Communication Issues in Collective Decision-Making

Nicolas Maudet
nicolas.maudet@lip6.fr

Université Pierre et Marie Curie

EPCL-BTC: 18th of November 2013

Recap:

Recall before lunch we mentionned three collective-decision making problems

1. voting
2. two-sided matching
3. resource allocation

Now I would like to investigate (a little bit) the communication requirements of these problems...

Outline of the Talk

Example: Allocating an Item

Consider the following situation:

> *There are two agents (A and B); and one object to allocate. Each agent x has a valuation $v_x \in \{0, 1, 2, 3\}$ for the object. The goal is to give the object to the agent who values it the most.*

Can we design efficient protocols to achieve this goal?

I. Segal. *Communication in Economic Mechanisms*. CES-2006.

Example: Allocating an Item

Consider the following situation:

> *There are two agents (A and B); and one object to allocate. Each agent x has a valuation $v_x \in \{0, 1, 2, 3\}$ for the object. The goal is to give the object to the agent who values it the most.*

Can we design efficient protocols to achieve this goal?

Protocol π_0: "One-sided Revelation"	bits
A gives her valuation	2
B computes the allocation, and send it	1
	total $\Rightarrow$ 3

Example: Allocating an Item

Consider the following situation:

> *There are two agents (A and B); and one object to allocate. Each agent x has a valuation $v_x \in \{0, 1, 2, 3\}$ for the object. The goal is to give the object to the agent who values it the most.*

Can we design efficient protocols to achieve this goal?

Protocol π_1: "English Auction"	bits
$p \leftarrow 0$, $X \leftarrow A$	
while stop:	
$\quad X \leftarrow \overline{X}$	
$\quad$ ask X "stop"' or "raise"	1
$\quad p \leftarrow p + 1$	
allocate to $\overline{X}$	
	total $\Rightarrow$ 1, 2, or 3

Example: Allocating an Item

Consider the following situation:

> *There are two agents (A and B); and one object to allocate. Each agent x has a valuation $v_x \in \{0, 1, 2, 3\}$ for the object. The goal is to give the object to the agent who values it the most.*

Can we design efficient protocols to achieve this goal?

Protocol π_2: "High/Low Bisection"	bits
A says whether her valuation $\{0, 1\}$ (low) or $\{2, 3\}$ (high)	1
B computes the allocation *(if low (if $v_B = 0$ then give to A else give to B))* *(if high (if $v_B = 3$ then give to B else give to A))* and send it	1
	total $\Rightarrow$ 2

Example: Borda voting

There are n agents and p candidates. Each agent x has a ranking $\succ_x$ of the candidates. We give p points to the first candidate, $p-1$ for the second, and so on. The goal is to select the candidate who maximizes the number of points.

- a naive protocol:
  1. each agent reports his own vote to the center ($n \log p!$ bits)
  2. the center sends back the result (name of the winner) ($n \log p$ bits)
- this is actually a universal protocol for any voting rule!
- for specific rules we may design more clever protocols

Conitzer & Sandholm. *Communication Complexity of Common Voting Rules*. EC-05.

Example: Single Transferable vote (STV)

if there exists a candidate c ranked first by a majority of votes
then c wins
else Repeat

let d be the candidate ranked first by the fewest voters;
eliminate d from all ballots
$\{$votes for d transferred to the next best remaining candidate$\}$;

Until there exists a candidate c ranked first by a majority of votes

3	4	3	2
a	b	c	d
d	d	d	c
b	a	a	b
c	c	b	a

Example: Single Transferable vote (STV)

if there exists a candidate c ranked first by a majority of votes
then c wins
else Repeat
let d be the candidate ranked first by the fewest voters;
eliminate d from all ballots
{votes for d transferred to the next best remaining candidate};
Until there exists a candidate c ranked first by a majority of votes

3	4	3	2
a	b	c	d
d	d	d	c
b	a	a	b
c	c	b	a

3	4	3	2
a	b	c	c
b	a	a	b
c	c	b	a

Example: Single Transferable vote (STV)

if there exists a candidate c ranked first by a majority of votes
then c wins
else Repeat
  let d be the candidate ranked first by the fewest voters;
  eliminate d from all ballots
  $\{$votes for d transferred to the next best remaining candidate$\}$;
Until there exists a candidate c ranked first by a majority of votes

3	4	3	2	3	4	3	2	7	5
a	b	c	d	a	b	c	c	b	c
d	d	d	c	b	a	a	b	c	b
b	a	a	b	c	c	b	a		
c	c	b	a						

Winner: b

Example: Single Transferable Vote (STV)

A slightly more involved protocol...

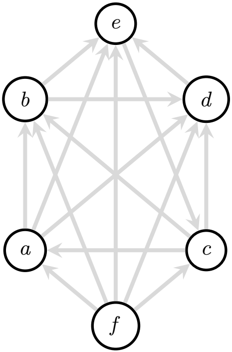
step 1 voters send their most preferred candidate to the central authority (C)
$\Rightarrow n \log p$ bits

step 2 let x be the candidate to be eliminated. All voters who had x ranked first receive a message from C asking them to send the name of their next preferred candidate. There were at most $\frac{n}{p}$ such voters
$\Rightarrow \frac{n}{p} \log p$ bits

step 3 similarly with the new candidate y to be eliminated. At most $\frac{n}{p-1}$ voters voted for y
$\Rightarrow \frac{n}{p-1} \log p$ bits
etc.

total $\leq n \log p (1 + \frac{1}{p} + \frac{1}{p-1} + \ldots + \frac{1}{2}) = O(n.(\log p)^2)$.

Condorcet winner: query complexity

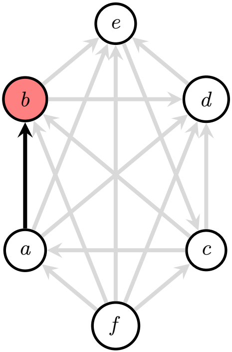


Example: Condorcet winner (query complexity)

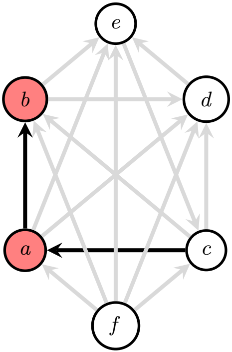
Now consider the slightly different query complexity model.

- A (di)graph is unknown to start with, and want to check whether some property holds in the graph by probing the fewest possible edges (= queries)
- How many (pairwise comparison) queries are necessary to check whether there is a Condorcet winner in a tournament?
- Note that to query one edge we may need to ask n agents
- Of course $p(p-1)/2$ are sufficient. Can we do better?

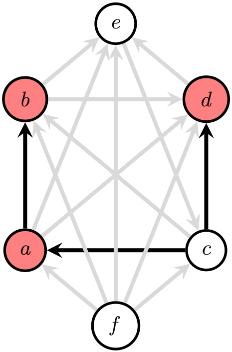
Example: Condorcet winner (query complexity)



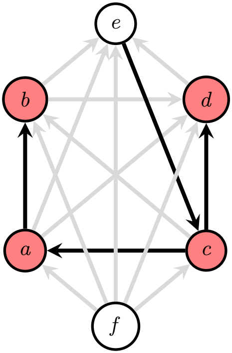
Example: Condorcet winner (query complexity)



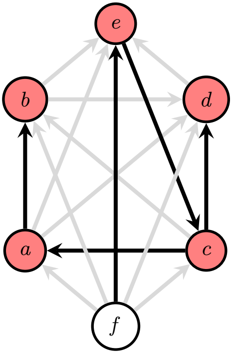
Example: Condorcet winner (query complexity)



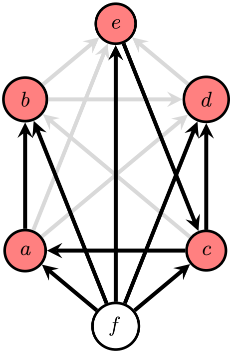
Example: Condorcet winner (query complexity)



Example: Condorcet winner (query complexity)



Example: Condorcet winner (query complexity)



Example: Condorcet winner (query complexity)

- start with an arbitrary query between two candidates
- mark the looser as discarded
- repeat $p - 2$ times:
  - take the winner of the previous query, query against a non-discarded candidate, mark the loser as discarded
  - note: each pairwise comparison discards exactly 1 new candidate
- after $p - 1$ questions we either know that there is no Condorcet winner, or there is a unique potential Condorcet winner
- then we need to check that this candidate beats all the remaining $p - 2$ ones
- this protocol requires $2p - 3$ queries

Example: Condorcet winner (query complexity)

- start with an arbitrary query between two candidates
- mark the looser as discarded
- repeat $p-2$ times:
  - take the winner of the previous query, query against a non-discarded candidate, mark the loser as discarded
  - note: each pairwise comparison discards exactly 1 new candidate
- after $p-1$ questions we either know that there is no Condorcet winner, or there is a unique potential Condorcet winner
- then we need to check that this candidate beats all the remaining $p-2$ ones
- this protocol requires $2p-3$ queries

Can we do better than this?

Example: Condorcet winner (query complexity)

1. build an almost complete *binary tree*, where leaves are labelled as candidates
2. repeat until the root is labelled
   - query about two leaves
   - label the father with the winner
   - cut the children
3. query about the candidate labelling the root (r) against all candidates not

How many queries?

Example: Condorcet winner (query complexity)

1. build an almost complete *binary tree*, where leaves are labelled as candidates
2. repeat until the root is labelled
   - query about two leaves
   - label the father with the winner
   - cut the children
3. query about the candidate labelling the root (r) against all candidates not

How many queries?
Step 2 takes $p-1$ queries.
Furthermore, r must have beaten at least $\lfloor \log_2(p) \rfloor$ during step 2.
Therefore there are $p-1-\lfloor \log_2(p) \rfloor$ during step 3.
The protocol requires at most $2p - \log_2(p) - 2$ queries.

A. Procaccia. *A note on the query complexity of the Condorcet winner problem.* Information Processing Letters 108(6), 2008.

Lessons so far...

- N agents, $\mathcal{O}$ alternatives (may be candidates, allocations, ...)
- important to know the amount of information that needs to be exchanged to compute the outcome, but it may be difficult to design the most efficient protocols
- no concern regarding the computational task faced by agents
- can we say more than analyzing concrete protocols?

Outline of the Talk

1. Motivation
2. Basics of communication complexity
3. Voting
4. Two-sided matching
5. Distributed resource allocation

Outline of the Talk

Communication Complexity Setting

Basic communication complexity setting

A set of n agents have to compute a function $f(x^1, \ldots, x^n)$ given that the input is distributed among the agents (x^1 privately known from agent 1, etc.)

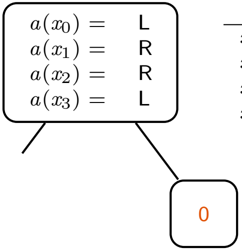
- protocols: specify a communication action by the agents, given its (private) input and the bits exchanged so far
- useful tree representation where each node is labelled by either agent a or agent b (case of two agents), with a function specifying whether to walk left (L) or right (R) depending on its private input.

Kushilevitz & Nisan. *Communication complexity*. Cambridge Univ. Press, 1997.

Protocols illustrated

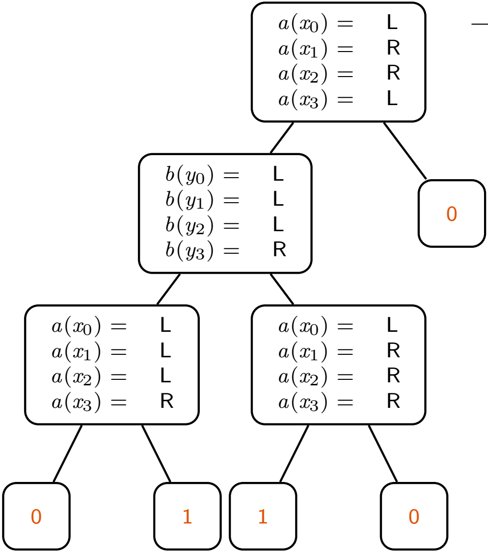
	y_0	y_1	y_2	y_3
x_0	0	0	0	1
x_1	0	0	0	0
x_2	0	0	0	0
x_3	1	1	1	0

Protocols illustrated



	y_0	y_1	y_2	y_3
x_0	0	0	0	1
x_1	0	0	0	0
x_2	0	0	0	0
x_3	1	1	1	0

Protocols illustrated



	y_0	y_1	y_2	y_3
x_0	0	0	0	1
x_1	0	0	0	0
x_2	0	0	0	0
x_3	1	1	1	0

Protocols illustrated

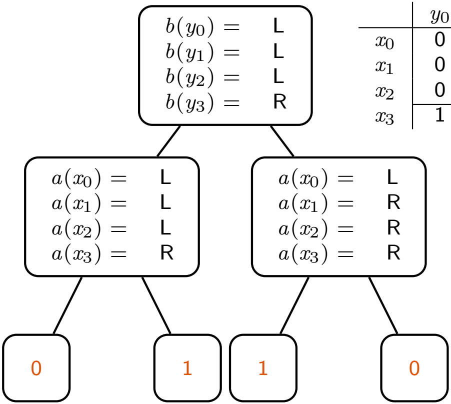
	y_0	y_1	y_2	y_3
x_0	0	0	0	1
x_1	0	0	0	0
x_2	0	0	0	0
x_3	1	1	1	0

Protocols illustrated

$$
\begin{aligned}
b(y_0) &= \text{L}\\
b(y_1) &= \text{L}\\
b(y_2) &= \text{L}\\
b(y_3) &= \text{R}
\end{aligned}
$$

	y_0	y_1	y_2	y_3
x_0	0	0	0	1
x_1	0	0	0	0
x_2	0	0	0	0
x_3	1	1	1	0

Protocols illustrated



	y_0	y_1	y_2	y_3
x_0	0	0	0	1
x_1	0	0	0	0
x_2	0	0	0	0
x_3	1	1	1	0

Cost of protocols

- the cost of a protocol is the number of bits exchanged (in the worst case), *i.e.* the height of the tree.
  $\Rightarrow$ on our example, the "best" cost is the second one (cost 2 vs. 3 for the first one)
- other models (*e.g.* average) are of course possible
- the communication complexity of a function f is the minimum cost of $\mathcal{P}$ among all protocols $\mathcal{P}$ that compute f.

Protocols

Observe that the protocols, as described, in fact partition the matrix of inputs into monochromatic (same output) rectangles

	y_0	y_1	y_2	y_3
x_0	0	0	0	1
x_1	0	0	0	0
x_2	0	0	0	0
x_3	1	1	1	0

$\Rightarrow$ 5 monochromatic rectangles

- the number of leaves is the number of rectangles in the partition
- the cost of any protocol for a function is at least $\log$ of the minimum number of rectangles

Protocols

Observe that the protocols, as described, in fact partition the matrix of inputs into monochromatic (same output) rectangles

	y_0	y_1	y_2	y_3
x_0	0	0	0	1
x_1	0	0	0	0
x_2	0	0	0	0
x_3	1	1	1	0

$\Rightarrow$ 5 monochromatic rectangles

- the number of leaves is the number of rectangles in the partition
- the cost of any protocol for a function is at least $\log$ of the minimum number of rectangles

Back to our first example...

	0	1	2	3
0	B	B	B	B
1	A	B	B	B
2	A	A	B	B
3	A	A	A	B

	0	1	2	3
0	A	B	B	B
1	A	B	B	B
2	A	A	A	B
3	A	A	A	B

	0	1	2	3
0	A	B	B	B
1	A	B	B	B
2	A	A	A	B
3	A	A	A	B

Non-deterministic (aka verification) protocols

- an omniscient agent knows the inputs (thus the outcome) and needs to convince all agents that this is the correct outcome
- it sends a message (a "proof") that each agent can accept or refuse
- a proof of $f(x, y) = z$ is the id of a monochromatic rectangle containing (x, y) that agents can check
- a proof system is a cover (instead of a partition) of the matrix by monochromatic rectangles
- lower bounds on non-deterministic protocols hold for the deterministic ones

Lower bound techniques

Maybe a super-wise friend can come up with a nice protocol...
How can we find lower bounds on the communication complexity?

- one of them is the fooling set technique (from TCS)
- another one is the budget protocol technique (from economics)

Note: These techniques actually yields lower bounds on non-deterministic protocols

The fooling set technique

- if we find a large number of inputs such that no two of them can be in the same rectangle, the number of rectangles must be large as well.
- when two input pairs (x_1, y_1) and (x_2, y_2) are in the same monochromatic rectangle, so do (x_1, y_2) and (x_2, y_1)

0	?
?	0

- fooling set: a collection of inputs such that none of them can be in the same monochromatic rectangle with another one
- Key result (Yao,1979): CC is at least $\log(\#\text{fooling set})$

	0	1	2	3
0	B	B	B	B
1	A	B	B	B
2	A	A	B	B
3	A	A	A	B

The budget protocol technique

- extends the idea that an optimal allocation can be justified by price equilibrium
- a budget equilibrium message is vector $\langle x, B_1, \ldots, B_n \rangle$ where
  - x is the proposed alternative
  - $B_i \subset X$ is a budget set proposed to i
- not all social choice problems can be analyzed with this technique (property of "intersection-monotonicity")
- has been used to show lower bounds for combinatorial auctions problems, matching problems, and many others...

I. Segal. *The communication requirements of social and supporting budget sets.* JET-2006.

N. Nisan & I. Segal. *The communication requirements of efficient allocations and supporting prices.* JET-2005.

Outline of the Talk

Communication complexity of voting rules

In our context, we have:

- f is the voting rule
- x_i is the ballot of voter i
- we are interested in a distinguished candidate a, so f returns 1 if a wins, ad 0 otherwise

A fooling set is then a set of profiles P_i such that :

1. there exists a candidate c such that $r(P^i) = c$
2. for any pair (i,j) $(i \neq j)$, there exists $(m_1, m_2, \ldots, m_n) \in \{i,j\}^n$ such that $r(v_1^{m_1}, v_2^{m_2}, \ldots, v_n^{m_n}) \neq c$

$\hookrightarrow$ we can "mix" the profiles by picking votes either in P^i or P^j and fool the function

V. Conitzer & T. Sandholm. *The communication complexity of common voting rules.* EC-05.

Example: Lower bound for the Borda rule
[Conitzer & Sandholm, EC05]

We note $p' = p - 2$ and $n' = (n-2)/4$, π an arbitrary permutation of candidates $\mathcal{X} \setminus \{a, b\}$ and $\overline{\pi}$ the "mirror" permutation.

1	2	3	4	$\cdots$	$n-1$	n
a	a	$\overline{\pi}$	$\overline{\pi}$	$\cdots$	a	$\overline{\pi}$
b	b	$\vdots$	$\vdots$		b	$\vdots$
π	π	$\vdots$	$\vdots$		π	$\vdots$
$\vdots$	$\vdots$	$\overline{\pi}$	$\overline{\pi}$		$\vdots$	$\overline{\pi}$
$\vdots$	$\vdots$	b	b		$\vdots$	a
π	π	a	a	$\cdots$	π	b

$\Rightarrow (p'!)^{n'}$ such profiles

Example: Lower bound for the Borda rule

[Conitzer & Sandholm, EC05]

We note $p' = p - 2$ and $n' = (n-2)/4$, π an arbitrary permutation of candidates $\mathcal{X} \setminus \{a, b\}$ and $\overline{\pi}$ the "mirror" permutation.

1	2	3	4	$\cdots$	$n-1$	n
a	a	$\overline{\pi}$	$\overline{\pi}$	$\cdots$	a	$\overline{\pi}$
b	b	$\vdots$	$\vdots$		b	$\vdots$
π	π	$\vdots$	$\vdots$		π	$\vdots$
$\vdots$	$\vdots$	$\overline{\pi}$	$\overline{\pi}$		$\vdots$	$\overline{\pi}$
$\vdots$	$\vdots$	b	b		$\vdots$	a
π	π	a	a	$\cdots$	π	b

$\Rightarrow (p'!)^{n'}$ such profiles

1. Does a wins in any such profile?

   Observe that a is 1 point ahead of any other candidate (thanks to voter n)

Example: Lower bound for the Borda rule
[Conitzer & Sandholm, EC05]

We note $p' = p - 2$ and $n' = (n-2)/4$, π an arbitrary permutation of candidates $\mathcal{X} \setminus \{a, b\}$ and $\overline{\pi}$ the "mirror" permutation.

1	2	3	4	$\cdots$	$n-1$	n	
a	a	$\overline{\pi}$	$\overline{\pi}$	$\cdots$	a	$\overline{\pi}$	
b	b	$\vdots$	$\vdots$		b	$\vdots$	
π	π	$\vdots$	$\vdots$		π	$\vdots$	$\Rightarrow (p'!)^{n'}$ such profiles
$\vdots$	$\vdots$	$\overline{\pi}$	$\overline{\pi}$		$\vdots$	$\overline{\pi}$	
$\vdots$	$\vdots$	b	b		$\vdots$	a	
π	π	a	a	$\cdots$	π	b	

1. Does a wins in any such profile?

   Observe that a is 1 point ahead of any other candidate (thanks to voter n)
2. Is it fooling?

   Take two profiles P_1 and P_2, for at least one voter $i \in \{1, \ldots, n'\}$ the vote differs. Thus at least one candidate $c \notin \{a, b\}$ must be ranked higher in P_1 than P_2. Mix profiles by picking votes 4i-3 and 4i -2 from P_1 and the rest from P_2. Now c get 2 aditional points and wins.

Unknown number of missing voters: how to store the current profile?

1	2	3	...
a	b	b	
b	c	a	
c	a	c	

↓

4	5
c	a
a	b
b	c

↓

The winner is: x

Storage: compilation complexity

Compiling Profiles

Unknown number of missing voters: how to store the current profile?

1	2	3	…
a	b	b	
b	c	a	
c	a	c	

compilation function → 10011000101

4	5
c	a
a	b
b	c

4	5
c	a
a	b
b	c

The winner is: x ↔ *same winner* ↔ The winner is: x

Storage: compilation complexity
Compilation Functions

- From the perspective of the device/entity aggregating the votes.
- We are after the best compilation functions for each voting rule.
- To start with, for any anonymous voting rule, compiling the profile into the corresponding voting situation is possible:

Profile:

1	2	3	4	5
a	b	b	c	a
b	c	a	a	b
c	a	c	b	c

Voting situation:

2	1	1	1
a	b	b	c
b	c	a	a
c	a	c	b

Hence the compilation requires at most $min(n \log p!, p! \log n)$.

- From the perspective of the device/entity aggregating the votes.
- We are after the best compilation functions for each voting rule.
- To start with, for any anonymous voting rule, compiling the profile into the corresponding voting situation is possible:

Profile:

1	2	3	4	5
a	b	b	c	a
b	c	a	a	b
c	a	c	b	c

Voting situation:

2	1	1	1
a	b	b	c
b	c	a	a
c	a	c	b

Hence the compilation requires at most $min(n \log p!, p! \log n)$.

- Very efficient when $n \gg p$. Eg. $n = 4703$ and $p = 4$ we get $min(4703 \log 24, 24 \log 4703)$ so 312 bits *vs.* 23515 bits.

Storage: compilation complexity

Compilation Functions for Specific Rules

- Intuitively, for specific voting rules one can get much better compilations, eg. for plurality just compile the score yields $p \log n$.
- But, again, these are upper bounds...
- In fact, the problem can be seen as a "one-round" communication complexity problem (the center must send the relevant information in one single message)

Storage: compilation complexity

Compiling Profiles: Methodology

- two profiles are equivalent for a voting rule if they return the same winner for any possible completion.
- the key is to characterize equivalence classes for each rules, and enumerate them (not always easy...).
- the compilation complexity is given by taking the $\log$ of this number (since we need to identify them).

Voting rule	*Characterization of equiv.*	*Compilation complexity*
Any voting rule	same profiles	$O(np \log p)$
Anonymous	same voting situations	$O(p! \log n)$
STV	for all $Z \subseteq C$ and $x \notin Z$, $score_{Pl}(x, P^{-Z}) = score_{Pl}(x, Q^{-Z})$	$\Omega(2^p \log n)$ $O(p2^p \log n)$
Plurality/runoff	$\mathcal{M}_P = \mathcal{M}_Q$ and $score_{Pl}(x, P) = score_{Pl}(x, Q)$	$\Theta\left(p^2 \log n\right)$
Cond. WMG	$\mathcal{M}_P = \mathcal{M}_Q$	$O\left(p^2 \log n\right)$
Borda	$score_B(x, P) = score_B(x, Q)$	$\Theta(p \log np)$
Plurality	$score_{Pl}(x, P) = score_{Pl}(x, Q)$	$\Theta\left(p \log(1 + \frac{n}{p}) + n \log(1 + \frac{p}{n})\right)$

Chevaleyre *et al.* *Compiling the votes of a subelectorate.* IJCAI, 2009.

Xia & Conitzer. *Compilation Complexity of Common Voting Rules.* AAAI, 2010.

Outline of the Talk

Communication complexity of stable marriage

We make use of canonical women's preferences (same for all women)

$$
\begin{array}{llll}
m_1: & w_1 \succ w_2 \succ w_3 & w_1: & m_1 \succ m_2 \succ m_3 \\
m_2: & w_3 \succ w_2 \succ w_1 & w_2: & m_1 \succ m_2 \succ m_3 \\
m_3: & w_1 \succ w_3 \succ w_2 & w_3: & m_1 \succ m_2 \succ m_3
\end{array}
$$

- Observe that instances with canonical women's preferences admits a single matching
- in this unique matching, man w_i is matched to his preferred woman among the ones not matched with men $1, \ldots, i-1$.

Communication complexity of stable marriage

Now suppose we query the men's preferences lists.
A query is of the form: "man i, who is your kth preferred woman?"

The adversary has the following strategy:

- for any man's preference i, answers the $k < i$ queries by answering that women w_k is in that cell;
- after $i - 1$ queries, adversary marks ($\uparrow$) the leftmost unqueried cell, and reserve it for w_i.
- when $k \geq i$: if cell $\uparrow$ is queried, answer w_i, otherwise answer with a non yet affected w_j for man m_i.

$$\begin{array}{llll} m_1: & . \succ . \succ . \succ . & w_1: & m_1 \succ m_2 \succ m_3 \succ w_3 \\ m_2: & . \succ . \succ . \succ . & w_2: & m_1 \succ m_2 \succ m_3 \succ w_3 \\ m_3: & . \succ . \succ . \succ . & w_3: & m_1 \succ m_2 \succ m_3 \succ w_3 \\ m_4: & . \succ . \succ . \succ . & w_4: & m_1 \succ m_2 \succ m_3 \succ w_3 \end{array}$$

Gusfield & Irving. *The stable marriage problem*. MIT Press.

Communication complexity of stable marriage

m_1 :	$w_1 \succ w_2 \succ w_4 \succ w_3$	w_1 :	$m_1 \succ m_2 \succ m_3 \succ w_3$
m_2 :	$w_2 \succ w_4 \succ w_3 \succ w_1$	w_2 :	$m_1 \succ m_2 \succ m_3 \succ w_3$
m_3 :	$w_1 \succ w_3 \succ w_4 \succ w_2$	w_3 :	$m_1 \succ m_2 \succ m_3 \succ w_3$
m_4 :	$w_1 \succ w_2 \succ w_3 \succ w_4$	w_4 :	$m_1 \succ m_2 \succ m_3 \succ w_3$

- Observe that doing so, the adversary makes sure that for man m_i, all women on the left (=preferred) have a lower index
- ... but then they must be matched with other men, ie. women w_i is the first unmatched for m_i
- the matching $(m_1, w_1), \ldots, (m_n, w_n)$ must be the unique matching

Now suppose for a man m_i we query less than i queries. By the strategy, there must exist an unqueried cell on or before the diagonal. By putting $z > i$ in that cell, the adversary prevents w_i from being matched to m_i (w_z will be instead). The matching would be different.

Hence $1 + 2 + \cdots + n = \frac{n\times(n+1)}{2}$ queries are required.

Uncoordinated markets: length of paths

We mentioned uncoordinated two-sided markets.
How about the length of the process?

- we mentioned the existence of a path to stability (under better responses)—even better: there is a path of polynomial length
- ... but Knuth also showed that convergence is not guaranteed w/o coordination (cycles can occur)
- if blocking pairs are selected uniformly at random, the probability of convergence is 1
- ... however, there are instances where the expected length of the process is exponential

probabilistic model checking used to study such stochastic systems

Knuth. *Mariages stables et leurs relations avec d'autres problèmes combinatoires*. .

Ackermann et al.. *Uncoordinated Two-sided Matching Markets*. EC-07.

Biro & Norman. *Analysis of Stochastic Matching Markets*. IJGT-2013.

Outline of the Talk

Length of paths in distributed resource allocation

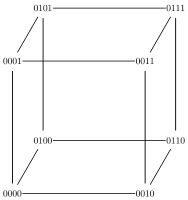
In distributed resource allocation it is often relevant to study the communication complexity in terms of the number of deals required to reach an outcome.

- there are $|\mathcal{N}|^{|\mathcal{O}|}$ allocations
- as it is possible to construct scenarios going through all the allocations, it is an upper bound
- without any restriction on the deal complexity, a path of length 1 is always possible
- with one-resource-at-a-time deals in additive domains, the path length is between $|\mathcal{O}|$ and $|\mathcal{O}| \times (|\mathcal{N}| - 1)$

Endriss & Maudet. *Communication Complexity of Multilateral Trading*. JAAMAS05.

Length of paths in distributed resource allocation

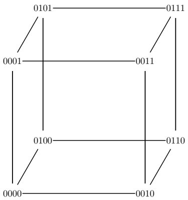
Suppose we only consider deals consisting of moving one-resource at-a-time without domain restriction.
Can we find lower bounds on the path length?



Related to the problem of finding a sequence of moves in an hypercube such, for any state s_i, any other state $s_{\geq i+2}$ in this sequence has a Hamming distance ≥ 2 with s_i (so that no "shortcuts" are possible)

Length of paths in distributed resource allocation

Suppose we only consider deals consisting of moving one-resource at-a-time without domain restriction.
Can we find lower bounds on the path length?



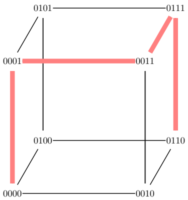
Related to the problem of finding a sequence of moves in an hypercube such, for any state s_i, any other state $s_{\geq i+2}$ in this sequence has a Hamming distance ≥ 2 with s_i (so that no "shortcuts" are possible)

- This actually corresponds to the snake in the box problem, very well studied. Their maximal length is $\mathcal{O}(2^n)$ (precisely, $\frac{77}{256}2^m - 2$)

Dunne. *Extremal behaviour in multiagent contract negotiation*. JAIR-05.

Length of paths in distributed resource allocation

Suppose we only consider deals consisting of moving one-resource at-a-time without domain restriction.
Can we find lower bounds on the path length?



Related to the problem of finding a sequence of moves in an hypercube such, for any state s_i, any other state $s_{\geq i+2}$ in this sequence has a Hamming distance ≥ 2 with s_i (so that no "shortcuts" are possible)

- This actually corresponds to the snake in the box problem, very well studied. Their maximal length is $\mathcal{O}(2^n)$ (precisely, $\frac{77}{256}2^m - 2$)

Dunne. *Extremal behaviour in multiagent contract negotiation*. JAIR-05.

Length of restricted negotiations

Can we construct a negotiation instance like these snake-in-the-box sequences? Let $\alpha = \alpha_1, \alpha_2, \ldots \alpha_n$ be such a sequence.

Now consider two agents, and fix their utilities such that

$$u_1(B) + u_2(\overline{B}) = k \text{ if } B = \alpha_k \text{ (and 0 otherwise)}$$

Hence α is the unique sequence of 1-deals from α_1 to α_n, because:

- no shortcut from α_i to $\alpha_{j>i+1}$
- in $A_i = \alpha_i$, no other allocation is IR except $A_{i+1} = \alpha_{i+1}$

Example $m = 4$, and $\alpha = 0000|1000|1010|1110|0110|0111|0101|1101$

	B	$\overline{B}$	u_1	u_2
α_1	0000	1111	1	0
	0001	1110	0	0
	0010	1101	0	0
	0011	1100	0	0
	0100	1011	0	0
α_7	0101	1010	7	0
α_5	0110	1001	5	0
α_6	0111	1000	6	0
⋮	⋮	⋮	⋮	⋮

A last puzzling question...

Consider the following situation:

> *There are seven cards $\{0, \ldots, 6\}$, and three agents A, B and C. A and B gets three cards, while C gets one. How, via public announcements, can A and B get to know each other cards, without C knowing anything?*

Can we design efficient protocols to achieve this goal?

- here we need to model the knowledge of the agents
- epistemic logic tools are appropriate here
- logics of public announcements model updates in Kripke structures

(...)

A last puzzling question...

Consider the following situation:

> *There are seven cards* $\{0, \ldots, 6\}$*, and three agents* A, B *and* C*.* A *and* B *gets three cards, while* C *gets one. How, via public announcements, can* A *and* B *get to know each other cards, without* C *knowing anything?*

Can we design efficient protocols to achieve this goal?

- here we need to model the knowledge of the agents
- epistemic logic tools are appropriate here
- logics of public announcements model updates in Kripke structures

(...)

- there are proofs that this can be done in two announcements
- there is a proof that this cannot be done in less

H. van Ditmarsch. *The russian cards problem*. Studia Logica.

A. Cyriac & K. Krishnan. *Lower bound for the communication complexity of the russian cards problem*. ArXiv-2008.

A last puzzling question...

The "modulo 7" solution:

- A tells B the sum of his cards modulo 7 (m_A)
- B tells A the sum of his cards modulo 7 (m_B)
- but as $m_A + m_B + C$ must be 0 ($0 + 1 + \cdots + 6$ modulo 7), A knows C (hence B), and vice versa.
- suppose C holds 6:
  observe that a statement $m_A = 0$ is equivalent in his eyes as $025 \vee 034 \vee 124$
- and similarly for the others... so he can't guess any card