Scheduling Splittable Jobs on Configurable Machines

Matthew Casey

Northeastern University → Northwestern University

Rajmohan Rajaraman

Northeastern University

David Stalfa

Northeastern University

Cheng Tan

Northeastern University

Abstract Definition

Configurable Machine Scheduling (CMS)

We schedule *splittable* jobs on *configurable* machines

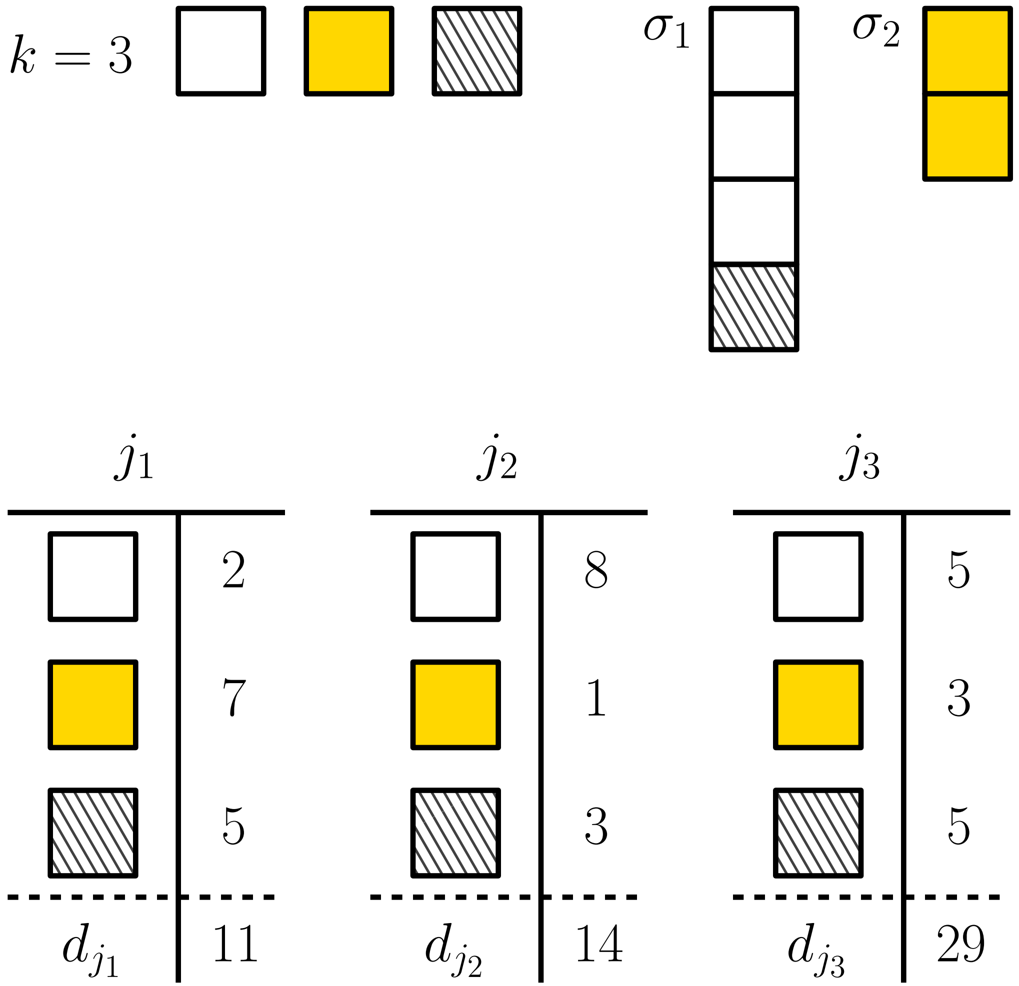
Configurable – Each machine is assigned a configuration, which specifies how the machine is partitioned

Splittable – Jobs have a demand, which is satisfied by assigning the job to multiple partitions

Formal Definition

Input:

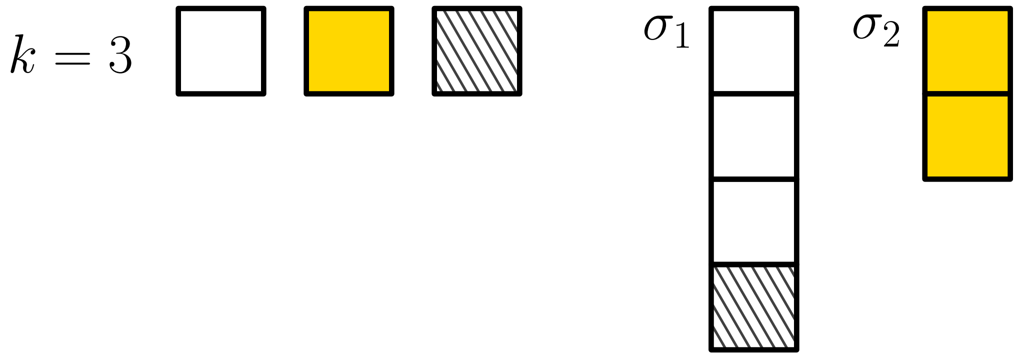
1. Set B of k block types
2. Set C of configurations, each a multiset of blocks
3. Set J of jobs each with a demand and demand table

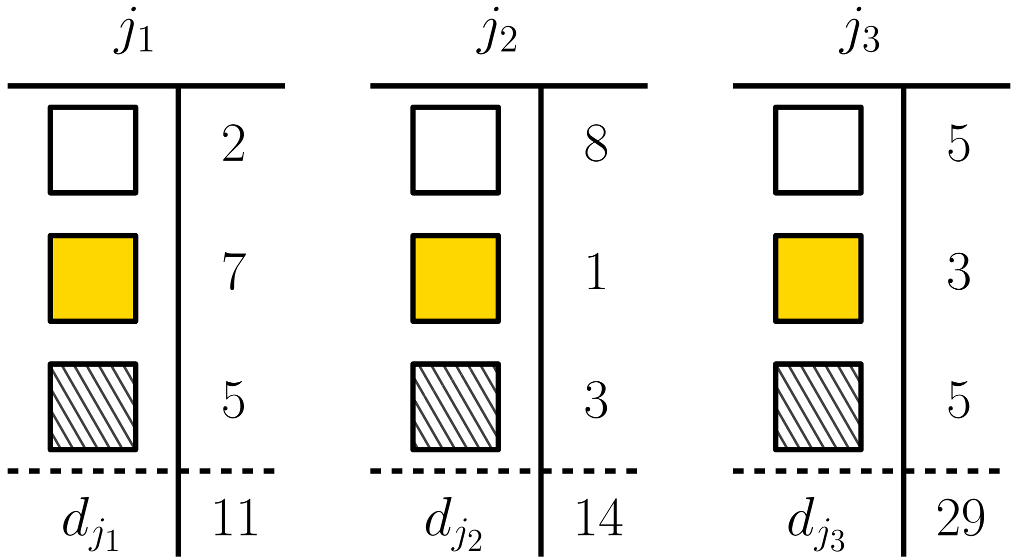


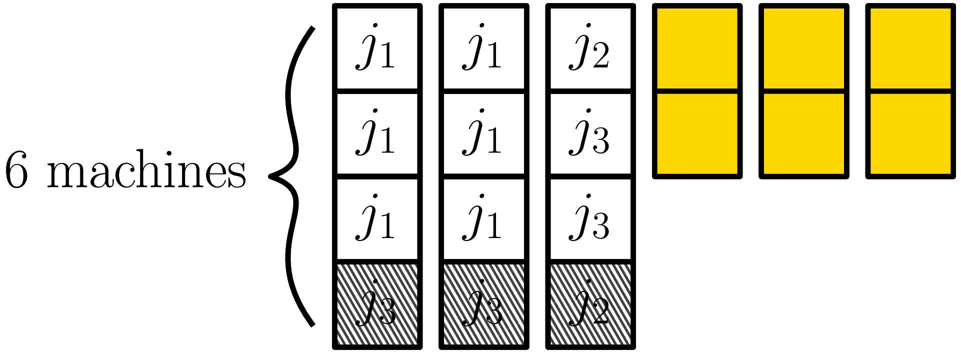
Formal Definition

Goal: Create a schedule where

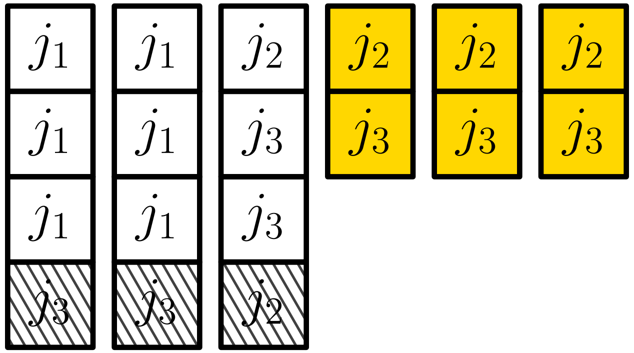
1. Each machine is assigned a configuration
2. Each block is assigned a job
3. The number of machines used is minimized

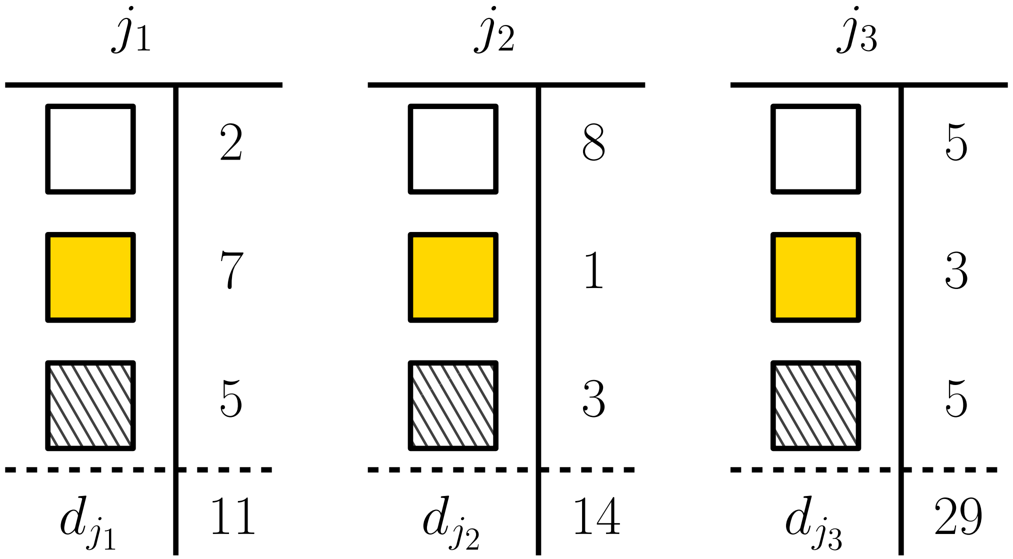


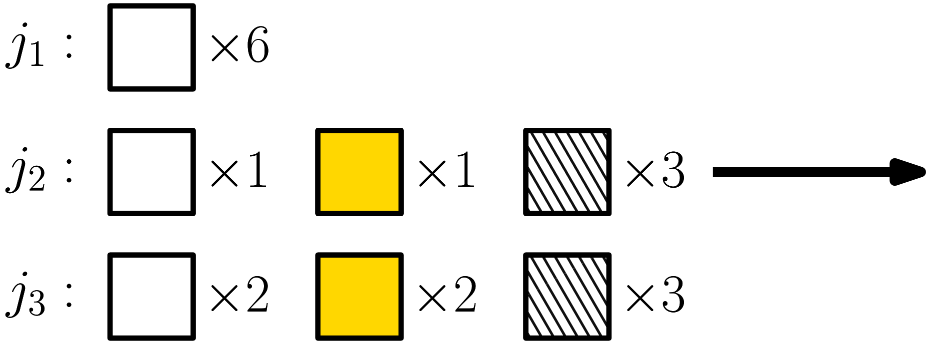




Formal Definition







$j_1 : 2 \times 6 = 12$

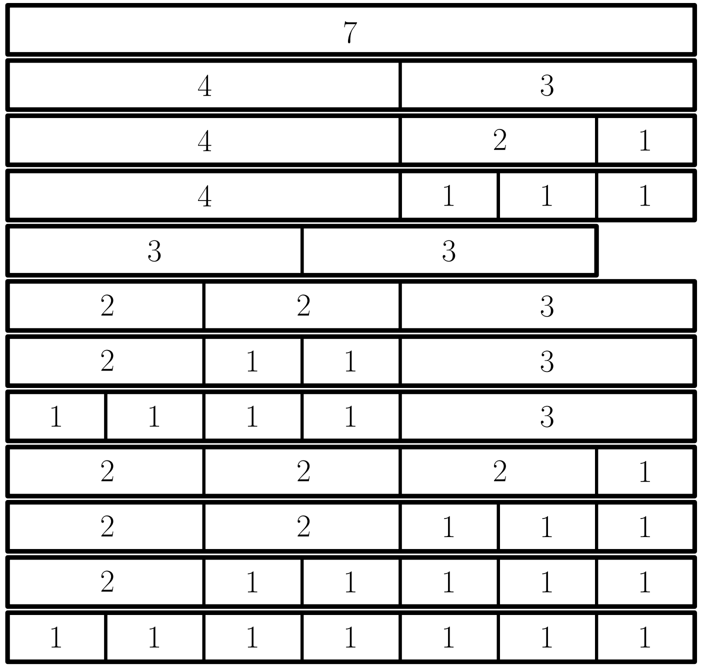
$j_2 : 8 \times 1 + 1 \times 1 + 3 \times 3 = 18$

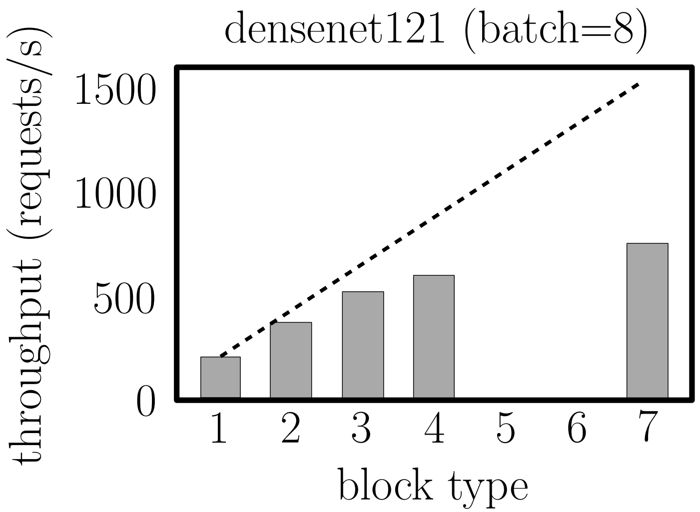
$j_3 : 5 \times 2 + 3 \times 3 + 5 \times 2 = 35$

Motivation

Datacenter scheduling AI inference tasks on modern GPUs:

- Datacenter uses a GPU with Multi-Instance GPU (MIG) like NVIDIA A100 [Configurations]
  - Allows partitioning into smaller hardware isolated GPU blocks
- Models sent to datacenter need to satisfy *d* inference requests per hour [Jobs]
- Profiling details number of requests satisfied per hour on each block [Demand Tables]





[Tan et al. 2021]

Related Work

- CMS generalizes multiset multicover
  - Has a logarithmic hardness that implies logarithmic hardness for CMS
  - We use the greedy algorithm for it [Rajagopalan and Vazirani 1999]
- CMS with implicit configurations generalizes bin packing
  - We use algorithm to find conic integer combinations in fixed dimension from bin packing with constant item types [Goemans and Rothvoss 2020]
- CMS with a single configuration is a fair allocation problem
  - Blocks represents items and jobs represent players
  - Minimize the maximum number of copies of any block

Our Contributions

1. Formally defining **Configurable Machine Scheduling**
2. Algorithms and hardness results for CMS and its variants

Problem	Algorithm	Approximation	Hardness
General	LP+ Greedy	$O(\log cnk)$	$\Omega(\log nk)$
$O(1)$ configurations	Extreme-Point LP Rounding	$(2+\varepsilon)\mathrm{OPT} + \|C\|$ $3+\varepsilon$	2
$O(1)$ configurations of $O(1)$ size	Small/Large Job LP	$1+\varepsilon$	?
$O(1)$ number of jobs and blocks, with all configurations up to a given size	Conic Integer Combinations in Fixed Dimension	1	-

Constant # of Configurations

Canonical LP:

- $x_{i,j}$: number of blocks of type *i* assigned to job *j*
- y_σ: number of machines assigned configuration σ

$$
\begin{array}{lll}
\sum_j x_{i,j} \leq \sum_{\sigma \in C} y_\sigma \cdot \sigma_i & i \in B & \text{there are sufficient blocks} \\
\sum_i f_j(i) \cdot x_{i,j} \geq d_j & j \in J & \text{demands are satisfied} \\
x_{i,j} \geq 0 & i \in B \text{ and } j \in J & \text{variables are nonnegative} \\
y_\sigma \geq 0 & \sigma \in C & \text{variables are nonnegative}
\end{array}
$$

Constant # of Configurations

Step 1: Enumeration

- Constant number of configurations allows for enumeration over the possible values for *y* variables

- Max number of configurations needed is $\sum_j d_j$

- For each configuration search over

$$L = \{ \lfloor (1+\varepsilon)^i \rfloor \mid 0 \leq i \leq \log_{1+\varepsilon}(\sum_j d_j) \}$$

- These search values will be within a factor $1 + \varepsilon$ of the optimum

Constant # of Configurations

Step 2: Feasibility LP

$$\sum_{j} x_{i,j} \le \sum_{\sigma \in C^*} y_\sigma \sigma_i \qquad i \in B \qquad \text{there are sufficient blocks}$$

$$\sum_{j} x_{i,j} \le \sum_{\sigma \in C^*} m_\sigma \sigma_i \qquad i \in B \qquad \text{there are sufficient blocks}$$

$$\sum_{i} f_j(i) \cdot x_{i,j} \ge d_j \qquad j \in J \qquad \text{demands are satisfied}$$

$$x_{i,j} \ge 0 \qquad i \in B \text{ and } j \in J \qquad \text{variables are nonnegative}$$

$$y_\sigma \ge 0 \qquad \sigma \in C \qquad \text{variables are nonnegative}$$

- Replace configuration variables *y* with fixed values $m \in L^{|C^*|}$
- Find an extreme point solution
- Replace configuration variables *y* with fixed values $m \in L^{|C^*|}$

Constant # of Configurations

Step 3: Construct Graph

- Construct an auxiliary graph where:
  - Nodes are jobs and blocks
  - An edge exists between a job *j* and block *i* if $x_{i,j} > 0$

- The components in this graph are either trees or have one cycle [Lenstra, Shmoys, Tardos 1987]

Constant # of Configurations

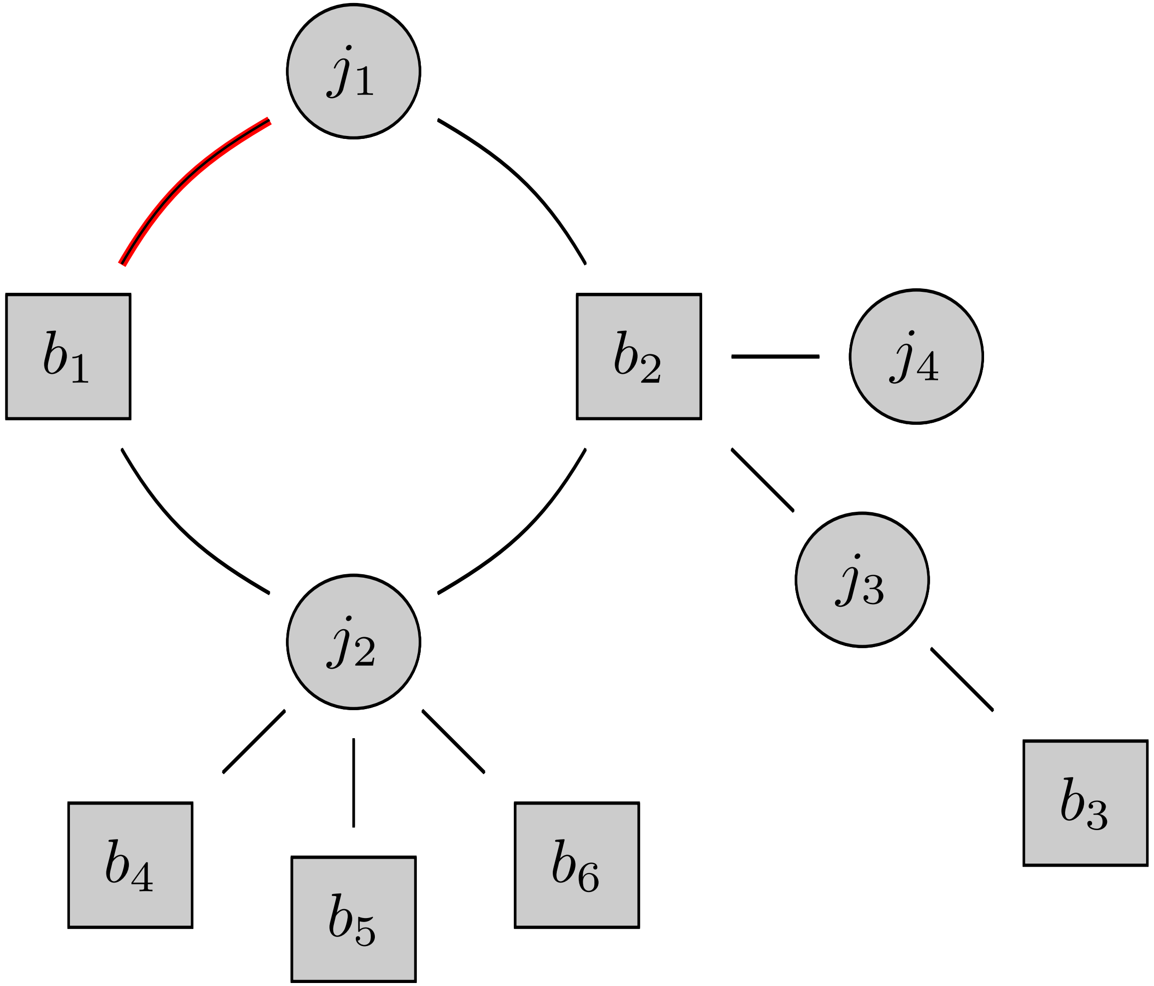
Step 3: Construct Graph

1. Pick any job in the cycle

2. Remove the adjacent edge that satisfies less demand (set *x*-variable to 0)

$$x_{b_1,j_1} \cdot f_{j_1}(b_1) \text{ vs } x_{b_2,j_1} \cdot f_{j_1}(b_2)$$

3. Make that job the root of the resulting tree



Constant # of Configurations

Step 4: Rounding

1. For each block *c* child to job *j*

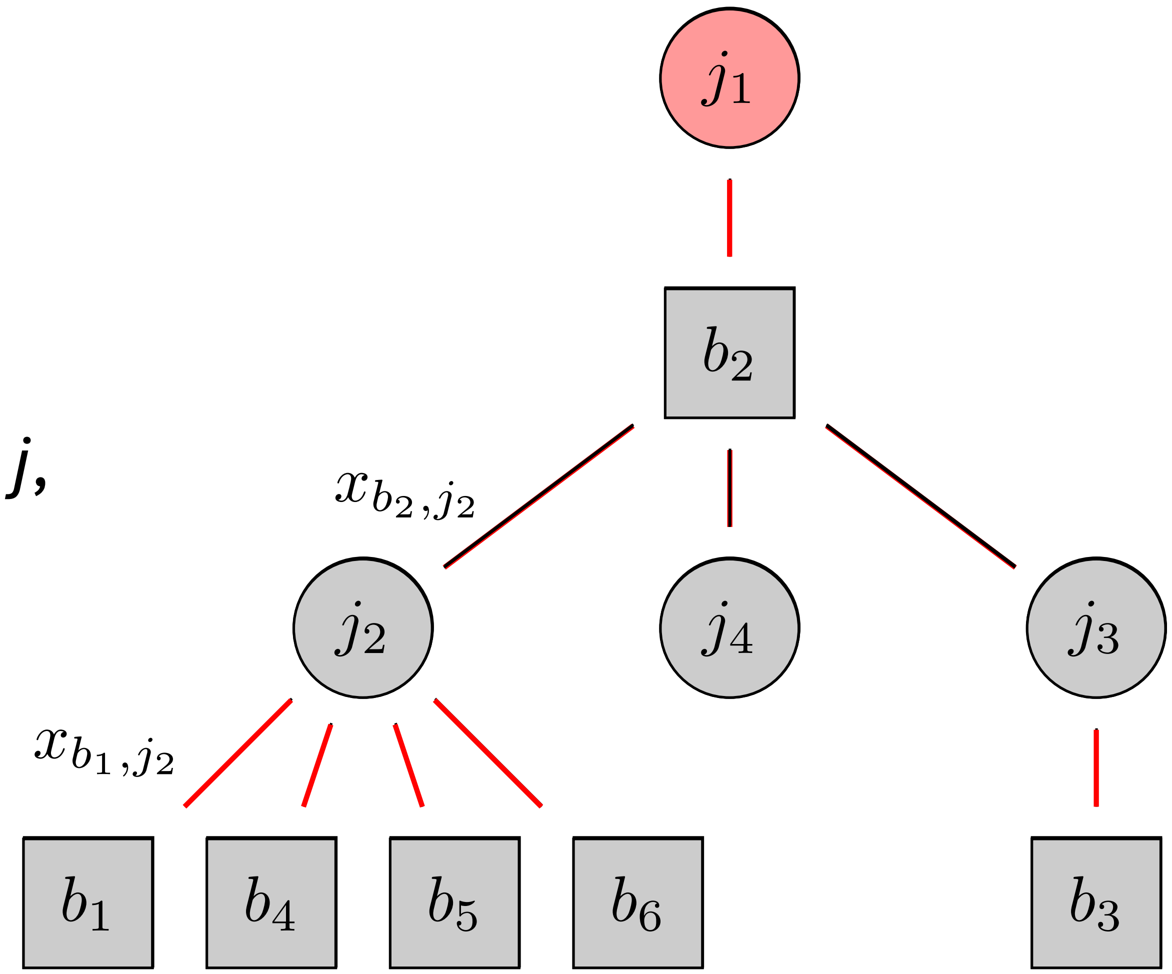
$$x^*_{c,j} = \lceil 2x_{c,j} \rceil$$

2. For each block *p* parent to job *j*,

$$x^*_{p,j} = \lfloor 2x_{p,j} \rfloor$$

3. For each configuration σ

$$m^*_\sigma = 2m_\sigma + 1$$



Constant # of Configurations

Theorem. The previous algorithm returns a solution in polynomial time with cost at most $(2+\varepsilon)OPT + |C|$ to the CMS problem if the number of configurations is constant.

Feasibility:

- Removed edge in cycle is covered by doubling the remaining edge
- Rounded values are all* larger than non-rounded values
- Only use two times + 1 of each block after rounding

Approximation Factor:

$$\sum_\sigma m^*_\sigma = \sum_\sigma (2m_\sigma + 1) = 2\sum_\sigma m_\sigma + |C| \leq 2(1+\varepsilon)\text{OPT} + |C|$$

Open Problems

1. Is there an Asymptotic PTAS or additive constant approximation when there are a *Constant Number of Configurations*?

2. Numerical CMS: each block has a size and only configurations whose blocks' sizes add up to some fixed capacity are allowed
   1. Is there a sublogarithmic approximation for Numerical CMS?

3. Leverage the structure of the config set to get better algorithms
   1. NVIDIA A100's block set is structured as a tree and every valid configuration is a set of blocks with no ancestor-descendant relations

4. Consider other objectives such as completion time or flow time, in both offline and online settings