Scheduling Using Timed Automata

Yasmina Abdeddaïm

November 28, 2002

Plan

- Introduction
- Deterministic Job-Shop Scheduling
- Preemptive Job-Shop Scheduling
- Task Graph Scheduling
- Scheduling Under Uncertainty

Plan

1. Introduction
    - Scheduling Problems
    - Timed Automata
2. Deterministic Job-Shop Scheduling
3. Preemptive Job-Shop Scheduling
4. Task Graph Scheduling
5. Scheduling Under Uncertainty

The Job-Shop Problem

- a set M of resources (machines)
- a Job $\quad J = (m_1, d_1), (m_2, d_2), \ldots, (m_k, d_k)$
- a Job-shop specification is a set $\mathcal{J} = \{J^1, \ldots, J^n\}$ of jobs

Example:

$$M = \{m_1, m_2\} \quad \mathcal{J} = \{J^1, J^2\}$$

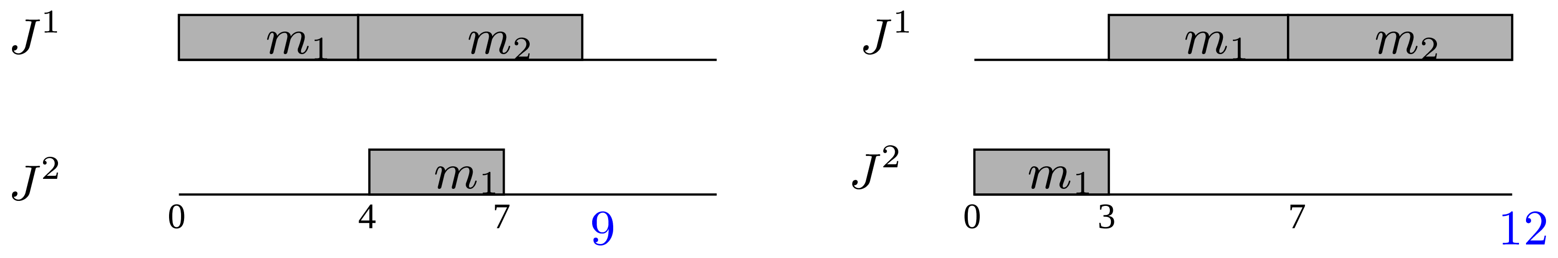
$$J^1 (m_1, 4), (m_2, 5)$$

$$J^2 (m_1, 3)$$

Feasible Schedules

Determining the execution times of the steps assuming

1. Each job is performed sequentially
2. A job can wait an arbitrary time between two steps
3. Each resource can be used only by one job at a time
4. No preemption



Sometimes it is better not to start a step although its machine is idle

The Problem

Find a feasible schedule that minimizes the total elapsed time from beginning to completion

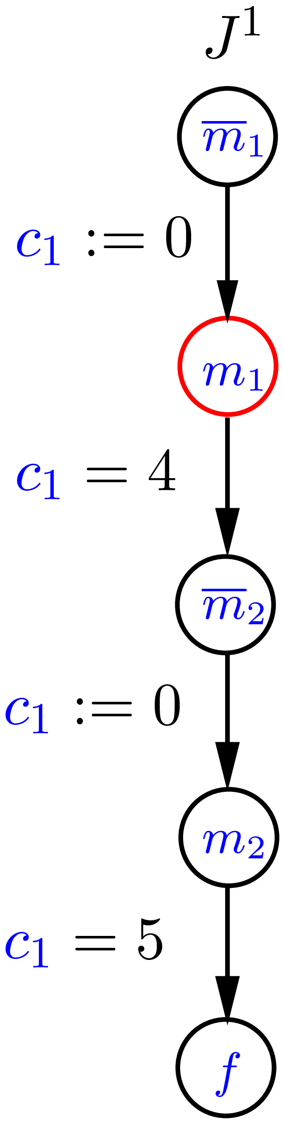
Complexity NP-complete

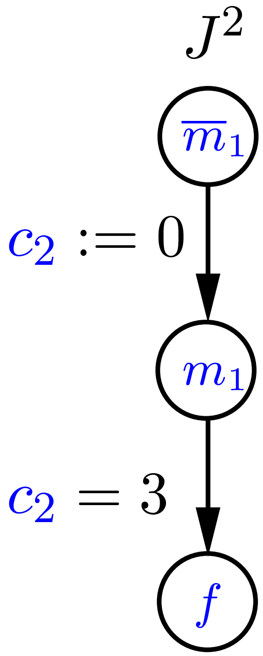
The problem has been treated by OR community using numerous methods for many years

Plan

1. Introduction
2. Deterministic Job-Shop Scheduling
   - Modeling With Timed Automata
   - Runs and Schedules
   - Shortest Path using Reachability
   - Laziness and Non-Laziness
   - Reducing the Search Space
   - Experimental Results
3. Preemptive Job-Shop Scheduling
4. Task Graph Scheduling
5. Scheduling Under Uncertainty

Modeling Jobs





$$J^1 = (m_1, 4), (m_2, 5)$$
$$J^2 = (m_1, 3)$$

Modeling Job-Shop

$$\mathcal{J} = \{J^1, \ldots, J^n\}$$

$$\{\mathcal{A}^1, \ldots, \mathcal{A}^n\} \quad \mathcal{A}^i = (Q^i, \{c_i\}, \Delta^i, s^i, f^i)$$

Mutual Exclusion Composition

$(q^1, \ldots, q^n)$ forbidden state if

q^i, q^j st $q^i = q^j = m \in M$

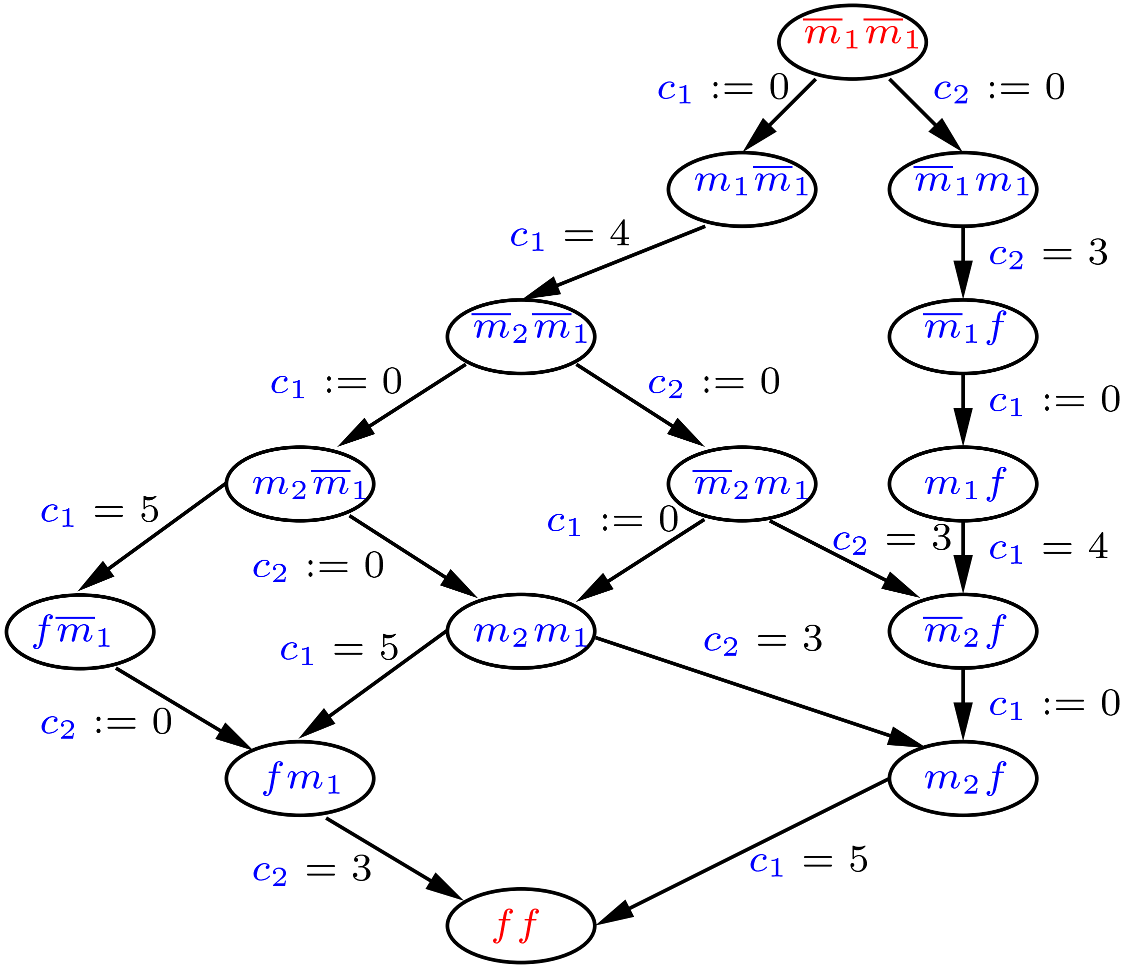
$$\mathcal{A} = (Q, \{c_1, \ldots, c_n\}, \Delta, s, f)$$

$Q = (Q_1 \times \ldots \times Q_n) -$ forbidden states

$((q^1, \ldots, q^i, \ldots, q^n), \phi, \delta, (q^1, \ldots, p^i, \ldots, q^n)) \in \Delta$

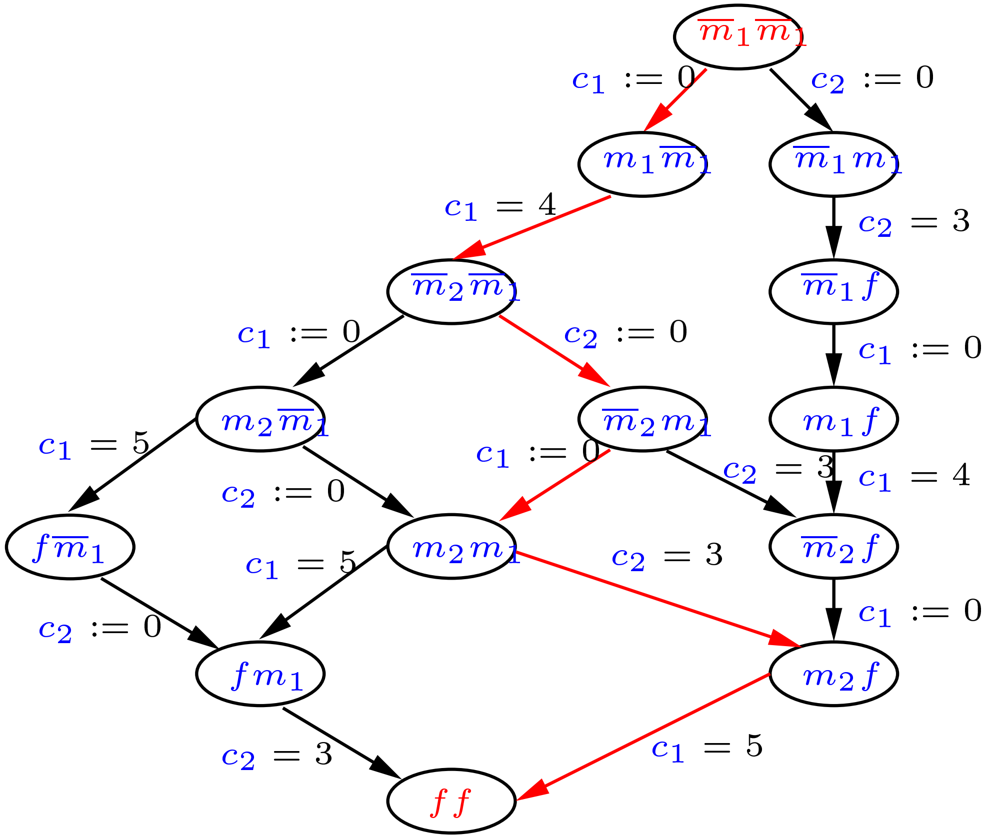
if $(q^i, \phi, \delta, p^i) \in \Delta^i$ (Interleaving semantics)

The Global Model of the Example



one (qualitative) path = infinitely many runs

Example of a run

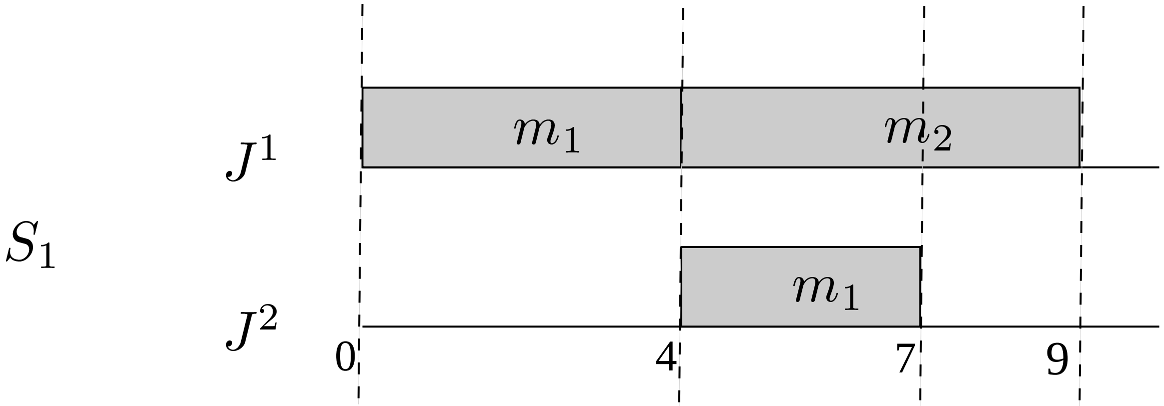


$(\overline{m}_1, \overline{m}_1, \perp, \perp) \xrightarrow{0} (m_1, \overline{m}_1, 0, \perp) \xrightarrow{4} (m_1, \overline{m}_1, 4, \perp) \xrightarrow{0} (\overline{m}_2, \overline{m}_1, \perp, \perp) \xrightarrow{0}$
$(m_2, \overline{m}_1, 0, \perp) \xrightarrow{0} (m_2, m_1, 0, 0) \xrightarrow{3} (m_2, m_1, 3, 3) \xrightarrow{0} (m_2, f, 3, \perp) \xrightarrow{2}$
$(m_2, f, 5, \perp) \xrightarrow{0} (f, f, \perp, \perp)$

Runs and Schedules

Every complete run of $\mathcal{A}$ corresponds to a feasible schedule

$$(\overline{m}_1, \overline{m}_1, \perp, \perp) \xrightarrow{0} (m_1, \overline{m}_1, 0, \perp) \xrightarrow{4} (m_1, \overline{m}_1, 4, \perp) \xrightarrow{0} (\overline{m}_2, \overline{m}_1, \perp, \perp) \xrightarrow{0} (m_2, \overline{m}_1, 0, \perp) \xrightarrow{0} (m_2, m_1, 0, 0) \xrightarrow{3} (m_2, m_1, 3, 3) \xrightarrow{0} (m_2, f, 3, \perp) \xrightarrow{2} (m_2, f, 5, \perp) \xrightarrow{0} (f, f, \perp, \perp)$$



Length of schedule = metric length of complete run

Job Shop problem $\Longleftrightarrow$ Shortest Path problem in $\mathcal{A}$

Shortest path using TA reachability

We add an auxiliary clock c_{n+1} never reset to zero

$$(\overline{m}_1, \overline{m}_1, \perp, \perp, 0) \xrightarrow{0} (m_1, \overline{m}_1, 0, \perp, 0) \xrightarrow{4} (m_1, \overline{m}_1, 4, \perp, 4) \xrightarrow{0}$$
$$(\overline{m}_2, \overline{m}_1, \perp, \perp, 4) \xrightarrow{0} (m_2, \overline{m}_1, 0, \perp, 4) \xrightarrow{0} (m_2, m_1, 0, 0, 4) \xrightarrow{3}$$
$$(m_2, m_1, 3, 3, 7) \xrightarrow{0} (m_2, f, 3, \perp, 7) \xrightarrow{2} (m_2, f, 5, \perp, 9) \xrightarrow{0} (f, f, \perp, \perp, 9)$$

The length of the optimal schedule = $\min \{t : (f, f, \perp, \perp, t) \text{ is reachable } \}$

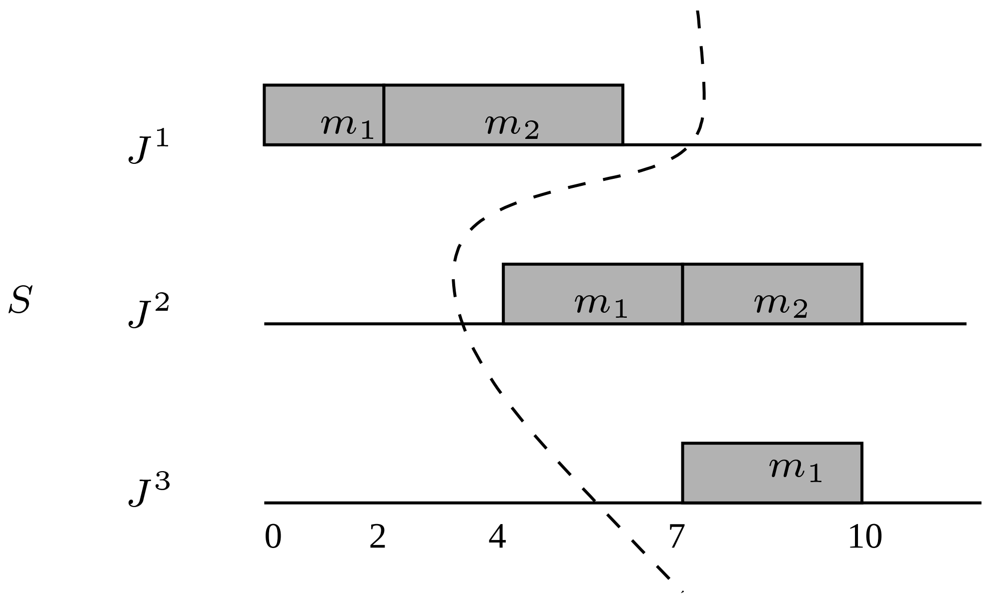
Plan

1. Introduction
2. Deterministic Job-Shop Scheduling
   - Modeling With Timed Automata
   - Runs and Schedules
   - Shortest Path using Reachability
   - Laziness and Non-Laziness
   - Reducing the Search Space
   - Expermimental Results
3. Preemptive Job-Shop Scheduling
4. Task Graph Scheduling
5. Scheduling Under Uncertainty

Getting rid of zones: Lazy and non-lazy schedule

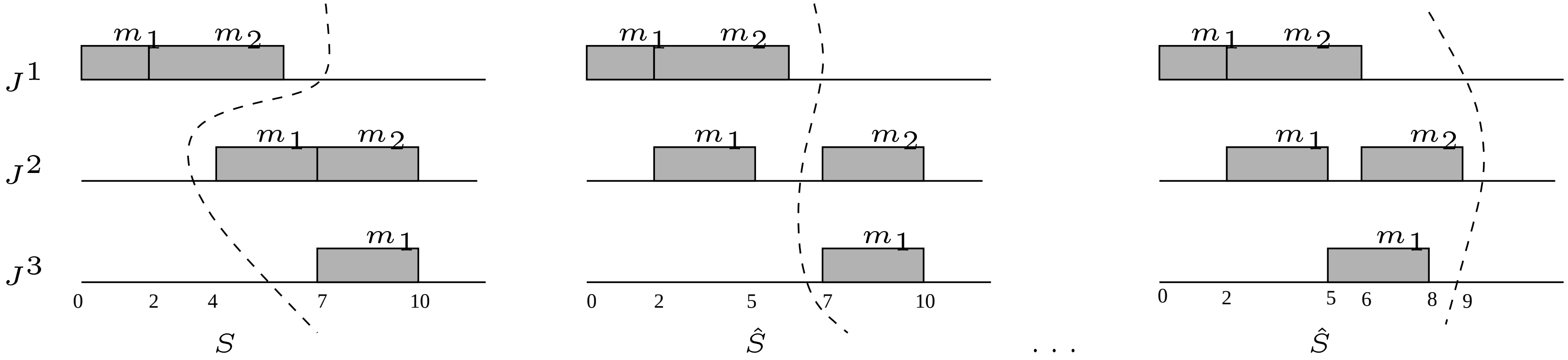
Intuition: it is useless to wait when no one else profits from your waiting

Laziness at (i, j) if job J^i and ressource m_j are idle immediately before starting step (m_j, d_j)



Non-Lazy schedules achieve the optimum

Every lazy schedule S can be transformed into a non-lazy schedule $\hat{S}$ with $|\hat{S}| \leq |S|$



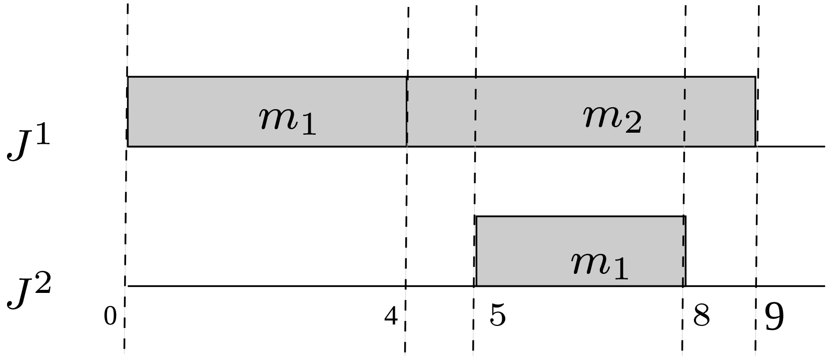
Every job-shop problem admits an optimal non-lazy solution

Immediate and non-Immediate runs

A non-Immediate run contain $(q, \mathbf{v}) \xrightarrow{t} (q, \mathbf{v} + t) \xrightarrow{0} (q', \mathbf{v}')$ while

$$\exists\ t' < t\ st\ (q, \mathbf{v}) \xrightarrow{t'} (q, \mathbf{v} + t') \xrightarrow{0} (q', \mathbf{v}'')$$

$(\overline{m}_1, \overline{m}_1, \bot, \bot) \xrightarrow{0} (m_1, \overline{m}_1, 0, \bot) \xrightarrow{4} (m_1, \overline{m}_1, 4, \bot) \xrightarrow{0} (\overline{m}_2, \overline{m}_1, \bot, \bot) \xrightarrow{0}$
$(m_2, \overline{m}_1, 0, \bot) \xrightarrow{1} (m_2, \overline{m}_1, 1, \bot) \xrightarrow{0} (m_2, m_1, 1, 0) \xrightarrow{3} (m_2, m_1, 4, 3) \xrightarrow{0}$
$(m_2, f, 4, \bot) \xrightarrow{1} (m_2, f, 5, \bot) \xrightarrow{0} (f, f, \bot, \bot)$

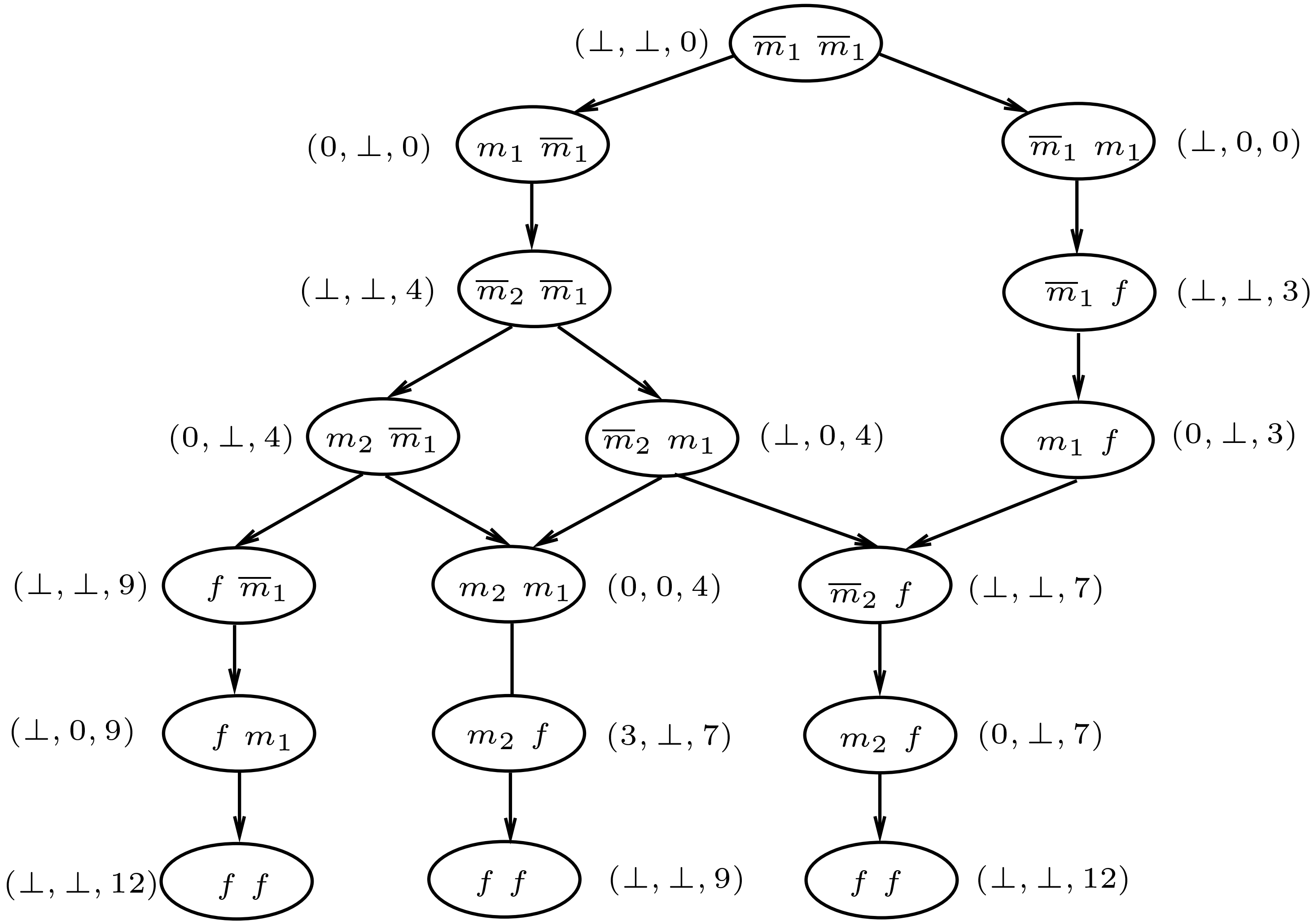


Every non-lazy schedule admits an immediate complete run

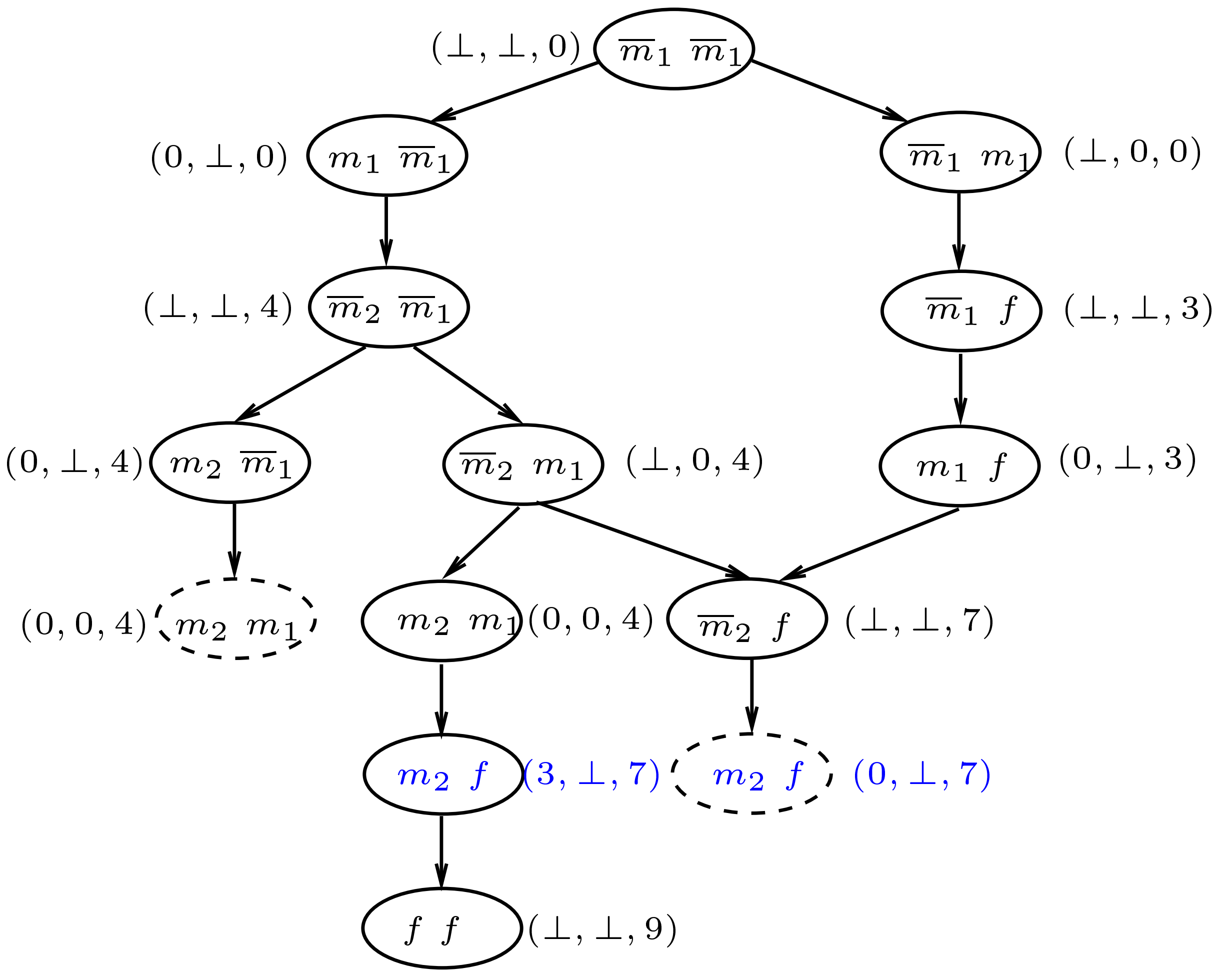
Hence it is sufficient to enumerate only immediate runs

There is one immediate run in every path

Immediate runs



Reducing the search space: Equality, Domination



$(q, \mathbf{v}, v_{n+1})$ dominate $(q, \mathbf{v}', v'_{n+1})$ if $v_{n+1} \leq v'_{n+1}$ and $\forall i \leq n\ v_i \geq v'_i$

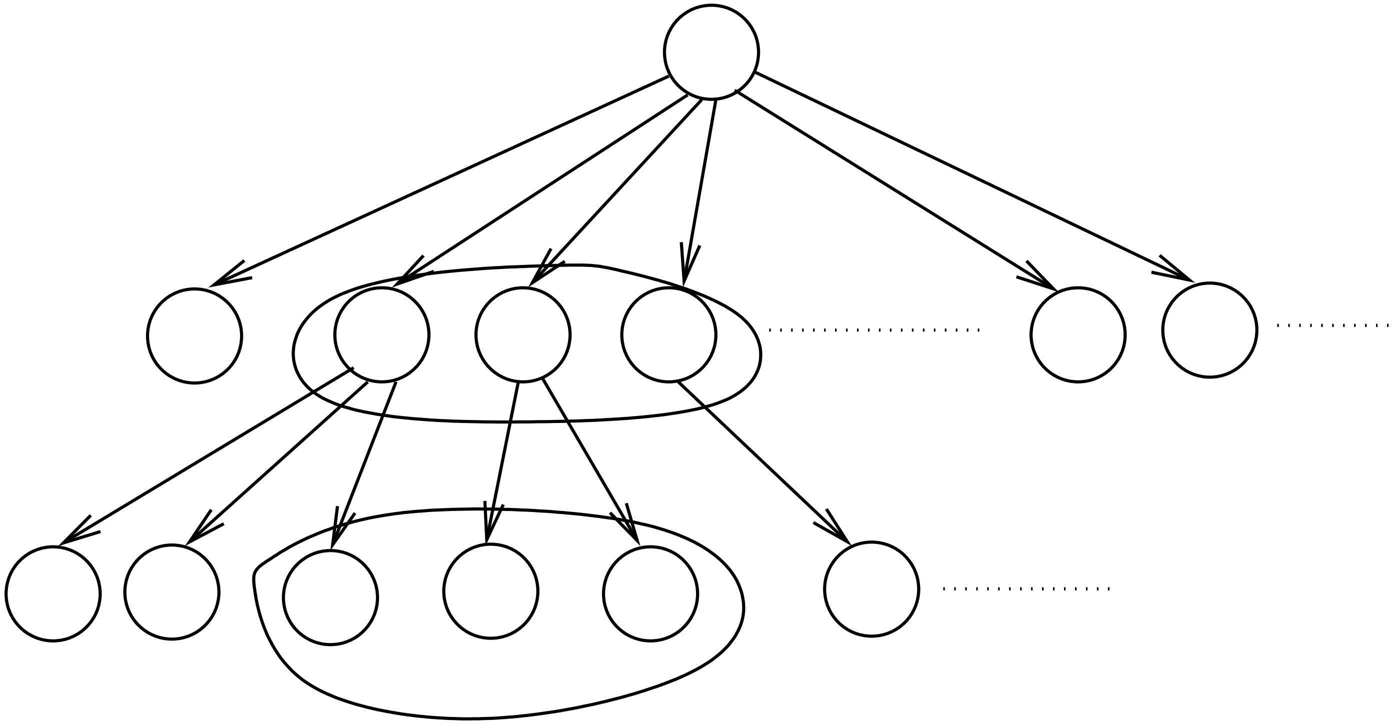
The Search Algorithm

- Explore only immediate runs
- Apply domination test
- Apply best-first search: explore the most promising nodes according to some estimation. Can stop before exploring all nodes.

With this algorithm we can find exact optimal solutions for 6×6 problems

Sub-Optimal Solution

- Breadth-first algorithm
- At each level of the search tree nodes are ordered according an evaluation function
- Explore only the w best nodes at any level



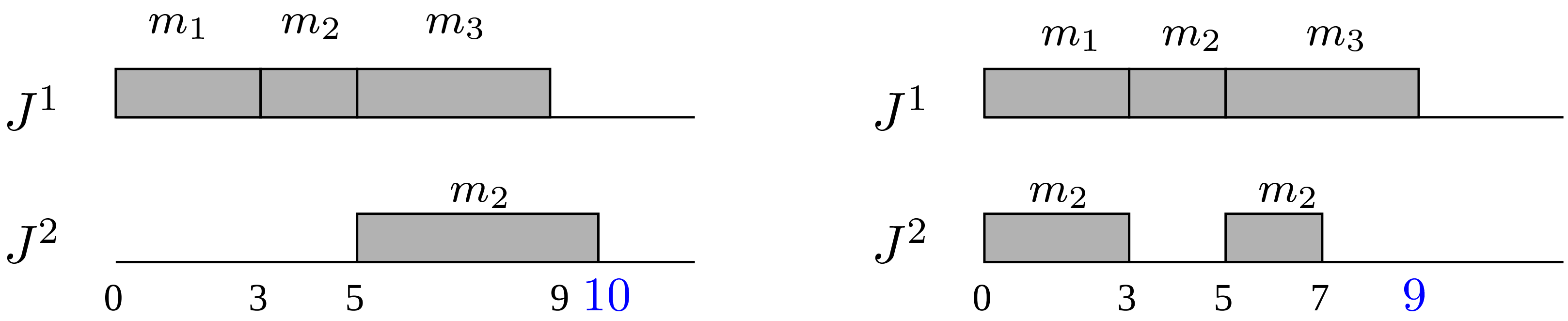
Benchmark Problems

problem			our algorithm			Opt
name	#j	#m	time	length	deviation	length
FT10	10	10	3	969	4.09 %	930
LA02	10	5	1	655	0.00 %	655
LA19	10	10	15	869	3.21 %	842
LA21	10	15	98	1091	4.03 %	1046
LA24	10	15	103	973	3.95 %	936
LA25	10	15	148	1030	5.42 %	977
LA27	10	20	300	1319	6.80 %	1235
LA29	10	20	149	1259	9.29 %	1152
LA36	15	15	188	1346	6.15 %	1268
LA37	15	15	214	1478	5.80 %	1397

Plan

1. Introduction
2. Deterministic Job-Shop Scheduling
3. Preemptive Job-Shop Scheduling
    - The Problem
    - Modeling With Stopwatch Automata
    - Runs and Schedules
    - Efficient Schedules
    - Searching for Efficient Runs
    - Experimental Results
4. Task Graph Scheduling
5. Scheduling Under Uncertainty

The Preemptive Job-Shop problem

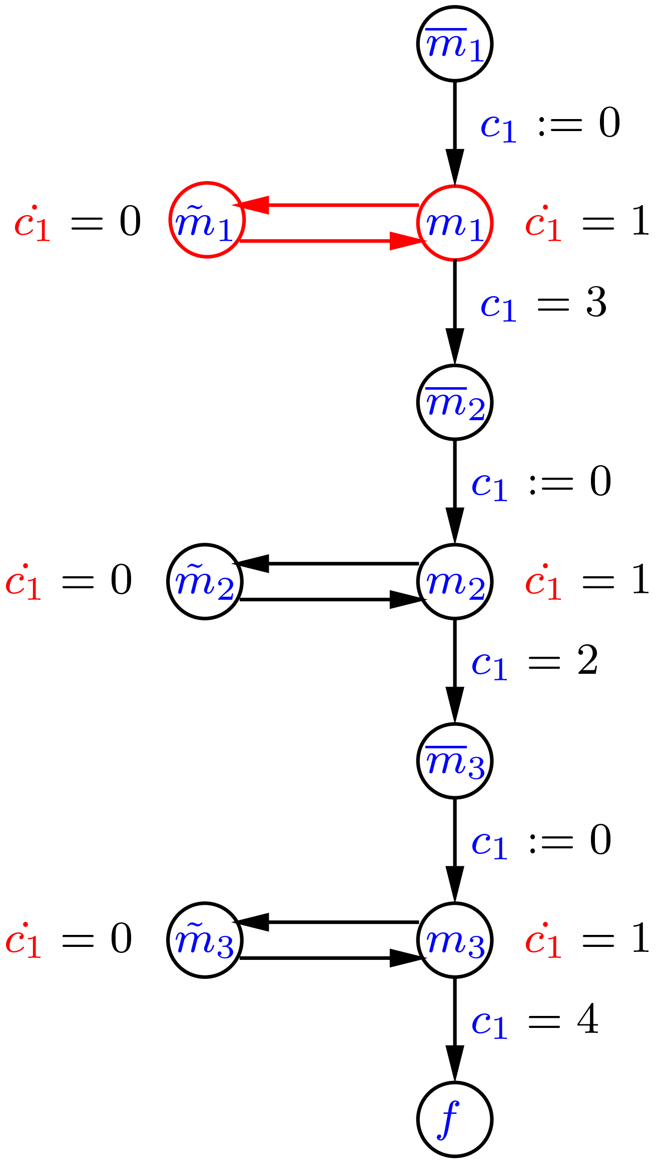


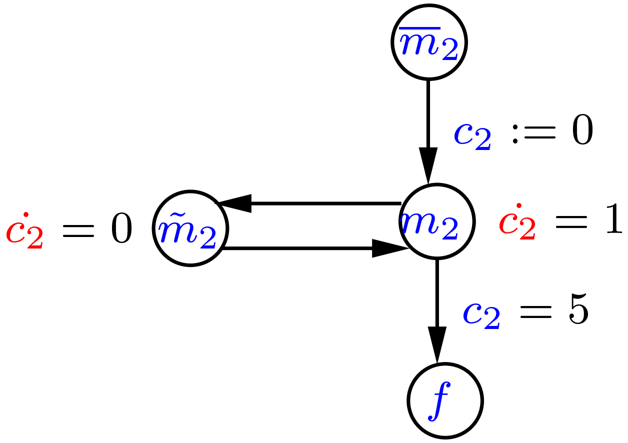
$$J^1 = (m_1, 3), (m_2, 2), (m_3, 4)$$

$$J^2 = (m_2, 5)$$

Here waiting when the machine is idle is useless

Modeling Jobs with Stopwatch Automata

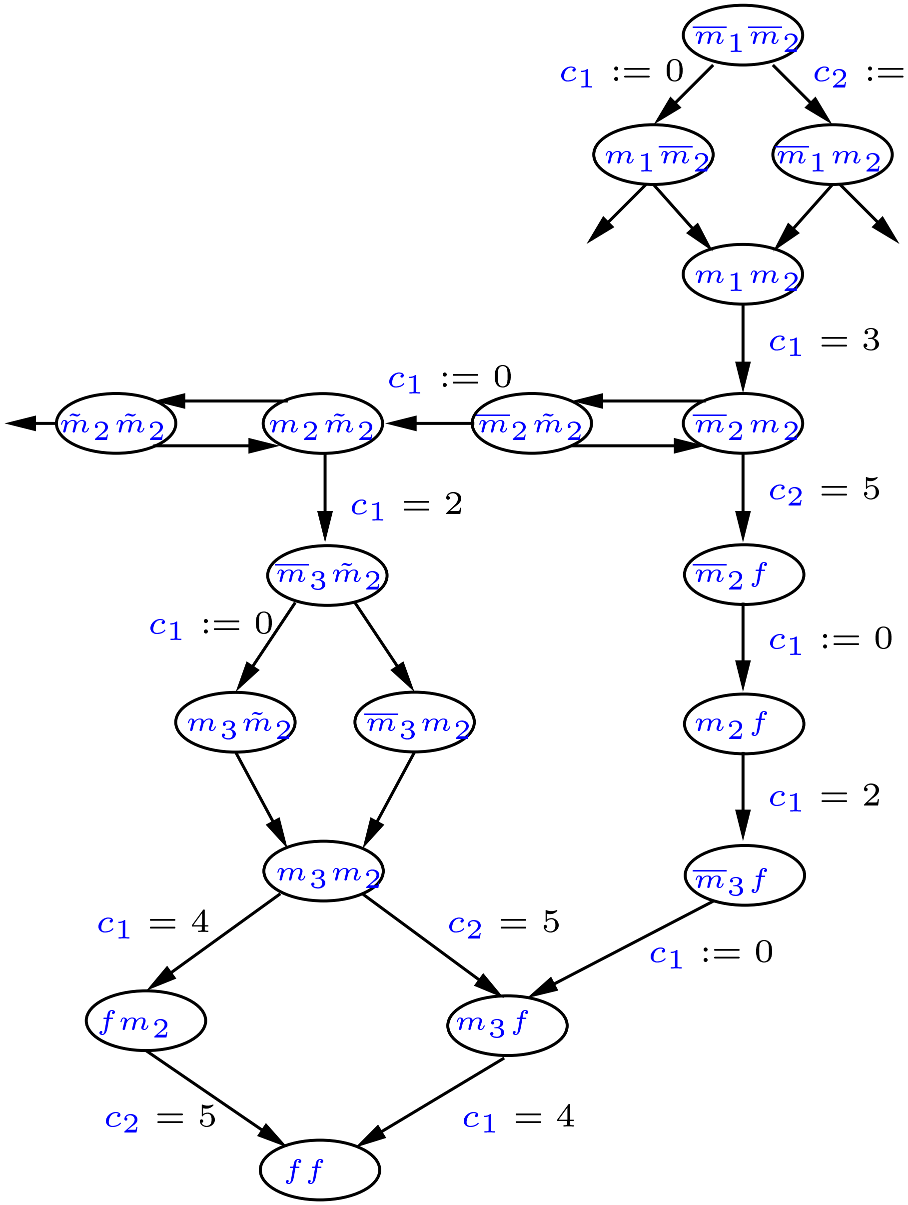




$$J_1 = (m_1, 3), (m_2, 2), (m_3, 4)$$

$$J_2 = (m_2, 5)$$

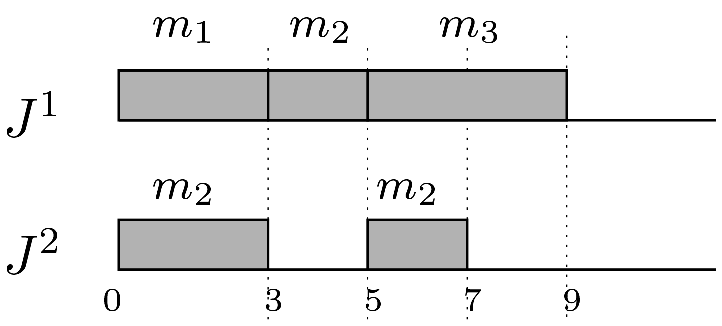
The Global Model of the Example



Runs and Schedules

Every complete run of $\mathcal{A}$ corresponds to a feasible schedule

$$(\overline{m}_1, \overline{m}_2, \bot, \bot) \xrightarrow{0} (m_1, \overline{m}_2, 0, \bot) \xrightarrow{0} (m_1, m_2, 0, 0) \xrightarrow{3} (m_1, m_2, 3, 3) \xrightarrow{0}$$
$$(\overline{m}_2, m_2, \bot, 3) \xrightarrow{0} (\overline{m}_2, \tilde{m}_2, \bot, 3) \xrightarrow{0} (m_2, \tilde{m}_2, 0, 3) \xrightarrow{2} (m_2, \tilde{m}_2, 2, 3) \xrightarrow{0}$$
$$(\overline{m}_3, \tilde{m}_2, \bot, 3) \xrightarrow{0} (\overline{m}_3, m_2, \bot, 3) \xrightarrow{0} (m_3, m_2, 0, 3) \xrightarrow{2} (m_3, m_2, 2, 5) \xrightarrow{0}$$
$$(m_3, f, 2, \bot) \xrightarrow{2} (m_3, f, 4, \bot) \xrightarrow{0} (f, f, \bot, \bot)$$

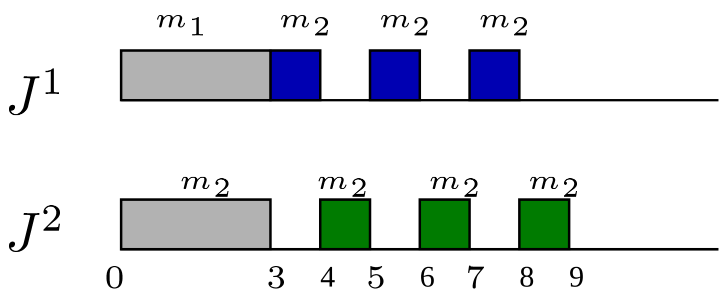


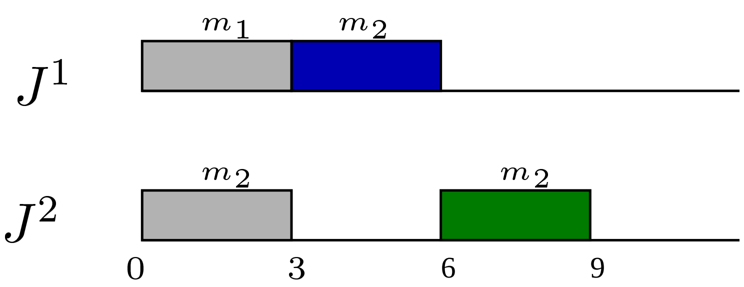
Preemptive Job Shop $\Longleftrightarrow$ Shortest Path in Stopwatch Automata

Problem: Infinitely many runs due to cycles

Useless premption and Efficient runs

Intuition: it is useless to give a machine to another job if you take it back before it terminate

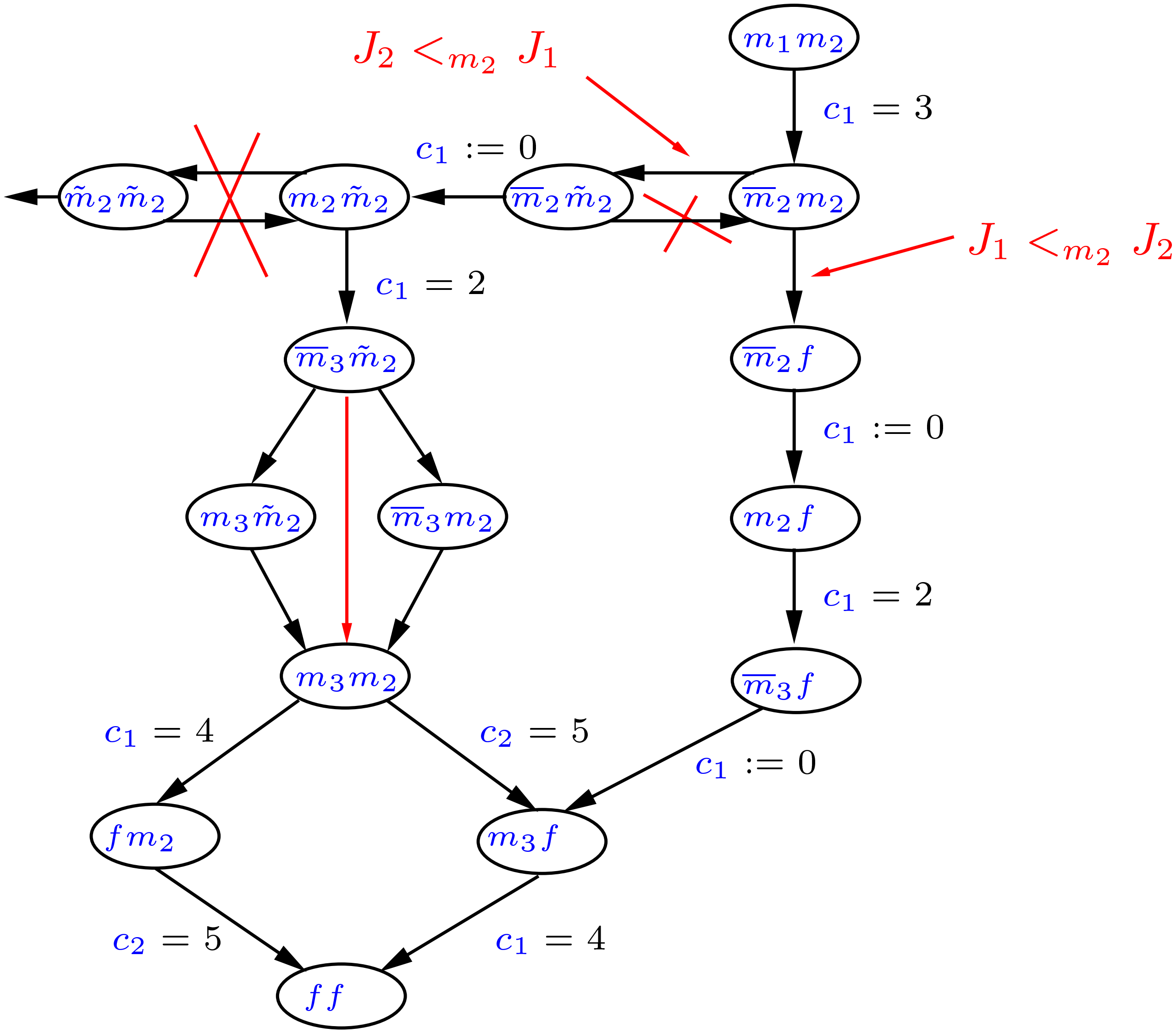




We could achieve the same result by giving priority to the one that terminates first

Every non-efficent schedule S can be transformed into an efficient schedule $\hat{S}$ with $|\hat{S}| \leq |S|$

How to enumerate efficient runs



Benchmark Problems

problem			preemptive			
name	#j	#m	optimum	PB96,PBb97	stopwatch	deviation
LA02	10	5	655	655	655	0.00 %
FT10	10	10	900	900	911	1.21 %
ABZ5	10	10	1203	1206	1250	3.76 %
ABZ6	10	10	924	924	936	1.28 %
ORB1	10	10	1035	1035	1093	5.31 %
ORB2	10	10	864	864	884	2.26 %
ORB3	10	10	973	994	1013	3.95 %
ORB4	10	10	980	980	1004	2.39 %
ORB5	10	10	849	849	887	4.28 %
LA19	10	10	812	812	843	3.68 %
LA20	10	15	871	871	904	3.65 %
LA21	10	15	1033	1033	1086	4.88 %
LA24	10	15	909	915	972	6.48 %
LA27	10	20	1235	1235	1312	5.87 %
LA37	15	15	1397	1397	1466	4.71 %
LA39	15	15	1221	1221	1283	4.83 %

Plan

1. Introduction

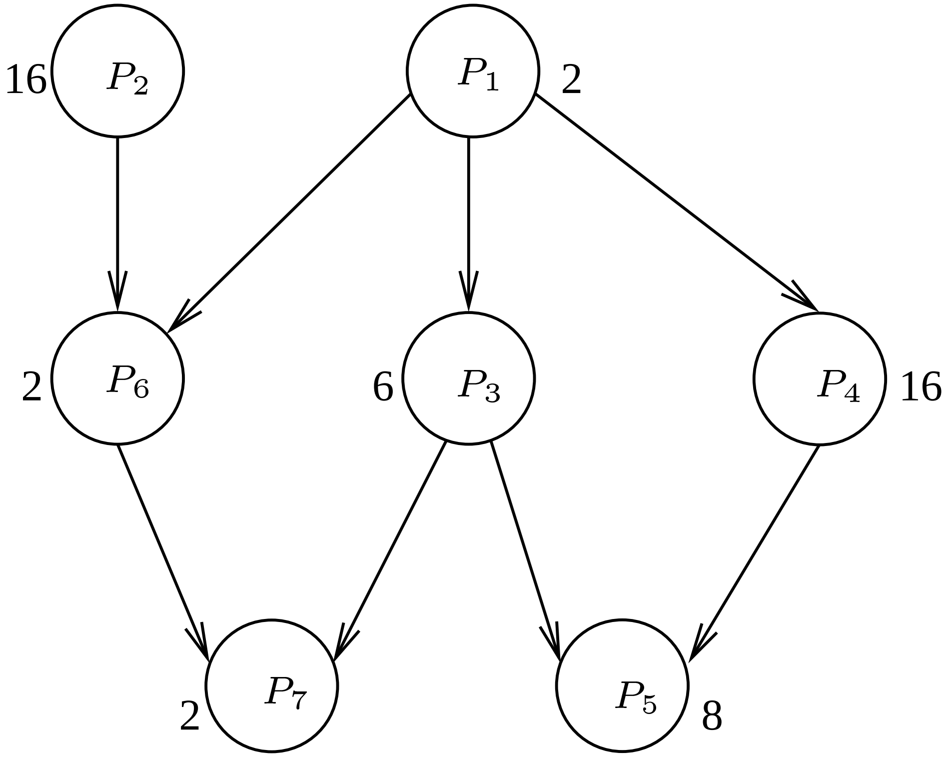
2. Deterministic Job-Shop Scheduling

3. Preemptive Job-Shop Scheduling

4. Task Graph Scheduling

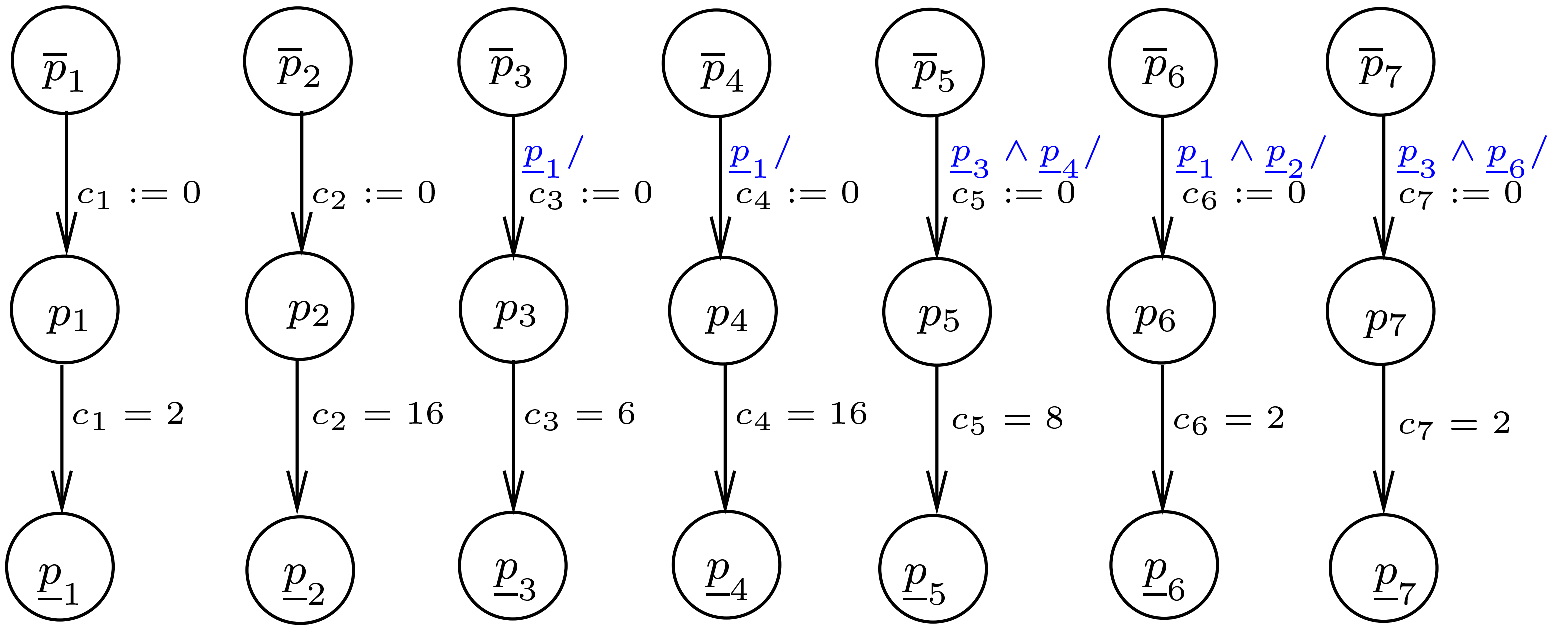
5. Scheduling Under Uncertainty

The Problem



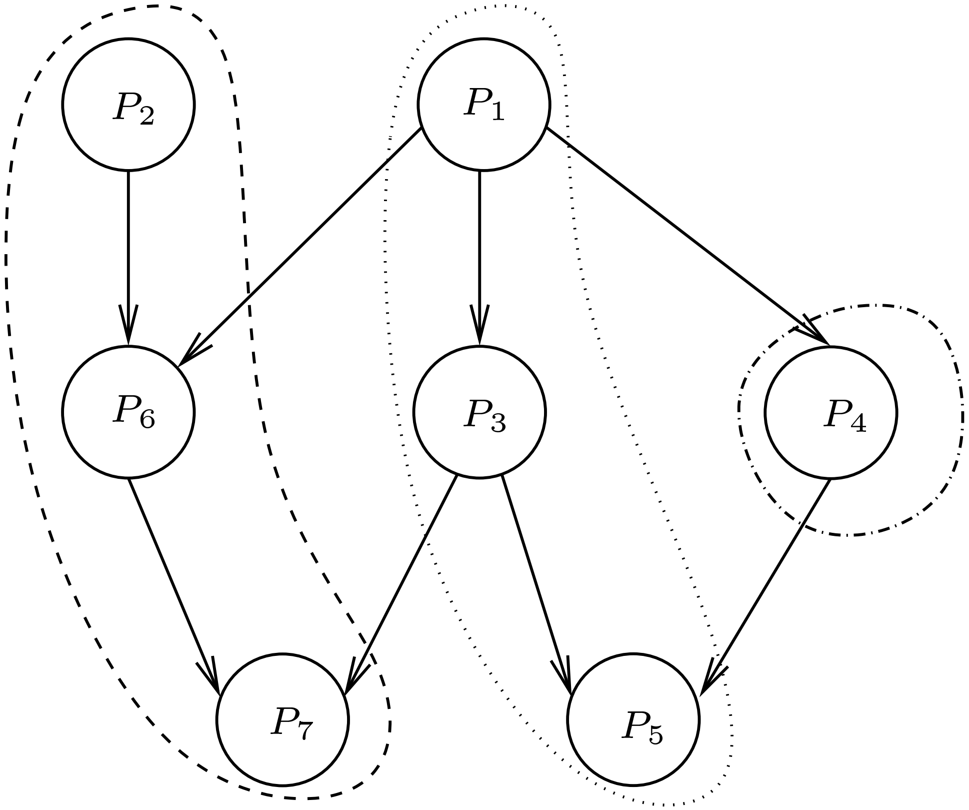
Schedule on n identical machines while respecting precedence

Modeling



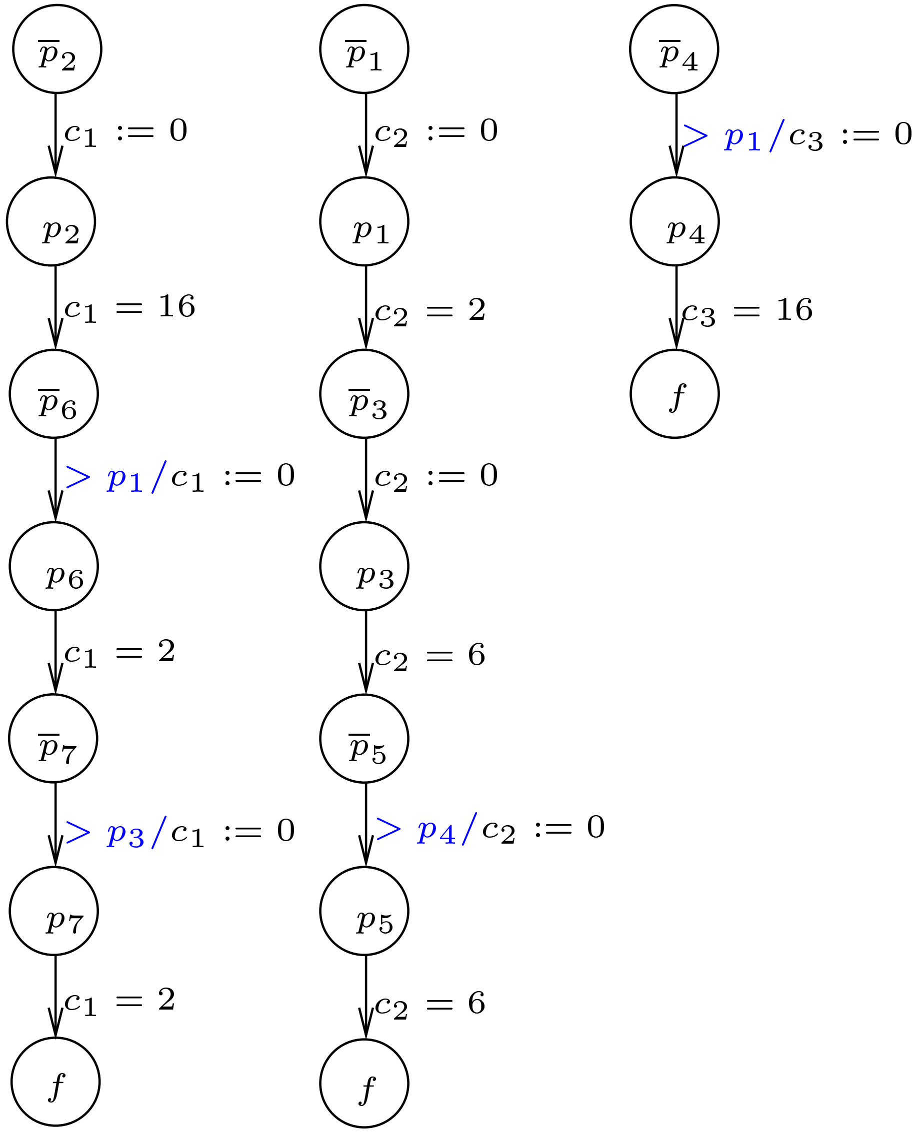
One state variable and one clock for each task

Decomposition into chains

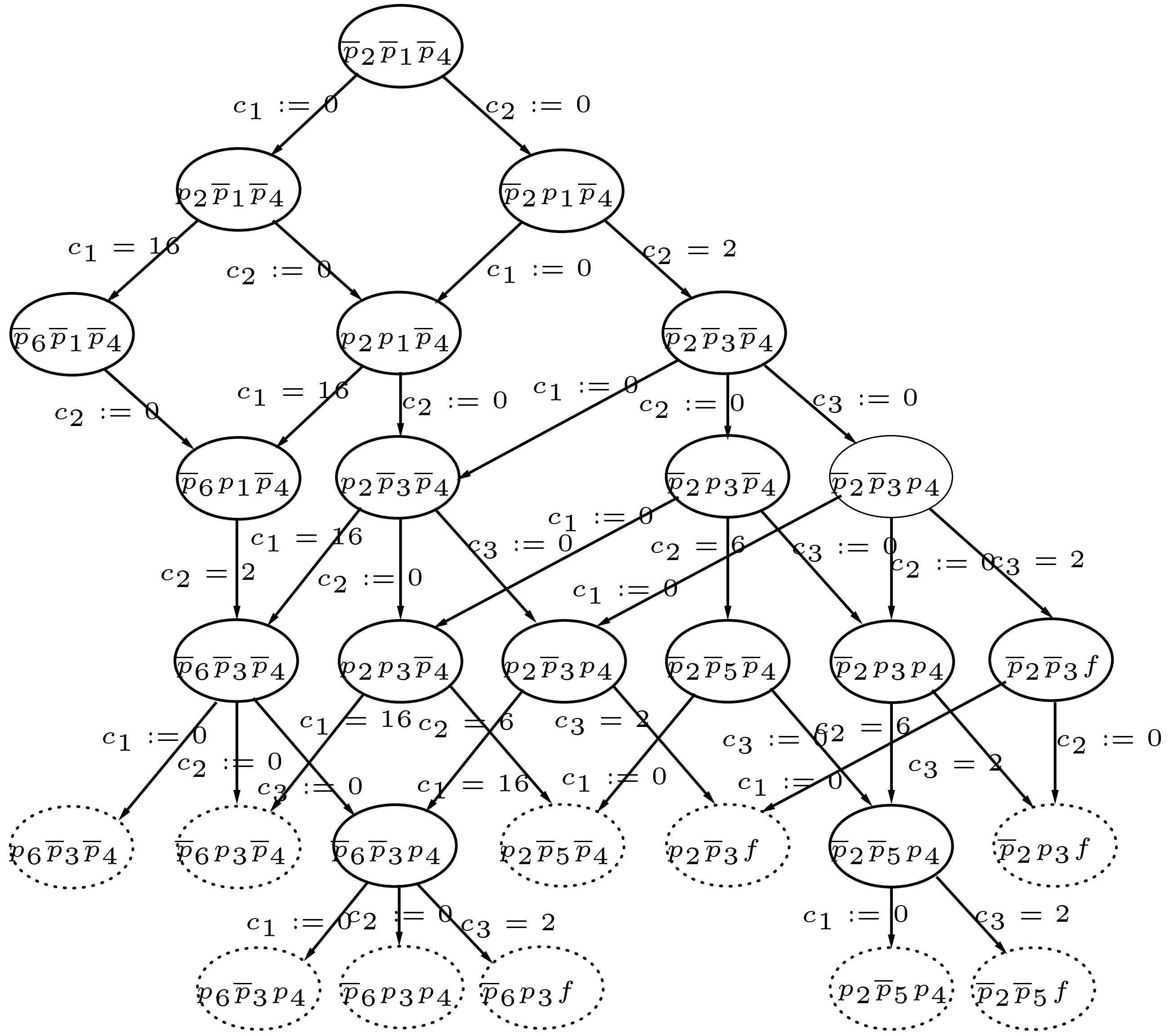


Chain = Job + some interaction

Automata for chains



Global model for 2 machines



Experimental Results

name	#tasks	#chains	# machines	Opt	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o	1137
235	1782	218	16	t.o	1150
233	1980	207	19	1118	1121
294	2014	141	17	1257	1261
295	2168	965	18	1318	1322
292	2333	318	3	8009	8009
298	2399	303	10	2471	2473

Plan

1. Introduction
2. Deterministic Job-Shop Scheduling
3. Preemptive Job-Shop Scheduling
4. Task Graph Scheduling
5. Scheduling Under Uncertainty

The Problem

A job shop problem where the duration of tasks is only known to be inside an interval $[l, u]$

$$J^1 = (m_1, 10)\ (m_3, [2, 4])\ (m_4, 5)$$
$$J^2 = (m_2, [2, 8])\ (m_3, 7)$$

Each instance $d = (d_1, d_2) = [2, 4] \times [2, 8]$ defines a deterministic problem

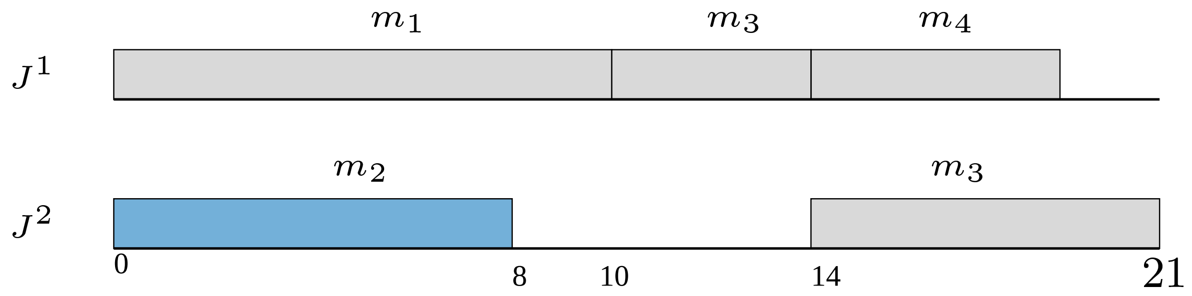
$$J^1 = (m_1, 10)\ (m_3, d_1)\ (m_4, 5)$$
$$J^2 = (m_2, d_2)\ (m_3, 7)$$

What is optimality ? With respect to which instance ?

Static (Worst-Case) Schedule

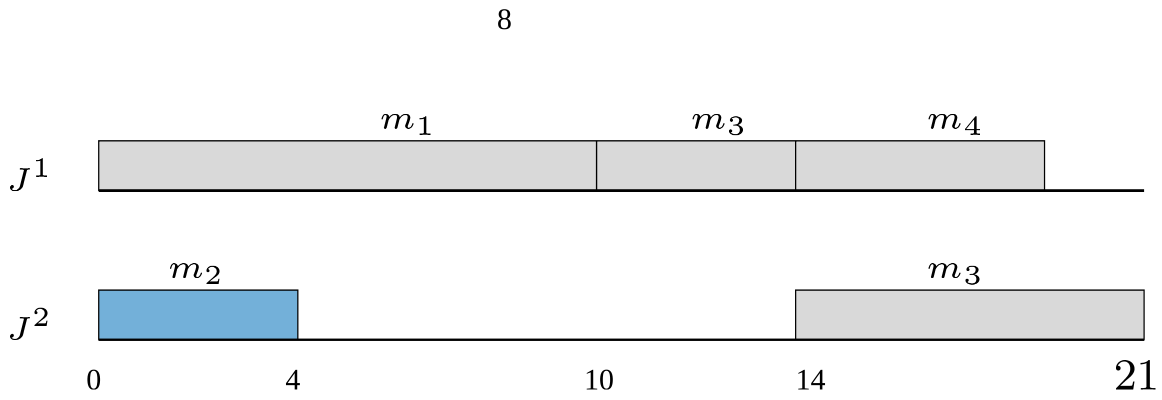
$J^1 = (m_1, 10)\ (m_3, 4)\ (m_4, 5)$
$J^2 = (m_2, [2, 8])\ (m_3, 7)$

$d = 8$



If we care only for worst case we do not change the schedule when $d = 4$

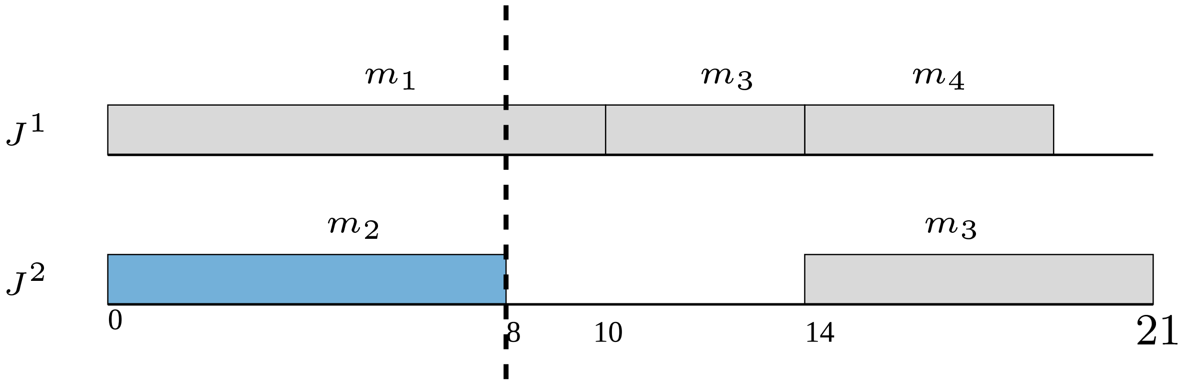
$d = 4$



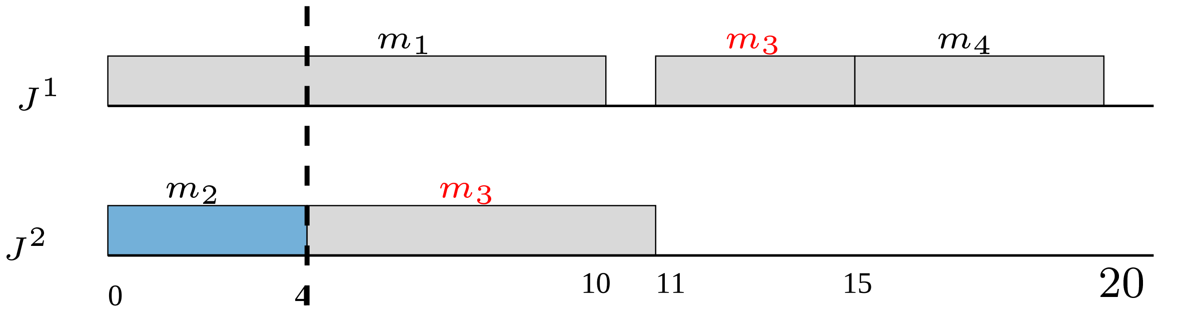
Adaptive Schedule: Online Rescheduling

$J^1 = (m_1, 10)\ (m_3, 4)\ (m_4, 5)$
$J^2 = (m_2, [2, 8])\ (m_3, 7)$

residual problem: $J_1' = (\mathbf{m_1}, \mathbf{2})\ (m_3, 4)\ (m_4, 5) \qquad J_2' = (m_3, 7)$

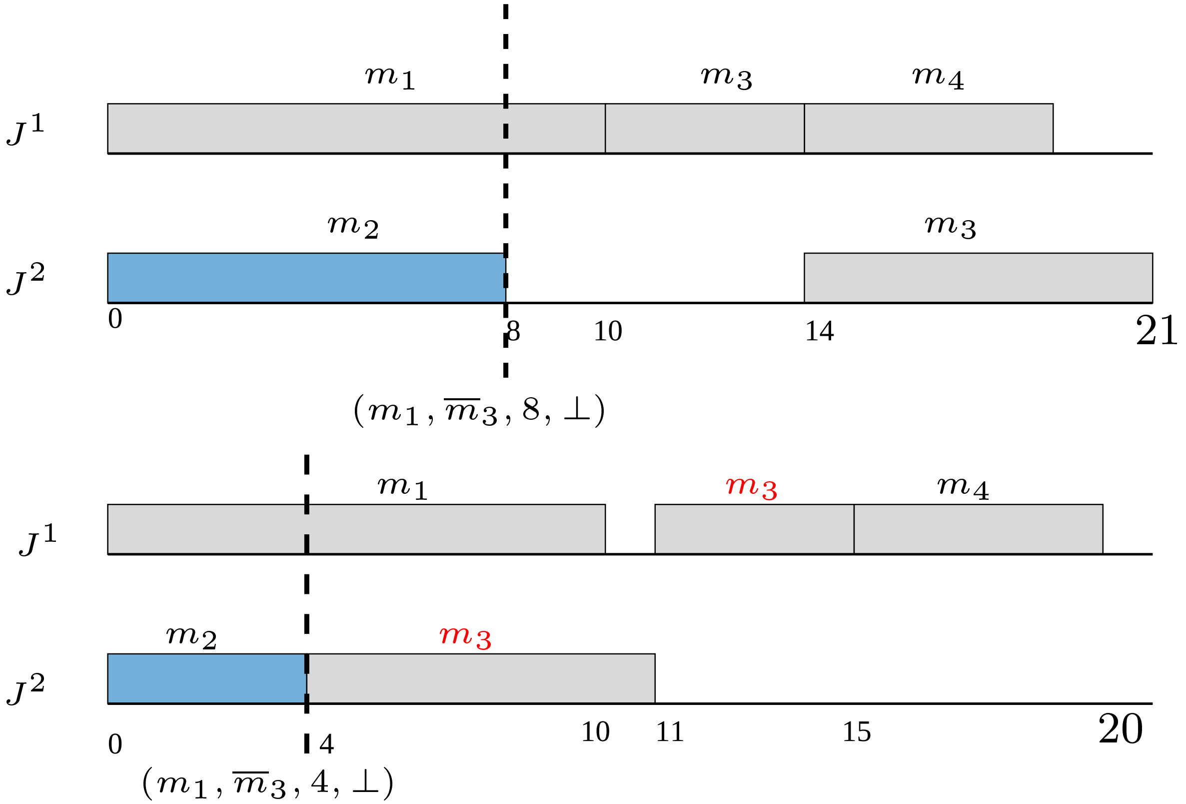


residual problem: $J_1' = (\mathbf{m_1}, \mathbf{6})\ (m_3, 4)\ (m_4, 5) \qquad J_2' = (m_3, 7)$



Adaptive Schedule: Offline Computed Strategy

$J^1 = (m_1, 10)\ (m_3, 4)\ (m_4, 5)$
$J^2 = (m_2, [2, 8])\ (m_3, 7)$



$$\text{Strategy: } \sigma(m_1, \overline{m}_3, c_1, \perp) \quad = \quad \begin{cases} \text{if } c_1 \leq 5 \text{ start } m_3 \\ \text{if } c_1 \geq 5 \text{ wait for } m_1 \text{ to terminate} \end{cases}$$

Computing strategy by backward reachability

value function: $h(q, q, c_1, c_2)$ = minimal time to reach (f, f)

$$
\begin{aligned}
h(f, f, \perp, \perp) &= 0 \\
h(m_4, f, c_1, \perp) &= 5 \dot{-} c_1 \\
h(f, m_3, \perp, c_2) &= 7 \dot{-} c_2 \\
&. \\
&. \\
h(m_1, \overline{m}_3, c_1, \perp) &= \min\{16, 21 \dot{-} c_1\} \\
&= \begin{cases} 16 & \text{if } c_1 \leq 5 \\ 21 \dot{-} c_1 & \text{if } c_1 \geq 5 \end{cases}
\end{aligned}
$$

$$
\sigma(m_1, \overline{m}_3, c_1, \perp) = \begin{cases} (m_1, m_3) & \text{if } c_1 \leq 5 \\ \perp & \text{if } c_1 \geq 5 \end{cases}
$$

Experimental Results

$$
\begin{array}{ll}
J_1: & (m_2,[\ 4,10])\ (m_4,[\ 1,\ \ 7])\ (m_3,[28,40])\ (m_1,[\ 7,15])\ (m_5,[\ 6,25])\ (m_6,[45,63]) \\
J_2: & (m_5,[14,25])\ (m_1,[34,46])\ (m_2,[\ 2,27])\ (m_4,[\ 9,14])\ (m_6,[14,29])\ (m_3,[32,48]) \\
J_3: & (m_4,[47,55])\ (m_6,[32,46])\ (m_1,[\ 4,12])\ (m_5,[\ 1,14])\ (m_2,[\ 5,16])\ (m_3,[\ 4,\ \ 9]) \\
J_4: & (m_6,[54,72])\ (m_2,[21,36])\ (m_3,[\ 1,\ \ 8])\ (m_4,[22,37])\ (m_1,[\ 7,18])\ (m_5,[\ 4,18])
\end{array}
$$

Inst	Opt	Static	%	Max	%	Min	%
1	172	204	18.60	172	0.00	205	19.186
2	193	210	8.80	193	0.00	193	0.00
3	175	208	16.00	181	3.43	194	10.85
4	177	199	12.42	177	0.00	196	10.73
5	186	209	12.36	189	1.16	189	1.61
6	176	203	15.34	177	0.57	197	11.93
7	180	206	14.44	180	0.00	195	8.33
8	167	204	22.15	171	2.40	183	9.58
9	202	206	1.98	202	0.00	203	0.49
10	166	202	6.87	166	0.00	197	18.67
11	189	202	6.87	189	0.00	221	16.93
12	176	199	13.06	176	0.00	192	9.09
13	180	204	13.33	180	0.00	192	6.66
14	167	204	22.15	171	2.40	178	6.58
15	178	204	14.60	180	1.12	201	12.92
16	175	202	15.42	182	4.00	204	17.24
17	174	202	16.09	174	0.00	191	9.77
18	176	201	14.20	180	2.27	192	9.09
19	170	199	17.05	170	0.00	182	70.5
20	167	202	20.95	168	0.60	183	9.58
21	185	210	13.51	185	0.00	200	8.10
22	158	203	28.48	163	3.16	185	17.08
23	179	199	11.17	179	0.00	210	17.31
Avg	175	203.33	16.18	179.19	2.39	191.93	9.67

Conclusions

Why reinvent the wheel? Job-shop scheduling problems can be solved using many other methods
Answer: automata and timed automata give dynamic models with well-defined semantics.

- We have shown that they are not inferior to other methods for job-shop, preemptive job-shop and task graph problems.
- We have shown that they can be used to model and solve problems, such as scheduling under uncertainty, which are not expressed naturally in traditional models.
- We have also shown that sometimes we can explore timed automata without using zones nor too refined discretizations.
- All algorithms have been implemented and will be integrated in future versions of Verimag tools.

Future Work

Extensions to the model

- Deadlines and release time (done); relative deadlines, upper bound constraints: this breaks the non-laziness results
- Different machines can do the same step at different speeds
- A task can hold more than one resource at a time (disk, printer) - this may cause deadlocks
- Discrete uncertainty: the successor of a step is known only when it terminates (in computer programs, for example).
- Additional steps that depend on the schedule (transportation between machines, communication between processors)
- Periodic tasks

Future Work

New Algorithms

- Compositional scheduling
- Combination of forward and backward search for strategy synthesis