Topic IV.1: Binary Tensors

Discrete Topics in Data Mining
Universität des Saarlandes, Saarbrücken
Winter Semester 2012/13

Topic IV.1: Binary Tensors

1. Closed Itemsets on Tensors
    1.1. Definitions
    1.2. Data-Peeler
2. Tiling on Tensors
    2.1. Tiling as a Tensor CP Decomposition
3. Boolean Tensor Decompositions
    3.1. Boolean Matrix Factorization
    3.2. Boolean vs. Normal Decompositions

Closed Itemsets on Tensors

- Closed itemsets on a binary matrix:
  - Combinatorial submatrix
  - All elements are 1
  - Adding any column would mean we would have to remove row(s) to satisfy the above requirements
    - And same holds for adding a row
- Closed itemsets on a binary (3-way) tensor:
  - Combinatorial subtensor
  - All elements are 1
  - Adding any fibre (on any mode) would mean we would have to remove fibre(s) from other modes to satisfy the above requirements

Cerf, Besson, Robardet & Boulicaut 2009

Some Constraints

- *Mode-wise minimum size*
  - Similar to standard minimum frequency
  - Monotonic for each mode
- *Minimum volume*
  - Similar to above (but not equivalent)
  - Monotonic for each mode
- δ-*isolated*
  - The fraction of 1s in any mode-i fibre passing thru the sub-tensor that are outside it must be more than δ
    - $\delta = 1 \Rightarrow$ all 1s in all fibres must be in the sub-tensor

3D Market Baskets?

- Why mine closed subtensors?
- Market basket data
  - Customers-by-products-by-shops
    - Good for large chains with different types of shops
- Anything-by-anything-by-time
  - Though looses the temporal autocorrelation
- Source IP-by-destination IP-by-destination port
  - Network data analysis
- …

Finding the Closed *n*-Way Itemsets

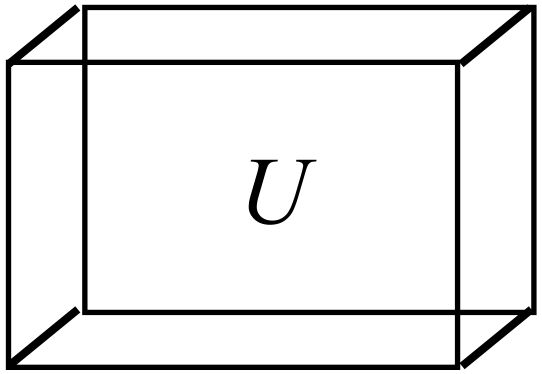
- Similar to traditional closed itemset mining, we want to find *all* itemsets satisfying our constraints
  - There are 2^{I+J+K} possible sets in I-by-J-by-K tensor
  - We hope we can prune the search space…
- The algorithm we're going to discuss is called *Data-Peeler*
- We represent our search space as a tree
  - Root represents all possible *n*-way itemsets
  - The leaves are the closed *n*-way itemsets
  - This tree we want to prune

The Enumeration Tree

- Every node contains two collections of index sets, U and V
  - Index sets define subtensors
- Every node represents all subtensors that contain U and are contained in $U \cup V$
  - The union is over the index sets
  - The root has empty U
  - The leaves have empty V
- It is possible that these tensors are *reduced*
  - Some modes are 0-dimensional

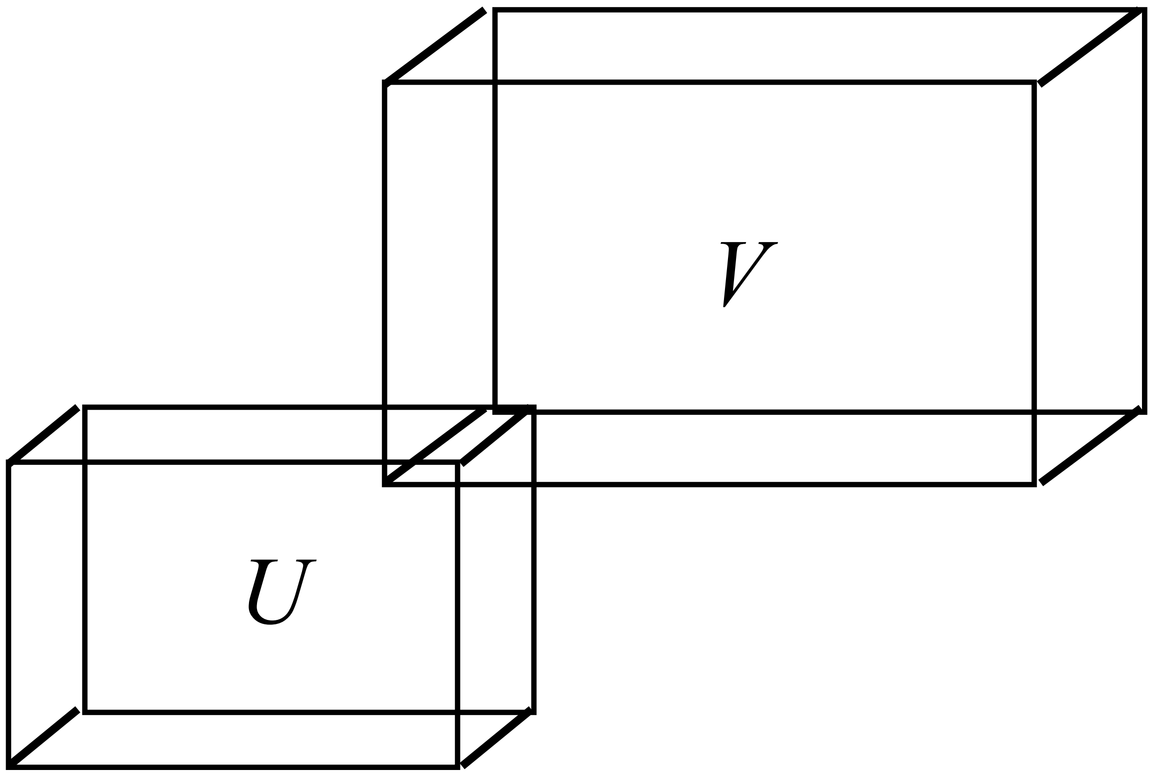
The Enumeration Tree

- Every node contains two collections of index sets, U and V
  - Index sets define subtensors
- Every node represents all subtensors that contain U and are contained in $U \cup V$
  - The union is over the index sets
  - The root has empty U
  - The leaves have empty V
- It is possible that these tensors are *reduced*
  - Some modes are 0-dimensional



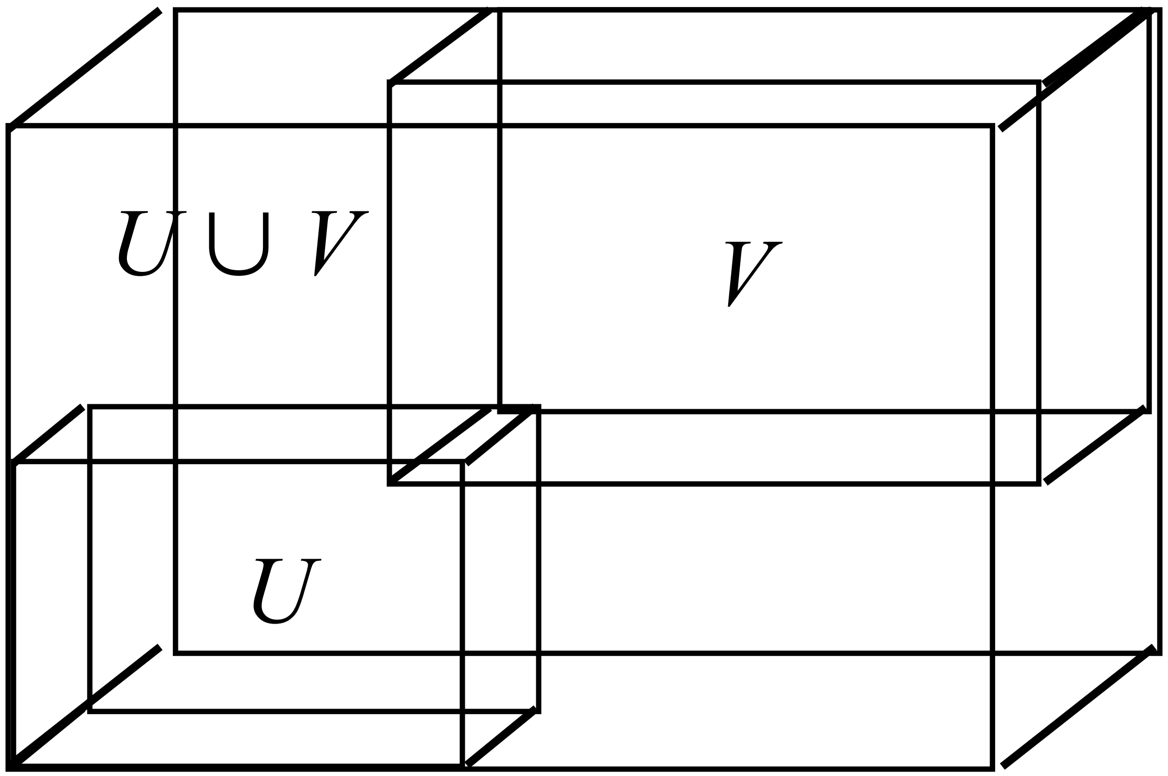
The Enumeration Tree

- Every node contains two collections of index sets, U and V
  - Index sets define subtensors
- Every node represents all subtensors that contain U and are contained in $U \cup V$
  - The union is over the index sets
  - The root has empty U
  - The leaves have empty V
- It is possible that these tensors are *reduced*
  - Some modes are 0-dimensional



The Enumeration Tree

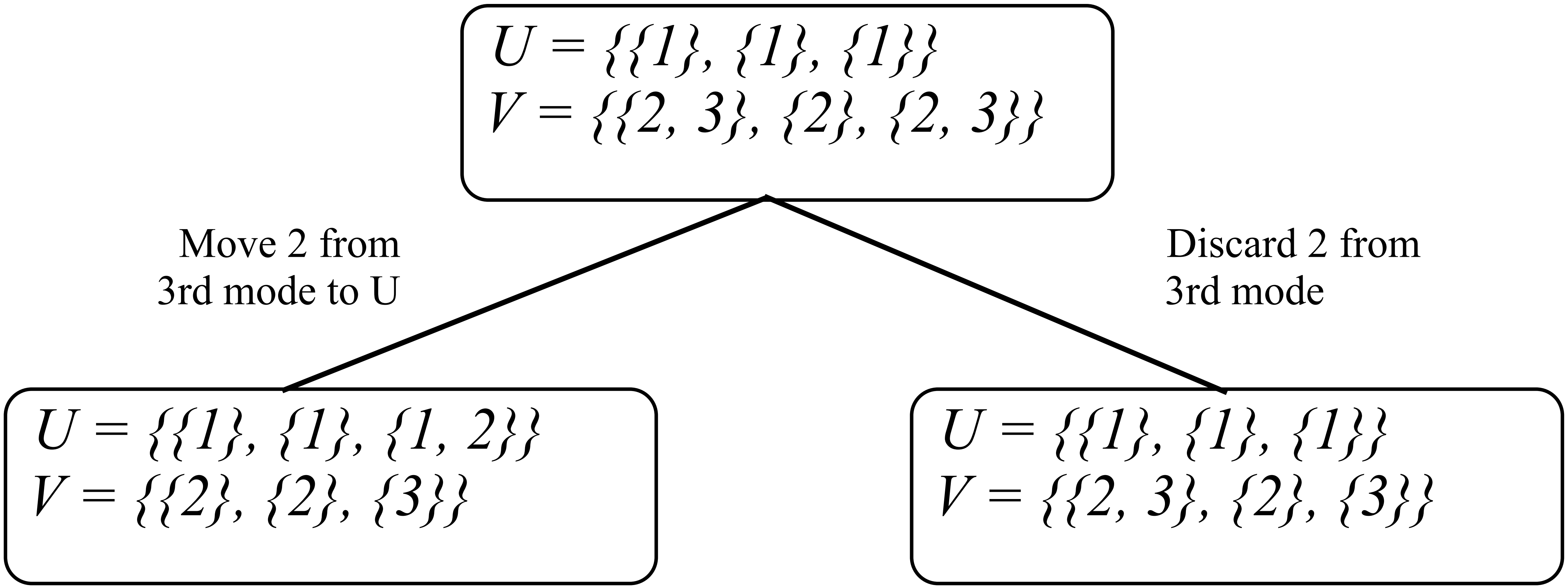
- Every node contains two collections of index sets, U and V
  - Index sets define subtensors
- Every node represents all subtensors that contain U and are contained in $U \cup V$
  - The union is over the index sets
  - The root has empty U
  - The leaves have empty V
- It is possible that these tensors are *reduced*
  - Some modes are 0-dimensional



Building the Tree

- At every node (U, V), select a dimension in a mode in V and remove it from V
- Create two childs
  - *Left*: Add that dimension in the correct mode in U
  - *Right*: Don't add
- For the left child, we can remove all those elements of V that cannot be added to the sub-tensor to and keep it all-1s

An Example



– If U was $\{\{1\},\{1\},\{1\}\}$ and V was $\{\{2, 3\}, \{2\}, \{2, 3\}\}$ and we moved 2 from the 3rd mode to U and (3, 1, 2) is a 0-element, the new V will be $\{\{2\}, \{2\}, \{3\}\}$

Checking for the Closedness

- We can check for the closedness during the enumeration
- If there exists a 1 in the tensor that is not in $U \cup V$ but which could be added to $U \cup V$ without breaking the all-1s property, then no child of this node will be closed
  - The node can be pruned, the closure will appear in other part of the tree
  - We don't need to try all 1s not in $U \cup V$, just those corresponding to the dimension removed in the ancestors of this node that themselves were right childs

An Example

$U = \{\{1\}, \{1\}, \{1\}\}$
$V = \{\{2, 3\}, \{2\}, \{2, 3\}\}$

$U = \{\{1\}, \{1\}, \{1, 2\}\}$
$V = \{\{2\}, \{2\}, \{3\}\}$

drop *2* from 3rd mode

$U = \{\{1\}, \{1\}, \{1\}\}$
$V = \{\{2, 3\}, \{2\}, \{3\}\}$

drop *3* from 1st mode

$U = \{\{1\}, \{1\}, \{1\}\}$
$V = \{\{2\}, \{2\}, \{3\}\}$

If {{1,2}, {1,2}, {1,2,3}} is full-1s subtensor, this node cannot yield to closed itemsets

Handling other constraints

- If other constraints have been issued, we can stop traversing the branch if *none of the* subtensors represented by the node satisfies the constraints
  - We can get the maximum sizes of modes from $U \cup V$
  - And the minimum sizes from U
- For example, for minimum size constraints, we stop if the size fo $U \cup V$ drops below the constraint
  - Similar for minimum volume
  - For δ-isolation, we can consider the fraction of 1s that are outside U w.r.t. the number of 1s that are inside $U \cup V$

Final Notes on Data-Peeler

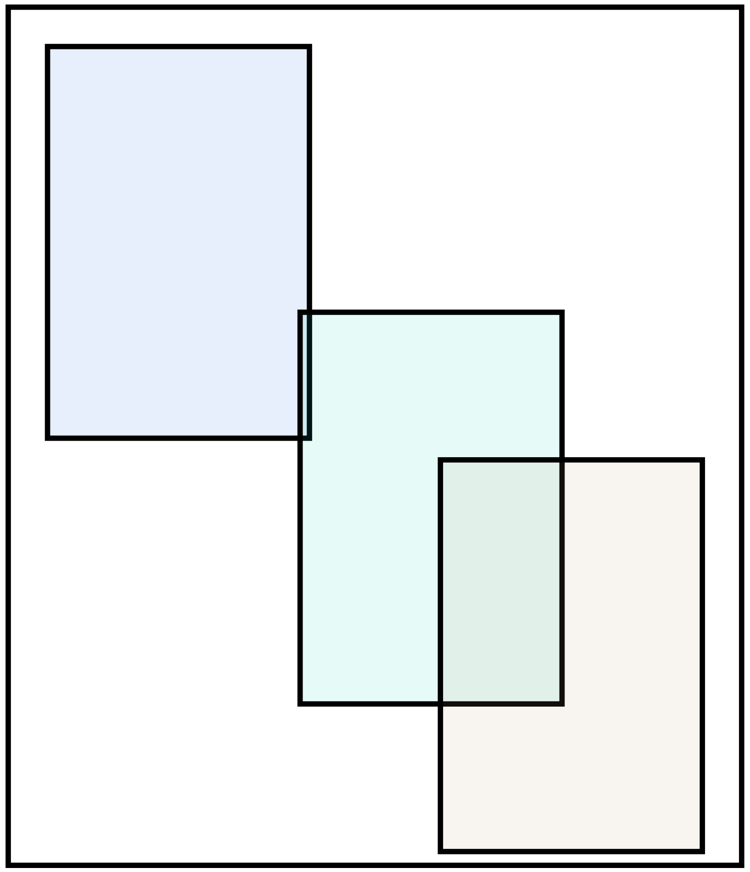
- The (greedy) strategy selecting the element to remove from V is crucial for fast execution
- Space complexity is $\prod_i I_i$ for I_1-by-I_2-by…-by-I_n tensor
  - A dense representation, won't work with huge-but-sparse tensors
  - The biggest data set used in the paper is 323-by-323-by-39-by-6
    - 24.4M elements
    - 602K closed itemsets

Tiling Tensors

- Tiling tensors is analogous to tiling matrices
- Similarly, we can use the closed *n*-way itemsets as building blocks for the tiling
  - Reduces to the set cover problem—again
- A tiling gives us a Boolean CP decomposition of the tensor

Matrix Tiling as Decomposition

- Each tile is a rank-1 submatrix
  - Outer product of two binary vectors
- If we sum two tiles, we get a non-binary matrix
  - Instead of sum, we can take the element-wise maximum
  - This is known as the **Boolean matrix product**

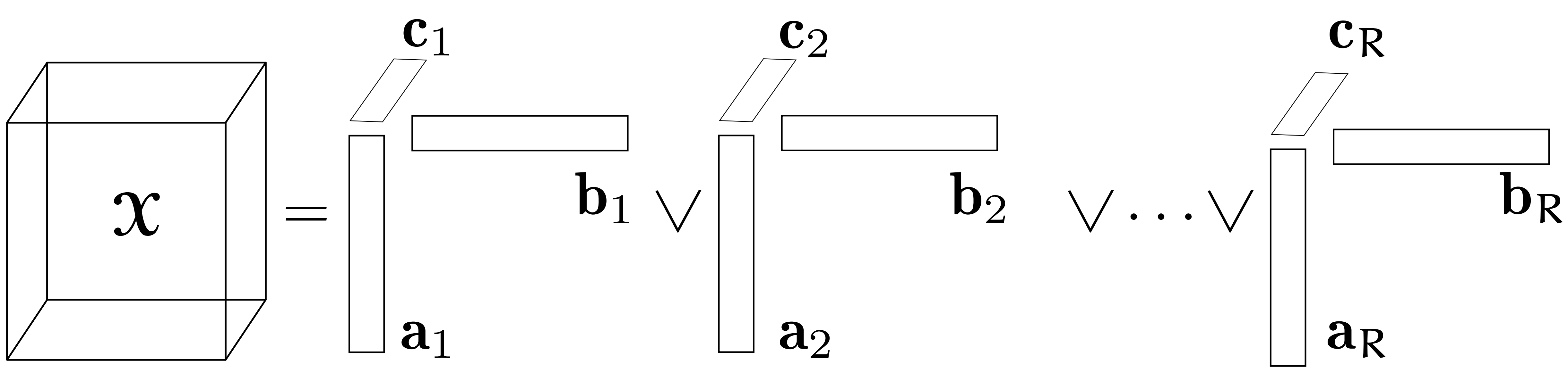


$$(\mathbf{A} \boxplus \mathbf{B})_{ij} = \bigvee_{i=1}^{k} a_{ik} b_{kj}$$

- Minimum tiling is finding the Boolean decomposition with minimum inner dimension

Tensor Tiling as CP Decomposition

- Analogously for tensors
  - A tile is a rank-1 tensor
  - Tiling is a Boolean sum of rank-1 tensors
  - Minimum tiling is about finding the smallest number of rank-1 tensors to exactly express the original tensor
    - *Boolean tensor rank!*



Boolean Tensor Decompositions

- We can transform both CP and Tucker decomposition into Boolean versions
  - Original tensors are required to be binary
  - All factors (and core tensor) are required to be binary
  - The summation is replaced by logical OR
  - The error measure is the Hamming distance between the original tensor and its decomposed representation
    - Equals to sums-of-squares of element-wise differences
    - Note: in (combinatorial) tiling, we don't allow "holes" in the tiles —this is more general

Miettinen 2011

A Bit About Boolean Matrix Factorizations

- Boolean matrix factorization (BMF) differs from normal factorizations in significant parts
  - Rank-1 Boolean matrices are rank-1 normal matrices
  - The **Boolean rank** of a matrix is the smallest number of rank-1 Boolean matrices needed to sum up to exactly create the matrix
    - Computing (or even a good approximation of) this rank is NP-hard
    - This is equivalent to the minimum tiling problem
  - Given k, finding the minimum-error rank-k BMF is also NP-hard
    - But note that this is *not* the same thing as maximum k-tiling

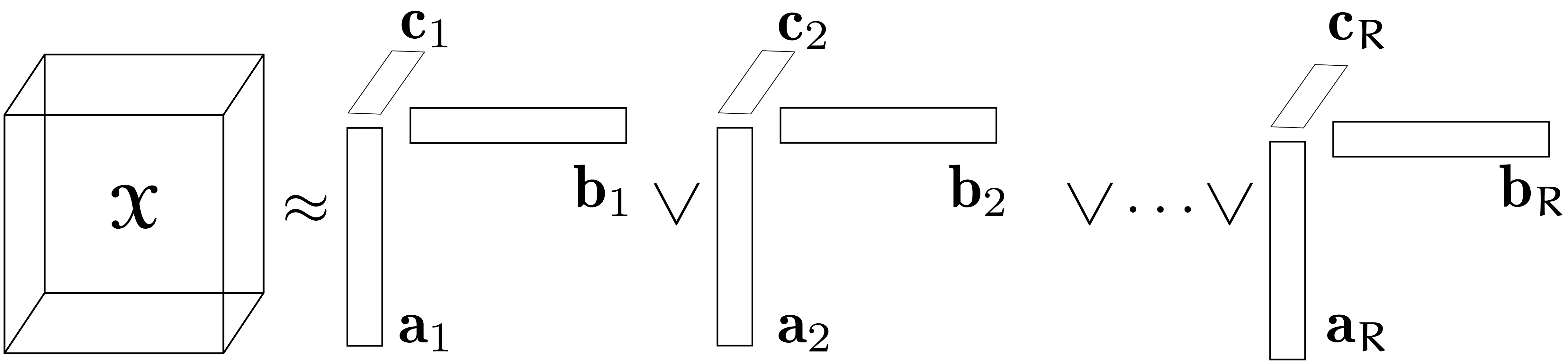
The Basis Usage Problem

- The *Basis Usage* (BU) problem is the following
  - Given a binary matrix **A** and a binary matrix **B**, find a binary matrix **C** s.t. $|\mathbf{A} - \mathbf{B} \boxplus \mathbf{C}|$ is minimized
    - Here $|\mathbf{A}|$ is the number of non-zeros in **A**
  - Equivalently: given a binary column vector **a** and a binary matrix **B**, find a binary column vector **c** s.t. $|\mathbf{a} - \mathbf{B} \boxplus \mathbf{c}|$ is minimized
    - With **B** fixed, every column of **A** can be solved separately
- The Basis Usage problem is equivalent to the *Positive-Negative Partial Set Cover* (±PSC) problem:
  - Given a set system $(P \cup N, S)$, $P \cap N = \emptyset$, find a subcollection $C \subseteq S$ such that $|N \cap (\cup C)| + |P \setminus (\cup C)|$ is minimized
    - Minimize the number of included negative elements plus not included positive elements

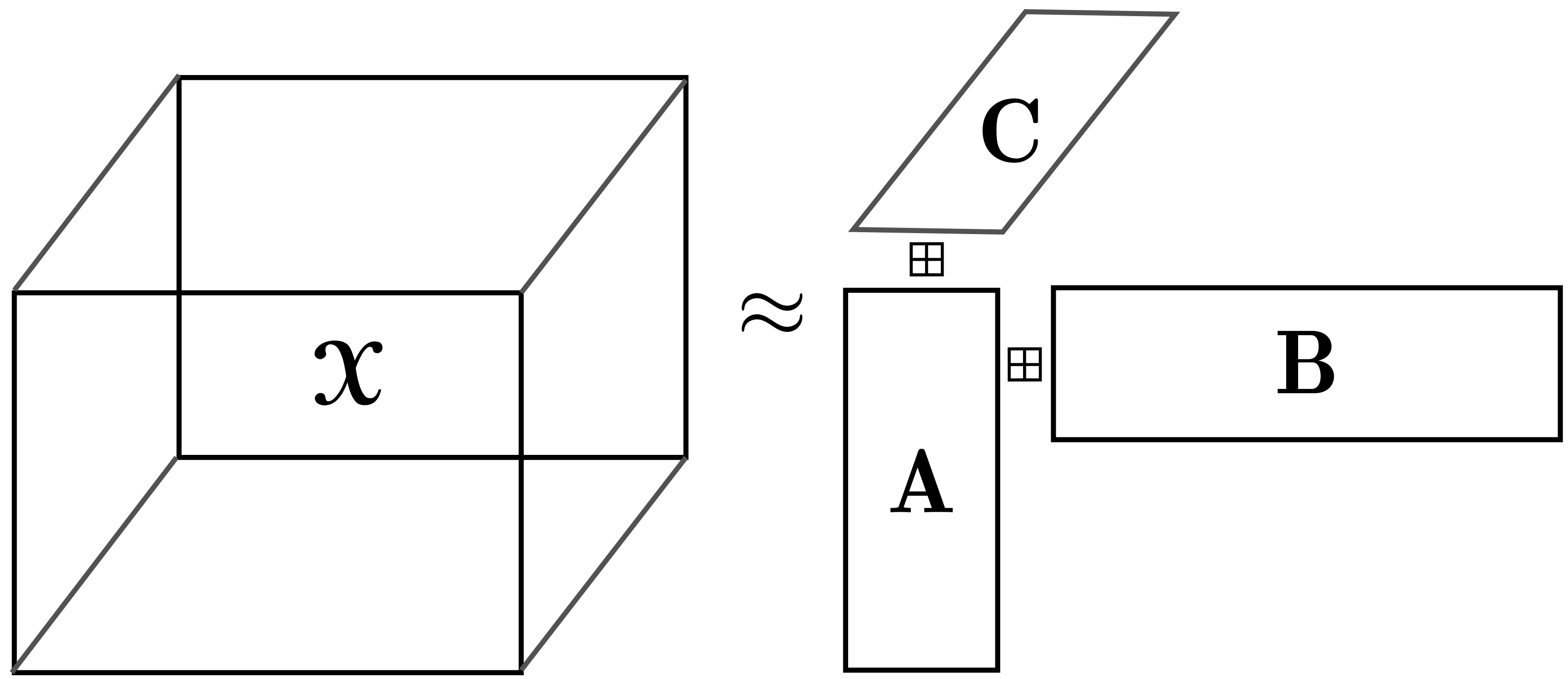
The Hardness of the BU Problem

- The BU problem is NP-hard (unsurprisingly)
- The BU problem is also NP-hard to approximate well
  - It is NP-hard to approximate the BU problem to within a factor of
    $$\Omega\left(2^{\log^{1-\varepsilon}|P|}\right)$$
    for any ε > 0
  - It is *quasi-NP-hard* to approximate the BU problem to within a factor of
    $$\Omega\left(2^{(4\log k)^{1-\varepsilon}}\right)$$
    - Quasi-NP-hardness: NP-hard unless NP $\subseteq$ DTIME($n^{\mathrm{polylog}(n)}$)
- All the results hold for ±PSC as well

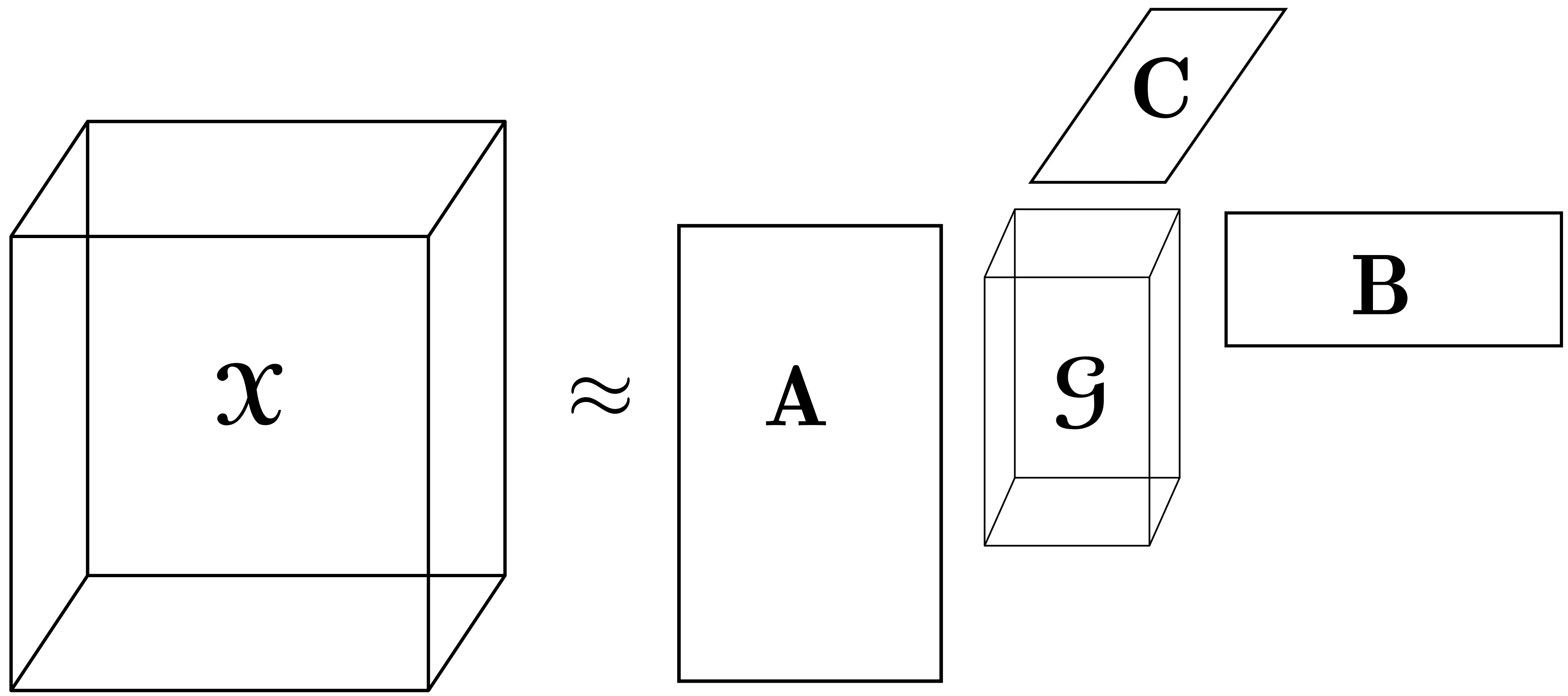
Boolean CP Decomposition



Boolean CP Decomposition



Boolean Tucker Decomposition



$$x_{ijk} \approx \bigvee_{p=1}^{P} \bigvee_{q=1}^{Q} \bigvee_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr}$$

Boolean Tensor Rank

- Boolean tensor rank is the minimum number of rank-1 Boolean tensors needed to be summed to get the original tensor
- Boolean tensor rank is NP-hard to compute
  - So is normal tensor rank
- Boolean tensor rank can be more than the smallest dimension
  - So can normal tensor rank
- But no more than $\min\{IJ, IK, JK\}$
  - Neither can normal tensor rank
- There is no Boolean border rank

Sparsity

- Binary matrix $\mathbf{X}$ of Boolean rank R and $|\mathbf{X}|$ 1s has Boolean rank-R decomposition $\mathbf{A} \boxplus \mathbf{B}$ such that $|\mathbf{A}| + |\mathbf{B}| \leq 2|\mathbf{X}|$
- Binary N-way tensor $\mathcal{X}$ of Boolean tensor rank R has Boolean rank-R CP-decomposition with factor matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$ such that $\sum_i |\mathbf{A}_i| \leq N|\mathcal{X}|$
- Both results are existential only and extend to approximate decompositions

An Algorithm for Boolean CP

- The normal CP can be solved using the ALS approach

$$\mathbf{X}_{(1)} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T$$

$$\mathbf{X}_{(2)} = \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T$$

$$\mathbf{X}_{(3)} = \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T$$

An Algorithm for Boolean CP

- The normal CP can be solved using the ALS approach

$$\mathbf{X}_{(1)} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T$$
$$\mathbf{X}_{(2)} = \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T$$
$$\mathbf{X}_{(3)} = \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T$$

- Similar equations hold for the Boolean CP
  - Khatri–Rao product is the same in Boolean arithmetic

$$\mathbf{X}_{(1)} = \mathbf{A} \boxplus (\mathbf{C} \odot \mathbf{B})^T$$
$$\mathbf{X}_{(2)} = \mathbf{B} \boxplus (\mathbf{C} \odot \mathbf{A})^T$$
$$\mathbf{X}_{(3)} = \mathbf{C} \boxplus (\mathbf{B} \odot \mathbf{A})^T$$

An Algorithm for Boolean CP

- The normal CP can be solved using the ALS approach

$$\mathbf{X}_{(1)} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T$$
$$\mathbf{X}_{(2)} = \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T$$
$$\mathbf{X}_{(3)} = \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T$$

- Similar equations hold for the Boolean CP
  - Khatri–Rao product is the same in Boolean arithmetic

$$\mathbf{X}_{(1)} = \mathbf{A} \boxplus (\mathbf{C} \odot \mathbf{B})^T$$
$$\mathbf{X}_{(2)} = \mathbf{B} \boxplus (\mathbf{C} \odot \mathbf{A})^T$$
$$\mathbf{X}_{(3)} = \mathbf{C} \boxplus (\mathbf{B} \odot \mathbf{A})^T$$

- But with Boolean, we don't have pseudo-inverses
  - The BU problem!

A Greedy Algorithm for the BU

- Consider the column case of BU
  - Find **x** to minimize $|\mathbf{a} - \mathbf{B} \boxplus \mathbf{x}|$
- Every element of **x** selects whether the corresponding column of **B** is added to the presentation of **a**
  - If an already-selected column of **B** has 1 in row *i*, we say that row *i* is *covered*
- The algorithm:
  - Try each column of **B** one-by-one and if the column covers more not-yet-covered 1s than it covers not-yet-covered 0s, set the corresponding element of **x** to 1

Back to the CP

- We can use the greedy BU algorithm instead of the pseudo-inverse with the equations
- But starting from random starting points won't give us very good factorizations
  - There are many local minima
- Instead, we can solve the ordinary BMF for the different matricizations to obtain the initial **A**, **B**, and **C**

The Tucker Case

- For the matrices, we can use same approach as with the CP

$$\mathbf{X}_{(1)} = \mathbf{A} \boxplus \mathbf{G}_{(1)} \boxplus (\mathbf{C} \otimes \mathbf{B})^T$$
$$\mathbf{X}_{(2)} = \mathbf{B} \boxplus \mathbf{G}_{(2)} \boxplus (\mathbf{C} \otimes \mathbf{A})^T$$
$$\mathbf{X}_{(3)} = \mathbf{C} \boxplus \mathbf{G}_{(3)} \boxplus (\mathbf{B} \otimes \mathbf{A})^T$$

- For the core, that's not the case
  - A small change can change everything
  - But the core is small, so we can afford more time with it

$$x_{ijk} \approx \bigvee_{p=1}^{P} \bigvee_{q=1}^{Q} \bigvee_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr}$$

- The algorithm
  - If $a_{ip}b_{jq}c_{kr} = 0$, the core's value doesn't matter
  - If there's $g_{pqr}a_{ip}b_{jq}c_{kr} = 1$, nothing else matters
  - For the rest, compute whether flipping g_{pqr} would help

Conclusions

- The tensor closed itemsets are natural generalizations of the normal ones
  - Mining is harder / pruning is not so efficient
- The Boolean tensor decompositions are natural analogues of the real-valued ones
  - Behave mostly similarly
  - Some computations are harder
  - Boolean tensor factorizations generalize tiling by allowing “holes” in the tiles

Essays for Topic IV

- *N*-way itemset mining v.s. normal itemset mining
  - What's so hard with tensors? Why not use *N*-way Apriori (how would it work)? Do also maximal and non-derivable itemset's definitions generalize to *N* modes?
- Noise-tolerant *N*-way itemsets
  - Cerf et al. 2013 present an algorithm for mining noise-tolerant (closed) *N*-way itemsets. Explain the (main) ideas. Can this be used to compute Boolean CP decomposition? How? Will the BU problem be a problem?
- Applications of tensor decompositions in data mining
  - Present some work that applies tensor decompositions in data mining. Explain the ideas. Are tensors necessary here? Is the work good?