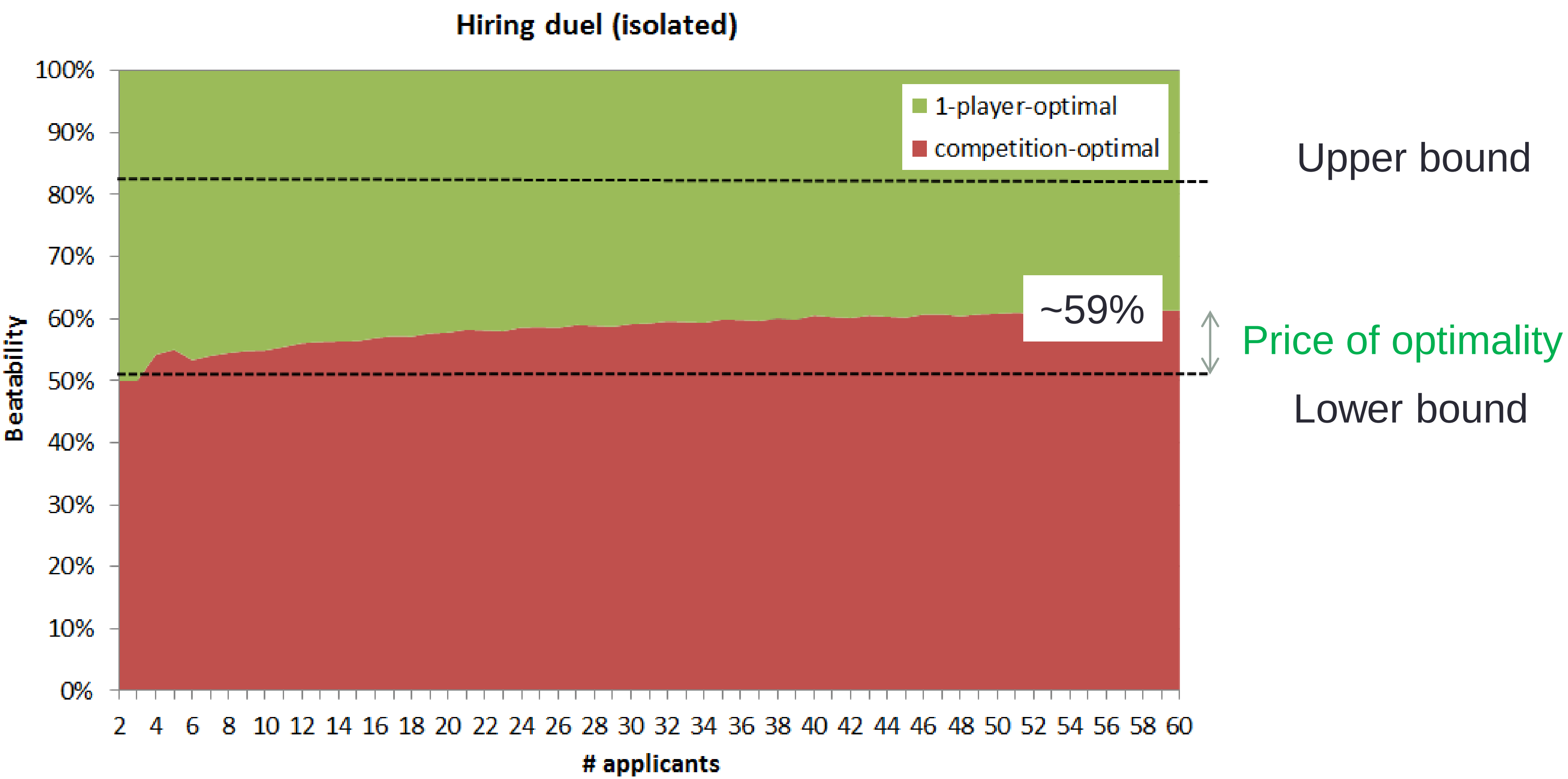
$$M_{iji'j'} = \sum_{r,r':1\le r<r'\le n} Pr[E_r|F_{ij}]\cdot Pr[E_{r'}|F_{i'j'}] + 0.5\sum_{1\le r\le n} Pr[E_r|F_{ij}]\cdot Pr[E_r\ |F_{i'j'}]$$

- Bayes $Pr[E_r|F_{ij}] = Pr[F_{ij}|E_r]\cdot Pr[E_r]/Pr[F_{ij}]$

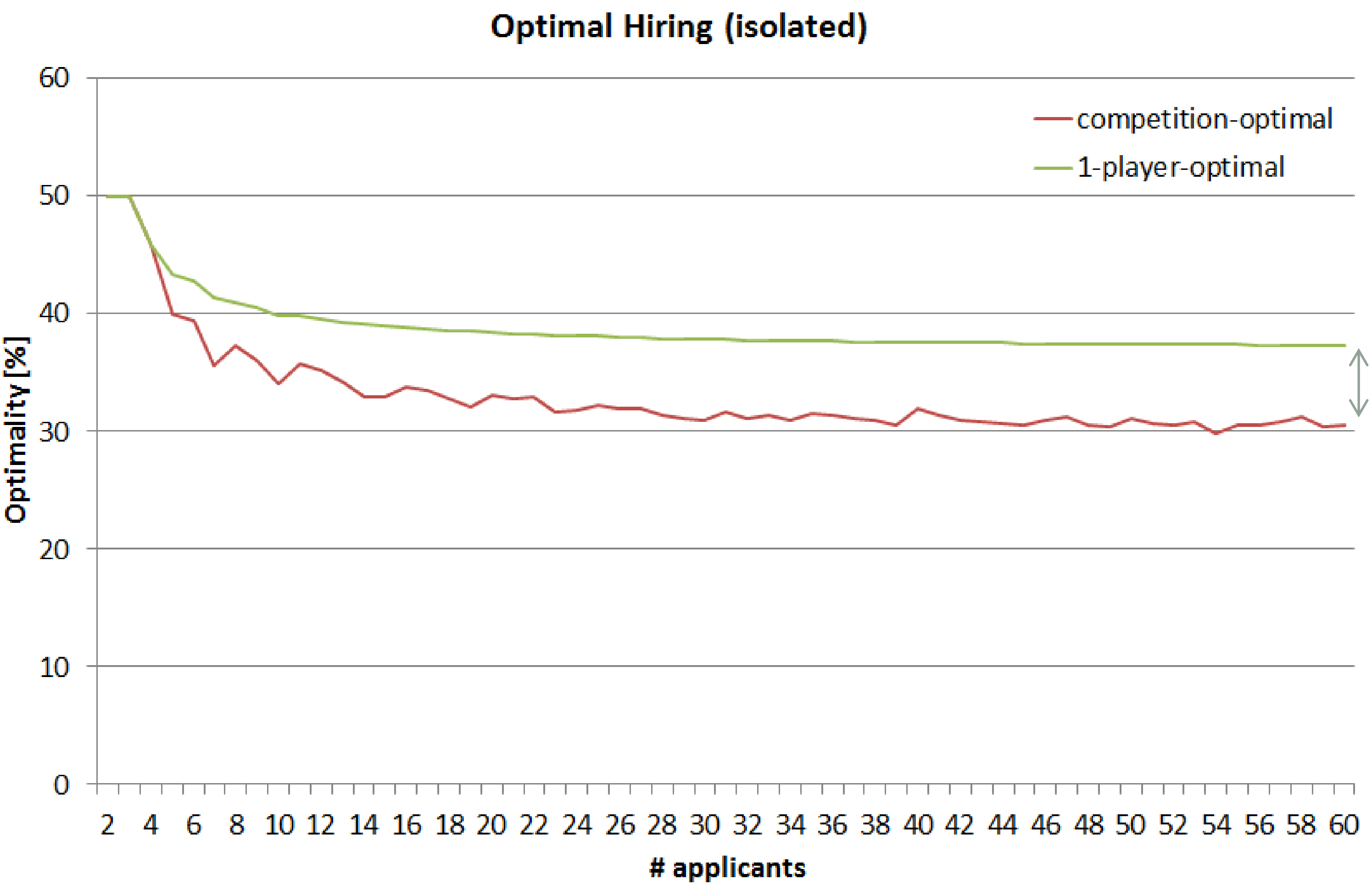
Hiring duel – no shared information

- $Pr[E_r] = \frac{1}{N}$
- $Pr[F_{ij}] = \frac{1}{i}$
- $Pr[F_{ij}|E_r] = \frac{\binom{r-1}{j-1}\binom{n-r}{i-j}}{\binom{n-1}{i-1}}$
- Wrapp it all into an LP, code it and simulate ☺

1-player algorithm in duel



Competitive algorithm in optimization



Price of anarchy

Outline

- **An introduction**
  - Snake/Tron
  - Ambitions
- **Ranking duel**
  - Example
  - Beatability
- **Bilinear duel framework**
  - Beatability of classical algorithms
  - Zero-sum => Min-max => Linear programming
- **Hiring duel**
  - Optimal single-player strategy
  - Competitive strategy
  - The price of optimality/anarchy
- **Conclusion**

Conclusion

- Bilinear framework was presented and applied to Hiring duel
- Bounds on beatability of optimization algorithms

Opt. Prob.	Opt. 1-pl.-strat	Upper bound	Lower bound
Ranking	greedy	1-1/n	1-1/n
Compression	Huffman Coding	3/4	2/3
Search	Binary search	5/8	5/8
Hiring	**n/e-stopping rule**	**0.82**	**0.51**

- There is a price of optimality & anarchy (not in Immorlica)

Hiring(60)

1-player optimal	Competitive algorithm
37%	30%

Thank you!