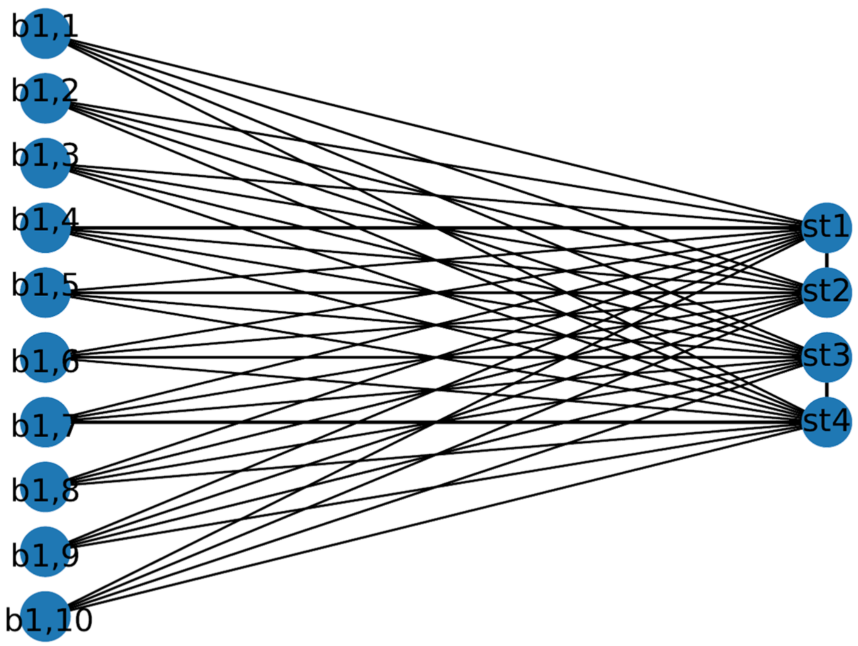
Efficient batch assignment for parallel-machine production scheduling

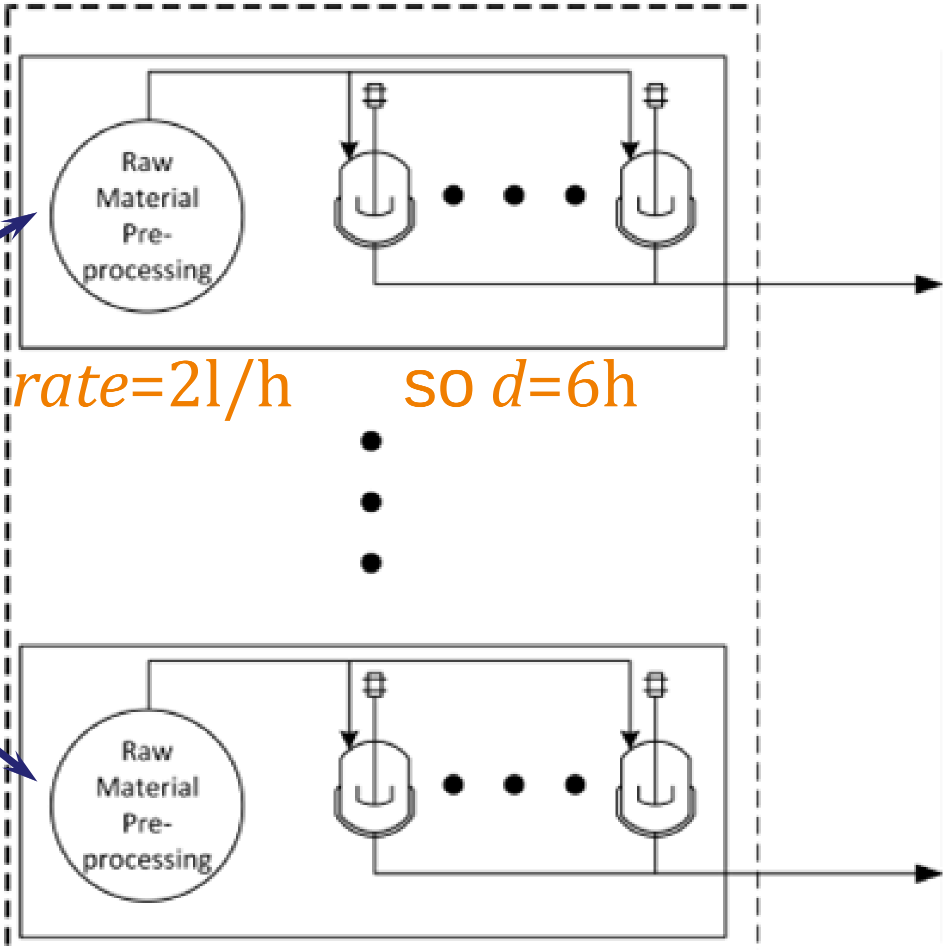
Christoph Fünfzig	CFS Software
Roderich Wallrath	Engineering & Technology, Bayer AG
Stefan Hubert	Engineering & Technology, Bayer AG

Planning of Batch Production

batches, grouped by parameters:
product type $u=1,2,3,..$
volume $v=12\mathrm{l}$



1. Production
2. Transfer to Filling



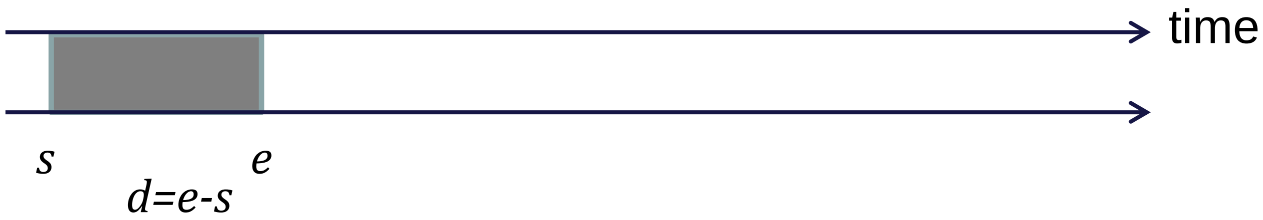
$rate=2\mathrm{l}/\mathrm{h}$ so $d=6\mathrm{h}$

$rate=4\mathrm{l}/\mathrm{h}$ so $d=3\mathrm{h}$

Formulation Lines

Scheduling with CP

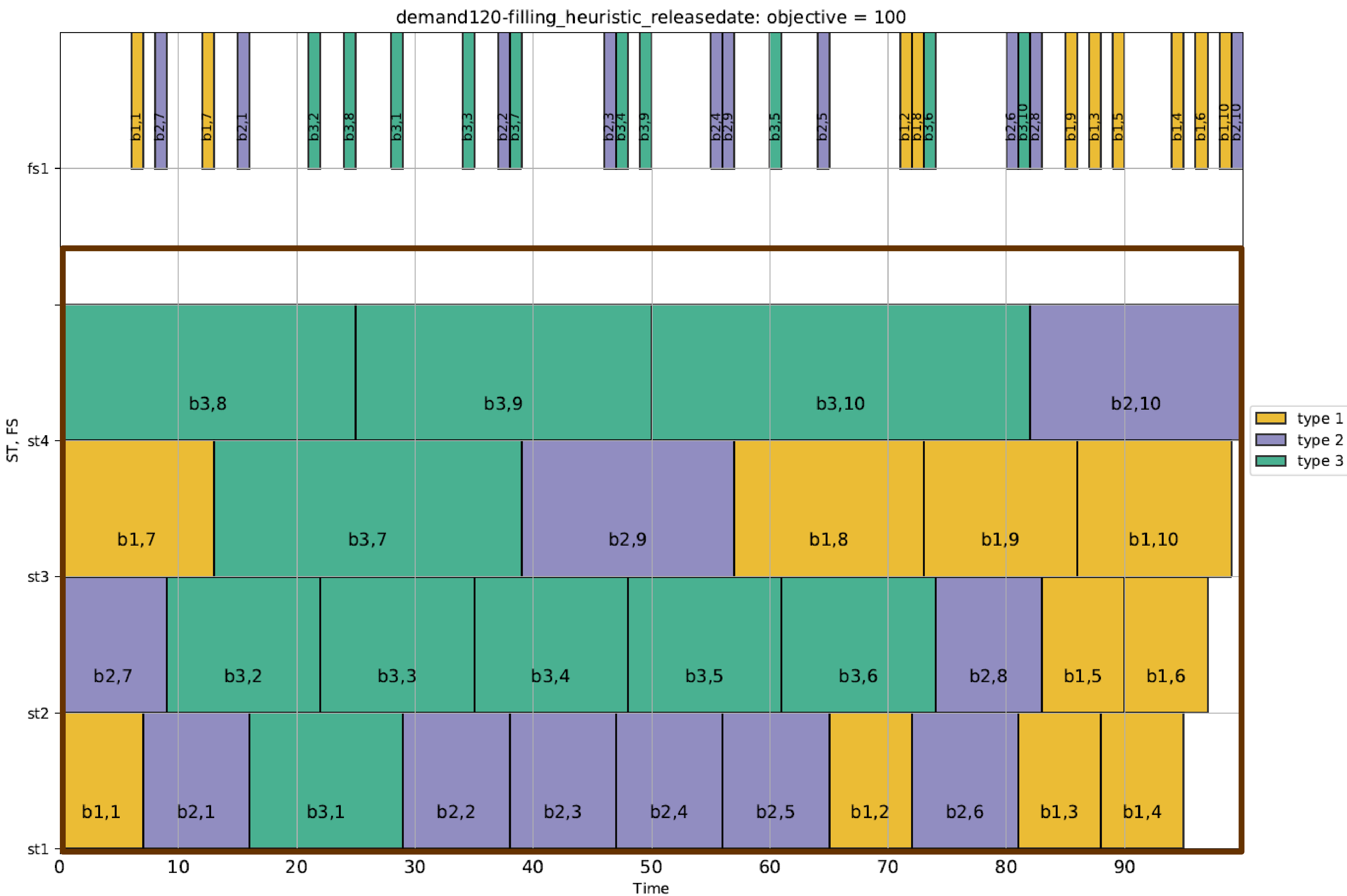
- *Interval* $I=[s, e[, s \leq e$ of duration $d:=e-s$ (for scheduling):



- *OptionalInterval* $(S_{i,j}; I_{i,j})$ (for assignment & scheduling):
  $S_{i,j}=1$ iff batch j processed on machine i $(1 \leq i \leq m, 1 \leq j \leq n)$

- Global constraints on each machine *i:*
  disjunctive processing (*NoOverlap*)
  ordering of processing intervals in time ($e(I_{i,j-1}) \leq s(I_{i,j})$ for $j>1$)

Gantt chart example (heuristic filling $12 \mathrm{l/h}$)



d=3h, 6h
d=4h, 8h
d=6h, 12h

Assignment & Scheduling

- Seen the scheduling part
  but how to do assignment to machines?
- General assignment or Blocks assignment
  (Contribution: Scheme with small #feasible assignments)

General assignment

batch index j

machine index i

$S_{1,1}; I_{1,1}$	$S_{1,2}; I_{1,2}$	$S_{1,3}; I_{1,3}$	$S_{1,4}; I_{1,4}$
$S_{2,1}; I_{2,1}$	$S_{2,2}; I_{2,2}$	$S_{2,3}; I_{2,3}$	$S_{2,4}; I_{2,4}$
$S_{3,1}; I_{3,1}$	$S_{3,2}; I_{3,2}$	$S_{3,3}; I_{3,3}$	$S_{3,4}; I_{3,4}$
$S_{4,1}; I_{4,1}$	$S_{4,2}; I_{4,2}$	$S_{4,3}; I_{4,3}$	$S_{4,4}; I_{4,4}$

$j=1: \sum_i S_{i,j} = 1$ $j=2$ $j=3$ $j=4$

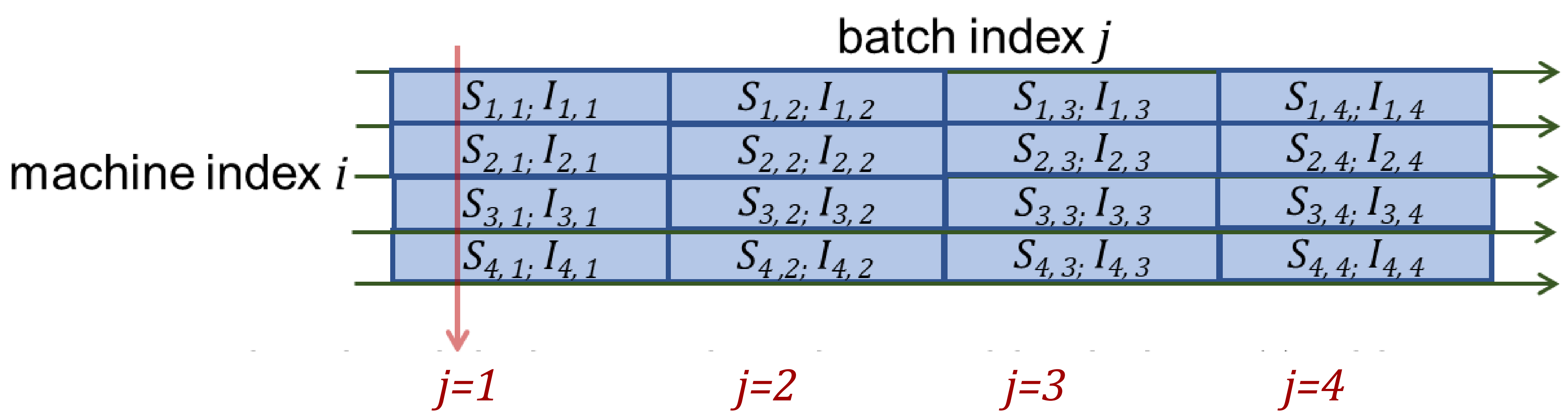
Exactly-n

batch index j

machine index i

$S_{1,1}=1$	$S_{1,2}=0$	$S_{1,3}=1$	$S_{1,4}=0$	$z_1 = \sum_j S_{1,j} = 2$
$S_{2,1}=0$	$S_{2,2}=1$	$S_{2,3}=0$	$S_{2,4}=1$	$z_2=2$
				$z_3=0$
				$z_4=0$

Blocks assignment



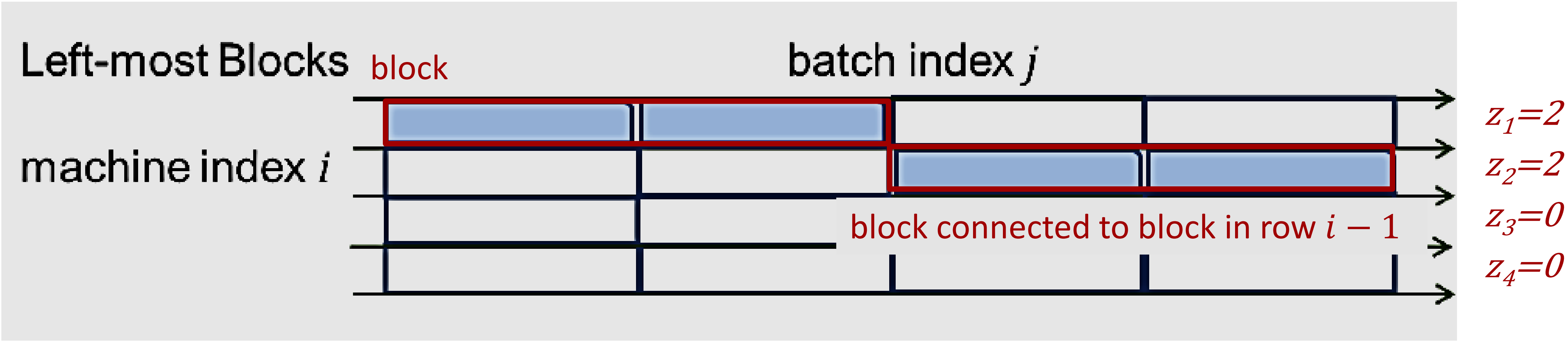


Table constraints formulation

- AllowedAssignments(variables, tuples_list)

$[\, S_{i,1}, .., S_{i,n} \,]$ one of

$[\, 0, 0, 0, 0 \,]$ z_i=0

$[\, 1, 0, 0, 0 \,]$ $[\, 0, 1, 0, 0 \,]$ $[\, 0, 0, 1, 0 \,]$ $[\, 0, 0, 0, 1 \,]$ z_i=1

$[\, 1, 1, 0, 0 \,]$ $[\, 0, 1, 1, 0 \,]$ $[0, 0, 1, 1 \,]$ z_i=2

$[\, 1, 1, 1, 0 \,]$ $[\, 0, 1, 1, 1 \,]$ z_i=3

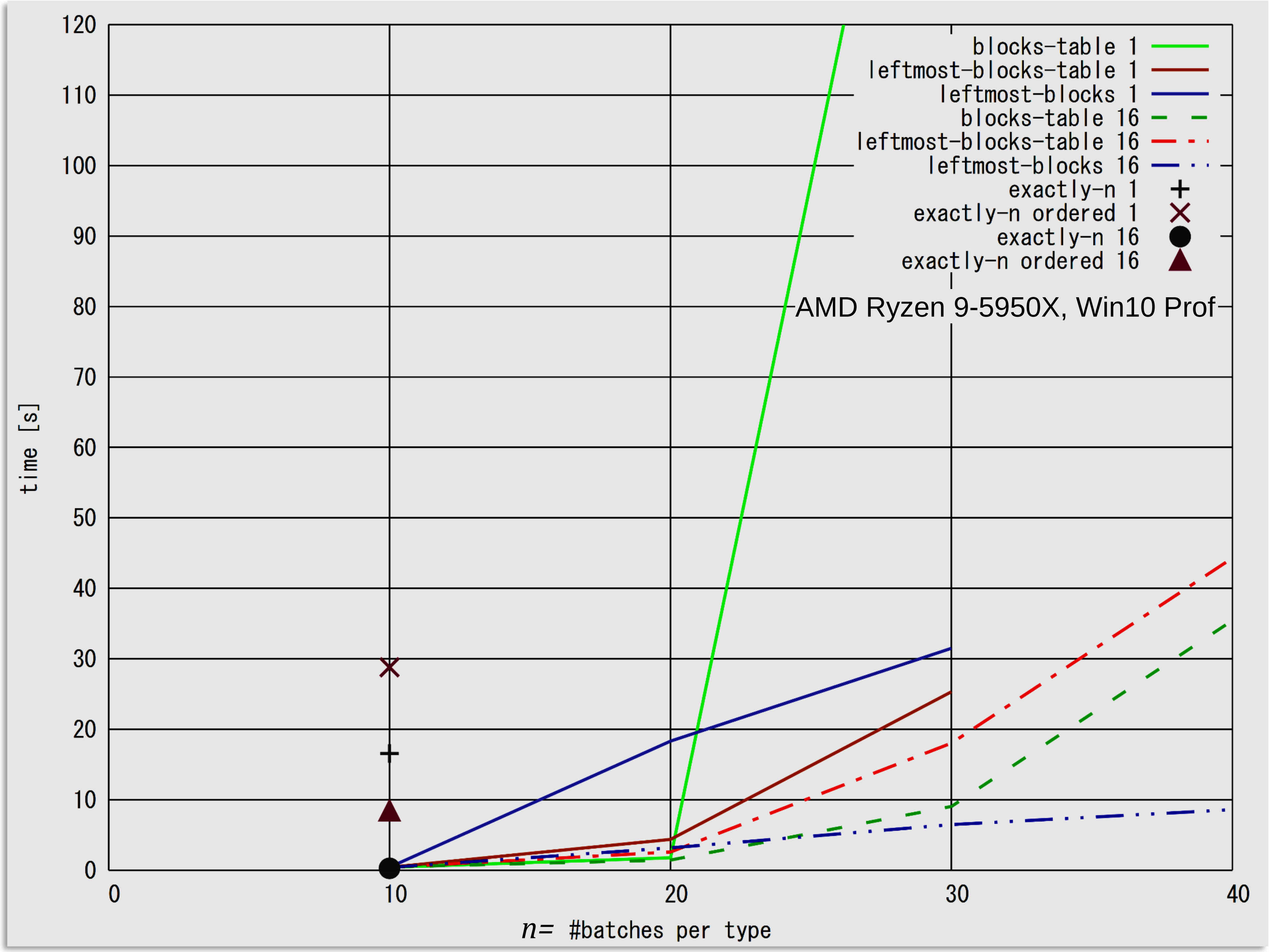
$[\, 1, 1, 1, 1 \,]$ z_i=4

Pure-boolean formulation

- New boolean variables $seq_{i,j}$
  $seq_{i,j}=1$ iff change from batch $j\text{-}1$ to j $(j>1)$
- Constraint per machine i
  $S_{i,1} + \Sigma_{j>1}\, seq_{i,j} + S_{i,n} = 0$ or 2

Additionally:
blocks can be forced to be *left-most*, starting with machine $i=1$ (*left-most blocks, left-most blocks table*).

Results: solution times for *m=4*, #types=*3*



Conclusion

- *Left-most blocks assignment*
  number of feasible assignments smaller,
  polynomial in number n of batches:

$$\boldsymbol{K(n,m)} = \sum_{k=1}^{m} \binom{m}{k} G(n,k) = \sum_{k=1}^{m} \binom{m}{k} \binom{n-1}{k-1} \leq m \cdot m^{\left\lfloor \frac{m}{2} \right\rfloor} \cdot (n-1)^{m-1}$$

- *Table constraints formulation* shorter and simpler.
- *Pure-boolean formulation* longer due to additional bools
  but benefits from clause learning inside CP-SAT.