A Priority-Rule Based Method for Scheduling in Chemical Batch Production

Christoph Schwindt
Rafael Fink, Norbert Trautmann

Clausthal University of Technology, Germany
University of Bern, Switzerland

supported by Deutsche Forschungsgemeinschaft under Grant Schw1178/1

IEEM Conference, Singapore, 2–5 Dec 2007

Outline

1. Scheduling problem
2. Priority-rule based method
    - Preprocessing
    - Iteration
3. Performance analysis
4. Conclusions

Outline

1. Scheduling problem
2. Priority-rule based method
   - Preprocessing
   - Iteration
3. Performance analysis
4. Conclusions

Scheduling problem I

Operations and states

- Set $\mathcal{O}$ of operations i (process executions) with processing times p_i
- Each operation transforms given amounts of input states into given amounts of output states
- Intermediate states may be chemically instable (perishable products)

Equipment

- Multi-purpose processing units
- Dedicated storage facilities of limited capacity for stocking states

Scheduling problem II

Constraints

- Non-overlapping execution of operations on processing units
- Sequence-dependent changeover times
- Availability of input materials and storage space for output products
- Immediate consumption of perishable products

Scheduling problem

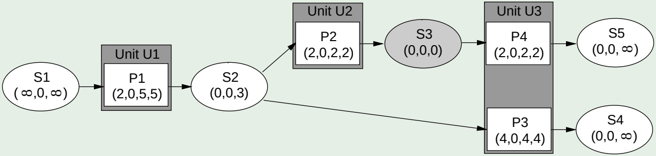
Determine feasible production schedule

- allocation of processing units to operations
- assignment of start times to operations

complying with constraints such that makespan is minimized

Scheduling problem III

Example (State-task network)



- 3 processing units
- 4 processes, no changeover times
- 4 storage facilities (1 finite intermediate storage)
- 5 states (1 perishable product)
- Primary requirements $\rho_{S4} = 8$, $\rho_{S5} = 2$
- Batching provides 6 operations ($2 \times P1$, $1 \times P2$, $2 \times P3$, $1 \times P4$)

Outline

1. Scheduling problem
2. Priority-rule based method
   - Preprocessing
   - Iteration
3. Performance analysis
4. Conclusions

Preprocessing I

Preprocessing principle

- Generate time lags δ_{ij} between starts of operations $i, j \in \mathcal{O}$ that are satisfied by some optimal schedule
- Translate time lags into operation-on-node network

(1) Operations of same process

- Arrange operations in arbitrary order
  - Operations have to be executed on same processing unit:
    $\triangleright$ $\delta_{ij} = p_i$ for any two consecutive operations i, j
  - There are alternative processing units:
    $\triangleright$ $\delta_{ij} = 0$ for any two consecutive operations i, j

Preprocessing II

(2) Intermediates produced by only one process

- Identify minimum time lags $\delta_{ij} = p_i$ between producing operations i and consuming operations j necessary to avoid shortages of input materials

(3) Intermediates consumed by only one process

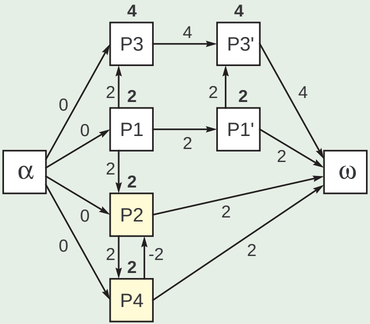
- Identify maximum time lags $-\delta_{ji} = p_i$ between producing operations i and consuming operations j necessary to avoid capacity overflows

(4) Perishable intermediates

- Associate fictitious storage facility of capacity zero
- Apply (3) in case of unique consuming operation

Preprocessing III

Example (Operation-on-node network)



- α, ω: production start, production end
- Longest path length d_{ij}: transitive time lag between starts of i and j
- Strong components $\mathcal{O}' \subseteq \mathcal{O}$ of network: any two nodes $i, j \in \mathcal{O}'$ mutually linked by transitive time lags

Iteration I

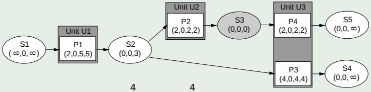
Priority-rule based method

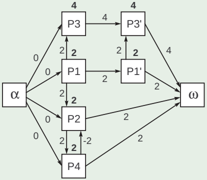
initialize $\mathcal{C} := \emptyset$ and put latest start times $LS_i := \infty$ for all $i \in \mathcal{O}$;
while $\mathcal{C} \neq \mathcal{O}$ **do**
  determine set $\mathcal{E}$ of eligible operations $i \in \mathcal{O} \setminus \mathcal{C}$;
  compute priority value $v(i)$ for all $i \in \mathcal{E}$;
  remove operation j with highest priority value $v(j)$ from set $\mathcal{E}$;
  compute earliest start time ES_j of j; (∗ disregard storage capacities ∗)
  if $ES_j \leq LS_j$ **then**
    schedule j at time ES_j and put $\mathcal{C} := \mathcal{C} \cup \{j\}$;
    update latest start times LS_i of operations $i \in \mathcal{O} \setminus \mathcal{C}$;
  else
    perform an unscheduling step;
  end if
end while

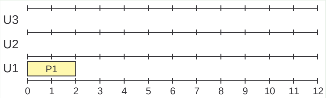
Iteration II

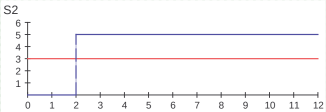
Example (Priority-rule based method)

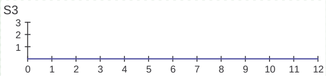
- $\mathcal{E} = \{P1\}$, $j = P1$
- $ES_{P1} = 0$, $LS_{P1} = \infty$
- Schedule $j = P1$ at time 0









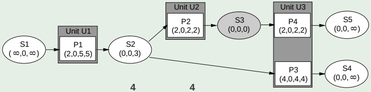


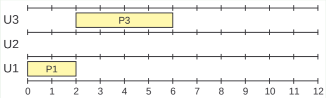
- $LS_{P2} = LS_{P3} = LS_{P3'} = 2$

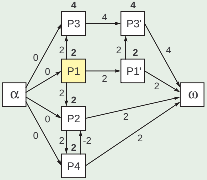
Iteration II

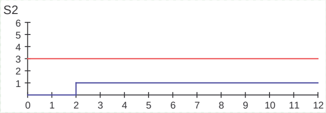
Example (Priority-rule based method)

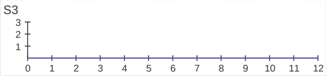
- $\mathcal{E} = \{P1', P2, P3\}$, $j = P3$
- $LS_{P3} = 2$, $ES_{P3} = 2$
- Schedule $j = P3$ at time 2









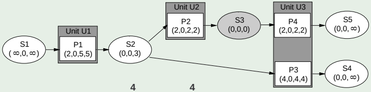


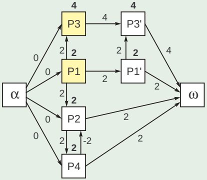
- $LS_{P2} = LS_{P3'} = \infty$

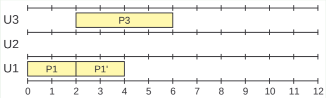
Iteration II

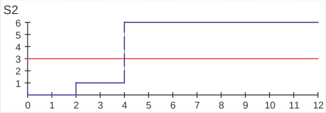
Example (Priority-rule based method)

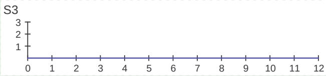
- $\mathcal{E} = \{P1'\}$, $j = P1'$
- $LS_{P1'} = \infty$, $ES_{P1'} = 2$
- Schedule $j = P1'$ at time 2









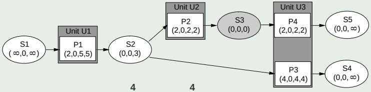


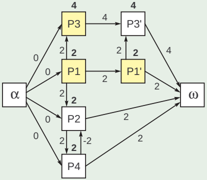
- $LS_{P2} = LS_{P3'} = 4$

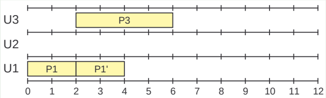
Iteration II

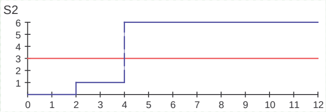
Example (Priority-rule based method)

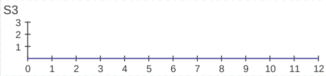
- $\mathcal{E} = \{P2, P3'\}$, $j = P3'$
- $LS_{P3'} = 4$, $ES_{P3'} = 6$
- $P3'$ cannot be scheduled







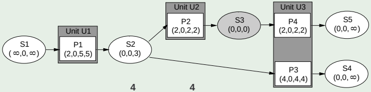


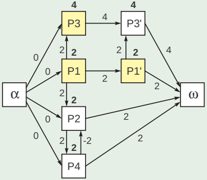


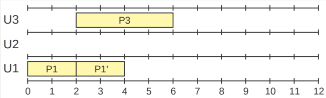
Iteration II

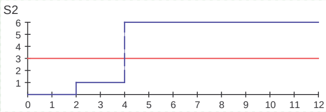
Example (Priority-rule based method)

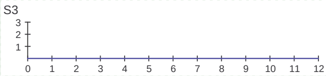
- Call unscheduling
- Origin of conflict: $i = P1'$
- Add release date $r_{P1'} = 4$









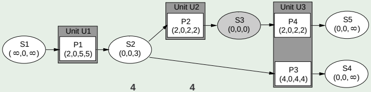


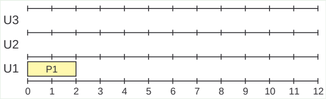
- put $\mathcal{C} := \emptyset$

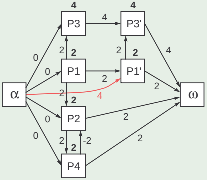
Iteration II

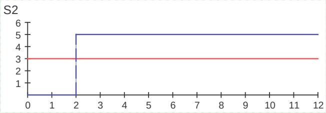
Example (Priority-rule based method)

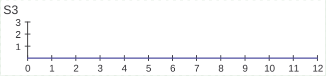
- $\mathcal{E} = \{P1\}$, $j = P1$
- $ES_{P1} = 0$, $LS_{P1} = \infty$
- Schedule $j = P1$ at time 0









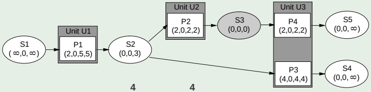


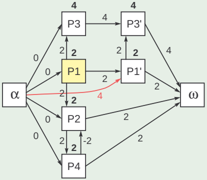
- $LS_{P2} = LS_{P3} = LS_{P3'} = 2$

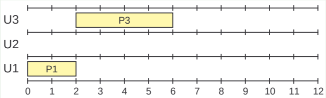
Iteration II

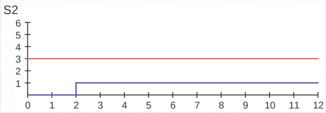
Example (Priority-rule based method)

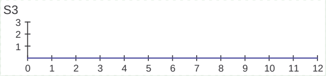
- $\mathcal{E} = \{P1', P2, P3\}$, $j = P3$
- $LS_{P3} = 2$, $ES_{P3} = 2$
- Schedule $j = P3$ at time 2









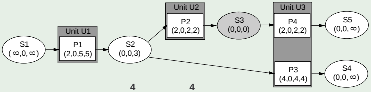


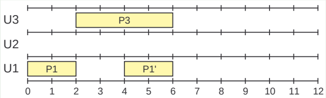
- $LS_{P2} = LS_{P3'} = \infty$

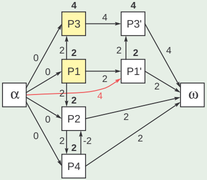
Iteration II

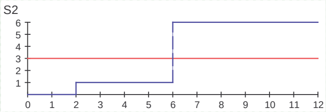
Example (Priority-rule based method)

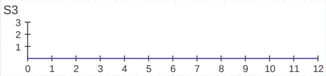
- $\mathcal{E} = \{P1'\}$, $j = P1'$
- $LS_{P1'} = \infty$, $ES_{P1'} = 4$
- Schedule $j = P1'$ at time 4









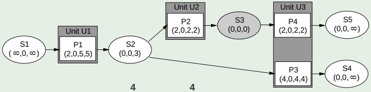


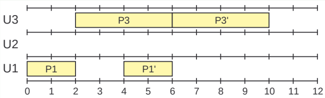
- $LS_{P2} = LS_{P3'} = 6$

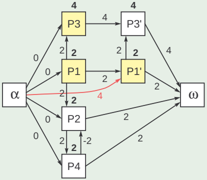
Iteration II

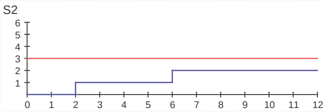
Example (Priority-rule based method)

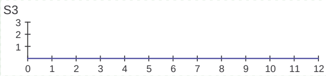
- $\mathcal{E} = \{P2, P3'\}$, $j = P3'$
- $LS_{P3'} = 6$, $ES_{P3'} = 6$
- Schedule $j = P3'$ at time 6









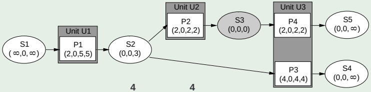


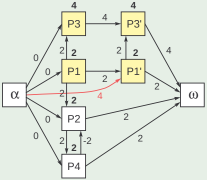
- $LS_{P2} = \infty$

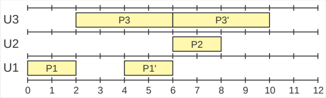
Iteration II

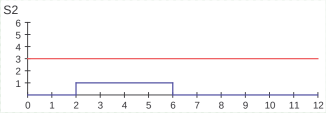
Example (Priority-rule based method)

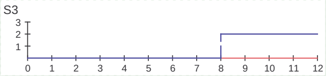
- $\mathcal{E} = \{P2\}$, $j = P2$
- $LS_{P2} = \infty$, $ES_{P2} = 6$
- Schedule $j = P2$ at time 6









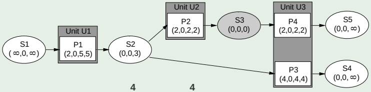


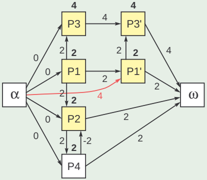
- $LS_{P4} = 8$

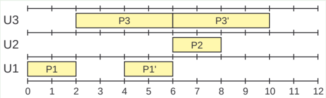
Iteration II

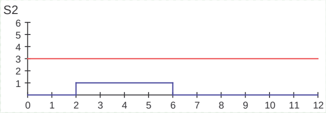
Example (Priority-rule based method)

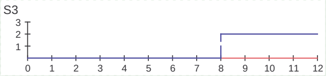
- $\mathcal{E} = \{P4\}$, $j = P4$
- $LS_{P4} = 8$, $ES_{P4} = 10$
- $P4$ cannot be scheduled







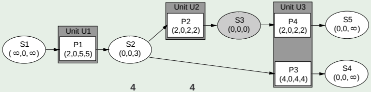


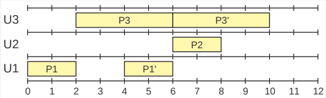


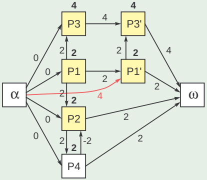
Iteration II

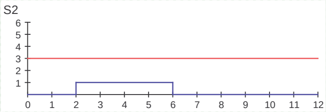
Example (Priority-rule based method)

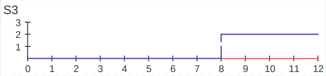
- Call unscheduling
- Origin of conflict: $i = P2$
- Add release date $r_{P2} = 8$









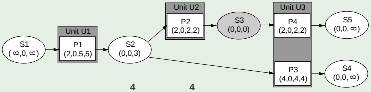


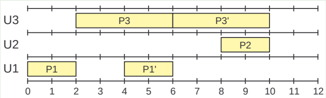
- put $\mathcal{C} := \{P1, P1', P3, P3'\}$

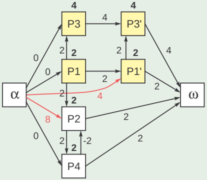
Iteration II

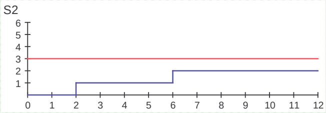
Example (Priority-rule based method)

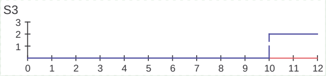
- $\mathcal{E} = \{P2\}$, $j = P2$
- $LS_{P2} = \infty$, $ES_{P2} = 8$
- Schedule $j = P2$ at time 8









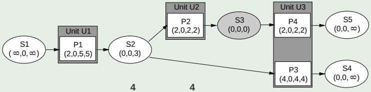


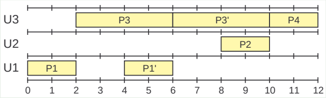
- $LS_{P4} = 10$

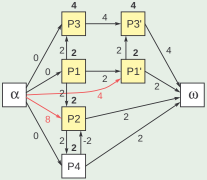
Iteration II

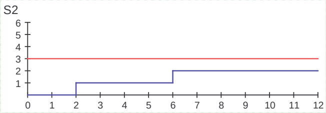
Example (Priority-rule based method)

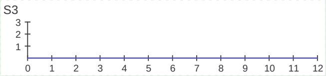
- $\mathcal{E} = \{P4\}$, $j = P4$
- $LS_{P4} = 10$, $ES_{P4} = 10$
- Schedule $j = P2$ at time 10











- $\mathcal{C} = \mathcal{O}$: stop

Outline

1. Scheduling problem
2. Priority-rule based method
   - Preprocessing
   - Iteration
3. Performance analysis
4. Conclusions

Test bed

Test set

- 22 instances generated by Blömer and Günther (2000)
- Operations obtained by solving MILP batching models

Implementation and test conditions

- Scheduling method implemented as randomized multistart procedure
- Tests performed on 2.08 GHz PC with 1 GB RAM
- Run time limit of 23 sec

Benchmark algorithms

- Time grid heuristic based on monolithic MILP model [BloGue00]
- Truncated branch-and-bound algorithm [NeuSchTra02]
- Two-phase priority-rule based method [SchTra04]

Computational results

Table: Results for instances of Blömer and Günther

Instance	1	2	3	4	5	6	7	8	9	10	11
[BloGue00]	36	42	42	48	44	48	52	48	54	60	68
[NeuSchTra02]	36	38	38	39	41	43	38	39	53	50	66
[SchTra04]	39	47	48	42	41	49	44	43	49	54	64
This paper	36	42	42	39	39	47	40	40	46	49	56

Instance	12	13	14	15	16	17	18	19	20	21	22
[BloGue00]	60	64	66	148	124	112	124	208	184	184	214
[NeuSchTra02]	52	50	57	114	80	91	91	135	100	112	134
[SchTra04]	48	57	61	100	80	92	90	159	97	124	114
This paper	45	50	53	80	66	72	70	98	78	82	83

Outline

1 Scheduling problem

2 Priority-rule based method
- Preprocessing
- Iteration

3 Performance analysis

4 Conclusions

Conclusions

Summary

- New priority-rule based method for batch production scheduling in the process industries
  - Operations-on-node network
  - Capacity-driven latest start times
  - Unscheduling technique
- Very good schedules obtained within short amount of CPU time

Current research

- Expansion of method to case of uncertain processing times, uncertain yields, and unit breakdowns
- Process scheduling of multi-product continuous production plants

References

Blömer F, Günther HO (2000)
LP-based heuristics for scheduling of chemical batch plants.
International Journal of Production Research **38**:1029–1051

Neumann K, Schwindt C, Trautmann N (2002)
Advanced production scheduling for batch plants in process industries
OR Spectrum 24:251–279

Schwindt C, Trautmann (2004)
A priority-rule based method for batch production scheduling in the process industries.
In: Ahr D, Fahrion R, Oswald M, Reinelt G (eds) Operations Research Proceedings 2003. Springer, Berlin, pp. 111–118