# No-Idle, No-Wait: When Shop Scheduling Meets Dominoes, Eulerian and Hamiltonian Paths

J.C. Billaut[1], F.Della Croce[2], *Fabio Salassa*[2], V. T'kindt[1]

1. Université Francois-Rabelais, CNRS, Tours, France
2. Politecnico di Torino, Torino, Italy

**Monash University**

Melbourne, February 19th, 2018

# Flow Shop Scheduling

# Flow Shop Scheduling

- There are $m$ machines and $n$ jobs.

# Flow Shop Scheduling

- There are $m$ machines and $n$ jobs.
- Each job contains exactly $m$ operations.

# Flow Shop Scheduling

- There are $m$ machines and $n$ jobs.
- Each job contains exactly $m$ operations.
- For each operation of each job a processing time is specified.

# Flow Shop Scheduling

- There are $m$ machines and $n$ jobs.
- Each job contains exactly $m$ operations.
- For each operation of each job a processing time is specified.
- No machine can perform more than one operation simultaneously.

# Flow Shop Scheduling

- There are $m$ machines and $n$ jobs.
- Each job contains exactly $m$ operations.
- For each operation of each job a processing time is specified.
- No machine can perform more than one operation simultaneously.
- Operations cannot be interrupted (no preemption).

# Flow Shop Scheduling

- Operations within one job must be performed in the specified order.

# Flow Shop Scheduling

- Operations within one job must be performed in the specified order.
- The first operation gets executed on the first machine, then (as the first operation is finished) the second operation on the second machine, and so until the $m$-th operation.

# Flow Shop Scheduling

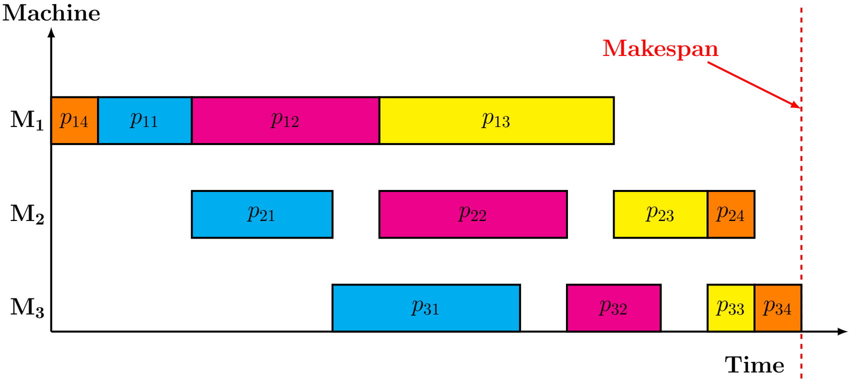
- Operations within one job must be performed in the specified order.
- The first operation gets executed on the first machine, then (as the first operation is finished) the second operation on the second machine, and so until the $m$-th operation.
- Jobs can be executed in any order.

# Flow Shop Scheduling

- Operations within one job must be performed in the specified order.
- The first operation gets executed on the first machine, then (as the first operation is finished) the second operation on the second machine, and so until the $m$-th operation.
- Jobs can be executed in any order.
- The problem is to determine an **optimal arrangement** of jobs.

# Flow Shop Scheduling – Example

| $j$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ |
|---|---|---|---|---|
| $p_{1,j}$ | 2 | 4 | 5 | 1 |
| $p_{2,j}$ | 3 | 4 | 2 | 1 |
| $p_{3,j}$ | 4 | 2 | 1 | 1 |



# Problem Description

# Problem Description

- We consider flow shop scheduling problems with **(machine) no-idle, (job) no-wait** constraints and makespan as objective.

# Problem Description

- We consider flow shop scheduling problems with **(machine) no-idle, (job) no-wait** constraints and makespan as objective.
- **Machine no-idle constraint**: use of very expensive equipment with the fee determined by the actual time consumption.

# Problem Description

- We consider flow shop scheduling problems with **(machine) no-idle, (job) no-wait** constraints and makespan as objective.
- **Machine no-idle constraint**: use of very expensive equipment with the fee determined by the actual time consumption.
- **Job no-wait constraint**: in metal-processing industries (e.g., hot rolling) where delays between operations interfere with the technological process (e.g., cooling down).

# Problem Description

- We consider flow shop scheduling problems with **(machine) no-idle, (job) no-wait** constraints and makespan as objective.
- **Machine no-idle constraint**: use of very expensive equipment with the fee determined by the actual time consumption.
- **Job no-wait constraint**: in metal-processing industries (e.g., hot rolling) where delays between operations interfere with the technological process (e.g., cooling down).
- We focus on problem $F2|$ no-idle, no-wait $|C_{\max}$.

# Literature

- Problem $F2||C_{\max}$ (Johnson rule $O(nlogn)$) $\in P$.

# Literature

- Problem $F2||C_{\max}$ (Johnson rule $O(nlogn)$) $\in P$.
- Problem $F2|$ no-idle $|C_{\max}$ (trivially packing the jobs on the second machine to the right from Johnson's schedule) $\in P$.

# Literature

- Problem $F2||C_{\max}$ (Johnson rule $O(nlogn)$) $\in P$.
- Problem $F2|$ no-idle $|C_{\max}$ (trivially packing the jobs on the second machine to the right from Johnson's schedule) $\in P$.
- Problem $F2|$ no-wait $|C_{\max}$ (special case of Gilmore-Gomory TSP) $\in P$.

# Literature

- Problem $F2||C_{\max}$ (Johnson rule $O(nlogn)$) $\in P$.
- Problem $F2|$ no-idle $|C_{\max}$ (trivially packing the jobs on the second machine to the right from Johnson's schedule) $\in P$.
- Problem $F2|$ no-wait $|C_{\max}$ (special case of Gilmore-Gomory TSP) $\in P$.
- Problem $F3||C_{\max}$ is $NP$-hard.

# Literature

- Problem $F2|$ no-wait $|\mathcal{G}$ (minimizing the number of interruptions on the last machine in a 2-machine no-wait flow shop) is solvable in $O(n^2)$ time (Hohn et al. 2012).

# Literature

- Problem $F2|$ no-wait $|\mathcal{G}$ (minimizing the number of interruptions on the last machine in a 2-machine no-wait flow shop) is solvable in $O(n^2)$ time (Hohn et al. 2012).

- Problem $F3|$ no-wait $|\mathcal{G}$ is $NP$-hard. (Hohn et al. 2012).

# Literature

- Problem $F2|$ no-wait $|\mathcal{G}$ (minimizing the number of interruptions on the last machine in a 2-machine no-wait flow shop) is solvable in $O(n^2)$ time (Hohn et al. 2012).
- Problem $F3|$ no-wait $|\mathcal{G}$ is $NP$-hard. (Hohn et al. 2012).
- Problems $F2||\sum C_j$, $F2|$ no-wait $|\sum C_j$, $F2|$ no-idle $|\sum C_j$, $F2|$ no-idle, no-wait $|\sum C_j$ are $NP$-hard (Adiri and Pohoryles 1982).

# $F2|$ no-idle, no-wait $|C_{\max}$

# $F2|$ no-idle, no-wait $|C_{\max}$

- Two Machines $M_1$, $M_2$.

# $F2|$ no-idle, no-wait $|C_{max}$

- Two Machines $M_1$, $M_2$.
- Flow Shop environment.

# $F2|$ no-idle, no-wait $|C_{\max}$

- Two Machines $M_1$, $M_2$.
- Flow Shop environment.
- $n$ jobs $1, 2, \ldots, n$ with processing times $p_{1j}$ and $p_{2j}$.

# $F2|$ no-idle, no-wait $|C_{\max}$

- Two Machines $M_1$, $M_2$.
- Flow Shop environment.
- $n$ jobs $1, 2, \ldots, n$ with processing times $p_{1j}$ and $p_{2j}$.
- *No-idle* time is allowed (both machines $M_1, M_2$ must work continuously).

# $F2|$ no-idle, no-wait $|C_{max}$

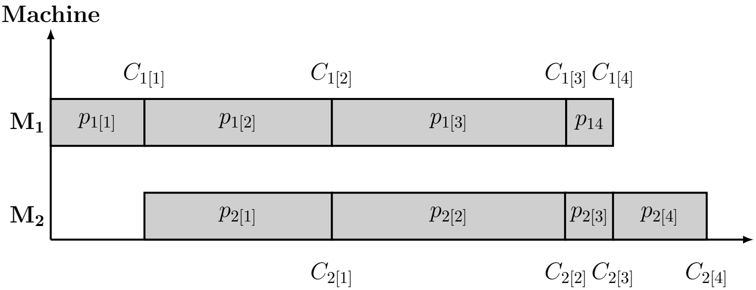
- Two Machines $M_1$, $M_2$.
- Flow Shop environment.
- $n$ jobs $1, 2, \ldots, n$ with processing times $p_{1j}$ and $p_{2j}$.
- *No-idle* time is allowed (both machines $M_1, M_2$ must work continuously).
- *No-wait* discipline (no buffer — each job must start on $M_2$ right after its completion on $M_1$).

# $F2|$ no-idle, no-wait $|C_{max}$

- Two Machines $M_1$, $M_2$.
- Flow Shop environment.
- $n$ jobs $1, 2, \ldots, n$ with processing times $p_{1j}$ and $p_{2j}$.
- *No-idle* time is allowed (both machines $M_1, M_2$ must work continuously).
- *No-wait* discipline (no buffer — each job must start on $M_2$ right after its completion on $M_1$).
- Makespan (i.e. total time that elapses from the beginning to the end) objective.

# $F2|$ no-idle, no-wait $|C_{max}$

- The **no-idle**, **no-wait** constraint is a very strong requirement.




- $p_{i[j]}$ denotes the processing time of the $j$-th job of a sequence $\sigma$ on machine $M_i$.
- $C_{i[j]}$ denotes the completion time of the $j$-th job of a sequence $\sigma$ on machine $M_i$.

# $F2|$ no-idle, no-wait $|C_{\max}$

## Lemma (1)

(C1) *A necessary condition to have a feasible solution for problem $F2|no-idle, no-wait|C_{\max}$ is that there always exists an indexing of the jobs so that $p_{1,2}, ...p_{1,n}$ and $p_{2,1}, ..., p_{2,n-1}$ constitute different permutations of the same vector of elements.*

# $F2|$ no-idle, no-wait $|C_{max}$

## Lemma (1)

(C1) *A necessary condition to have a feasible solution for problem $F2|no - idle, no - wait|C_{max}$ is that there always exists an indexing of the jobs so that $p_{1,2}, ...p_{1,n}$ and $p_{2,1}, ..., p_{2,n-1}$ constitute different permutations of the same vector of elements.*

| $j$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ |
|---|---|---|---|---|---|
| $p_{1,j}$ | 5 | 8 | 7 | 6 | 7 |
| $p_{2,j}$ | 8 | 5 | 6 | 7 | 7 |

# $F2|$ no-idle, no-wait $|C_{max}$

## Lemma (1)

(C1) *A necessary condition to have a feasible solution for problem $F2|no-idle, no-wait|C_{max}$ is that there always exists an indexing of the jobs so that $p_{1,2},...p_{1,n}$ and $p_{2,1},...,p_{2,n-1}$ constitute different permutations of the same vector of elements.*

# $F2|$ no-idle, no-wait $|C_{max}$

## Lemma (1)

(C1) *A necessary condition to have a feasible solution for problem $F2|no-idle, no-wait|C_{\max}$ is that there always exists an indexing of the jobs so that $p_{1,2}, ...p_{1,n}$ and $p_{2,1}, ..., p_{2,n-1}$ constitute different permutations of the same vector of elements.*

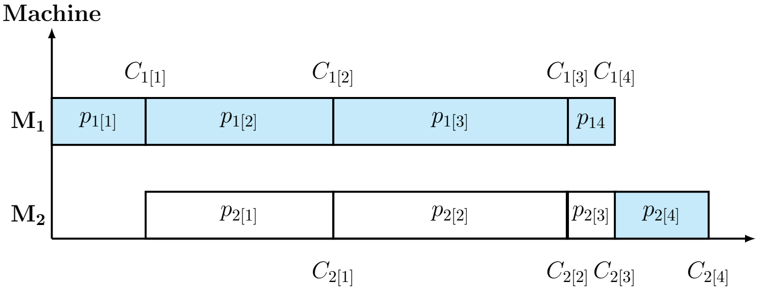
(C2) *When the above condition (C1) holds, then*

- Case 1 *if $p_{1,1} \neq p_{2,n}$, every feasible sequence must have a job with processing time $p_{1,1}$ in first position and a job with processing time $p_{2,n}$ in last position.*
- Case 2 *if $p_{1,1} = p_{2,n}$ and there exists a feasible sequence, then there do exist at least n feasible sequences each starting with a different job by simply rotating the starting sequence as in a cycle.*

# $F2|$ no-idle, no-wait $|C_{\max}$

## Lemma (2)

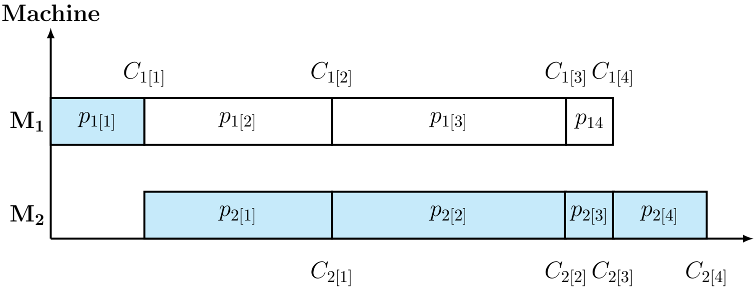
*The makespan of any feasible sequence $\sigma$ is given by the processing time of the* **last** *(first) job on the* **second** *(first) machine plus the sum of jobs processing times on the* **first** *(second) machine.*



# $F2|$ no-idle, no-wait $|C_{\max}$

## Lemma (2)

*The makespan of any feasible sequence $\sigma$ is given by the processing time of the last* **(first)** *job on the second* **(first)** *machine plus the sum of jobs processing times on the first* **(second)** *machine.*



# $F2|$ no-idle, no-wait $|C_{\max}$: an example

A 9-job instance of problem $F2|no-idle, no-wait|C_{\max}$.

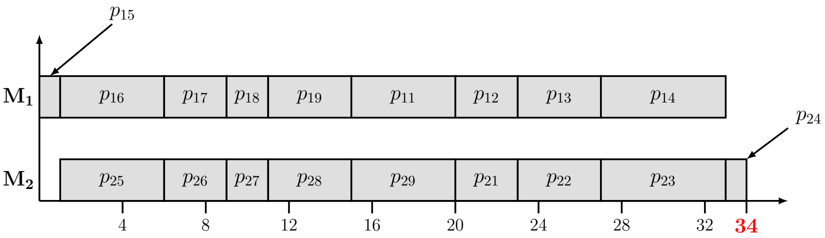
| $j$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ | $J_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $p_{1,j}$ | 5 | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 |
| $p_{2,j}$ | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 | 5 |

# $F2|$ no-idle, no-wait $|C_{\max}$: an example

A 9-job instance of problem $F2|no-idle, no-wait|C_{\max}$.

| $j$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ | $J_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $p_{1,j}$ | 5 | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 |
| $p_{2,j}$ | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 | 5 |

and the corresponding optimal solution $C_{max} = 34$



# $F2|$ no-idle, no-wait $|C_{\max}$

- Due to Lemma 1 and $F2|no-idle, no-wait|\mathcal{G}$ problem, the optimal solution can be calculated in $O(n^2)$ time...

# $F2|$ no-idle, no-wait $|C_{\max}$

- Due to Lemma 1 and $F2|no-idle, no-wait|\mathcal{G}$ problem, the optimal solution can be calculated in $O(n^2)$ time...

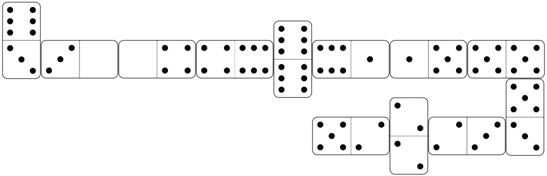
...but...

# $F2|$ no-idle, no-wait $|C_{\max}$

- Due to Lemma 1 and $F2|no-idle, no-wait|\mathcal{G}$ problem, the optimal solution can be calculated in $O(n^2)$ time...

...but...

- ...we decided to link the problem to the game of dominoes



# Dominoes

- The **Single Player Domino (SPD) problem** (where a single player tries to lay down all dominoes in a chain with the numbers matching at each adjacency) is **polynomially solvable**: it can be seen as a **eulerian path problem on an undirected multigraph**.

# Dominoes

- The **Single Player Domino (SPD) problem** (where a single player tries to lay down all dominoes in a chain with the numbers matching at each adjacency) is **polynomially solvable**: it can be seen as a **eulerian path problem on an undirected multigraph**.
- Here, we refer to the **oriented version of SPD called OSPD** where all dominoes have an orientation (given a tile with numbers $i$ and $j$, only the orientation $i \rightarrow j$ is allowed but not viceversa).

# Dominoes

- The **Single Player Domino (SPD) problem** (where a single player tries to lay down all dominoes in a chain with the numbers matching at each adjacency) is **polynomially solvable**: it can be seen as a **eulerian path problem on an undirected multigraph**.
- Here, we refer to the **oriented version of SPD called OSPD** where all dominoes have an orientation (given a tile with numbers $i$ and $j$, only the orientation $i \rightarrow j$ is allowed but not viceversa).
- **Problem OSPD is polynomially solvable** (can be seen as a eulerian path problem on a **directed** multigraph).

# Problem $F2|no-idle, no-wait|C_{\max}$ vs OSPD

**Proposition**

$F2|no-idle, no-wait|C_{\max} \propto OSPD.$

# Problem $F2|no-idle, no-wait|C_{\max}$ vs OSPD

**Proposition**

$F2|no-idle, no-wait|C_{\max} \propto OSPD.$

**Proof.**

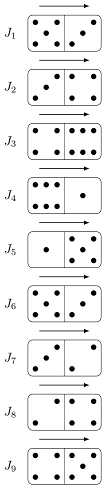
By generating for each job $J_j$ a related domino tile $\{p_{1,j}, p_{2,j}\}$, any complete sequence of oriented dominoes in OSPD corresponds to a feasible sequence for $F2|no-idle, no-wait|C_{\max}$. Then, due to Lemma 1, the jobs processing times either respect case 1 or case 2 of condition C2. □

# An example

A 9-job instance of problem $F2|no-idle, no-wait|C_{\max}$.

| $i$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ | $J_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $p_{1,j}$ | 5 | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 |
| $p_{2,j}$ | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 | 5 |

and the corresponding dominoes of the related OSPD problem

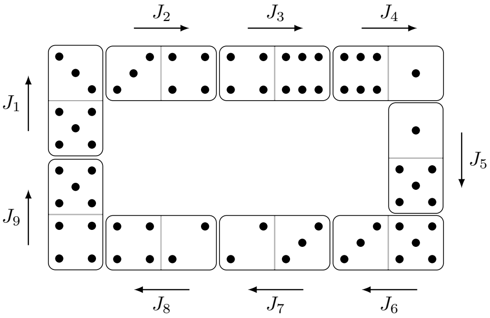


# An example

A 9-job instance of problem $F2|no-idle, no-wait|C_{\max}$.

| $j$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ | $J_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $p_{1,j}$ | 5 | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 |
| $p_{2,j}$ | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 | 5 |

and the corresponding OSPD solution

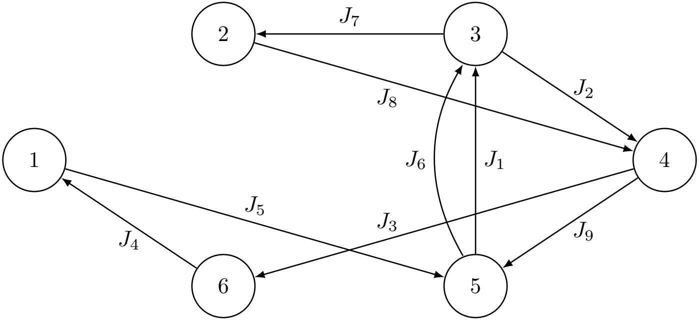


# An example

A 9-job instance of problem $F2|no-idle, no-wait|C_{\max}$.

| $i$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ | $J_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $p_{1,i}$ | 5 | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 |
| $p_{2,i}$ | 3 | 4 | 6 | 1 | 5 | 3 | 2 | 4 | 5 |

and the corresponding oriented multigraph



# Complexity of $F2|$ no-idle, no-wait $|C_{\max}$

**Proposition**

*Problem* $F2|no - idle, no - wait|C_{\max}$ *can be solved in* $O(n)$ *time.*

# Complexity of $F2|$ no-idle, no-wait $|C_{max}$

**Proposition**

*Problem* $F2|no-idle, no-wait|C_{max}$ *can be solved in* $O(n)$ *time.*

**Proof.**

**[Sketch]**: The generation of the oriented multigraph can be done in linear time and the graph has $O(n)$ arcs. Besides, it is known (Fleischner 1991) that computing an Eulerian path in an oriented graph with $n$ arcs can be done in $O(n)$ time. □

# $F2|$ no-idle, no-wait $|C_{\max}$ vs the Hamiltonian Path problem

- Problem $F2|no-idle, no-wait|C_{\max}$ is also linked to a special case of the Hamiltonian Path problem on a connected digraph.

# $F2|$ no-idle, no-wait $|C_{\max}$ vs the Hamiltonian Path problem

- Problem $F2|no-idle, no-wait|C_{\max}$ is also linked to a special case of the Hamiltonian Path problem on a connected digraph.
- Consider a digraph $G(V, A)$ that has the following property: $\forall v_i, v_j \in V$, either $S_i \cap S_j = \emptyset$, or $S_i = S_j$ where we denote by $S_i$ the set of successors of vertex $v_i$.

# $F2|$ no-idle, no-wait $|C_{\max}$ vs the Hamiltonian Path problem

- Problem $F2|no-idle, no-wait|C_{\max}$ is also linked to a special case of the Hamiltonian Path problem on a connected digraph.

- Consider a digraph $G(V, A)$ that has the following property: $\forall v_i, v_j \in V$, either $S_i \cap S_j = \emptyset$, or $S_i = S_j$ where we denote by $S_i$ the set of successors of vertex $v_i$.

- In other words, each pair of vertices either has no common successors or has all successors in common.

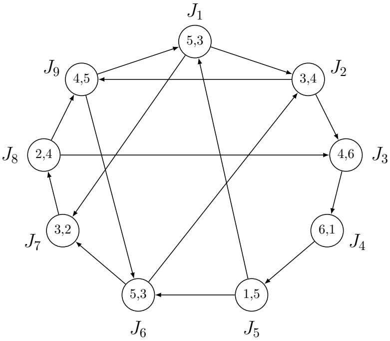
# $F2|$ no-idle, no-wait $|C_{\max}$ vs the Hamiltonian Path problem

- Problem $F2|no-idle, no-wait|C_{\max}$ is also linked to a special case of the Hamiltonian Path problem on a connected digraph.
- Consider a digraph $G(V, A)$ that has the following property: $\forall v_i, v_j \in V$, either $S_i \cap S_j = \emptyset$, or $S_i = S_j$ where we denote by $S_i$ the set of successors of vertex $v_i$.
- In other words, each pair of vertices either has no common successors or has all successors in common.
- We denote the Hamiltonian path problem in that graph as the Common/Distinct Successors Hamiltonian Oriented Path (CDSHOP*) problem.

# $F2|$ no-idle, no-wait $|C_{\max}$ vs the Hamiltonian Path problem

- $F2|no-idle, no-wait|C_{\max} \propto$ CDSHOP easily holds.
- The CDSHOP problem corresponding to the considered $F2|$ no-idle, no-wait $|C_{\max}$ instance.

| $i$ | $p_{1,i}$ | $p_{2,i}$ |
|---|---|---|
| $J_1$ | 5 | 3 |
| $J_2$ | 3 | 4 |
| $J_3$ | 4 | 6 |
| $J_4$ | 6 | 1 |
| $J_5$ | 1 | 5 |
| $J_6$ | 5 | 3 |
| $J_7$ | 3 | 2 |
| $J_8$ | 2 | 4 |
| $J_9$ | 4 | 5 |



# Complexity of CDSHOP

Proposition

$CDSHOP \propto F2|no-idle, no-wait|C_{\max}$, *hence,* $CDSHOP \in P$.

# Complexity of CDSHOP

## Proposition

$CDSHOP \propto F2|no-idle, no-wait|C_{\max}$, *hence,* $CDSHOP \in P$.

## Proof.

**[Sketch]:**

- For any instance of CDSHOP with $n$ vertices, we generate an instance of $F2|no-idle, no-wait|C_{\max}$ with $n$ jobs where, if there is an arc from $v_i$ to $v_j$, then, we have $p_{2,i} = p_{1,j}$.

# Complexity of CDSHOP

## Proposition

*$CDSHOP \propto F2|no-idle, no-wait|C_{\max}$, hence, $CDSHOP \in P$.*

## Proof.

**[Sketch]:**

- For any instance of CDSHOP with $n$ vertices, we generate an instance of $F2|no-idle, no-wait|C_{\max}$ with $n$ jobs where, if there is an arc from $v_i$ to $v_j$, then, we have $p_{2,i} = p_{1,j}$.
- If a feasible sequence of $F2|no-idle, no-wait|C_{\max}$ exists, then, for each consecutive jobs $J_i, J_j$ with $J_i \rightarrow J_j$, $p_{2,i} = p_{1,j}$ holds. Hence, there is an arc from $v_i$ to $v_j$. Thus, the corresponding sequence of vertices in CDSHOP constitutes an hamiltonian directed path.

# Complexity of CDSHOP

## Proposition

*CDSHOP* $\propto F2|no-idle, no-wait|C_{\max}$, *hence, CDSHOP* $\in P$.

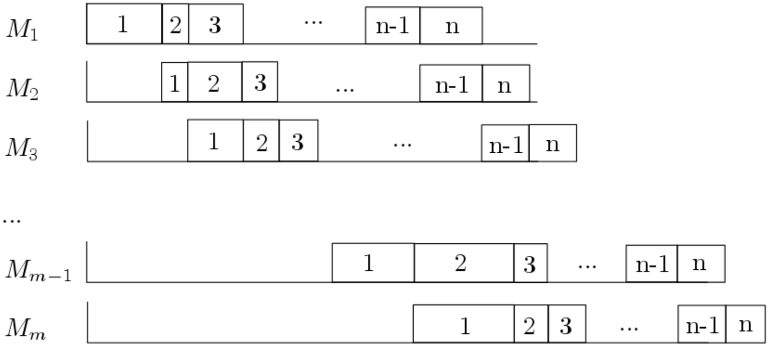
## Proof.

**[Sketch]:**

- For any instance of CDSHOP with $n$ vertices, we generate an instance of $F2|no-idle, no-wait|C_{\max}$ with $n$ jobs where, if there is an arc from $v_i$ to $v_j$, then, we have $p_{2,i} = p_{1,j}$.
- If a feasible sequence of $F2|no-idle, no-wait|C_{\max}$ exists, then, for each consecutive jobs $J_i, J_j$ with $J_i \rightarrow J_j$, $p_{2,i} = p_{1,j}$ holds. Hence, there is an arc from $v_i$ to $v_j$. Thus, the corresponding sequence of vertices in CDSHOP constitutes an hamiltonian directed path.
- Conversely, if a path exists for *CDSHOP*, the related sequence of jobs in $F2|no-idle, no-wait|C_{\max}$ is also feasible.

# Problem $F|$ no-idle, no-wait $|C_{max}$

The **no-idle, no-wait** constraint on $m$ machines.



# Problem $F|$ no-idle, no-wait $|C_{\max}$

## Lemma (3)

(C3) *A necessary condition to have a feasible solution for problem* $F|no-idle, no-wait|C_{\max}$ *is that there always exists an indexing of the jobs so that* $p_{j+1,1}, ..., p_{j+1,n-1}$ *and* $p_{j,2}, ..., p_{j,n}$*, for* $j = 1, ...m-1$*, constitute different permutations of the same vector of elements.*

# Problem $F|$ no-idle, no-wait $|C_{\max}$

## Lemma (3)

(C3) *A necessary condition to have a feasible solution for problem $F|no-idle, no-wait|C_{\max}$ is that there always exists an indexing of the jobs so that $p_{j+1,1}, \ldots, p_{j+1,n-1}$ and $p_{j,2}, \ldots, p_{j,n}$, for $j = 1, \ldots m-1$, constitute different permutations of the same vector of elements.*

(C4) *When the above condition (C3) holds, then*

Case 1 *if $(p_{1,1} \neq p_{2,n}$ or $p_{2,1} \neq p_{3,n}$ or … or $p_{m-1,1} \neq p_{m,n})$, every feasible sequence must have a job with processing times $(p_{1,1}, \ldots, p_{m-1,1})$ on machines $1$ to $(m-1)$ in first position and a job with processing time $(p_{2,n}, \ldots, p_{m,n})$ on machines 2 to $m$ in last position.*

Case 2 *if $(p_{1,1} = p_{2,n}$ and $p_{2,1} = p_{3,n}$ and … and $p_{m-1,1} = p_{m,n})$ and there exists a feasible sequence, then there do exist at least $n$ feasible sequences each starting with a different job by simply rotating the starting sequence as in a cycle.*

# Problem $F|$ no-idle, no-wait $|C_{\max}$

- We can evince that in an optimal sequence, if job $J_i$ immediately precedes job $J_k$, we have that $p_{j+1,i} = p_{j,k}$, $\forall j = 1, ..., m-1$ holds.

- Then, for a feasible 3-job subsequence $(\ell, i, k)$ we must have:
    1. $[p_{2,\ell}; ...; p_{m,\ell}] = [p_{1,i}; ...; p_{m-1,i}]$ and,
    2. $[p_{2,i}; ...; p_{m,i}] = [p_{1,k}; ...; p_{m-1,k}]$.

This can be represented in terms of vectorial dominoes as follows.

| | | | | | |
|---|---|---|---|---|---|
| $[p_{1,\ell}; ...; p_{m-1,\ell}]$ | $[p_{2,\ell}; ...; p_{m,\ell}]$ | $[p_{1,i}; ...; p_{m-1,i}]$ | $[p_{2,i}; ...; p_{m,i}]$ | $[p_{1,k}; ...; p_{m-1,k}]$ | $[p_{2,k}; ...; p_{m,k}]$ |
| domino 1 | | domino 2 | | domino 3 | |

# Problem $F|$ no-idle, no-wait $|C_{\max}$: an example

As an example, a 3-job instance on 4 machines of problem $F|no-idle, no-wait|C_{\max}$.

| $i$ | $J_1$ | $J_2$ | $J_3$ |
|---|---|---|---|
| $p_{1,i}$ | 5 | 3 | 4 |
| $p_{2,i}$ | 3 | 4 | 9 |
| $p_{3,i}$ | 4 | 9 | 7 |
| $p_{4,i}$ | 9 | 7 | 2 |

induces the following vectorial dominoes

domino 1 $\rightarrow$

| | | | | | |
|---|---|---|---|---|---|
| 5 | 3 | 4 | 3 | 4 | 9 |

domino 2 $\rightarrow$

| | | | | | |
|---|---|---|---|---|---|
| 3 | 4 | 9 | 4 | 9 | 7 |

domino 3 $\rightarrow$

| | | | | | |
|---|---|---|---|---|---|
| 4 | 9 | 7 | 9 | 7 | 2 |

# Problem $F|$ no-idle, no-wait $|C_{\max}$

## Proposition

*Problem $F|no-idle, no-wait|C_{\max}$ can be solved to optimality in $O(mnlog(n))$ time.*

# Problem $F|$ no-idle, no-wait $|C_{max}$

## Proposition

*Problem* $F|no-idle, no-wait|C_{\max}$ *can be solved to optimality in* $O(mnlog(n))$ *time.*

## Proof.

**[Sketch]:**

The result can be proved by showing that any instance of the $F|no-idle, no-wait|C_{\max}$ problem can be reduced in polynomial time to a vectorial OSPD that is always solved by computing an Eulerian path in an oriented graph with $n$ arcs. □

# Complexity of problems $(J2, O2)|$ no-idle, no-wait $|C_{\max}$

- Job-shop (J) problem: operations of a job totally ordered
- Open-shop (O) problem: no ordering constraints on operations

### Proposition

*Problems* $J2|no-idle, no-wait|C_{\max}$ *and* $O2|no-idle, no-wait|C_{\max}$ *are NP-hard in the strong sense.*

# Complexity of problems $(J2, O2)|$ no-idle, no-wait $|C_{\max}$

- Job-shop (J) problem: operations of a job totally ordered
- Open-shop (O) problem: no ordering constraints on operations

Proposition

*Problems $J2|no-idle, no-wait|C_{\max}$ and $O2|no-idle, no-wait|C_{\max}$ are NP-hard in the strong sense.*

Proof.

**[Sketch of proof for problem $J2|no-idle, no-wait|C_{\max}$]:**
We show that NMTS (Numerical Matching with Target Sums) reduces to $J2|no-idle, no-wait|C_{\max}$. □

Thank You.