Approaches for machine selection and buffer allocation in stochastic flow lines

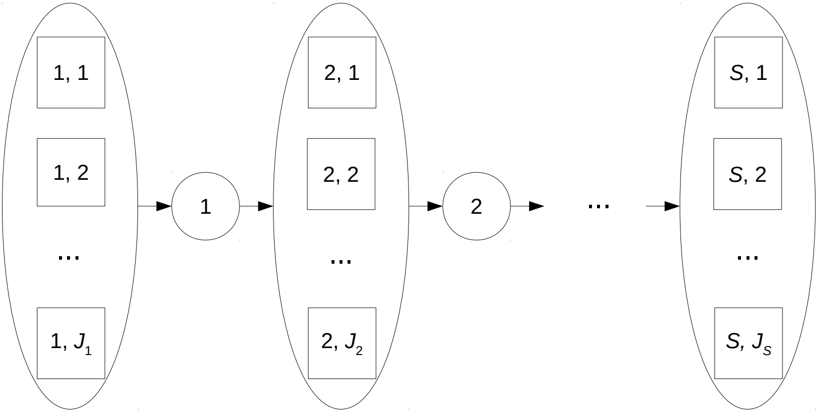
Stefan Helber

Leibniz Universität Hannover

SMMSO 2017, Lecce

1. Problem and motivation
2. Conceptual optimization model
3. Brute force MIP model
4. LocalSolver plus flow line decomposition
5. Elements of a Branch & Bound approach
6. No numerical results, but ...

Stochastic flow lines with alternative machines



Problem

- Serial production process
- Single product with target production rate $PR^{\min}$
- Decision I: Selection of one the alternative machines $j = 1, ..., J_s$ with stochastic processing times $T_{s,j}$ for each station s
- Decision II: Capacity b_k of the buffer behind station $s = 1, ..., S - 1$

- Objective: Minimize required capital budget for machines and buffers

Conceptual model

$$\text{Min } = \sum_{s=1}^{S}\sum_{j=1}^{J_s} cr^M_{s,j}\cdot v_{s,j} + \sum_{s=1}^{S-1} cr^B_s \cdot x_s$$

$$\sum_{j=1}^{J_s} v_{s,j} = 1, \qquad s = 1, ..., S$$

$$PR(\underline{v}, \underline{x}) \geq PR^{\min}$$

$$v_{s,j} \in \{0,1\}, \qquad s = 1, ..., S; j = 1, ..., J_s$$

$$x_s \in \{0,1,2,3,.....\}, \qquad s = 1, ..., S-1$$

Conceptual model

$$\text{Min } = \sum_{s=1}^{S}\sum_{j=1}^{J_s} cr_{s,j}^{M} \cdot v_{s,j} + \sum_{s=1}^{S-1} cr_s^{B} \cdot x_s$$

$$\sum_{j=1}^{J_s} v_{s,j} = 1, \qquad s = 1, ..., S$$

$$PR(\underline{v}, \underline{x}) \geq PR^{\min}$$

$$v_{s,j} \in \{0, 1\}, \qquad s = 1, ..., S; j = 1, ..., J_s$$

$$x_s \in \{0, 1, 2, 3, ...\}, \qquad s = 1, ..., S-1$$

Difficulties: $PR(\underline{v}, \underline{x})$ non-linear, no closed-form expression, integrality constraints on decision variables

Solving the model

Dimensions:

- Performance evaluation methodology
- Optimization methodology

Solving the model

Dimensions:

- Performance evaluation methodology
- Optimization methodology

Approaches:

- Simulation optimization in an LP
- LocalSolver plus decomposition
- Branch & Bound plus decomposition

Brute force MIP model I (court. Sophie Weiss)

Main features:

Brute force MIP model I (court. Sophie Weiss)

Main features:

- Sampling of large number of processing times d_{sjw} for workpieces w at stage s for machine alternative j

Brute force MIP model I (court. Sophie Weiss)

Main features:

- Sampling of large number of processing times d_{sjw} for workpieces w at stage s for machine alternative j
- Propagation of starting and finishing times XS_{sw} and XF_{sw} via linear constraints

Brute force MIP model I (court. Sophie Weiss)

Main features:

- Sampling of large number of processing times d_{sjw} for workpieces w at stage s for machine alternative j
- Propagation of starting and finishing times XS_{sw} and XF_{sw} via linear constraints
- W_0 work piece for warm-up phase of the line

Brute force MIP model I (court. Sophie Weiss)

Main features:

- Sampling of large number of processing times d_{sjw} for workpieces w at stage s for machine alternative j
- Propagation of starting and finishing times XS_{sw} and XF_{sw} via linear constraints
- W_0 work piece for warm-up phase of the line
- Very general and flexible, very time-consuming

Brute force MIP model I (court. Sophie Weiss)

Main features:

- Sampling of large number of processing times d_{sjw} for workpieces w at stage s for machine alternative j
- Propagation of starting and finishing times XS_{sw} and XF_{sw} via linear constraints
- W_0 work piece for warm-up phase of the line
- Very general and flexible, very time-consuming
- Limited usefulness, computation of reference values

$$\text{Min } = \sum_{s=1}^{S}\sum_{j=1}^{J_s} cr_{s,j}^{M} \cdot V_{s,j} + \sum_{s=1}^{S-1} cr_s^B \cdot X_s \tag{1}$$

$$\sum_{j=1}^{J_s} V_{s,j} = 1, \qquad \forall s \tag{2}$$

$$XS_{s,w} + \sum_{j=1}^{J_s} d_{s,j,w} \cdot V_{s,j} \leq XF_{s,w}, \qquad \forall s, \forall w \tag{3}$$

$$XF_{s,w} \leq XS_{s+1,w}, \qquad \forall s \leq S-1, \forall w \tag{4}$$

$$XF_{s,w} \leq XS_{s,w+1}, \qquad \forall s, \forall w \leq W-1 \tag{5}$$

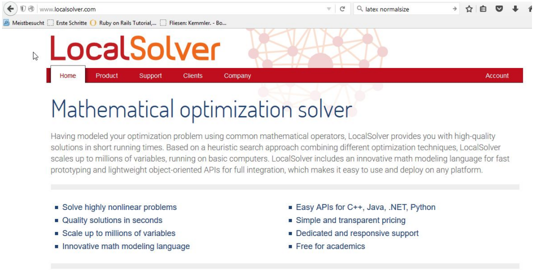
$$XF_{S,W} - XF_{S,W_0} \leq \frac{W - W_0}{PR^{\min}} \tag{6}$$

$$XS_{s+1,w} - XF_{s,w+b} \leq M \cdot (1 - Y_{s,b}), \qquad \forall s \leq S-1, \forall b, \forall w \leq W-b \tag{7}$$

$$\sum_{b=0}^{B_s} Y_{s,b} = 1, \qquad \forall s \leq S-1 \tag{8}$$

$$X_s = \sum_{b=0}^{B_s} b \cdot Y_{s,b}, \qquad \forall s \leq S-1 \tag{9}$$

LocalSolver-Approach



- Commercial software, academic licenses
- Heuristic search algorithms
- Combinatorial problems, discrete decision variables
- Specific math-modeling language
- APIs for C++, Python etc.
- New cool feature: Native functions !!!

Code Example

Constraint

$$\sum_{j=1}^{J_s} V_{s,j} = 1, \qquad \forall s$$

Code Example

Constraint

$$\sum_{j=1}^{J_s} V_{s,j} = 1, \qquad \forall s$$

Use of the C++ API:

```
// Exactly one machine is selected per station

for (int i = 0; i < nbStations; i++) {
LSExpression nbMachinesSelected = MyModel.sum();
for (int j = 0; j < nbCandidateMachines[i]; j++){
nbMachinesSelected += X[i][j];
}
MyModel.constraint(nbMachinesSelected == 1);
}
```

Flow line decomposition in native function

Flow line decomposition:

- Each station s characterized by $\mathrm{E}[T_s]$ and c_s^2

Flow line decomposition in native function

Flow line decomposition:

- Each station s characterized by $\mathrm{E}[T_s]$ and c_s^2
- Decomposition into system of GI/G/1/N stopped arrival queues (Buzacott, Liu, Shanthikumar, Manitz)

Flow line decomposition in native function

Flow line decomposition:

- Each station s characterized by $\mathrm{E}[T_s]$ and c_s^2
- Decomposition into system of GI/G/1/N stopped arrival queues (Buzacott, Liu, Shanthikumar, Manitz)
- Iterative algorithm determines production rate & buffer levels

Flow line decomposition in native function

Flow line decomposition:

- Each station s characterized by $\mathrm{E}[T_s]$ and c_s^2
- Decomposition into system of GI/G/1/N stopped arrival queues (Buzacott, Liu, Shanthikumar, Manitz)
- Iterative algorithm determines production rate & buffer levels
- Fast and accurate

Flow line decomposition in native function

Flow line decomposition:

- Each station s characterized by $\mathrm{E}[T_s]$ and c_s^2
- Decomposition into system of GI/G/1/N stopped arrival queues (Buzacott, Liu, Shanthikumar, Manitz)
- Iterative algorithm determines production rate & buffer levels
- Fast and accurate
- Implemented in C++ as a LocalSolver native function

Flow line decomposition in native function

Flow line decomposition:

- Each station s characterized by $\mathrm{E}[T_s]$ and c_s^2
- Decomposition into system of GI/G/1/N stopped arrival queues (Buzacott, Liu, Shanthikumar, Manitz)
- Iterative algorithm determines production rate & buffer levels
- Fast and accurate
- Implemented in C++ as a LocalSolver native function
- Called by LocalSolver via API during during each LocalSolver search move

Branch&Bound: Relaxation

Relaxation of integrality constraints

- Stations mixed by fractions $0 \leq \overline{v}_{s,j} \leq 1$ of machines

Branch&Bound: Relaxation

Relaxation of integrality constraints

- Stations mixed by fractions $0 \leq \overline{v}_{s,j} \leq 1$ of machines
- Buffer sizes $\overline{x}_s$ real-valued

Branch&Bound: Relaxation

Relaxation of integrality constraints

- Stations mixed by fractions $0 \leq \overline{v}_{s,j} \leq 1$ of machines
- Buffer sizes $\overline{x}_s$ real-valued
- Evaluation via GI/G/1/K queueing model decomposition

Branch&Bound: Relaxation

Relaxation of integrality constraints

- Stations mixed by fractions $0 \leq \overline{v}_{s,j} \leq 1$ of machines
- Buffer sizes $\overline{x}_s$ real-valued
- Evaluation via GI/G/1/K queueing model decomposition

$\mathrm{E}[T_s]$ and $\mathrm{Var}[T_s]$ of stochastic virtual mixed processing times T_s

Branch&Bound: Relaxation

Relaxation of integrality constraints

- Stations mixed by fractions $0 \leq \overline{v}_{s,j} \leq 1$ of machines
- Buffer sizes $\overline{x}_s$ real-valued
- Evaluation via GI/G/1/K queueing model decomposition

$E[T_s]$ and $Var[T_s]$ of stochastic virtual mixed processing times T_s

$$T_s = \sum_{j=1}^{J_s} \overline{v}_{s,j} \cdot T_{s,j}$$

Important assumption: perfect correlation between $T_{s,i}$ and $T_{s,j}$!!!!!

Branch & Bound: Lower bounds on required budget

Basic idea

1. Start with cheapest currently possible configuration

Branch & Bound: Lower bounds on required budget

Basic idea

1. Start with cheapest currently possible configuration
2. Determine numerical "gradient" of $PR()$ of $\overline{v}_{s,j}$, $\overline{x}_s$

Branch & Bound: Lower bounds on required budget

Basic idea

1. Start with cheapest currently possible configuration
2. Determine numerical "gradient" of $PR()$ of $\overline{v}_{s,j}$, $\overline{x}_s$
3. Phase I: Increase budget until $PR \geq PR^{min}$

Branch & Bound: Lower bounds on required budget

Basic idea

1. Start with cheapest currently possible configuration
2. Determine numerical "gradient" of $PR()$ of $\overline{v}_{s,j}$, $\overline{x}_s$
3. Phase I: Increase budget until $PR \geq PR^{min}$
4. Iterate

Branch & Bound: Lower bounds on required budget

Basic idea

1. Start with cheapest currently possible configuration
2. Determine numerical "gradient" of $PR()$ of $\overline{v}_{s,j}$, $\overline{x}_s$
3. Phase I: Increase budget until $PR \geq PR^{min}$
4. Iterate
    1. Phase II: Re-distribute current budget while PR increases
    2. Phase III: Decrease budget until $PR \approx PR^{min}$
5. Terminate when budget stops to decrease for feasible solution or when PR^{min} is not reached in Phase I

Branch & Bound: Branching

Basic ideas

1. Branch on fractional values machine selection and buffer size variables $\overline{v}_{s,j}, \overline{x}_s$

Branch & Bound: Branching

Basic ideas

1. Branch on fractional values machine selection and buffer size variables $\overline{v}_{s,j}, \overline{x}_s$
2. Add constraints on lower and upper bounds on $\overline{v}_{s,j}, \overline{x}_s$

Branch & Bound: Branching

Basic ideas

1. Branch on fractional values machine selection and buffer size variables $\overline{v}_{s,j}, \overline{x}_s$
2. Add constraints on lower and upper bounds on $\overline{v}_{s,j}, \overline{x}_s$
3. Depth-first search (LIFO problem processing)

Branch & Bound: Branching

Basic ideas

1. Branch on fractional values machine selection and buffer size variables $\overline{v}_{s,j}, \overline{x}_s$
2. Add constraints on lower and upper bounds on $\overline{v}_{s,j}, \overline{x}_s$
3. Depth-first search (LIFO problem processing)

Observation: Relaxed selection variables $\overline{v}_{s,j}$ often binary, buffer variables $\overline{x}_s$ never

Difficulties:

Gradient calculations

1. Numerous constraints on the gradients

Difficulties:

Gradient calculations

1. Numerous constraints on the gradients
2. Rosen's projection method requires numerical solution of LSE

Difficulties:

Gradient calculations

1. Numerous constraints on the gradients
2. Rosen's projection method requires numerical solution of LSE
3. PR only approximated
4. Gradients only approximated

Difficulties:

Gradient calculations

1. Numerous constraints on the gradients
2. Rosen's projection method requires numerical solution of LSE
3. PR only approximated
4. Gradients only approximated

Steepest ascent method

1. PR highly non-linear, frequent gradient updates
2. Termination, numerical issues

An extremely preliminary conclusion

First impression from Branch & Bound

- Methods seems to work (in principle)
- Algorithm complex and not yet stable
- First feasible solutions can be found quickly
- Bounds seem to be strong

An extremely preliminary conclusion

First impression from Branch & Bound

- Methods seems to work (in principle)
- Algorithm complex and not yet stable
- First feasible solutions can be found quickly
- Bounds seem to be strong

Future work

- Improve stability
- Serious numerical study
- Model variants, e.g., space limitations

An extremely preliminary conclusion

First impression from Branch & Bound

- Methods seems to work (in principle)
- Algorithm complex and not yet stable
- First feasible solutions can be found quickly
- Bounds seem to be strong

Future work

- Improve stability
- Serious numerical study
- Model variants, e.g., space limitations

Thank you!!