Analyse et mise en œuvre des communications entre agents

N. Maudet, LIP6, Sorbonne Université

May 2018

Cours FosyMA - 2018 -2019

Un exemple introductif

Example : protocols for allocating one good

Consider the following situation :

> **Problem** *: Two agents (A and B); one object to allocate. Each agent x has a valuation $v_x \in \{0, 1, 2, 3\}$ for the object.*
> **Goal** *: assign the object to the agent who values it the most (if same valuation, any agent is fine).*

Can we design efficient protocols to achieve this goal ?

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

Example : protocols for allocating one good

Consider the following situation :

> **Problem** *: Two agents (A and B) ; one object to allocate. Each agent x has a valuation $v_x \in \{0, 1, 2, 3\}$ for the object.*
>
> **Goal** *: assign the object to the agent who values it the most (if same valuation, any agent is fine).*

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 : protocols for allocating one good

Consider the following situation :

> **Problem** : *Two agents (A and B) ; one object to allocate. Each agent x has a valuation $v_x \in \{0, 1, 2, 3\}$ for the object.*
> **Goal** : *assign the object to the agent who values it the most (if same valuation, any agent is fine).*

Can we design efficient protocols to achieve this goal ?

Protocol π_1 : "English Auction"	bits
$p \leftarrow 0$, $X \leftarrow B$	
while continue :	
$p \leftarrow p + 1$	
ask X "continue ?"	1
$X \leftarrow \overline{X}$	
allocate to $\overline{X}$	
	total $\Rightarrow$ 1, 2, or 3

Example : protocols for allocating one good

Consider the following situation :

> **Problem** : *Two agents (A and B) ; one object to allocate. Each agent x has a valuation $v_x \in \{0, 1, 2, 3\}$ for the object.*
>
> **Goal** : *assign the object to the agent who values it the most (if same valuation, any agent is fine).*

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

Un outil pour analyser les communications : la complexité de communication

Communication Complexity Setting

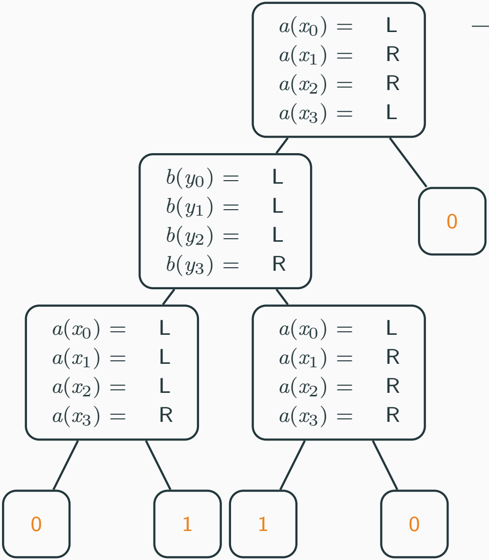
Basic communication complexity setting

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.

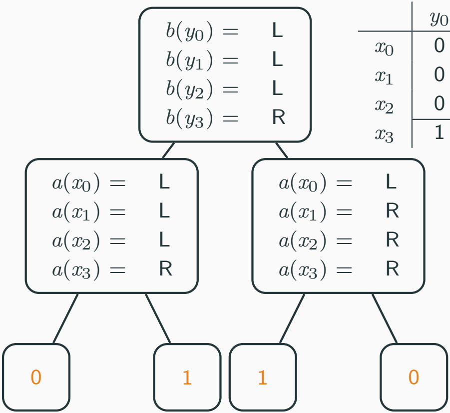
Kushilevitz & Nisan. *Communication complexity*. Cambridge Univ. Press, 1997.

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

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

Techniques pour déterminer des bornes inf

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

La technique des *fooling sets*

- 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	$\boxed{\text{B}}$	B	B	B
1	A	$\boxed{\text{B}}$	B	B
2	A	A	$\boxed{\text{B}}$	B
3	A	A	A	$\boxed{\text{B}}$

Quelques protocoles utiles

The gossip problem

Consider the following situation :

Problem *: n agents ; each agent x holding a secret X. When two agents communicate, they share their secrets.*

Goal *: reach a state where all the agents know all the secrets*

How many exchanges are needed to reach the goal ?

The gossip problem

Consider the following situation :

> **Problem** *: n agents ; each agent x holding a secret X. When two agents communicate, they share their secrets.*
>
> **Goal** *: reach a state where all the agents know all the secrets*

How many exchanges are needed to reach the goal ?

Start with 4 agents...

The gossip problem

General case : the **busy body** solution

- all the agents speak to some designated agent n-1
- who becomes expert and then communicate back to all the agents (except the last one) n-2
- hence summing up to $2n - 3$

The gossip problem

General case : the **busy body** solution

- all the agents speak to some designated agent n-1
- who becomes expert and then communicate back to all the agents (except the last one) n-2
- hence summing up to $2n - 3$

Can we do better ?

The gossip problem

General case : the **four people** solution

- each agent communicates to one of 4 people — n-4
- the four people exchange their secrets — 4
- they communicate back to the other agents — n-4
- hence summing up to — $2n - 4$

The gossip problem

But this assumes of course a centralized orchestration.

What about **distributed gossip protocols**?

Algorithm 1 : ANY

repeat
  select to agents who did not call each other
  let a call b
until *all agents are experts*;

Apt et al. *Epistemic protocols for distributed gossiping*. TARK-05.

van Ditmarsch et al. *Reachability and expectation in gossiping*. PRIMA-17.

The gossip problem

But this assumes of course a centralized orchestration.

What about **distributed gossip protocols**?

Algorithm 2 : CO

repeat
  select two agents who did not call each other
  let a call b
until *all agents are experts*;

Apt et al. *Epistemic protocols for distributed gossiping*. TARK-05.

van Ditmarsch et al. *Reachability and expectation in gossiping*. PRIMA-17.

The gossip problem

But this assumes of course a centralized orchestration.

What about **distributed gossip protocols**?

Algorithm 3 : LNS

repeat
  select two agents a such that a does not know b's secret
  let a call b
until *all agents are experts*;

Apt et al. *Epistemic protocols for distributed gossiping*. TARK-05.

van Ditmarsch et al. *Reachability and expectation in gossiping*. PRIMA-17.

Agrégation d'information

Considérons le problème suivant.

- Chaque agent détient individuellement une valeur x_i (par exemple la place qui lui reste dans le sac),
- On souhaite déterminer la place disponible par agent en moyenne dans le systeme.

Protocole Push-Sum (Kempe, Dobra, Gehrke)

A tout tour t, chaque agent maintient une somme $s_{t,i}$, initialisée à $s_{0,i} \leftarrow x_i$, ainsi qu'un poids $w_{t,i}$, initialisé à $w_{0,i} \leftarrow 1$. A chaque tour t :

1. Soit $\{(\hat{s}_r, \hat{w}_r)\}$ l'ensemble des messages reçus par i pendant le tour précédent.
2. Soit $s_{t,i} \leftarrow \sum \hat{s}_r$, et $w_{t,i} \leftarrow \sum \hat{w}_r$
3. L'agent i choisit un agent avec qui communiquer parmi ses voisins, de manière uniforme, noté $f_t(i)$
4. L'agent i envoie le message $(\frac{1}{2} s_{t,i}, \frac{1}{2} w_{t,i})$ à $f_t(i)$ et à lui-même
5. Le ratio $\frac{s_{t,i}}{w_{t,i}}$ est l'estimation de la moyenne au temps t

Protocole Push-Sum (Kempe, Dobra, Gehrke)

Les garanties de convergence de ce protocole sont spectaculaires :

> Le protocole Push-Sum converge vers une “bonne” estimation de la moyenne en $\mathcal{O}(\log n)$ tours. Comme chaque tour implique l'envoi de n messages, on obtient $\mathcal{O}(n \log n)$ messages au total.

Notons que si le protocole est présenté sous forme de "tours", ce qui suggère une approche synchrone, le protocole peut très bien être utilisé en pratique de manière asynchrone : il suffit à chaque agent de se fier à sa propre horloge et de décider à quel moment effectuer le "push" (transmettre le message au voisin sélectionné).

Note : Dans ce cas toutefois, la garantie sur la vitesse de convergence est simplement conjecturée par les auteurs.

Protocole d'alerte

Considérons le problème suivant :

- on souhaite déclencher une alerte lorsqu'un certain nombre d'observations ont été effectuées (S),
- ces observations sont réalisées par un ensemble d'agents sentinelles (k),
- on suppose les observations distinctes, pour chaque agent.

Protocole d'alerte

Considérons le problème suivant :

- on souhaite déclencher une alerte lorsqu'un certain nombre d'observations ont été effectuées (S),
- ces observations sont réalisées par un ensemble d'agents sentinelles (k),
- on suppose les observations distinctes, pour chaque agent.

Solution naive chaque agent envoie un bit pour chaque observation effectuée au coordinateur

Protocole d'alerte par seuil adaptatif

Idee il faut plusieurs observations sur chaque site avant de pouvoir declencher le niveau d'alerte. Plus precisément, au moins l'une des sentinelles doit avoir realisé S/k observations

Algorithm 4 : Protocole des seuils adaptatifs

Chaque sentinelle commence avec un seuil individuel d'alerte $t \leftarrow S/k$

repeat

  repeat

    A chaque observation par i, $n_i \leftarrow n_i + 1$

  until *une sentinelle* s *ait réalisé* t *observations* ;

  La sentinelle s envoie un message au centre

  Le centre collecte les n_i de chaque sentinelle

  $S \leftarrow S - \sum n_i$ (maj nombre d'observations manquantes)

  $t \leftarrow S/k$ (maj seuil)

until *S=k* ;

repeat

  Envoyer chaque observation au centre : $S \leftarrow S - 1$

until *S=0* ;

Exemple (exercice TD)

Condorcet winner : query complexity

Consider the following situation :

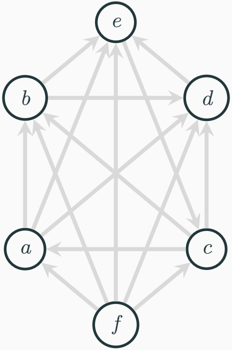
> **Problem** *: n agents with preferences over m options expressed as linear orders, inducing a majority graph.*
>
> **Goal** *: determine whether one option beats all the other ones in pairwise comparison*

Example : b is a Condorcet winner

$$\begin{array}{ll} 1: & a \succ b \succ c \\ 2: & b \succ c \succ a \\ 3: & c \succ b \succ a \end{array}$$

How many edges of the majority graph do we need to query to answer this question ?

Condorcet winner : query complexity

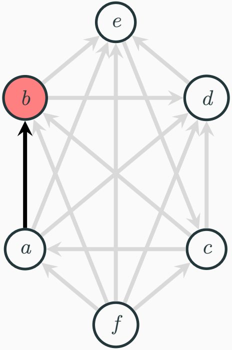


Condorcet winner : query complexity

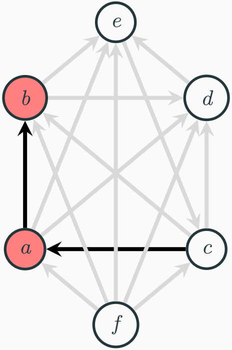
Analyzed under the **query complexity** model.

- 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
- Of course $p(p-1)/2$ are sufficient. Can we do better ?
- A property is evasive if all edges must be queried (in the worst case)

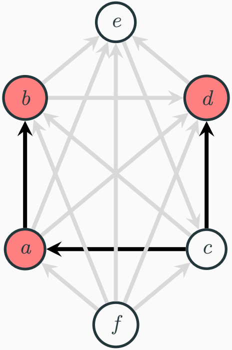
Condorcet winner : query complexity



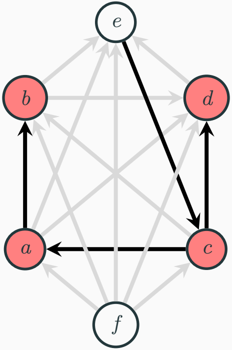
Condorcet winner : query complexity



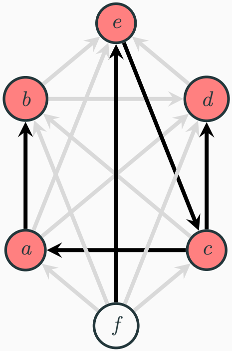
Condorcet winner : query complexity



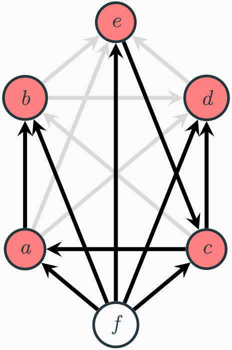
Condorcet winner : query complexity



Condorcet winner : query complexity



Condorcet winner : query complexity



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

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 ?

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 ?

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.

More about this...

Balasubramanian et al.. *Finding scores in tournaments*. J. of Algorithms, 1997.

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

Dey. *Query Complexity of Tournament Solutions*. ArXiv, 2018.