Topological arguments and Kolmogorov complexity

Andrei Romashenko (joint work with Alexander Shen)
LIRMM, CNRS & UM2, Montpellier; on leave from ИППИ РАН, Москва

Supported by ANR NAFIT grant
Apologies
Apologies

- off-topic
Apologies

- off-topic message: not only general topology can be useful in computer science
Apologies

- off-topic message: not only general topology can be useful in computer science
- no blackboard
Conditional complexity as distance

$I(C_{x|y})$, conditional complexity of $x$ given $y$, minimal length of a program that maps $y$ to $x$. $I(C_{x|y})$ depends on the programming language, is minimal up to $O(1)$ for some "optimal" languages; one of them is fixed. $I(C_{x|y})$ measures "how far is $x$ from $y$" in a sense, but not symmetric task: given string $x$ and number $n$, find $y$ such that $C_{x|y} = n + O(1)$ and $C_{y|x} = n + O(1)$.

$I(C_{x})$ should be at least $n$. 
Conditional complexity as distance

- $C(x|y)$, conditional complexity of $x$ given $y$, minimal length of a program that maps $y$ to $x$
Conditional complexity as distance

- \( C(x|y) \), conditional complexity of \( x \) given \( y \), minimal length of a program that maps \( y \) to \( x \)
- depends on the programming language, is minimal up to \( O(1) \) for some “optimal” languages; one of them is fixed
Conditional complexity as distance

- $C(x|y)$, conditional complexity of $x$ given $y$, minimal length of a program that maps $y$ to $x$
- depends on the programming language, is minimal up to $O(1)$ for some “optimal” languages; one of them is fixed
- $C(x|y)$ measures “how far is $x$ from $y$” in a sense, but not symmetric
Conditional complexity as distance

- $C(x|y)$, conditional complexity of $x$ given $y$, minimal length of a program that maps $y$ to $x$
- depends on the programming language, is minimal up to $O(1)$ for some “optimal” languages; one of them is fixed
- $C(x|y)$ measures “how far is $x$ from $y$” in a sense, but not symmetric
- task: given string $x$ and number $n$, find $y$ such that $C(x|y) = n + O(1)$ and $C(y|x) = n + O(1)$
Conditional complexity as distance

- $C(x|y)$, conditional complexity of $x$ given $y$, minimal length of a program that maps $y$ to $x$
- depends on the programming language, is minimal up to $O(1)$ for some “optimal” languages; one of them is fixed
- $C(x|y)$ measures “how far is $x$ from $y$” in a sense, but not symmetric
- task: given string $x$ and number $n$, find $y$ such that $C(x|y) = n + O(1)$ and $C(y|x) = n + O(1)$
- not always possible: $C(x)$ should be at least $n$
M. Vyugin theorem and its extension

Theorem: if \( C(x) > 2n \), there exists \( y \) such that:

\[
C(x_jy) = n + O(1) \quad \text{and} \quad C(y_jx) = n + O(1).
\]

The proof uses a game argument. In fact, \( C(x) > n + O(\log n) \) is enough. But for completely different reasons: a simple topological fact: if a continuous mapping of a circle \( S_1 \) to \( \mathbb{R}^2 \) turns around some point, then any its continuous extension to a mapping of a disk \( D^2 \) covers it.

Strangely, for \( C(x) \gg n \) the argument does not work (only for \( C(x) = \text{poly}(n) \)). So \( C(x) = n + O(\log n) \) is enough, but two essentially different arguments are needed at both ends.
Theorem: if $C(x) > 2n$, there exists $y$ such that $C(x|y) = n + O(1)$ and $C(y|x) = n + O(1)$.
M. Vyugin theorem and its extension

- Theorem: if $C(x) > 2n$, there exists $y$ such that $C(x|y) = n + O(1)$ and $C(y|x) = n + O(1)$.
- proof uses a game argument
M. Vyugin theorem and its extension

- Theorem: if $C(x) > 2n$, there exists $y$ such that $C(x|y) = n + O(1)$ and $C(y|x) = n + O(1)$.
- proof uses a game argument
- in fact $C(x) > n + O(\log n)$ is enough
M. Vyugin theorem and its extension

- **Theorem:** if $C(x) > 2n$, there exists $y$ such that $C(x|y) = n + O(1)$ and $C(y|x) = n + O(1)$.
- proof uses a game argument
- in fact $C(x) > n + O(\log n)$ is enough
- but for completely different reasons
Theorem: if \( C(x) > 2n \), there exists \( y \) such that
\[ C(x|y) = n + O(1) \]
and
\[ C(y|x) = n + O(1). \]

Proof uses a game argument

\( C(x) > n + O(\log n) \) is enough

But for completely different reasons

Simple topological fact: if a continuous mapping of a circle \( S^1 \) to \( \mathbb{R}^2 \) turns around some point \( O \), then any its continuous extension to a mapping of a disk \( D^2 \) covers \( O \).
M. Vyugin theorem and its extension

- **Theorem:** if $C(x) > 2n$, there exists $y$ such that $C(x|y) = n + O(1)$ and $C(y|x) = n + O(1)$.
- **proof uses a game argument**
- in fact $C(x) > n + O(\log n)$ is enough
- but for completely different reasons
- **simple topological fact:** if a continuous mapping of a circle $S^1$ to $\mathbb{R}^2$ turns around some point $O$, then any its continuous extension to a mapping of a disk $D^2$ covers $O$
- strangely, for $C(x) \gg n$ this argument does not work (only for $C(x) \leq \text{poly}(n)$)
Theorem: if $C(x) > 2n$, there exists $y$ such that $C(x|y) = n + O(1)$ and $C(y|x) = n + O(1)$.

Proof uses a game argument.

In fact, $C(x) > n + O(\log n)$ is enough.

But for completely different reasons.

Simple topological fact: if a continuous mapping of a circle $S^1$ to $\mathbb{R}^2$ turns around some point $O$, then any its continuous extension to a mapping of a disk $D^2$ covers $O$.

Strangely, for $C(x) \gg n$ this argument does not work (only for $C(x) \leq \text{poly}(n)$).

So $C(x) \geq n + O(\log n)$ is enough, but two essentially different arguments are needed at both ends.
Why topology can be useful

I simple example: imagine we want \( C(x \mid y) = n \) and know that \( C(x) = n \).

Let \( y \) be \( x \), then \( C(x \mid y) = O(1) \).

Let us remove bits in \( y \) one by one (e.g., from right to left). \( C(x \mid y) \) then changes but gradually: \( C(x \mid y_0) \) and \( C(x \mid y_1) \) are \( C(x \mid y) + O(1) \).

At the end \( y \) is empty, and \( C(x \mid y) = C(x) = n \).

Discrete intermediate value theorem guarantees that \( C(x \mid y) = n + O(1) \) for some \( y \) on the way.
Why topology can be useful

- simple example: imagine we want \( C(x|y) = n \) and know that \( C(x) \geq n \).
Why topology can be useful

- simple example: imagine we want $C(x|y) = n$ and know that $C(x) \geq n$.
- let $y$ be $x$, then $C(x|y) = O(1)$
simple example: imagine we want $C(x|y) = n$ and know that $C(x) \geq n$.

let $y$ be $x$, then $C(x|y) = O(1)$

let us remove bits in $y$ one by one (e.g., from right to left)
Why topology can be useful

- simple example: imagine we want $C(x|y) = n$ and know that $C(x) \geq n$.
- let $y$ be $x$, then $C(x|y) = O(1)$
- let us remove bits in $y$ one by one (e.g., from right to left)
- $C(x|y)$ then changes but gradually: $C(x|y0)$ and $C(x|y1)$ are $C(x|y) + O(1)$
Why topology can be useful

- simple example: imagine we want $C(x|y) = n$ and know that $C(x) \geq n$.
- let $y$ be $x$, then $C(x|y) = O(1)$
- let us remove bits in $y$ one by one (e.g., from right to left)
- $C(x|y)$ then changes but gradually: $C(x|y0)$ and $C(x|y1)$ are $C(x|y) + O(1)$
- at the end $y$ is empty, and $C(x|y) = C(x) \geq n$
Why topology can be useful

- simple example: imagine we want $C(x|y) = n$ and know that $C(x) \geq n$.
- let $y$ be $x$, then $C(x|y) = O(1)$
- let us remove bits in $y$ one by one (e.g., from right to left)
- $C(x|y)$ then changes but gradually: $C(x|y0)$ and $C(x|y1)$ are $C(x|y) + O(1)$
- at the end $y$ is empty, and $C(x|y) = C(x) \geq n$
- discrete intermediate value theorem guarantees that $C(x|y) = n + O(1)$ for some $y$ on the way
\(O(\log n)\) precision is easy
\( O(\log n) \) precision is easy

- to get \( C(y|x) = n \) we need to put some \( n \) bits of new information (that is not in \( x \)) into \( y \)
$O(\log n)$ precision is easy

- to get $C(y|x) = n$ we need to put some $n$ bits of new information (that is not in $x$) into $y$
- to get $C(x|y) = n$ we need to put in $y$ all the information about $x$ except for $n$ bits
$O(\log n)$ precision is easy

- to get $C(y|x) = n$ we need to put some $n$ bits of new information (that is not in $x$) into $y$
- to get $C(x|y) = n$ we need to put in $y$ all the information about $x$ except for $n$ bits
- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
$O(\log n)$ precision is easy

- to get $C(y|x) = n$ we need to put some $n$ bits of new information (that is not in $x$) into $y$
- to get $C(x|y) = n$ we need to put in $y$ all the information about $x$ except for $n$ bits
- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
- $p$ is incompressible
$O(\log n)$ precision is easy

- to get $C(y|x) = n$ we need to put some $n$ bits of new information (that is not in $x$) into $y$
- to get $C(x|y) = n$ we need to put in $y$ all the information about $x$ except for $n$ bits
- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
- $p$ is incompressible
- let $y$ be $p$ without $n$ bits
$O(\log n)$ precision is easy

- to get $C(y|x) = n$ we need to put some $n$ bits of new information (that is not in $x$) into $y$
- to get $C(x|y) = n$ we need to put in $y$ all the information about $x$ except for $n$ bits
- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
- $p$ is incompressible
- let $y$ be $p$ without $n$ bits
- plus some random $n$ bits (independent from $p$)
$O(\log n)$ precision is easy

- to get $C(y|x) = n$ we need to put some $n$ bits of new information (that is not in $x$) into $y$
- to get $C(x|y) = n$ we need to put in $y$ all the information about $x$ except for $n$ bits
- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
- $p$ is incompressible
- let $y$ be $p$ without $n$ bits
- plus some random $n$ bits (independent from $p$)
- then both $C(x|y)$ and $C(y|x)$ are $n + O(\log n)$
**$O(\log n)$ precision is easy**

- to get $C(y|x) = n$ we need to put some $n$ bits of new information (that is not in $x$) into $y$
- to get $C(x|y) = n$ we need to put in $y$ all the information about $x$ except for $n$ bits
- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
- $p$ is incompressible
- let $y$ be $p$ without $n$ bits
- plus some random $n$ bits (independent from $p$)
- then both $C(x|y)$ and $C(y|x)$ are $n + O(\log n)$
- $O(1)$ cannot be obtained in this way (since all the arguments about random and independent bits work with $O(\log n)$ precision only)
Putting pieces together
Putting pieces together

- let \( p \) be the shortest program for \( x \), so \( |p| = C(x) \geq n \)
Putting pieces together

- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
- let $q$ be a random (incompressible) string of length $2n$ when $p$ is known (independent from $p$)
Putting pieces together

- let \( p \) be the shortest program for \( x \), so \( |p| = C(x) \geq n \)
- let \( q \) be a random (incompressible) string of length \( 2n \) when \( p \) is known (independent from \( p \))
- for every \( k \in [0, C(x)] \) and every \( l \in [0, 2n] \) consider
  \[
y_{k,l} = (k\text{-bit prefix of } p, l\text{-bit prefix of } q)
  \]
Putting pieces together

- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
- let $q$ be a random (incompressible) string of length $2n$ when $p$ is known (independent from $p$)
- for every $k \in [0, C(x)]$ and every $l \in [0, 2n]$ consider
  \[ y_{k,l} = (k\text{-bit prefix of } p, l\text{-bit prefix of } q) \]
- mapping $(k, l) \mapsto (C(x|y_{k,l}), C(y_{k,l}|x))$
Putting pieces together

- let $p$ be the shortest program for $x$, so $|p| = C(x) \geq n$
- let $q$ be a random (incompressible) string of length $2n$ when $p$ is known (independent from $p$)
- for every $k \in [0, C(x)]$ and every $l \in [0, 2n]$ consider

$$y_{k,l} = (k\text{-bit prefix of } p, l\text{-bit prefix of } q)$$

- mapping $(k, l) \mapsto (C(x|y_{k,l}), C(y_{k,l}|x))$
Topological details

I mapping is defined on a grid (rectangle) and maps neighbor points to a points at distance ‘Lipschitz continuity’ covers \((n; n)\) with \(O(1)\) precision. I reduction to continuous version: interpolation on triangles (linear). I preimage may be not in the grid, but neighbor grid point gives \(O(1)\) precision. I Alternative: repeat the proof for discrete case.
Topological details

- mapping is defined on a grid (rectangle)
Topological details

- mapping is defined on a grid (rectangle)
- and maps neighbor points to a points at $O(1)$ distance
Topological details

- mapping is defined on a grid (rectangle)
- and maps neighbor points to a points at $O(1)$ distance
- “Lipschitz continuity”
Topological details

- mapping is defined on a grid (rectangle)
- and maps neighbor points to a points at $O(1)$ distance
- “Lipschitz continuity”
- covers $(n, n)$ with $O(1)$ precision
Topological details

- mapping is defined on a grid (rectangle)
- and maps neighbor points to a points at $O(1)$ distance
- “Lipschitz continuity”
- covers $(n, n)$ with $O(1)$ precision
- reduction to continuous version: interpolation on triangles (linear)
Topological details

- mapping is defined on a grid (rectangle)
- and maps neighbor points to a points at $O(1)$ distance
- “Lipschitz continuity”
- covers $(n, n)$ with $O(1)$ precision
- reduction to continuous version: interpolation on triangles (linear)
- preimage may be not in the grid, but neighbor grid point gives $O(1)$-precision
Topological details

- mapping is defined on a grid (rectangle)
- and maps neighbor points to a points at $O(1)$ distance
- “Lipschitz continuity”
- covers $(n, n)$ with $O(1)$ precision
- reduction to continuous version: interpolation on triangles (linear)
- preimage may be not in the grid, but neighbor grid point gives $O(1)$-precision
- Alternative: repeat the proof for discrete case
Comments

I why we need $C(x)$ be polynomial? if $C(x)$ is very large, the value of $k$ may contain a lot of information about $z$.

I it is not necessary (unlike for original Vyugin argument) to have the same targets for $C(x_jy)$ and $C(y_jx)$.

I other applications of the same type of argument: for every $x, y$ that are almost independent ($I(x:y)$ is small compared to $C(x)$ and $C(y)$) one can find $z$ such that $C(x_\mid z) = C(x)/2 + O(1)$ and $C(y_\mid z) = C(y)/2 + O(1)$.

I similar statement for halving complexity of three or more strings by adding a condition: under the assumption of independence (can be weakened but not eliminated).

I an open problem in the general case (problematic case: $x$ and $y$ are very close to each other, but not completely identical).
Comments

- why we need $C(x)$ be polynomial? if $C(x)$ is very large, the value of $k$ may contain a lot of information about $z$
Comments

- why we need $C(x)$ be polynomial? if $C(x)$ is very large, the value of $k$ may contain a lot of information about $z$
- it is not necessary (unlike for original Vyugin argument) to have the same targets for $C(x|y)$ and $C(y|x)$
why we need $C(x)$ be polynomial? if $C(x)$ is very large, the value of $k$ may contain a lot of information about $z$

it is not necessary (unlike for original Vyugin argument) to have the same targets for $C(x|y)$ and $C(y|x)$

other applications of the same type of argument: for every $x$, $y$ that are almost independent ($I(x : y)$ is small compared to $C(x)$ and $C(y)$) one can find $z$ such that $C(x|z) = C(x)/2 + O(1)$ and $C(y|z) = C(y)/2 + O(1)$
Comments

- why we need \( C(x) \) be polynomial? if \( C(x) \) is very large, the value of \( k \) may contain a lot of information about \( z \)
- it is not necessary (unlike for original Vyugin argument) to have the same targets for \( C(x|y) \) and \( C(y|x) \)
- other applications of the same type of argument: for every \( x, y \) that are almost independent (\( I(x : y) \) is small compared to \( C(x) \) and \( C(y) \)) one can find \( z \) such that \( C(x|z) = C(x) / 2 + O(1) \) and \( C(y|z) = C(y) / 2 + O(1) \)
- similar statement for halving complexity of three or more strings by adding a condition:
why we need \( C(x) \) be polynomial? if \( C(x) \) is very large, the value of \( k \) may contain a lot of information about \( z \)

it is not necessary (unlike for original Vyugin argument) to have the same targets for \( C(x|y) \) and \( C(y|x) \)

other applications of the same type of argument: for every \( x, y \) that are almost independent (\( I(x : y) \) is small compared to \( C(x) \) and \( C(y) \)) one can find \( z \) such that \( C(x|z) = C(x)/2 + O(1) \) and \( C(y|z) = C(y)/2 + O(1) \)

similar statement for halving complexity of three or more strings by adding a condition:

under the assumption of independence (can be weakened but not eliminated).
Comments

- why we need $C(x)$ be polynomial? if $C(x)$ is very large, the value of $k$ may contain a lot of information about $z$
- it is not necessary (unlike for original Vyugin argument) to have the same targets for $C(x|y)$ and $C(y|x)$
- other applications of the same type of argument: for every $x, y$ that are almost independent ($I(x : y)$ is small compared to $C(x)$ and $C(y)$) one can find $z$ such that $C(x|z) = C(x)/2 + O(1)$ and $C(y|z) = C(y)/2 + O(1)$
- similar statement for halving complexity of three or more strings by adding a condition:
- under the assumption of independence (can be weakened but not eliminated).
- an open problem in the general case (problematic case: $x$ and $y$ are very close to each other, but not completely identical)
Thanks!
Original game argument

I \[ C(x) > 3^n \], there exists \( y \) such that \( C(x) \mid y \) and \( C(y) \mid x \) are \( n + O(1) \).

We replaced \( 2^n \) by \( 3^n \) to simplify explanations (and in any case this is already covered).

We present some game

Then show why winning this game is enough

And finally show how to win the game
Original game argument

- If $C(x) > 3n$, there exists $y$ such that $C(x|y)$ and $C(y|x)$ are $n + O(1)$.
Original game argument

- If $C(x) > 3n$, there exists $y$ such that $C(x|y)$ and $C(y|x)$ are $n + O(1)$
- We replaced $2n$ by $3n$ to simplify explanations (and in any case this is already covered)
Original game argument

- If $C(x) > 3n$, there exists $y$ such that $C(x|y)$ and $C(y|x)$ are $n + O(1)$
- We replaced $2n$ by $3n$ to simplify explanations (and in any case this is already covered)
- We present some game
Original game argument

- If $C(x) > 3n$, there exists $y$ such that $C(x|y)$ and $C(y|x)$ are $n + O(1)$
- we replaced $2n$ by $3n$ to simplify explanations (and in any case this is already covered)
- we present some game
- then show why winning this game is enough
Original game argument

- If $C(x) > 3n$, there exists $y$ such that $C(x|y)$ and $C(y|x)$ are $n + O(1)$
- we replaced $2n$ by $3n$ to simplify explanations (and in any case this is already covered)
- we present some game
- then show why winning this game is enough
- and finally show how to win the game
Dating agency and its task

I two countable sets $X$ and $Y$ the game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$.

I An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x; y)$ is dissolved.

I $y$ then becomes free; the agency may either find a new pair for $x$ from the dissolved pair (among free elements of $Y$ not tried with $x$ previously) or declare $x$ hopeless and do not try to find a pair for $x$ anymore (#free in $Y$ incremented).

I the refusals appear (and are processed by the agency) one at a time.

I each element can produce $< N$ refusals (parameter of the game), but no restrictions for #(being refused).

I agency obligations:

I $2N$ attempts for each element.

I $2N + 3$ hopeless elements; all others in $X$ are ultimately connected to some $y \in Y$ and this connection lasts forever.
Dating agency and its task

- two countable sets $X$ and $Y$
Dating agency and its task

- two countable sets $X$ and $Y$
- game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$. 

The game proceeds as follows:

- An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x, y)$ is dissolved.
- Then, the agency may either find a new pair for $x$ from the dissolved pair (among free elements of $Y$ not tried with $x$ previously) or declare $x$ hopeless and do not try to find a pair for $x$ anymore (if free in $Y$ incremented).
- Refusals appear (and are processed by the agency) one at a time.
- Each element can produce $\leq N$ refusals (parameter of the game), but no restrictions for #being refused.

Agency obligations:

- At most $2N$ attempts for each element.
- At most $2N - 3$ hopeless elements; all others in $X$ are ultimately connected to some $y \in Y$, and this connection lasts forever.
Dating agency and its task

- two countable sets $X$ and $Y$
- game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$.
- An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x, y)$ is dissolved
Dating agency and its task

- two countable sets $X$ and $Y$
- game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$.
- An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x, y)$ is dissolved
- $y$ then becomes free; the agency may either
Dating agency and its task

- two countable sets $X$ and $Y$
- game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$.
- An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x, y)$ is dissolved
- $y$ then becomes free; the agency may either
  - find a new pair for $x$ from the dissolved pair (among free elements of $Y$ not tried with $x$ previously) or
Dating agency and its task

- two countable sets $X$ and $Y$
- game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$.
- An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x, y)$ is dissolved
- $y$ then becomes free; the agency may either
  - find a new pair for $x$ from the dissolved pair (among free elements of $Y$ not tried with $x$ previously) or
  - declare $x$ hopeless and do not try to find a pair for $x$ anymore ($\#$free in $Y$ incremented)
Dating agency and its task

- two countable sets \(X\) and \(Y\)
- game starts with a perfect matching, i.e., one to one correspondence between \(X\) and \(Y\).
- An element of \(X\) or \(Y\) can refuse the current partner, then the current relationship \((x, y)\) is dissolved
- \(y\) then becomes free; the agency may either
  - find a new pair for \(x\) from the dissolved pair (among free elements of \(Y\) not tried with \(x\) previously) or
  - declare \(x\) hopeless and do not try to find a pair for \(x\) anymore (#free in \(Y\) incremented)
- the refusals appear (and are processed by the agency) one at a time
Dating agency and its task

- two countable sets $X$ and $Y$
- game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$.
- An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x,y)$ is dissolved
- $y$ then becomes free; the agency may either
  - find a new pair for $x$ from the dissolved pair (among free elements of $Y$ not tried with $x$ previously) or
  - declare $x$ hopeless and do not try to find a pair for $x$ anymore (#free in $Y$ incremented)
- the refusals appear (and are processed by the agency) one at a time
- each element can produce $< N$ refusals (parameter of the game), but no restrictions for #(being refused)
Dating agency and its task

- two countable sets $X$ and $Y$
- game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$.
- An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x, y)$ is dissolved
- $y$ then becomes free; the agency may either
  - find a new pair for $x$ from the dissolved pair (among free elements of $Y$ not tried with $x$ previously) or
  - declare $x$ hopeless and do not try to find a pair for $x$ anymore (#free in $Y$ incremented)
- the refusals appear (and are processed by the agency) one at a time
- each element can produce $< N$ refusals (parameter of the game), but no restrictions for #being refused
- agency obligations:
Dating agency and its task

- two countable sets \( X \) and \( Y \)
- game starts with a perfect matching, i.e., one to one correspondence between \( X \) and \( Y \).
- An element of \( X \) or \( Y \) can refuse the current partner, then the current relationship \((x, y)\) is dissolved
- \( y \) then becomes free; the agency may either
  - find a new pair for \( x \) from the dissolved pair (among free elements of \( Y \) not tried with \( x \) previously) or
  - declare \( x \) hopeless and do not try to find a pair for \( x \) anymore (#free in \( Y \) incremented)
- the refusals appear (and are processed by the agency) one at a time
- each element can produce \(< N\) refusals (parameter of the game), but no restrictions for #(being refused)
- agency obligations:
  - \( \leq 2N \) attempts for each element
Dating agency and its task

- two countable sets $X$ and $Y$
- game starts with a perfect matching, i.e., one to one correspondence between $X$ and $Y$.
- An element of $X$ or $Y$ can refuse the current partner, then the current relationship $(x, y)$ is dissolved
- $y$ then becomes free; the agency may either
  - find a new pair for $x$ from the dissolved pair (among free elements of $Y$ not tried with $x$ previously) or
  - declare $x$ hopeless and do not try to find a pair for $x$ anymore (#free in $Y$ incremented)
- the refusals appear (and are processed by the agency) one at a time
- each element can produce $< N$ refusals (parameter of the game), but no restrictions for $(\text{being refused})$
- agency obligations:
  - $\leq 2N$ attempts for each element
  - $\leq 2N^3$ hopeless elements; all others in $X$ are ultimately connected to some $y \in Y$ and this connection lasts forever
Why computable winning strategy is enough
Why computable winning strategy is enough

- $X = Y = \mathbb{B}^*$
Why computable winning strategy is enough

- $X = Y = \mathbb{B}^*$
- initial matching: identity $(x, x)$
Why computable winning strategy is enough

- $X = Y = \mathbb{B}^*$
- initial matching: identity $(x, x)$
- $u$ refuses $v$ if $C(v|u) < n$ (here $u$ may be in $X$ or in $Y$)
Why computable winning strategy is enough

- $X = Y = \mathbb{B}^*$
- initial matching: identity $(x, x)$
- $u$ refuses $v$ if $C(v|u) < n$ (here $u$ may be in $X$ or in $Y$)
- less than $N = 2^n$ refusals for each $u$
Why computable winning strategy is enough

- $X = Y = \mathbb{B}^*$
- initial matching: identity $(x, x)$
- $u$ refuses $v$ if $C(v|u) < n$ (here $u$ may be in $X$ or in $Y$)
- less than $N = 2^n$ refusals for each $u$
- computable behavior
Why computable winning strategy is enough

- \( X = Y = \mathbb{B}^* \)
- initial matching: identity \((x, x)\)
- \( u \) refuses \( v \) if \( C(v|u) < n \) (here \( u \) may be in \( X \) or in \( Y \))
- less than \( N = 2^n \) refusals for each \( u \)
- computable behavior
- agency produces \( O(N^3) = O(2^{3n}) \) hopeless elements of complexity \( 3n + O(1) \) (identified by \( 3n + O(1) \) bit ordinal number)
Why computable winning strategy is enough

- $X = Y = B^*$
- initial matching: identity $(x, x)$
- $u$ refuses $v$ if $C(v|u) < n$ (here $u$ may be in $X$ or in $Y$)
- less than $N = 2^n$ refusals for each $u$
- computable behavior
- agency produces $O(N^3) = O(2^{3n})$ hopeless elements of complexity $3n + O(1)$ (identified by $3n + O(1)$ bit ordinal number)
- for every $x$ that is not hopeless its final partner $y$ has $C(y|x)$ and $C(x|y)$ at most $n + O(1)$: determined by a ordinal number that is $O(N) = 2^{n+O(1)}$
Why computable winning strategy is enough

- $X = Y = \mathbb{B}^*$
- initial matching: identity $(x, x)$
- $u$ refuses $v$ if $C(v|u) < n$ (here $u$ may be in $X$ or in $Y$)
- less than $N = 2^n$ refusals for each $u$
- computable behavior
- agency produces $O(N^3) = O(2^{3n})$ hopeless elements of complexity $3n + O(1)$ (identified by $3n + O(1)$ bit ordinal number)
- for every $x$ that is not hopeless its final partner $y$ has $C(y|x)$ and $C(x|y)$ at most $n + O(1)$: determined by a ordinal number that is $O(N) = 2^{n+O(1)}$
- but both complexities are at least $n$, otherwise refused
How to win the game
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)

- the first is $< N$; the second a priori is unbounded, but also will be kept $< N$ due to agency strategy
How to win the game

- each element not currently matched keeps “experience” = (#refusals sent, #refusals received)
- the first is < $N$; the second a priori is unbounded, but also will be kept < $N$ due to agency strategy
- when $(x, y)$ is terminated, numbers updated
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)
- the first is < N; the second a priori is unbounded, but also will be kept < N due to agency strategy
- when (x, y) is terminated, numbers updated
- invariant: in all pairs people have matching experiences (#sent = #received for the other)
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)
- the first is $< N$; the second a priori is unbounded, but also will be kept $< N$ due to agency strategy
- when $(x, y)$ is terminated, numbers updated
- invariant: in all pairs people have matching experiences ($#sent = #received$ for the other)
- corollary: #refusals received $< N$
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)
- the first is $< N$; the second a priori is unbounded, but also will be kept $< N$ due to agency strategy
- when $(x, y)$ is terminated, numbers updated
- invariant: in all pairs people have matching experiences (#sent = #received for the other)
- corollary: #refusals received $< N$
- new partner for $x$ is found if possible (=there is $y \in Y$ with matching experience not tried earlier with $x$)
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)
- the first is $< N$; the second a priori is unbounded, but also will be kept $< N$ due to agency strategy
- when $(x, y)$ is terminated, numbers updated
- invariant: in all pairs people have matching experiences (#sent = #received for the other)
- corollary: #refusals received $< N$
- new partner for $x$ is found if possible (=there is $y \in Y$ with matching experience not tried earlier with $x$)
- otherwise $x$ is declared hopeless
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)
- the first is $< N$; the second a priori is unbounded, but also will be kept $< N$ due to agency strategy
- when $(x, y)$ is terminated, numbers updated
- invariant: in all pairs people have matching experiences (#sent = #received for the other)
- corollary: #refusals received $< N$
- new partner for $x$ is found if possible (=there is $y \in Y$ with matching experience not tried earlier with $x$)
- otherwise $x$ is declared hopeless
- invariant: for matching experiences the number of non-matched people in $X$ and $Y$ are the same
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)

- the first is $< N$; the second a priori is unbounded, but also will be kept $< N$ due to agency strategy

- when $(x, y)$ is terminated, numbers updated

- invariant: in all pairs people have matching experiences (#sent = #received for the other)

- corollary: #refusals received $< N$

- new partner for $x$ is found if possible (=there is $y \in Y$ with matching experience not tried earlier with $x$)

- otherwise $x$ is declared hopeless

- invariant: for matching experiences the number of non-matched people in $X$ and $Y$ are the same

- $\leq 2N$ attempts for each (experience increases each time)
How to win the game

- each element not currently matched keeps “experience”=(#refusals sent, #refusals received)
- the first is \(< N\); the second a priori is unbounded, but also will be kept \(< N\) due to agency strategy
- when \((x, y)\) is terminated, numbers updated
- invariant: in all pairs people have matching experiences (#sent = #received for the other)
- corollary: #refusals received \(< N\)
- new partner for \(x\) is found if possible (=there is \(y \in Y\) with matching experience not tried earlier with \(x\))
- otherwise \(x\) is declared hopeless
- invariant: for matching experiences the number of non-matched people in \(X\) and \(Y\) are the same
- \(\leq 2N\) attempts for each (experience increases each time)
- there are \(N^2\) experience classes; if class reaches \(2N\), it stops growing since \(y\) can be always found in the class (\(< 2N\) are tried earlier with given \(x\)), so \(O(N^3)\) hopeless
Thanks
Thanks

- to the organizers who accepted to consider these arguments
Thanks

- to the organizers who accepted to consider these arguments
- to Misha Vyugin and Andrej Muchnik who invented the game argument and its generalization for several strings $y_i$
Thanks

- to the organizers who accepted to consider these arguments
- to Misha Vyugin and Andrej Muchnik who invented the game argument and its generalization for several strings $y_i$
- to Laurent Bienvenu who convinced us to write this simple argument down
Thanks

- to the organizers who accepted to consider these arguments
- to Misha Vyugin and Andrej Muchnik who invented the game argument and its generalization for several strings $y_i$
- to Laurent Bienvenu who convinced us to write this simple argument down
- to all colleagues (ESCAPE team in Marseille and Montpellier, participants of Kolmogorov seminar in Moscow)
Thanks

- to the organizers who accepted to consider these arguments
- to Misha Vyugin and Andrej Muchnik who invented the game argument and its generalization for several strings $y_i$
- to Laurent Bienvenu who convinced us to write this simple argument down
- to all colleagues (ESCAPE team in Marseille and Montpellier, participants of Kolmogorov seminar in Moscow)
- to the audience for following the talk to that point :-)}