Applications of Lovász Local Lemma and Entropy Compression
presentation by Julian Marx

1. **Paper**

The first part of the talk is about a textbook chapter by Michael Molloy and Bruce Reed where they introduce Lovász local lemma and its power in showing upper bounds on colourability. It states:

Theorem 1 (Lovász Local Lemma). *Given a set of (bad) events $A_1, ..., A_n$ such that for each $1 \leq i \leq n$:*

1. $Pr(A_i) \leq p < 1$
2. *A_i is mutually independent of a set of all but at most d other events*

If $4pd \leq 1$, then $Pr(\bigcap \overline{A_i}) > 0$.

Theorem 2. *If H is a hypergraph such that each hyperedge has size at least k and intersects at most 2^{k-3} other hyperedges, then H is 2-colourable.*

The idea of the proof is to independently randomly assign each vertex a colour and look at the events

$$A_e := \text{all vertices of } e \text{ have the same colour}$$

with which we get $p = \left(\frac{1}{2}\right)^k$ and $d = 2^{k-3}$ since each A_e only depends on A_f with $e \cap f \neq \emptyset$ by the following principle:

Mutual Independence Principle *Suppose that $X = X_1, \cdots, X_m$ is a sequence of independent random experiments. Suppose further that $A_1, \cdots, A_n$ is a set of events, where each A_i is determined by $F_i \subset X$. If $F_i \cap (F_1, \cdots, F_k) = \emptyset$ then A_i is mutually independent of $A_1, \cdots, A_k$.*

This applies since our random experiments are the colourings of each vertex, so A_e only depends on the set of events e.
Now all LLL tells us that the probability that we have a proper colouring is non-zero implying that there must exist one.

The second application is about list colourings which are defined as follows:

Definition 3. Given a graph $G = (V, E)$, a set of colours C and some list $L(v) \subset C$ for all $v \in V$.
A colouring of G is called list-colouring if every vertex v is assigned a colour in $L(v)$.

Theorem 4. *If $L(v) \geq l$ for each vertex $v \in V$, and each colour is acceptable for at most $\frac{l}{8}$ of the neighbours of any one vertex, then there exists a list-colouring.*

Now we're randomly assigning each vertex a colour from their list (after first reducing all lists to exactly size l) but in this case the event A_e as defined in the previous theorem won't do the trick. However we may look at the event

$$A_{i,e} = \text{both vertices of } e \text{ have the colour i}$$

for each colour i and edge $e = (x, y) \in E$ with $i \in L(x) \cap L(y)$.
Now we can choose $p = \left(\frac{1}{l}\right)^2$ and $d = \frac{l^2}{4}$ since by the mutual independence principle $A_{i,(x,y)}$ only depends on $A_{j,(y,z)}$ with $j \in L(y) \cap L(z)$ (by assumption for each $j \in L(y)$ (l choices), there are at most $\frac{l}{8}$ such z) and $A_{j,(x,z)}$ with $j \in L(x) \cap L(z)$ (analogously limited by $\frac{l^2}{8}$).

2. **Paper**

The second part is about a paper by Louis Esperet and Aline Parreau which in turn is about Entropy Compression. This is a tool to show that a given probabilistic algorithm with a potential endless loop terminates. If the algorithm at every step is determined by a random input, the idea is to keep track of these with a smaller record (containing less information), so that at any point we can reversely determine the previous random inputs from the current state of the algorithm together with the record. Due to information conservation this would imply that some random inputs must actually lead to the algorithm terminating before this point.
With this tool they developed an algorithm finding an acyclic-edge colouring (defined below) using $4(\Delta - 1)$ colours for any graph with maximum degree Δ.

Definition 5. An edge-colouring of a graph G is called acyclic if it is a proper colouring and every circle in G contains at least 3 colours.

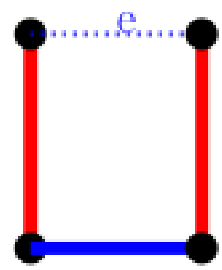
On page 4 figure 1 the reader can find an illustration of this definition.

Theorem 6. *Every graph has an acyclic edge-colouring using at most* $4\Delta - 4$ *colours.*

The algorithm used will be pretty straight forward and just randomly start colouring (using colours not used by neighbours) until a 2-coloured cycle is created. If that happens we're uncolouring the edge last coloured (and thereby deleting all potentially created 2-coloured cycles) and uncolouring all but 2 edges (See figure 2 on page 4 for an illustration).
Since this will maintain an acyclic colouring throughout every step all we need to show is that this algorithm terminates. Here Entropy Compression will come into play.
Further to limit the size of the record we will also make sure no 2-coloured 4-cycles are created any point. Additionally for us to be able to recreate this algorithm from a record, we need to be a bit more deterministic, leading us to put an order on the edges and always pick the smallest uncoloured edge. This leads to the following algorithm:

Given uncoloured graph
while graph not fully coloured **do**
  $e \leftarrow$ min uncoloured edge
  Choose random $F \in [2\Delta - 2]$
  Assign e F-th colour, that is not:
  - used by neighbour
  - forming 2-coloured 4-cycle

  if 2-coloured cycle $ex_2x_3...x_{2k}$ is formed **then**
    Uncolour e as well as $x_4, ..., x_{2k}$
  end if
end while



A key observation is that the numbers of colours that are involved in a conflict (used by a neighbour of e or forming a 4-cycle) is limited by $2(\Delta - 1)$ implying that there actually is an F-th colour. To see this, observe that there are only at most $2(\Delta - 1)$ neighbours and a colour creating a 4-cycle would look like in the picture above. Thus it corresponds to a colour used by exactly 2 neighbours (every colour can only appear once per vertex of e) and thereby we have bijection between colours double counted (by counting neighbours) and colours creating 4-cycles and thus maintaining the bound of $2(\Delta - 1)$.

Now it is time to define the record. We want to keep track of which cycle was created but to keep it as information compact as possible, we're doing it the following way:
Let e_t be the edge coloured at step t.
We're going to define an order on all the cycles of the same length that contain e_t and then our record will be:
$R_t = (k, l)$ if the cycle created is the l-th 2k-cycle containing e_t (on a given ordering)
$R_t = \emptyset$ otherwise
Now the following two lemmas show that our record is enough to retrace the algorithm and find the random inputs. Letting X_t set of uncoloured edges at step t we get the following result:

Lemma 7. At each step t, X_t is completely determined by the record $(R_i)_{i \leq t}$.

This is shown by induction. The idea is that given X_{t-1}, we can easily tell from the record R_t which edges were coloured and which uncoloured. For this it is important to note that we know e_t since $e_t = \min X_{t-1}$.

Now with this we can already prove that we can recreate the random inputs from the record, where Φ_t denotes the partial colouring after step t and F_t the random input at step t:

Lemma 8. Given Φ_t and $(R_i)_{i \leq t}$, we can determine $(F_i)_{i \leq t}$.

Here the idea is to determine Φ_{t-1} so that we can use induction and only need to find F_t.
Again the record R_t will tell us the necessary information. Since if a cycle was created and deleted, the two remaining colours give us the full colouring C previously had.

Now this lemma gives us a surjection from the possible records and partial colourings at the end (There is at most $(4(\Delta - 1) + 1)^{|E|}$ since every edge can be coloured in $4(\Delta - 1)$ ways or still be uncoloured) to the possible random inputs that are not yet terminated. So this implies that there is less of the latter, i.e.:

Corollary 9. #inputs not terminated at step $t \leq (4(\Delta-1)+1)^m$#possible records$(R_i)_{i\leq t}$

Knowing this the goal now is to limit the right hand side such that we get #inputs not terminated at step $t < (2(\Delta-1))^t$ for some t. Meaning at least one combination of inputs must have terminated (since in total there are $(2(\Delta-1))^t$ combinations of inputs) and have given us the existence of an acyclic colouring.

To do this we're going to relate the record to Dyck words:

Definition 10. A **partial Dyck word** is a word on the alphabet $\{0,1\}$ such that any prefix contains at least as many 0s as 1s.
A **Dyck word** is a partial Dyck word that has exactly as many 0s as 1s.

A **descent** of a (partial) Dyck word is a maximal sequence of 1s.

Lemma 11. #possible records$(R_i)_{i\leq t} \leq (\Delta-1)^t$#partial Dyck words with t 0s and all descends of size at least 4 and even

The idea is that if $R_i = (k,l)$, then $l \leq (\Delta-1)^{2k-2}$ (since this is a limit of $2k$-cycles containing e_t). Implying that if we store R_i as the word 11....1 of length $2k-2$ there is at most $(\Delta-1)^{\#1\text{s}}$ possibilities of what l was (and thereby this is bound for the information we've lost). Now we can put a zero in front of each of these words for R_i and put them together which will be a partial Dyck word since every 0 corresponds to an edge coloured and 1 to an edge uncoloured, thereby in any prefix $\#0s - \#1s$ is just the number of coloured edges at that point i.e. $\#0s \geq \#1s$ and each descends is of length $2k-2$ for some entry $R_t(k,l)$ (See figure 3).

Now with some very technical work the right hand side was bounded in the paper and lead to the result:

Corollary 12. #possible records$(R_i)_{i\leq t} \leq (\Delta-1)^t C 2^t t^{-\frac{3}{2}}$ for some constant $C > 0$.

With which we can prove the main result:

Corollary 13. The algorithm must terminate and give a proper acyclic edge-colouring if the number of available colours is $\geq 4(\Delta-1)$

Further using an additional colour this algorithm finds a colouring in $O(m\Delta\log\Delta)$ expected steps

The first part just follows from

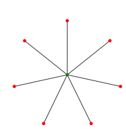
$$Pr(\text{algorithm not terminated}) = \frac{\#\text{inputs not terminated at step } t}{\#\text{total possible at step } t} \leq \text{constant } t^{-\frac{3}{2}}$$

since each combination of inputs has equal probability. The second part works similar but is a bit more technical.

The interesting thing of the algorithm is that its application are not only limited to acyclic colourings but with some minor adjustments can be applied for all sorts of colourings. For example another possible application are star-colourings:

Definition 14. A **Star Colouring** is a proper colouring such that the vertices of any two colours induce a forest of stars.

Here 'star' is just another name for graphs of the form $K_{1,k}, k \geq 0$ since it looks like a star with one cell at the center (see drawing).



Lemma 15. A proper colouring is a star-colouring iff every path on 4 vertices has at least 3 colours.

This alternative definition leads to the following definition:

Definition 16. A k**-Star Colouring** is a proper colouring such that any path on $2k$ vertices uses at least 3 colours.

Theorem 17. *Every Graph with maximum degree Δ has a k-star-colouring using $C_{2k-2}k^{\frac{1}{2k-2}}\Delta^{\frac{2k-1}{2k-2}}+\Delta$ colours, where $C_l = l(l-1)^{\frac{1}{l}-1}$*

In particular for $k=2$, this means that there exists a star-colourings using $2\sqrt{2}\Delta^{2k-1}$ colours.

Here we are just going to use the same algorithm but instead of disallowing 2-coloured cycles we're disallowing $2k$-paths (and are colouring vertices instead of edges) and using K colours:

Given uncoloured graph
while graph not fully coloured **do**
  $v \leftarrow$ min uncoloured vertex
  Choose random $F \in K - \Delta$
  Assign v F-th colour, that is not used by neighbour
  if 2-coloured $2k$-path is formed **then**
    Uncolour v as well as all but two consecutive vertices on the path
  end if
end while

The analysis will be the same as before.

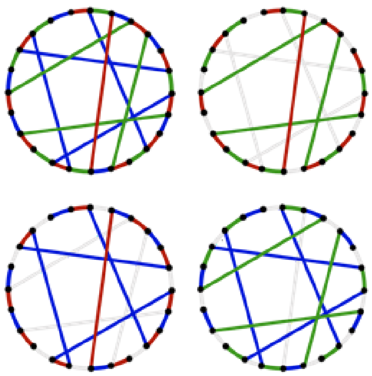


Figure 1: Any two colours induce a cycle-free subgraph.

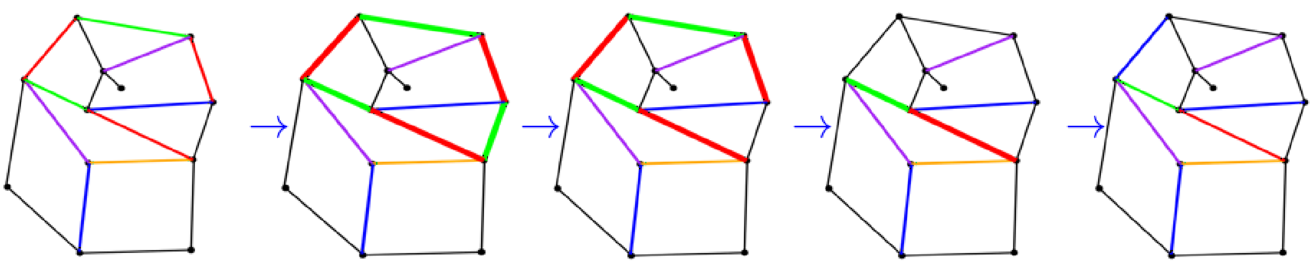


Figure 2: We're colouring an edge and thereby create a 2-coloured cycle (marked with thicker lines). So first we're uncolouring the last coloured edge and then every edge on the cycle but the two edges after it. After which we just continue colouring.

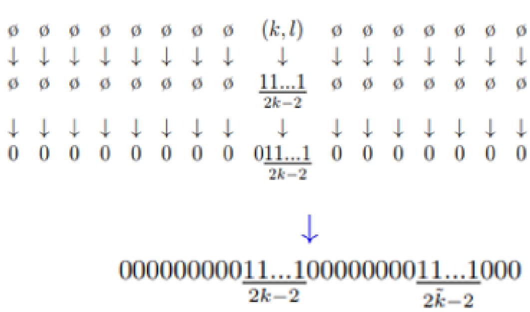


Figure 3: Creating a Dyck word from the record