7 Colourings, and other hard problems

We motivate this lecture with a historical excursion: Besides the "7 bridges of Königsberg" problem, another milestone in the early development of graph theory is the "4 *colour problem*" which originated in the middle of 19th century, and remained unsolved for more than 100 years!

Briefly, the task is to show that any political map can be coloured with just 4 colours... Easy? Not, and (unsuccessful, though) attempts to solve this simple looking question stimulated the development of most of the areas of contemporary graph theory.

BTW, yet another historical excursion discovers the "*chess knight*" problem, much older than the two previous. Can you traverse the complete chessboard with a knight? And how does this very old question relates to modern graph theory? □

Brief outline of this lecture

- Definition of a colouring of a graph. Basic properties.
- Some variants of the colouring problems.
- Hamiltonian cycle, and some other "difficult" problems.
- Algorithmic complexity (and NP-compl.) of basic graph problems.

7.1 Proper colourings and the chromatic number

Imagine that a given graph models relations between objects, the neighbouring pairs being somehow similar, and in need of a distinguishing "colouring" which makes all adjacent pairs of different colours. How can this be formulated mathematically? □

Definition: A *(proper) colouring* of a graph G with k colours is any assignment

$$c : V(G) \to \{1, 2, \ldots, k\},$$

such that every pair of adjacent vertices gets distinct colours, i.e. $c(u) \neq c(v)$ for all $\{u, v\} \in E(G)$.

Definition 7.1. The chromatic number of a graph G
is the least integer $\chi(G)$ such that G can be properly coloured with $\chi(G)$ colours. □

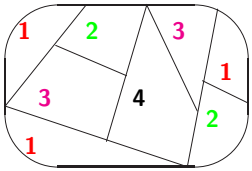
- The numbers $1, 2, \ldots, k$ used to colour the vertices of a graph G are naturally called the (vertex) colours (it is easier than to say white, red, blue, black, etc).
- Notice that the chromatic number and colourings make sense only for loopless graphs! Parallel edges, on the other hand, do not matter at all.

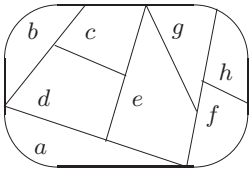
Lemma 7.3. *Let G be a simple graph (i.e., no loops) and $H \subseteq G$ an arbitrary subgraph of it. Then $\chi(H) \leq \chi(G)$.*

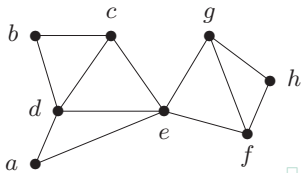
We say that the chromatic number of a graph is *subgraph-monotone*. □

Lemma 7.4. *Let G be a simple n-vertex graph. Then $\chi(G) \leq n$, and an equality holds if and only if $G \simeq K_n$ is complete.*

Example 7.5. *What is the relation of the above graph colouring with "map colourings"?*

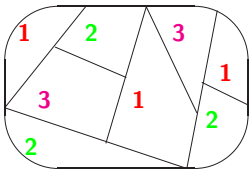


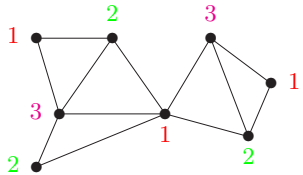




The map regions (assumed connected in the picture) are represened by the vertices of a graph, and neighbouring pairs of regions (sharing a section of a boundary) define the edges of this graph, as in the picture. The meaning of a proper colouring is now clear. □

We can, however, find a better colouring of our map, just looking at the derived graph as follows.



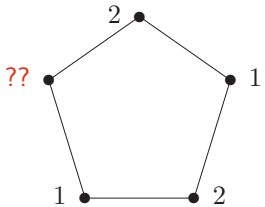


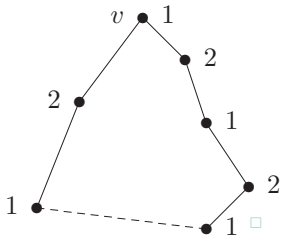
□

Theorem 7.6. *A nonempty gr. G has chromatic number 1 if and only if G is edgeless. G has chromatic number at most 2 if and only if G contains no odd cycles.* □

Proof: The first claim is trivial. □

Considering the second one, we easily see that an odd cycle needs 3 colours. Conversely, we may run a breadth-first search (BFS) through G from any v, and colour the vertices by their distance from v — alternatively giving colours $1, 2, 1, 2, \ldots$





Why this simple colouring works; is proper? Notice that any "failure" (i.e. an edge of G between two vertices that got the same colour) would imply an existence of an odd-length closed walk in G, and that in turn forces an existence of an odd cycle, a contradiction. □

7.2 How to Colour a Graph

Greedy approach to graph colouring

Definition: A graph G is *k-degenerated* if every subgraph of G contains a vertex of degree at most k.

An example of a k-degenerated graph is that of maximum degree k, but there exist more k-degenerated examples with much higher maximum degree. On the other hand, it is not enough for G just to have some vertex of small degree. □

Theorem 7.7. *Every k-degenerated loopless graph can be greedily (and properly) coloured by $k+1$ colours.* □

Proof: We pick any vertex v_1 of degree $\leq k$, and we recursively colour the subgraph $G - v_1$ with $k+1$ colours. Then we notice that the neighbours of v_1 carry at most k distinct colours, and so we can properly colour also v_1 with the remaining colour. □□

Very easy and nice, right? But the general situation with colouring algorithms will be much more difficult. Still, one slight strengthening is possible.

Theorem 7.8. *Let G be a connected simple graph of maximum degree $k \geq 2$. Then $\chi(G) \leq k$ with the only exceptions when G is an odd cycle or a complete graph.*

Proof (a sketch): For $k = 2$ this follows from Theorem 7.6. Let now $k \geq 3$. In one direction, $\chi(K_{k+1}) = k+1$. So assume that G is not complete, and that all degrees in G are equal k (since the greedy approach of Theorem 7.7 could be applied otherwise).□

- In the first step, we find two non-adjacent vertices u, v with a common neighbour w. If $G-\{u, v\}$ is not connected, we colour it parts separately by induction.□
- So $G - \{u, v\}$ is connected. The second step argues that the graph H resulting from $G - w$ by identifying u with v into one vertex is $(k-1)$-degenerated. □
- Hence H can be greedily k-coloured using Theorem 7.7. After we "split" into original u and v, we get a colouring of $G - w$ having u, v of the same colour. Then w "sees" at most $k-1$ distinct colours in its neighbourhood of size k, and a proper colour can be chosen for w.

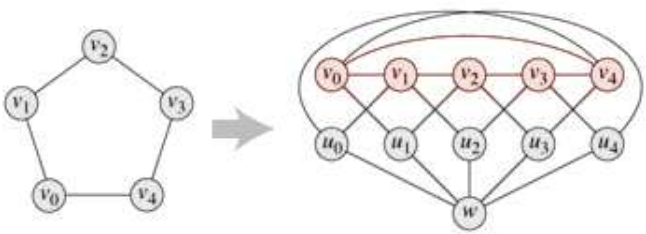
□

Graphs of high chromatic number

Thinking about graphs which cannot be coloured with few colours, the first ones coming to mind are the cliques. But what else, are these the only structures forcing high chromatic number? □

No, we can even show that there exist triangle-free graphs of arbitrarily high chromatic number! For instance, the construction of Mycielski gives:

Proposition 7.9. *The graph H obtained from any graph G by the following sketched construction has the chromatic number $\chi(G)+1$, and H is triangle-free if G was.*



□

Even more, one can read the following surprisingly strong result of Erdős:

Theorem 7.10. *For any $c, r > 0$ there exists a graph of chromatic number at least c and not containing a cycle of length less than r.*

7.3 Variants of colouring problems, and Hamiltonicity

For start, we look at the case in which not vertices, but edges receive colours.

Definition 7.11. The edge chromatic number $\chi_e(G)$ of a graph G.
We are looking for an *edge colouring* $c_e(E(G)) \to \{1, 2, \ldots, k\}$ such that no two edges sharing a vertex have the same colour.

The least number k for which an edge colouring of G exists is called the edge chromatic number $\chi_e(G)$. □

The theorem of Vizing gives a sharp approximation of edge chromatic number.

Theorem 7.12. *If G is a simple graph, then $\Delta(G) \leq \chi_e(G) \leq \Delta(G) + 1$.* □

For most of graphs, it actually holds $\Delta(G) = \chi_e(G)$. Can you easily come with graphs of the other kind?

Still, the edge colourability problem remains very hard, and it is also related to the 4 colour problem (as edge colouring of planar bridgeless graphs).

Selecting colours from distinct lists

For another view, imagine that the colours of vertices are not selected from a global pool, but instead from separate (local) lists at each vertex.

Definition 7.13. Choosability (the list chromatic number) of a graph G.
Given is a graph G with an assignment of "colour lists" $L : V(G) \to \binom{\mathbb{N}}{k}$ (k-element subsets of colours). The task is to find a *list colouring* $c_{ch}(V(G)) \to \mathbb{N}$ such no two adjacent vertices receive the same colour, and $c_{ch}(v) \in L(v)$ for every vertex v.

The least size k of the colour lists which guarantees an existence of a list colouring of G (against all choices of L) is called the *choosability* (list chromatic number) $ch(G)$ of the graph G. □

Choosability (surprise?) can generally be much higher than ordinary chromatic number.

Proposition 7.14. *For every integer k there exists a bipartite (chromatic number 2) graph of choosability greater than k.*

Hamiltonian graphs

Another typical hard graph question is that about an existence of a cycle (or path) through all vertices of the given graph:

Definition: A cycle C in a graph G is called *Hamiltonian* if C spans all vertices of G. A *Hamiltonian path* P in G is defined analogically.

A graph G is *Hamiltonian* if G contains a Hamiltonian cycle. □

Though it may sound strange, the Hamiltonian cycle problem is also related to some attempts to solve the 4 colour problem. This is, though, out of the scope of our lecture.

Instead, we present the following result of Dirac with a very nice short proof.

Theorem 7.15. *Every graph on $n \geq 3$ vertices and with minimum degree $\geq n/2$ is Hamiltonian.*

7.4 $\mathcal{NP}$-completeness of many graph problems

Recall the definition of the class $\mathcal{NP}$ – those decision problems for which there exists a polynomial certificate. The *$\mathcal{NP}$-complete* problems are those having the highest difficulty within the class $\mathcal{NP}$, i.e. those to which every member of $\mathcal{NP}$ can be *reduced*.

□

As we shall see, most of natural graph problems are $\mathcal{NP}$-complete. We start the excursion from the following classical "master" problem:

Problem 7.16. 3-SAT (a spec. version of satisfiability)
The following problem is $\mathcal{NP}$-complete:

Input: Logic formula Φ in a conjunctive normal form, such that every clause of Φ contains ≤ 3 literals.

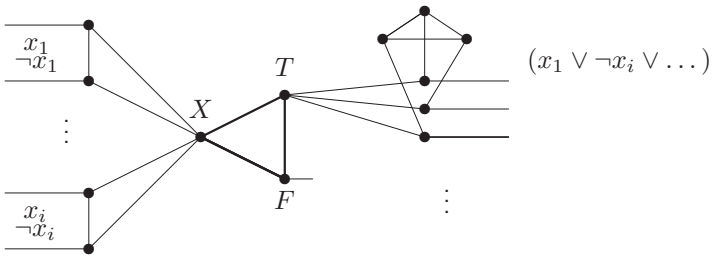
Output: Is there a valuation of the Φ-variables that makes Φ true?

For instance, $\Phi \equiv (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

Problem 7.17. 3-COL (3-Colouring of graphs)
The following problem is $\mathcal{NP}$-complete:

Input: Graph G.
Output: Can the vertices of G be properly coloured using three colours?

Proof (a sketch): We show a polynomial reduction from 3-SAT. □
For a given G we construct a formula Φ: The basis of the graph is a triangle with vertices denoted by X, T, F. Each variable x_i in Φ is assigned a vertex pair adjacent to X. Each clause of Φ is assigned a subgraph on 6 vertices (three of them adjacent to T), as in the picture. Then the remaining free "halfedges" are joined together in the way corresponding to the literals in the clauses.

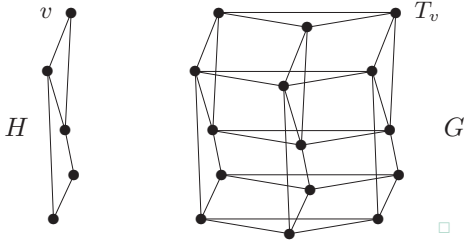


Then one may easily check that G has a 3-colouring iff Φ is satisfiable. □

Problem 7.18. IS (Independent Set)
The following problem is $\mathcal{NP}$-complete:

Input: Graph G, and an integer k.
Output: Is there an independent subset of size (at least) k in G? □

Proof: We show a polynomial reduction from 3-COL.
Let H be a graph on n vertices which should be 3-coloured. We set $k = n$, and construct a graph G made of three disjoint copies of H as shown :



□

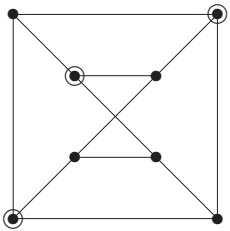
Assume $c : V(H) \to \{1, 2, 3\}$ is a 3-colouring of H. Then one can choose $k = n$ independent vertices in G – for each $v \in V(H)$ choosing the $c(v)$-th copy of v in the graph G. Conversely, if I is an independent subset in G of size $k = n$, then every triangle T_v, $v \in V(H)$ intersects I just in one vertex. That determines the colour for v in H. □

Definition: A *vertex cover* in a graph G is such a set $C \subseteq V(G)$ that every edge of G is incident with a vertex of C, i.e. $G - C$ is independent.

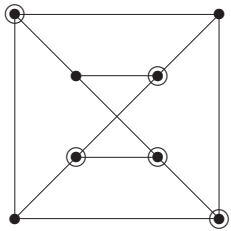
Problem 7.19. VC (Vertex Cover)
The following problem is $\mathcal{NP}$-complete:

Input: Graph G, and an integer k.
Output: Is there a vertex cover of size at most k in G? □

Proof: We show a polynomial reduction from IS. Notice that the complement $C = V(G) \setminus I$ of an independent set I is actually a vertex cover. So the reduction works with the same graph G and with $k = n - \ell$. □



independent set



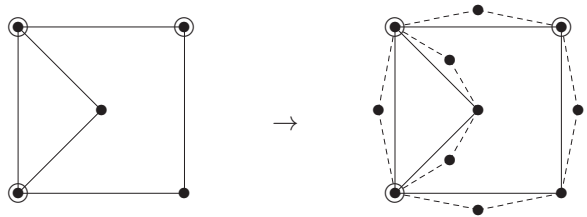
vertex cover

Definition: A *dominating set* in a graph G is such a set $D \subseteq V(G)$ that every vertex of G not in D has a neighbour in D.

Problem 7.20. DOM (Dominating set)
The following problem is $\mathcal{NP}$-complete:

Input: Graph G, and an integer k.
Output: Is there a dominating set of size at most k in G?□

Proof (a sketch): Given any graph H, we construct an input graph G for DOM as follows: For every edge $e \in E(H)$, a new vertex v_e is added, forming a triangle with e.



vertex cover dominating set

Now a vertex cover (cf. Problem 7.19) of the former graph is the same as a dominating set in the latter graph. □

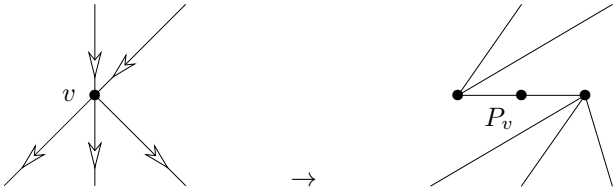
Problem 7.21. HC (Hamiltonian cycle, directed)
The following problem is $\mathcal{NP}$-complete:

Input: Directed graph G.
Output: Can we find a directed cycle in G passing through all vertices? □

Problem 7.22. HAM (Hamiltonian cycle)
The following problem is $\mathcal{NP}$-complete:

Input: Graph G.
Output: Can we find a (undirected) circuit in G passing through all vertices? □

Proof:



It is an easy reduction from the previous problem HC. Each vertex v of a directed graph H is replaced with three vertices forming a path P_v in the graph G. then the directed edges coming into v are joined to the first vertex of P_v, and the edges leaving from v are joined to the last vertex of P_v. □

7.5 The interesting story of Vertex Cover

Consider the (at first glance) very similar problems of a vertex cover and of a dominating set in a graph—both are among the classical $\mathcal{NP}$-complete problems. Yet, we discover a huge difference between them, briefly outlined as follows. □

- If, in the computational complexity analysis, we focus on the value of the input parameter k, then we still cannot solve Dominating Set in a better way than exhaustively checking (almost) all k-tuples of vertices.

  Even when k is fixed small, say $k = 10,\ 20$, this an intractable problem. □

- On the other hand, a Vertex Cover of size k can be decided by a very simple algorithm running in time $O(2^k \cdot n)$, which is quite usable for small fixed values of k such as $k = 10,\ 20$, giving actually a linear time algorithm!

Algorithm 7.24. k**-VC (Vertex Cover)**
For any fixed k we are solving the following problem.

Input: Graph G.
Output: Is there a vertex cover of size at most k in G? □

We initialize $C = \emptyset$ and $F = E(G)$.

- *If $F = \emptyset$, then C is returned as a vertex cover.*
  If, otherwise, $|C| \geq k$, then the return value is "NO".
- *We pick an arbitrary edge $f = uv \in F$, and for each of its ends $x = u, v$ we do:*
  - *$C' = C \cup \{x\}$, and the edge set F' results from F by removing all edges incident with x in G.*
  - *This algorithm is called recursively for G, C' and F'.* □

Finally, how many (self-)recursive calls occurs in Algorithm 7.24 altogether? Every call generates two further recursive calls, but only up to a fixed depth k. Hence the total running time is asymptotically only $O(2^k \cdot n)$. □

Remark: The factor 2^k can be improved by more careful choice of branching. (2006: 1.2738^k)