How difficult is it to compute the winner(s) of a game?

Olivier Hudry

Télécom ParisTech, Paris, France

The context: who is the winner?

- A game with n players.
- For any pair $\{i, j\}$ of distinct players, i and j compete together.
- We assume that there is no tie: i defeats j or conversely.
- We would like to rank the players or at least to choose a winner.

TELECOM PARIS
école nationale supérieure des télécommunications

The lack of transitivity

- The results obtained from the $n(n-1)/2$ possible matches do not always provide a transitive structure: i may defeat j, while j defeats k who, in his turn, defeats i.
- The result is a (*round-robin*) *tournament*, i.e. an antisymmetric, complete, binary relation, and not necessarily a linear order (= transitive tournament).
- Question: how to rank the players from the obtained tournament or at least how to determine a winner (or several winners)?
- This leads to the so-called *tournament solutions*.

A link with the context of voting theory

- Similar issues appear in the context of voting theory when a pairwise comparison method (Condorcet, 1785) is applied:
  - ⇨ n candidates;
  - ⇨ we compute the number m_{xy} of voters who prefer candidate x to candidate y;
  - ⇨ majority rule: x is collectively preferred to y if $m_{xy} > m_{yx}$.
- The result is still a tournament (called the *majority tournament*), even if the voters´ preferences are linear orders (« voting paradox », « Condorcet effect », « Condorcet cycle »...).

TELECOM PARIS
école nationale supérieure des télécommunications

The associated graph

- We build a *directed graph* $T = (X, A)$ where
  - ⇨ X = set of players;
  - ⇨ there is an *arc* (= directed edge) from x to y if x defeats y.

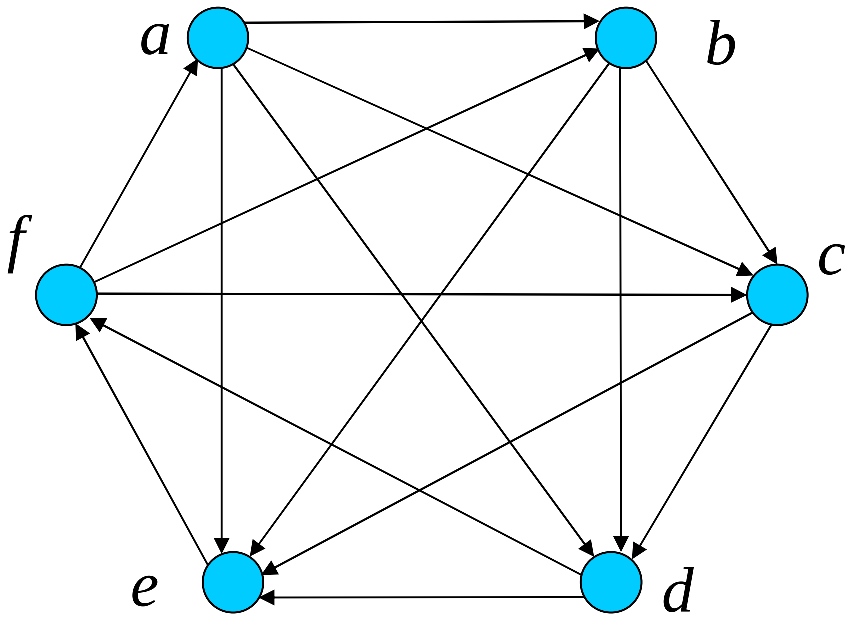
- As there is no tie, T is a *tournament* (= complete asymmetric directed graph).

T may be transitive (then, it is a linear order), but not necessarily.

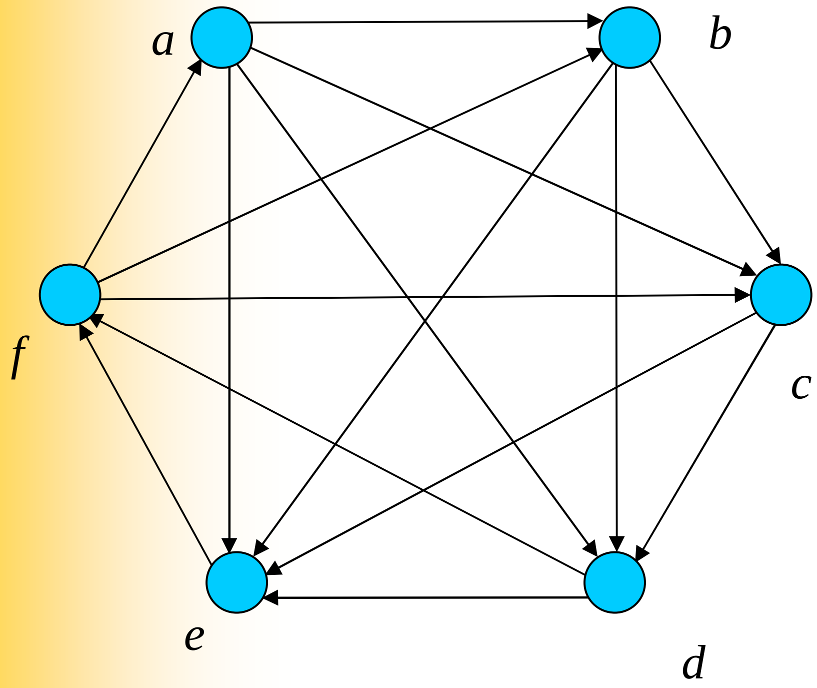
TELECOM PARIS
école nationale supérieure des télécommunications

An example

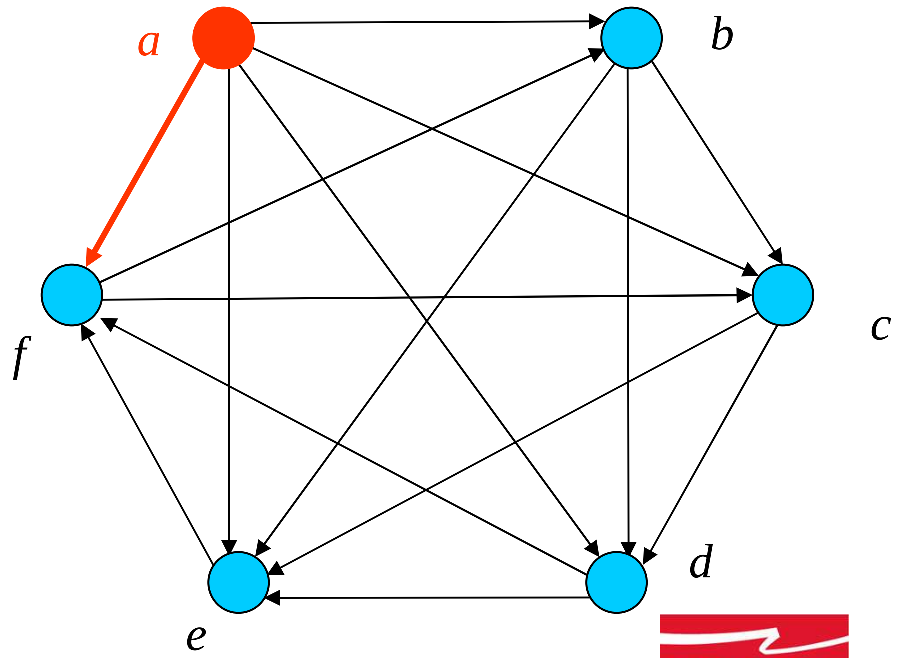
- $n = 6$ players: a, b, c, d, e, f.



- An *absolute winner* (a *Condorcet winner* for an election...) is a player W who defeats any other player; in T, all the arcs leave W; the outdegree of W is equal to $n - 1$. When there exists an absolute winner, there is only one.



no absolute winner



a = absolute winner

Tournament solution S

Let $T = (X, A)$ be a tournament on n vertices.

- A *tournament solution* is any mapping S satisfying:

  $S: T \to S(T) \subseteq X$ with $S(T) \neq \varnothing$

  $S(T) = \{$*winners* of T with respect to $S\}$.

- How to design S?

- What is the complexity of S?

  * S is *polynomial* if there exists a polynomial algorithm to compute $S(T)$;
  * S is *NP-hard* if the computation of $S(T)$ is NP-hard.

TELECOM PARIS
école nationale supérieure des télécommunications

Main features of the theory of algorithmic complexity

- Combinatorial optimization problem (COP): given a finite set X and a function f defined from X to $\{0, 1, 2, ...\}$, compute the minimum of f over X:

$$\text{minimize } f(x) \text{ with } x \in X.$$

- Decision problem: problem in which we set a question with « yes » or « no » as its answer. Ex:

  1. Composite number: given an integer n, do there exist two integers $p > 1$ and $q > 1$ with $n = pq$?

  2. DP (Decision problem associated with COP): given X, f and an integer K, does there exist $x \in X$ with $f(x) \leq K$?

TELECOM PARIS
école nationale supérieure des télécommunications

Links between COP and DP

- Any algorithm solving COP may solve DP: it is sufficient to compute the minimum of f and to compare it to K.
- Conversely, any algorithm A solving DP can be used to solve COP:

  - start with $K = f(x_0)$ where x_0 denotes any element of X

  - while the answer provided by A is « yes », do:

  $K \leftarrow K - 1.$

  The last value of K for which the answer is « yes » provides the minimum of f.

Complexity of an algorithm

- To solve a problem Π, we need an *algorithm*, i.e. a method designed to solve Π. The efficiency of an algorithm A can be measured by its (time) *complexity*.
- The complexity of A is given by the number of elementary operations (such as the additions, the comparisons, and so on) performed to solve the considered instance I.
- If the complexity of A can be upper-bounded by a polynomial in the size of I (here, n), A is said to be *polynomial* and Π also is said to be *polynomial*.

Class P, class NP

- P = {polynomial decision problems}.
- Many problems are not known to belong to P.
- NP = {decision problems such that we can check the answer « yes », in polynomial time, thanks to some information (*certificate*) guessed and provided by somebody else}.
- Ex. 1. Composite number belongs to NP.

  2. DP often belongs to NP: it is sufficient to « guess » an appropriate $x \in X$ with $f(x) \leq K$ and to be able to check that x fits in polynomial time.
- $P \subseteq NP$. Open problem: $P = NP$ or $P \subset NP$?

TELECOM PARIS
école nationale supérieure des télécommunications

NP-complete problems, NP-hard problems

- For two problems Π_1 and Π_2, we say (approximation…) that Π_2 is at least as difficult as Π_1 ($\Pi_1 \leq \Pi_2$) if the polynomiality of Π_2 would involve the one of Π_1.
- An *NP-complete problem* Π is a decision problem belonging to NP and which is at least as difficult as any other problem of NP:

$$1.\ \Pi \in \mathrm{NP}; \qquad 2.\ \forall\, \Pi' \in \mathrm{NP},\ \Pi' \leq \Pi.$$

- An *NP-hard problem* is a problem at least as difficult as any problem of NP (or, equivalently, as difficult as any NP-complete problem).
- Usually, DP is polynomial if and only if COP is polynomial. When DP is NP-complete, COP is NP-hard.

CPU time (1000 operations per second)

n	10	20	30	40	50
$\log_{10}(n)$					
n					
n^2					
n^3					
n^5					
2^n					
10^n					
$n\,!$					
n^n					

CPU time (1000 operations per second)

n	10	20	30	40	50
$\log_{10}(n)$	0.001 s	0.0013 s	0.0015 s	0.0016 s	0.0017 s
n	0.01 s	0.02 s	0.03 s	0.04 s	0.05 s
n^2	0.1 s	0.4 s	0.9 s	1.6 s	2.5 s
n^3	1 s	8 s	27 s	64 s	125 s
n^5	1.7 mn	53.3 mn	6.75 h	28.3 h	3.6 days
2^n					
10^n					
$n\,!$					
n^n					

CPU time (1000 operations per second)

n	10	20	30	40	50
$\log_{10}(n)$	0.001 s	0.0013 s	0.0015 s	0.0016 s	0.0017 s
n	0.01 s	0.02 s	0.03 s	0.04 s	0.05 s
n^2	0.1 s	0.4 s	0.9 s	1.6 s	2.5 s
n^3	1 s	8 s	27 s	64 s	125 s
n^5	1.7 mn	53.3 mn	6.75 h	28.3 h	3.6 days
2^n	1 s	17.5 mn	12.4 days	34.9 years	357 cent.
10^n	116 days	3×10^7 centuries	3×10^{17} centuries	3×10^{27} centuries	3×10^{37} centuries
$n!$	1 h	7.7×10^5 centuries	8.4×10^{19} centuries	2.6×10^{35} centuries	9.6×10^{51} centuries
n^n	116 days	3.3×10^{13} centuries	6.5×10^{31} centuries	3.8×10^{51} centuries	2.8×10^{72} centuries

Copeland solution C (1951)

- A vertex x is a *Copeland winner* if the number of players defeated by x (called the *Copeland score* $s(x)$ of x) is maximum.

Ex.

Scores:

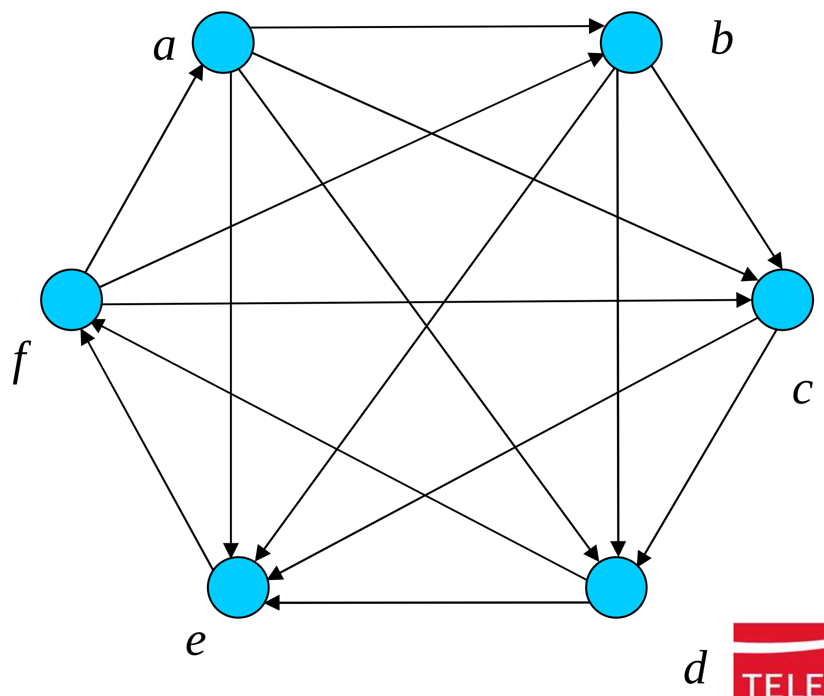
$s(a) = 4$, $s(b) = 3$,

$s(c) = 2$, $s(d) = 2$,

$s(e) = 1$, $s(f) = 3$;

the Copeland winner is a

(preorder: $a > b \sim f > c \sim d > e$).



Zermelo solution Z (1929): maximum likelihood

- Let $p(x, y)$ denote the probability that x defeats y.

  Assume that the results are independent: the fact that x defeats y does not provide information about the result between z and t.

  Assume that each player x is characterized by a *strength* w_x such that we have: $p(x, y) = w_x / (w_x + w_y)$.

  Then the probability to obtain T knowing the strengths w_x is:

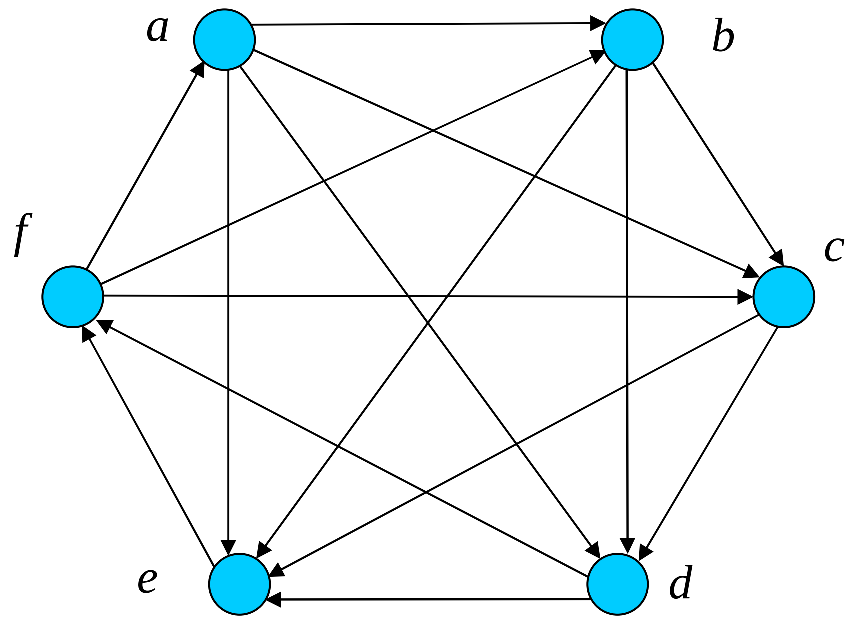
$$p(T / \{w_x\}) = \prod_{(x,y)\in A} \frac{w_x}{w_x + w_y}$$

- Given T, Zermelo's maximum likelihood method consists in computing the strengths w_x^* maximizing $p(T / \{w_x\})$ and then in ranking the players according to the decreasing values of the w_x^*'s.

- Ex.

Maximizing $\prod_{(x,y)\in A} \frac{w_x}{w_x + w_y}$

with $\sum_x w_x = 1$



gives: $w_a = 0.27$, $w_b = w_f = 0.18$, $w_c = w_d = 0.14$, $w_e = 0.09$.

The Zermelo winner is a (preorder: $a > b \sim f > c \sim d > e$).

- Theorem (L.R. Ford Jr, 1957)

Copeland method and Zermelo method lead to the same winners (more precisely, the same rankings).

Uncovered set *UC* (Fishburn, 1977, Miller, 1980)

- A player x is *covered* by a player y in T if y defeats x and if all the players defeated by x are defeated by y: $(x, z) \in A \Rightarrow (y, z) \in A$.

  A player x is *uncovered* if no player covers x.

- The winners of T according to UC are the uncovered players of T.

- We may iterate UC to get UC^2, UC^3, … $UC^n = UC^{n+1} = \ldots = UC^\infty$.

TELECOM PARIS
école nationale supérieure des télécommunications

Uncovered set *UC* (Fishburn, 1977, Miller, 1980)

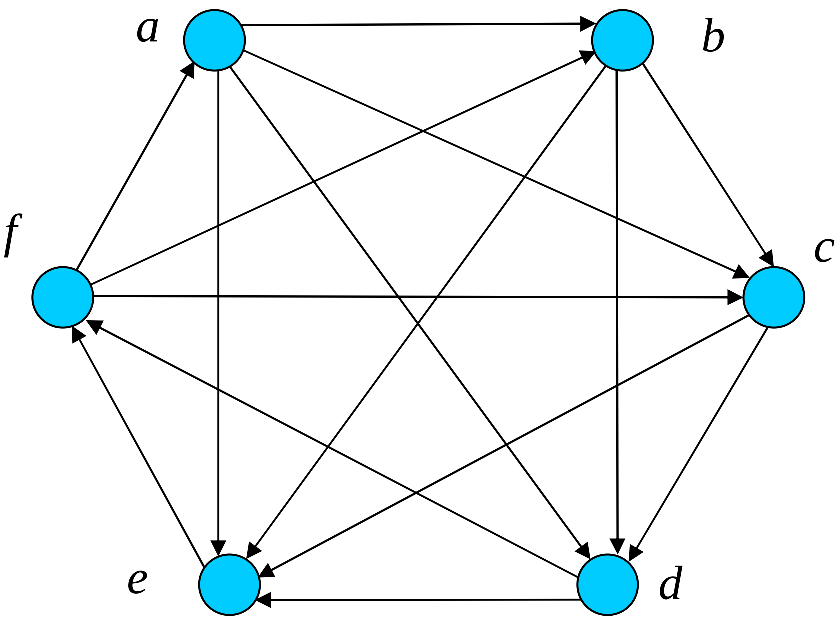
- Ex.

  b and c are covered by a;

  e is covered by d;

  a, d and f are uncovered:

  $$UC(T) = \{a, d, f\}.$$



- $UC^2(T) = UC^3(T) = \ldots = \{a, d, f\}$.

Markovian solution *MS* (Levchenkov, 1992, Laslier, 1993)

- A *Markov chain* is defined on X from $T = (X, A)$ with the following transition probabilities:

$$p(x \to y) = \begin{cases} 0 \text{ if } (x, y) \in A \\ 1/(n-1) \text{ if } (y, x) \in A \\ s(x)/(n-1) \text{ if } x = y \end{cases}$$

where $s(x)$ denotes the Copeland score (= the outdegree) of x. This defines a stochastic matrix $P = (p(x \to y))$.

- Then we start from any vertex and we randomly go from the current vertex to another one with a probability given by P. After k steps, the probability distribution $\pi_k = (\pi_{x,k})_{x \in X}$ is given by

$$\pi_k = \pi_{k-1}.P = \pi_0.P^k,$$

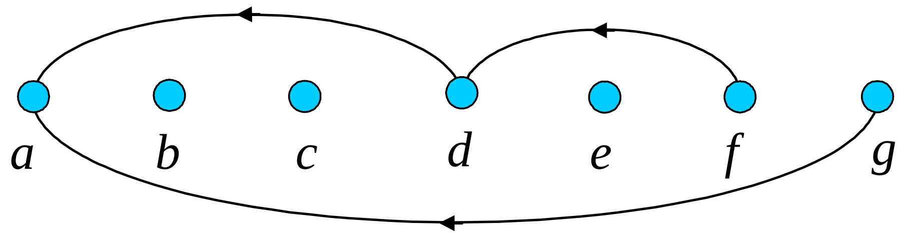
where π_0 denotes the initial probability distribution.

TELECOM PARIS
école nationale supérieure des télécommunications

- The Markovian theory shows that π_k admits a limit $\pi^* = (\pi_x^*)_{x \in X}$ when k tends to infinity with $\pi^* = \pi^* P$.
  - ⇨ π_x^* can be interpreted as the strength of x.
  - ⇨ The Markovian solution consists in sorting the candidates according to their strengths: the winners are the candidates x maximizing π_x^*.
- Ex.

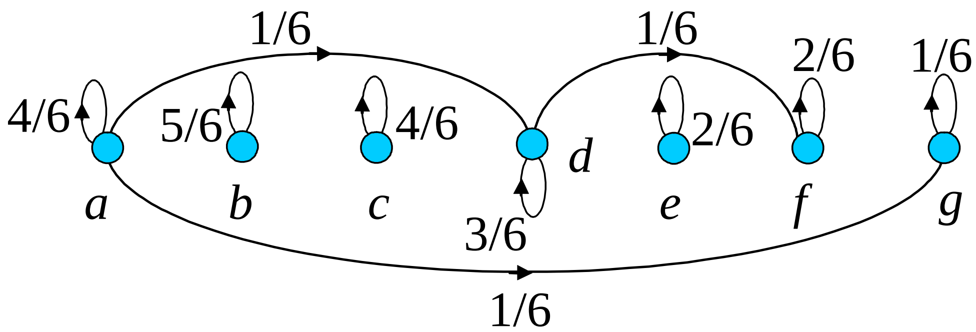
tournament

(missing arcs: $\rightarrow$)



Markovian chain

(missing arcs: $\leftarrow$)



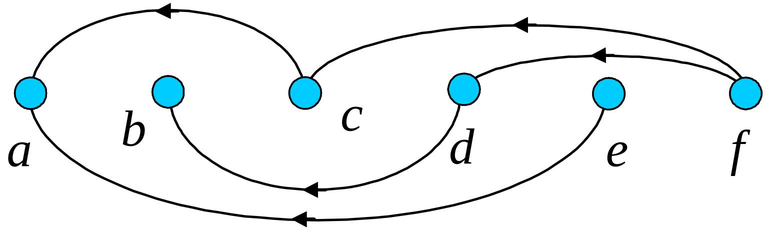
- ⇨ $\pi^* = (0.561, 0.738, 0.246, 0.241, 0.050, 0.009, 0.112)$.
- ⇨ b is the winner ($b > a > c > d > g > e > f$).

TELECOM PARIS
école nationale supérieure des télécommunications

Minimal covering set *MC* (Dutta, 1988)

- Let $Y \subset X$. For $y \in Y$ and $x \in X - Y$, we say that *y covers x in* $Y \cup \{x\}$ if y covers x in $T_{|Y \cup \{x\}}$.
- Y is a *covering set of T* if, for any $x \in X - Y$, x is covered in $Y \cup \{x\}$.
  Ex: UC and UC^∞ are covering sets.
- The *minimal covering set MC(T)* of T is the smallest (w.r.t. inclusion) covering set of T: $\forall\, x \in X - MC(T)$, x is covered in $MC(T) \cup \{x\}$ and $\forall\, Y \subset MC(T)$, $\exists\, x \in X - Y$ which is not covered in $Y \cup \{x\}$.
- Ex.

  (missing arcs: $\rightarrow$)



$MC(T) = \{a, b, c\}$

$UC = UC^\infty = \{a, b, c, d, e, f\}$.

TELECOM PARIS
école nationale supérieure des télécommunications

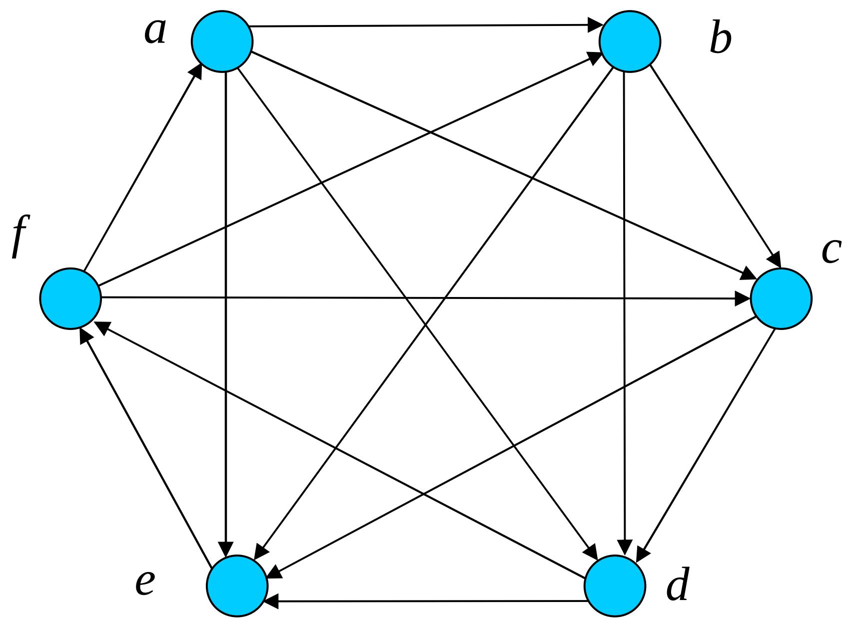
Bank's solution *B*: maximal transitive subtournament (Banks, 1985)

- $B(T) = \{x \in X$ such that there exists a maximal (w.r.t. inclusion) transitive subtournament, i.e. a linear order, of T with x as its absolute winner$\}$.

- Ex.

  $B(T) = \{a, d, f\}$

  a because of $a > b > c > d$
  d because of $d > e > f$
  f because of $f > a > b > c$.



Slater's solution *Sl*: linear orders at minimum distance (Slater, 1961)

- For any linear order O, let $d(O, T)$ be the symmetric difference distance between O and T: $d(O, T) = |\{(x, y) \in X^2 \text{ with } (x, y) \in O \text{ and } (y, x) \in T\}|$.

  Rk: $d(O, T)$ measures the number of disagreements between O and T.

- A *Slater order of T* is a linear order O^* minimizing d. This minimum distance is called the *Slater index i(T) of T*.

- A *Slater winner of T* is the absolute winner of any Slater order of T.

TELECOM PARIS
école nationale supérieure des télécommunications

Slater's solution *Sl*: linear orders at minimum distance (Slater, 1961)

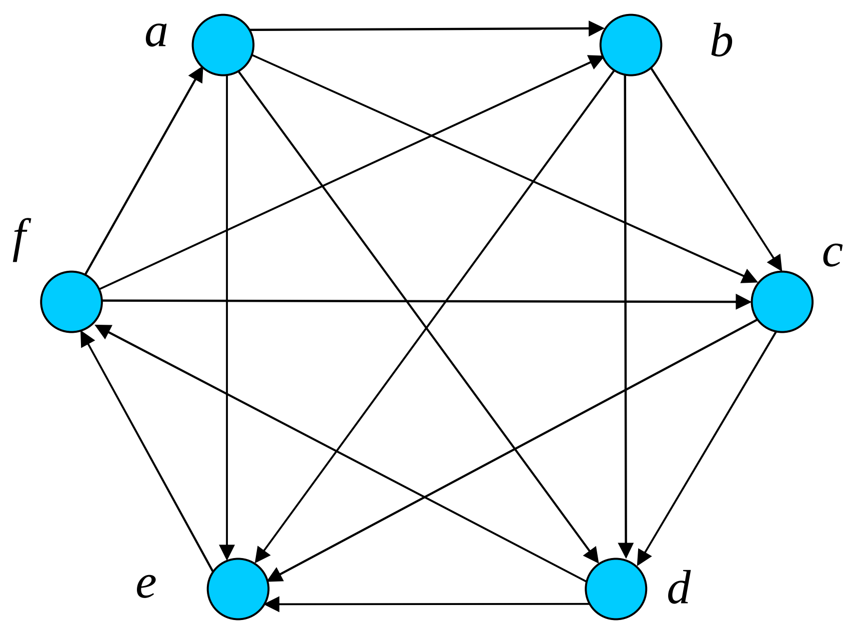
- Ex.

1 Slater order:

$f > a > b > d > c > e$,

$i(T) = 2$,

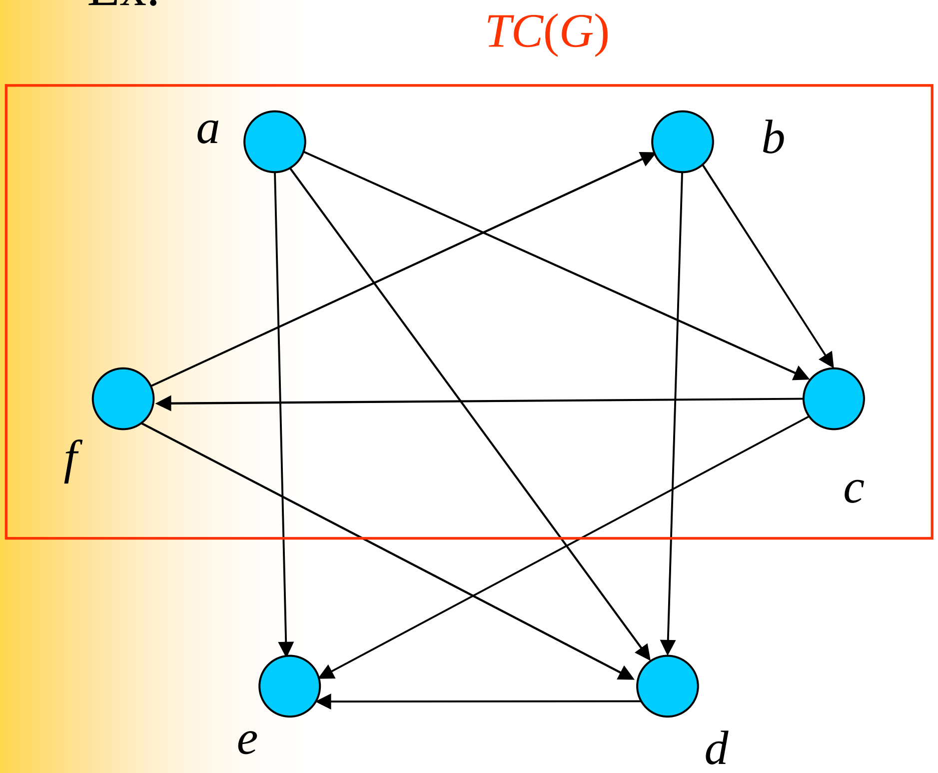
$Sl(T) = \{f\}$.



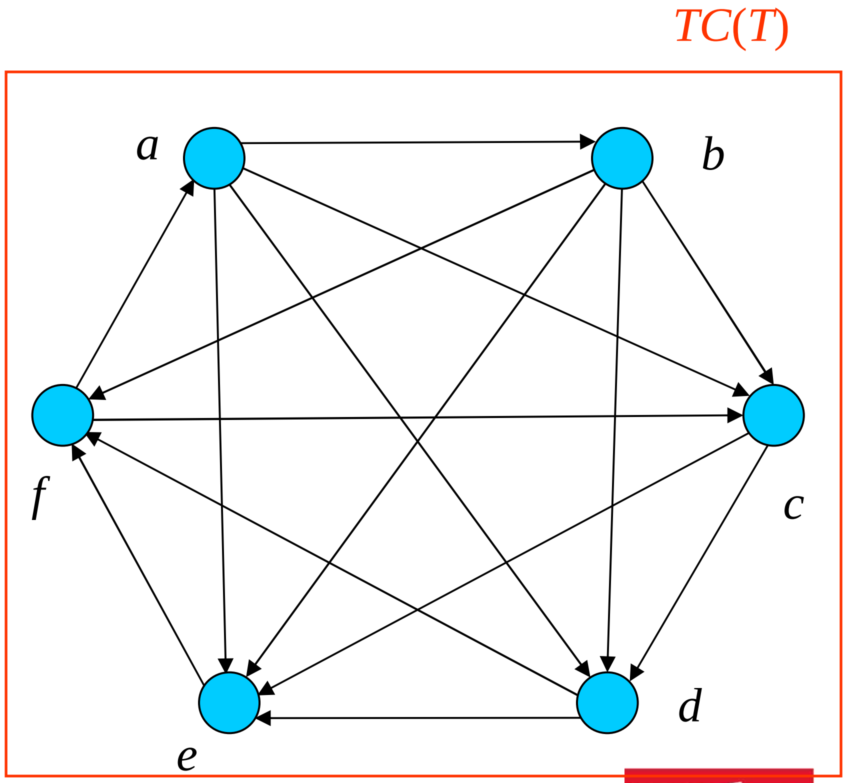
Top cycle *TC*

- Let $G = (X, A)$ be a directed graph. The *top cycle* $TC(G)$ *of* G is the union of the strongly connected components of G with no incoming arcs.

Ex.



G



T

Tournament equilibrium set *TEQ* (Schwartz, 1990)

- For any tournament solution S and any tournament $T = (X, A)$, define the (asymmetric) *contestation relation* $D(S, T)$ by:

$$x\, D(S, T)\, y \Leftrightarrow x \in S(T_{|N^-\{y\}}),$$

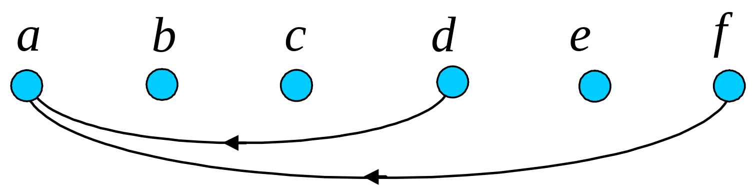
where $N^-\{y\}$ denotes the set of predecessors (in-neighbours) of y.

- We define a *contestation graph* $D_S(T)$: $D_S(T) = (X, D(S, T))$ and a new tournament solution S^* by: $S^*(T) = TC(D_S(T))$.

- TEQ is the only tournament solution S with $S^* = S$:

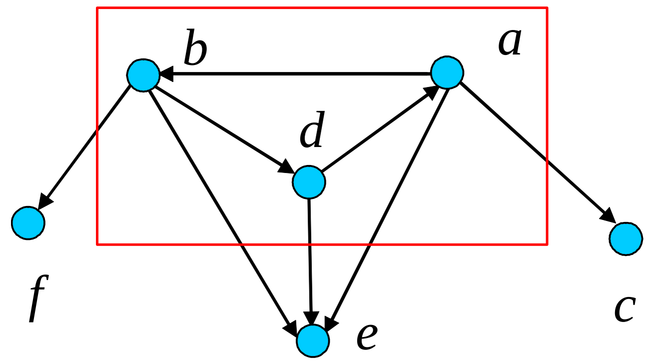
$$TEQ(T) = TC(D_{TEQ}(T)).$$

- Ex.



vertex x	a	b	c	d	e	f
$N^{-}\{x\}$	$\{d, f\}$	$\{a\}$	$\{a, b\}$	$\{b, c\}$	$\{a, b, c, d\}$	$\{b, c, d, e\}$
$TEQ(T_{\mid N^{-}\{x\}})$	$\{d\}$	$\{a\}$	$\{a\}$	$\{b\}$	$\{a, b, d\}$	$\{b\}$

$D_{TEQ}(T)$:



⇨ $TEQ(T) = \{a, b, d\}$.

Complexity results

- Polynomial methods:
  - ⇨ Copeland method and Zermelo method, in O(n^2).
  - ⇨ Uncovered set *UC*, in O(n^2) for checking that a player is a winner, in O($n^{2.38}$) to compute all the winners (multiplication of two $n \times n$ matrices).
  - ⇨ $\forall\, k \geq 1$, UC^k (the complexity depends on k).
  - ⇨ Markovian solution *MS*, in O($n^{2.38}$) (resolution of a linear system).
  - ⇨ Minimum covering set *MC* (as a linear programming problem; F. Brandt and F. Fischer, 2008).
  - ⇨ the computation of a Banks winner, in O(n^2) (O. H., 2004).

TELECOM PARIS
école nationale supérieure des télécommunications

Complexity results

- NP-hard methods:
  - ⇨ checking that a given vertex is a Banks winner (G. Woeginger, 2003)
  - ⇨ the computation of all the Banks winners
  - ⇨ Slater method *Sl* (N. Alon, 2006; P. Charbit, S. Thomassé, A. Yeo, 2007; V. Conitzer, 2006; O. H., 2010)
  - ⇨ *TEQ* (F. Brandt, F. Fischer, P. Harrenstein, M. Mair, 2010)

Links between these tournament solutions

	UC	$TC(UC)$	UC^{∞}	C, Z	Sl	B	MC
UC							
$TC(UC)$	$\subset$						
UC^{∞}	$\subset$	$\subset$					
C, Z	$\subset$	$\varnothing$ 13	$\varnothing$ 9				
Sl	$\subset$	$\subset$	$\varnothing$ 8	$\varnothing$ 6			
B	$\subset$	$\subset$	$\cap \neq \varnothing$	$\varnothing$ 13	$\varnothing$ 14		
MC	$\subset$	$\subset$	$\subset$	$\varnothing$ 9	$\varnothing$ 8	$\cap \neq \varnothing$	
TEQ	$\subset$	$\subset$	$\subset$	$\varnothing$ 9	$\varnothing$ 8	$\subset$	$\subset$

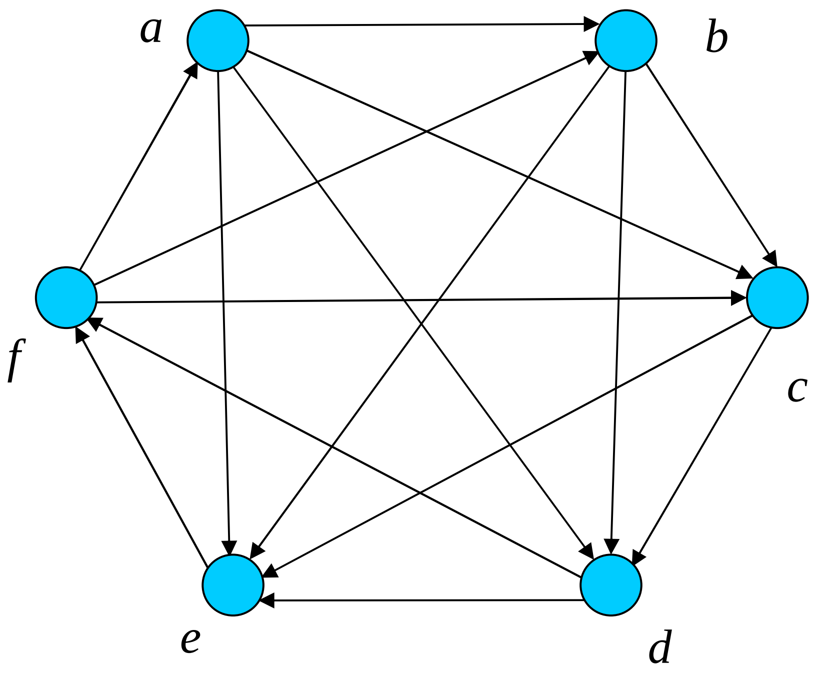
$S \subset S'$: $\forall\ T$, $S(T) \subseteq S'(T)$ and $\exists\ T$ s.t. $S(T) \neq S'(T)$

$S \varnothing^{k} S'$: $\forall\ n \geq k$, $\exists\ T$ with n vertices s.t. $S(T) \cap S'(T) = \varnothing$

$S \cap S' \neq \varnothing$: $\forall\ T$, $S(T) \cap S'(T) \neq \varnothing$

TELECOM PARIS
école nationale supérieure des télécommunications

An illustration



- $C(T) = Z(T) = \{a\}$
- $UC(T) = \{a, d, f\}$
- $UC^{\infty}(T) = \{a, d, f\}$
- $MC(T) = \{a, d, f\}$
- $MS(T) = \{a\}$

  (1 order : $a > f > d > b > c > e$)
- $B(T) = \{a, d, f\}$
- $Sl(T) = \{f\}$

  (1 Slater order : $f > a > b > d > c > e$)
- $TEQ(T) = \{a, d, f\}$

Short bibliography

- P. Fishburn (1977) Condorcet social choice functions, *SIAM Journal of Applied Mathematics*, 33, 469-489.
- O. Hudry (2009) A survey on the complexity of tournament solutions, *Mathematical Social Sciences*, 57 (3), 292-303.
- O. Hudry (2009) Complexity of voting procedures, *Encyclopedia of Complexity and Systems Science*, R. Meyers (ed.), 9942-9965.
- J.-F. Laslier (1997) *Tournament Solutions and Majority Voting*, Springer, Berlin, Heidelberg, New York.
- J. W. Moon (1968) *Topics on tournaments*, Holt, Rinehart and Winston, New York.
- H. Moulin (1986) Choosing from a tournament, *Social Choice and Welfare* 3: 272-291.

Thank you for your attention!

TELECOM PARIS
école nationale supérieure des télécommunications