Numerical comparison of Kanban mechanisms for production systems with time-dependent processing times

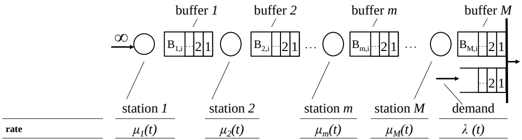
Justus Arne Schwarz
joint work with Raik Stolletz

SMMSO 2017, June 2017, Lecce Italy

Agenda

1. Introduction & Motivation
2. Problem formulation: Proactive Kanban Allocation Problem
3. Solution approaches
   - Sectioning algorithm
   - Simplifying heuristics
4. Numerical insights
5. Conclusions and future research

Motivation - time-dependent flow lines



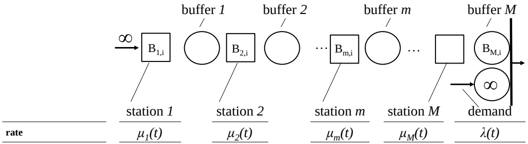
- Exponentially distributed times between demand arrivals: rate $\lambda(t)$.
- Exponentially distributed processing times at station m: rate $\mu_m(t)$.

Time-dependent impacts

- Seasonal demand
- Replacement of machinery
- Learning effects

(Takahashi and Nakamura, 2002; Jaikumar and Bohn, 1992; Terwiesch and Bohn, 2001)

Motivation - time-dependent flow lines



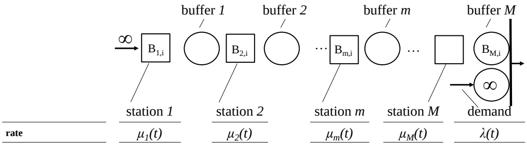
- Exponentially distributed times between demand arrivals: rate $\lambda(t)$.
- Exponentially distributed processing times at station m: rate $\mu_m(t)$.

Time-dependent impacts

- Seasonal demand
- Replacement of machinery
- Learning effects

(Takahashi and Nakamura, 2002; Jaikumar and Bohn, 1992; Terwiesch and Bohn, 2001)

Motivation - time-dependent flow lines



- Exponentially distributed times between demand arrivals: rate $\lambda(t)$.
- Exponentially distributed processing times at station m: rate $\mu_m(t)$.

Time-dependent impacts

- Seasonal demand
- Replacement of machinery
- Learning effects

(Takahashi and Nakamura, 2002; Jaikumar and Bohn, 1992; Terwiesch and Bohn, 2001)

Should buffer capacities be changed over time?

if, so

How should buffer capacities be changed over time?

Related literature

- Flow lines correspond to Kanban systems (Berkley, 1991)
- Kanban cards allow for the flexible adaption of buffer capacities

Used information to trigger reconfiguration

Past data	Current system status	Time-dependent distribution parameters
Reactive Kanban Takahashi&Nakamura 2002 Takahashi 2003 Takahashi et al. 2004	**Adaptive Kanban** Tardif & Maaseidvaag 2001	**Proactive Kanban** Today‘s talk

- Existing approaches *react* to changes in demand
  Time-dependent processing times (Schwarz and Stolletz 2017)

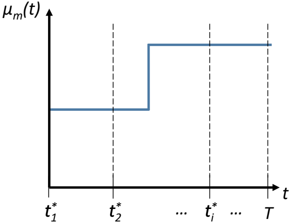
New approach to *plan* for changes of flow line parameters

Agenda

Proactive Kanban Systems

Assumptions:

- Planing horizon is divided in I intervals
- Maximum of one change of the allocation per interval
- Allocation is changed at first demand arrival in each interval



Indices

$m = 1, ..., M$	Stages in the flow line
$i = 1, ..., I$	Intervals

Parameters

T	Length of planning horizon
b_m^{max}	Maximum buffer capacity at stage m
$0 = t_1^*, ..., t_i^*, ..., t_I^*$	Beginning of the ith interval

Decision variables

$\mathbf{B}$	Buffer allocation matrix, with
$B_{m,i}$	Planned buffer capacities at stage m in interval i

Proactive Kanban Card Setting Problem

$$\text{minimize } \mathrm{E}[W(\mathbf{B})] \tag{1a}$$

s.t.:

$$SL^{\alpha}(\mathbf{B}) \geq \alpha^{*} \tag{1b}$$

$$0 \leq B_{m,i} \leq b_{m}^{max} \qquad \forall m, \forall i \tag{1c}$$

with

Dependent variables

$W(\mathbf{B})$	Average WIP in the line over the planning horizon
$SL^{\alpha}(\mathbf{B})$	Achieved α-service level during the planning horizon

- Steady-state version of the problem addressed by Duri et al. (2000)

Solution approaches

Simulation-optimization approach, accounting for time-dependency

- Establish monotonicity properties
- Derive dominance criteria for allocations
- Derivation of upper and lower bounds
- Exploit dominance criteria and bounds in a sectioning algorithm

Simplifying heuristics, relaxing time-dependency

- Partially ignore time-dependency and application of steady-state methods

Impact of t-d. buffer capacities on expected WIP and service level

Theorem 1

For $M = 1$ the cumulated no. of departed workpiece $D_M(t)$ from station M and $D_{M+1}(t)$ from buffer M are stochastically increasing in $B_{M,i}$ $\forall i$.

Corollary 1.2

$SL^\alpha(\mathbf{B})$ is increasing in $B_{M,i}$ $\forall i$.

Corollary 1.1

$\mathrm{E}[W(\mathbf{B})]$ is increasing in $B_{M,i}$ $\forall i$.

Proof: Based on sample path arguments

- In contrast to steady-state $\mathrm{E}[W(\mathbf{B})]$ is not necessarily convex
- Numerical studies suggest: Corollary 1.1 and 1.2 hold for $M > 1$ and generally distributed processing times and demand

Dominance criteria

- Based on Corollary 1.1 and 1.2:

Every *feasible* allocation **B** with $SL^{\alpha}(\mathbf{B}) \geq \alpha^*$

- excludes all $\mathbf{B}'$ with $B'_{m,i} \geq B_{m,i} \; \forall m, \forall i$,
- and $E[W(\mathbf{B})]$ provides an upper bound (UB_W) on the objective value

Every *infeasible* allocation **B** with $SL^{\alpha}(\mathbf{B}) < \alpha^*$

- excludes all allocations $\mathbf{B}'$ with $B'_{m,i} \leq B_{m,i} \; \forall m, \forall i$
- if in addition $E[W(\mathbf{B})] > UB_W$ holds
  - exclude all allocations $\mathbf{B}'$ with $B'_{m,i} \geq B_{m,i} \; \forall m, \forall i$

Derivation of (conditional) upper and lower bounds

For a given buffer m' in interval i':

Determine (conditional) lower bound on $B_{m',i'}$

- Set all not considered buffers to upper bound (or fixed value)
- Search for smallest value of $B_{m',i'}$ such that $SL^{\alpha} \geq \alpha^*$

Determine (conditional) upper bound on $B_{m',i'}$

- Set all not considered buffers to lower bound (or fixed value)
- Search for smallest value of $B_{m',i'}$ such that $SL^{\alpha} \geq \alpha^*$

Sectioning Algorithm - Key ideas

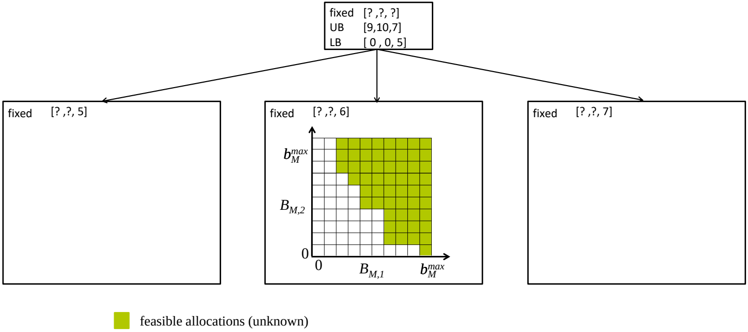
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer

fixed	[? ,?, ?]
UB	[9,10,7]
LB	[0 , 0, 5]

Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

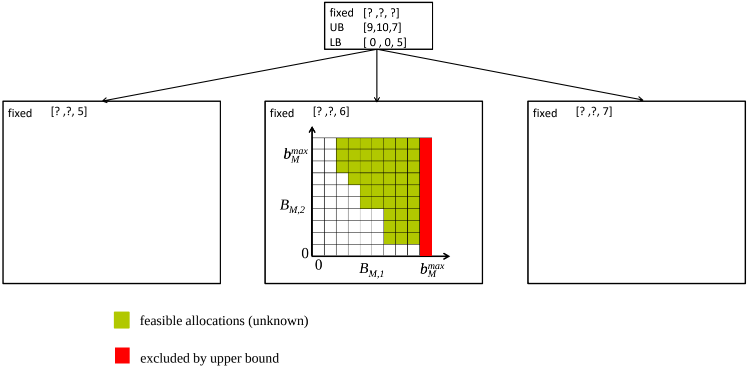
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

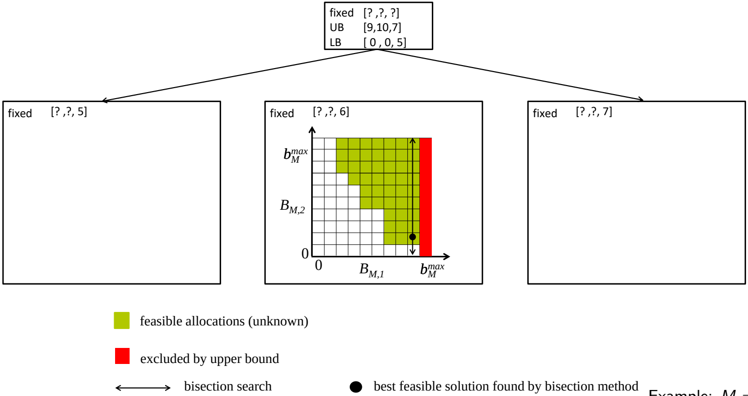
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

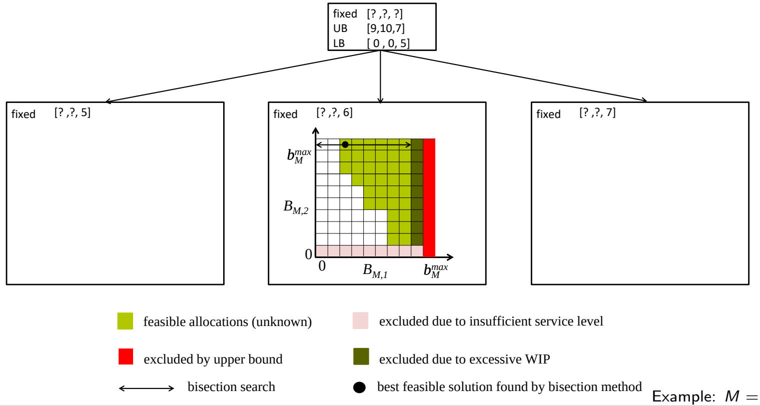
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

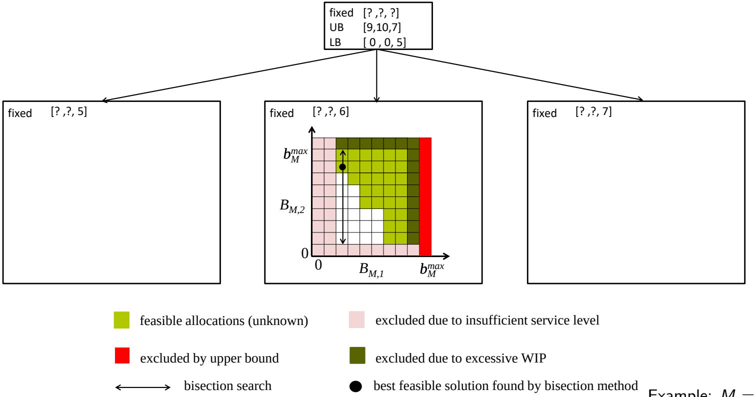
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

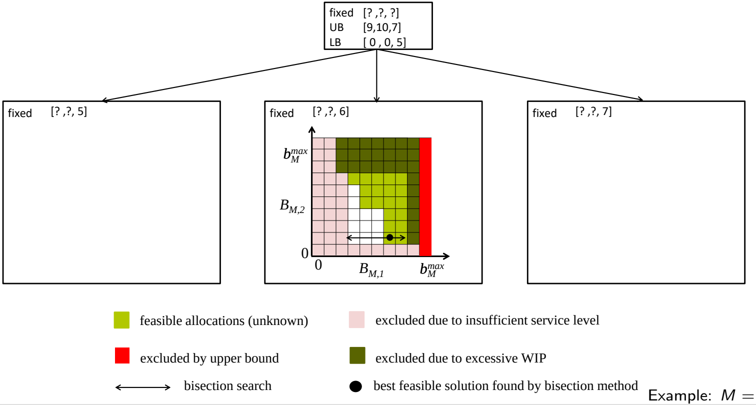
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

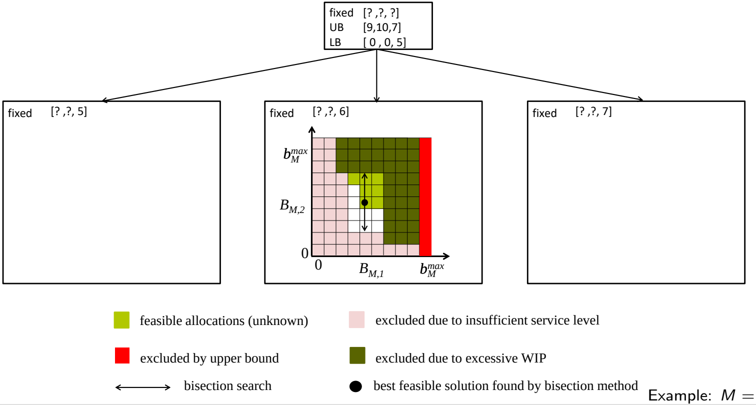
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

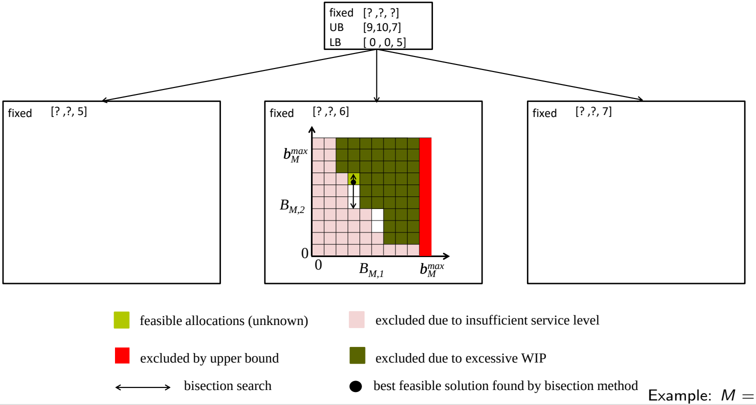
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

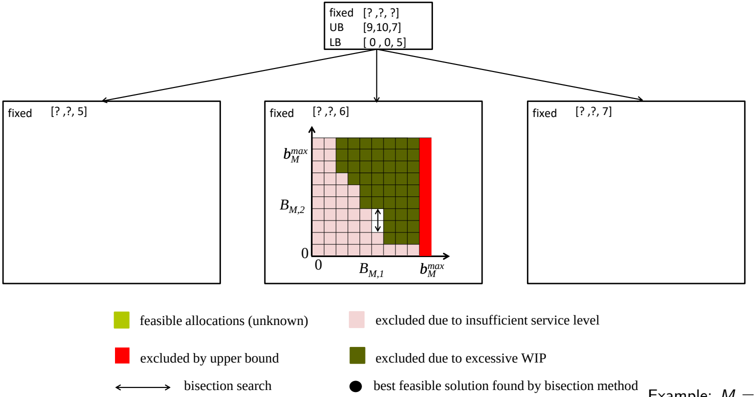
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

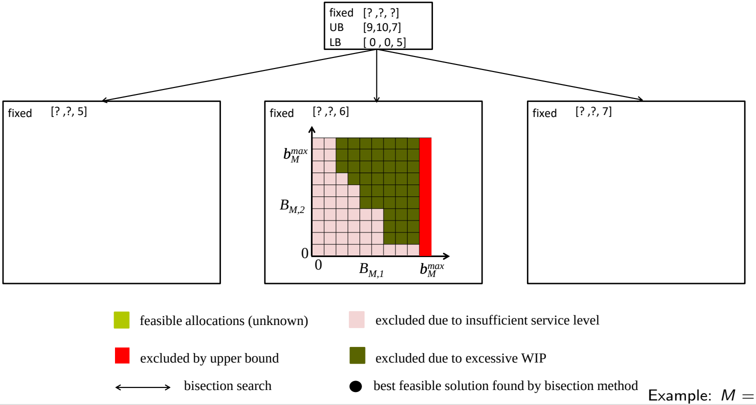
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

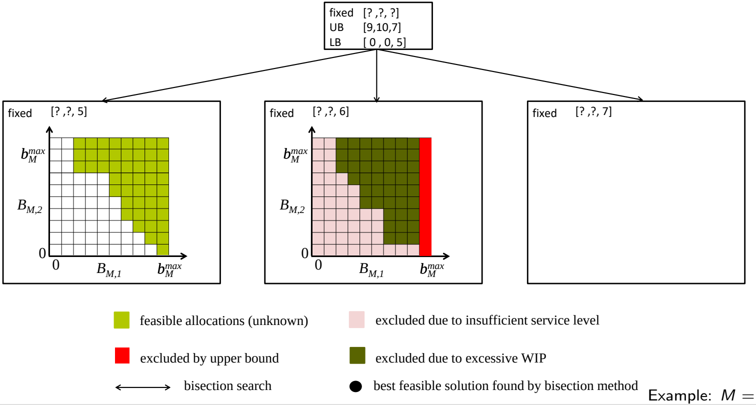
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

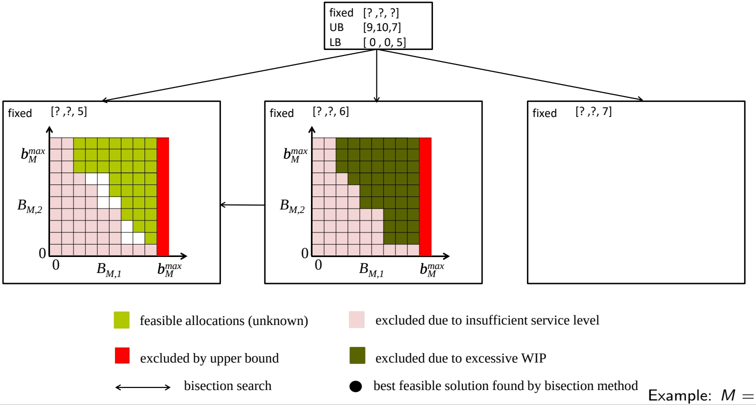
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

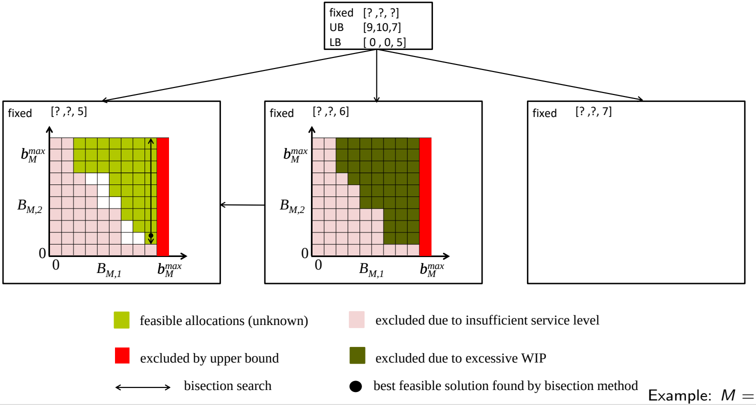
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

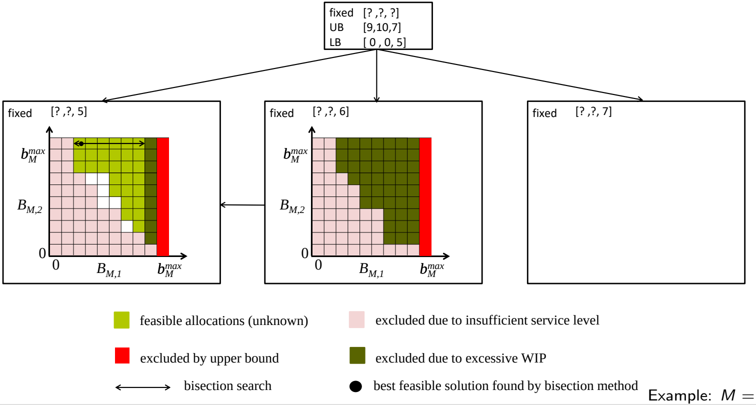
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

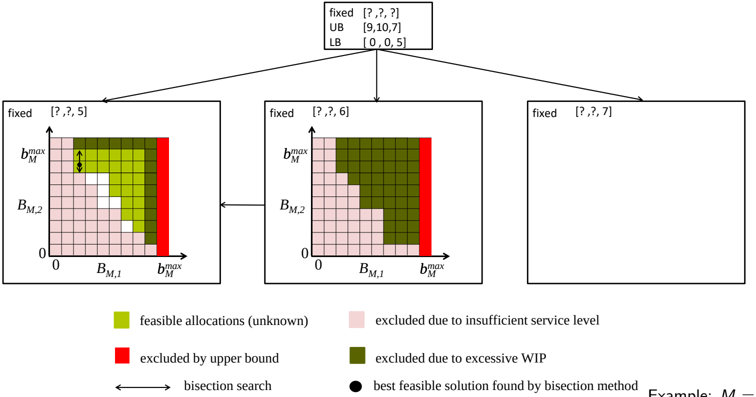
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

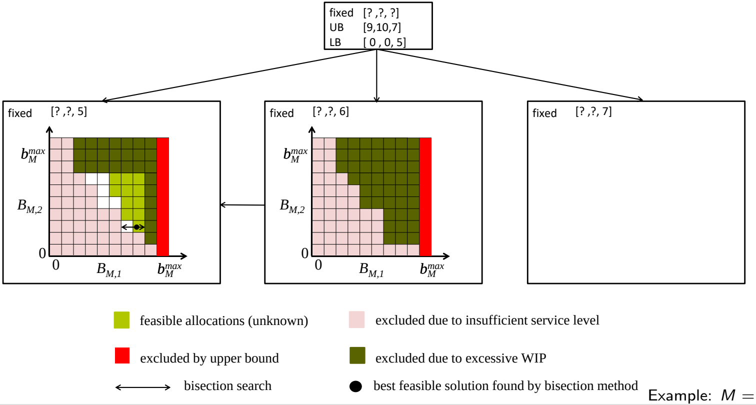
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

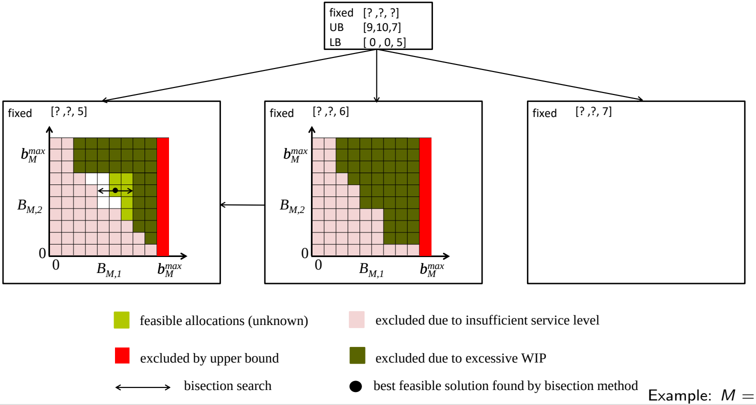
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

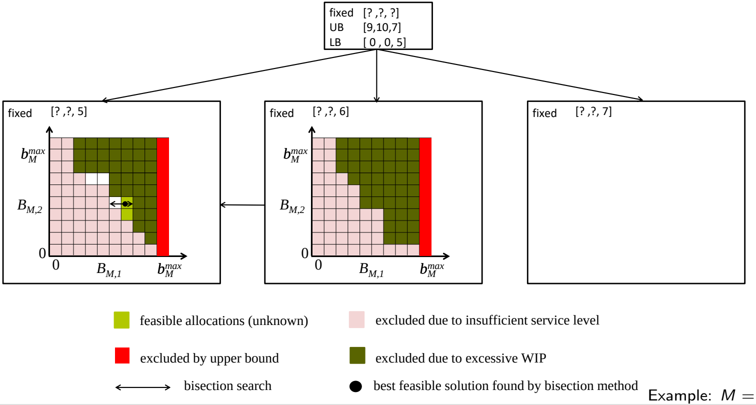
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

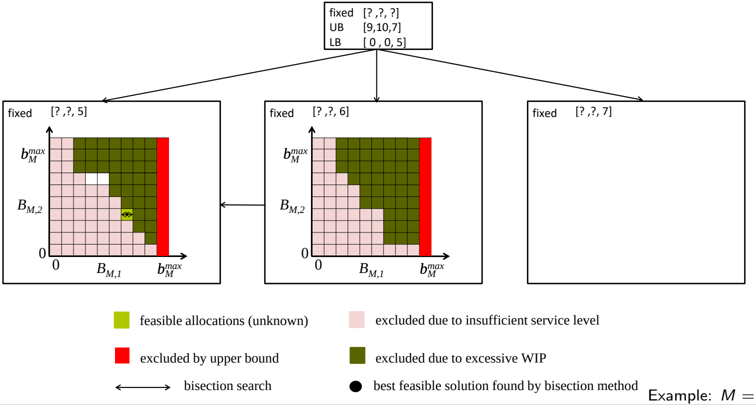
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

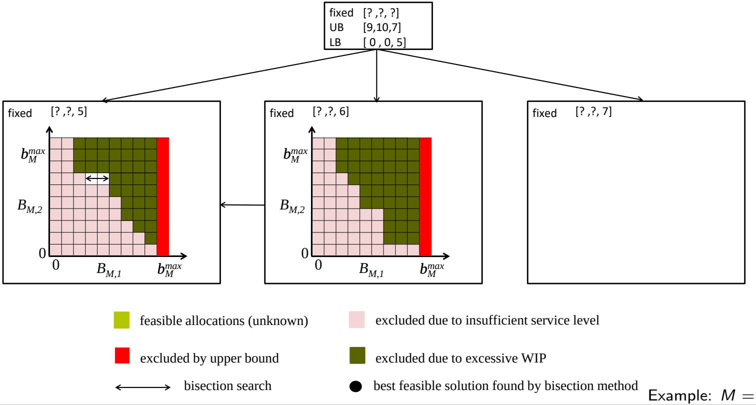
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

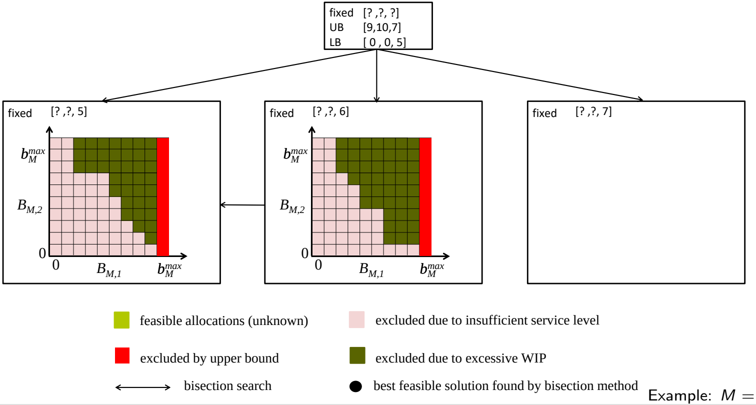
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

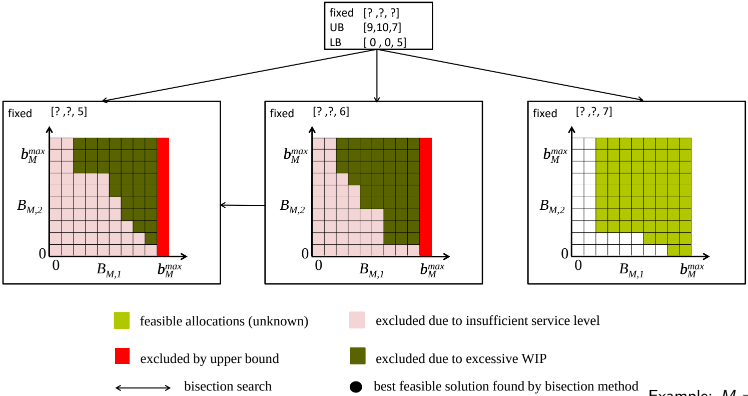
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

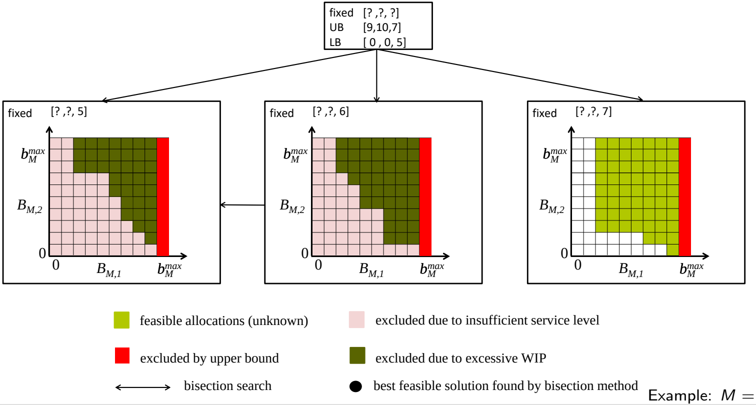
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

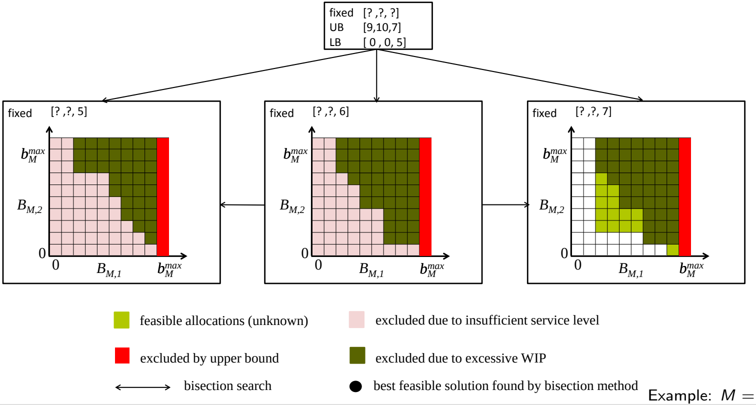
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

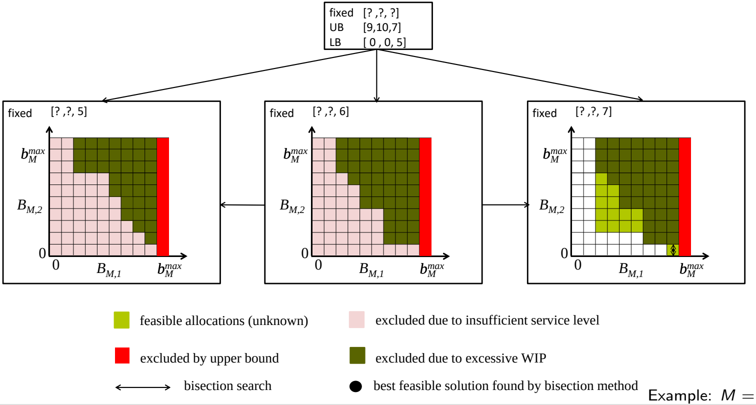
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

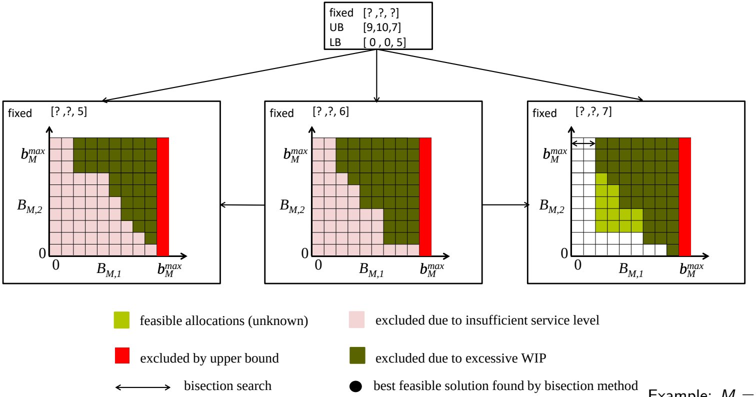
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

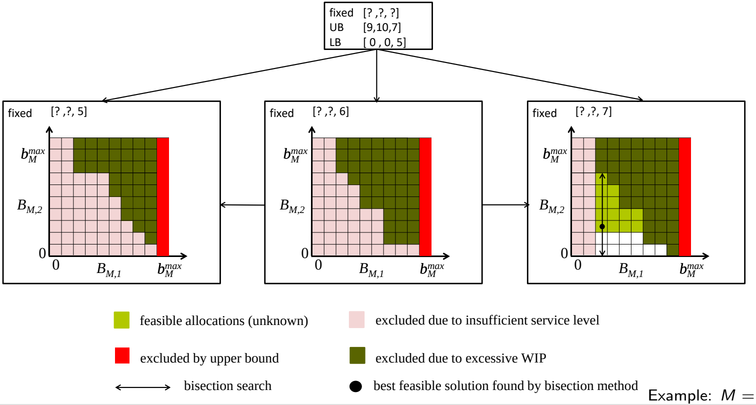
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

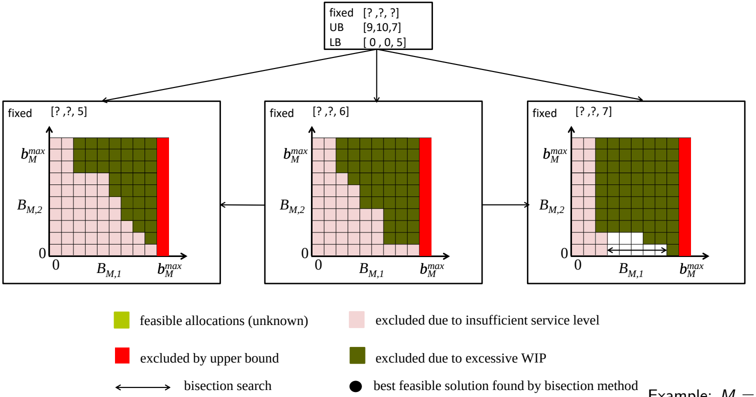
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

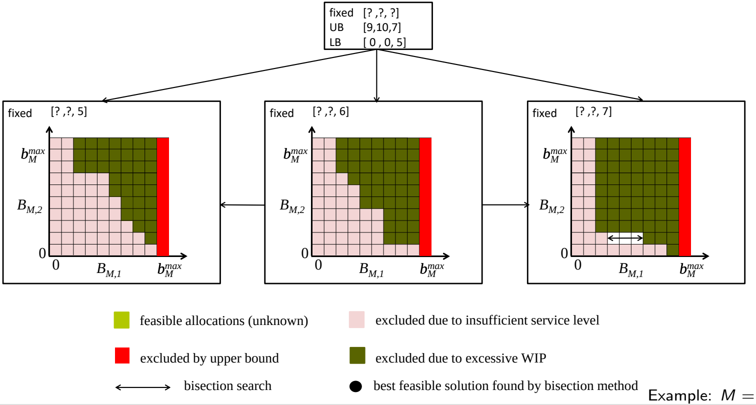
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

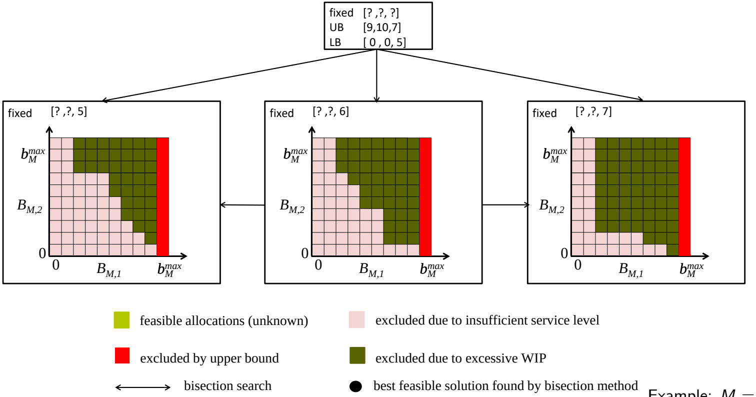
- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Key ideas

- while (no. of not fixed buffers $B_{m,i} > 1$)
  - Determine conditional lower and upper bounds
  - Determine buffer to be fixed and fix its capacity
- Search for smallest feasible value of unfixed buffer



Example: $M = 1$, $I = 3$, $b_M^{max} = 10$

Sectioning Algorithm - Summary

Recursive algorithm formulation

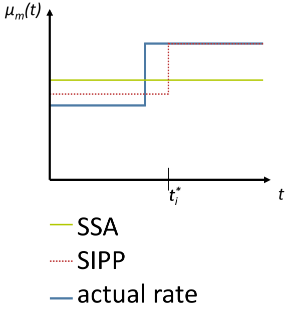
```
function IterateOnALevel( fixedValues, fixedBuffer, bufferValue )
    fixedValues_fixedBuffer ← bufferValue
    if noOfNotFixedBuffers > 1 then
        DetmineConditionalLowerBounds(fixedValues)
        DetmineConditionalUpperBounds(fixedValues)
        nextFixedBuffer ← DetmineNextFixedBuffer(fixed)
        nextFixedBufferValue ← ⌊(UB(fixedBuffer) − LB(fixedBuffer))/2⌋
IterateOnALevel(fixedValues, nextFixedBuffer, nextFixedBufferValue)
    else
        BiSecSearchSmallestFeasibleAlloc(NotFixedBuffer)
    end if
    if LB(fixedBuffer) < bufferValue then
        IterateOnALevel(fixedValues, fixedBuffer, smallerbufferValue)
    end if
    if UB(fixedBuffer) > bufferValue then
        IterateOnALevel(fixedValues, fixedBuffer, largerBufferValue)
    end if
end function
```

- The algorithm terminates (worst case: complete enumeration)
- If it exists, the algorithm returns the optimal solution

Agenda

Basic heuristic solutions: Relax time-dependency

Parameter generation



Solving the problem

Simple **S**tationary **A**pproximation (SSA)

- Solving a single steady-state problem
- Constant allocation

Stationary **I**ndependent **P**eriod by **P**eriod Approximation (SIPP)

- Solving $I + 1$ steady-state problems
- Time-dependent allocation

Constant allocation (CONST)

- Solving a Proactive Kanban Card Setting Problem with $I = 1$
- Constant allocation

Agenda

Comparison of solution approaches

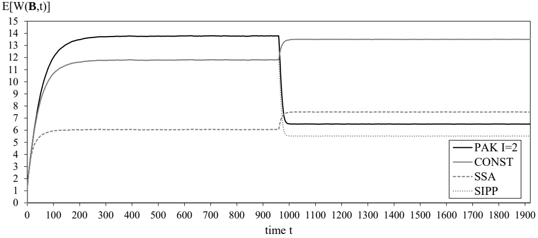
Parameters:

- $M = 1,\ b_M^{max} = 20$
- $I = 2,\ t_2^* = 960$
- Poisson demand
- $\lambda = 0.5,\ \alpha^* = 0.98$
- Exponentially distributed processing times
- $\mu_1(t) = 2/3,\ t \in [0; 960)$, $\mu_1(t) = 1,\ t \in [960; 1920)$

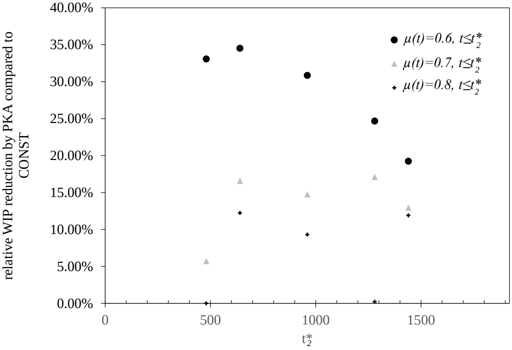
- Evaluation in sectioning algorithm: 50,0000 replications

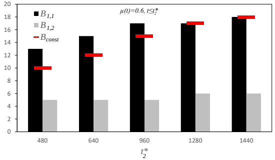
Results:

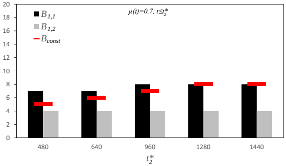
Approach	$B_{1,0}$	$B_{1,1}$	$SL^{\alpha}(\mathbf{B})$	$E[W(\mathbf{B})]$
SSA	7	7	0.941	6.715
SIPP	13	5	0.973	8.440
CONST	12	12	0.980	12.395
PKA	15	6	0.980	9.850

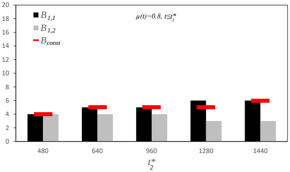


Impact of magnitude in the rate change and its timing









- The greater the change in the rates, the greater the savings compared to constant allocations

$M=1,\ b_M^{max}=20, I=2,\ t_1^*=960,\ \lambda=0.5,\ \alpha^*=0.95,\ \mu_1(t)=1,\ \ t\in[t_2^*;1920)$

Heuristic solutions for multi-stage systems

Parameters:

- $M = 2$; $b_m^{max} = 20$ $\alpha^* = 0.95$
- $I = 1$, $t_1^* = 960$

t	$\mu_1(t)$	$cv_1^2(t)$	$\mu_2(t)$	$cv_2^2(t)$	$\mu_3(t)$	$cv_3^2(t)$	$\lambda(t)$
$[0, 960)$	2/3	1	1	1	1	1	0.5
$[960, 1920]$	1	0.5	1	1	1	1	0.5

Results:

M	Approach	$B_{1,1}$	$B_{1,2}$	$B_{2,1}$	$B_{2,2}$	$B_{3,1}$	$B_{3,2}$	$SL^{\gamma}(\mathbf{B})$	$E[W(\mathbf{B})]$
2	CONST	4	4	6	6	-	-	0.950	10.15
	PKA	3	1	9	5	-	-	0.951	8.984 (-11.4%)
3	CONST	3	3	2	2	8	8	0.952	13.365
	PKA	4	1	2	1	8	7	0.951	11.812 (-11.6%)

Performance of the seactioning algorithm

Computational effort:

No. of decision variables	2	3	4	6
Average no. of evaluated alloc.	18.88	165	508	44,444
Average % of all alloc. evaluated	4.28	1.78	0.26	0.05

Solution quality:

- Optimality of allocations is verified by complete enumeration for problems with $(I+1) \cdot M \leq 4$

Conclusions and future research

- New approach for Kanban allocation in stochastic and time-dependent flow lines
- Sectioning based simulation-optimization approach

Managerial insights:

- Time-dependent allocation results in improvements compared to constant allocations
- Processing rate changes may require to change the structure of the buffer allocation

Future research:

- Development of advanced heuristics
- Problem extensions
  - Service level goals for subperiods of the planning horizon
  - Extensions to other control policies such as CONWIP

Thank you for your attention

References I

Ammar, M. and S. Gershwin (1980, dec). Equivalence relations in queueing models of manufacturing networks. In *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, pp. 715–721. IEEE.

Anantharam, V. and P. Tsoucas (1990). Stochastic concavity of throughput in series of queues with finite buffers. *Advances in Applied Probability 22*(3), 761–763.

Berkley, B. J. (1991). Tandem queues and kanban-controlled lines. *International Journal of Production Research 29*(10), 2057–2081.

Chao, X. and M. Pinedo (1992, dec). On reversibility of tandem queues with blocking. *Naval Research Logistics 39*(7), 957–974.

Chao, X., M. Pinedo, and K. Sigman (1989, apr). On the Interchangeability and Stochastic Ordering of Exponential Queues in Tandem with Blocking. *Probability in the Engineering and Informational Sciences 3*(02), 223.

Cheng, D. W. (1992). Second order properties in a tandem queue with general blocking. *Operations Research Letters 12*(3), 139–144.

Cheng, D. W. (1995, may). Line Reversibility of Tandem Queues with General Blocking. *Management Science 41*(5), 864–873.

Duri, C., Y. Frein, and M. Di Mascolo (2000). Comparison among three pull control policies: Kanban, base stock, and Generalized Kanban. *Annals of Operations Research 93*(1), 41–69.

Glasserman, P. and D. D. Yao (1996). Structured buffer-allocation problems. *Discrete Event Dynamic Systems: Theory and Applications 6*(1), 9–41.

Jaikumar, R. and R. E. Bohn (1992). A dynamic approach to operations management: An alternative to static optimization. *International Journal of Production Economics 27*(3), 265–282.

Lev, B. and D. I. Toof (1980, sep). The role of internal storage capacity in fixed cycle production systems. *Naval Research Logistics Quarterly 27*(3), 477–487.

Meester, L. E. and J. G. Shanthikumar (1990). Concavity of the Throughput of Tandem Queueing Systems with Finite Buffer Storage Space. *Advances in Applied Probability 22*(3), 764–767.

References II

Schwarz, J. A. (2015). *Analysis of buffer allocations in time-dependent and stochastic flow lines*. Ph. D. thesis, University of Mannheim.

So, K. C. (1997). Optimal buffer allocation strategy for minimizing work-in-process inventory in unpaced production lines. *IIE Transactions 29*(1), 81–88.

Soyster, A. L. and D. I. Toof (1976). Some comparative & design aspects of fixed cycle production systems. *Naval Research Logistics Quarterly 23*(3), 437–454.

Takahashi, K. and N. Nakamura (2002). Decentralized reactive Kanban system. *European Journal of Operational Research 139*(2), 262–276.

Tayur, S. R. (1992). Properties of serial kanban systems. *Queueing Systems 12*(3), 297–318.

Tayur, S. R. (1993). Structural Properties and a Heuristic for Kanban-Controlled Serial Lines. *Management Science 39*(11), 1347–1368.

Terwiesch, C. and R. E. Bohn (2001). Learning and process improvement during production ramp-up. *International Journal of Production Economics 70*(1), 1–19.