Pina Kolling
piko0011

Summary: Lecture 8

Summary for the chapters *9.3* and *9.4*. [3, 1]

Undirected graph

An undirected graph G is a pair of sets (V, E) where V is the finite set of nodes and E is a set of unordered pairs in V that are symmetric:

$$\forall\, i, j \in E, i \neq j :\ (i, j) \in E \Rightarrow (j, i) \in E$$

IndependentSet

IndependentSet

Input: An undirected Graph $G = (V, E)$ and a number k.
Question: Is there a set $I \subseteq V$ of $k = |I|$ nodes with no edges in between? (IndependentSet)

3SAT

Like the Sat problem, 3Sat is determining the satisfiability of a formula in CNF where each clause is limited to at most three literals.

IndependentSet is NP-complete.

Proof idea:

- triangle construction: any independent set can contain at most one node of the triangle

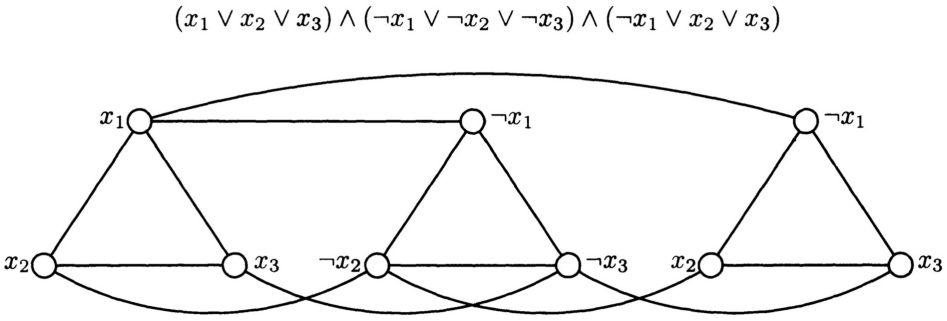


Figure 1: Graph with triangles [3]

- consider only graphs whose nodes can be partitioned in m disjoint triangles
  $\rightarrow$ independent set can contain at moast m nodes (one from each triangle)
- reduction from 3Sat to IndependentSet
- construct graph of formula ϕ:
  - each literal as a node
  - clauses as triangles
  - edges between nodes in different triangles if they correspond to the same literal (negated)
  - $K = m$ (m clauses)

- given: instance ϕ of 3SAT with m clauses $C_1, ..., C_m$
- each clause $C_i = (\alpha_{i1} \vee \alpha_{i2} \vee \alpha_{i3})$ (with α as boolean variables or negation of those)
- reduction R constructs a graph: $R(\phi) = (G, K)$ where $K = m$ and $G = (V, E)$
- nodes $V = \{v_{ij} : \ i = 1, ..., m; j = 1, 2, 3\}$
  nodes for each of the m clauses (i) for each of the 3 literals (j)
- edges $E = \{[v_{ij}, v_{ik}] : \ i = 1, ..., m; j \neq k\} \cup \{[v_{ij}, v_{lk}] : \ i \neq l, \alpha ij = \neg\alpha_{lk}\}$
  edges between the nodes in one clause (triangle edges)
  edges between nodes with the same corresponding literal, but negated

- there is an indepentent set I of K nodes in G only if ϕ is satisfiable
- I must contain a node from each triangle
- negated literals are connected: I cannot contain a literal and its negation
- I is a truth assignment of ϕ:
  - true literals: nodes in I
  - one true literal per clause

HamiltonPath is NP-complete

HamiltonPath

A HAMILTONPATH is a path in a graph that visits each node exactly once.

HAMILTONPATH is NP-complete.

Proof idea:

- reduction from 3SAT to HAMILTONPATH
- given: formula ϕ in CNF with n variales $x_1, ..., x_n$ and m clauses $C_1, ..., C_m$ with each 3 variables
- construct a graph $R(\phi)$ that has a hamilton path only if ϕ is satisfiable:
- boolean variables:
  - choice between true and false
  - all occurences of x must have the same value (and $\neg x$ the opposite)
  - use *choice* gadget (like flip flop)
- XOR:

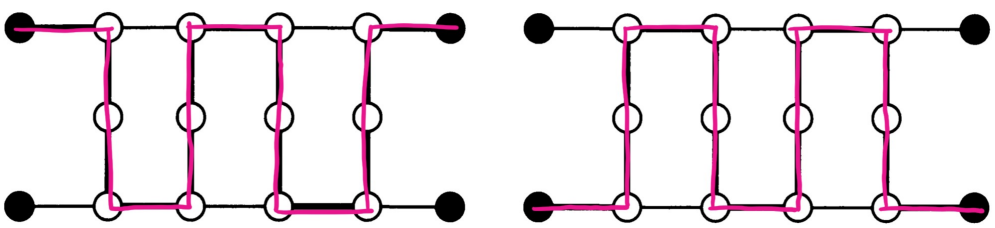


Figure 2: XOR subgraph from the book [3] with the relevant edges marked additionally

  - use *consistency* gadget
  - because of hamilton path: there are only two ways to traverse through this sub graph (as shown above)
  - leads to exclusive or (XOR)

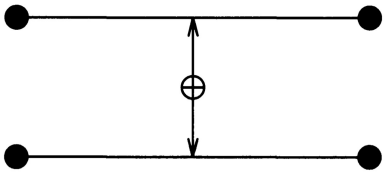


Figure 3: XOR connecting two independent edges (*consistency* gadget) [3]

- clauses:
  - triangles for clause construction
  - one side for each literal
  - if literal is false: hamilton path traveres triangle side
  - at least one literal need to be true: else all three edges of triangle will be traversed and this is not a hamilton path
- put everything together as graph G:
  - G has n copies of the *choice* gadget as a chain (one for each variable)

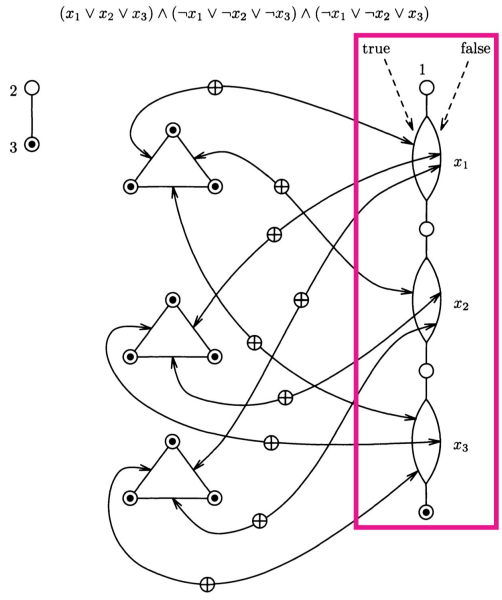


Figure 4: *Choice* gadgets marked in graph from the book [3]

- G has m triangles (one for each clause) with edges for each clause in the triangle

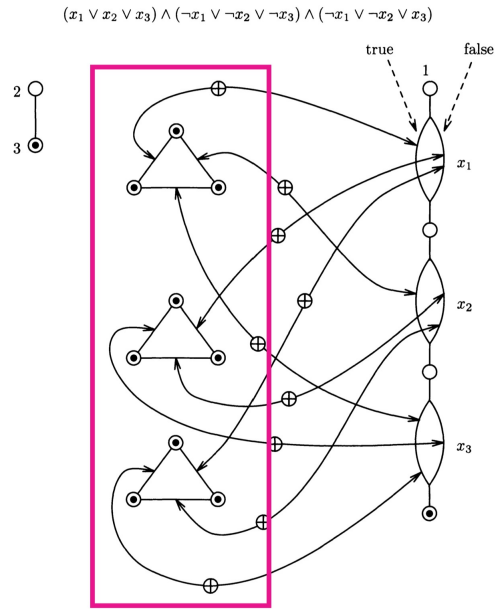


Figure 5: Clauses marked in graph from the book [3]

- finally all $3m$ nodes of the triangles, the last node of the chain of *choice* gadgets and a new node 3 are connected with all possible edges
- a single node 2 is connectes to the node 3

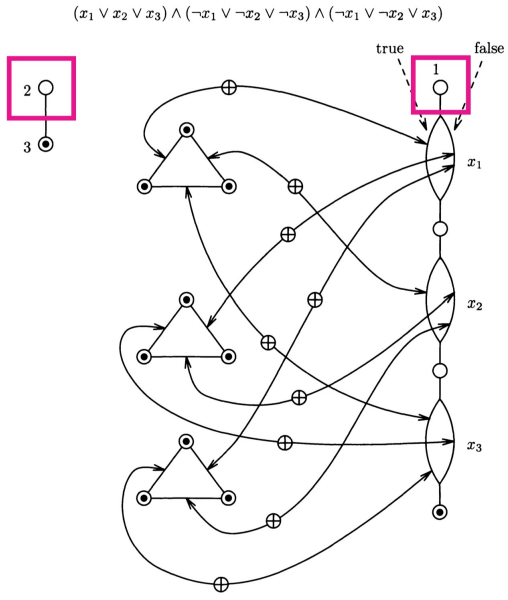


Figure 6: Nodes 1 and 2 marked (start and end node) in graph from the book [3]

- graph has a hamilton path only if ϕ has a satisfying truth assignment
- for hamilton path: start node is node 1 and end node is node 2
- from node 1 it must traverse one of the parallel edges of the *choice* gadget for the first variable
- exclusive ors must be traversed
- whole chain of *choice* gadgets will be traversed
  $\rightarrow$ in this way a truth assignment T is created

- then the triangles are traversed and it ends up in node 2 if there is a hamilton path and ϕ is satisfyable

TSP(D)

TSP(D)

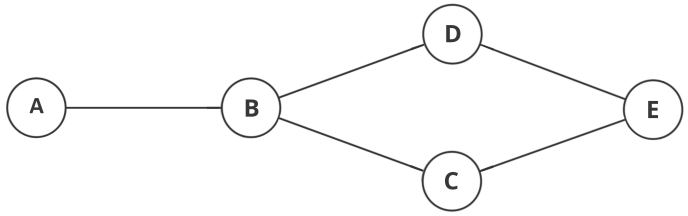
TSP(D) is a decision version of TSP.
Input: A $n \times n$ distance matrix and a bound $B \in \mathbb{N}$
Question: Is there a round tour of length $\leq B$ that visits all *cities*?

TSP(D) is NP-complete.

Proof idea:

- reduce from HamiltonPath to Tsp
- given: graph G with n nodes
- design: matrix d_{ij} and a budget B of nodes with $B = |V| + 1$ such that there is a tour of length B or less only if the G has a hamilton path
- d_{ij} usually contains the distance from city i to city j
- n cities: one node for each city in the graph
  $\rightarrow n$ nodes
- distance between two cities i and j is 1 if there is an edge $[i, j]$ and 2 otherwise

Example:



	A	B	C	D	E
A	–	1	2	2	2
B	1	–	1	1	2
C	2	1	–	2	1
D	2	1	2	–	1
E	2	2	1	1	–

Figure 7: Corresponding table to the graph

- undirected: distances are symmetric, leads to $d_{ij} = d_{ji}$
- set limit to $B = |V| + 1 = 6$
- $\sum_{i=1}^{n} d_{\pi(i),\pi(i+1)}$ is as small as possible
- π is a permutation

The following sum for the example can at most be 6:

$$\begin{aligned}
\text{A to B:}\quad & d_{\pi(0),\pi(1)} = 1 \\
\text{B to C:}\quad & d_{\pi(1),\pi(2)} = 1 \\
\text{C to E:}\quad & d_{\pi(2),\pi(3)} = 1 \\
\text{E to D:}\quad & d_{\pi(3),\pi(4)} = 1 \\
\text{D to A:}\quad & d_{\pi(4),\pi(0)} = 2 \\
& \sum = 6
\end{aligned}$$

- the answer to the TSP problem is *yes*
- the graph contains a hamilton path

TODO
proof
Questions:

Knapsack

Knapsack

Given is a set of items with a weight and a value. The task is to choose which items to include so that the total weight is less than the given limit of the knapsack and the total value is as large as possible.

Recalling my Dynamic Programming knowledge for the KNAPSACK problem
The formula for the dynamic programming of this problem is the following:

$$Opt(i,j) = \begin{cases} 0, & \text{for } 0 \leq j \leq size \\ Opt(i-1,j), & \text{for } j < w[i] \\ max\{Opt(i-1,j), v[i] + Opt(i-1, j-w[i])\}, & \text{else} \end{cases}$$

- if $0 \leq j \leq w$: there are no objects which could be put into the bag
- second case: object i does not fit into the bag and the optimal solution is found with the objects from index 1 to $i-1$
- else: object i is either part of the optimal solution or it consists out of the objects 1 to $i-1$

Table fill-out example:

- left: table of the values and weight of each item with index i is shown on the left
- right: table that gets filled in with a dynamic programming approach of the formula above, the maximum bag size is 7
- first column in the table represent the objects and the last row the weight
- entries in the table show the maximum value
- maximum value for the size 7 is 10.

i	$w[i]$	$v[i]$
1	1	1
2	3	4
3	2	3
4	4	6
5	6	8

5	0	1	3	4	6	7	9	10
4	0	1	3	4	6	7	9	10
3	0	1	3	4	5	7	8	8
2	0	1	1	4	5	5	5	5
1	0	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7

ExactCoverBy3Sets

Given a set X, with $|X| = 3q$ (so, the size of X is a multiple of 3), and a collection C of 3-element subsets of X. Can we find a subset C' of C where every element of X occurs in exactly one member of C'? (So, C' is an exact cover of X). [2]

Example ExactCoverBy3Sets:

- suppose X was $\{1, 2, 3, 4, 5, 6\}$
- if C was $\{\{1, 2, 3\}, \{2, 3, 4\}, \{1, 2, 5\}, \{2, 5, 6\}, \{1, 5, 6\}\}$ then $C' = \{\{2, 3, 4\}, \{1, 5, 6\}\}$ is an exact cover because each element in X appears exactly once
- if instead, C was $\{\{1, 2, 3\}, \{2, 4, 5\}, \{2, 5, 6\}\}$, then any C' will not be an exact cover (all 3 subsets need to cover all elements in X exactly once)
- note that an exact cover C' will contain exactly q elements. [2]

Knapsack is NP-complete.

Proof idea:

- special case of Knapsack where $v_i = w_i$ and $K = W$
- K is the goal value (sum of values v_i needs to be greater than K)
- W is the maximum bag size (sum of weights w_i has to be smaller than W)
- given: set of n integers $w_i, ..., w_n$ and an integer K
- find: subset of the given integers that sums up exactly to K
-

TODO
Questions:

References

[1] Martin Berglund. *Lecture notes in Computational Complexity.*

[2] *Exact Cover by 3-Sets.* `https://npcomplete.owu.edu/2014/06/10/exact-cover-by-3-sets/`, last opened: 03.12.22.

[3] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, 1994.