On efficient compression at almost minimum description length

Marius Zimand

Towson University

2016 Capital Area Theory Day, May 26, 2016

In praise of short descriptions

Aristotle: *Nature operates in the shortest way possible.*

William of Ockham: *"Entia non sunt multiplicanda praeter necessitatem." (Entities must not be multiplied beyond necessity. -Occam's razor)*

Galileo: *Nature [...] makes use of the easiest and simplest means for producing her effects.*

Newton: *We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances.*

Minimum Description Length, according to Solomonoff, Kolmogorov, Chaitin

- Fix U, universal Turing machine.

Minimum Description Length, according to Solomonoff, Kolmogorov, Chaitin

- Fix U, universal Turing machine.
- If $U(p) = x$, we say that p is a program (or description) for x.

Minimum Description Length, according to Solomonoff, Kolmogorov, Chaitin

- Fix U, universal Turing machine.
- If $U(p) = x$, we say that p is a program (or description) for x.
- $C_U(x) = \min(|p| \mid p \text{ is a program for } x)$.

Minimum Description Length, according to Solomonoff, Kolmogorov, Chaitin

- Fix U, universal Turing machine.
- If $U(p) = x$, we say that p is a program (or description) for x.
- $C_U(x) = \min(|p| \mid p$ is a program for $x)$.
- For every other TM M, $C_U(x) \leq C_M(x) + \text{const.}$.

Minimum Description Length, according to Solomonoff, Kolmogorov, Chaitin

- Fix U, universal Turing machine.
- If $U(p) = x$, we say that p is a program (or description) for x.
- $C_U(x) = \min(|p| \mid p$ is a program for $x)$.
- For every other TM M, $C_U(x) \leq C_M(x) + \text{const.}$.
- We drop the subscript, and write $C(x)$ - the Kolmogorov complexity of x (MDL of x).

Minimum Description Length, according to Solomonoff, Kolmogorov, Chaitin

- Fix U, universal Turing machine.
- If $U(p) = x$, we say that p is a program (or description) for x.
- $C_U(x) = \min(|p| \mid p \text{ is a program for } x)$.
- For every other TM M, $C_U(x) \leq C_M(x) + \text{const.}$.
- We drop the subscript, and write $C(x)$ - the Kolmogorov complexity of x (MDL of x).
- $C(x) \leq |x| + O(1)$, for every x.

Minimum Description Length, according to Solomonoff, Kolmogorov, Chaitin

- Fix U, universal Turing machine.
- If $U(p) = x$, we say that p is a program (or description) for x.
- $C_U(x) = \min(|p| \mid p$ is a program for $x)$.
- For every other TM M, $C_U(x) \leq C_M(x) + \text{const.}$.
- We drop the subscript, and write $C(x)$ - the Kolmogorov complexity of x (MDL of x).
- $C(x) \leq |x| + O(1)$, for every x.
- A program p for x with $|p| = C(x)$ is a shortest program for x.

Minimum Description Length, according to Solomonoff, Kolmogorov, Chaitin

- Fix U, universal Turing machine.
- If $U(p) = x$, we say that p is a program (or description) for x.
- $C_U(x) = \min(|p| \mid p \text{ is a program for } x)$.
- For every other TM M, $C_U(x) \leq C_M(x) + \text{const.}$.
- We drop the subscript, and write $C(x)$ - the Kolmogorov complexity of x (MDL of x).
- $C(x) \leq |x| + O(1)$, for every x.
- A program p for x with $|p| = C(x)$ is a shortest program for x.
- A program p for x with $|p| \leq C(x) + c$ is a c-short program for x.

Compression at MDL

- Given x, can we compute a shortest program for x?

Compression at MDL

- Given x, can we compute a shortest program for x?
- NO.

Compression at MDL

- Given x, can we compute a shortest program for x?
- NO.
- Given x and $C(x)$; we can compute a shortest program for x by exhaustive search.

Compression at MDL

- Given x, can we compute a shortest program for x?
- NO.
- Given x and $C(x)$; we can compute a shortest program for x by exhaustive search.
- The running time is larger than any computable function.

Compression at MDL

- Given x, can we compute a shortest program for x?
- NO.
- Given x and $C(x)$; we can compute a shortest program for x by exhaustive search.
- The running time is larger than any computable function.

Theorem (Bauwens, Z., 2014)

Let $t(n)$ be a computable function. If an algorithm on input $(x, C(x))$ computes in time $t(n)$ a program p for x, then $|p| = C(x) + \Omega(n)$ for infinitely many x. (where $n = |x|$).

Compression at MDL if we allow some small error probability

Theorem (Bauwens, Z., 2014)

There exists a probabilistic polynomial time algorithm E such that for all n-bit strings x, for all $\epsilon > 0$,

1. *E on input $x, C(x)$ and $1/\epsilon$, outputs a string p of length $\leq C(x) + \log^2(n/\epsilon)$,*
2. *p is a program for x with probability $1 - \epsilon$.*

- So, finding a short program for x, given x and $C(x)$, can be done in probabilistic poly. time, but any deterministic algorithm takes time larger than any computable function!
- Decompression (reconstructing x from p) cannot run in polynomial time, when compression is done at minimum description length (or close to it).

Relaxing the promise

- The promise that the compressor knows $C(x)$ is quite demanding.
- But it's enough if the compressor knows only an upper bound $k \geq C(x)$.

Theorem (Z.,2016)

There exists a probabilistic polynomial time algorithm E such that for all n-bit strings x, for all $\epsilon > 0$,

1. *E on input x, ~~$C(x)$~~ k and $1/\epsilon$, outputs a string p of length $\leq$ ~~$C(x)$~~ $k + \log^3(n/\epsilon)$,*
2. *p is a program for x with probability $1 - \epsilon$, provided $k \geq C(x)$.*

A surprising relativization

- Suppose Alice wants to send x to Bob, who has y. How many bits does Alice need to send?
- Think that x is the updated version of a file, y is the old version. If Alice knows y, she can send `diff(x,y)`.

A surprising relativization

- Suppose Alice wants to send x to Bob, who has y. How many bits does Alice need to send?
- Think that x is the updated version of a file, y is the old version. If Alice knows y, she can send `diff(x,y)`.
- But suppose Alice does not know y.

A surprising relativization

- Suppose Alice wants to send x to Bob, who has y. How many bits does Alice need to send?
- Think that x is the updated version of a file, y is the old version. If Alice knows y, she can send `diff(x,y)`.
- But suppose Alice does not know y.
- It's possible to compress x to almost MDL conditioned by y, without knowing y.

A surprising relativization

- Suppose Alice wants to send x to Bob, who has y. How many bits does Alice need to send?
- Think that x is the updated version of a file, y is the old version. If Alice knows y, she can send `diff(x,y)`.
- But suppose Alice does not know y.
- It's possible to compress x to almost MDL conditioned by y, without knowing y.

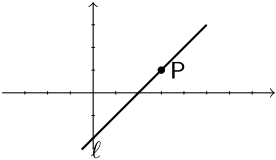
Theorem

There exist algorithms E and D such that E runs in probabilistic poly. time and for all n-bit strings x and y, for all $\epsilon > 0$,

1. *E on input x, k and $1/\epsilon$, outputs a string p of length $\leq k + \log^3(n/\epsilon)$,*
2. *D on input p, y outputs x with probability $1 - \epsilon$, provided $k \geq C(x \mid y)$.*

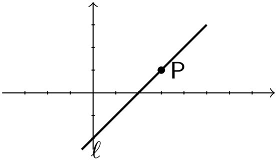
Distributed compression of correlated sources

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.
- ℓ : $2n$ bits of information (intercept, slope in GF[2^n]).
- P : $2n$ bits of information (the 2 coord. in GF[2^n]).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.



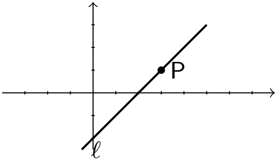
Distributed compression of correlated sources

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.
- ℓ : $2n$ bits of information (intercept, slope in GF[2^n]).
- P : $2n$ bits of information (the 2 coord. in GF[2^n]).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.



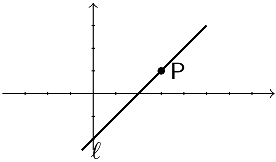
Distributed compression of correlated sources

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.
- ℓ : $2n$ bits of information (intercept, slope in GF[2^n]).
- P : $2n$ bits of information (the 2 coord. in GF[2^n]).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.



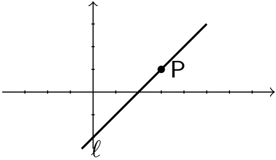
Distributed compression of correlated sources

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.
- ℓ : $2n$ bits of information (intercept, slope in $GF[2^n]$).
- P : $2n$ bits of information (the 2 coord. in $GF[2^n]$).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.



Distributed compression of correlated sources

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.
- ℓ : $2n$ bits of information (intercept, slope in GF[2^n]).
- P : $2n$ bits of information (the 2 coord. in GF[2^n]).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.

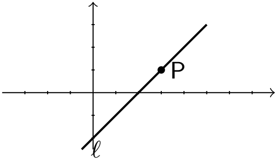


- QUESTION 1: Can Alice send $2n$ bits, and Bob n bits? Yes, of course. But is it just because of the simple geometric relation between ℓ and P?

  Ans: We have seen that it works for any x, y with the complexity profile $C(x) = 2n, C(y) = 2n, C(x \mid y) = n$.

Distributed compression of correlated sources

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.
- ℓ : $2n$ bits of information (intercept, slope in GF[2^n]).
- P : $2n$ bits of information (the 2 coord. in GF[2^n]).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.



- QUESTION 2: Can Alice send $1.5n$ bits, and Bob $1.5n$ bits? Can Alice send $1.74n$ bits, and Bob $1.26n$ bits?

  Ans: Yes (essentially, ... there is a $\mathrm{polylog}(n)$ overhead.) And it works for any x, y with the given complexity profile.

Kolmogorov complexity version of the Slepian-Wolf Theorem- 2 sources

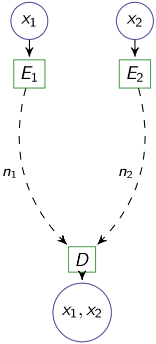
Theorem

There exist probabilistic poly.-time algorithms E_1, E_2 *and algorithm* D *such that for all integers* n_1, n_2 *and* n*-bit strings* x_1, x_2,

if $n_1 + n_2 \geq C(x_1, x_2)$, $n_1 \geq C(x_1 \mid x_2)$, $n_2 \geq C(x_2 \mid x_1)$,

then

- E_i *on input* (x_i, n_i) *outputs a string* p_i *of length* $n_i + O(\log^3 n)$, *for* $i = 1, 2$,
- D *on input* (p_1, p_2) *outputs* (x_1, x_2) *with probability* $1 - 1/n$.



There is an analogous version for any constant number of sources.

One proof sketch

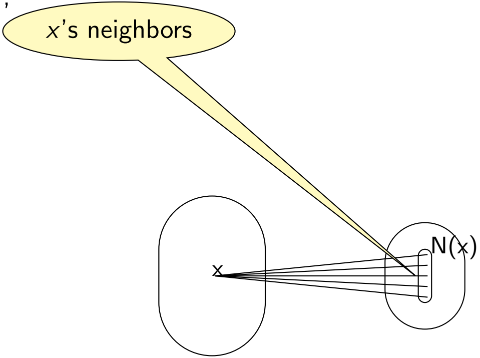
Theorem (Z.,2016)

There exists a probabilistic polynomial-time algorithm E such that for all n-bit strings x, for all $\epsilon > 0$,

1. *E on input x, k and $1/\epsilon$, outputs a string p of length $\leq k + \log^3(n/\epsilon)$,*
2. *p is a program for x with probability $1 - \epsilon$, provided $k \geq C(x)$.*

Graphs with the rich owner property

Bipartite graph G, with left degree D;
parameters k, δ;



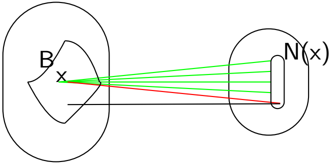
Graphs with the rich owner property

Bipartite graph G, with left degree D;
parameters k, δ;

x is a rich owner w.r.t B if

small regime case: $|B| \leq 2^k$
x owns $(1 - \delta)$ of $N(x)$

large regime case: $|B| \geq 2^k$
then x bla bla bla...not used here
(but used in the Slepian-Wolf theorem).



Graphs with the rich owner property

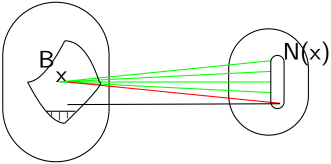
Bipartite graph G, with left degree D;
parameters k, δ;

x is a rich owner w.r.t B if

small regime case: $|B| \leq 2^k$
x owns $(1-\delta)$ of $N(x)$

large regime case: $|B| \geq 2^k$
then x bla bla bla...not used here
(but used in the Slepian-Wolf theorem).

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



Graphs with the rich owner property

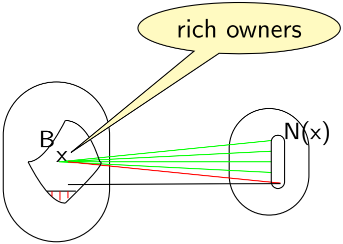
Bipartite graph G, with left degree D;
parameters k, δ;

x is a rich owner w.r.t B if

small regime case: $|B| \leq 2^k$
x owns $(1 - \delta)$ of $N(x)$

large regime case: $|B| \geq 2^k$
then x bla bla bla...not used here
(but used in the Slepian-Wolf theorem).

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



Graphs with the rich owner property

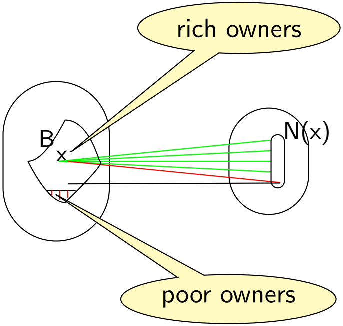
Bipartite graph G, with left degree D;
parameters k, δ;

x is a rich owner w.r.t B if

small regime case: $|B| \leq 2^k$
x owns $(1-\delta)$ of $N(x)$

large regime case: $|B| \geq 2^k$
then x bla bla bla...not used here
(but used in the Slepian-Wolf theorem).

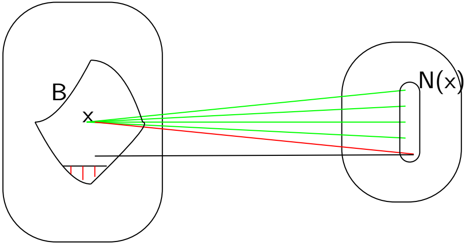
G has the (k,δ) rich owner property:
$\forall B \subseteq L$,
all nodes in B except at most $\delta \cdot |B|$ are rich owners w.r.t. B



Theorem (based on the (Raz-Reingold-Vadhan 2002) extractor)

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

- $L = \{0,1\}^n$
- $R = \{0,1\}^{k+O(\log^3(n/\delta))}$
- $D(\textit{left degree}) = 2^{O(\log^3(n/\delta))}$



Proof sketch (cont. 2)

- Let x be an n-bit string, and $k \geq C(x)$,

Proof sketch (cont. 2)

- Let x be an n-bit string, and $k \geq C(x)$,
- **Compression of** x**.** Consider G with $(k+1, \delta)$-rich owner property. Pick p a random neighbor of x (viewed as a left node).
  $|p| = k + O(\log^3(n/\delta))$.
  Also compute a fingerprint $h(x)$ of length $O(\log(n/\delta))$ that with prob. $1-\delta$ isolates x from any n strings of length n.

Proof sketch (cont. 2)

- Let x be an n-bit string, and $k \geq C(x)$,
- **Compression of x.** Consider G with $(k+1, \delta)$-rich owner property. Pick p a random neighbor of x (viewed as a left node).
  $|p| = k + O(\log^3(n/\delta))$.
  Also compute a fingerprint $h(x)$ of length $O(\log(n/\delta))$ that with prob. $1 - \delta$ isolates x from any n strings of length n.
- **Decompression**. We reconstruct x from p and $h(x)$.

Proof sketch (cont. 2)

- Let x be an n-bit string, and $k \geq C(x)$,
- **Compression of** x**.** Consider G with $(k+1, \delta)$-rich owner property. Pick p a random neighbor of x (viewed as a left node).
  $|p| = k + O(\log^3(n/\delta))$.
  Also compute a fingerprint $h(x)$ of length $O(\log(n/\delta))$ that with prob. $1-\delta$ isolates x from any n strings of length n.
- **Decompression**. We reconstruct x from p and $h(x)$.
- Take $B = \{u \mid C(u) \leq C(x)\}$.

Proof sketch (cont. 2)

- Let x be an n-bit string, and $k \geq C(x)$,
- **Compression of x.** Consider G with $(k+1, \delta)$-rich owner property. Pick p a random neighbor of x (viewed as a left node).
  $|p| = k + O(\log^3(n/\delta))$.
  Also compute a fingerprint $h(x)$ of length $O(\log(n/\delta))$ that with prob. $1-\delta$ isolates x from any n strings of length n.
- **Decompression**. We reconstruct x from p and $h(x)$.
- Take $B = \{u \mid C(u) \leq C(x)\}$.
- $|B| < 2^{k+1}$, so B is in the small regime case.

Proof sketch (cont. 2)

- Let x be an n-bit string, and $k \geq C(x)$,
- **Compression of x.** Consider G with $(k+1, \delta)$-rich owner property. Pick p a random neighbor of x (viewed as a left node).
  $|p| = k + O(\log^3(n/\delta))$.
  Also compute a fingerprint $h(x)$ of length $O(\log(n/\delta))$ that with prob. $1-\delta$ isolates x from any n strings of length n.
- **Decompression**. We reconstruct x from p and $h(x)$.
- Take $B = \{u \mid C(u) \leq C(x)\}$.
- $|B| < 2^{k+1}$, so B is in the small regime case.
- The set of poor owners w.r.t B has size bounded by $\delta|B| \leq \delta 2^{C(x)+1}$.

Proof sketch (cont. 2)

- Let x be an n-bit string, and $k \geq C(x)$,
- **Compression of** x**.** Consider G with $(k+1, \delta)$-rich owner property. Pick p a random neighbor of x (viewed as a left node).
  $|p| = k + O(\log^3(n/\delta))$.
  Also compute a fingerprint $h(x)$ of length $O(\log(n/\delta))$ that with prob. $1 - \delta$ isolates x from any n strings of length n.
- **Decompression**. We reconstruct x from p and $h(x)$.
- Take $B = \{u \mid C(u) \leq C(x)\}$.
- $|B| < 2^{k+1}$, so B is in the small regime case.
- The set of poor owners w.r.t B has size bounded by $\delta|B| \leq \delta 2^{C(x)+1}$.
- Since the poor owners can be enumerated, a poor owner u has complexity bounded by

$$\begin{aligned} C(u) &\leq C(x) - \log(1/\delta) + 2\log C(x) + O(1) \\ &< C(x). \end{aligned}$$

Proof sketch (cont. 2)

- Let x be an n-bit string, and $k \geq C(x)$,
- **Compression of x.** Consider G with $(k+1, \delta)$-rich owner property. Pick p a random neighbor of x (viewed as a left node).
  $|p| = k + O(\log^3(n/\delta))$.
  Also compute a fingerprint $h(x)$ of length $O(\log(n/\delta))$ that with prob. $1-\delta$ isolates x from any n strings of length n.
- **Decompression**. We reconstruct x from p and $h(x)$.
- Take $B = \{u \mid C(u) \leq C(x)\}$.
- $|B| < 2^{k+1}$, so B is in the small regime case.
- The set of poor owners w.r.t B has size bounded by $\delta|B| \leq \delta 2^{C(x)+1}$.
- Since the poor owners can be enumerated, a poor owner u has complexity bounded by

$$\begin{aligned} C(u) &\leq C(x) - \log(1/\delta) + 2\log C(x) + O(1) \\ &< C(x). \end{aligned}$$

- So, x is a rich owner w.r.t. B.

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.
  2. p has a single neighbor with complexity $C(x)$, namely x.

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.
  2. p has a single neighbor with complexity $C(x)$, namely x.
  3. but p may have many neighbors with complexity $> C(x)$.

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.
  2. p has a single neighbor with complexity $C(x)$, namely x.
  3. but p may have many neighbors with complexity $> C(x)$.
- For each $j = 1, \ldots, k$, we want to find the first program q of length j s.t. $x' = U(q)$ is a neighbor of p, and make a list with the x's.

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.
  2. p has a single neighbor with complexity $C(x)$, namely x.
  3. but p may have many neighbors with complexity $> C(x)$.
- For each $j = 1, \ldots, k$, we want to find the first program q of length j s.t. $x' = U(q)$ is a neighbor of p, and make a list with the x's.
- Such a list can be enumerated.

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.
  2. p has a single neighbor with complexity $C(x)$, namely x.
  3. but p may have many neighbors with complexity $> C(x)$.
- For each $j = 1, \ldots, k$, we want to find the first program q of length j s.t. $x' = U(q)$ is a neighbor of p, and make a list with the x's.
- Such a list can be enumerated.
- x is on the list.

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.
  2. p has a single neighbor with complexity $C(x)$, namely x.
  3. but p may have many neighbors with complexity $> C(x)$.
- For each $j = 1, \ldots, k$, we want to find the first program q of length j s.t. $x' = U(q)$ is a neighbor of p, and make a list with the x's.
- Such a list can be enumerated.
- x is on the list.
- The list may contain $\leq n$ other strings (at most one at each complexity level larger than $C(x)$).

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.
  2. p has a single neighbor with complexity $C(x)$, namely x.
  3. but p may have many neighbors with complexity $> C(x)$.
- For each $j = 1, \ldots, k$, we want to find the first program q of length j s.t. $x' = U(q)$ is a neighbor of p, and make a list with the x's.
- Such a list can be enumerated.
- x is on the list.
- The list may contain $\leq n$ other strings (at most one at each complexity level larger than $C(x)$).
- Using the fingerprint $h(x)$, the decompressor distinguishes x from the other strings, and halts the enumeration when some enumerated string has the right fingerprint. This must be x, with high probability.

Proof sketch (cont. 3)

- So, with prob. $1 - \delta$:
  1. p does not have neighbors with complexity $< C(x)$.
  2. p has a single neighbor with complexity $C(x)$, namely x.
  3. but p may have many neighbors with complexity $> C(x)$.
- For each $j = 1, \ldots, k$, we want to find the first program q of length j s.t. $x' = U(q)$ is a neighbor of p, and make a list with the x's.
- Such a list can be enumerated.
- x is on the list.
- The list may contain $\leq n$ other strings (at most one at each complexity level larger than $C(x)$).
- Using the fingerprint $h(x)$, the decompressor distinguishes x from the other strings, and halts the enumeration when some enumerated string has the right fingerprint. This must be x, with high probability.
- q.e.d.

Thank you.

References:

B. Bauwens, M. Zimand, Linear list approximation for short programs (or the power of a few random bits), CCC 2014 (and ECCC TR15-017).

M. Zimand, Kolmogorov complexity version of Slepian-Wolf coding, arXiv:1511.03602.

J. Teutsch, M. Zimand, A brief on short descriptions, SIGACT News, 47(1):42-67, March 2016,