Entropy Compression Method

František Kardoš
LaBRI, Université de Bordeaux

Introduction

Entropy compression method

- analyze the performance of randomized algorithms
- prove that the algorithm eventually finds a solution

Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

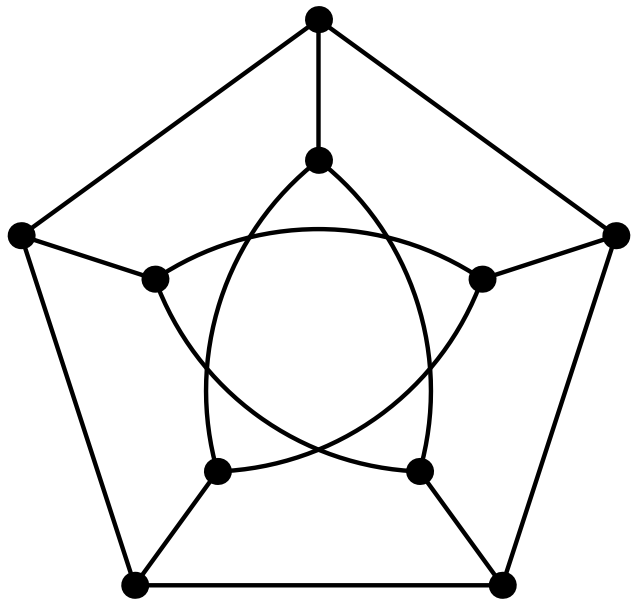
is acyclic if there is no bichromatic cycle.

Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

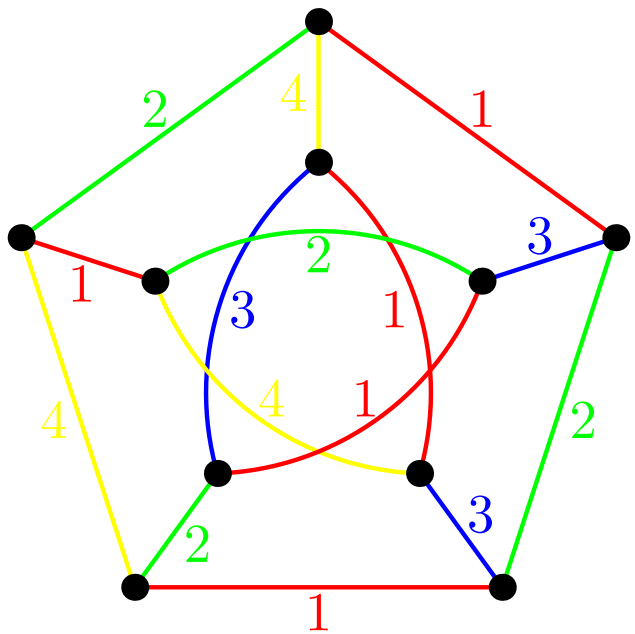


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

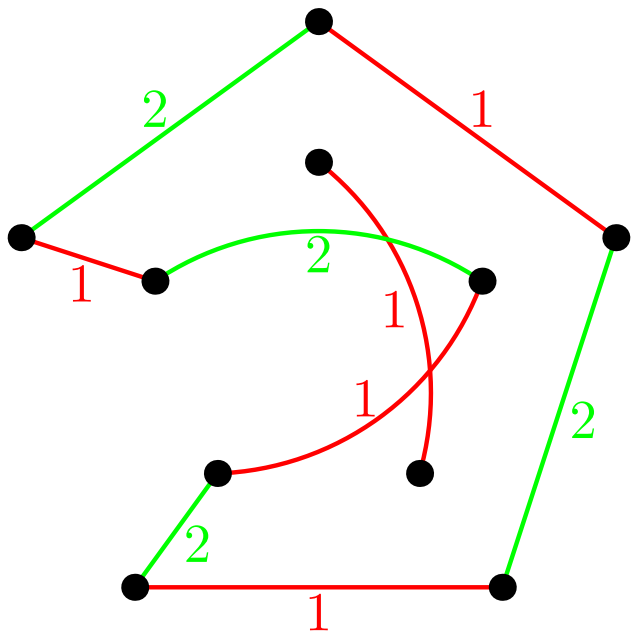


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

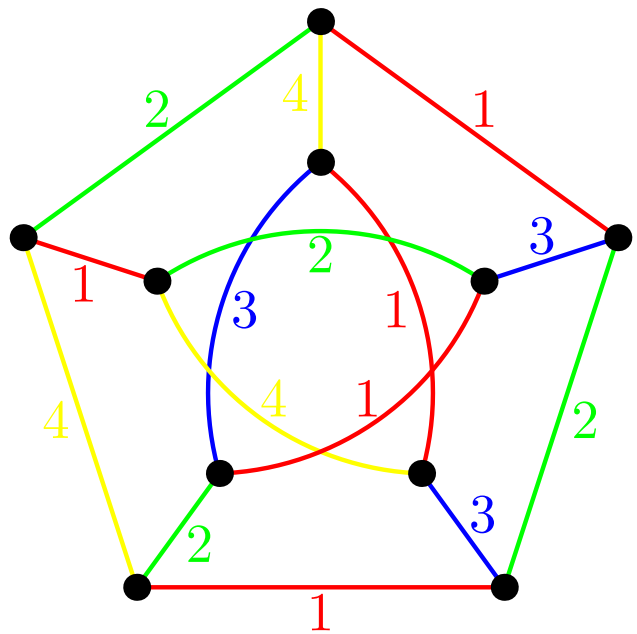


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

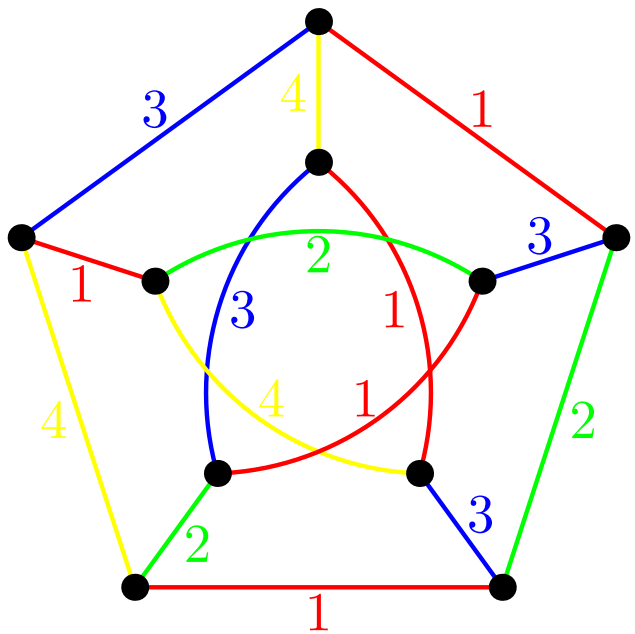


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

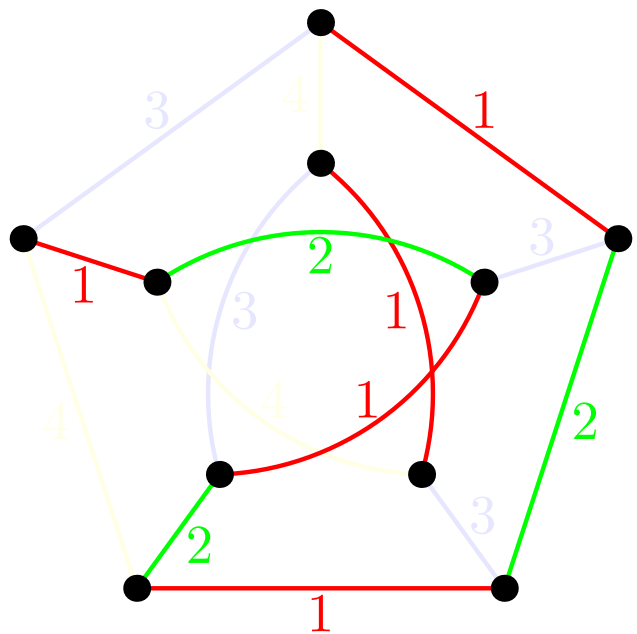


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

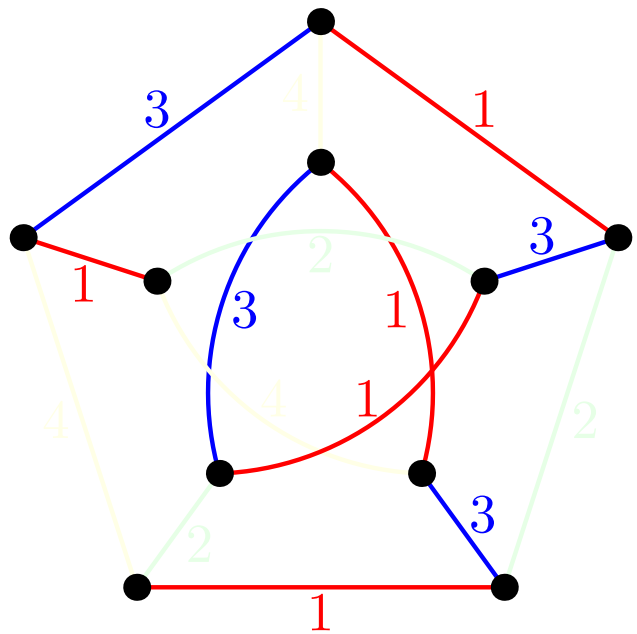


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

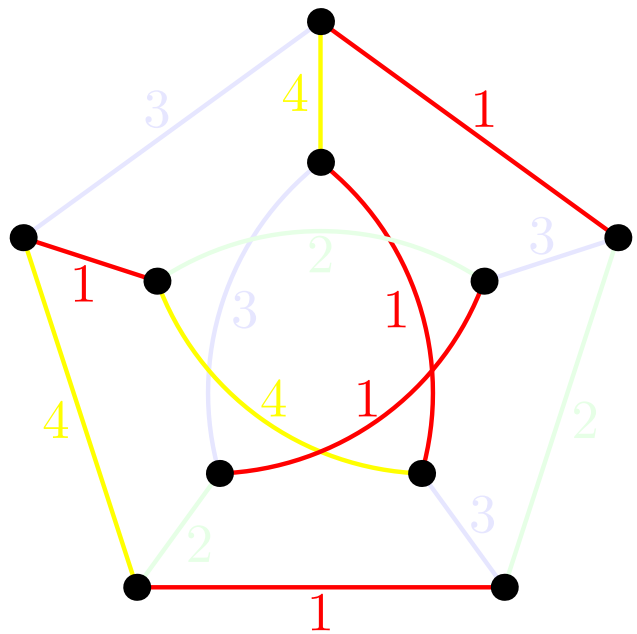


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

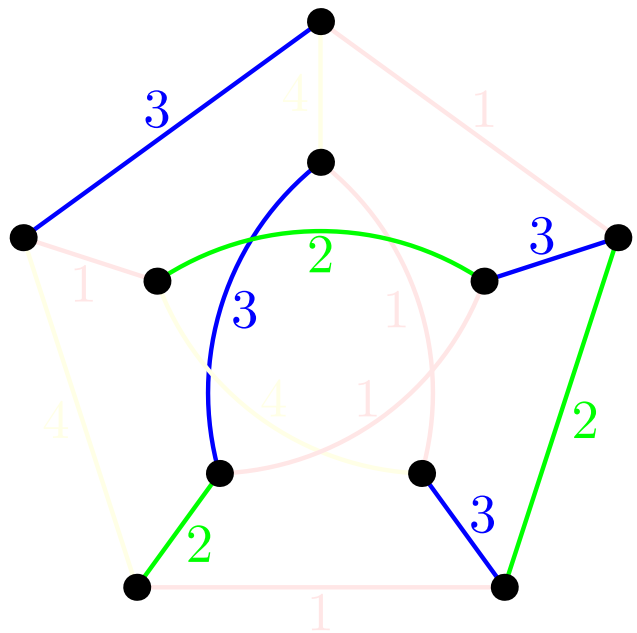


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

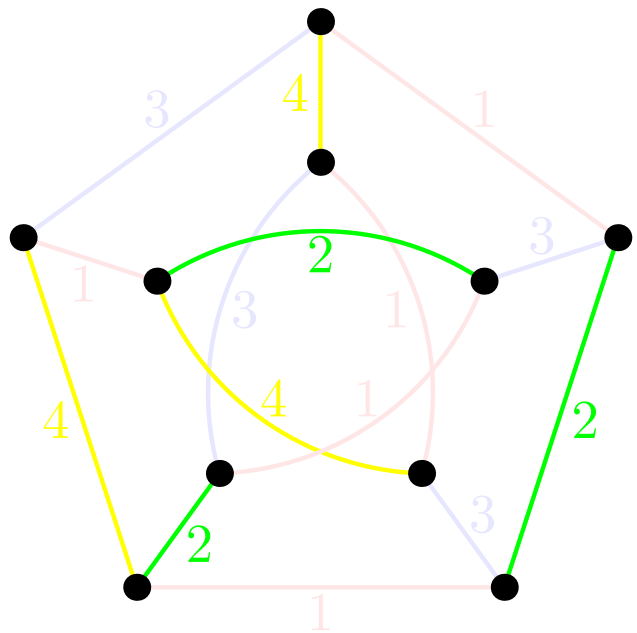


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

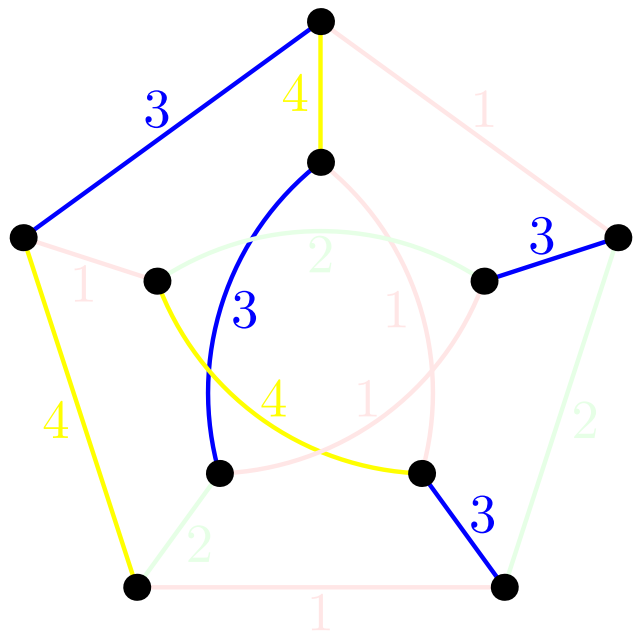


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.

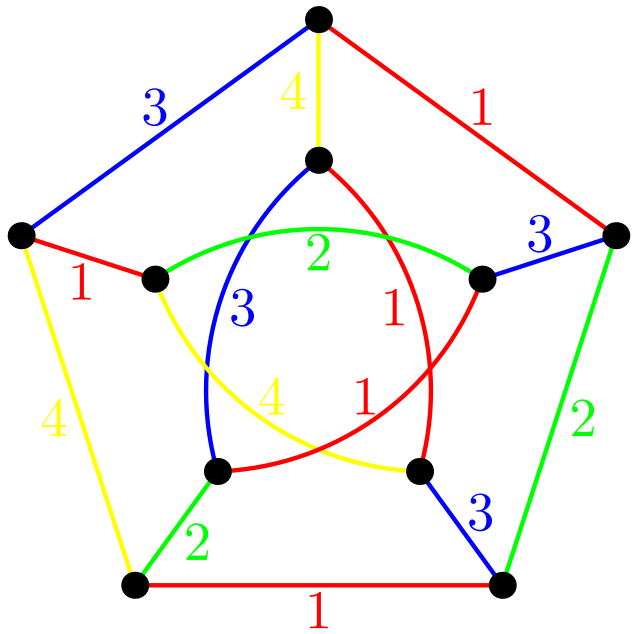


Acyclic edge coloring

Let G be a graph. A (proper) edge coloring

$$\varphi : E(G) \to [1, k]$$

is acyclic if there is no bichromatic cycle.



Acyclic edge coloring

The smallest k such that G admits an acyclic edge coloring with k colors is the acyclic chromatic index of G, denoted $\chi'_a(G)$.

Acyclic edge coloring

The smallest k such that G admits an acyclic edge coloring with k colors is the acyclic chromatic index of G, denoted $\chi'_a(G)$.

Clearly, for every graph G,

$$\chi'_a(G) \geq \chi'(G) \geq \Delta(G)$$

where $\chi(G)$ is the chromatic index of G and $\Delta(G)$ is the maximum degree of G.

Acyclic edge coloring

The smallest k such that G admits an acyclic edge coloring with k colors is the acyclic chromatic index of G, denoted $\chi'_a(G)$.

Clearly, for every graph G,

$$\chi'_a(G) \geq \chi'(G) \geq \Delta(G)$$

where $\chi(G)$ is the chromatic index of G and $\Delta(G)$ is the maximum degree of G.

For Petersen graph P we have

$$\chi'_a(P) = \chi'(P) = 4.$$

Upper bounds: conjectured and known

Theorem (Vizing 1964)

$\chi'(G) \leq \Delta + 1$.

Upper bounds: conjectured and known

Theorem (Vizing 1964)

$\chi'(G) \leq \Delta + 1$.

Conjecture (Fiamčík 1978, Alon et al. 2001)

$\chi'_a(G) \leq \Delta + 2$.

Upper bounds: conjectured and known

Theorem (Vizing 1964)

$\chi'(G) \leq \Delta + 1.$

Conjecture (Fiamčík 1978, Alon et al. 2001)

$\chi'_a(G) \leq \Delta + 2.$

Theorem (greedy algorithm)

$\chi'_a(G) \leq 2\Delta(\Delta - 1) + 1.$

Upper bounds: conjectured and known

Theorem (Vizing 1964)

$\chi'(G) \leq \Delta + 1.$

Conjecture (Fiamčík 1978, Alon et al. 2001)

$\chi'_a(G) \leq \Delta + 2.$

Theorem (greedy algorithm)

$\chi'_a(G) \leq 2\Delta(\Delta - 1) + 1.$

Theorem (Molloy and Reed 1998)

$\chi'_a(G) \leq 16\Delta.$

Upper bounds: conjectured and known

Theorem (Vizing 1964)

$\chi'(G) \leq \Delta + 1.$

Conjecture (Fiamčík 1978, Alon et al. 2001)

$\chi'_a(G) \leq \Delta + 2.$

Theorem (greedy algorithm)

$\chi'_a(G) \leq 2\Delta(\Delta - 1) + 1.$

Theorem (Molloy and Reed 1998)

$\chi'_a(G) \leq 16\Delta.$

Theorem (Esperet and Parreau 2018)

$\chi'_a(G) \leq 4\Delta - 4.$

Randomized algorithm

Let G be a graph, let $e_1, e_2, \ldots, e_m$ be the edges of G.
Let C be a set of colors, let $|C| = K$.

Randomized algorithm

Let G be a graph, let $e_1, e_2, \ldots, e_m$ be the edges of G.
Let C be a set of colors, let $|C| = K$.

For a partially colored graph G and an edge e, let $F(e)$ denote the set of forbidden colors for e: the colors on the edges adjacent to e.

Randomized algorithm

Let G be a graph, let $e_1, e_2, \ldots, e_m$ be the edges of G.
Let C be a set of colors, let $|C| = K$.

For a partially colored graph G and an edge e, let $F(e)$ denote the set of forbidden colors for e: the colors on the edges adjacent to e.

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the first two

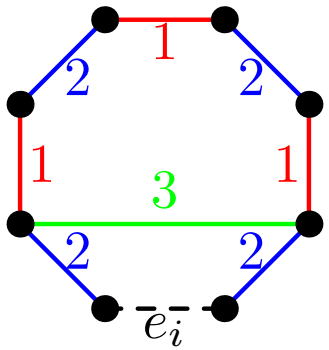
until the whole graph is colored.

Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

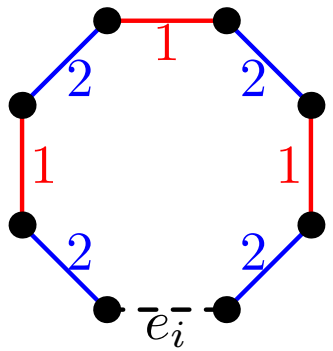


Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

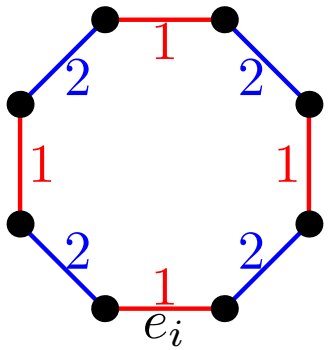


Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

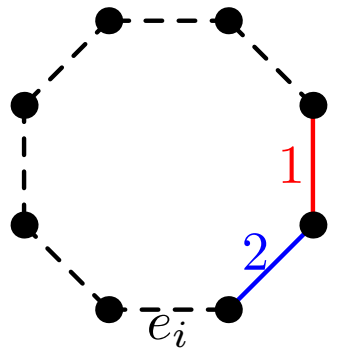


Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.



Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

Does the algorithm ever stop?

Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

Does the algorithm ever stop?

Suppose not. Consider all the possible runs of the algorithm.

Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

Does the algorithm ever stop?

Suppose not. Consider all the possible runs of the algorithm. For any N, each run does not stop even after N rounds.

Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

Does the algorithm ever stop?

Suppose not. Consider all the possible runs of the algorithm.
For any N, each run does not stop even after N rounds.
Let's try to encode a run in a log file.

Randomized algorithm

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

Does the algorithm ever stop?

Suppose not. Consider all the possible runs of the algorithm. For any N, each run does not stop even after N rounds. Let's try to encode a run in a log file. We need to encode

- the new color for e_i
- (eventually) the path to uncolor.

Entropy compression principle

Let the number of rounds N be fixed.

Entropy compression principle

Let the number of rounds N be fixed.

If at each round, we can choose one of (at least) k colors, then there are at least k^N different runs, none of which succeeds to color the whole graph.

Entropy compression principle

Let the number of rounds N be fixed.

If at each round, we can choose one of (at least) k colors, then there are at least k^N different runs, none of which succeeds to color the whole graph.

On the other hand, given a final coloring after N rounds and the content of the log file, we can reconstruct the whole run.

Entropy compression principle

Let the number of rounds N be fixed.

If at each round, we can choose one of (at least) k colors, then there are at least k^N different runs, none of which succeeds to color the whole graph.

On the other hand, given a final coloring after N rounds and the content of the log file, we can reconstruct the whole run.

If we can prove that the number of possible combinations of $\{\text{final coloring} \times \text{log file}\}$ is in $o(k^N)$, then we get a contradiction: a run that stops before round N must exist.

Log file: which edges are colored and which are not

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

Log file: which edges are colored and which are not

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

If we know the set of colored edges before round j, we know which edge will be colored at round j.

Log file: which edges are colored and which are not

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

If we know the set of colored edges before round j, we know which edge will be colored at round j.
To know the set of colored edges after round j, it suffices to know which (if ever) path is uncolored.

Log file: which edges are colored and which are not

Repeat

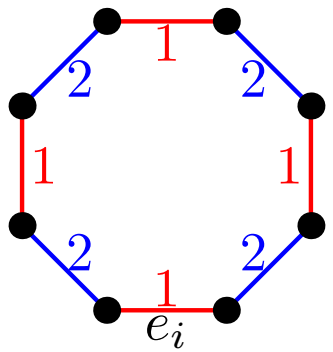
- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

If we know the set of colored edges before round j, we know which edge will be colored at round j.
To know the set of colored edges after round j, it suffices to know which (if ever) path is uncolored.
To encode a path of length ℓ starting at e_i, we can use a word of length ℓ over $[1, \Delta]$.

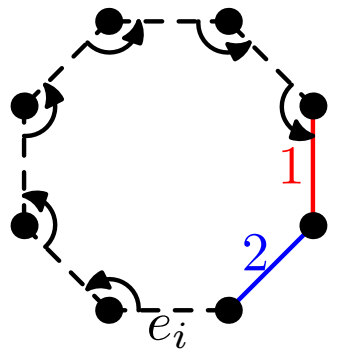
Log file: which edges are colored and which are not

If we know the set of colored edges before round j, we know which edge will be colored at round j.
To know the set of colored edges after round j, it suffices to know which (if ever) path is uncolored.
To encode a path of length ℓ starting at e_i, we can use a word of length ℓ over $[1, \Delta]$.



Log file: which edges are colored and which are not

If we know the set of colored edges before round j, we know which edge will be colored at round j.
To know the set of colored edges after round j, it suffices to know which (if ever) path is uncolored.
To encode a path of length ℓ starting at e_i, we can use a word of length ℓ over $[1, \Delta]$.

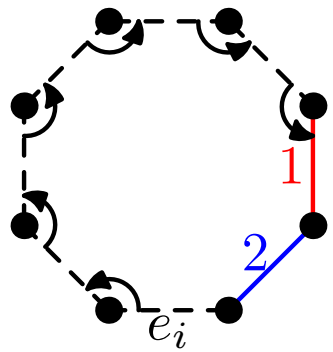


Log file: what else?

Given

- the coloring after round j
- the information whether a path was uncolored or not
- (eventually) the uncolored path

we can determine the coloring before round j.

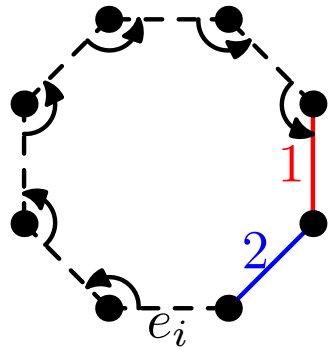


Log file: what else?

Given

- the coloring after round j
- the information whether a path was uncolored or not
- (eventually) the uncolored path

we can determine the coloring before round j.



In particular, we can determine the color assigned to e_i.

Log file

Log file contains

- for each round, a boolean to know whether there was a conflict or not; and eventually
- the number of edges to uncolor, and
- for each uncolored edge, a value from $[1, \Delta]$.

Log file

Log file contains

- for each round, a boolean to know whether there was a conflict or not; and eventually
- the number of edges to uncolor, and
- for each uncolored edge, a value from $[1, \Delta]$.

Alternatively, log file can contain

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.

Log file

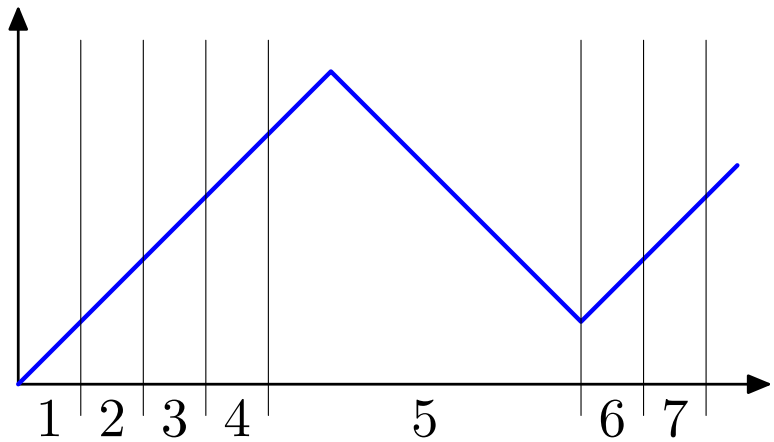
Log file contains

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.

Log file

Log file contains

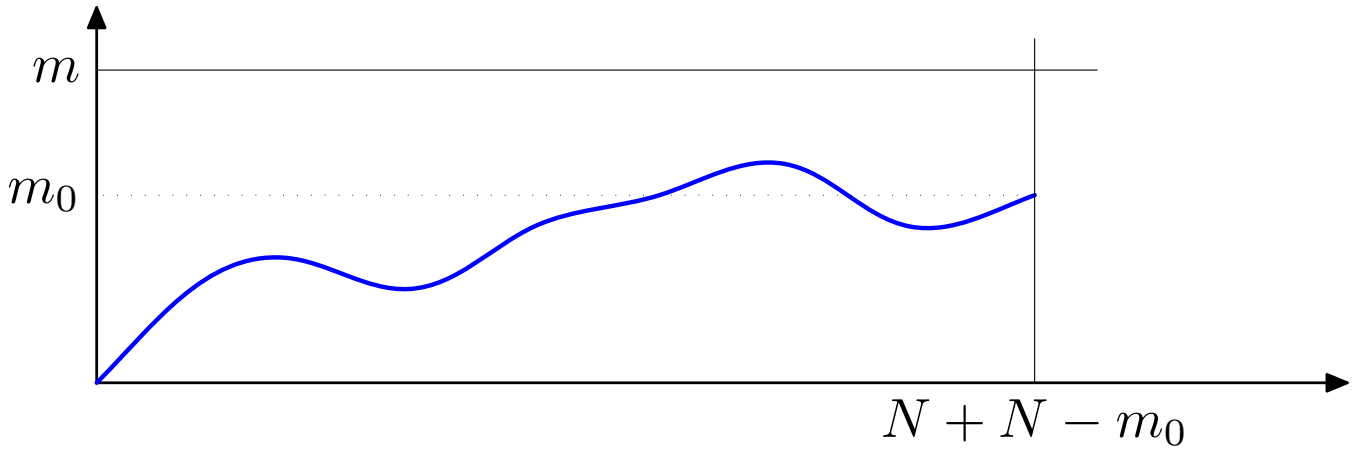
- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.



Log file

Log file contains

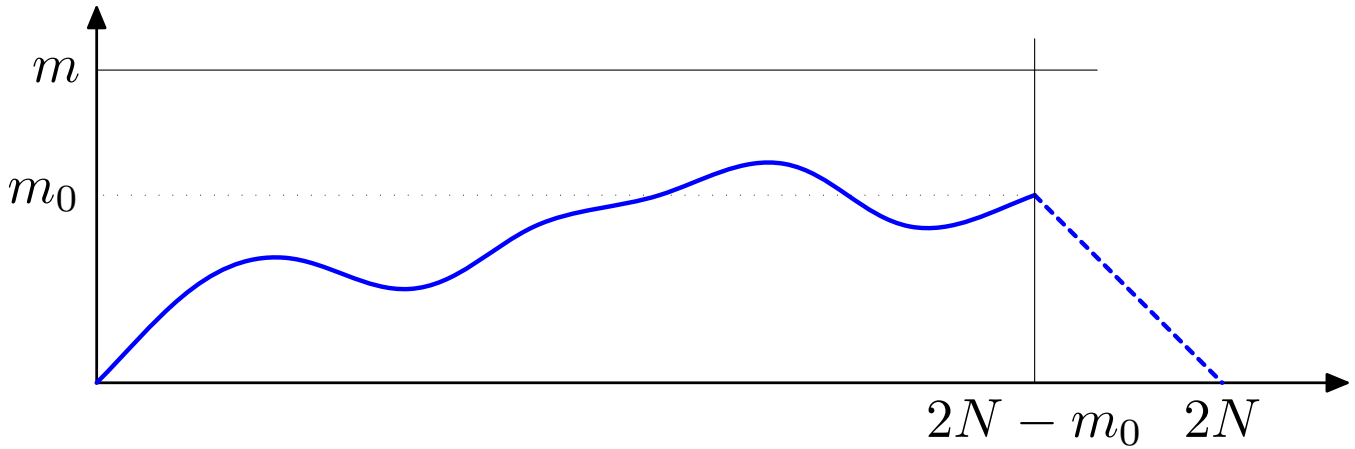
- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.



Log file

Log file contains

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.



Dyck words

A Dyck word of length $2N$ is a word over $\{\nearrow, \searrow\}$ representing a path from $(0, 0)$ to $(2N, 0)$ never crossing the zero line.

Dyck words

A Dyck word of length $2N$ is a word over $\{\nearrow, \searrow\}$ representing a path from $(0,0)$ to $(2N,0)$ never crossing the zero line.

It is known that the number of Dyck words of length $2N$ is the N-th Catalan number

$$C_N = \frac{1}{N+1}\binom{2N}{N} \sim \frac{4^N}{N^{3/2}\sqrt{\pi}}$$

Log file

Log file contains

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.

Log file

Log file contains

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.

How many different log files and different final colorings can there be?

Log file

Log file contains

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.

How many different log files and different final colorings can there be?

Let K be the total number of colors, let m be the number of edges of G. The number of outcomes is at most

Log file

Log file contains

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.

How many different log files and different final colorings can there be?

Let K be the total number of colors, let m be the number of edges of G. The number of outcomes is at most

$$K^m \cdot \frac{4^N}{N^{3/2}\sqrt{\pi}} \cdot \Delta^N$$

Log file

Log file contains

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.

How many different log files and different final colorings can there be?

Let K be the total number of colors, let m be the number of edges of G. The number of outcomes is at most

$$K^m \cdot \frac{4^N}{N^{3/2}\sqrt{\pi}} \cdot \Delta^N = \frac{K^m}{N^{3/2}\sqrt{\pi}} \cdot (4\Delta)^N$$

Log file

Log file contains

- a series of booleans, indicating whether an edge is colored or uncolored; and
- for each uncolored edge, a value from $[1, \Delta]$.

How many different log files and different final colorings can there be?

Let K be the total number of colors, let m be the number of edges of G. The number of outcomes is at most

$$K^m \cdot \frac{4^N}{N^{3/2}\sqrt{\pi}} \cdot \Delta^N = \frac{K^m}{N^{3/2}\sqrt{\pi}} \cdot (4\Delta)^N = o((4\Delta)^N)$$

How many different colorings?

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

How many different colorings?

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

For an edge e_i, there are at most 2Δ forbidden colors.

How many different colorings?

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

For an edge e_i, there are at most 2Δ forbidden colors.

There are at least $(K - 2\Delta)^N$ different runs,

How many different colorings?

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

For an edge e_i, there are at most 2Δ forbidden colors.

There are at least $(K - 2\Delta)^N$ different runs,
but only $o((4\Delta)^N)$ different outcomes.

How many different colorings?

Repeat

- pick the first uncolored edge, say e_i
- choose a random color from $C \setminus F(e_i)$
- if a bicolored cycle appears, uncolor e_i together with all the edges of the cycle but the last two

until the whole graph is colored.

For an edge e_i, there are at most 2Δ forbidden colors.

There are at least $(K - 2\Delta)^N$ different runs,
but only $o((4\Delta)^N)$ different outcomes.

As long as $K \geq 6\Delta$, the algorithm must find a valid coloring.

Conclusion

Entropy compression: the history of a given process can be recorded in an efficient way – the amount of additional information that is recorded at each step of the process is (on average) less than the amount of new information randomly generated at each step.

Conclusion

Entropy compression: the history of a given process can be recorded in an efficient way – the amount of additional information that is recorded at each step of the process is (on average) less than the amount of new information randomly generated at each step.

Used to prove the existence of a solution in various settings: graph colourings, formula satisfiability, combinatorics of words, etc.

Conclusion

Entropy compression: the history of a given process can be recorded in an efficient way – the amount of additional information that is recorded at each step of the process is (on average) less than the amount of new information randomly generated at each step.

Used to prove the existence of a solution in various settings: graph colourings, formula satisfiability, combinatorics of words, etc.

Thank you for your attention!