Initial knowledge and impossibility results in distributed computing

David Ilcinkas

CNRS & Université de Bordeaux (LaBRI)

GT Complexité et Algorithmes
November 22, 2012

Graph algorithms

Centralized case

- 1 process/algorithm
- Complete knowledge of the instance/graph

Distributed case

- n processes (one per vertex) cooperating in a graph
- Partial (initial) knowledge of the instance/graph

Graph algorithms

Centralized case

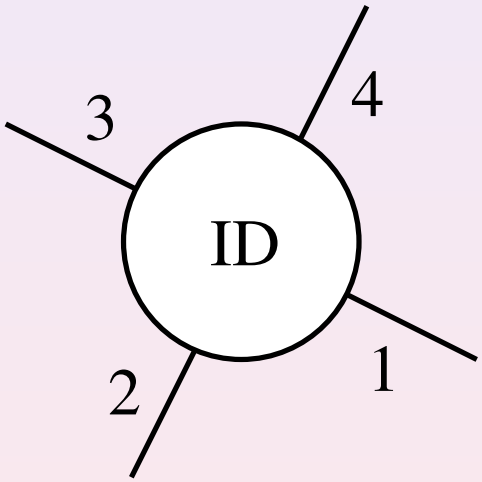
- 1 process/algorithm
- Complete knowledge of the instance/graph

Distributed case

- n processes (one per vertex) cooperating in a graph
- Partial (initial) knowledge of the instance/graph

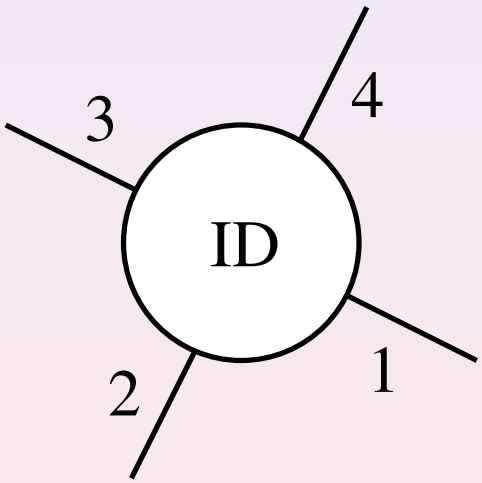
Types of knowledge

Local knowledge



Types of knowledge

Local knowledge



Global knowledge

- number n of vertices
- diameter
- maximum degree
- genus, girth, treewidth, etc.

Knowledge and performance

Observation

The existence/quality of the algorithmic solutions often depends on the availability of global knowledge about the network.

- [Lynch, PODC 1989]: A hundred impossibility proofs for distributed computing.
- [Fich, Ruppert, Distributed Computing, 2003]: Hundreds of impossibility results for distributed computing.

Some examples

Synchronous deterministic broadcast in n-node radio networks of diameter D

- Time $\Omega(n \log D)$ if no information
  [Clementi, Monti, Silvestri, TCS 2003]
- Time $O(D + \log^2 n)$ if complete knowledge
  [Kowalski, Pelc, Distributed Computing 2006]

Wakeup in arbitrary networks

[Awerbuch, Goldreich, Peleg, Vainish, J. of ACM, 1990]
Initial knowledge of the topology within radius ρ

- $\Theta(n^{1+\Theta(1)/\rho})$ messages of bounded length

Some examples

Synchronous deterministic broadcast in n-node radio networks of diameter D

- Time $\Omega(n \log D)$ if no information
  [Clementi, Monti, Silvestri, TCS 2003]
- Time $O(D + \log^2 n)$ if complete knowledge
  [Kowalski, Pelc, Distributed Computing 2006]

Wakeup in arbitrary networks

[Awerbuch, Goldreich, Peleg, Vainish, J. of ACM, 1990]
Initial knowledge of the topology within radius ρ

- $\Theta(n^{1+\Theta(1)/\rho})$ messages of bounded length

On the need of a priori knowledge

[A. Korman, J.-S. Sereni and L.Viennot. Toward more Localized Local Algorithms: Removing Assumptions concerning Global Knowledge. PODC'2011]

Result

A rather general method to transform an algorithm so that it does not need initial global knowledge anymore.

- the transformed algorithm has, up to a small constant, the same time complexity
- the method applies to a wide family of both deterministic and randomized algorithms; in particular most of those for MIS, Maximal Matching, coloring...

On the need of a priori knowledge

[A. Korman, J.-S. Sereni and L.Viennot. Toward more Localized Local Algorithms: Removing Assumptions concerning Global Knowledge. PODC'2011]

Result

A rather general method to transform an algorithm so that it does not need initial global knowledge anymore.

- the transformed algorithm has, up to a small constant, the same time complexity
- the method applies to a wide family of both deterministic and randomized algorithms; in particular most of those for MIS, Maximal Matching, coloring...

Case study: graph exploration with termination

Graph exploration...

A mobile entity (robot/agent) must visit at least once each vertex of an anonymous graph.

...with termination

The mobile entity must stop after finite time, i.e. detect termination.

Anonymous :

- vertices with no IDs
- local labeling of the edges (port numbers)

Case study: graph exploration with termination

Graph exploration...

A mobile entity (robot/agent) must visit at least once each vertex of an anonymous graph.

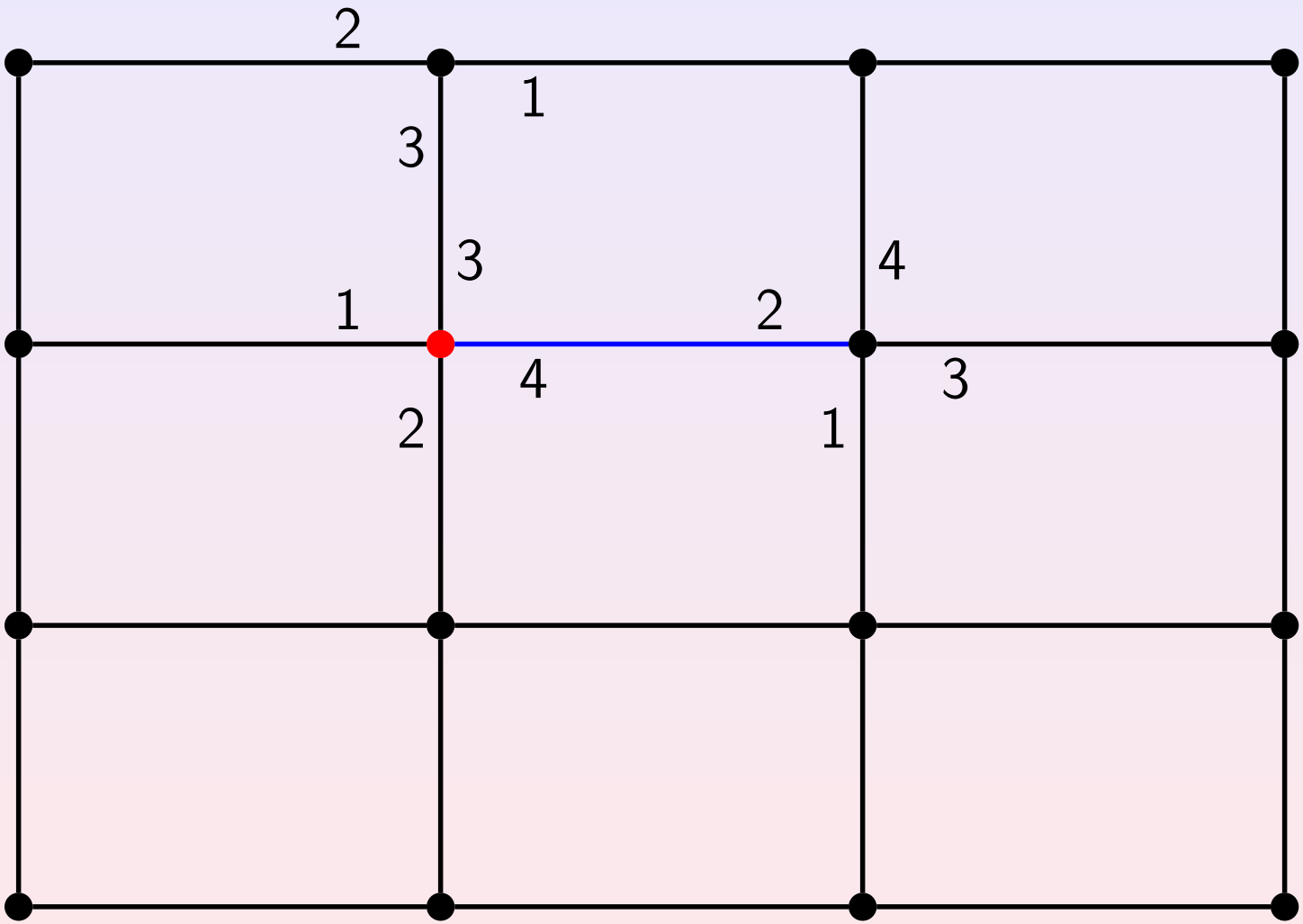
...with termination

The mobile entity must stop after finite time, i.e. detect termination.

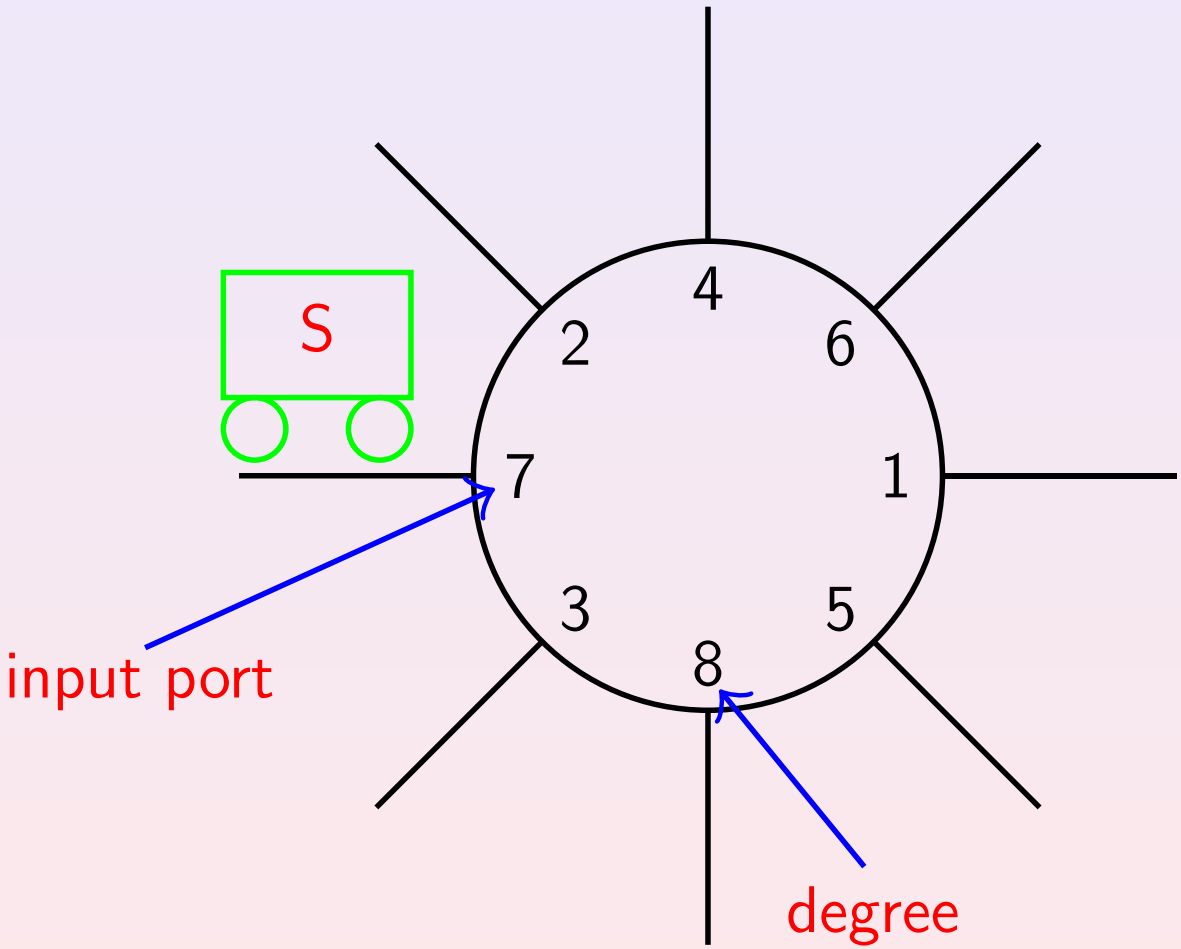
Anonymous :

- vertices with no IDs
- local labeling of the edges (port numbers)

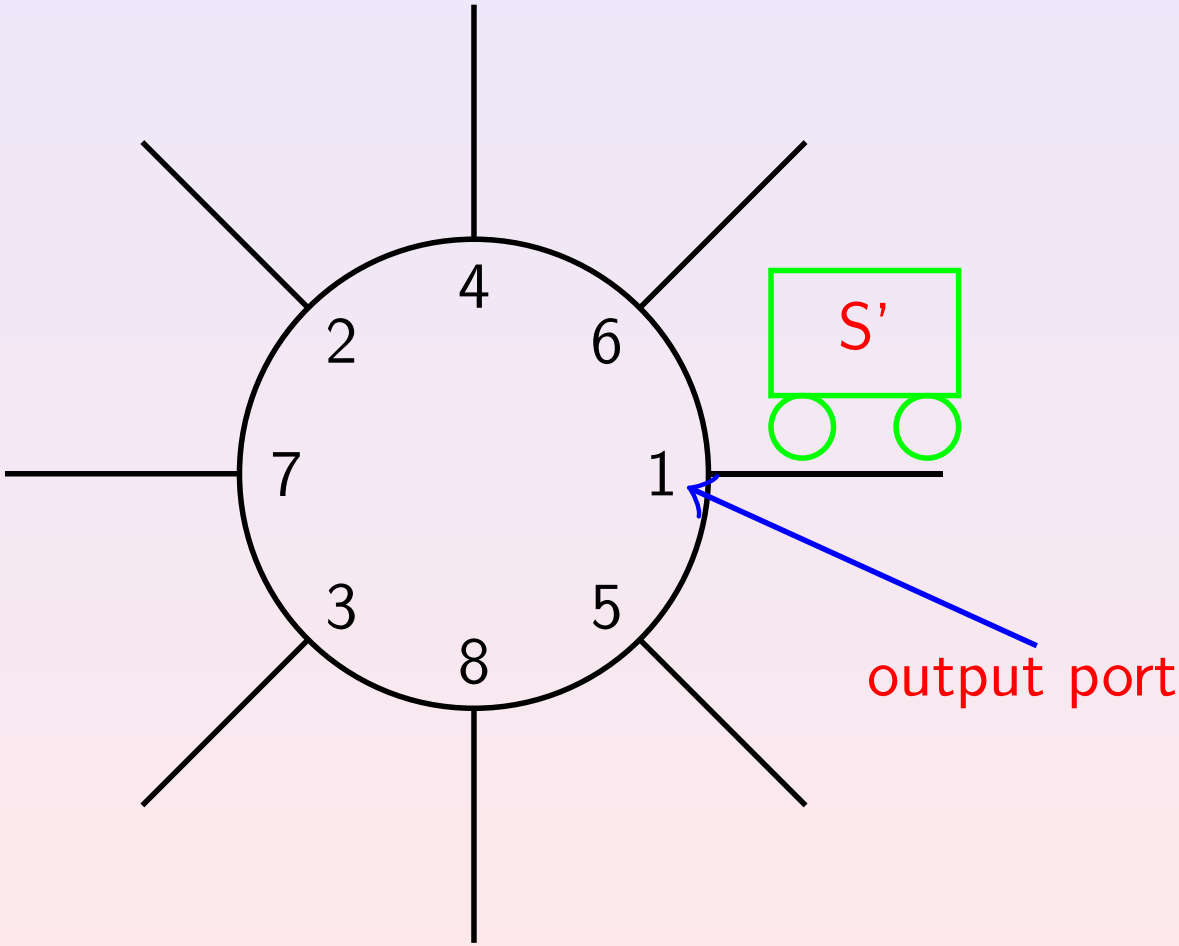
Example of an anonymous graph



Agent's model



Agent's model



Positive result

Theorem

There exists an agent able to solve the graph exploration with termination problem if it knows an upper bound $\hat{n}$ on the number n of nodes of the to-be-explored graph.

Proof

Explore all paths of length $\hat{n}$ coming out of the starting vertex, and then stop.

Corollary

The knowledge of an upper bound on n is sufficient to solve graph exploration with termination.

Is this knowledge necessary?

Positive result

Theorem

There exists an agent able to solve the graph exploration with termination problem if it knows an upper bound $\hat{n}$ on the number n of nodes of the to-be-explored graph.

Proof

Explore all paths of length $\hat{n}$ coming out of the starting vertex, and then stop.

Corollary

The knowledge of an upper bound on n is sufficient to solve graph exploration with termination.

Is this knowledge necessary?

Positive result

Theorem

There exists an agent able to solve the graph exploration with termination problem if it knows an upper bound $\hat{n}$ on the number n of nodes of the to-be-explored graph.

Proof

Explore all paths of length $\hat{n}$ coming out of the starting vertex, and then stop.

Corollary

The knowledge of an upper bound on n is sufficient to solve graph exploration with termination.

Is this knowledge necessary?

Positive result

Theorem

There exists an agent able to solve the graph exploration with termination problem if it knows an upper bound $\hat{n}$ on the number n of nodes of the to-be-explored graph.

Proof

Explore all paths of length $\hat{n}$ coming out of the starting vertex, and then stop.

Corollary

The knowledge of an upper bound on n is sufficient to solve graph exploration with termination.

Is this knowledge necessary?

Negative result

Theorem

The knowledge of an upper bound on n is necessary to solve graph exploration with termination.

Proof

- Assume there exists an agent exploring without the knowledge of an upper bound on n.
- Agent on a triangle: termination at time k.
- Agent on a cycle C_{2k}: termination before exploration

Negative result

Theorem

The knowledge of an upper bound on n is necessary to solve graph exploration with termination.

Proof

- Assume there exists an agent exploring without the knowledge of an upper bound on n.
- Agent on a triangle: termination at time k
- Agent on a cycle C_{2k}: termination before exploration

Negative result

Theorem

The knowledge of an upper bound on n is necessary to solve graph exploration with termination.

Proof

- Assume there exists an agent exploring without the knowledge of an upper bound on n.
- Agent on a triangle: termination at time k
- Agent on a cycle C_{2k}: termination before exploration

Are you convinced?

Negative result

Theorem

The knowledge of an upper bound on n is necessary to solve graph exploration with termination.

Proof

- Assume there exists an agent exploring without the knowledge of an upper bound on n.
- Agent on a triangle: termination at time k
- Agent on a cycle C_{2k}: termination before exploration

Are you convinced?
Is it a satisfactory proof?

Negative result

Theorem

The knowledge of an upper bound on n is necessary to solve graph exploration with termination.

Proof

- Assume there exists an agent exploring without the knowledge of an upper bound on n.
- Agent on a triangle: termination at time k
- Agent on a cycle C_{2k}: termination before exploration

Are you convinced?
Is it a satisfactory proof? No: parity of n.

Attempt to formalize

A priori knowledge / advice

- The agent has to explore graph G.
- At the beginning of the algorithm's execution, the agent is provided an advice (a priori knowledge) from an oracle $\mathcal{O}$ as a binary string $\mathcal{O}(G)$.

$\mathcal{O}$ induces a partition $(F_0, F_1, \cdots)$ of the graph family $\mathcal{G}$: $F_i = \{G \in \mathcal{G} | \mathcal{O}(G) = i\}$

To know an upper bound on n

$$\forall i \; \exists \hat{n}_i \; \forall G \in F_i \;\; n(G) \leq \hat{n}_i$$

Attempt to formalize

A priori knowledge / advice

- The agent has to explore graph G.
- At the beginning of the algorithm's execution, the agent is provided an advice (a priori knowledge) from an oracle $\mathcal{O}$ as a binary string $\mathcal{O}(G)$.

$\mathcal{O}$ induces a partition $(F_0, F_1, \cdots)$ of the graph family $\mathcal{G}$:
$F_i = \{G \in \mathcal{G} | \mathcal{O}(G) = i\}$

To know an upper bound on n

$$\forall i \; \exists \hat{n}_i \; \forall G \in F_i \;\; n(G) \leq \hat{n}_i$$

Attempt to formalize

A priori knowledge / advice

- The agent has to explore graph G.
- At the beginning of the algorithm's execution, the agent is provided an advice (a priori knowledge) from an oracle $\mathcal{O}$ as a binary string $\mathcal{O}(G)$.

$\mathcal{O}$ induces a partition $(F_0, F_1, \cdots)$ of the graph family $\mathcal{G}$:
$F_i = \{G \in \mathcal{G} | \mathcal{O}(G) = i\}$

To know an upper bound on n

$$\forall i \ \exists \hat{n}_i \ \forall G \in F_i \quad n(G) \leq \hat{n}_i$$

Negative result (bis)

Theorem

An agent without the knowledge of an upper bound on n cannot explore all cycles.

Reminder: to know upper bound: $\forall i\ \exists \hat{n}_i\ \forall G \in F_i \quad n(G) \leq \hat{n}_i$

Proof

- Assume there exists an agent exploring without the knowledge of an upper bound on n.
- $\exists i\ \forall N\ \exists G_N^{(i)} \in F_i \quad n(G_N^{(i)}) > N$
- Agent on $G_1^{(i)}$: termination at time k
- Agent on $G_k^{(i)}$: termination before exploration

Negative result (bis)

Theorem

An agent without the knowledge of an upper bound on n cannot explore all cycles.

Reminder: to know upper bound: $\forall i \; \exists \hat{n}_i \; \forall G \in F_i \quad n(G) \leq \hat{n}_i$

Proof

- Assume there exists an agent exploring without the knowledge of an upper bound on n.
- $\exists i \; \forall N \; \exists G_N^{(i)} \in F_i \quad n(G_N^{(i)}) > N$
- Agent on $G_1^{(i)}$: termination at time k
- Agent on $G_k^{(i)}$: termination before exploration

End of story?

Set of all cycles $\subset$ set of all graphs

Corollary

The knowledge of an upper bound on n is also necessary to explore all graphs.

End of story?

Set of all cycles $\subset$ set of all graphs

Corollary

The knowledge of an upper bound on n is also necessary to explore all graphs.

WRONG!
It is not monotonic...
$\mathcal{O}(G) = 0$ if G is a path and $\mathcal{O}(G) = n(G)$ otherwise

Comparison between oracles

Idea: concentrate on the information that the agent cannot compute alone

Definition: $\mathcal{O} \succeq \mathcal{O}'$

There exists a mobile agent algorithm $\mathcal{A}$ such that $\mathcal{O} + \mathcal{A}$ can simulate the output of $\mathcal{O}'$.

[Chandra, Toueg, JACM 1996], [Godard, Métivier, ToCS 2003]

Theorem (new try)

For every $\mathcal{O}$ permitting to solve the exploration in all graphs, there exists $\mathcal{O}'$ giving an upper bound, such that $\mathcal{O} \succeq \mathcal{O}'$.

Proof

$\mathcal{O}$ and the corresponding exploration algorithm $\mathcal{A}_{\mathcal{O}}$ give an upper bound on the size of G: the stopping time of $\mathcal{A}_{\mathcal{O}}$.

Comparison between oracles

Idea: concentrate on the information that the agent cannot compute alone

Definition: $\mathcal{O} \succeq \mathcal{O}'$

There exists a mobile agent algorithm $\mathcal{A}$ such that $\mathcal{O} + \mathcal{A}$ can simulate the output of $\mathcal{O}'$.

[Chandra, Toueg, JACM 1996], [Godard, Métivier, ToCS 2003]

Theorem (new try)

For every $\mathcal{O}$ permitting to solve the exploration in all graphs, there exists $\mathcal{O}'$ giving an upper bound, such that $\mathcal{O} \succeq \mathcal{O}'$.

Proof

$\mathcal{O}$ and the corresponding exploration algorithm $\mathcal{A}_{\mathcal{O}}$ give an upper bound on the size of G: the stopping time of $\mathcal{A}_{\mathcal{O}}$.

Comparison between oracles

Idea: concentrate on the information that the agent cannot compute alone

Definition: $\mathcal{O} \succeq \mathcal{O}'$

There exists a mobile agent algorithm $\mathcal{A}$ such that $\mathcal{O} + \mathcal{A}$ can simulate the output of $\mathcal{O}'$.

[Chandra, Toueg, JACM 1996], [Godard, Métivier, ToCS 2003]

Theorem (new try)

For every $\mathcal{O}$ permitting to solve the exploration in all graphs, there exists $\mathcal{O}'$ giving an upper bound, such that $\mathcal{O} \succeq \mathcal{O}'$.

Proof

$\mathcal{O}$ and the corresponding exploration algorithm $\mathcal{A}_{\mathcal{O}}$ give an upper bound on the size of G: the stopping time of $\mathcal{A}_{\mathcal{O}}$.

Comparison between oracles

Idea: concentrate on the information that the agent cannot compute alone

Definition: $\mathcal{O} \succeq \mathcal{O}'$

There exists a mobile agent algorithm $\mathcal{A}$ such that $\mathcal{O} + \mathcal{A}$ can simulate the output of $\mathcal{O}'$.

[Chandra, Toueg, JACM 1996], [Godard, Métivier, ToCS 2003]

Theorem (new try)

For every $\mathcal{O}$ permitting to solve the exploration in all graphs, there exists $\mathcal{O}'$ giving an upper bound, such that $\mathcal{O} \succeq \mathcal{O}'$.

Proof

$\mathcal{O}$ and the corresponding exploration algorithm $\mathcal{A}_{\mathcal{O}}$ give an upper bound on the size of G: the stopping time of $\mathcal{A}_{\mathcal{O}}$.

End of story? (bis)

Corollary

The equivalence class of every oracle giving an upper bound on n is minimal for the problem of exploring all graphs.

"Problem" 1: Can several minimal classes exist?

"Problem" 2

The equivalence class of every oracle giving an upper bound on n is minimal for the problem of exploring all paths.

End of story? (bis)

Corollary

The equivalence class of every oracle giving an upper bound on n is minimal for the problem of exploring all graphs.

"Problem" 1: Can several minimal classes exist?

"Problem" 2

The equivalence class of every oracle giving an upper bound on n is minimal for the problem of exploring all paths.

End of story? (bis)

Corollary

The equivalence class of every oracle giving an upper bound on n is minimal for the problem of exploring all graphs.

"Problem" 1: Can several minimal classes exist?

"Problem" 2

The equivalence class of every oracle giving an upper bound on n is minimal for the problem of exploring all paths.

Back to the starting point

Theorem

The empty/neutral oracle is strictly weaker than every exploration oracle for the family of all graphs.

Proof

- Agent on an triangle: termination at time k
- Agent on a cycle C_{2k}: termination before exploration

Can't we say anything more?

Theorem

Every oracle inducing a finite number of parts F_i is strictly weaker than every exploration oracle for the family of all graphs.

Back to the starting point

Theorem

The empty/neutral oracle is strictly weaker than every exploration oracle for the family of all graphs.

Proof

- Agent on an triangle: termination at time k
- Agent on a cycle C_{2k}: termination before exploration

Can't we say anything more?

Theorem

Every oracle inducing a finite number of parts F_i is strictly weaker than every exploration oracle for the family of all graphs.

Back to the starting point

Theorem

The empty/neutral oracle is strictly weaker than every exploration oracle for the family of all graphs.

Proof

- Agent on an triangle: termination at time k
- Agent on a cycle C_{2k}: termination before exploration

Can't we say anything more?

Theorem

Every oracle inducing a finite number of parts F_i is strictly weaker than every exploration oracle for the family of all graphs.

Back to the starting point

Theorem

The empty/neutral oracle is strictly weaker than every exploration oracle for the family of all graphs.

Proof

- Agent on an triangle: termination at time k
- Agent on a cycle C_{2k}: termination before exploration

Can't we say anything more?

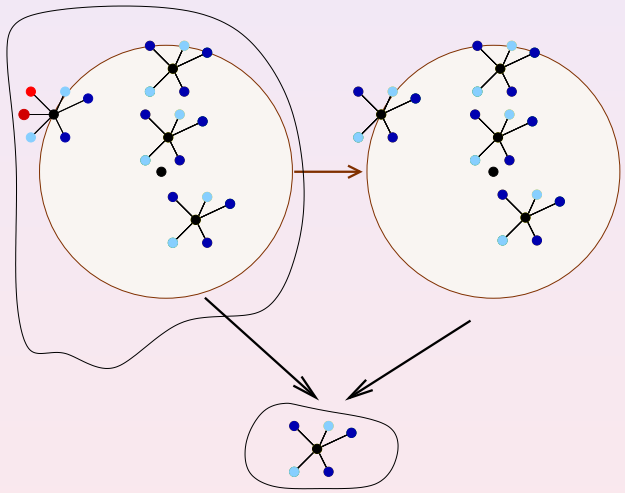
Theorem

Every oracle inducing a finite number of parts F_i is strictly weaker than every exploration oracle for the family of all graphs.

Precise characterisation

Definition

The graph H is a quasi-covering of radius r of the graph G if there exists a vertex u of H such that the ball in H centered at u and of radius r is indistinguishable from G.



See [J. Chalopin, E. Godard and Y. Métivier. Local Terminations and Distributed Computability in Anonymous Networks. Distributed Computing 2008]

Precise characterisation

Definition

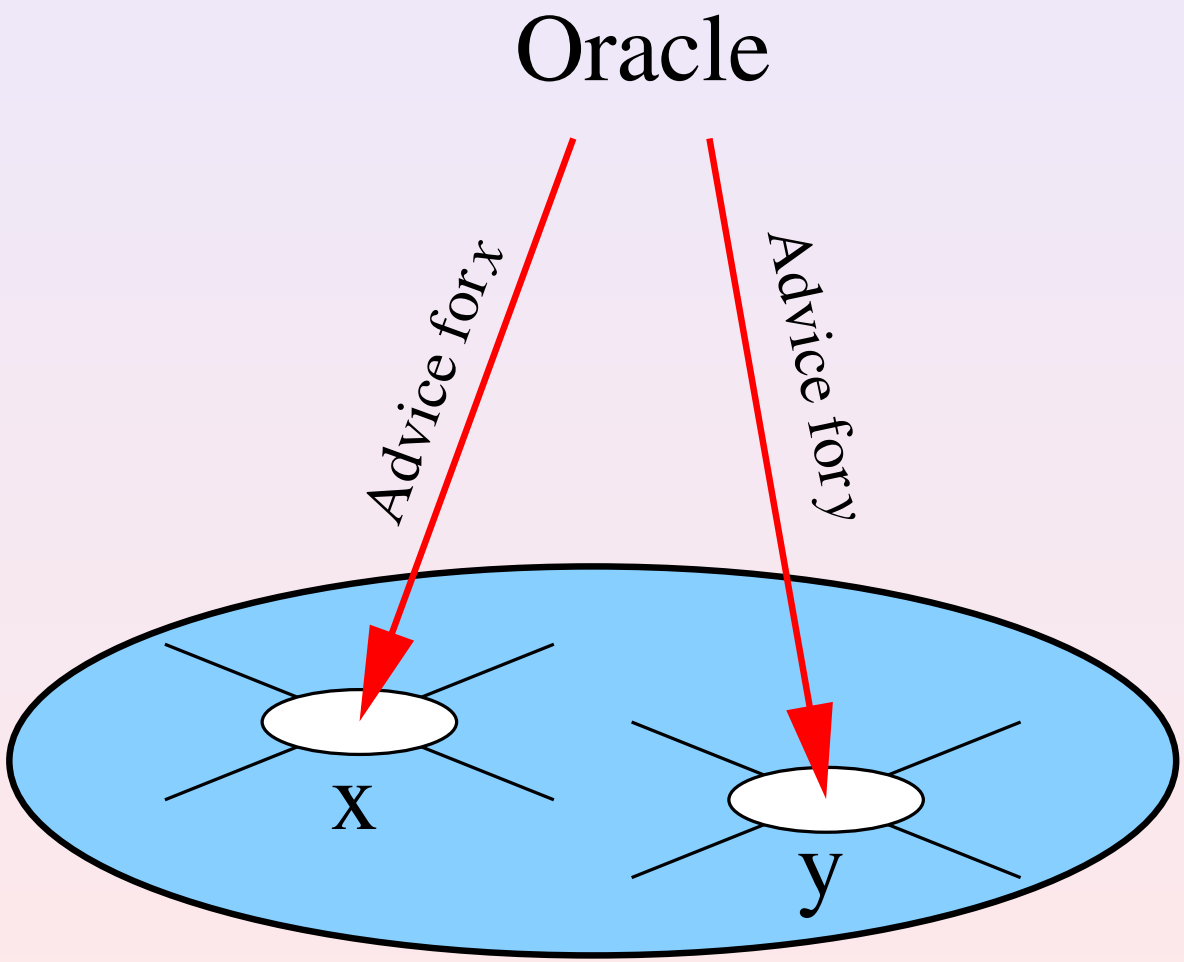
The graph H is a quasi-covering of radius r of the graph G if there exists a vertex u of H such that the ball in H centered at u and of radius r is indistinguishable from G.

Characterisation

An oracle allows an agent to solve exploration in a family $\mathcal{F}$ if and only if there exists no part F_i of $\mathcal{F}$ containing quasi-coverings of unbounded radius of a common graph $G \in F_i$.

Global knowledge given by an oracle

General framework in distributed computing:



Distributed computing with advice

Design of two elements

the oracle $\mathcal{O}$

- takes the whole graph G as input
- returns advice $\mathcal{O}(G, x)$ to every node (or agent) x

the algorithm $\mathcal{A}$

- executed locally by every node (or agent)
- using the pieces of advice given by the oracle

Qualitative and quantitative approaches

Drawback of the qualitative approach

Difficult to compare

- Algorithm knowing n
- Algorithm knowing D
- Algorithm knowing the neighborhood

Quantitative approach

What is the minimum size of advice permitting to solve problem $\mathcal{P}$?

Quantitative questions about the required knowledge, regardless of what kind of knowledge is supplied.

Qualitative and quantitative approaches

Drawback of the qualitative approach

Difficult to compare

- Algorithm knowing n
- Algorithm knowing D
- Algorithm knowing the neighborhood

Quantitative approach

What is the minimum size of advice permitting to solve problem $\mathcal{P}$?

Quantitative questions about the required knowledge, regardless of what kind of knowledge is supplied.

One possible application

Two fundamental tasks

Disseminating a message M from a source to all the nodes of a network

- Wakeup: a node cannot communicate before it has received the message M
- Broadcast: a node can communicate at any time

Problem

Achieving these tasks using a number of messages linear in n

One possible application

Two fundamental tasks

Disseminating a message M from a source to all the nodes of a network

- Wakeup: a node cannot communicate before it has received the message M
- Broadcast: a node can communicate at any time

Problem

Achieving these tasks using a number of messages linear in n

Results

[P. Fraigniaud, D. Ilcinkas, and A. Pelc. *Communication algorithms with advice*. JCSS 2010]

Wakeup

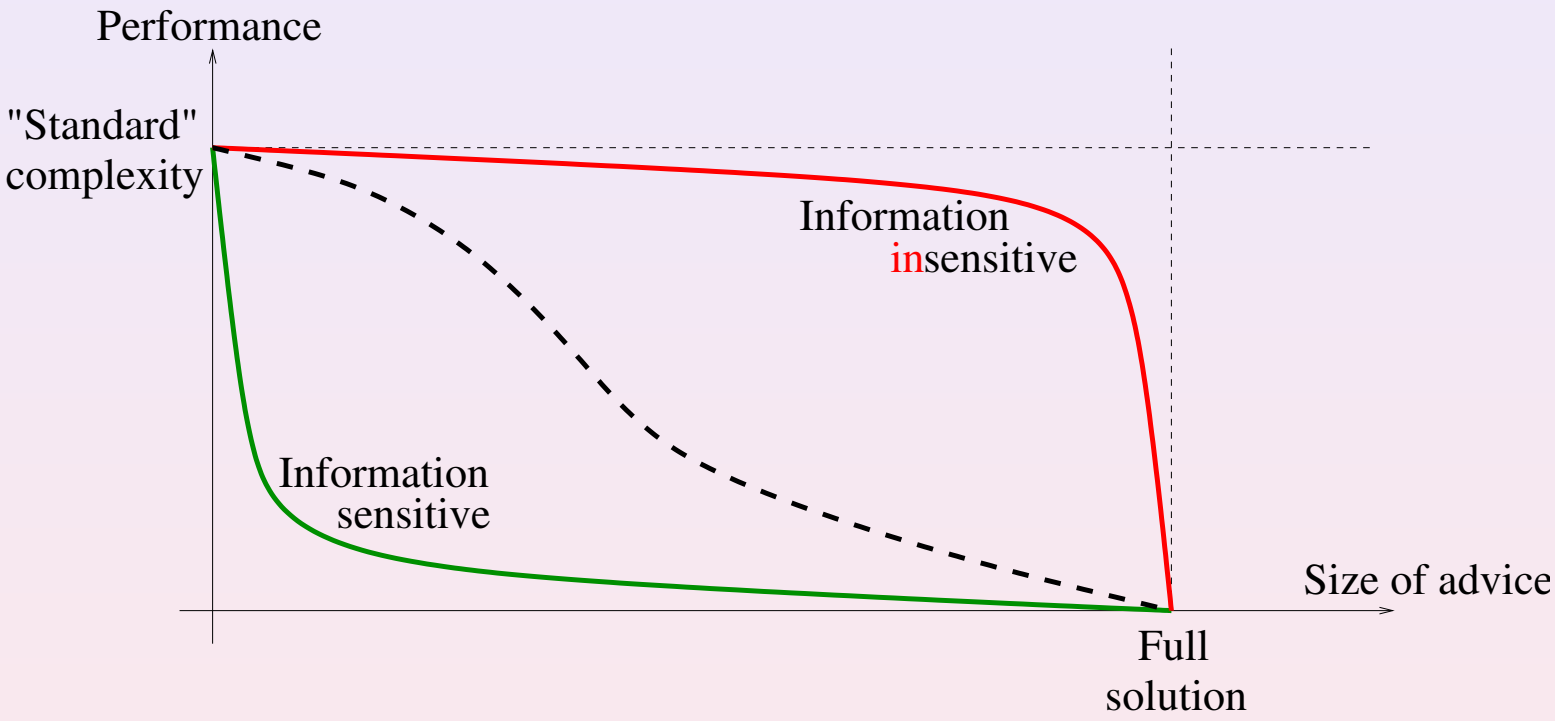
Minimum oracle size is $\Theta(n \log n)$ bits.

Broadcast

Minimum oracle size is $\Theta(n)$ bits.

Permits to clearly distinguish the two problems.

Another possible point of view



Two things to remember

- Proving that some initial knowledge is necessary must be done with extra care.
- The advice/oracle point of view can be insightful.

Thank you

for your attention