The Weighted Average Constraint

Alessio Bonfietti <alessio.bonfietti@unibo.it>

Michele Lombardi <michele.lombardi2@unibo.it>

DEIS, University of Bologna

Main Topic

Paper topic: a global constraint for weighted average expressions

$$\text{average}([\mathtt{W_i}], [\mathtt{V_i}], \mathtt{Y}) \leftrightarrow \mathtt{Y} = \frac{\sum_{i=0}^{n-1} \mathtt{W_i} \cdot \mathtt{V_i}}{\sum_{i=0}^{n-1} \mathtt{W_i}}$$

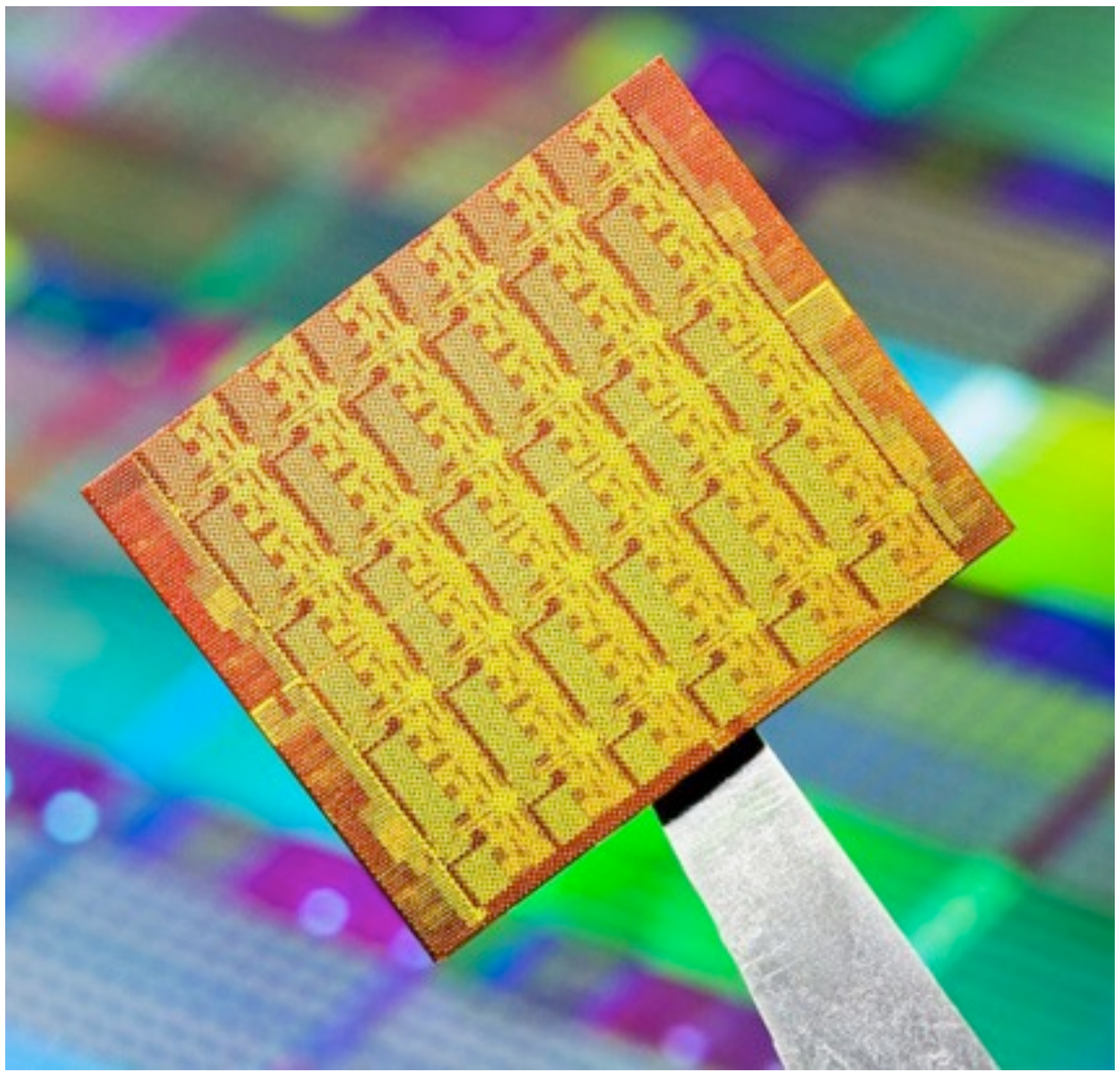
Equivalent for integer variables:

$$\text{average}([\mathtt{W_i}], [\mathtt{V_i}], \mathtt{Y}) \leftrightarrow \mathtt{Y} = round\left(\frac{\sum_{i=0}^{n-1} \mathtt{W_i} \cdot \mathtt{V_i}}{\sum_{i=0}^{n-1} \mathtt{W_i}}\right)$$

Motivation

Context: workload dispatching for high performance computing

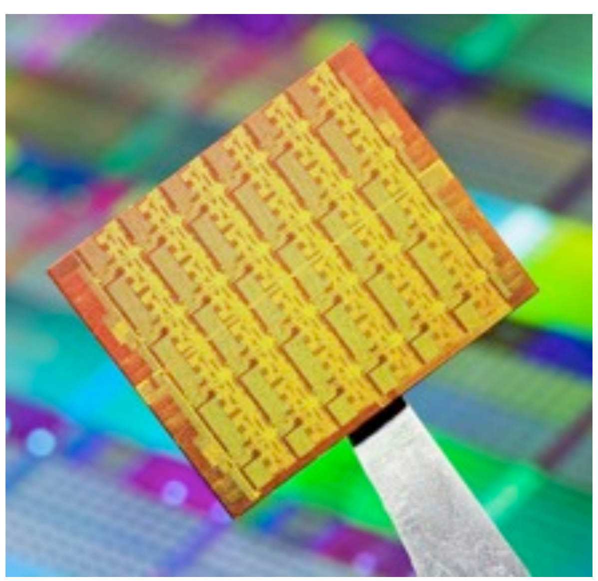
Server room



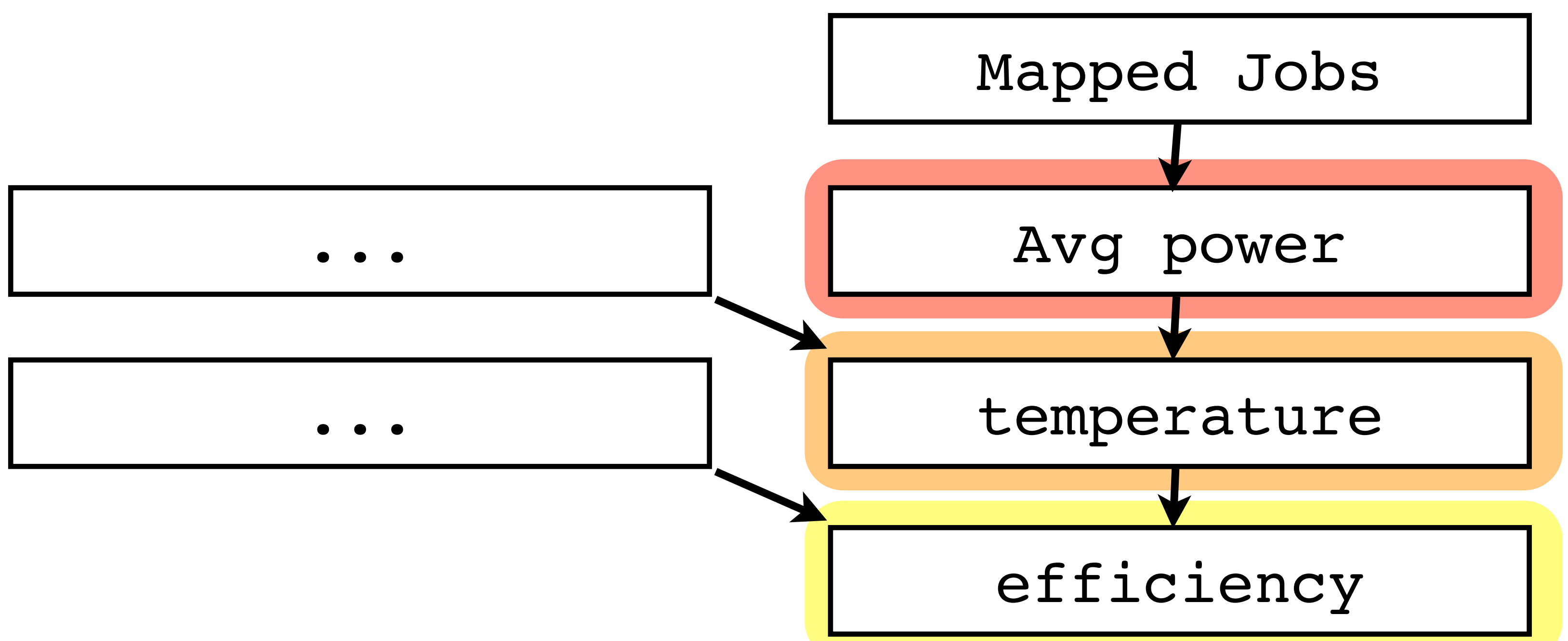
Multi-core Platforms

Motivation

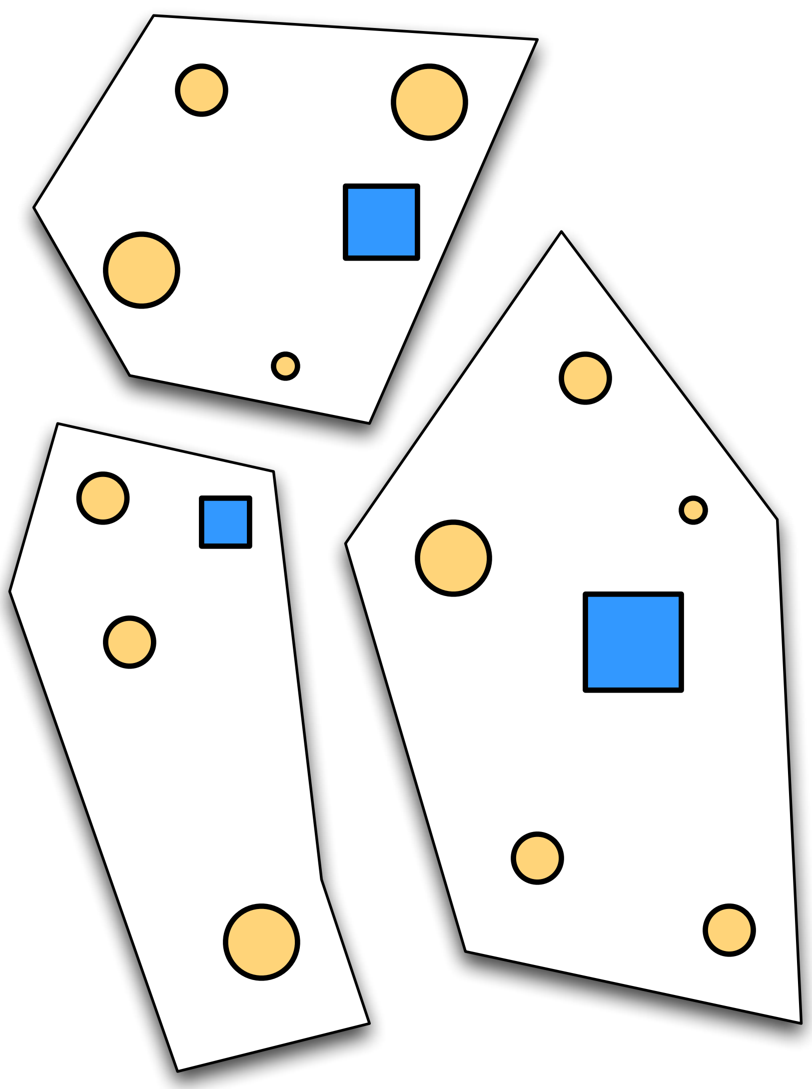
Context: workload dispatching for high performance computing

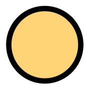


- Jobs arrive in batches
- Jobs are assigned to different machines/cores
- Local scheduling (by the OS)
- **Obj:** maximize worst core efficiency

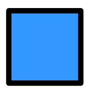


Context: assignment problems with balancing issues





`customer (size = demand)`



`facility (size = capacity)`

Single Source Capacitated Facility Problem

- Assign customers to facility
- Meet capacity constraints

...with Fair Travel Times

- Balance the average travel time per facility

Which Model?

Modeling Choices:

- Assignment variables:

$$\texttt{X}_\texttt{i} \in \{0..m-1\} \quad \forall i = 0 \ldots n-1$$

```
n = #customers/jobs
m = #facilities/cores
```

- For each facility/core *k*:

$$\texttt{POWER} = \frac{\sum_{i=0}^{n-1} (\texttt{X}_\texttt{i} = k) \cdot power_i}{\sum_{i=0}^{n-1} (\texttt{X}_\texttt{i} = k)}$$

$$\texttt{TTIME} = \frac{\sum_{i=0}^{n-1} (\texttt{X}_\texttt{i} = k) \cdot ttime_i}{\sum_{i=0}^{n-1} (\texttt{X}_\texttt{i} = k)}$$

By abstracting a little bit:

$$\texttt{Y} = \frac{\sum_{i=0}^{n-1} \texttt{W}_\texttt{i} \cdot v_i}{\sum_{i=0}^{n-1} \texttt{W}_\texttt{i}}$$

Which Model?

How do we model this expression? $\mathtt{Y} = \frac{\sum_{i=0}^{n-1} \mathtt{W_i} \cdot v_i}{\sum_{i=0}^{n-1} \mathtt{W_i}}$

- Fixed denominator

$$\mathtt{Y} = \frac{\sum_{i=0}^{n-1} \mathtt{W_i} \cdot v_i}{w}$$

sum constraint!

spread and deviation to improve filtering

- Just post it!

$$\sum_{i=0}^{n-1} \mathtt{W_i} \cdot v_i = \mathtt{Y} \cdot \sum_{i=0}^{n-1} \mathtt{W_i}$$

Likely weak propagation...

Which Model?

How do we model this expression? $\mathtt{Y} = \frac{\sum_{i=0}^{n-1} \mathtt{X_i} \cdot v_i}{\sum_{i=0}^{n-1} \mathtt{X_i}}$

- Otherwise, we need a new global constraint:

$$\text{average}([\mathtt{W_i}], [\mathtt{V_i}], \mathtt{Y})$$

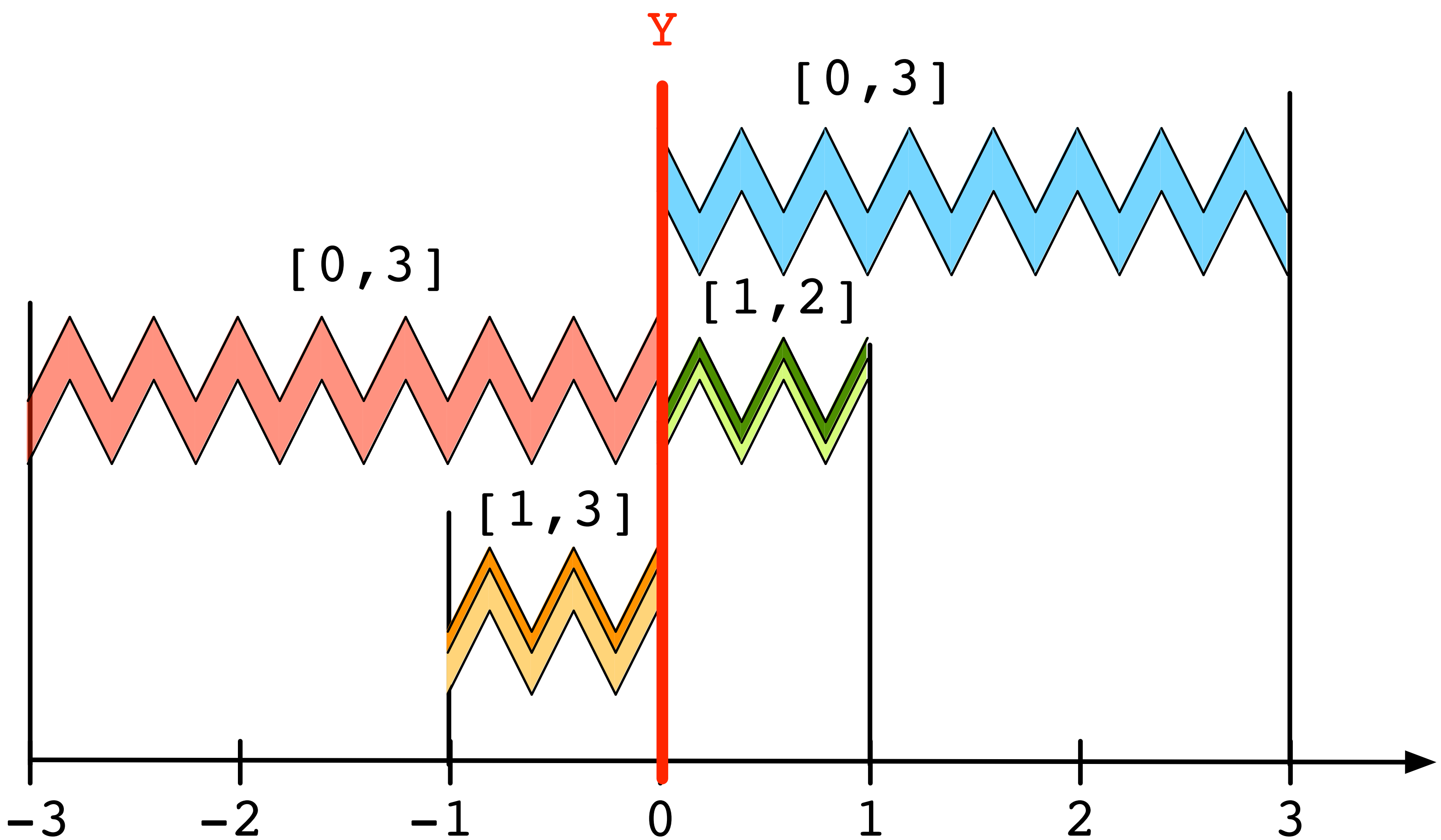
Here it is!

Spring equivalent: average as a bar pulled by metal spring

- Weights w_i = spring thickness, Values v_i = anchor points

Filtering

Spring equivalent: average as a bar pulled by metal spring



$$V = [-3, 1, 1, 3]$$

$$W_0 = [0, 3]$$

$$W_1 = [1, 3]$$

$$W_2 = [1, 2]$$

$$W_3 = [0, 3]$$

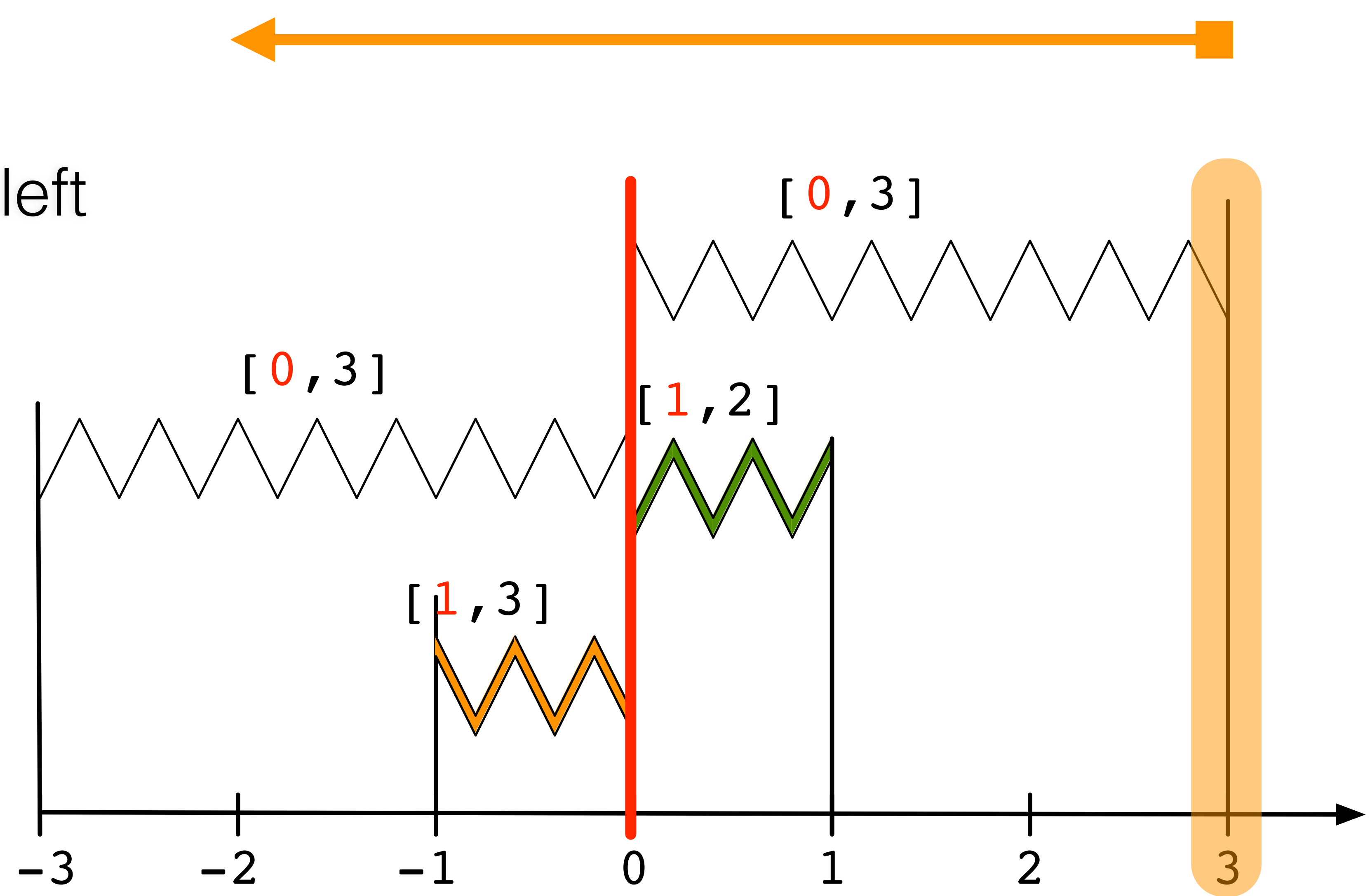
dark light

- Assumption 1: fixed values (adapted to variable v_i)
- Assumption 2: continuous domains (adapted to integer domains)

Pruning the Average Variable

Y upper bound = right-most position for the bar

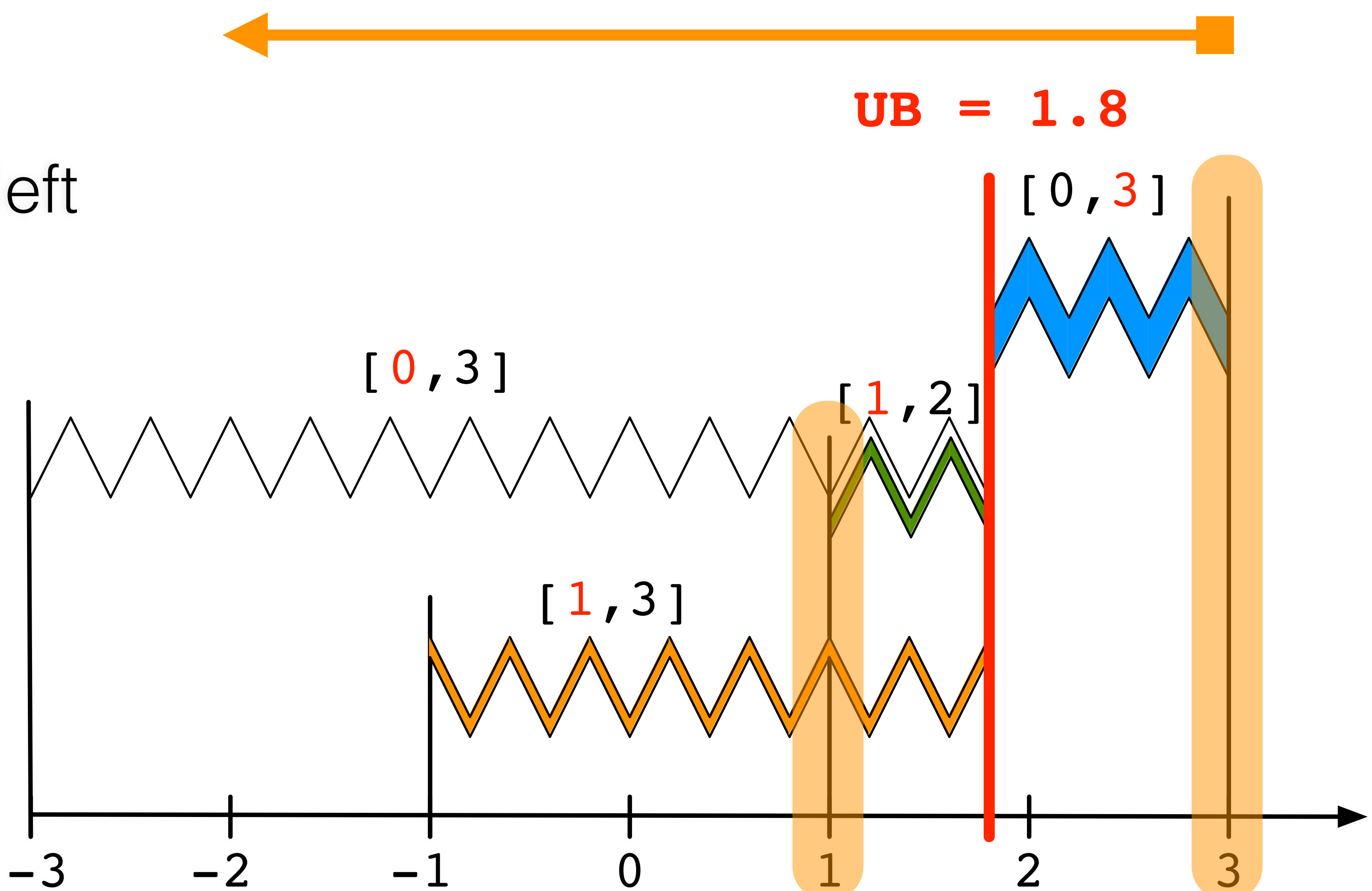
- Minimize all weights
- Scan w_i from right to left



Pruning the Average Variable

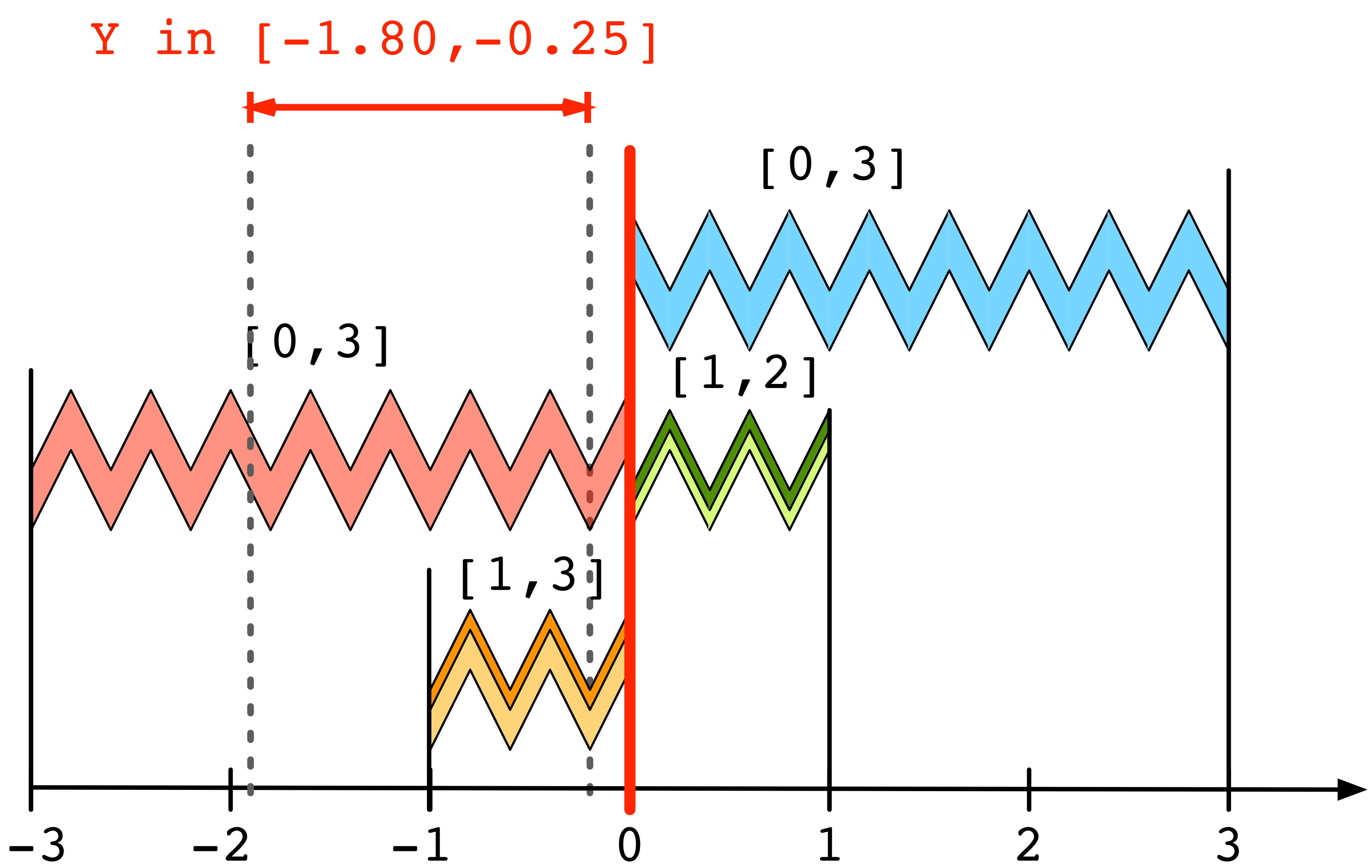
Y upper bound = right-most position for the bar

- Minimize all weights
- Scan w_i from right to left
- Maximize w_i if:
  $v_i > \text{current avg}$
- Repeat the process
- WC complexity: *O(n)* + *O(n log(n))* for the ordering



Pruning the Weight Variables

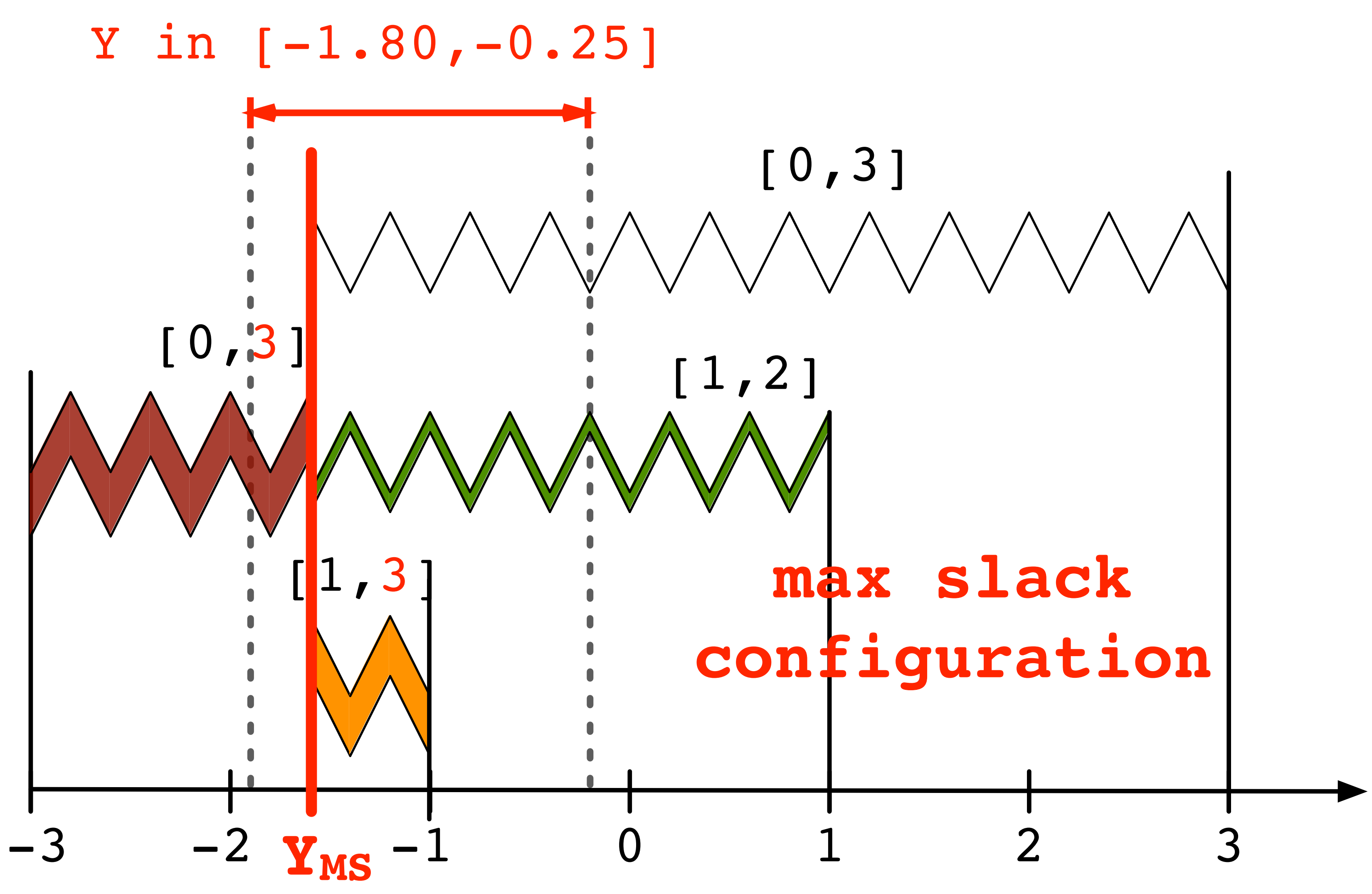
w_i **upper bound =** largest thickness so that the Y boundaries are not crossed



Pruning the Weight Variables

$\texttt{w}_\texttt{i}$ upper bound = largest thickness so that the $\texttt{Y}$ boundaries are not crossed

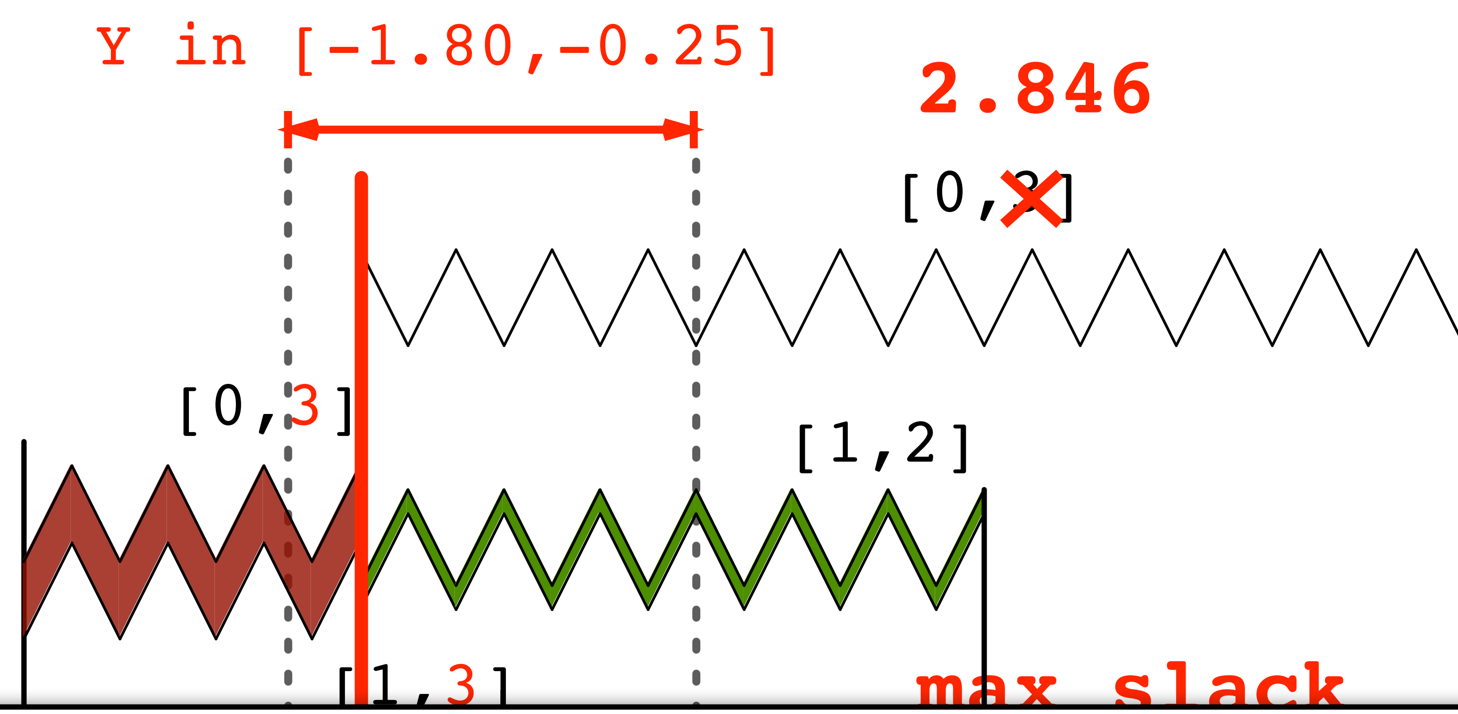
- Maximize $\texttt{w}_\texttt{i}$ if:
  $v_i \leq \max(\text{Y})$
- Minimize $\texttt{w}_\texttt{i}$ if:
  $v_i > \max(\text{Y})$
- UB if $v_i > \max(\text{Y})$
- LB if $v_i < \max(\text{Y})$



Pruning the Weight Variables

$\mathtt{w_i}$ **upper bound =** largest thickness so that the $\mathtt{Y}$ boundaries are not crossed

- Maximize $\mathtt{w_i}$ if:
  $v_i \leq \max(\mathrm{Y})$
- Minimize $\mathtt{w_i}$ if:
  $v_i > \max(\mathrm{Y})$
- UB if $v_i > \max(\mathrm{Y})$
- L

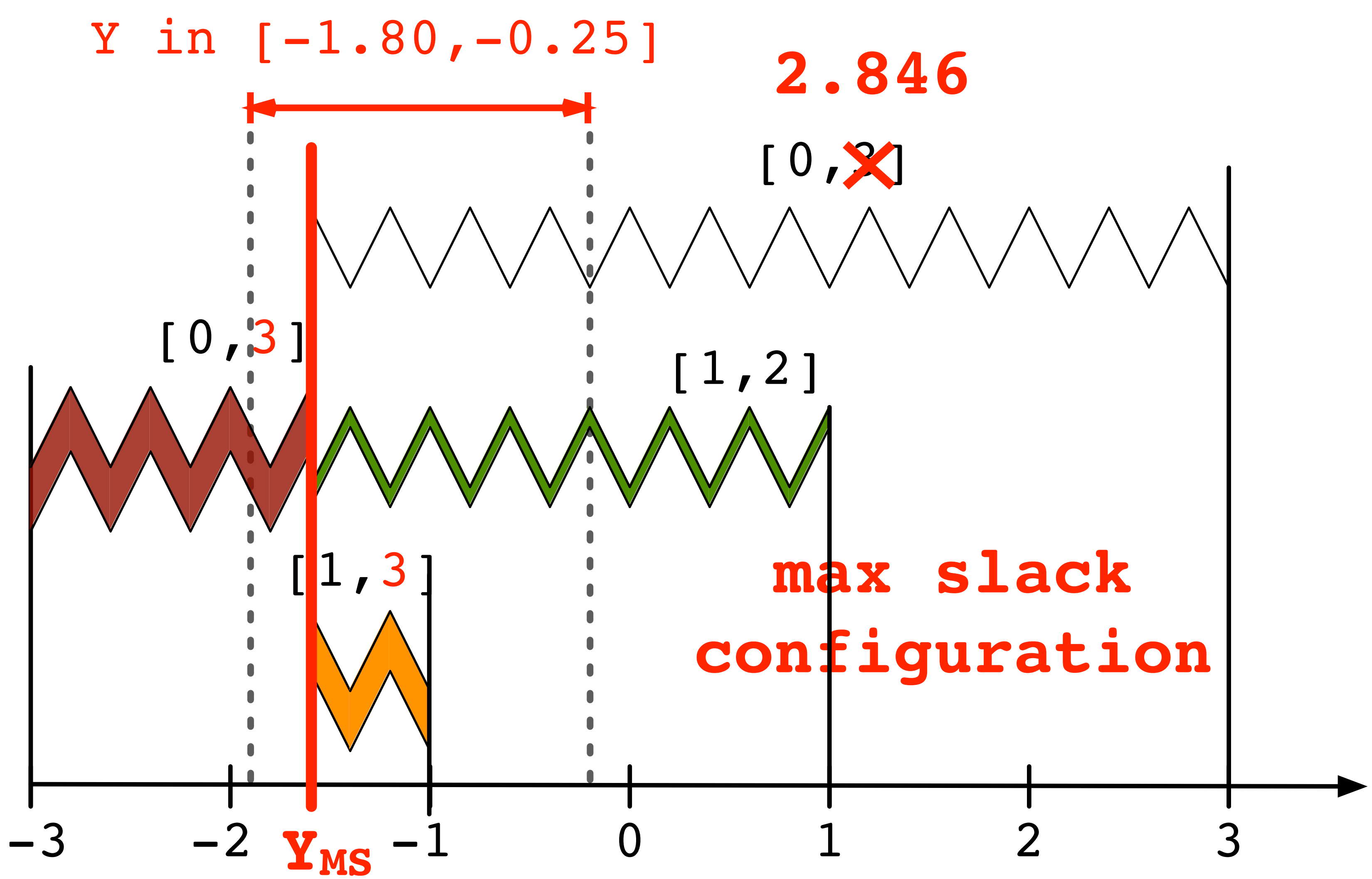


By solving: $$\frac{\mathrm{num}(\mathrm{Y_{MS}}) + (\max(\mathtt{W_i}) - \min(\mathtt{W_i})) \cdot v_i}{\mathrm{den}(\mathrm{Y_{MS}}) + (\max(\mathtt{W_i}) - \min(\mathtt{W_i}))} \leq \max(\mathrm{Y})$$

Pruning the Weight Variables

w_i upper bound = largest thickness so that the Y boundaries are not crossed

- Maximize w_i if:
  $$v_i \leq \max(Y)$$
- Minimize w_i if:
  $$v_i > \max(Y)$$
- UB if $v_i > \max(Y)$
- LB if $v_i < \max(Y)$
- WC complexity: *O(n)*



Incremental Filtering

Problems of this class can grow pretty large:

- Thermal Aware Workload Dispatching: 120 to 480 jobs
- Fair Capacitated Facility Location: 50 customer, 16-50 locations

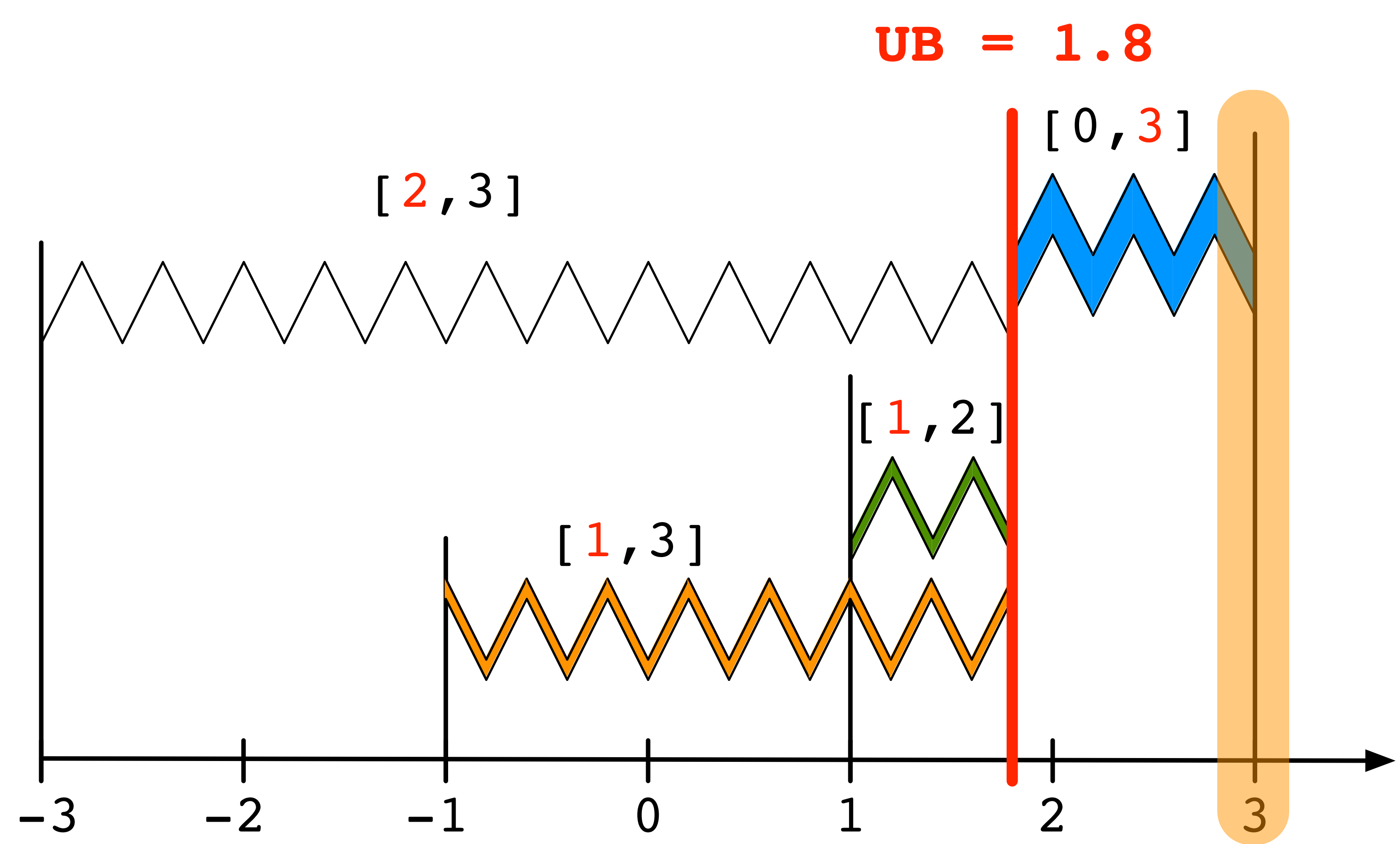
Incremental filtering can save a lot of computation time

- Rules for the fixed values case
- Particularly effective for {0,1} weights

Incremental Filtering for the `Y` Variable

Store:

- `num(`Y_{UB}`)/den(`Y_{UB}`)`
- Index of the last maximized w_i

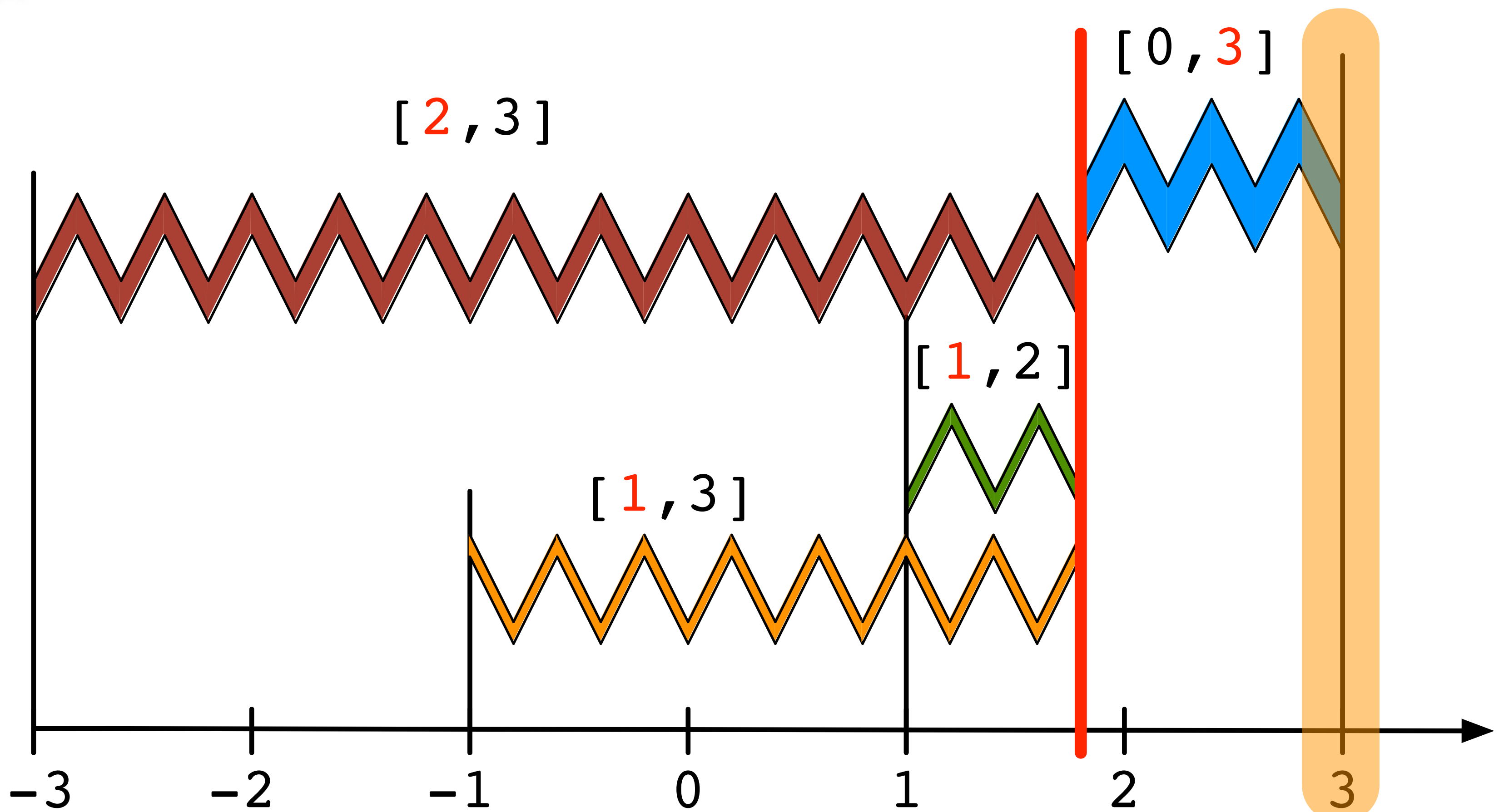


Incremental Filtering for the Y Variable

Store:

- num(Y_{UB})/den(Y_{UB})
- Index of the last maximized w_i

When a weight changes:



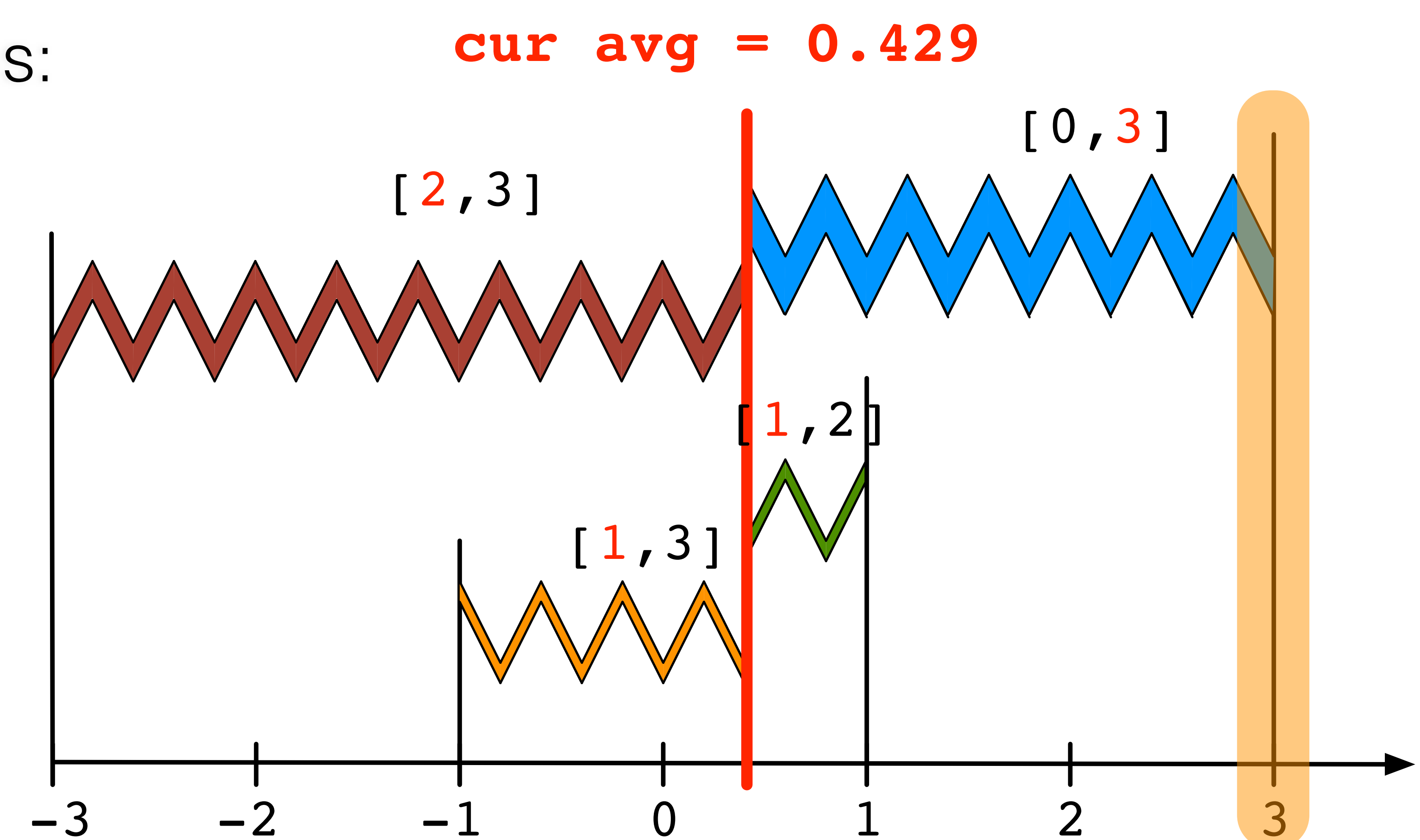
Incremental Filtering for the Y Variable

Store:

- num(Y_UB)/den(Y_UB)
- Index of the last maximized w_i

When a weight changes:

- Update current avg



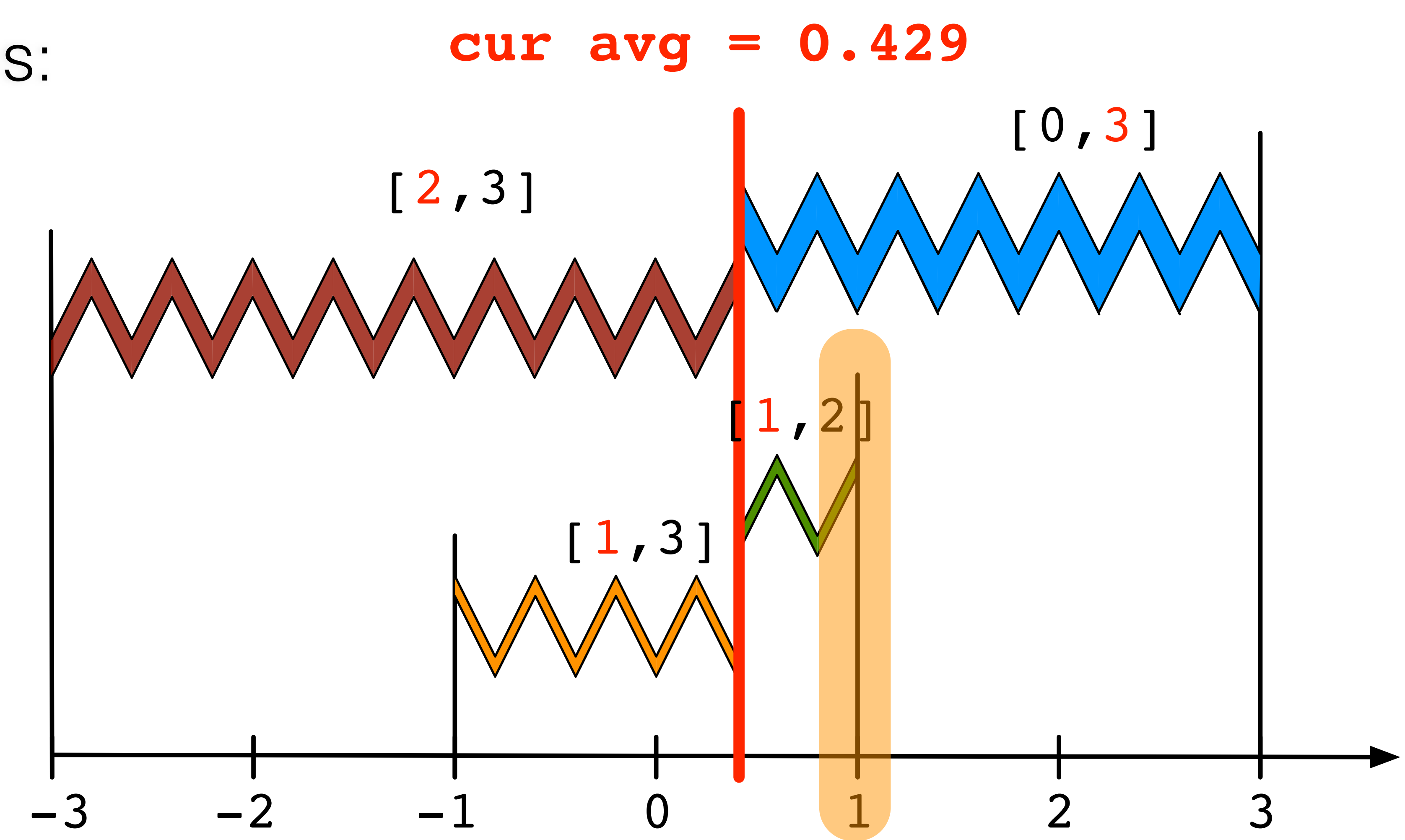
Incremental Filtering for the Y Variable

Store:

- num(Y$_{UB}$)/den(Y$_{UB}$)
- Index of the last maximized W$_i$

When a weight changes:

- Update current avg



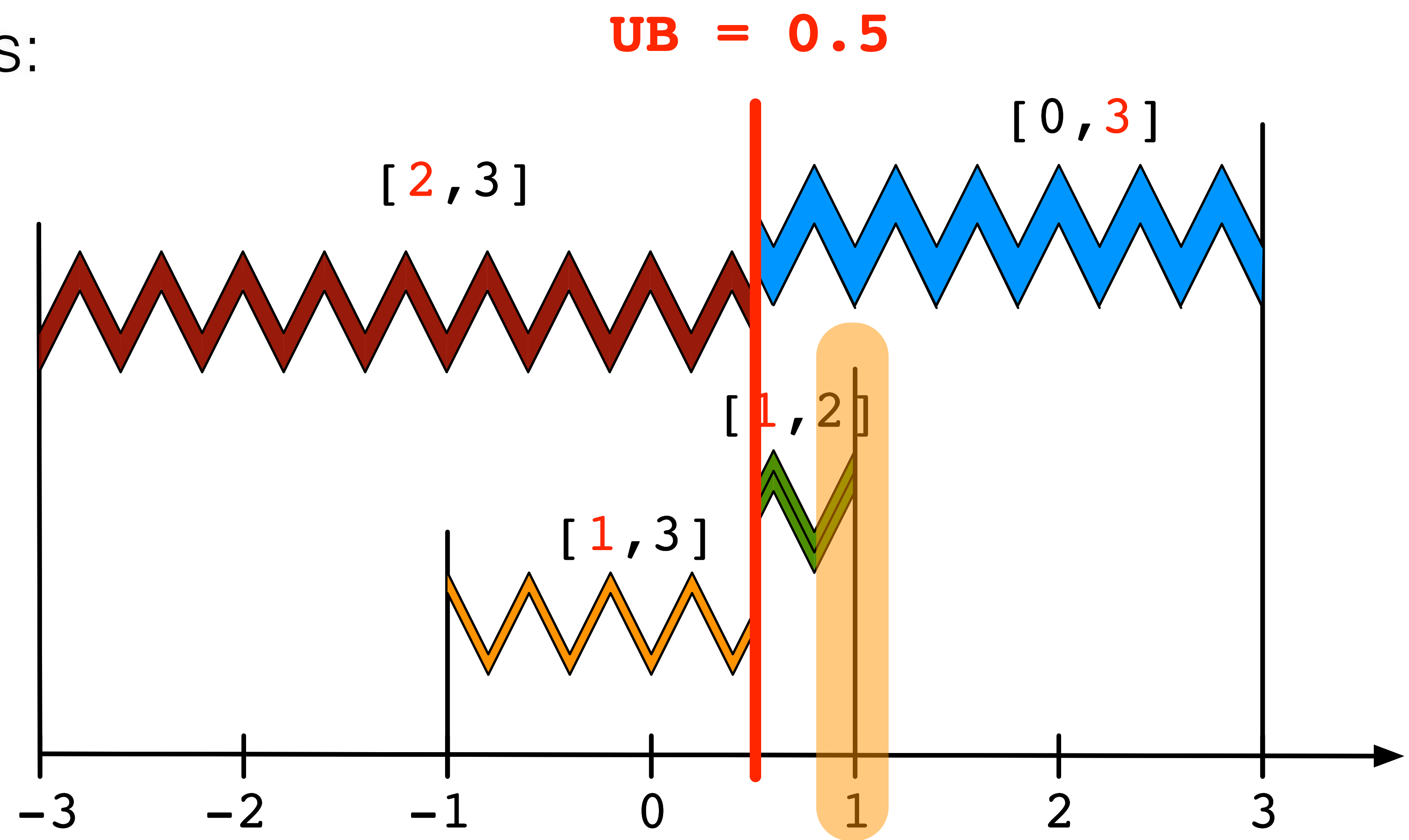
Incremental Filtering for the `Y` Variable

Store:

- `num(Y_UB)/den(Y_UB)`
- Index of the last maximized w_i

When a weight changes:

- Update current avg
- Maximize new w_i
- No more than *n* shifts
- No more than n × dom size updates
- WC complexity *O(n × dom size)*



Experimental Results

Experiments on:

- Capacitated Facility Location (max worst case average travel time)
- Thermal aware workload dispatching: max worst case efficiency

Benchmarks:

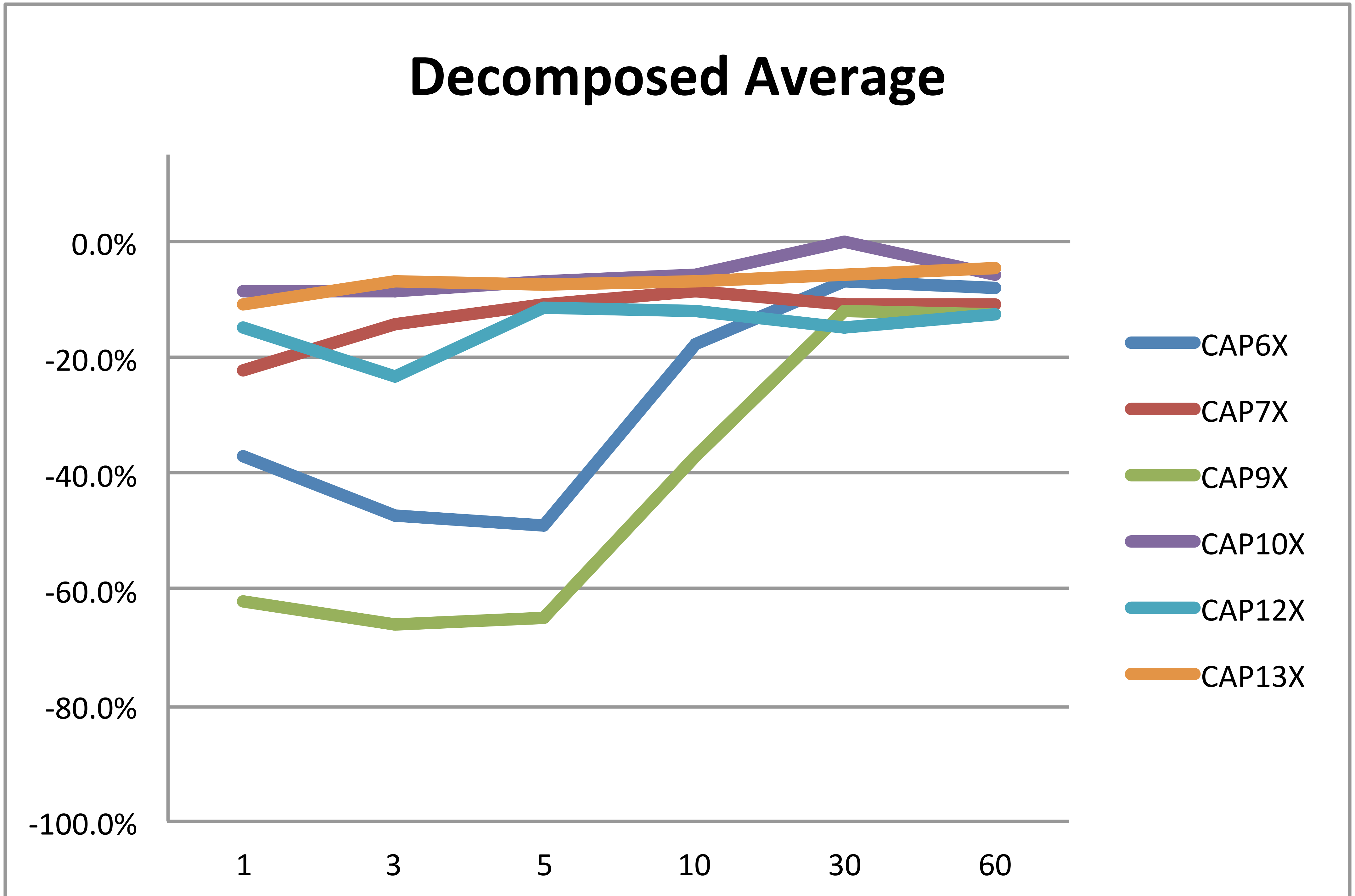
- Problem #1: Single Source instances by Beasley in the OR-Library
- Problem #1: custom (publicly available) instances

Solution method (goal: testing constraint propagation):

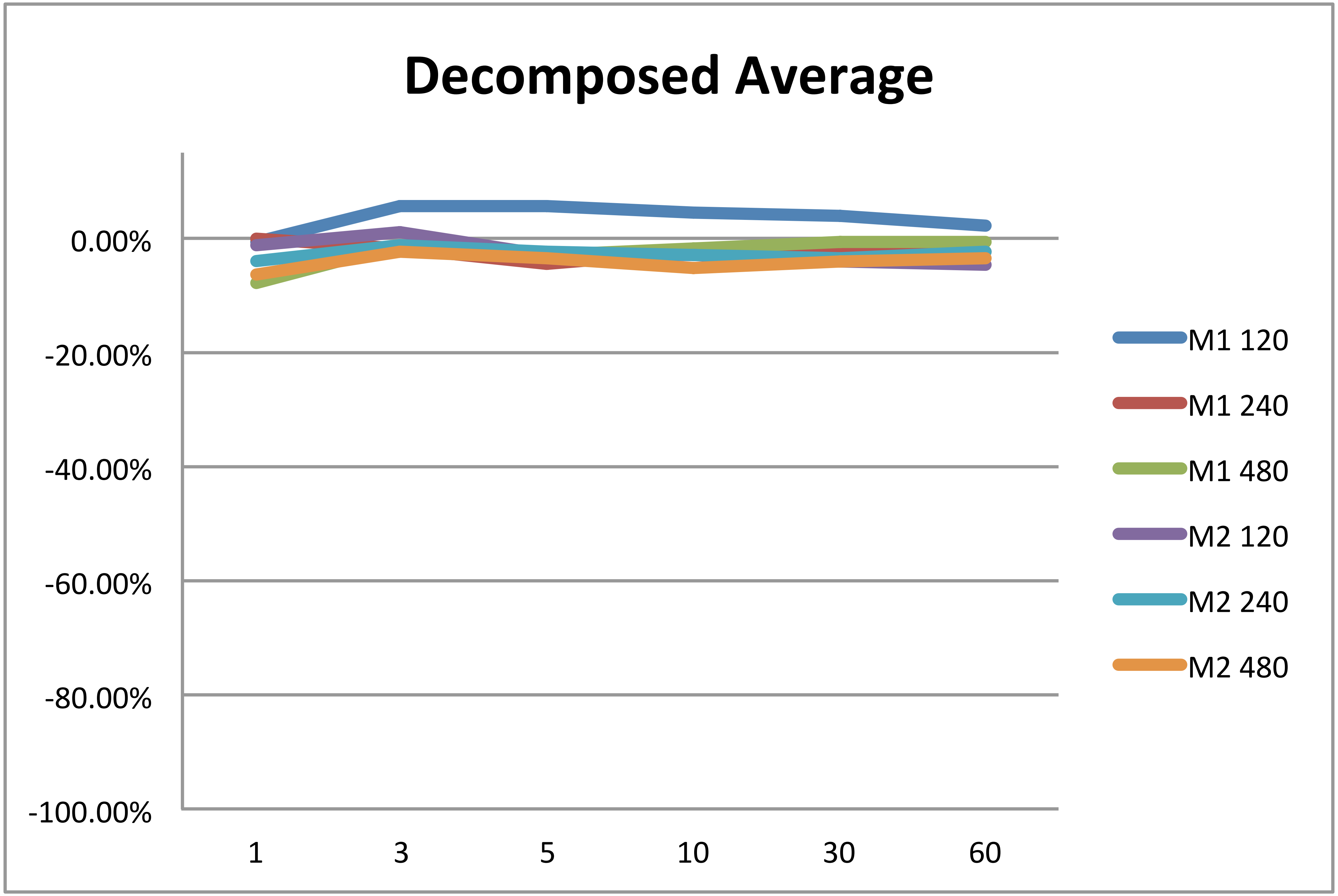
- Random restarts with fixed threshold
- Random variable and value selection

Compare with competitor approaches

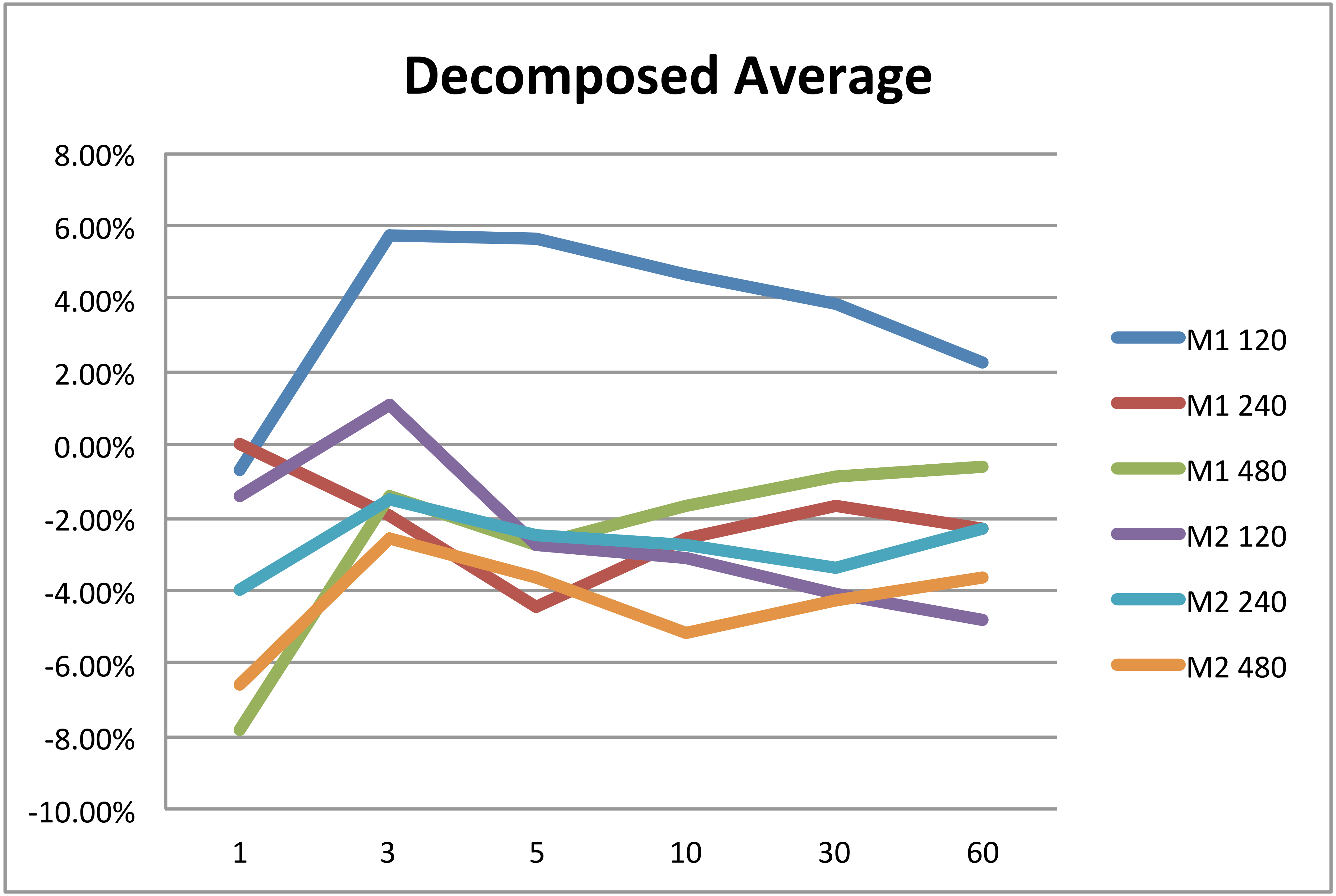
Results for Capacitated Facility Location



Results for Thermal Aware Dispatching



Results for Thermal Aware Dispatching



Conclusion

Main results

- A global constraint for weighted average expressions
- Useful for allocation problems with balancing components
- (Incremental) Filtering algorithms

Future work directions

- Apply incremental filtering ideas from the sum constraint
- More application scenarios
- Devise constraints for other classical inputs to machine learning models

Thanks!

Questions?

Funded by a
Google
grant

DEIS, University of Bologna