Polynomial time algorithms in Kolmogorov complexity theory

Marius Zimand

Towson University

CCR 2015

What is this talk about

- It will challenge the common perception that most objects in Kolmogorov complexity are uncomputable.
- In fact, not only are many important objects computable, they are **efficiently** computable (i.e., computable in polynomial time,)

  provided a few help bits are available, or a small error probability is allowed, or some reasonable complexity assumptions hold.

What is this talk about

- It will challenge the common perception that most objects in Kolmogorov complexity are uncomputable.
- In fact, not only are many important objects computable, they are **efficiently** computable (i.e., computable in polynomial time, )
  provided a few help bits are available, or a small error probability is allowed, or some reasonable complexity assumptions hold.

What is this talk about

- It will challenge the common perception that most objects in Kolmogorov complexity are uncomputable.
- In fact, not only are many important objects computable, they are **efficiently** computable (i.e., computable in polynomial time, )
  provided a few help bits are available, or a small error probability is allowed, or some reasonable complexity assumptions hold.
- It is a survey talk.
- Most results are not new; a few are new.

Kolmogorov complexity: notation

- U - optimal universal TM.
- If $U(p) = x$, we say p is a program for x.
- If $U(p, y) = x$, we say p is a program for x conditioned by y.
- $C(x) = \min\{|p| \mid p \text{ program for } x\}$.
- $C(x \mid y) = \min\{|p| \mid p \text{ program for } x \text{ conditioned by } y\}$.

Kolmogorov complexity: notation

- U - optimal universal TM.
- If $U(p) = x$, we say p is a program for x.
- If $U(p, y) = x$, we say p is a program for x conditioned by y.
- $C(x) = \min\{|p| \mid p \text{ program for } x\}$.
- $C(x \mid y) = \min\{|p| \mid p \text{ program for } x \text{ conditioned by } y\}$.
- $|x| =$ length of x; in general we denote $|x|$ by n.

Kolmogorov complexity: uncomputability results

1. $C(x)$ - canonical example of an uncomputable function.

Kolmogorov complexity: uncomputability results

1. $C(x)$ - canonical example of an uncomputable function.
2. Finding a shortest program for x: also uncomputable.

Kolmogorov complexity: uncomputability results

1. $C(x)$ - canonical example of an uncomputable function.
2. Finding a shortest program for x: also uncomputable.
3. No algorithm enumerates more than finitely many strings with high complexity.

Kolmogorov complexity: uncomputability results

1. $C(x)$ - canonical example of an uncomputable function.
2. Finding a shortest program for x: also uncomputable.
3. No algorithm enumerates more than finitely many strings with high complexity.
4. Can we prove all statements "$C(x) \geq k$?" NO.

Kolmogorov complexity: uncomputability results

1. $C(x)$ - canonical example of an uncomputable function.
2. Finding a shortest program for x: also uncomputable.
3. No algorithm enumerates more than finitely many strings with high complexity.
4. Can we prove all statements "$C(x) \geq k$?" NO.
5. Can we effectively label k-tuples of strings $(x_1, \ldots, x_k)$ with $\{\mathrm{Random}, \mathrm{Non-Random}\}^k$, so that at least one label is correct for each k-tuplet? NO, for every k (Teutsch, Z., 2014).

Kolmogorov complexity: uncomputability results

1. $C(x)$ - canonical example of an uncomputable function.
2. Finding a shortest program for x: also uncomputable.
3. No algorithm enumerates more than finitely many strings with high complexity.
4. Can we prove all statements "$C(x) \geq k$?" NO.
5. Can we effectively label k-tuples of strings $(x_1, \ldots, x_k)$ with $\{\mathrm{Random}, \mathrm{Non-Random}\}^k$, so that at least one label is correct for each k-tuplet? NO, for every k (Teutsch, Z., 2014).
6. No unbounded, computable function is a lower bound for Kolmogorov complexity (Zvonkin, Levin).

Kolmogorov complexity: uncomputability results

1. $C(x)$ - canonical example of an uncomputable function.
2. Finding a shortest program for x: also uncomputable.
3. No algorithm enumerates more than finitely many strings with high complexity.
4. Can we prove all statements "$C(x) \geq k$?" NO.
5. Can we effectively label k-tuples of strings $(x_1, \ldots, x_k)$ with $\{\mathrm{Random}, \mathrm{Non-Random}\}^k$, so that at least one label is correct for each k-tuplet? NO, for every k (Teutsch, Z., 2014).
6. No unbounded, computable function is a lower bound for Kolmogorov complexity (Zvonkin, Levin).
7. We want to compute a list of integers containing $C(x)$. Any such computable list must have size $\Omega(|x|)$ for infinitely many x. (Beigel, Buhrman, Fejer, Fortnow, Grabowski, Longpré, Muchnik, Stephan, Torenvliet, 2006).

Computing short programs

- Given x and $C(x)$, it is possible to compute a shortest program for x.

Computing short programs

- Given x and $C(x)$, it is possible to compute a shortest program for x.
- But the computation time is larger than any computable function.

Computing short programs

- Given x and $C(x)$, it is possible to compute a shortest program for x.
- But the computation time is larger than any computable function.
- In fact, for any computable function $t(n)$ if an algorithm on input $(x, C(x))$ computes in time $t(n)$ a program p for x, then $|p| \geq C(x) + \Omega(n)$ for infinitely many n. (Bauwens, Z. , 2014).

Computing short programs

- Given x and $C(x)$, it is possible to compute a shortest program for x.
- But the computation time is larger than any computable function.
- In fact, for any computable function $t(n)$ if an algorithm on input $(x, C(x))$ computes in time $t(n)$ a program p for x, then $|p| \geq C(x) + \Omega(n)$ for infinitely many n. (Bauwens, Z. , 2014).
- There is a probabilistic polynomial time algorithm that on input (x, ℓ) returns a string p of length $\leq \ell + O(\log^2(n))$, and if $\ell = C(x)$ then, with probability 0.99, p is a program for x (Bauwens, Z. , 2014).

Computing short programs

- Given x and $C(x)$, it is possible to compute a shortest program for x.
- But the computation time is larger than any computable function.
- In fact, for any computable function $t(n)$ if an algorithm on input $(x, C(x))$ computes in time $t(n)$ a program p for x, then $|p| \geq C(x) + \Omega(n)$ for infinitely many n. (Bauwens, Z. , 2014).
- There is a probabilistic polynomial time algorithm that on input (x, ℓ) returns a string p of length $\leq \ell + O(\log^2(n))$, and if $\ell = C(x)$ then, with probability 0.99, p is a program for x (Bauwens, Z. , 2014).
- The above is a <u>promise</u> algorithm. If the promise $\ell = C(x)$ holds, then the output has the coveted property (with high probability), if it does not hold, then no guarantee.

Computing short conditional programs

- On input $(x, y, C(x \mid y))$ it is possible to compute a program p of x conditioned by y of length $|p| = C(x \mid y)$.

Computing short conditional programs

- On input $(x, y, C(x \mid y))$ it is possible to compute a program p of x conditioned by y of length $|p| = C(x \mid y)$.
- (Muchnik's Theorem, 2002): On input $(x, \cancel{y, C(x \mid y)})$ and $O(\log n)$ help bits, one can compute a string p of length $C(x \mid y) + O(\log n)$ such that (p, y) is a program for x.

Computing short conditional programs

- On input $(x, y, C(x \mid y))$ it is possible to compute a program p of x conditioned by y of length $|p| = C(x \mid y)$.
- (Muchnik's Theorem, 2002): On input $(x, \cancel{y, C(x \mid y)})$ and $O(\log n)$ help bits, one can compute a string p of length $C(x \mid y) + O(\log n)$ such that (p, y) is a program for x.
- (Musatov, Romashchenko, Shen, 2011): Different proof for Muchnik's Th.

Muchnik's Theorem

Theorem (Muchnik's Theorem)

For every x, y of complexity at most n, there exists p such that

- (p, y) *is a program for* x.
- $C(p \mid x) = O(\log n)$,
- $|p| = C(x \mid y) + O(\log n)$,

Muchnik's Theorem

Theorem (Muchnik's Theorem)

overhead

For every x, y of comp… most n, there exists p such that

- (p, y) *is a program for* x
- $C(p \mid x) = O(\log n)$,
- $|p| = C(x \mid y) + O(\log n)$,

Muchnik's Theorem

help bits

Theorem (Muchnik's Theorem)

overhead

For every x, y of complexity at most n, there exists p such that

- (p, y) *is a program for* x
- $C(p \mid x) = O(\log n)$,
- $|p| = C(x \mid y) + O(\log n)$,

Muchnik's Theorem in polynomial time

- (Bauwens, Makhlin, Vereshchagin, Z., 2013) On input x one can compute in polynomial time a list containing a string p of length $C(x \mid y) + O(\log n)$ such that (p, y) is a program for x.

Muchnik's Theorem in polynomial time

- (Bauwens, Makhlin, Vereshchagin, Z., 2013) On input x one can compute in polynomial time a list containing a string p of length $C(x \mid y) + O(\log n)$ such that (p, y) is a program for x.
- (Teutsch, 2014) .... $C(x \mid y) + \cancel{O(\log n)} O(1)$.... list size $= n^{7+\epsilon}$.

Muchnik's Theorem in polynomial time

- (Bauwens, Makhlin, Vereshchagin, Z., 2013) On input x one can compute in polynomial time a list containing a string p of length $C(x \mid y) + O(\log n)$ such that (p, y) is a program for x.
- (Teutsch, 2014) .... $C(x \mid y) + \cancel{O(\log n)} O(1)$.... list size $= n^{7+\epsilon}$.
- (Z., 2014) .... $C(x \mid y) + O(1)$.... list size $= n^{6+\epsilon}$.

Muchnik's Theorem in polynomial time

- (Bauwens, Makhlin, Vereshchagin, Z., 2013) On input x one can compute in polynomial time a list containing a string p of length $C(x \mid y) + O(\log n)$ such that (p, y) is a program for x.
- (Teutsch, 2014) .... $C(x \mid y) + \cancel{O(\log n)} O(1)$.... list size $= n^{7+\epsilon}$.
- (Z., 2014) .... $C(x \mid y) + O(1)$.... list size $= n^{6+\epsilon}$.
- Is it possible that list size $= 1$? Yes, if a promise condition holds and if we allow some small error probability.

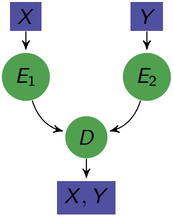
Muchnik's Theorem in polynomial time

- (Bauwens, Makhlin, Vereshchagin, Z., 2013) On input x one can compute in polynomial time a list containing a string p of length $C(x \mid y) + O(\log n)$ such that (p, y) is a program for x.
- (Teutsch, 2014) .... $C(x \mid y) + \cancel{O(\log n)} O(1)$.... list size $= n^{7+\epsilon}$.
- (Z., 2014) .... $C(x \mid y) + O(1)$.... list size $= n^{6+\epsilon}$.
- Is it possible that list size $= 1$? Yes, if a promise condition holds and if we allow some small error probability.
- Promise algorithms: The input must satisfy some promise. If it doesn't, then the algorithm does not guarantee anything (but still halts).

Muchnik's Theorem in polynomial time

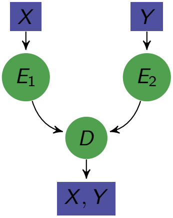
- (Bauwens, Makhlin, Vereshchagin, Z., 2013) On input x one can compute in polynomial time a list containing a string p of length $C(x \mid y) + O(\log n)$ such that (p, y) is a program for x.
- (Teutsch, 2014) .... $C(x \mid y) + \cancel{O(\log n)} O(1)$.... list size $= n^{7+\epsilon}$.
- (Z., 2014) .... $C(x \mid y) + O(1)$.... list size $= n^{6+\epsilon}$.
- Is it possible that list size $= 1$? Yes, if a promise condition holds and if we allow some small error probability.
- Promise algorithms: The input must satisfy some <u>promise</u>. If it doesn't, then the algorithm does not guarantee anything (but still halts).
- (Bauwens, Z., 2014) On input x, ℓ one can compute in probabilistic polynomial time a string p of length $\ell + (\log^2 n)$ and if the promise $\ell = C(x \mid y)$ holds, then, with probability 0.99, (p, y) is a program for x.

Slepian-Wolf Theorem



$(X_1, Y_1), \ldots, (X_n, Y_n)$, n i.i.d. random variables, $\{0,1\}$-valued, with joint distribution $p(x, y)$.
$X = (X_1, \ldots, X_n)$, $Y = (Y_1, \ldots, Y_n)$.
$E_1 : \{0,1\}^n \rightarrow \{0,1\}^{r_1 n}$,
$E_2 : \{0,1\}^n \rightarrow \{0,1\}^{r_2 n}$.
Rates r_1, r_2 are achievable if with high probability $D(E_1(X), E_2(Y)) = (X, Y)$.

Slepian-Wolf Theorem

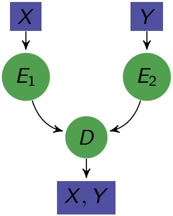


$(X_1, Y_1), \ldots, (X_n, Y_n)$, n i.i.d. random variables, $\{0,1\}$-valued, with joint distribution $p(x, y)$.
$X = (X_1, \ldots, X_n)$, $Y = (Y_1, \ldots, Y_n)$.
$E_1 : \{0,1\}^n \rightarrow \{0,1\}^{r_1 n}$,
$E_2 : \{0,1\}^n \rightarrow \{0,1\}^{r_2 n}$.
Rates r_1, r_2 are achievable if with high probability $D(E_1(X), E_2(Y)) = (X, Y)$.

Question: What rates are achievable?

Clearly it is necessary that $r_1 + r_2 \geq H(X_i, Y_i)$, $r_1 \geq H(X_i \mid Y_i)$, $r_2 \geq H(Y_i \mid X_i)$.

Slepian-Wolf Theorem



$(X_1, Y_1), \ldots, (X_n, Y_n)$, n i.i.d. random variables, $\{0,1\}$-valued, with joint distribution $p(x, y)$.
$X = (X_1, \ldots, X_n)$, $Y = (Y_1, \ldots, Y_n)$.
$E_1 : \{0,1\}^n \to \{0,1\}^{r_1 n}$,
$E_2 : \{0,1\}^n \to \{0,1\}^{r_2 n}$.
Rates r_1, r_2 are achievable if with high probability $D(E_1(X), E_2(Y)) = (X, Y)$.

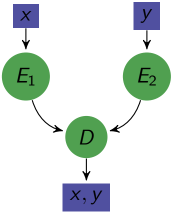
Question: What rates are achievable?

Clearly it is necessary that $r_1 + r_2 \geq H(X_i, Y_i)$, $r_1 \geq H(X_i \mid Y_i)$, $r_2 \geq H(Y_i \mid X_i)$.

Theorem (Slepian-Wolf)

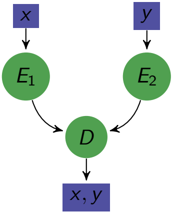
Any pair (r_1, r_2) satisfying the above inequalities is achievable.

Kolmogorov complexity versions of the Slepian-Wolf Theorem - (1)



x, y binary strings
$E_1 : \{0,1\}^* \to \{0,1\}^*$,
$E_2 : \{0,1\}^* \to \{0,1\}^*$.
Decoding task: $D(E_1(x), E_2(y)) = (x, y)$.

Kolmogorov complexity versions of the Slepian-Wolf Theorem - (1)

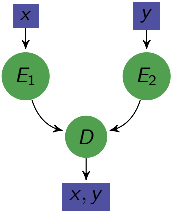


x, y binary strings
$E_1 : \{0,1\}^* \to \{0,1\}^*$,
$E_2 : \{0,1\}^* \to \{0,1\}^*$.
Decoding task: $D(E_1(x), E_2(y)) = (x, y)$.

Question: What rates s, t ((i.e., lengths of $E_1(x), E_2(y)$) are achievable?

Clearly it is necessary that $s + t \geq C(x, y)$, $s \geq C(x \mid y)$, $t \geq C(y \mid x)$.

Kolmogorov complexity versions of the Slepian-Wolf Theorem - (1)



x, y binary strings
$E_1 : \{0,1\}^* \to \{0,1\}^*$,
$E_2 : \{0,1\}^* \to \{0,1\}^*$.
Decoding task: $D(E_1(x), E_2(y)) = (x, y)$.

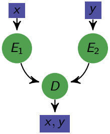
Question: What rates s, t ((i.e., lengths of $E_1(x), E_2(y)$) are achievable?

Clearly it is necessary that $s + t \geq C(x, y)$, $s \geq C(x \mid y)$, $t \geq C(y \mid x)$.

Theorem (Muchnik Theorem)

$s = C(x \mid y) + O(\log n)$, $t = C(y)$ is achievable (provided E_1, E_2 can use $O(\log n)$ help bits).

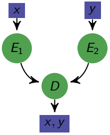
Kolmogorov complexity versions of the Slepian-Wolf Theorem -(2)



x, y binary strings
$E_1 : \{0,1\}^* \to \{0,1\}^*$,
$E_2 : \{0,1\}^* \to \{0,1\}^*$.
Decoding task: $D(E_1(x), E_2(y)) = (x, y)$.

Question: What rates s, t (i.e., lengths of $E_1(x), E_2(y)$) are achievable?

Kolmogorov complexity versions of the Slepian-Wolf Theorem -(2)



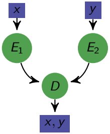
x, y binary strings
$E_1 : \{0,1\}^* \to \{0,1\}^*$,
$E_2 : \{0,1\}^* \to \{0,1\}^*$.
Decoding task: $D(E_1(x), E_2(y)) = (x, y)$.

Question: What rates s, t (i.e., lengths of $E_1(x), E_2(y)$) are achievable?

[Teutsch 2014,]
$s = C(x \mid y) + O(1), t = C(y)$ are achievable with polynomial time E_1 and E_2 (provided E_1, E_2 can use $O(\log n)$ help bits).

[Bauwens, Z., 2014,] $s = C(x \mid y) + O(\log^2 n), t = C(y)$ are achievable with probabilistic polynomial time E_1 and E_2 (provided E_1 knows $C(x \mid y)$, E_2 knows $C(y)$).

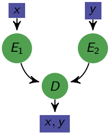
Kolmogorov complexity versions of the Slepian-Wolf Theorem -(3)



x, y binary strings
$E_1 : \{0,1\}^* \to \{0,1\}^*$,
$E_2 : \{0,1\}^* \to \{0,1\}^*$.
Decoding task: $D(E_1(x), E_2(y)) = (x, y)$.

Question: What rates s, t (i.e., lengths of $E_1(x), E_2(y)$) are achievable?

Kolmogorov complexity versions of the Slepian-Wolf Theorem -(3)

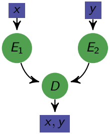


x, y binary strings
$E_1 : \{0,1\}^* \to \{0,1\}^*$,
$E_2 : \{0,1\}^* \to \{0,1\}^*$.
Decoding task: $D(E_1(x), E_2(y)) = (x, y)$.

Question: What rates s, t (i.e., lengths of $E_1(x), E_2(y)$) are achievable?

[Z. 2015,] Roughly any s, t with $s + t \geq C(x, y), s \geq C(x \mid y), t \geq C(y \mid x)$ are achievable (if a few help bits are available, or some promise conditions hold).

Kolmogorov complexity versions of the Slepian-Wolf Theorem -(3)



x, y binary strings
$E_1 : \{0,1\}^* \to \{0,1\}^*$,
$E_2 : \{0,1\}^* \to \{0,1\}^*$.
Decoding task: $D(E_1(x), E_2(y)) = (x, y)$.

Question: What rates s, t (i.e., lengths of $E_1(x), E_2(y)$) are achievable?

[Z. 2015, ]

- Let s, t be such that $s \geq C(x \mid y) + O(\log^3 n)$, $t \geq C(y \mid x) + O(\log n)$, and $s + t \geq C(x, y)$.
  $|E_1(x)| = s, |E_2(y)| = t$ is achievable with polynomial time E_1 and E_2 (provided E_1 can use $O(\log^3 n)$ help bits, and E_2 can use $O(\log n)$ help bits).
- Let s, t be such that $s \geq C(x \mid y) + O(\log^3 n)$, $t \geq C(y \mid x) + O(\log^3 n)$, and $s + t \geq C(x, y)$.
  $|E_1(x)| = s, |E_2(y)| = t$ is achievable with probabilistic polynomial time E_1 and E_2 (provided that E_1 knows $C(x), C(x \mid y)$, E_2 knows $C(y \mid x)$).

Coding Theorems

Theorem (Shannon Source Coding Theorem)

Let X be a random variable with finite support.
Then there is a way to code the support of X such that
$E[|code(X)|] \leq H(X) + 1$.

(H is Shannon entropy; $H(X) = E[-\log p(X)]$.
So, $E[|code(X)|] \leq E[-\log p(X)] + 1$).

Theorem (Levin, Chaitin)

Let μ be a left c.e. semi-measure.
Then for all x, $K(x) \leq -\log \mu(x) + O(1)$.
(K is the prefix-free Kolmogorov complexity.)

Coding Theorems

Theorem (Shannon Source Coding Theorem)

Let X be a random variable with finite support.
Then there is a way to code the support of X such that
$E[|code(X)|] \leq H(X) + 1$.

(H is Shannon entropy; $H(X) = E[-\log p(X)]$.
So, $E[|code(X)|] \leq E[-\log p(X)] + 1$).

Theorem (Levin, Chaitin)

Let μ be a left c.e. semi-measure.
Then for all x, $K(x) \leq -\log \mu(x) + O(1)$.
(K is the prefix-free Kolmogorov complexity.)

Is there a polynomial-time Coding Theorem?

Polynomial-time Coding Theorem

• A probability distribution is P-samplable if there exists a polynomial time (family of) computable function $F : \{0,1\}^m \to \{0,1\}^n$, with $n \geq m^{\Omega(1)}$, such that

$$\mu(x) = \frac{|\{w \in \{0,1\}^m \mid F(w) = x\}|}{2^m}.$$

Theorem (Antunes-Fortnow, 2009,)

Assume assumption H.
If μ is P-samplable, there exists a polynomial p, such that for all x,
$C^p(x) \leq -\log(\mu(x)) + O(\log n)$.
($C^p(\cdot)$ is the Kolm. complexity with time bound p.)

Polynomial-time Coding Theorem

• A probability distribution is P-samplable if there exists a polynomial time (family of) computable function $F : \{0,1\}^m \to \{0,1\}^n$, with $n \geq m^{\Omega(1)}$, such that

$$\mu(x) = \frac{|\{w \in \{0,1\}^m \mid F(w) = x\}|}{2^m}.$$

Theorem (Antunes-Fortnow, 2009,)

Assume assumption H.
If μ is P-samplable, there exists a polynomial p, such that for all x,
$C^p(x) \leq -\log(\mu(x)) + O(\log n)$.
($C^p(\cdot)$ is the Kolm. complexity with time bound p.)

Assumption H: $\exists f \in \mathrm{E}$ which cannot be computed in space $2^{o(n)}$.

Polynomial-time Coding Theorem

• A probability distribution is P-samplable if there exists a polynomial time (family of) computable function $F : \{0,1\}^m \to \{0,1\}^n$, with $n \geq m^{\Omega(1)}$, such that

$$\mu(x) = \frac{|\{w \in \{0,1\}^m \mid F(w) = x\}|}{2^m}.$$

Theorem (Antunes-Fortnow, 2009,)

Assume assumption H.
If μ is P-samplable, there exists a polynomial p, such that for all x,
$C^p(x) \leq -\log(\mu(x)) + O(\log n)$.
($C^p(\cdot)$ is the Kolm. complexity with time bound p.)

Assumption H: $\exists f \in \mathrm{E}$ which cannot be computed in space $2^{o(n)}$.

$\mathrm{E} = \cup_{c>0} \mathrm{DTIME}[2^{cn}]$

Some proofs...

The typical route:

1. Find an appropriate combinatorial object for the job.
2. Show that it exists using the probabilistic method.
3. Construct it in polynomial time using tools from the theory of pseudo randomness:

   expanders, extractors, dispersers, pseudo-random generators.

Example of a combinatorial object

Key tool: bipartite graphs $G = (L, R, E \subseteq L \times R)$ with the rich owner property:

For any $B \subseteq L$ of size $|B| \approx K$, most x in B own most of their neighbors (these neighbors are not shared with any other node from B).

Example of a combinatorial object

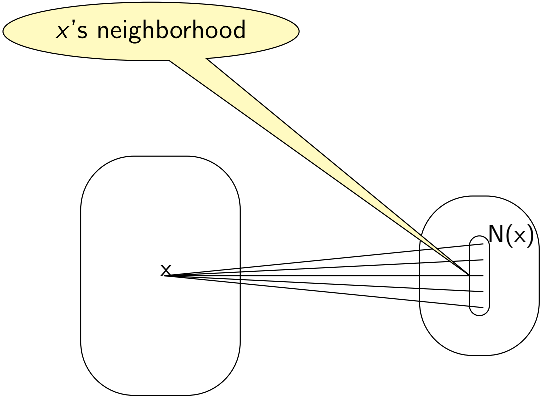
Key tool: bipartite graphs $G = (L, R, E \subseteq L \times R)$ with the rich owner property:

For any $B \subseteq L$ of size $|B| \approx K$, most x in B own most of their neighbors (these neighbors are not shared with any other node from B).

- $x \in B$ owns $y \in N(x)$ w.r.t. B if $N(y) \cap B = \{x\}$.
- $x \in B$ is a rich owner if x owns $(1-\delta)$ of its neighbors w.r.t. B.
- $G = (L, R, E \subseteq L \times R)$ has the (K, δ)-rich owner property if

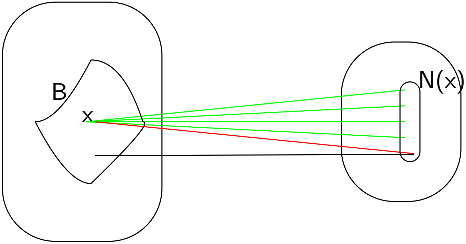
for all B with $|B| \leq K$, $(1-\delta)K$ of the elements in B are rich owners w.r.t. B.

Bipartite graph G



Bipartite graph G

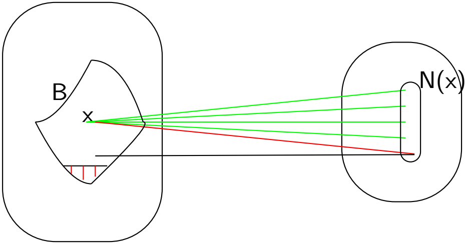
x is a rich owner w.r.t B if x owns $(1-\delta)$ of $N(x)$



Bipartite graph G

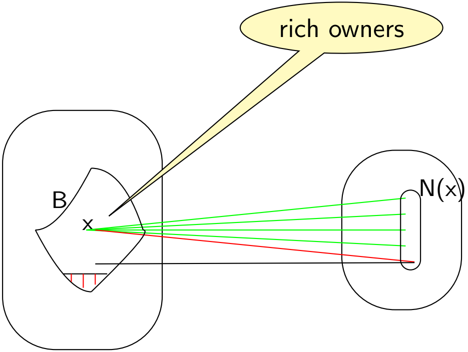
x is a rich owner w.r.t B if x owns $(1-\delta)$ of $N(x)$

G has the (K, δ) rich owner property: $\forall B \subseteq L$, of size at most K, all nodes in B except at most $\delta \cdot K$ are rich owners w.r.t. B



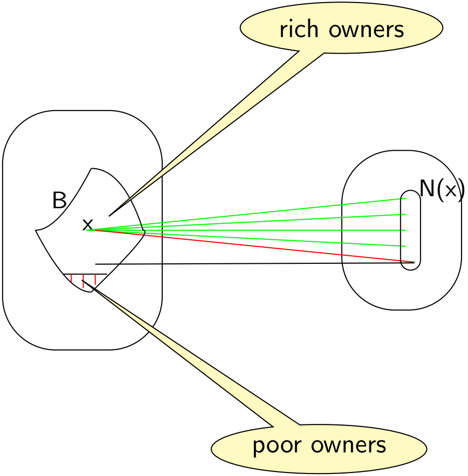
x is a rich owner w.r.t B if x owns $(1-\delta)$ of $N(x)$

G has the (K, δ) rich owner property: $\forall B \subseteq L$, of size at most K, all nodes in B except at most $\delta \cdot K$ are rich owners w.r.t. B



x is a rich owner w.r.t B if x owns $(1-\delta)$ of $N(x)$

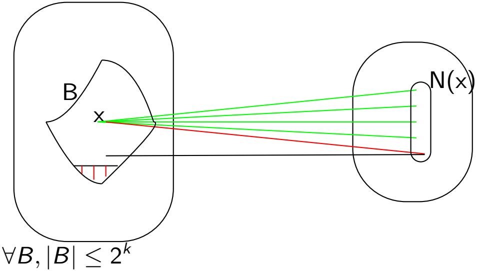
G has the (K, δ) rich owner property: $\forall B \subseteq L$, of size at most K, all nodes in B except at most $\delta \cdot K$ are rich owners w.r.t. B



Theorem (Bauwens, Z'14)

There exists a poly time computable (uniformly in n, k and $1/\delta$) graph with the rich owner property for parameters $(2^k, \delta)$ with:

- $L = \{0,1\}^n$
- $R = \{0,1\}^{k+O(\log^2(n/\delta))}$
- $D(\textit{left degree}) = 2^{O(\log^2(n/\delta))}$.



Short programs in probabilistic poly. time

Theorem (Bauwens, Z., 2014)

There exists a probabilistic poly. time algorithm A such that
- *On input* (x, δ) *and promise parameter* k, A *outputs* p,
- $|p| = k + \log^2(|x|/\delta)$,
- *If the promise condition* $k = C(x)$ *holds, then,*

with probability $(1 - \delta)$, p *is a program for* x.

Lemma

There exists a **poly-time** *algorithm A that*
Input: $x \in \{0,1\}^n$, $k \in \mathbf{N}$, $\delta > 0$
Output: list of size $2^{\log^2(n/\delta)}$, *each element of length* $k + O(\log^2(n/\delta))$
If $k = C(x)$ *then* $(1-\delta)$ *of the elements are programs for* x.
(each element of the list printed in poly time).

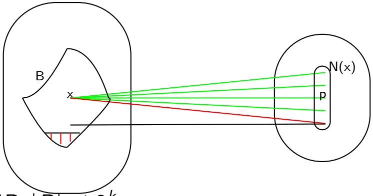
Lemma

There exists a **poly-time** *algorithm A that*
Input: $x \in \{0,1\}^n$, $k \in \mathbf{N}$, $\delta > 0$
Output: list of size $2^{\log^2(n/\delta)}$, *each element of length* $k + O(\log^2(n/\delta))$
If $k = C(x)$ *then* $(1-\delta)$ *of the elements are programs for* x.
(each element of the list printed in poly time).

The theorem follows immediately by taking p to be a random element from the list $A(x, k, \delta)$.

Theorem [Bauwens, Z'14] There exists a poly.time computable (uniformly in n, k and $1/\delta$) graph with the rich owner property for parameters $(2^k, \delta)$ with:

- $L = \{0,1\}^n$
- $R = \{0,1\}^{k+O(\log^2(n/\delta))}$
- $D(\text{left degree}) = 2^{O(\log^2(n/\delta))}$.



$\forall B, |B| \le 2^k$

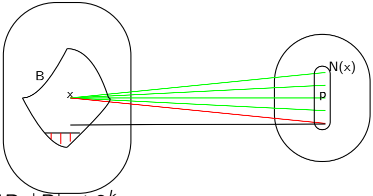
We obtain our lists:

- List for x: $N(x)$
- Any $p \in N(x)$ owned by x w.r.t. $B = \{x' \mid C(x') \le k\}$ is a program for x.

How to construct x from p: Enumerate B till we find an element that owns p. This is x.

Theorem [Bauwens, Z'14] There exists a poly.time computable (uniformly in n, k and $1/\delta$) graph with the rich owner property for parameters $(2^k, \delta)$ with:

- $L = \{0,1\}^n$
- $R = \{0,1\}^{k+O(\log^2(n/\delta))}$
- D(left degree) $= 2^{O(\log^2(n/\delta))}$.



$\forall B, |B| \leq 2^k$

We obtain our lists:

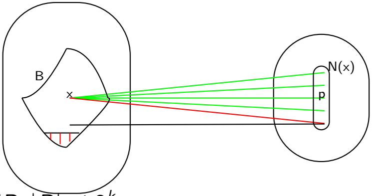
- List for x: $N(x)$
- Any $p \in N(x)$ owned by x w.r.t. $B = \{x' \mid C(x') \leq k\}$ is a program for x.

How to construct x from p: Enumerate B till we find an element that owns p. This is x.

- So if x is a rich owner, $(1-\delta)$ of his neighbors are programs for it.

Theorem [Bauwens, Z'14] There exists a poly.time computable (uniformly in n, k and $1/\delta$) graph with the rich owner property for parameters $(2^k, \delta)$ with:

- $L = \{0,1\}^n$
- $R = \{0,1\}^{k+O(\log^2(n/\delta))}$
- D(left degree) $= 2^{O(\log^2(n/\delta))}$.



$\forall B, |B| \le 2^k$

We obtain our lists:

- List for x: $N(x)$
- Any $p \in N(x)$ owned by x w.r.t. $B = \{x' \mid C(x') \le k\}$ is a program for x.

How to construct x from p: Enumerate B till we find an element that owns p. This is x.

- So if x is a rich owner, $(1-\delta)$ of his neighbors are programs for it.
- What if x is a poor owner? There are few poor owners, so x has complexity $< k$.

Building graphs with the rich owner property

- Step 1: $(1 - \delta)$ of $x \in B$ **partially** own $(1 - \delta)$ of its neighbors.

Building graphs with the rich owner property

shared with only $\mathrm{poly}(n)$ nodes

- Step 1: $(1-\delta)$ of $x \in B$ **partially** own $(1-\delta)$ of its neighbors.

Building graphs with the rich owner property

shared with only $\mathrm{poly}(n)$ nodes

- Step 1: $(1 - \delta)$ of $x \in B$ **partially** own $(1 - \delta)$ of its neighbors.

- Step 2: $(1 - \delta)$ of $x \in B$ ~~**partially**~~ own $(1 - \delta)$ of its neighbors.

Building graphs with the rich owner property

shared with only $\mathrm{poly}(n)$ nodes

- Step 1: $(1-\delta)$ of $x \in B$ **partially** own $(1-\delta)$ of its neighbors.
- Step 2: $(1-\delta)$ of $x \in B$ ~~**partially**~~ own $(1-\delta)$ of its neighbors.

Step 1 is done with extractors that have small entropy loss.

Step 2 is done by hashing.

Extractors

$E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$,

$$|\mathrm{Prob}(E(U_B\ , U_d\ ) \in A) - \mathrm{Prob}(U_m\ \in A)| \leq \epsilon,$$

Extractors

uniform distr. on $\{0,1\}^m$

$E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-extractor for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$,

$$|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(U_m \in A)| \leq \epsilon,$$

uniform distr. on B

uniform distr. on $\{0,1\}^d$

Extractors

$E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$,

$$|\mathrm{Prob}(E(U_B\ , U_d\ ) \in A) - \mathrm{Prob}(U_m\ \in A)| \leq \epsilon,$$

or, in other words,

$$\left| \frac{|E(B,A)|}{|B| \cdot 2^d} - \frac{|A|}{2^m} \right| \leq \epsilon.$$

The entropy loss is $s = k + d - m$.

Step 1

GOAL : $\forall B \subseteq L$ **with** $|B| \approx K$**, most nodes in** B **share most of their neighbors with only** $\mathrm{poly}(n)$ **other nodes from** B**.**

We can view an extractor E as a bipartite graph G_E with $L = \{0,1\}^n, R = \{0,1\}^m$ and left-degree $D = 2^d$.

If E is a (k, ϵ)-extractor, then it has low congestion:
for any $B \subseteq L$ of size $|B| \approx 2^k$, most $x \in B$ share most of their neighbors with only $O(1/\epsilon \cdot 2^s)$ other nodes in B.

Step 1

GOAL : $\forall B \subseteq L$ **with** $|B| \approx K$**, most nodes in** B **share most of their neighbors with only** $\mathrm{poly}(n)$ **other nodes from** B**.**

We can view an extractor E as a bipartite graph G_E with $L = \{0,1\}^n, R = \{0,1\}^m$ and left-degree $D = 2^d$.

If E is a (k, ϵ)-extractor, then it has low congestion: for any $B \subseteq L$ of size $|B| \approx 2^k$, most $x \in B$ share most of their neighbors with only $O(1/\epsilon \cdot 2^s)$ other nodes in B.

proof on next slide

Step 1

GOAL : $\forall B \subseteq L$ with $|B| \approx K$, most nodes in B share most of their neighbors with only $\mathrm{poly}(n)$ other nodes from B.

We can view an extractor E as a bipartite graph G_E with $L = \{0,1\}^n, R = \{0,1\}^m$ and left-degree $D = 2^d$.

If E is a (k, ϵ)-extractor, then it has low congestion:
for any $B \subseteq L$ of size $|B| \approx 2^k$, most $x \in B$ share most of their neighbors with only $O(1/\epsilon \cdot 2^s)$ other nodes in B.

By the probabilistic method: There are extractors whith entropy loss $s = O(\log(1/\epsilon))$ and log-left degree $d = O(\log n/\epsilon)$.

[Guruswami, Umans, Vadhan, 2009] Poly-time extractors with entropy loss $s = O(\log(1/\epsilon))$ and log-left degree $d = O(\log^2 n/\epsilon)$.

So for $1/\epsilon = \mathrm{poly}(n)$, we get our GOAL.

Extractors have low congestion

DEF: $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$, $|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(A)| \leq \epsilon$.
The entropy loss is $s = k + d - m$.

Extractors have low congestion

DEF: $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$, $|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(A)| \leq \epsilon$.
The entropy loss is $s = k + d - m$.

Lemma

Let E be a (k,ϵ)-extractor, $B \subseteq L$, $|B| = \frac{1}{\epsilon}2^k$.
Then all $x \in B$, except at most 2^k, share $(1-2\epsilon)$ of $N(x)$ with at most $2^s(\frac{1}{\epsilon})^2$ other nodes in B.

Extractors have low congestion

DEF: $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$, $|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(A)| \leq \epsilon$.
The entropy loss is $s = k + d - m$.

Lemma

Let E be a (k,ϵ)-extractor, $B \subseteq L$, $|B| = \frac{1}{\epsilon}2^k$.
Then all $x \in B$, except at most 2^k, share $(1-2\epsilon)$ of $N(x)$ with at most $2^s(\frac{1}{\epsilon})^2$ other nodes in B.

PROOF. Restrict left side to B. Avg-right-degree $= \frac{|B|2^d}{2^m} = \frac{1}{\epsilon} \cdot 2^s$.

Take A - the set of right nodes with $\deg_B \geq (2^s(1/\epsilon)) \cdot (1/\epsilon)$. Then $|A|/|R| \leq \epsilon$.

Take B' the nodes in B that do not have the property, i.e., they have $> 2\epsilon$ fraction of neighbors in A.

$|\mathrm{Prob}(E(U_{B'}, U_d) \in A) - |A|/|R|| > |2\epsilon - \epsilon| = \epsilon$.

So $|B'| \leq 2^k$.

Step 2

GOAL: Reduce sharing most neighbors with $\mathrm{poly}(n)$ other nodes, to sharing them with no other nodes.

y is shared by x with $x_2, \ldots, x_{\mathrm{poly}(n)}$

x y

Step 2

GOAL: Reduce sharing most neighbors with $\mathrm{poly}(n)$ other nodes, to sharing them with no other nodes.

Let $x_1, x_2, \ldots, x_{\mathrm{poly}(n)}$ be n-bit strings.

Consider $p_1, \ldots, p_T$ the first T prime numbers, where $T = (1/\delta) \cdot n \cdot \mathrm{poly}(n)$.

For every x_i, for $(1-\delta)$ of the T prime numbers, (x_i mod p) is unique in (x_1 mod $p, \ldots, x_T$ mod p).

y is shared by x with $x_2, \ldots, x_{\mathrm{poly}(n)}$

x ——— y

Step 2

GOAL: Reduce sharing most neighbors with $\mathrm{poly}(n)$ other nodes, to sharing them with no other nodes.

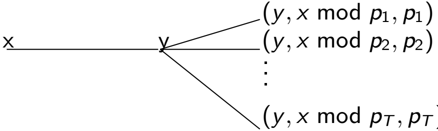
Let $x_1, x_2, \ldots, x_{\mathrm{poly}(n)}$ be n-bit strings.

Consider $p_1, \ldots, p_T$ the first T prime numbers, where $T = (1/\delta) \cdot n \cdot \mathrm{poly}(n)$.

For every x_i, for $(1-\delta)$ of the T prime numbers, $(x_i \bmod p)$ is unique in $(x_1 \bmod p, \ldots, x_T \bmod p)$.

In this way, by "splitting" each edge into T new edges we reach our GOAL.

y is shared by x with $x_2, \ldots, x_{\mathrm{poly}(n)}$



Step 2

GOAL: Reduce sharing most neighbors with $\mathrm{poly}(n)$ other nodes, to sharing them with no other nodes.

Let $x_1, x_2, \ldots, x_{\mathrm{poly}(n)}$ be n-bit strings.

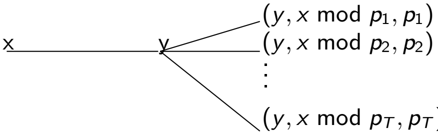
Consider $p_1, \ldots, p_T$ the first T prime numbers, where $T = (1/\delta) \cdot n \cdot \mathrm{poly}(n)$.

For every x_i, for $(1-\delta)$ of the T prime numbers, (x_i mod p) is unique in (x_1 mod $p, \ldots, x_T$ mod p).

In this way, by "splitting" each edge into T new edges we reach our GOAL.

Cost: overhead of $O(\log n)$ to the right nodes and the left degree increases by a factor of $T = \mathrm{poly}(n)$.

y is shared by x with $x_2, \ldots, x_{\mathrm{poly}(n)}$



Polynomial time Coding Theorem

Theorem (Antunes, Fortnow)

Let us assume complexity assumption H holds.
Let μ be a P-samplable distribution.
There exists a polynomial p such that for every x, $C^p(x) \leq -\log \mu(x) + O(\log n)$.

Polynomial time Coding Theorem

Theorem (Antunes, Fortnow)

Let us assume complexity assumption H holds.
Let μ be a P-samplable distribution.
There exists a polynomial p such that for every x, $C^p(x) \leq -\log \mu(x) + O(\log n)$.

Assumption H

$\exists f \in \mathrm{E}$ which cannot be computed in space $2^{o(n)}$.

Polynomial time Coding Theorem

Theorem (Antunes, Fortnow)

Let us assume complexity assumption H holds.
Let μ be a P-samplable distribution.
There exists a polynomial p such that for every x, $C^p(x) \leq -\log \mu(x) + O(\log n)$.

Assumption H

$\exists f \in \mathrm{E}$ which cannot be computed in space $2^{o(n)}$.

$\mathrm{E} = \cup_{c>0}\mathrm{DTIME}[2^{cn}]$

Assumption H implies pseudo-random generators that fool PSPACE predicates

[Nisan-Wigderson'94, Klivans - van Melkebeek'02, Miltersen'01]

If H is true, then there exists a pseudo-random generator g that fools any predicate computable in PSPACE.

There exists $g : \{0,1\}^{c \log n} \to \{0,1\}^n$ such that for any T computable in PSPACE

$$\left|\mathrm{Prob}[T(g(U_s))] - \mathrm{Prob}_R[T(U_n)]\right| < \epsilon.$$

Theorem [Antunes, Fortnow] Assume H holds. Let μ be a P-samplable distribution. There exists a polynomial p such that for every x, $C^p(x) \leq -\log \mu(x) + O(\log n)$.

Proof (sketch):

- There is poly time $F : \{0,1\}^m \to \{0,1\}^n$, $n \geq m^{\Omega(1)}$, s.t.

$$\mu(x) = |\{w \in \{0,1\}^m \mid F(w) = x\}|/2^m.$$

- Pick maximal k such that $\mu(x) \geq 2^{k-m}$.
- Let $T_x = \{w \in \{0,1\}^m \mid F(w) = x\}$.
- We need that some $w \in T_x$ has $C^p(w) \leq m - k + O(\log n)$.
- Let $\mathrm{HEAVY}_k = \{x' \mid |x'| = n, |T_{x'}| \geq 2^k\}$. $|\mathrm{HEAVY}_k| \leq 2^m/2^k$ (because the $T_{x'}$ are disjoint).
- Take $\ell = m - k + c \log n$, consider random $H : \{0,1\}^\ell \to \{0,1\}^m$.
- H is good if range(H) intersects every $T_{x'}$ with x' in HEAVY_k.
- By coupon collecting, most H are good (if c is large enough).

- Checking "H is good" is in PSPACE. So there is poly time G_1

$$G_1 : \{0,1\}^{poly(n)} \to \{0,1\}^{|H|}$$

  so that for most v, $G_1(v)$ is a good H.
- Checking "$G_1(v)$ is good" is in PSPACE. So there is poly time G_2

$$G_2 : \{0,1\}^{O(\log(n))} \to \{0,1\}^{|v|}$$

  so that for most v', $G_2(v')$ is a good v, $G_1(G_2(v'))$ is a good H.
- For some v', range($G_1(G_2(v'))$) intersects T_x.
- So there is z such that $G_1(G_2(v'))(z) = w$ and $F(w) = x$.
- So $C^p(x) \le |z| + |v'| = m - k + c \log n + O(\log n) \le -\log(\mu(x)) + O(\log n)$.

Kolmogorov complexity version of the Slepian-Wolf Th.

Theorem (Z)

Let x, y be binary strings and s, t numbers such that
- $s + t \geq C(x, y)$
- $s \geq C(x \mid y)$
- $t \geq C(y \mid x)$.

There exists strings p, q such that

(1) $|p| = s + O(\log^3 |x|)$, $|q| = t + O(\log(|x| + |y|))$.

(2) $C^{\mathrm{poly}}(p \mid x) = O(\log^3 |x|), C^{\mathrm{poly}}(q \mid y) = O(\log |y|)$

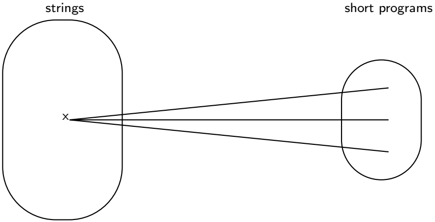
(3) (p, q) *is a program for* (x, y).

On-line matching

Bipartite graphs satisfying 2^k-online matching.

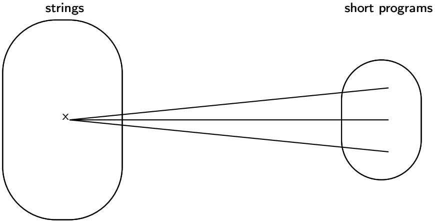
On-line matching

Bipartite graphs satisfying 2^k-online matching.



On-line matching

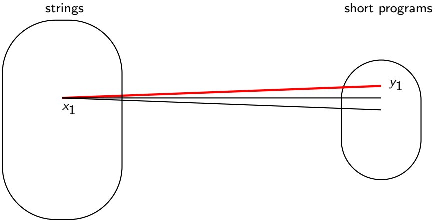
Bipartite graphs satisfying 2^k-online matching.



- matching requests arrive one by one
- request x: match $x \in LEFT$ with a free node $y \in N(x)$,
- Promise: there are $\leq 2^k$ requests.
- Requirement: all requests should be satisfied online (before seeing the next request).

On-line matching

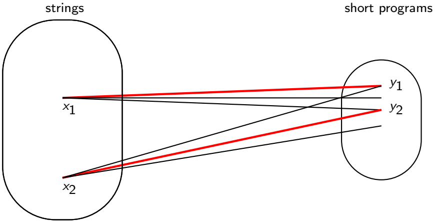
Bipartite graphs satisfying 2^k-online matching.



- matching requests arrive one by one
- request x: match $x \in LEFT$ with a free node $y \in N(x)$,
- Promise: there are $\leq 2^k$ requests.
- Requirement: all requests should be satisfied online (before seeing the next request).

On-line matching

Bipartite graphs satisfying 2^k-online matching.



- matching requests arrive one by one
- request x: match $x \in LEFT$ with a free node $y \in N(x)$,
- Promise: there are $\leq 2^k$ requests.
- Requirement: all requests should be satisfied online (before seeing the next request).

Proof overview

- Let $n = C(x)$. We can assume that $|x| = n$.
- Let $n_1 = C(x \mid y), n_2 = C(y \mid x)$.
- Let $A = \{(x', y') \in \{0,1\}^{|x|} \times \{0,1\}^{|y|} \mid C(x' \mid y') \leq n_1, C(y' \mid x') \leq n_2\}$.
- We show that there exist explicit bipartite graphs G_1, G_2 with the following property:
  1. We enumerate A
  2. Each enumerated (x', y') is matched on-line with some (p', q') such that (x', p') edge in G_1 and (y', q') edge in G_2.
  3. In particular (x, y) is matched to (p, q), so (p, q) is a description of (x, y).
  4. p and q have the desired lengths.
  5. G_1 has left degree $D_1 = 2^{O(\log^3 |x|)}$, so $C^{\mathrm{poly}}(p \mid x) = O(\log^3 |x|)$.
  6. G_2 has left degree $D_2 = 2^{O(\log |y|)}$, so $C^{\mathrm{poly}}(q \mid y) = O(\log |y|)$.

Proof overview

right neighbor is computed in poly time

- Let $n = C(x)$. We can assume $|x| = n$.
- Let $n_1 = C(x \mid y), n_2 = C(y \mid x)$
- Let $A = \{(x', y') \in \{0,1\}^{|x|} \times \{0,1\}^{|y|} \mid C(x' \mid y') \le n_1, C(y' \mid x') \le n_2\}$.
- We show that there exist explicit bipartite graphs G_1, G_2 with the following property:
  1. We enumerate A
  2. Each enumerated (x', y') is matched on-line with some (p', q') such that (x', p') edge in G_1 and (y', q') edge in G_2.
  3. In particular (x, y) is matched to (p, q), so (p, q) is a description of (x, y).
  4. p and q have the desired lengths.
  5. G_1 has left degree $D_1 = 2^{O(\log^3 |x|)}$, so $C^{\mathrm{poly}}(p \mid x) = O(\log^3 |x|)$.
  6. G_2 has left degree $D_2 = 2^{O(\log |y|)}$, so $C^{\mathrm{poly}}(q \mid y) = O(\log |y|)$.

Refined "rich owner" property

- We use bipartite graphs $G = (L, R, E \subseteq L \times R)$, where $R = \{0,1\}^\ell \times \{0,1\}^m$.
- For $m' < m$, the m'-prefix of $(y_1, y_2) \in R$ is (y_1, y_2'), where y_2' is the m'-prefix of y_2.
- The m'-level of G, is G' obtained from G by collapsing the right nodes that have the same m'-prefix.
- G has the incremental $(2^k, \delta)$-rich owner property if for any $m' < m$, the m'-level of G has the $(2^{k-(m-m')}, \delta)$-rich owner property.

Combining [Raz, Reingold, Vadhan'99] and [Bauwens,Z'14], there exists $G_1 = (L_1, R_1, E_1 \subseteq L_1 \times R_1)$ incremental $(2^s, \delta)$ rich owner property, with

1. $L_1 = \{0,1\}^{|x|}$,
2. $R_1 = \{0,1\}^{O(\log^3(|x|/\delta))} \times \{0,1\}^s$,
3. left degree $= 2^{d_1}$, $d_1 = O(\log^3 |x|/\delta)$,
4. G_1 is explicit: given left node x and i, we can produce the i-th neighbor of x in poly time.

Combining [Raz, Reingold, Vadhan'99] and [Bauwens,Z'14], there exists

$G_1 = (L_1, R_1, E_1 \subseteq L_1 \times R_1)$ incremental $(2^s, \delta)$ rich owner property, with

1. $L_1 = \{0,1\}^{|x|}$,
2. $R_1 = \{0,1\}^{O(\log^3(|x|/\delta))} \times \{0,1\}^s$,
3. left degree $= 2^{d_1}$, $d_1 = O(\log^3 |x|/\delta)$,
4. G_1 is explicit: given left node x and i, we can produce the i-th neighbor of x in poly time.

Using [Teutsch'14], there exists

$G_2 = (L_2, R_2, E_2 \subseteq L_2 \times R_2)$ that satisfies $2^{t+c\log(|x|+|y|)}$ on-line matching requests, with

1. $L_2 = \{0,1\}^{|y|}$,
2. $R_2 = \{0,1\}^{t+O(\log(|x|+|y|))+1}$,
3. left degree $= 2^{d_2}$, $d_2 = O(\log |y|)$,
4. G_2 is explicit.

We build $G = (L, R, E \subseteq L \times R) = G_1 \times G_2$ in the obvious way:
- $L = L_1 \times L_2$,
- $R = R_1 \times R_2$,

$(x, y), (p, q) \in E$ iff $[(x, p) \in E_1$ and $(y, q) \in E_2$

We view R organized into clusters:
- one cluster for each $p \in R_1$
- each cluster is a copy of R_2.

x_{reduced}= keep the first s bits of x, and fill with $(|x| - s)$ zeroes.

Matching process

- Enumerate
  $A = \{(x', y') \in \{0,1\}^{|x|} \times \{0,1\}^{|y|} \mid C(x' \mid y') \le n_1, C(y' \mid x') \le n_2$.
- When (x', y') is enumerated ...
- Step 1. Select at random the r-th neighbor of x'_{reduced} in G_1; this is $p_{x',r}$.
- Step 2. We say that y' makes a request to cluster $p_{x',r}$. If y' has not made a request before to cluster $p_{x',r}$, take $q_{y'}$ to be first unused node in the cluster (if there is one).
  (x', y') is matched to $(p_{x',r}, q_{y'})$.

Claim

With probability $1 - 2\delta$, (x, y) finds a match.

Ignoring some minor technical details, this ends the proof (as in the overview).

Proof of Claim (sketch)

- x_{reduced} is a rich owner in G_1 with respect to $\{x'_{\mathrm{reduced}} \mid x' \in \{0,1\}^{|x|}\}$ (otherwise $C(x_{\mathrm{reduced}})$ small, so $C(x)$ small, contradiction).
- So at most 2^{n-s} strings x' can be matched to $p_{x,r}$ (namely, those x' that have the same reduced form as x_{reduced}).
- At most $2^{n-s} \cdot 2^{n_2}$ strings y' make a request to cluster $p_{x,r}$ (because if (x', y') makes a request then $C(y' \mid x') \leq n_2$).
- $s + t \geq C(x, y) \geq C(x) + C(y \mid x) - O(\log(|x| + |y|)) = n + n_2 - O(\log(|x| + |y|))$.
- So the number of requests is at most $2^{n-s} \cdot 2^{n_2} \leq 2^{t+O(\log(|x|+|y|)}$.
- Since G_2 satisfies these many requests, the first request made by any y' is satisfied.

Proof of Claim (sketch)-cont.

- So the first request (x', y) is satisfied.
- We show that with probability $1 - \delta$, $x' = x$. This implies that (x, y) finds a match, and we're done.
- Suppose $x' \neq x$.
- $C(x \mid y) \leq n_1$ (hypothesis) and $C(x' \mid y) \leq n_1$ (because $(x', y) \in A$).
- x, x' share $p_{x,r}$; so they also share the n_1-prefix of $p_{x',r}$ in the n_1-level of G_1.
- So, either x is a poor owner w.r.t. $B = \{u \mid C(u \mid y) \leq n_1\}$, but then $C(x \mid y) \leq n_1$, FALSE,
- or x is a rich owner, and the node was chosen among those few neighbors that are shared- this happens with probability at most δ.

Thank you.