Algorithm Theory

Georg Moser Mircea Dan Hernest

Institute of Computer Science @ UIBK

Summer 2007

Content

Reminder: MAXCUT

Definition MAXCUT

- a cut in an undirected graph $G = (V, E)$ is a partition of the nodes in S and $V - S$
- the size of the cut $(S, V - S)$ is the number of edges connecting the sets.

Definition MAXCUT(D)

$$\text{MAXCUT}(D) = \{(G, K) : G \text{ has a cut of size } K \text{ or more}\}$$

Theorem MINCUT

the cut with the smallest possible size is polytime computable

Theorem

MAXCUT(D) is **NP**-complete

Definition Colouring

$$\text{3-COLOURING} = \left\{ G : \begin{array}{l} G \text{ is an undirected graph colourable} \\ \text{with } 3 \text{ colours such that no two adja-} \\ \text{cent nodes have the same colour} \end{array} \right\}$$

the problem to colour G with k colours is called k-COLOURING

Theorem

3-COLOURING is **NP**-complete

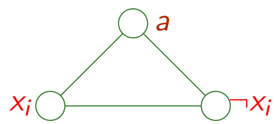
Proof

- easy to see: 3-COLOURING $\in$ **NP**

to show completeness, we reduce from NAESAT; assume $\varphi \in$ 3CNF

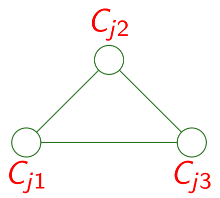
Variable-gadget

1. for each variable x_i we employ:
2. all triangles share the node a



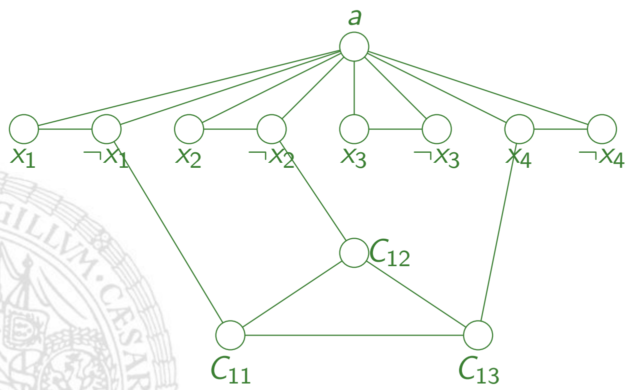
Clause-gadget

1. each clause C_j is represented by
2. and C_{ji} is connected with the i^{th} literal-node of C_j



Example

Construction of $R(\neg x_1 \vee \neg x_2 \vee x_4)$



Lemma

if $\varphi \in$ NAESAT, then $R(\varphi) \in$ 3-COLOURING

Proof

- suppose $T(\varphi) =$ **true**, and T is alternating i.e., T fulfills the NAESAT condition

use colours $\{0, 1, 2\}$:

1. colour node a by colour 2
2. if $T(x_i) =$ **true**, colour the variable-node $(\neg)x_i$ by 1 (0)
   if $T(x_i) =$ **false**, colour $(\neg)x_i$ by 0 (1)
3. assume C_{j1}, C_{j2} to be connected
   to a true and false literal respectively
   colour C_{j1} by 0, C_{j2} by 1
4. the remaining clause-node is coloured by 2 □

Lemma

if $R(\varphi) \in$ 3-COLOURING, then $\varphi \in$ NAESAT

Proof

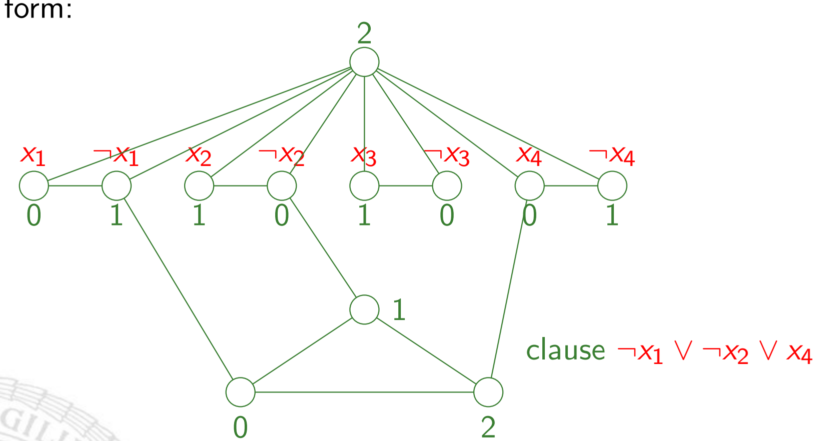
- suppose $R(\varphi)$ can be 3-coloured

rename the colours to $\{0, 1, 2\}$:

1. assume a is coloured by 2
2. if x_i is coloured by 1, set $T(x_i) =$ **true**
   if x_i is coloured by 0, set $T(x_i) =$ **false**
3. the clause-triangles are 3-colourable, only if T admits alternating truth-values in the clauses
4. suppose otherwise:
   if $\exists$ a clause C such that all literals in C get the same truth value (under T)
   then the corresponding clause-triangle cannot be coloured as either 1 or 0 cannot be used □

Example

consider formula $\varphi = \neg x_1 \vee \neg x_2 \vee x_4$; the coloured graph $R(\varphi)$ has the form:



the corresponding assignment T reads

$T(x_1) =$ **false** $T(x_2) =$ **true** $T(x_3) =$ **true** $T(x_4) =$ **false**

Definition — TRIPARTITE MATCHING

given

- $B = \{b_1, \ldots, b_n\}$ — boys
- $G = \{g_1, \ldots, g_n\}$ — girls
- $H = \{h_1, \ldots, h_n\}$ — homes
- $T \subseteq B \times G \times H$

TRIPARTITE MATCHING:

$$\left\{ (B, G, H, T) : \begin{array}{l} T \text{ contains } n \text{ triples, such that for distinct} \\ (b, g, h), (b', g', h') \in T \text{ we have } b \neq b', g \neq g', \\ \text{and } h \neq h' \end{array} \right\}$$

Theorem

TRIPARTITE MATCHING is **NP**-complete

Proof Idea

reduction from 3SAT □

Definition — EXACT COVER BY 3-SETS

EXACT COVER BY 3-SETS:

$$\left\{ (U, F) : \begin{array}{l} F = \{S_1, \ldots, S_n\},\ S_j \subseteq U,\ |U| = 3m, \text{ for some } m \\ \text{and } |S_j| = 3,\ \{S_{j_1}, \ldots, S_{j_m}\} \subseteq F \text{ and } S_{j_i} \cap S_{j_{i'}} = \emptyset, \\ i \neq i' \text{ and } \bigcup_{i=1}^{m} S_{j_i} = U \end{array} \right\}$$

Theorem

EXACT COVER BY 3-SETS is **NP**-complete

Proof Sketch

1. EXACT COVER BY 3-SETS is a generalisation of TRIPARTITE MATCHING
   set $U = B \cup G \cup H$
   and $n = |T|$
2. as reduction we employ the identity reduction □

SET COVERING:

$$\left\{ (U, F, B) : \begin{array}{l} F = \{S_1, \ldots, S_n\},\ S_j \subseteq U,\ \{S_{j_1}, \ldots, S_{j_B}\} \subseteq F \text{ and} \\ \bigcup_{i=1}^{B} S_{j_i} = U \end{array} \right\}$$

- generalises EXACT COVER BY 3-SETS
  set $B = m$
  allow overlaps

SET PACKING:

$$\left\{ (U, F, K) : \begin{array}{l} F = \{S_1, \ldots, S_n\},\ S_j \subseteq U,\ \{S_{j_1}, \ldots, S_{j_K}\} \subseteq F \text{ and} \\ S_{j_i} \cap S_{j_{i'}} = \emptyset, \text{ for } i \neq i' \end{array} \right\}$$

- generalises EXACT COVER BY 3-SETS
  set $K = m$
  drop complete covering

Theorem

SET COVERING, SET PACKING are **NP**-complete

Definition — INTEGER PROGRAMMING

given

- a system of linear inequalities, in n variables
- with integer coefficients
- has the system an integer solution?

Theorem

INTEGER PROGRAMMING is **NP**-complete

Remark

INTEGER PROGRAMMING is powerful in expressing other problems:

- consider SET COVERING
- this problem can be expressed by:
  $A \cdot x \geqslant 1 \qquad \sum_{i=1}^{n} x_i \leqslant B \qquad x_i \in \{0, 1\}$
- A is a $m \times n$-matrix, where $m = |U|$ and A's columns are bit vectors of the sets S_i
- x, 1 are column vectors

Example

Consider $U = \{1,2,3,4\}$, $F = \{\{1,2\},\{3,4\},\{2,3\}\}$, $B = 2$

$$Ax = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \geqslant \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$x_1 + x_2 + x_3 \leqslant 2$$

$$x_1 \in \{0,1\}$$

$$x_2 \in \{0,1\}$$

$$x_3 \in \{0,1\}$$

1. x_i is 1 iff S_i is in the cover
2. the only solution of the equations is $x_1 = x_2 = 1$ and $x_3 = 0$
3. hence $\{\{1,2\},\{3,4\}\}$ builds the cover.

Definition KNAPSACK

given

- a set U, such that $|U| = n$
- for each $i \in [1,n]$: v_i is the value of item i, w_i its weight
- numbers K, W
- $\exists$ subset $S \subseteq U$, $\sum_{i \in S} w_i \leqslant W$, $\sum_{i \in S} v_i \geqslant K$?

Theorem

KNAPSACK is **NP**-complete

Proof Sketch

- it easy to see that KNAPSACK $\in$ **NP**

for completeness, we use reduction from EXACT COVER BY 3-SETS

Special Case:

for all i: $v_i = w_i$ and $K = W$

let $\{S_1, \ldots, S_n\}$ be a family of 3-sets, s.t.
$S_i \subseteq U = \{1,2,\ldots,3m\}$, we look for disjoint sets that cover U

1. consider S_i as bit-vector in $\{0,1\}^{3m}$
2. representing the bit-vectors as integers reduce into the special case of KNAPSACK
3. then binary addition almost simulates disjoint set union

$$U = \{1,2,3,4,5,6\},\ F = \{\{1,2,5\},\{1,3,5\},\{1,5,6\}\}$$

S_1	1	1	0	0	1	0
S_2	1	0	1	0	1	0
S_3	1	0	0	0	1	1
U	1	1	1	1	1	1

4. problem carry
5. use base $n+1$ instead of 2
   i.e. the integer w_i is representing S_i is defined as:

$$w_i = \sum_{j \in S_i} (n+1)^{3m-j} \quad \text{and} \quad K = \sum_{j=0}^{3m-1} (n+1)^j$$

$\square$