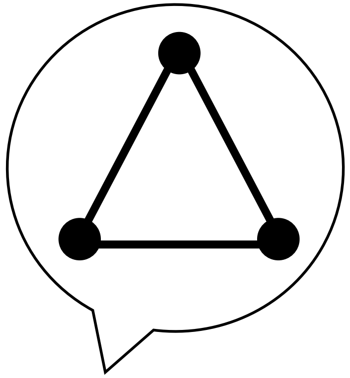
3d matching
knapsack
pseudo-polynomial algorithms
& strong np completeness

dimitris myrisiotis

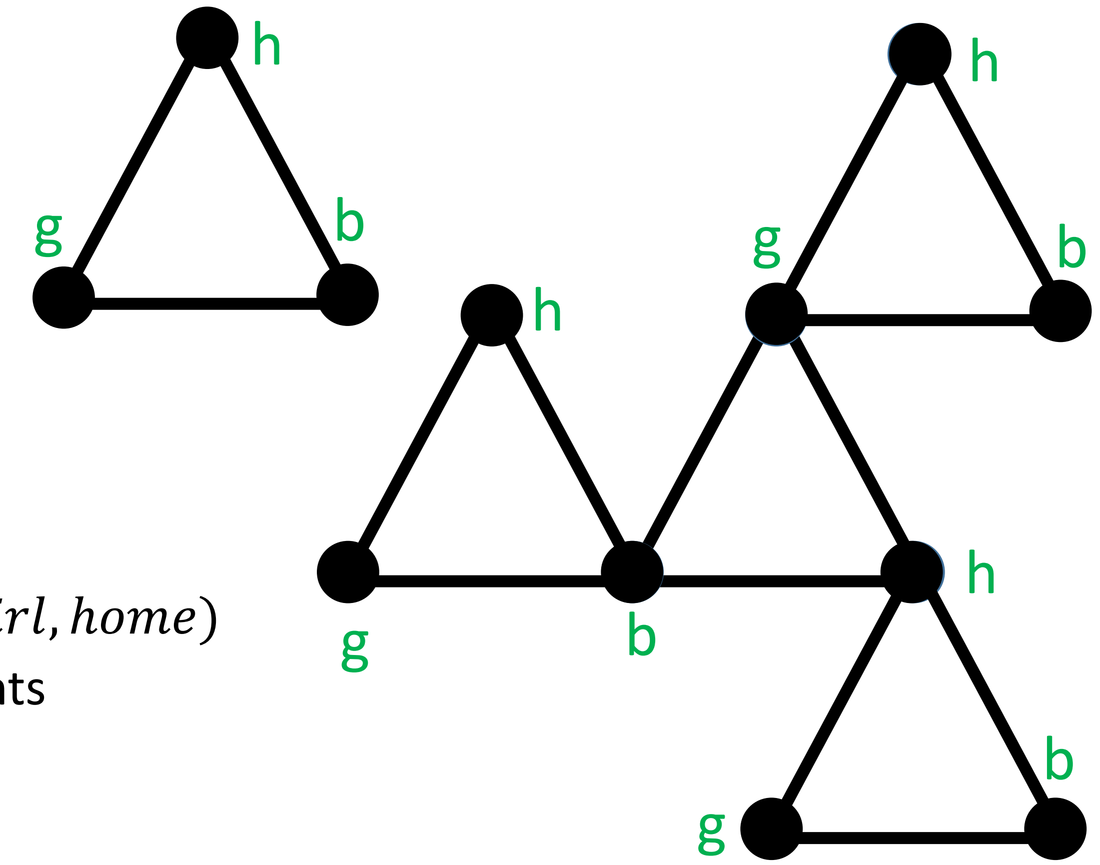
3d matching

3d matching



- input
  - 3 sets
    - boys, girls, homes
    - n elements in each
  - m preferences
    - $(boy, girl, home)$
    - can be viewed as a graph

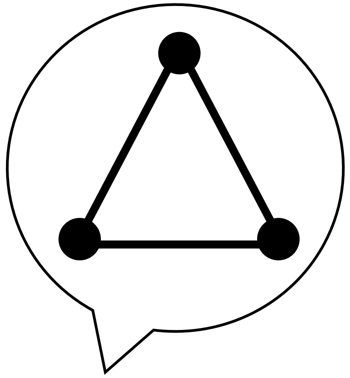
that decomposes to K_3 graphs



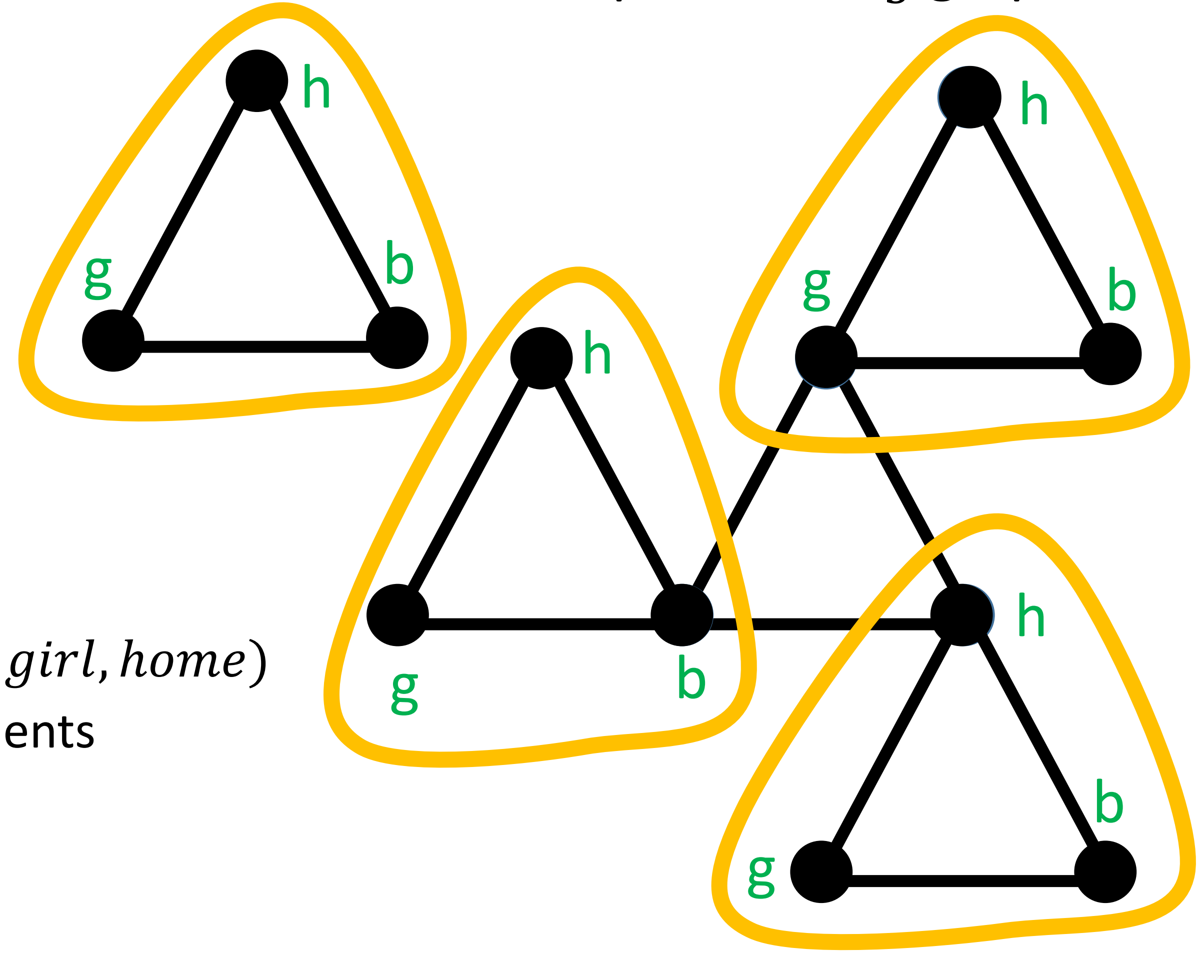
- output
  - yes
    - if there exists a set of n triplets $(boy, girl, home)$ such that, every triplet has unique elements
  - no
    - otherwise

3d matching

- input
  - 3 sets
    - boys, girls, homes
    - n elements in each
  - m preferences
    - $(boy, girl, home)$
    - can be viewed as a graph
- output
  - yes
    - if there exists a set of n triplets $(boy, girl, home)$ such that, every triplet has unique elements
  - no
    - otherwise



that decomposes to K_3 graphs

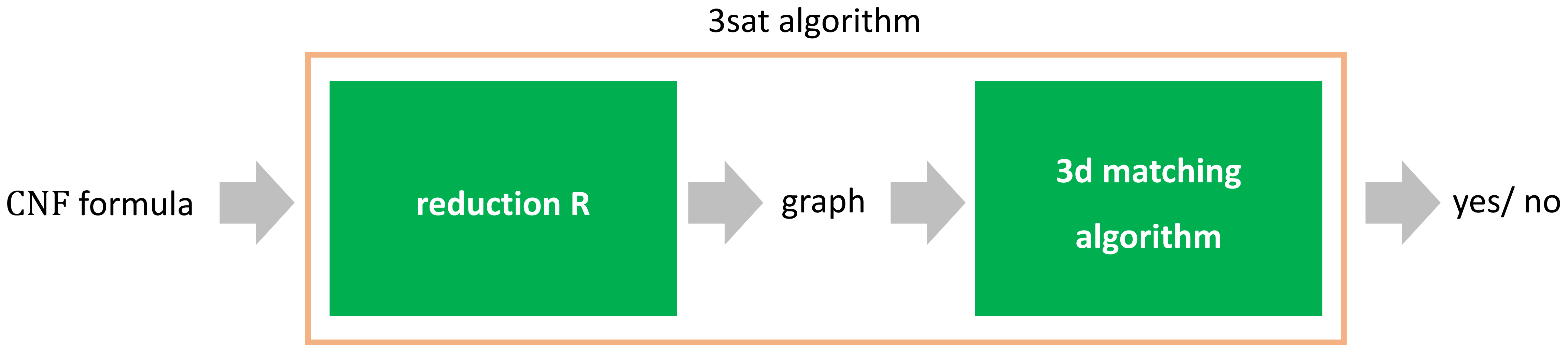


3d matching is np complete

- why?
  - 3d matching $\in$ $\mathbf{NP}$
  - 3sat $\leq$ 3d matching
  - 3sat is $\mathbf{NP}$-complete
- reduction R
  - input
    - a CNF formula
  - output
    - a graph

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_6 \vee x_3) \wedge \ldots \wedge (\neg x_8 \vee \neg x_5 \vee x_7)$$

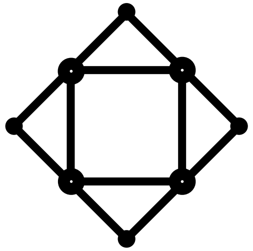
3d matching is np complete

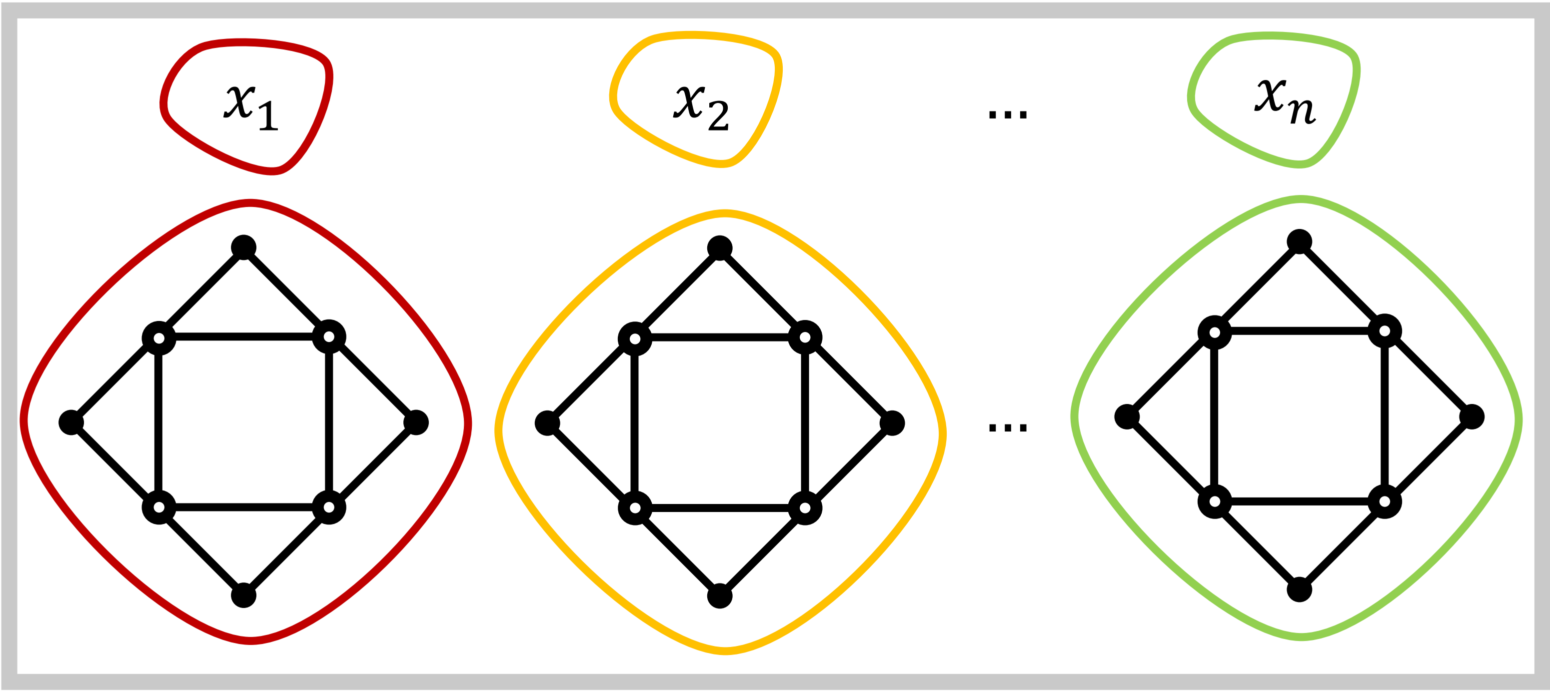


3sat $\leq$ 3d matching

- input
  - CNF formula
    - variables $x_1, x_2, \ldots, x_n$
- output
  - graph
    - for each variable

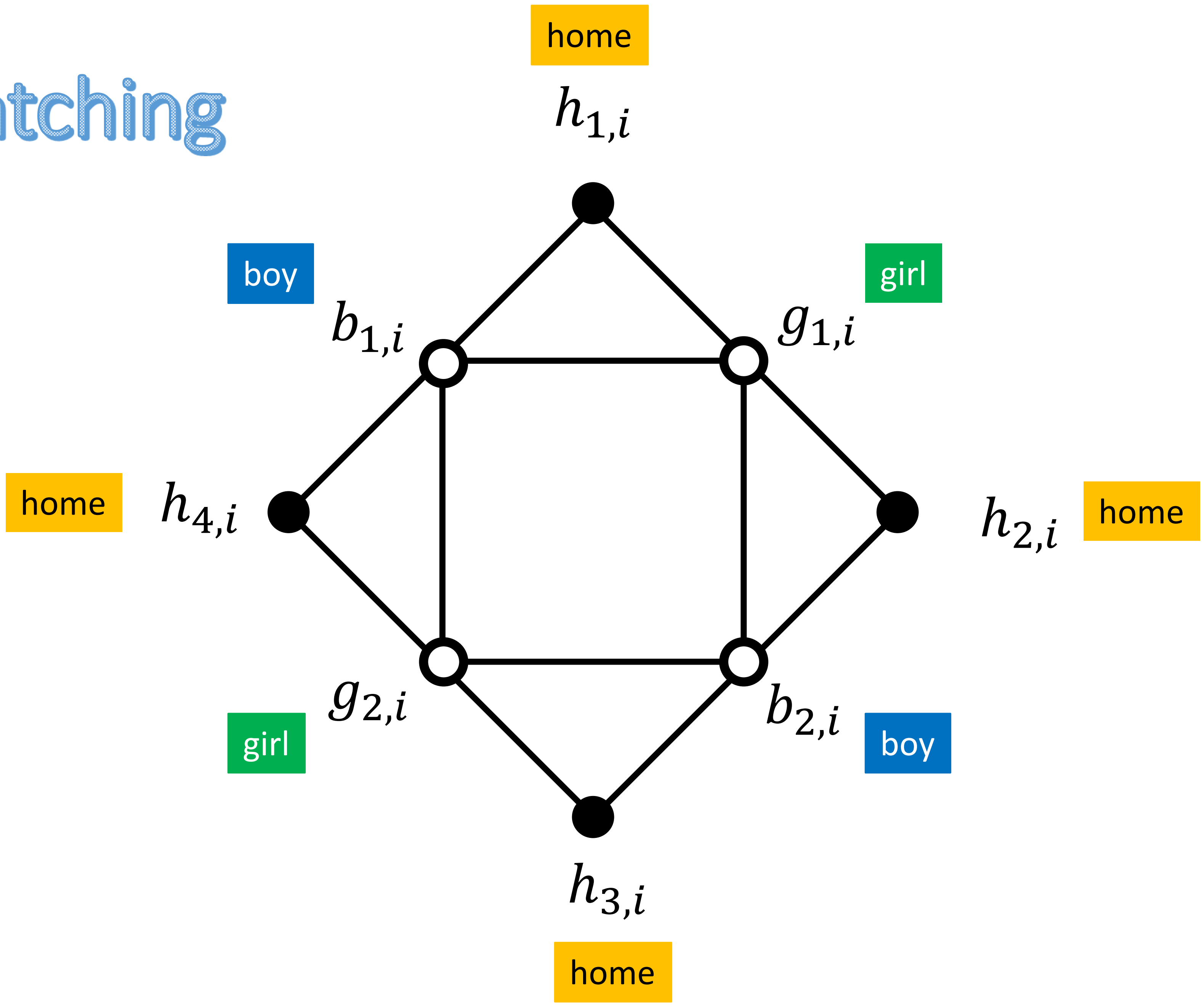
one gadget





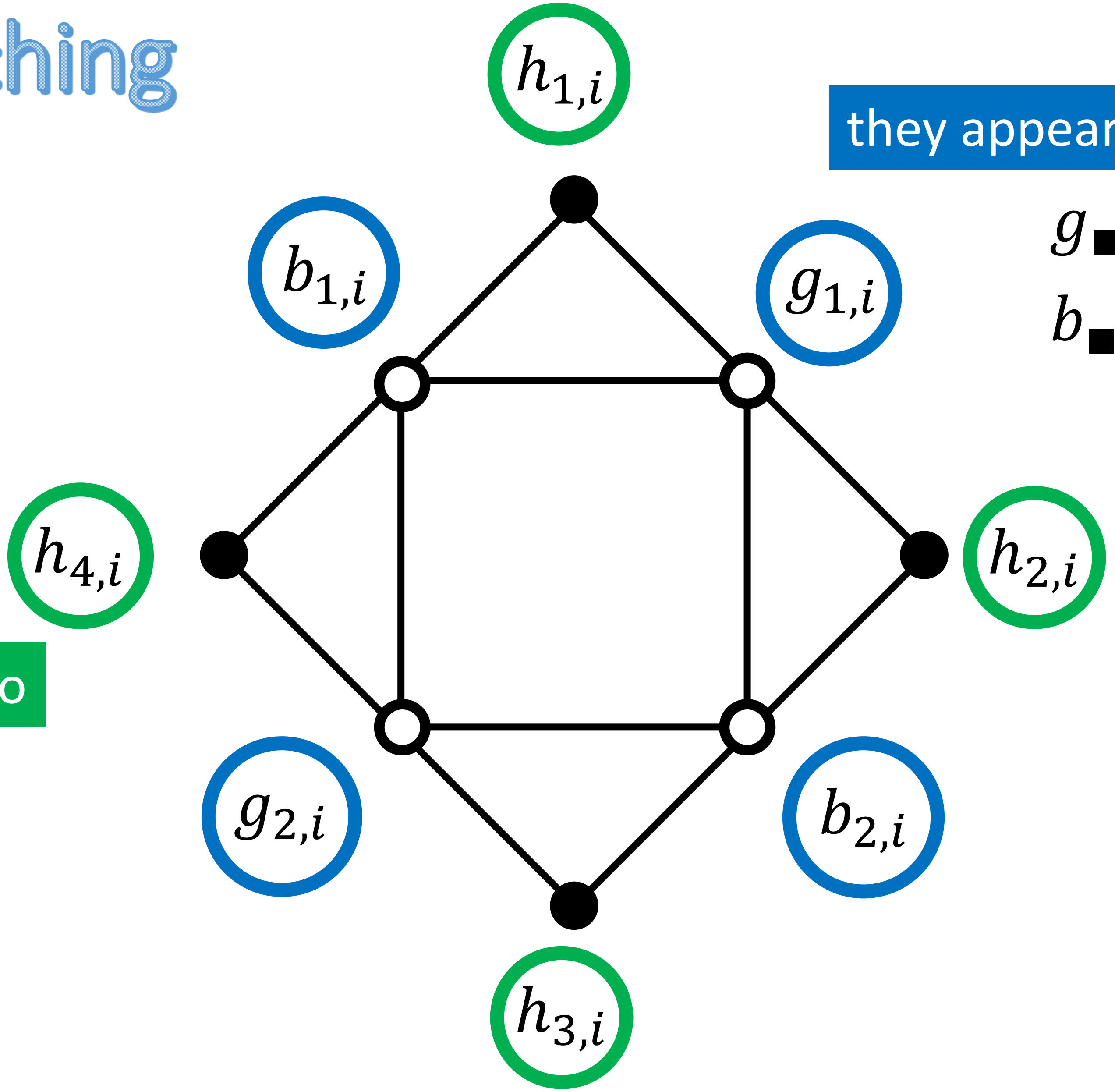
3sat ≤ 3d matching

- gadget for x_i



3sat ≤ 3d matching

- gadget for x_i



they appear only here

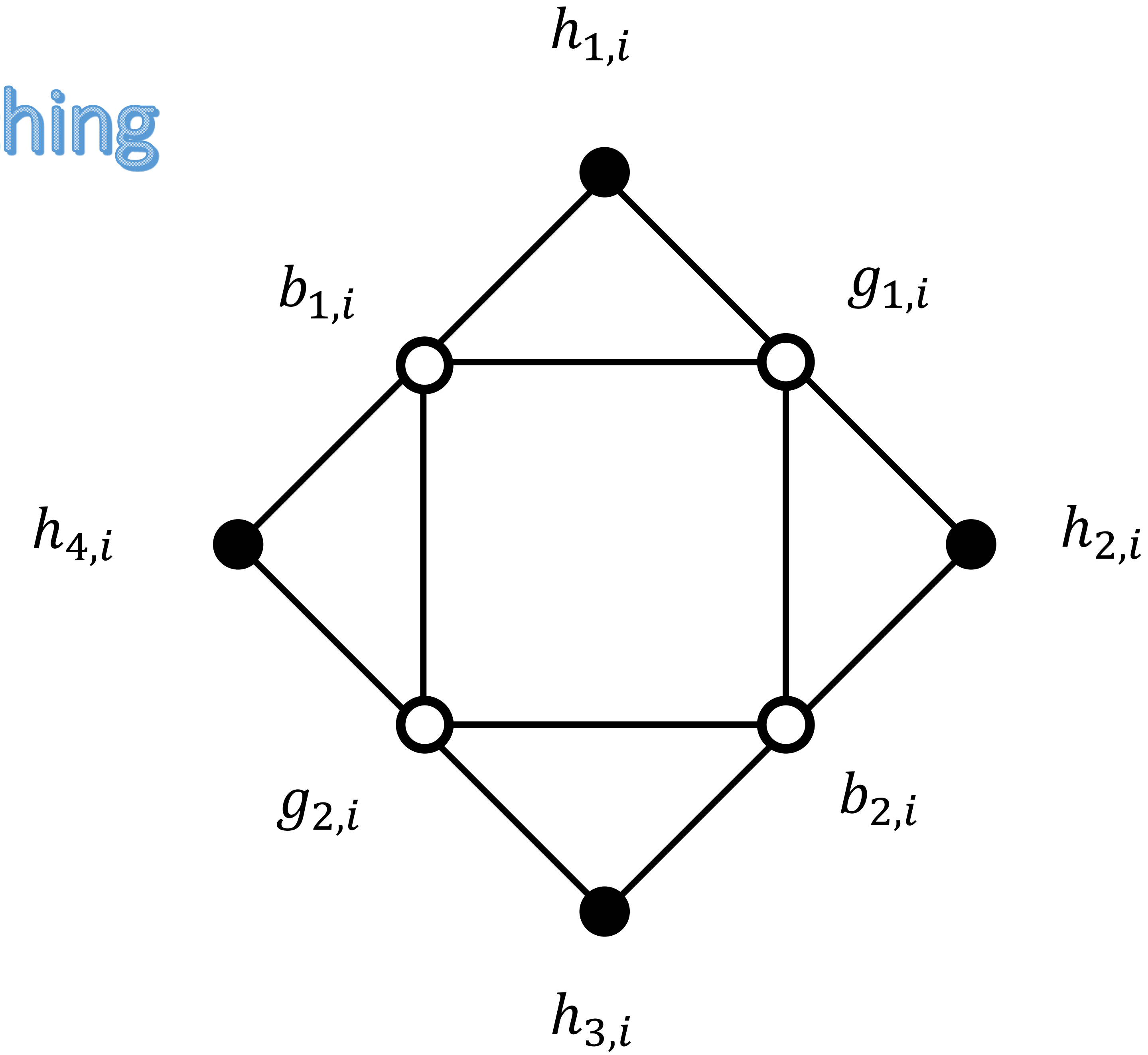
$$g_{\blacksquare,i} \not\equiv g_{\blacksquare,j}$$
$$b_{\blacksquare,i} \not\equiv b_{\blacksquare,j}$$

they appear in other gadgets, too

$$h_{\blacksquare,i} \equiv h_{\blacksquare,j}$$

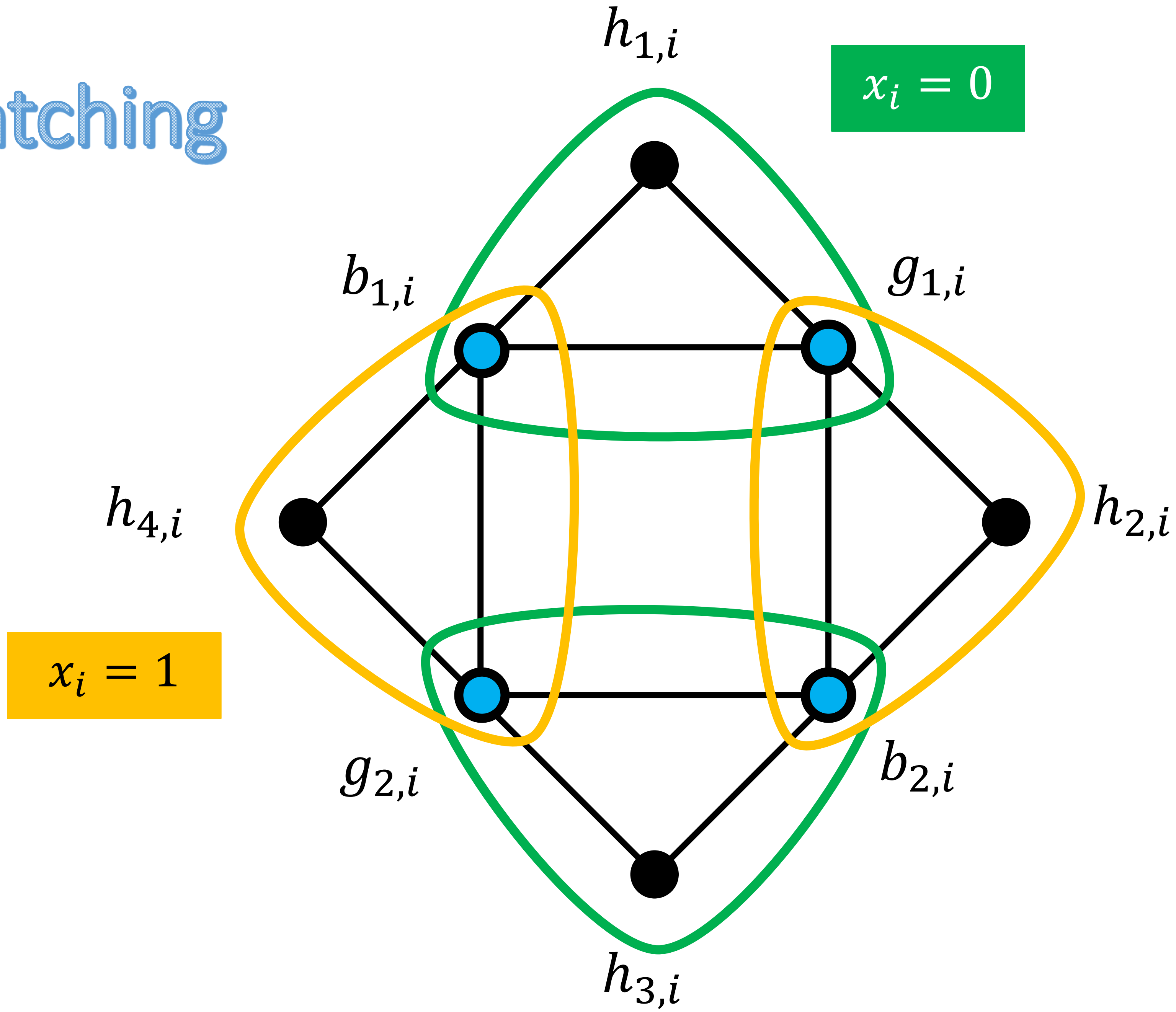
3sat ≤ 3d matching

- gadget for x_i
  - on/ off behavior



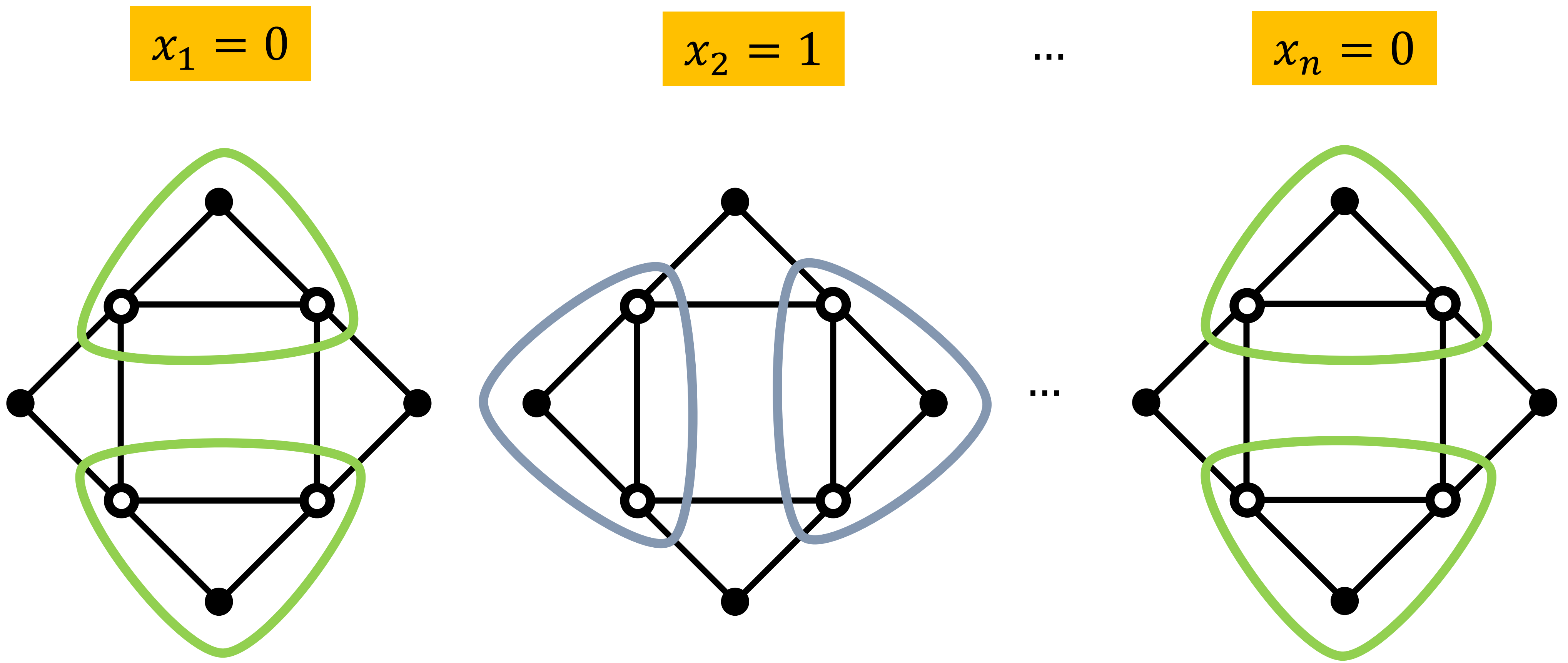
3sat ≤ 3d matching

- gadget for x_i
  - on/ off behavior



3sat ≤ 3d matching

- after we apply the 3d matching algorithm



3sat ≤ 3d matching

- what about the clauses?
- for clause $c = (x_i \vee \neg x_j \vee x_k)$
  - introduce a new boy and a new $girl$,
    - b_c and g_c
  - relate the clause to three triplets
    - $(b_c,\ g_c,\ h_{1,i})$
    - $(b_c,\ g_c,\ h_{2,j})$
    - $(b_c,\ g_c,\ h_{1,k})$
- $c = 1$ if and only if
  $x_i = 1$ or,
  $x_j = 0$ or,
  $x_k = 1$

$$(b_c, g_c, h_{1,i})$$

$$(b_c, g_c, h_{2,j})$$

$$(b_c, g_c, h_{1,k})$$

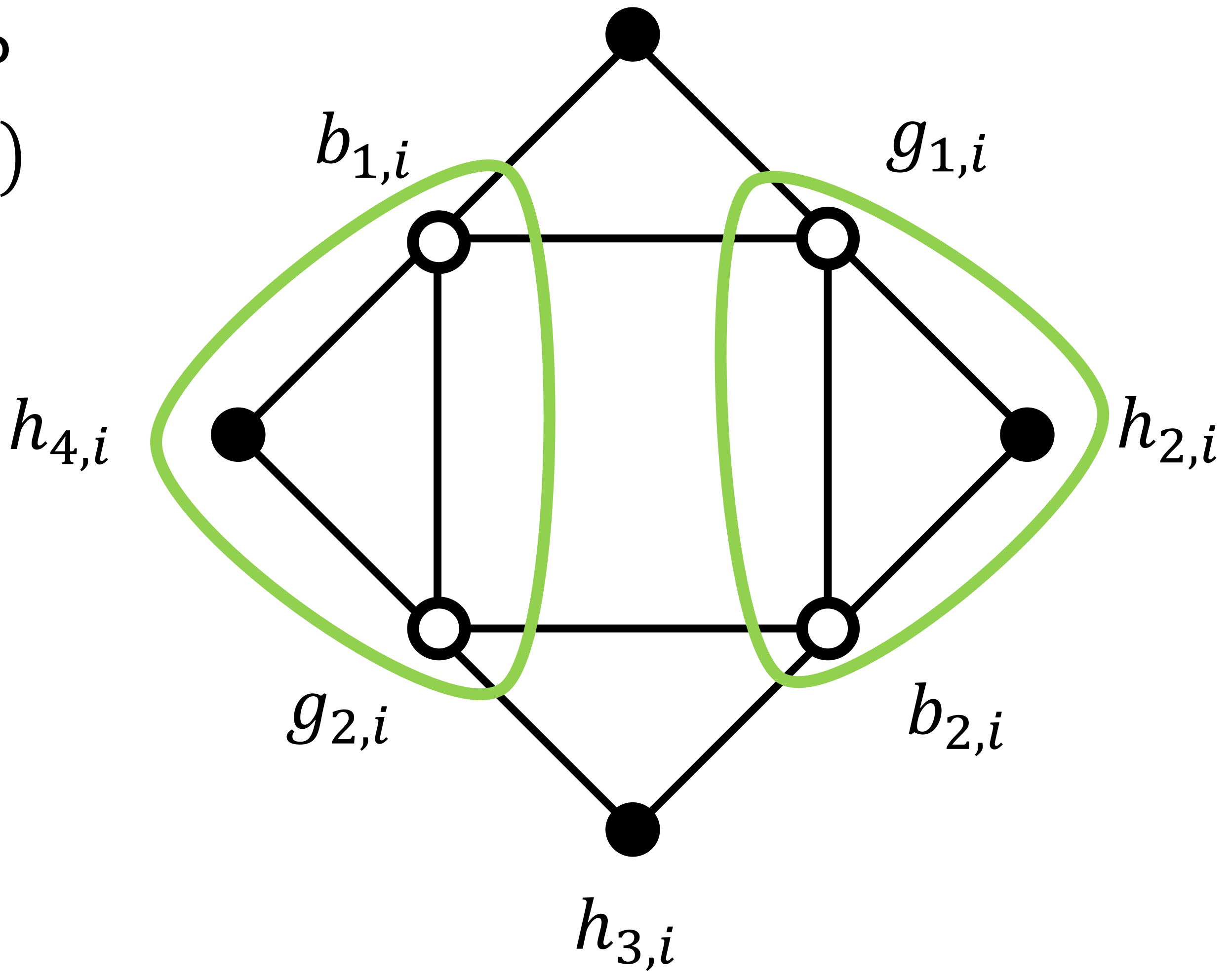
3sat ≤ 3d matching

b_c, g_c → $h_{1,i}$ empty!

- what about the clauses?
  - clause $c = (x_i \vee \neg x_j \vee x_k)$
  - three triplets
    - $(b_c,\ g_c,\ h_{1,i})$
    - $(b_c,\ g_c,\ h_{2,j})$
    - $(b_c,\ g_c,\ h_{1,k})$

$x_i = 1$

$(b_c, g_c, h_{1,i})$

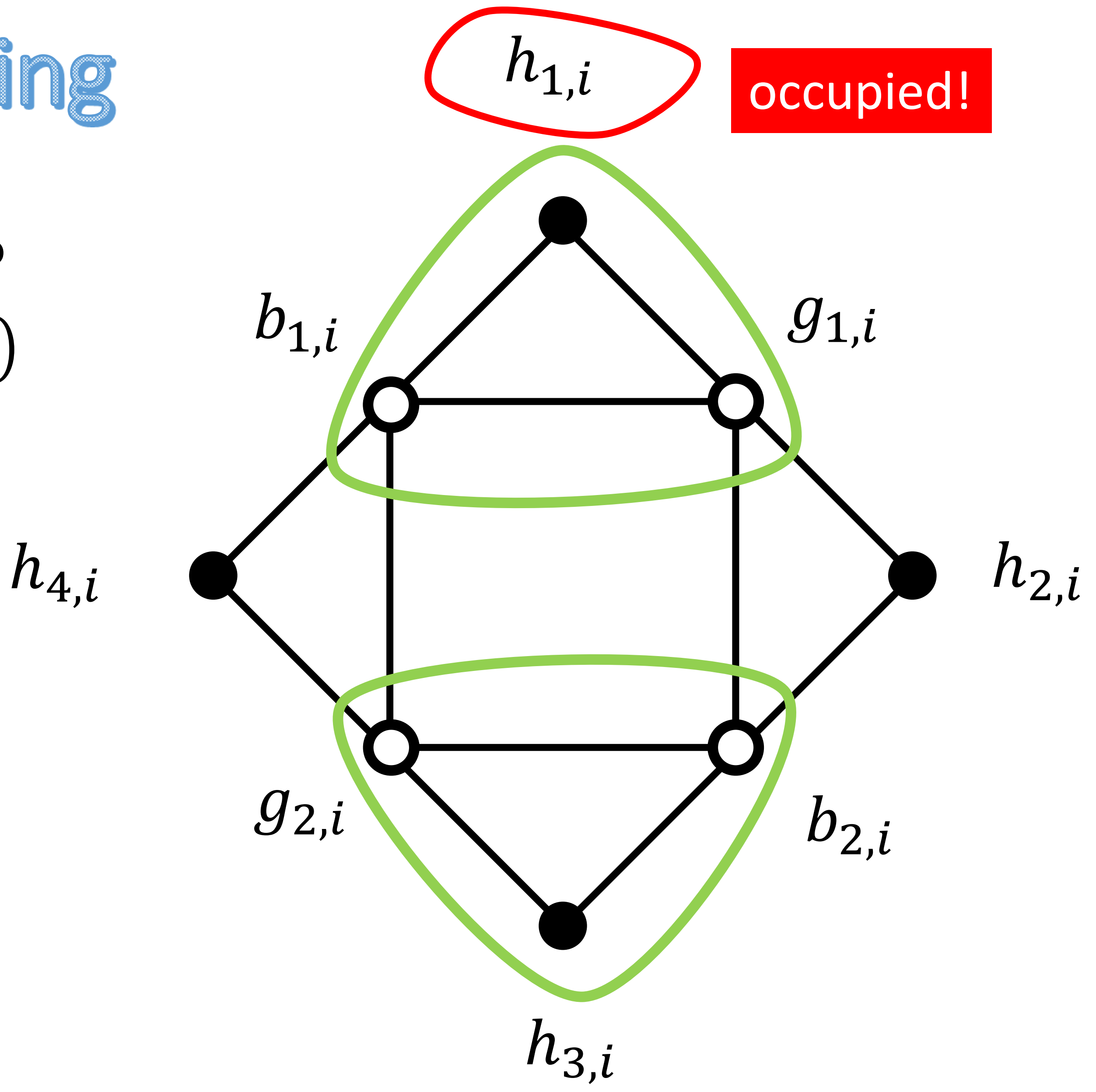


3sat ≤ 3d matching

- what about the clauses?
  - clause $c = (x_i \vee \neg x_j \vee x_k)$
  - three triplets
    - $(b_c,\ g_c,\ h_{1,i})$
    - $(b_c,\ g_c,\ h_{2,j})$
    - $(b_c,\ g_c,\ h_{1,k})$

what if… $x_i = 0$ **?**

$(b_c, g_c, h_{1,i})$

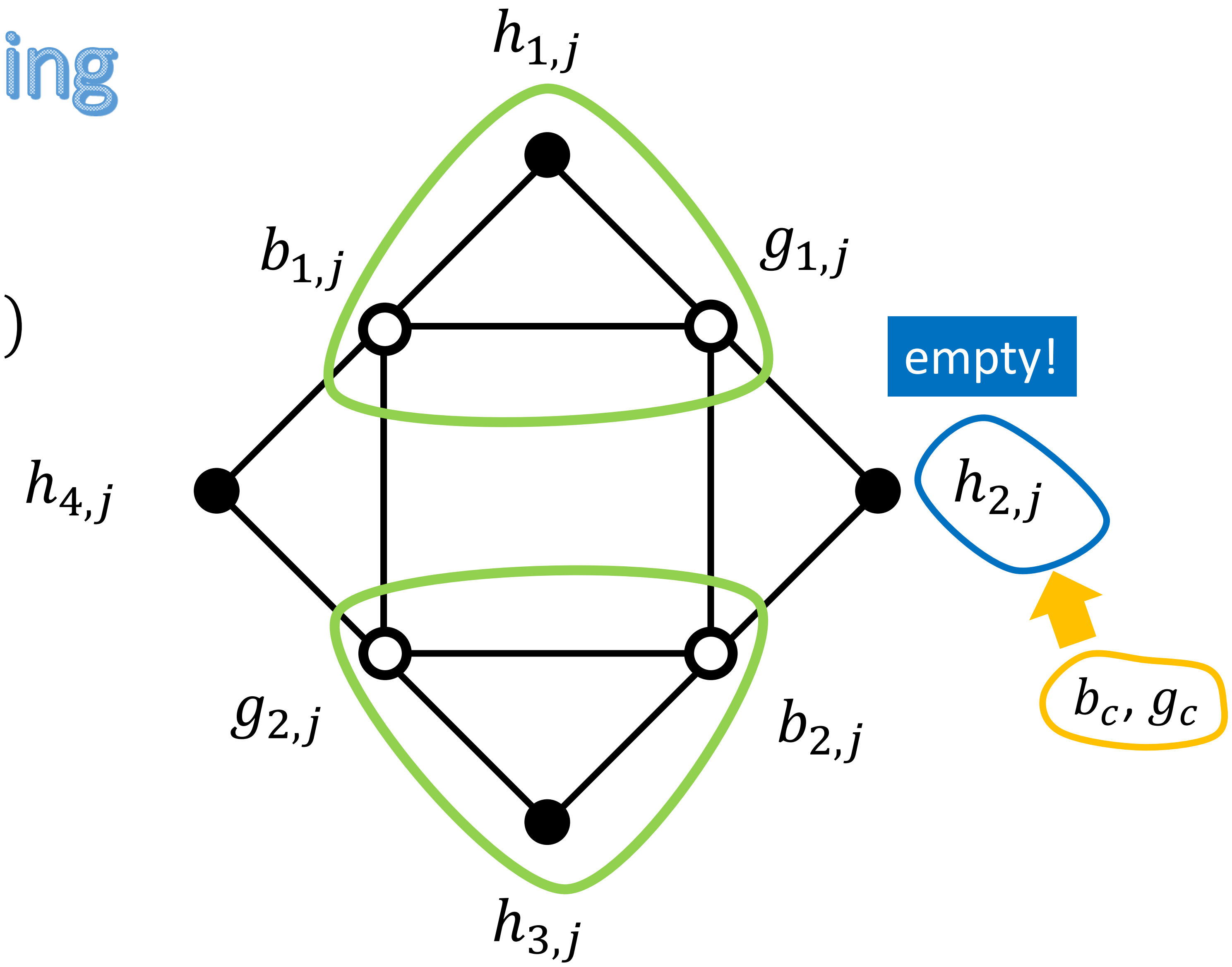


3sat ≤ 3d matching

- what about clauses?
  - clause $c = (x_i \vee \neg x_j \vee x_k)$
  - three triplets
    - $(b_c,\ g_c,\ h_{1,i}\ )$
    - $(b_c,\ g_c,\ h_{2,j}\ )$
    - $(b_c,\ g_c,\ h_{1,k}\ )$

$x_j = 0$

$(b_c, g_c, h_{2,j}\ )$

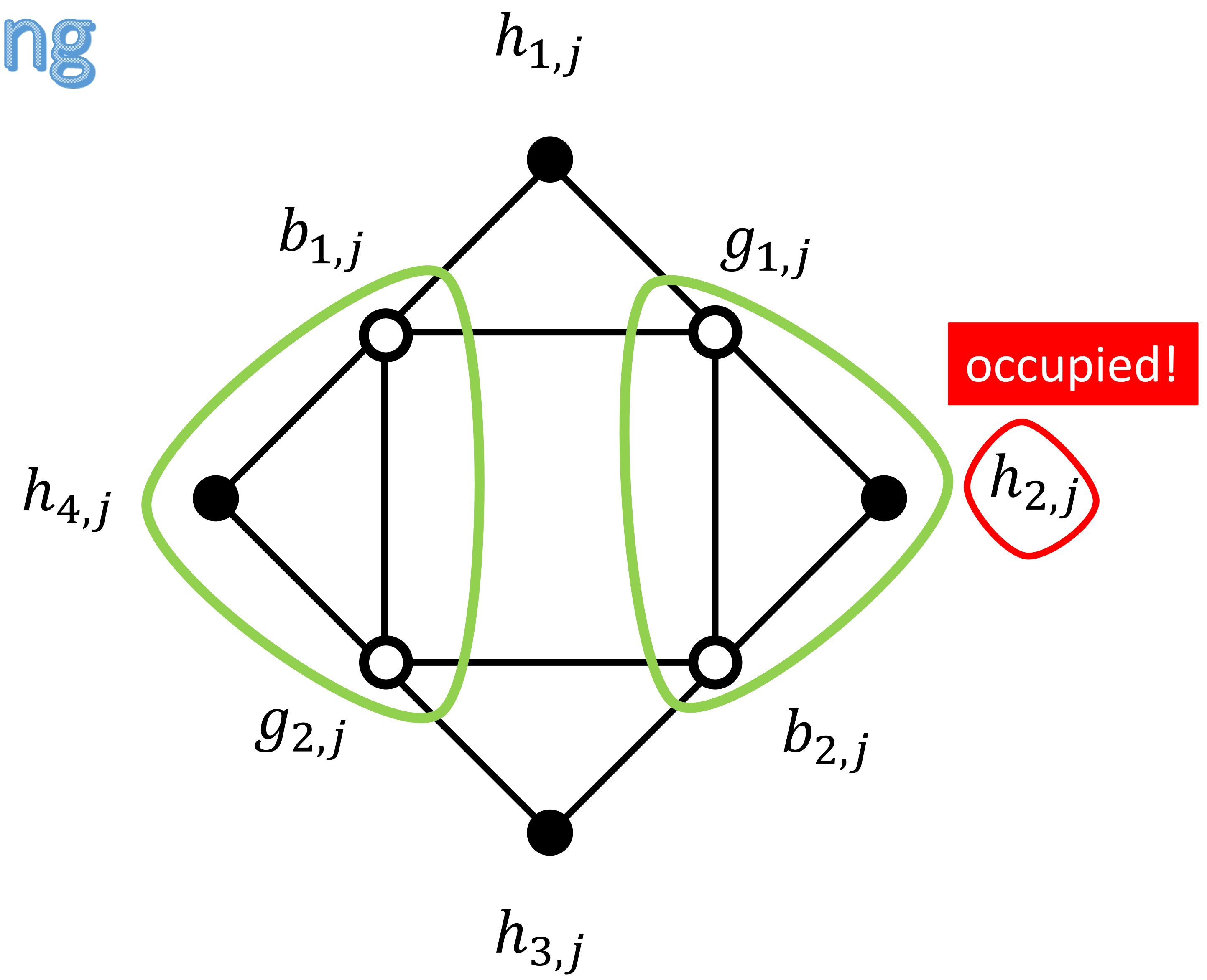


3sat ≤ 3d matching

- what about clauses?
  - clause $c = (x_i \vee \neg x_j \vee x_k)$
  - three triplets
    - $(b_c,\ g_c,\ h_{1,i})$
    - $(b_c,\ g_c,\ h_{2,j})$
    - $(b_c,\ g_c,\ h_{1,k})$

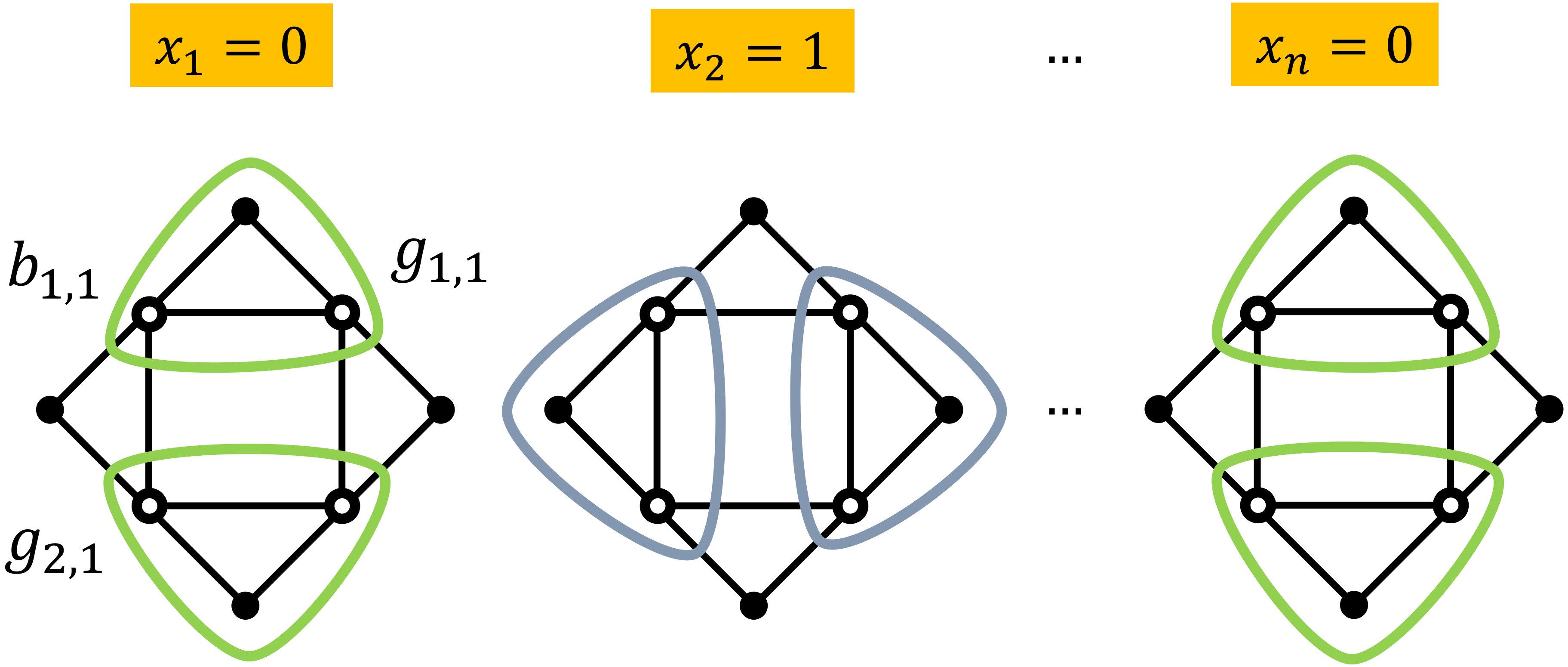
what if… $x_j = 1$ **?**

$(b_c, g_c, h_{2,j})$



3sat ≤ 3d matching

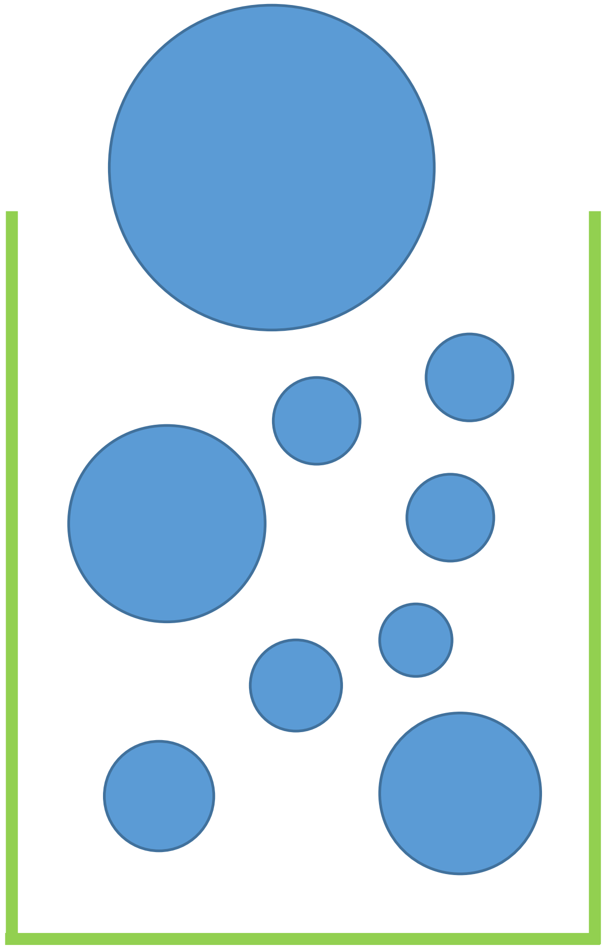
- graph



knapsack

knapsack

- input
  - a set of n items, S
    - each has a weight, w_i
    - each has a value, v_i
  - a number W
    - the maximum weight we can lift
  - a number K
    - the minimum value that satisfies us
- output
  - yes
    - if there is a subset of S that weights at most W and its value is at least K
  - no
    - otherwise



knapsack (special case)

- input
  - a set of integers S
  - a number W
- output
  - yes
    - if there is a subset of S that adds up to W
  - no
    - otherwise

subset-sum!

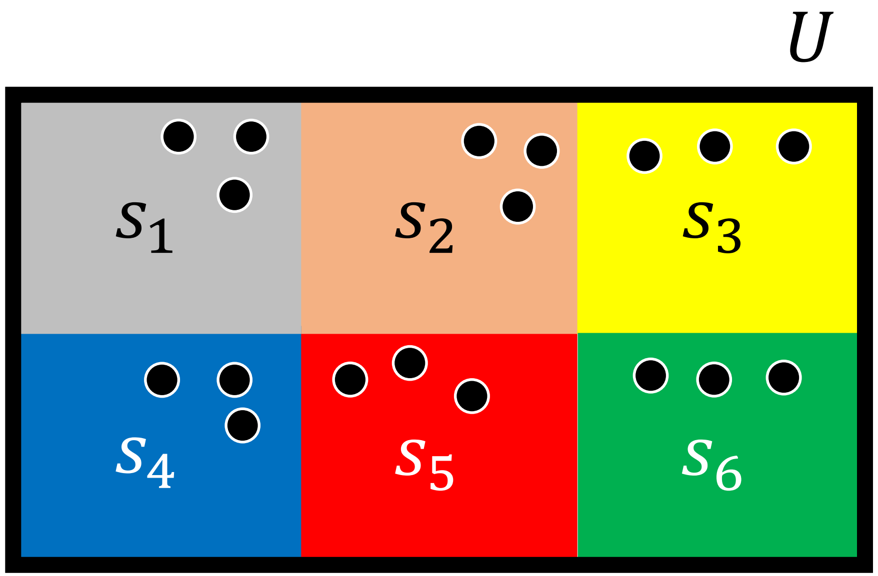
knapsack (special case) is np complete

- why?
  - knapsack $\in \mathbf{NP}$
  - exact cover by 3-sets $\leq$ knapsack
  - exact cover by 3-sets is $\mathbf{NP}$-complete
- exact cover by 3-sets
  - input
    - a set U, $|U| = 3m$
    - a collection F of n subsets of U
      - each subset contains 3 elements
  - output
    - yes
      - if there are m sets in F that are disjoint and have U as their union
    - no
      - otherwise

U

knapsack (special case) is np complete

- why?
  - knapsack $\in \mathbf{NP}$
  - exact cover by 3-sets $\leq$ knapsack
  - exact cover by 3-sets is $\mathbf{NP}$-complete
- exact cover by 3-sets
  - input
    - a set U, $|U| = 3m$
    - a collection F of n subsets of U
      - each subset contains 3 elements
  - output
    - yes
      - if there are m sets in F that are disjoint and have U as their union
    - no
      - otherwise



exact cover by 3-sets ≤ knapsack

- $input(\text{exact cover by } 3 - \text{sets}) \rightarrow input(\text{knapsack})$
  - subset collection F becomes,
  - F'
  - F' is a collection of n strings of $\{0,1\}^{3m}$
    - each respective to a subset of F
    - 011001 is respective to the subset that contains the $\mathbf{2^{nd}}$, the $\mathbf{3^{rd}}$ and the $\boldsymbol{last}$ element of U,

    if U is a 6-element set
  - set union becomes,
    - number addition

exact cover by 3-sets $\leq$ knapsack

- solve knapsack,
  - for set F' and for number $W = 2^{3m} - 1$
  - m subsets that cover $U \rightarrow m$ numbers that add up to $2^{3m} - 1$
    - number $2^{3m} - 1 \leftrightarrow$ string $1^{3m} = 11111 \ldots 1$
- but, there is a bug!
  - $U = \{1,2,3,4\}$

$\{3,4\}$, $\{2,4\}$, $\{2,3,4\}$

$0011 + 0101 + 0111 = 1111$

$\{3,4\} \cup \{2,4\} \cup \{2,3,4\} = \{2,3,4\} \neq \{1,2,3,4\}$

change

representation

base to

$3 \cdot m + 1$

pseudo-polynomial algorithms

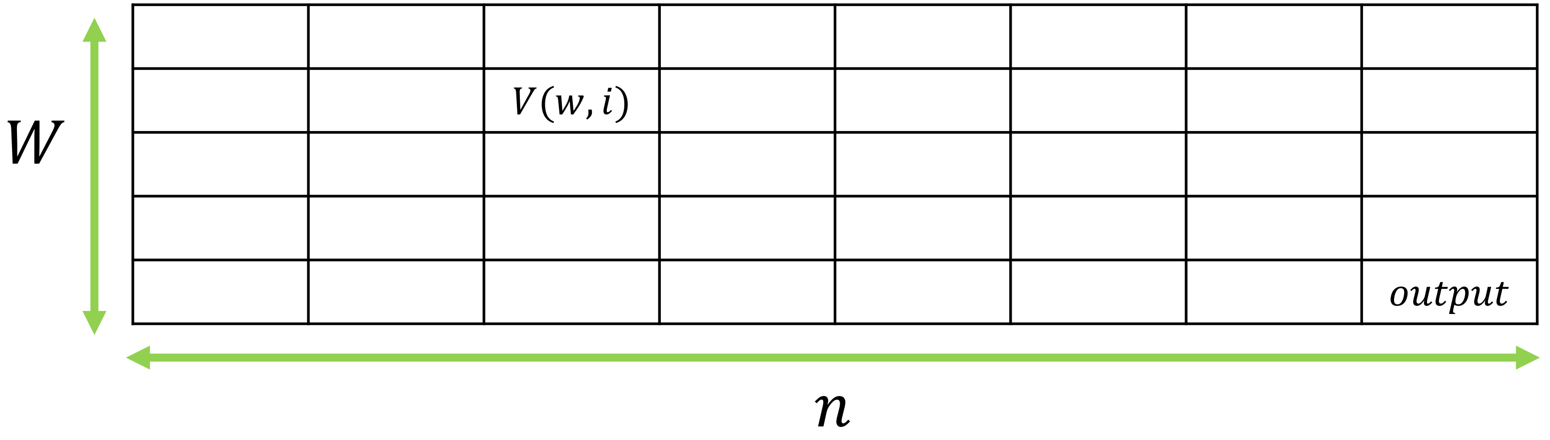
pseudo-polynomial algorithms

- solving knapsack
  - via dynamic programming
    - using this,

$$V(w, i+1) = \max\{V(w,i), v_{i+1} + V(w - w_{i+1}, i)\}$$

$$0 \le w \le W, 0 \le i \le n$$

- Also $V(w,0) = 0, \forall w$



$n \cdot W$

pseudo-polynomial algorithms, solving knapsack

- $V(w, i+1) = \max\{V(w,i), v_{i+1} + V(w - w_{i+1}, i)\}$

  $V(w,i) =$

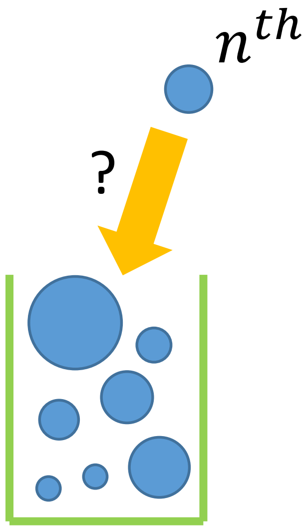
  maximum attainable value using i items that weight at most w

- what happens at the end?
- $V(W,n) = \max\{V(W,n-1), v_n + V(W - w_n, n-1)\}$
  - item n is not chosen

    $V(W,n) = V(W,n-1)$

  - item n is chosen

    $V(W,n) = v_n + V(W - w_n, n-1)$



pseudo-polynomial algorithms

- so, we can solve knapsack in $n \cdot W$ steps
- but, is $n \cdot W$ polynomial time?
  - actually, no
  - input
    - $n + 1$ numbers: $item_1, item_2, \ldots, item_n$ and W
  - input size
    - $\log(item_1) + \log(item_2) + \cdots + \log(item_n) + \log W$
  - $\log(item_1) = \log(item_2) = \cdots = \log(item_n) = \log W$
  - input size

$$(n+1) \cdot \log W = \Theta(n \cdot \log W)$$

- complexity **as a function of the input size** $n \cdot W = n \cdot 2^{\log W}$

pseudo-polynomial algorithms

- so, we can solve knapsack in $n \cdot W$ steps
- input size

$$\Theta(n \cdot \log W)$$

- complexity **as a function of the input size** $n \cdot 2^{\log W}$
- increase in n
  - more items in the set S
  - linear increase in performance cost
- increase in $\log W$
  - more representation bits
  - exponential increase in performance cost!

strong np-completeness

strong np completeness

- strongly **NP**-complete problems
  - the problems that remain **NP**-complete even if

$$\forall \text{instance}, \forall \text{number} :$$

$$\text{number} \in \text{instance} \rightarrow \text{number} \leq \text{poly}(\text{ size}(\text{ instance }))$$

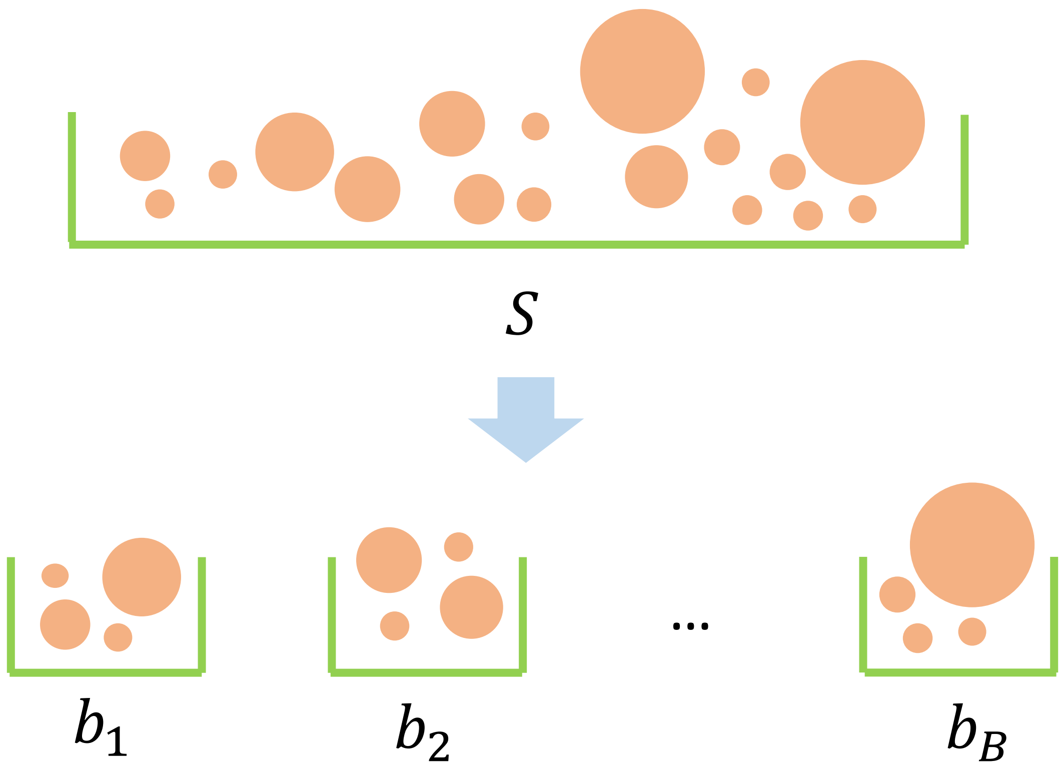
where,

$$\text{size}(\blacksquare) = \#\text{ of representation bits of } \blacksquare$$

- knapsack is not strongly **NP**-complete
  - we created $\mathit{exponentially\ large}$ numbers in its reduction
- bin packing is strongly **NP**-complete!

bin packing

bin packing



- input
  - a set of n integers, S
    - a set of items
  - an integer B
    - bins
  - an integer C
    - bin capacity
- output
  - yes
    - if S can be partitioned into B subsets such that, the total sum of each subset is at most C
  - no
    - otherwise

bin packing is strongly np complete

- why?
  - bin packing $\in$ $\mathbf{NP}$
  - 3d matching $\leq$ bin packing
  - 3d matching is $\mathbf{NP}$-complete
  - the numbers that we create for our reduction are at most,

$$(\,\text{size}(\,\text{input}\,)\,)^4$$

3d matching $\leq$ bin packing

- $input(\text{3d matching}) \rightarrow input(\text{bin packing})$
- $input(\text{3d matching})$
  - n boys, n girls, n homes and m triplets
- $input(\text{bin packing})$
  - m bins
  - $4m$ items
    - one for each triplet
    - one for each boy, girl or home occurrence in the triplets

3d matching $\leq$ bin packing

- what are the $4m$ items?
  - $b_i[q]$
    - $b_2[5] \rightarrow 5^{th}$ occurrence of boy b_2
  - $g_j[q]$
    - girl occurrences
  - $h_k[q]$
    - home occurrences
  - t_l
    - triplet occurrences
    - each triplet appears only once!

m triplets
m boy occurrences
m girl occurrences
m home occurrences

3d matching ≤ bin packing

- what are the item sizes?

$M = \Theta(n)$

	item	size
first occurrence	$b_i[1]$	$10M^4 + iM + 1$
	$b_i[q]$	$11M^4 + iM + 1$
first occurrence	$g_j[1]$	$10M^4 + jM^2 + 2$
	$g_j[q]$	$11M^4 + jM^2 + 2$
first occurrence	$h_k[1]$	$10M^4 + kM^3 + 4$
	$h_k[q]$	$8M^4 + kM^3 + 4$
	$t_l = (b_i, g_j, h_k)$	$10M^4 + 8 - iM - jM^2 - kM^3$

- $bin\ capacity = C = 40M^4 + 15$

3d matching ≤ bin packing

- suppose there is a way to fit these items into m bins,

then,

- there is 3d matching in the sets of boys, girls and homes!

why?

3d matching $\leq$ bin packing

- we have that,
  - each bin is full
  - each bin contains exactly 4 items
    - a triplet and,
    - its 3 respective elements (b, g, h)
  - these 3 respective elements are either all first occurrences or,

    $none$ is a first occurrence
- the bins that contain the **first occurrences** compose a 3d matching
  - because $only$ these bins have $unique$ elements!

3d matching $\leq$ bin packing

- some questions
  - why are the bins full?
  - why each bin contains 4 elements?
  - why each bin contains a triple, a boy, a girl and a home?
  - why each triple is contained in a common bin with its $respective$ boy, girl and home?
  - why in each bin the $boy - girl - home$ items are **either all first** occurrences **or none** is a **first** occurrence?

3d matching ≤ bin packing

- why are all the bins full?

  $total\ item\ weight = m \cdot bin\ capacity = total\ capacity$

  - $total\ item\ weight = \boldsymbol{boys}\ total\ size + \boldsymbol{girls}\ total\ size + \boldsymbol{homes}\ total\ size + \boldsymbol{triplets}\ total\ size$
  - $boys\ total\ size = \sum \sum size(b_i[j])$
    - similarly for girls, homes and triplets
  - we supposed that the items fit in the bins
    - so, every bin is full!

3d matching $\leq$ bin packing

- why each bin contains 4 elements?

because each item has

size $\sim \frac{1}{4} \cdot (bin\ capacity)$

3d matching ≤ bin packing

- why each bin contains a triple, a boy, a girl and a home?
  - triplet (b_i, g_j, h_k)
  - $b_i[1]$
  - $\boldsymbol{b}_j[1]$ two boys
  - $h_k[1]$
  - $total\ size =$

$$C = 40M^4 + 15$$

$(10M^4 + 8 - iM - jM^2 - kM^3) + (10M^4 + iM + 1) +$

$+ (10M^4 + j\boldsymbol{M} + \mathbf{1}) + (10M^4 + kM^3 + 4)$

$= (10M^4 + 10M^4 + 10M^4 + 10M^4) + (iM - iM) + (j\boldsymbol{M} - jM^2) + (kM^3 - kM^3) +$

$(8 + 1 + \mathbf{1} + 4) = 40M^4 + \mathbf{14} + (j\boldsymbol{M} - jM^2) \neq C$

3d matching ≤ bin packing

- why each triple is contained in a common bin with its *respective* boy, girl and home?
  - triplet (b_i, g_j, h_k)
  - $b_i[q]$
  - $g_f[q']$ mismatched girl
  - $h_k[q'']$
  - $total\ size =$

    $(10M^4 + 8 - iM - jM^2 - kM^3) + (11M^4 + iM + 1) +$

    $+(11M^4 + fM^2 + 2) + (8M^4 + kM^3 + 4)$

    $= (10M^4 + 11M^4 + 11M^4 + 8M^4) + (iM - iM) + (fM^2 - jM^2) + (kM^3 - kM^3) +$

    $(8 + 1 + 2 + 4)$

    $= 40M^4 + 15 + (fM^2 - jM^2) \neq C$

$$C = 40M^4 + 15$$

3d matching ≤ bin packing

- why, in each bin, the $boy–girl–home$ items are either all first occurrences or none is a first occurrence?
  - triplet (b_i, g_j, h_k)
  - $b_i[1]$
  - $g_j[1]$ first occurrences
  - $h_k[1]$
  - $total\ size =$

$$(10M^4 + 8 - iM - jM^2 - kM^3) + (10M^4 + iM + 1) +$$

$$+ (10M^4 + jM^2 + 2) + (10M^4 + kM^3 + 4)$$

$$= (10M^4 + 10M^4 + 10M^4 + 10M^4) + (iM - iM) + (jM^2 - jM^2) + (kM^3 - kM^3) +$$

$$(8 + 1 + 2 + 4) = 40M^4 + 15 = C$$

$$C = 40M^4 + 15$$

3d matching ≤ bin packing

- why, in each bin, the $boy–girl–home$ items are either all first occurrences or none is a first occurrence?
  - triplet (b_i, g_j, h_k)
  - $b_i[q]$
  - $g_j[q']$ $\quad$ $non - first$ occurrences
  - $h_k[q'']$
  - $total\ size =$

$(10M^4 + 8 - iM - jM^2 - kM^3) + (11M^4 + iM + 1) +$

$+(11M^4 + jM^2 + 2) + (8M^4 + kM^3 + 4)$

$= (10M^4 + 11M^4 + 11M^4 + 8M^4) + (iM - iM) + (jM^2 - jM^2) + (kM^3 - kM^3) +$

$(8 + 1 + 2 + 4)$

$= 40M^4 + 15 = C$

$$C = 40M^4 + 15$$

3d matching ≤ bin packing

- why, in each bin, the $boy\text{–}girl\text{–}home$ items are either all first occurrences or none is a first occurrence?
  - triplet (b_i, g_j, h_k)
  - $b_i[q]$
  - $g_j[1]$ mixed occurrences
  - $h_k[q'']$
  - $total\ size =$

$$(10M^4 + 8 - iM - jM^2 - kM^3) + (11M^4 + iM + 1) +$$
$$+(10M^4 + jM^2 + 2) + (8M^4 + kM^3 + 4)$$
$$= (10M^4 + 11M^4 + 10M^4 + 8M^4) + (iM - iM) + (jM^2 - jM^2) + (kM^3 - kM^3) +$$
$$(8 + 1 + 2 + 4)$$
$$= 39M^4 + 15 \neq C$$

$$C = 40M^4 + 15$$

final remarks (on strong np completeness)

- if we had a pseudo-polynomial algorithm for bin packing,

then,

- we would have a polynomial algorithm for it!

  because,

  the numbers that emerged in the reduction from 3d matching to bin packing are $polynomially\ large$ in the input length

$$M^4 = \Theta(n^4)$$

- and, since bin packing is strongly $\mathbf{NP}$-complete, we would get,

$$\mathbf{P} = \mathbf{NP}$$

thank you!