INF421, Lecture 8 Graphs

Leo Liberti

LIX, École Polytechnique, France

Lecture summary

- Graph definitions
- Operations on graphs
- Combinatorial problems on graphs
- Easy and hard problems
- Modelling problems for a generic solution method

Course

- Objective: to teach you some data structures and associated algorithms
- Evaluation: TP noté en salle info le 16 septembre, Contrôle à la fin. Note: max $(CC, \frac{3}{4}CC + \frac{1}{4}TP)$
- Organization: fri 26/8, 2/9, 9/9, 16/9, 23/9, 30/9, 7/10, 14/10, 21/10, amphi 1030-12 (Arago), TD 1330-1530, 1545-1745 (SI31,32,33,34)
- Books:
 - 1. Ph. Baptiste & L. Maranget, Programmation et Algorithmique, Ecole Polytechnique (Polycopié), 2009
 - 2. G. Dowek, Les principes des langages de programmation, Editions de l'X, 2008
 - 3. D. Knuth, The Art of Computer Programming, Addison-Wesley, 1997
 - 4. K. Mehlhorn & P. Sanders, Algorithms and Data Structures, Springer, 2008
 - Website: www.enseignement.polytechnique.fr/informatique/INF421
- **Contact:** liberti@lix.polytechnique.fr (e-mail subject: INF421)

INF421, Lecture 8 - p. 2



The minimal knowledge

- Operations on graphs: complement, line graph, contraction
- Decision/optimization problems: finding subgraphs with given properties
- Easy problems: solvable in polynomial time (P), e.g. minimum cost spanning tree, shortest paths, maximum matching
- Hard problems: efficient method for solving one would solve all of them (NP-hard), e.g. maximum clique, maximum stable set, vertex colouring
- Mathematical Programming: a generic model-and-solve approach

Graph definitions

Motivation

The ultimate data structure

Every time you see arrow connecting boxes, circles or black dots in a computer science course, you can think of graphs and digraphs!



Warning: not all such extensions are trivial

Example

- If G is a graph, we indicate its set of vertices by V(G) and its set of edges by E(G)
- Example of extension to digraphs: If G is a digraph, we indicate its set of nodes by V(G) and its set of arcs by A(G)

- Digraph G = (V, A): relation A on set V
 - V: set of nodes

- Graph G = (V, E): symmetric relation E on set V
 - V: set of vertices
 - E: set of edges $\{u, v\}$ with $u, v \in V$



Simple (di)graphs: relation is irreflexive (I.e., v not related to itself for all $v \in V$)

Stars

Stars: sets of nodes/vertices or arcs/edges adjacent to a given node

 $\forall v \in V(G),$

- if G is undirected,
 - $N(v) = \{u \in V \mid \{u, v\} \in E(G)\}$
 - $\bullet \ \delta(v) = \{\{u,v\} \mid u \in N(v)\}$
- if G is directed,
 - $N^+(v) = \{ u \in V \mid (v, u) \in E(G) \}$
 - $\delta^+(v) = \{(v, u) \mid u \in N^+(v)\}$
 - $N^-(v) = \{ u \in V \mid (u,v) \in E(G) \}$
 - $\delta^{-}(v) = \{(u, v) \mid u \in N^{-}(v)\}$
- $|N(v)| = \text{degree, } |N^+(v)| = \text{outdegree, } |N^-(v)| = \text{indegree Of } v$
- If v belongs to two graphs G, H, write $N_G(v)$ and $N_H(v)$ (similarly for other star notation)

Cutsets

- Let H = (U, F) be a subgraph of G = (V, E) (i.e. $U \subsetneq V$)
- The cutset $\delta(H) = \left(\bigcup_{u \in U} \delta(u)\right) \smallsetminus F$ is the edge set "separating" U and $V \smallsetminus U$
- E.g. let $U = \{1, 2, 6\}$ and H = G[U], then $\delta(H)$ is shown by the red edges below



- Similar definitions hold for directed cutsets
- If G is undirected, $\delta(U) = \delta(V \smallsetminus U)$ for all $U \subseteq V(G)$

Subgraphs

• A graph H = (U, F) is a subgraph of G = (V, E) if $U \subseteq V$, $F \subseteq E$ and $\forall \{u, v\} \in F \ (u, v \in U)$



- A subgraph H = (U, F) of G = (V, E) is spanning if U = V
- A subgraph H = (U, F) of G = (V, E) is induced by U if $\forall u, v \in U \ (\{u, v\} \in E \rightarrow \{u, v\} \in F)$





Connectedness

A graph is connected if there are no empty nontrivial cutsets





Connected

Not connected: $\delta(\{1,2,6\}) = arnothing$

- Each maximal connected subgraph of a graph is a connected component
- Most graph algorithms assume the input graph to be connected: if not, just apply it to each connected component

INF421, Lecture 8 - p. 1

Paths and cycles

- Let G be a graph and $u, v \in V(G)$
 - A simple path P from u to v in G is a connected subgraph of G s.t.:
 - 1. each vertex w in P different from u, v has |N(w)| = 2
 - 2. if $u \neq v$ then |N(u)| = |N(v)| = 1
 - 3. if u = v then either $E(P) = \varnothing$ or |N(u)| = |N(v)| = 2
- We indicate a path from u to v by the notation $u \rightarrow v$
- If P is a path $u \to v$, then u, v are called the endpoints of the path
- A simple cycle is a simple path with equal endpoints
- Definitions in Lecture 6 equivalent but more general
- Will simply say paths/cycles to mean *simple* paths/cycles



Complement graph

• Given G = (V, F) with |V| = n, the complement of G is $\overline{G} = (V, E(K_n) \smallsetminus F)$



• The complement of K_n is the empty graph (V, \emptyset) on n vertices

Complete graph

• The complete graph K_n on n vertices has all possible edges



K_n is also called *n*-clique; a complete graph on a vertex set *U* is denoted by *K*(*U*)

INF421, Lecture 8 - p. 14



Forests and trees

A forest is a graph with no cycles

 $\begin{array}{c} & & & 7 \\ & 5 & 4 \\ & & & \\ 6 & & & 3 \\ & & & \\ & & 1 & - - 2 \end{array}$

A tree is a connected forest



If a tree is a subgraph of another graph G, we also call it a spanning tree



Graphs modulo symmetry

 Symmetries act on vertex labels — can represent equivalence classes of graphs modulo symmetry by simply ignoring labels



Not easy to treat mathematically, though...

Line graphs

 $N(4) = \{3, 7, 6\}$

- Given a graph G = (V, E) where $E = \{e_1, \ldots, e_m\}$
- The line graph of G is

 $N(7) = \{3, 4, 6\}$

 $L(G) = (E, \{\{e_i, e_j\} \mid e_i \cap e_j \neq \varnothing\})$

- Every vertex of L(G) is an edge of G
- Two vertices e_i, e_j of L(G) are adjacent if there is v ∈ V such that e_i, e_j ∈ δ(v)



Property: the degree $|N_{L(G)}(e)|$ of a vertex $e = \{u, v\}$ of L(G)is $|N_G(u)| + |N_G(v)| - 2$. L(G) can be constructed from Gin polynomial time (how?)



Operations on graphs

INF421, Lecture 8 - p. 17



Subgraph contraction

- Let G = (V, E), $U \subseteq V$ and H = G[U]
- The contraction G/U is a sort of "G modulo H" contract(G,U):
 - 1: Let z be a new vertex not in V;
 - 2: Add vertex *z*;
 - 3: for $\{u, v\} \in \delta(H)$ (assume WLOG $u \in U, v \in V \smallsetminus U$) do
 - 4: add edge $\{v, z\}$;
 - 5: remove edge $\{u, v\}$;
 - 6: end for
 - 7: remove G[U];
 - 8: return *G*;
- At the end of the contraction algorithm, the whole subgraph *H* is "replaced" by a single vertex *z*

```
\begin{tabular}{ll} \beg
```

Subgraph contraction is equivalent to a sequence of edge contractions

Minors

- The graph *H* is a minor of the graph *G* if it is isomorphic to the graph obtained by a sequence of contractions
- Useful to underline "essential structure" of a complicated graph



Contract some triangles

Combinatorial problems on graphs

Graph optimization problems

- To most decision problems on graphs there is a corresponding *optimization problem*
- Let $\mu : \mathbb{G} \to \mathbb{R}$ be a "measure" function for graphs
- E.g. μ could be the number of vertices, or of edges SUBGRAPH PROBLEM, optimization version (SP_{P, μ}). Given a graph G, does it have a subgraph H with property P and having minimum/maximum measure μ ?
- Given a property *P* and a graph measure μ, the set of instances of SP_{P,μ} is an optimization problem

The subgraph problem

- Let G be the class of all graphs
- For a set of valid propositions P(G) (for G ∈ G), a typical decision problem in graph theory is the following:
 SUBGRAPH PROBLEM (SP_P). Given a graph G, does it have a sub-

graph H with property P?

- Decision problem: YES/NO question parametrized over symbols representing the instance (i.e. the input)
- Formally, a decision problem is the set of all possible inputs
- Require decision problems to always provide a certificate (a proof that certifies the answer)
- E.g. if $P(H) \equiv (H \text{ is a cycle})$ the certificate is the cycle
- NP is the class of decision problems whose certificates for YES instances can be verified in polynomial time w.r.t. the instance size

INF421, Lecture 8 - p. 26

Easy problems

- We let P be the class of decision or optimization problems which can be solved in polynomial time
- We call problems in P "easy"
 - MINIMUM SPANNING TREE (MST)

Seen in Lecture 6

- SHORTEST PATH PROBLEM (SPP) from a vertex v to all other vertices To be seen in Lecture 9
- MAXIMUM MATCHING problem (MATCHING) Discussed in INF550

Matching: subgraph given by set of mutually non-adjacent edges A maximum matching M, $\mu(M) = |E(M)|$



Hard(er) problems

Maximum clique

CLIQUE PROBLEM (CLIQUE). Given a graph G, what is the largest n such that G has K_n as a subgraph?

• In Clique, $\mu(H) = |V(H)|$ and $P(H) \equiv H = K(V(H))$





 Several applications to social networks and bioinformatics

INF421, Lecture 8 - p. 30

Clique and NP-completeness

- The decision version of CLIQUE is: k-CLIQUE PROBLEM (k-CLIQUE). Given a graph G and an integer k > 0, does G have K_k as a subgraph?
- Consider the following result (which we won't prove) Thm.

[Karp 1972] If CLIQUE \in P then P = NP

- Any decision problem for which such a result holds is called NP-complete
- It is not known whether NP-complete problems can be solved in polynomial time; the current guess is NO

Solving NP-complete problems

Essentially, proving NP-completeness of a problem amounts to say "it's really hard"

If it were easy, every problem in $\ensuremath{\text{NP}}$ would be easy, which is unlikely: so it's likely to be hard

- Solution methods for NP-complete problems include:
 - exact but exponential worst-case complexity algorithms
 - heuristic algorithms

whenever they find a YES answer they provide a certificate, but there is no guarantee that they are able to determine whether an answer is NO in a finite amount of time

- Some optimization problems are also such that "solvable in polynomial time" implies P = NP: they are called NP-hard
- With NP-hard optimization problems, can use approximation algorithms their solutions have a μ -value which is no more than f(|G|)-fold worse than the optimal one (e.g. $\mu \leq f(|G|)\mu^*$, where μ^* is the cost of an optimal solution)

Stables

A stable (or independent set) of a graph G = (V, E) is a subset $U \subseteq V$ such that $\forall u, v \in U \ (\{u, v\} \notin E)$ Thm.

U is a stable in G if and only if $\overline{G}[U]$ is a clique





- Decision problem: k-STABLE Given G and $k \in \mathbb{N}$, is there a stable $U \subseteq V(G)$ of size k?
- Optimization problem: STABLE
 Given G, find the stable of G of maximum size

Stable heuristic

- It suffices to give an algorithm for STABLE, the one for CLIQUE will follow trivially (why?)
- The following greedy method will find a maximal stable

```
1: U = \emptyset;

2: order V by increasing values of |N(v)|;

3: while V \neq \emptyset do

4: v = \min V;

5: U \leftarrow U \cup \{v\};

6: V \leftarrow V \smallsetminus (\{v\} \cup N(v))

7: end while
```

Worst-case: O(n) (given by an empty graph)

NP-completeness of *k*-Stable

Thm.

X

k-STABLE is **NP-complete**

Proof

Consider an instance $(\boldsymbol{G},\boldsymbol{k})$ of $\boldsymbol{k}\text{-}\mathrm{CLIQUE}$

The complement graph \bar{G} can be obtained in polynomial time $\ (*)$

It is easy to show that $\overline{\overline{G}} = G$ (**)

By $(\ast\ast)$ and previous thm.,

(G,k) is a YES instance of $k\text{-}\mathsf{CLIQUE}$ iff (\bar{G},k) is a YES instance of $k\text{-}\mathsf{STABLE}$

By (*), if $k\text{-Stable} \in \mathbf{P}$ then $k\text{-Clique} \in \mathbf{P}$

By NP-completeness of $k\text{-}\mathsf{Clique},\,k\text{-}\mathsf{Stable}\in\mathbf{P}$ implies $\mathbf{P}=\mathbf{NP}$

Hence *k*-STABLE is **NP-**complete

- **I** How to show that a problem \mathcal{P} is **NP**-complete:
 - **•** Take another **NP**-complete problem \mathcal{Q} "similar" to \mathcal{P}
 - Transform (with a poly. alg.) an instance of ${\mathcal Q}$ to an instance of ${\mathcal P}$
 - Show that transformation preserves the YES/NO property

INF421, Lecture 8 - p. 34

Heuristic fails

- The above algorithm may fail to find a maximum stable
- When choosing second element of U, instead of choosing leftmost vertex, choose:



 Then algorithm stops immediately with a stable of cardinality 2

A polynomial case

- Not all instances of an NP-complete problem are hard
- Let P be a decision problem and C⊆P be an infinite set of instances for which there exists a polynomial algorithm
- **9** Then $C \in \mathbf{P}$, and C is a polynomial case of P
- For example, let $\mathcal{L} = \{H \in \mathbb{G} \mid \exists G \in \mathbb{G} \ (H = L(G))\}$ be the class of graphs that are line graphs of another graph

Thm.		
A maximum matching in	G	is
a stable in $L(G)$		



Since MATCHING∈P and finding L(G) can be done in polynomial time, STABLE_L ∈P

INF421, Lecture 8 - p. 37

Vertex colouring example



Vertex colouring

Decision problem

VERTEX *k*-COLOURING PROBLEM (*k*-VCP). Given a graph G = (V, E)and an integer k > 0, find a function $c : V \to \{1, ..., k\}$ such that $\forall \{u, v\} \in E \ (c(u) \neq c(v))$

Optimization problem

VERTEX COLOURING PROBLEM (VCP). Given a graph G = (V, E), find the minimum $k \in \mathbb{N}$ such that there is a function $c : V \rightarrow \{1, \ldots, k\}$ with $\forall \{u, v\} \in E \ (c(u) \neq c(v))$

- Applications to scheduling and wireless networks
- In general, find how to allocate network resources to a few capacities such that there is no conflict

INF421, Lecture 8 - p. 38



Vertex colouring heuristic

Thm.

Each color set $C_k = \{v \in V \mid c(v) = k\}$ is a stable

Use stable set heuristic as a sub-step

1: k = 1; 2: U = V; 3: while $U \neq \emptyset$ do 4: $C_k = \text{maximalStable}(G[U])$; 5: $U \leftarrow U \smallsetminus C_k$; 6: $k \leftarrow k + 1$; 7: end while

Worst-case: O(n) (given by an empty or complete graph)

Model-and-solve

Mathematical programming

- Take e.g. the STABLE problem
- Input (also called parameters):
 - set of vertices V
 - \bullet set of edges E
- **•** Output: $x : V \to \{0, 1\}$

 $\forall v \in V \quad x(v) = \begin{cases} 1 & \text{if } v \in \text{maximum stable} \\ 0 & \text{otherwise} \end{cases}$

- We also write $x_v = x(v)$
- We'd like $x = (x_v \mid v \in V) \in \{0, 1\}^{|V|}$ to be the characteristic vector of the maximum stable S^*
- $x_1, \ldots, x_{|V|}$ are also called decision variables

INF421, Lecture 8 - p. 42

Objective function

- If we take x = (0, 0, 0, 0, 0, 0, 0), S* = Ø and |S*| = 0 (minimum possible value)
- If we take x = (1, 1, 1, 1, 1, 1, 1) = 1, $|S^*| = |V| = 7$ has the maximum possible value
- Characteristic vector x should satisfy the objective function

$$\max_{x} \sum_{v \in V} x_v$$

Constraints

- Consider the solution x = 1
- *x* certainly maximizes the objective
- ... but $S^* = V$ is not a stable!

x = 1 is an infeasible solution

- The feasible set is the set of all vectors in $\{0,1\}^{|V|}$ which encode stable sets
- Defining property of a stable:

Two adjacent vertices cannot both belong to the stable

In other words,

choose at most one vertex adjacent to each edge

Written formally,

$$\forall \{u, v\} \in E \quad x_u + x_v \le 1$$

E COLL POLYTECHNIQU

Verify the constraints

- x = (0, 1, 0, 0, 0, 0, 1, 1) encodes $S^* = \{2, 6, 7\}$
- $x_u + x_v = 2$ only for $\{u, v\} \in F = \{\{2, 6\}, \{2, 7\}, \{6, 7\}$
- $I Notice F \cap E = \emptyset$
- Hence, $x_u + x_v \le 1$ for all $\{u, v\} \in E$



The magical method

- But WE CAN!
- Use generic MP solvers
- These algorithms can solve ANY MP formulation expressed with linear forms, or prove that there is no solution
- Based on Branch-and-Bound (BB)
- The YES certificate is the characteristic vector of a feasible solution
- The NO certificate is the whole BB tree, which implicitly (and intelligently) enumerates the feasible set
- YES certificate lengths are polynomial, NO certificates may have exponential length

So what?

OK, so the Mathematical Programming (MP) formulation

$$\max_{x} \sum_{v \in V} x_{v}$$

$$\forall \{u, v\} \in E \quad x_{u} + x_{v} \leq 1$$

$$x \in \{0, 1\}^{|V|}$$

describes STABLE correctly

As long as we can't solve it, why should we care?





CLIQUE and MATCHING

Clique (use complement graph):

$$\max_{x} \sum_{v \in V} x_{v}$$

$$\forall \{u, v\} \notin E, u \neq v \quad x_{u} + x_{v} \leq 1$$

$$x \in \{0, 1\}^{|V|}$$

Matching:

$$\max_{x} \sum_{\{u,v\} \in E} x_{uv}$$

$$\forall u \in V \sum_{v \in N(u)} x_{uv} \leq 1$$

$$x \in \{0,1\}^{|E|}$$

Warning: although $MATCHING \in P$, solving the MP formulation with BB is exponential-time

How to

- Come see me, l'll give you a personal demo
- Go to www.ampl.com and download the AMPL software, student version
- AMPL is for modelling, i.e. writing MP formulations
- Still from www.ampl.com, you can download a student version of the ILOG CPLEX BB implementation

And tomorrow?

If you're interested in modelling problems as MPs

- M1:
 - INF572 (Optimization: Modelling and Software)
 - MAP557 (Optimization: Theory and Applications)
- M2:
 - MPRO (Master Parisien en Recherche Operationnelle) http://uma.ensta-paristech.fr/mpro/



End of Lecture 8