Scheduling Theory: Cycle Blocking Ordered Flow Shop

by

Richard J. Caron†, Esaignani Selvarajh‡‡, and Jimmy Troung*

†† Department of Mathematics and Statistics, University of Windsor,
Windsor, ON N9B 3P4
‡‡ Odette School of Business, University of Windsor
Windsor, ON N9B 3P4

WMSR # 11-02
May 20, 2011
Scheduling Theory: Cycle Blocking Ordered Flow Shop

Dr. Caron, Dr. Selvarajah, Jimmy Truong
Department of Mathematics and Statistics. University of Windsor.

May 20, 2011

Abstract: By imposing a machine order on the Ordered Blocking Flow Shop problem, a new problem arises called Cycle Blocking. We present results for the Cycle Blocking problem for systems with \( n \) jobs and 3 machines. These results are the precursors to finding algorithms that search for schedules with minimal makespan.

Acknowledgments: This report documents the work was completed by Jimmy Truong in the summer of 2010 with the support of the NSERC USRA program and Dr. Selvarajah’s NESRC Discovery Grant.

1 Introduction.

We first describe the general problem known as Permutation Flow Shop and introduce conditions which make it a Blocking Flowing Shop problem. Furthermore, we introduce yet more conditions which make the problem an Ordered Blocking Flow Shop problem and lastly we impose a final condition which makes it a Cycle Blocking problem.

Starting at time zero, \( n \) jobs must be processed, in the same order, on each of the \( m \) machines. Each job is processed by machine 1, in order, to machine \( m \). Each machine may only process a single job at a time. The processing time of job \( k \) on machine \( j \) is denoted by \( p_{j,k} \), where \( j \in \{1, 2, 3, \ldots, m\} \) and \( k \in \{1, 2, 3, \ldots, n\} \). Setup times are included in the processing time. These times are fixed, known in advance and non-negative. The objective is to minimize makespan, the time required to process all jobs on a all machines.

Given a permutation, \( P \), of the \( n \) jobs, \([k]\) indicates the job that occupies position \( k \) in the sequence. For example, if \( P = (2, 3, 1) \), then \([1] = 2, [2] = 3, [3] = 1 \). For this permutation, on every machine, job 3 is the second job to be processed.

A schedule is said to be feasible if at all times, no more than one job is being processed by each of the machines and each machine processes the jobs in the same order. Note that each feasible schedule is associated with a permutation, by considering the order of the jobs processed on any of the machines. Also note that two feasible schedules associated with the same permutation may not have the same makespan. This is because having idle time does not change the order of the jobs.
In a feasible schedule associated to a permutation, let $e_{j,k}$ be the time at which machine $j$ receives the job $[k]$ and $f_{j,k}$ the time machine $j$ releases job $[k]$. The time at which the last job $[n]$ is released from the last machine, machine $m$, is referred to as maximum completion time or $C_{\text{max}}$. With the conditions stated, this is a Permutation Flow Shop and its $C_{\text{max}}$ problem can be formalized as follows:

A job may only be released after it has been completely processed. Although, before, during and after the processing of a job, it may spend time idling on a machine.

\[ e_{j,k} + p_{j,[k]} \leq f_{j,k} \quad j = 1, 2, 3, ..., m \quad k = 1, 2, 3, ..., n \]  

(1)

A job may only be received by the next machine if the previous job on that machine has been released.

\[ e_{j,k} \geq f_{j,k-1} \quad j = 1, 2, 3, ..., m \quad k = 1, 2, 3, ..., n \]  

(2)

A job may only be received by the next machines if job has been released by the previous machine.

\[ e_{j,k} \geq f_{j-1,k} \quad j = 1, 2, 3, ..., m \quad k = 1, 2, 3, ..., n \]  

(3)

When the last job from the last machine is released, the schedule is complete and the processing for each job is complete. The maximum completion time and the release time of the last job from the last machine are the same.

\[ C_{\text{max}} = f_{m,n} \]  

(4)

The initial conditions are $f_{j,0} = 0$, $\forall j$ and $f_{0,k} = 0$, $\forall k$.

The schedule is said to be semi-active if equation (1) satisfies equality, that is $e_{j,k} + p_{j,[k]} = f_{j,k}$, $\forall i, j$ and equations (2) and (3) are superseded by $e_{j,k} = \max \{ f_{j,k-1}, f_{j-1,k} \}$

When there is no storage space between stages, if job $k$ finishes its operation on machine $j$ and if machine $j + 1$ is still busy with a job, the completed job $k$ has to remain on machine $j$ blocking it. This is considered Blocking Flow Shop and requires an additional equation to summarize the previously described condition.

A job must wait on a machine until the job on the next machine is released.

\[ f_{j,k} \geq f_{j+1,k-1} \quad j = 1, 2, 3, ..., m \quad k = 1, 2, 3, ..., n \]  

(5)
Since we want jobs to have the ability to be released from the last machine, the initial condition \( f_{m+1,k} = 0 \) \( k = 1, 2, 3..., n \) must be added.

A schedule of a blocking flow shop is said to be semi-active if equation (??) and equation (??) are superseded by:

\[
f_{j,k} = \max\{e_{j,k} + p_{j,k}; f_{j+1,k-1}\}
\]

Note that a schedule is semi-active if no other schedule associated with the same permutation has a smaller makespan. A schedule can be made semi-active by removing as much idle time as possible, that is, to receive, process and release jobs as soon as it is possible.

1.1 Problem Description.

This work deals with a special case of Blocking Flow Shop, when there are \( m = 3 \) machines and for a given permutation of \( n \) jobs, a job occupying a machine may not be released except until the operations on all 3 machines are complete, at which time all jobs able to move to their next machine must. The conditions imply that each permutation is associated with only one schedule. Consequently, we refer to a permutation of jobs as a schedule.

For any schedule of \( n \) jobs, the jobs are released and received by the machines in three. Each time a triplet of jobs is released and received, a cycle occurs. Each cycle is identified by the jobs on each machine. For this reason we introduce notation to indicate which job is on each of the 3 machines. Given a schedule, \( T \), with job \( a \) followed by job \( b \) then by job \( c \) somewhere in the schedule, denoted \( T = (..., a, b, c, ..) \), then there exist some cycle with job \( a \) on machine 3, job \( b \) on machine 2 and job \( c \) on machine 1. The cycle time of such a cycle from schedule \( T \) is denoted \( c_{T(a,b,c)} = \max\{p_{3,a}, p_{2,b}, p_{1,a}\} \), is the time spent by jobs \( a, b \) and \( c \) on machines \( 1, 2 \) and \( 3 \) respectively. When a machine has no job or the null job occupying it, 0 is written for the job number in the cycle notation. For example we can consider the first cycle of any schedule, \( c_{T(0,0,1)} \) denotes the cycle time of the cycle which has the null job on machine 3 and the null job on machine 2 and the first job \( [1] \) on machine 1 from schedule \( T \). For the last 2 cycles of any schedule, the null job will occupy machine 1, and we omit the job number for the null job. For example, consider the second last cycle, \( c_{T([n-1],[n])} \) is understood as the cycle which has the job in position \( n - 1 \) on machine 3 and the job in position \( n \) on machine 2 and the null job on machine 1. \( c_{(0,0,0)} \) denotes the null cycle which is defined as a cycle that has only null jobs on each machine or is a non-existing cycle. Since jobs must move together in cycles, this problem is considered Cycle Blocking Flow Shop or Cycle Blocking for short and can be formalized for \( m \) machines by the following additional equations:

Let \( c_{j,[k+j-1]} = \max\{p_{m,[k-(m-j)]}, p_{m-1,k-(m-j+1)}, p_{m-2,k-(m-j+2)}, \cdots, p_{j+1,[k-1]}, p_{j,[k]}, p_{j-1,[k+1]}, \cdots, p_{3,[k+j-3]}, p_{2,[k+j-2]}, p_{1,[k+j-1]}\} \)
A job is released from its machine exactly after every machine is finished processing its current job.

\[ f_{j,[k]} = e_{j,[k]} + c_{j,[k+j-1]} \quad j = 1, 2, 3, ..., m \quad k = 1, 2, 3, ..., n \] (7)

A job is received by the next machine as soon as it is released.

\[ e_{j,[k]} = f_{j,[k-1]} \quad j = 1, 2, 3, ..., m \quad k = 1, 2, 3, ..., n \] (8)

With \( f_{1,0} = 0 \) being the initial conditions, and when a processing time, \( p_{j,k} \), from \( c_{j,[k+j-1]} \) has \( j \neq 1, 2, 3, ..., m \), or \( k \neq 1, 2, 3, ..., n \), then \( p_{j,k} = 0 \). The objective is minimize \( f_{m,[n]} \), the makespan.

## 2 Ordered Cycle Blocking Flow Shop: Machine Order (123)

### 2.1 Conditions

Here we impose conditions that give an ordering to the jobs and an ordering to the machines. In this case, the machine order is (123). Ordering the jobs and ordering the machines can be formally stated as:

For any job \( i \), the processing time increases as the machine number increase.

\[ p_{j,i} \geq p_{k,i}, \quad \forall \ j > k \] (9)

For any machine \( m \), the processing time of jobs increase as job numbers increase.

\[ p_{j,i} \geq p_{j,l}, \quad \forall \ i > l \] (10)

When the machine order is (123), we say that machine 1 is dominated by machine 2 who is dominated by machine 3, or we say machine 3 dominates machine 2 who dominates machine 1. When the jobs are ordered, the problem is called Ordered Cycle Blocking Flow Shop or Ordered
Cycle Blocking for short.

From conditions (??) and (??), it follows directly that:
\[ p_{j,i} \geq p_{k,l}, \; \forall j > k \text{ and } \forall i > l \] (11)

This condition is equivalent to (??) and (??).

**Proof.** Since \( p_{j,i} \geq p_{k,i} \forall j > k \) by (??) and \( p_{k,i} \geq p_{k,l} \forall i > l \) by (??). Then \( p_{j,i} \geq p_{k,i} \geq p_{k,l} \forall j > k \text{ and } \forall i > l \). Therefore, \( p_{j,i} \geq p_{k,l} \forall j > k \text{ and } \forall i > l \) as required.

\[ \square \]

## 2.2 Techniques

Since schedules are essentials just sequences of jobs, it will be helpful to examine a few techniques that allow us to manipulate sequences. While there exist many ways to construct, reconstruct, sort and compare sequences, such as insertions, removals, pairwise interchanges, sorting algorithms. All of these techniques can be applied to a sequence of jobs, but only insertions and pairwise interchanges will be used in the following proofs.

### 2.2.1 Pairwise Interchange

Given a schedule, \( S = (..., [k-1], a, [k+1], ..., [l-1], b, [l+1], ...), \) of \( n \) jobs, with job \( a \) in position \( k \) and job \( b \) in position \( l \), we say a pairwise interchange has been performed on job \( a \) and job \( b \) if the resulting schedule, \( S' \), has job \( a \) in position \( l \), job \( b \) in position \( k \) and all other jobs remain unchanged, that is \( S' = (..., [k-1], b, [k+1], ..., [l-1], a, [l+1], ...). \)

We first investigate the effects on cycle time after a pairwise interchange of any 2 jobs from a given schedule has been performed. We examine \( S \) and \( S' \). Since \( S' \) is a permutation of \( S \), we adopt, from combinatorial mathematics, cycle notation to denote the permutation. Let \( \sigma = (a \; b) \), then \( \sigma \) is that action which takes job \( a \) to job \( b \) and job \( b \) to job \( a \). When \( \sigma \) is applied to \( S \), we write \( \sigma(S) = S' \). For permutations involving more than 2 jobs, say \( k \) jobs, we have \( \tau = (a_1 \; a_2 \; ... \; a_k) \) denoting the action which takes job \( a_1 \) to \( a_2 \), \( a_2 \) to \( a_3 \), ..., \( a_{k-1} \) to \( a_k \) and lastly \( a_k \) to \( a_1 \).

We see that there are 6 relevant cycles from each schedule to consider. \( c_S^{[[k-2],[k-1],a]} \) becomes \( c_{S'}^{[[k-2],[k-1],a]} \). We denote a change in jobs of a cycle by an arrow. For the 6 relevant cycles we have:
These 6 cycle changes may not all be unique depending on the positions of job $a$ and job $b$. 

\[
\begin{align*}
C_{\{k-2,k-1,a\}} & \rightarrow C_{\{k-2,k-1,b\}} \\
C_{\{k-1,a,k+1\}} & \rightarrow C_{\{k-1,b,k+1\}} \\
C_{\{a,k+1,k+2\}} & \rightarrow C_{\{b,k+1,k+2\}} \\
C_{\{l-2,l-1,b\}} & \rightarrow C_{\{l-2,l-1,a\}} \\
C_{\{l-1,a,l+1\}} & \rightarrow C_{\{l-1,a,l+1\}} \\
C_{\{b,l+1,l+2\}} & \rightarrow C_{\{a,l+1,l+2\}} 
\end{align*}
\]
Lemma 2.1. Given \( n \) ordered jobs and 3 machines, with machine order \((123)\), let \( S = (\ldots, a, n-1, b \neq n-2, n, c, \ldots) \) be a schedule of the \( n \) jobs and let \( S' = \sigma(S) \) where \( \sigma = (b \ n-2) \), then \( c_{max}^{S'} \leq c_{max}^{S} \).

Proof. Depending on the position job \( n-2 \) relative to job \( b \neq n-2 \), \( S \) may take 2 forms, either \( S = (\ldots, d, e, n-2, f, g, \ldots, a, n-1, b \neq n-2, n, c, \ldots) \) or \( S = (\ldots, a, n-1, b \neq n-2, n, c, \ldots, d, e, n-2, f, g, \ldots) \). We assume the former and later see that it did not matter. The cycle changes that occur due to \( \sigma \) are:

\[
\begin{align*}
&c_{(d,e,n-2)}^S \rightarrow c_{(d,e,b)}^{S'} \\
&c_{(e,n-2,f)}^S \rightarrow c_{(e,b,f)}^{S'} \\
&c_{(n-2,f,g)}^S \rightarrow c_{(b,f,g)}^{S'} \\
&c_{(a,n-1,b)}^S \rightarrow c_{(a,n-1,n-2)}^{S'} \\
&c_{(n-1,b,n)}^S \rightarrow c_{(n-1,n-2,n)}^{S'} \\
&c_{(b,n,c)}^S \rightarrow c_{(n-2,n,c)}^{S'}
\end{align*}
\]

From here there are two cases, \( p_{2,n} \geq p_{3,n-1} \) or \( p_{2,n} < p_{3,n-1} \).

Case 1. \( p_{2,n} \geq p_{3,n-1} \).

All cycle changes result in non-increasing cycle times. In fact, \( c_{(d,e,b)}^{S'} = \max\{p_{3,d}, p_{2,e}, p_{1,b}\} \leq c_{(d,e,n-2)}^S = \max\{p_{3,d}, p_{2,e}, p_{1,n-2}\} \) since \( b < n-2 \) and by (??) \( p_{1,b} \leq p_{1,n-2} \). Similarly, \( c_{(e,b,f)}^{S'} = \max\{p_{3,e}, p_{2,b}, p_{1,f}\} \leq c_{(e,n-2,f)}^S = \max\{p_{3,e}, p_{2,n-2}, p_{1,f}\} \) and \( c_{(b,f,g)}^{S'} = \max\{p_{3,b}, p_{2,f}, p_{1,g}\} \leq c_{(n-2,f,g)}^S = \max\{p_{3,b}, p_{2,n-2}, p_{1,g}\} \) since \( b < n-2 < n-1 \) and by (??) \( p_{1,b} \leq p_{1,n-2} \leq p_{2,n-1} \). Similarly, \( c_{(a,n-1,n-2)}^{S'} = \max\{p_{3,a}, p_{2,n-1}\} = c_{(a,n-1,b)}^S \) since \( b < n-2 < n-1 \) and by (??) \( p_{1,b} \leq p_{1,n-2} \leq p_{2,n-1} \). Lastly, \( c_{(n-1,b,n)}^{S'} = p_{2,n} = c_{(b,n,c)}^S \) since \( c < n \) and by (??) \( p_{1,c} \leq p_{2,n} \) and by our assumption \( p_{2,n} \geq p_{3,n-1} \).

Case 2. \( p_{2,n} < p_{3,n-1} \).

Here we see that \( c_{(n-2,n,c)}^{S'} = p_{3,n-2} \geq c_{(b,n,c)}^S \) since \( b < n-2 \) and by (??) \( p_{3,b} \leq p_{3,n-2} \) and by our assumption \( p_{2,n} < p_{3,n-2} \). Not all cycle times are non-increasing as before. Here, instead we show that the net change in cycle time is non-increasing, by comparing \( c_{(n-2,n,c)}^{S'} \) to \( c_{(n-2,f,g)}^S \), leaving us to compare \( c_{(b,f,g)}^{S'} \) to \( c_{(b,n,c)}^S \). All other cycles need not be considered since they will behave the same as in case 1. With this said, \( c_{(n-2,n,c)}^{S'} = p_{3,n-2} = c_{(n-2,f,g)}^S \) since \( f < n \) and \( g < n \) by (??) and our assumption \( p_{2,f} \leq p_{2,n} < p_{3,n-1} \) and \( p_{1,g} \leq p_{2,n} < p_{3,n-1} \). Lastly, \( c_{(b,f,g)}^S = \max\{p_{3,b}, p_{2,f}, p_{1,g}\} \leq c_{(b,n,c)}^S = \max\{p_{1,b}, p_{2,n}\} \) since \( f < n \) and \( g < n \) by (??) \( p_{2,f} \leq p_{2,n} \) and \( p_{1,g} \leq p_{2,n} \).

Therefore, from case 1 and case 2, \( c_{max}^{S'} \leq c_{max}^{S} \) as required.
We note that the 6 cycles being not necessarily unique does not affect the proof. We also note that jobs $d, e,$ and $c$ all may be the null job, a job with processing times of 0 on all machines or non-existing job, but this does not affect the proof either. Lastly, we go back to our first assumption, that $S = (\ldots, d, e, n − 2, f, g, \ldots, a, n − 1, b \neq n − 2, n, c, \ldots)$ and see that the proof is independent of the form that schedule $S$ takes. The only thing this affects is which jobs may possibly be null jobs. This completes the proof.

We can further improve the previous lemma by performing another pairwise interchange of jobs and also show that the resulting schedule will have a makespan equal to or smaller than the makespan of the original schedule.

**Lemma 2.2.** Given $n$ ordered jobs and 3 machines, with machine order $(123)$, let $S = (\ldots, a, b, n − 1, n − 2, n, c, \ldots)$ be a schedule of $n$ jobs and let $S' = \sigma(S) = (\ldots, a, b, n − 2, n − 1, n, c, \ldots)$ where $\sigma = (n − 1 n − 2)$, then $c_{\text{max}}^S \leq c_{\text{max}}^{S'}$.

**Proof.** Since the jobs undergoing the pairwise interchange are adjacent, there will only be 4 relevant cycles to consider. The cycle changes that occur due to $\sigma$ are:

- $c_{(a,b,n-1)}^S \rightarrow c_{(a,b,n-2)}^{S'}$
- $c_{(b,n-1,n-2)}^S \rightarrow c_{(b,n-2,n-1)}^{S'}$
- $c_{(n-1,n-2,n-2)}^S \rightarrow c_{(n-2,n-1,n)}^{S'}$
- $c_{(n-2,n-1,c)}^S \rightarrow c_{(n-1,n,c)}^{S'}$

From here there are 2 cases: $p_{2,n} \geq p_{3,n-1}$ or $p_{2,n} < p_{3,n-1}$.

**Case 1.** $p_{2,n} \geq p_{3,n-1}$.

All cycle times are not increased as a result of $\sigma$. In fact, $c_{(a,b,n-2)}^{S'} = \max\{p_{3,a}, p_{2,b}, p_{1,n-2}\} \leq c_{(a,b,n-1)}^S = \max\{p_{3,a}, p_{2,b}, p_{1,n-1}\}$ since $n − 2 < n − 1$ and by (??) $p_{1,n-2} < p_{1,n-1}$. We see that $c_{(b,n-2,n-1)}^{S'} = \max\{p_{3,b}, p_{2,n-2}, p_{1,n-1}\} \leq c_{(b,n-1,n-2)}^S = \max\{p_{3,b}, p_{2,n-1}, p_{1,n-2}\}$ since $n − 2 < n − 1$ and by (??) $p_{2,n-2} < p_{2,n-1}$ and $p_{1,n-1} < p_{2,n-1}$. Similarly, $c_{(n-1,n-2,n)}^{S'} \leq c_{(n-2,n-1,n)}^S$. Lastly, $c_{(n-1,n,c)}^{S'} = p_{2,n} = c_{(n-2,n,c)}^S$ since $p_{2,n} \geq p_{3,n-1} \geq p_{3,n-2}$ by our assumption.

**Case 2.** $p_{2,n} < p_{3,n-1}$.

Here we show the net change in cycle times is non-positive. We compare $p_{2,n} < p_{3,n-1}$ to $c_{(n-1,n-2,n)}^S$ leaving us to compare $c_{(n-2,n-1,n)}^{S'}$ to $c_{(n-2,n,c)}^S$. For the first comparison, $p_{2,n} < p_{3,n-1} = p_{3,n-1} = c_{(n-1,n-2,n)}^S$ since $c < n$ and by (??) and by our assumption $p_{1,c} < p_{2,n} < p_{3,n-1}$. For the second comparison, $c_{(n-2,n-1,n)}^{S'} = \max\{p_{3,n-2}, p_{2,n-1}, p_{1,n}\} \leq c_{(n-2,n,c)}^S = \max\{p_{3,n-2}, p_{2,n}\}$ since by (??) $p_{2,n-1} \leq p_{2,n}$ and $p_{1,n} \leq p_{2,n}$.

Therefore, from case 1 and case 2, $c_{\text{max}}^{S'} \leq c_{\text{max}}^S$ as required.
2.2.2 Insertion

Sometimes a schedule is already predetermined and a new job is introduced to the job set and what is needed is for the new job to be inserted somewhere into the preexisting schedule. The questions under consideration are how does inserting a new job into an existing schedule affect cycle times and makespan and where in the schedule should the new job be inserted to minimize the increase in makespan.

We will see that inserting a new job into an existing schedule will create 3 cycles where there was 2 cycles. For the following lemma, we will see that inserting a new job, under certain conditions, will possibly increase the cycle time of 2 cycles and add another entire cycle.

Lemma 2.3. Given \( n \) ordered jobs and 3 machines, with machine order \((123)\), let \( S = (..., a, b, c, d, ...,) \) be a schedule of the \( n \) jobs, let job \( n + 1 \) be a new job which satisfy \( p_{j,i} \geq p_{k,l}, \forall j > k \) and \( \forall i > l \) when added to the current job set, and let \( T = (..., a, b, n + 1, c, d, ...,) \), the resulting schedule when job \( n + 1 \) is inserted in schedule \( S \) between 2 arbitrary jobs, then \( c_{a,b,c} < c_{a,b,n+1} \) \( c_{b,c,d} < c_{b,n+1,c} \), and \( c_{n+1,c,d} = p_{3,n+1} \) is the cycle time associated with the added cycle.

Proof. The cycle changes that occur due to the insertions of job \( n + 1 \) are:

\[
\begin{align*}
  c_{a,b,c}^S & \rightarrow c_{a,b,n+1}^T \\
  c_{b,c,d}^S & \rightarrow c_{b,n+1,c}^T \\
  c_{0,0,0}^S & \rightarrow c_{n+1,c,d}^T
\end{align*}
\]

All other cycles remain unchanged. Also we use the null cycle to indicate a non-existing cycle has become an existing cycle.

First, \( c_{a,b,c}^S = \max \{ p_{3,a}, p_{2,b}, p_{1,c} \} \leq c_{a,b,n+1}^T = \max \{ p_{3,a}, p_{2,b}, p_{1,n+1} \} \) since \( c < n + 1 \) and by (??) \( p_{1,c} < p_{1,n+1} \). Secondly, \( c_{b,c,d}^S = \max \{ p_{3,b}, p_{2,c}, p_{1,d} \} \leq c_{b,n+1,c}^T = \max \{ p_{3,b}, p_{2,n+1}, p_{1,c} \} \) since \( c < n + 1 \) and \( d < n + 1 \) and by (??) \( p_{2,c} \leq p_{2,n+1} \) and \( p_{1,d} \leq p_{2,n+1} \). This leaves \( c_{n+1,c,d}^T = p_{3,n+1} \) as the cycle time associated with the added cycle. It follows directly that \( c_{n+1,c,d}^T = p_{3,n+1} \) from (??).

At this point we will note that depending on where job \( n + 1 \) is inserted into the schedule \( S \), jobs \( a, b, c, \) and \( d \) may or may not be null jobs. Once again having any of these jobs be the null job does not affect the proof. This completes the proof.

For \( n \) ordered jobs, we define \( S = (1, 2, 3, ..., n - 2, n - 1, n) \) to be the natural schedule. For the next theorem we will show that, under machine order \((123)\), the natural schedule has a minimum makespan. The proof is by induction and involves isolating the largest job, removing it from the job set, then inserting it into an arbitrary schedule of the remaining jobs. We will attempt to show that inserting the largest job so that it follows the next 2 largest jobs in the schedule will give the least
amount of increase in makespan. However, this will have to be treated in cases as 1 case requires special attention.

**Theorem 2.1.** Given n ordered jobs and 3 machines, with machine order (123), the natural schedule has a minimum makespan.

**Proof.** For a job set containing 1 job, the natural schedule is the only schedule. Therefore it gives the minimum makespan. Assume that for n ordered jobs the natural schedule has a minimum makespan. Now for a job set containing n + 1 ordered jobs, we identify the largest job which is job n + 1 and remove it from the job set. Let S = (1, 2, 3, ..., n − 2, n − 1, n) be the natural schedule of remaining n jobs and let T = (1, 2, 3, ..., n − 2, n − 1, n, n + 1) be the resulting schedule when job n + 1 is inserted as the last job in schedule S. Also fix another schedule U = (..., a, b, c, d, ...) of the remaining n jobs such that when job n + 1 is inserted between job b and job c the resulting schedule V = (..., a, b, n + 1, c, d, ...) gives a minimum makespan. We will proceed to show that schedule T and schedule V must be the same.

Suppose job a ≠ job n − 1 and job b ≠ job n. From here there are 2 cases either job a ≠ job n or job a = job n.

Case 1. job a ≠ job n.

The cycle changes due to the insertion of job n + 1 in schedule U and in schedule S are as followed:

\[
\begin{align*}
 c_{a,b,c}^U &\rightarrow c_{a,b,n+1}^V \\
 c_{b,c,d}^U &\rightarrow c_{b,n+1,c}^V \\
 c_{0,0,0}^U &\rightarrow c_{n+1,c,d}^V \\
 c_{n-1,n}^S &\rightarrow c_{n-1,n,n+1}^T \\
 c_n^S &\rightarrow c_{n,n+1}^T \\
 c_{0,0,0}^V &\rightarrow c_{n+1}^T
\end{align*}
\]

We now compare the increase of makespan from schedule S to schedule T to the increase of makespan from schedule U to schedule V.

First we see that \( c_{n-1,n,n+1}^T - c_{n-1,n}^S \leq c_{a,b,n+1}^V - c_{a,b,c}^U \). In fact, since \( c_{n-1,n,n+1}^T = \max\{p_{3,n-1}, p_{2,n}, p_{1,n+1}\} \), if \( c_{n-1,n,n+1}^T = p_{3,n-1} \) then \( c_{n-1,n}^S = p_{3,n-1} \) so \( c_{n-1,n,n+1}^T - c_{n-1,n}^S = 0 \) and by lemma ??, \( c_{a,b,n+1}^V - c_{a,b,c}^U \geq 0 \). If \( c_{n-1,n,n+1}^T = p_{2,n} \), then \( c_{n-1,n}^S = p_{2,n} \) so by the same argument the above holds true. If \( c_{n-1,n,n+1}^T = p_{1,n+1} \), then \( c_{a,b,n+1}^V = p_{1,n+1} \) since \( a < n - 1 \) and \( b < n \) by (??) \( p_{3,a} \leq p_{3,n-1} \leq p_{1,n+1} \) and \( p_{2,b} \leq p_{2,n} \leq p_{1,n+1} \). But also since \( a < n - 1 \), \( b < n \) and \( c \leq n \) by (??) we have \( c_{a,b,c}^U \leq c_{n-1,n}^S \), so \( c_{n-1,n,n+1}^T - c_{n-1,n}^S \leq c_{a,b,n+1}^V - c_{a,b,c}^U \) holds. In a similarly manner, we get \( c_{n+1}^T - c_{n}^S \leq c_{a,b,n+1}^V - c_{a,b,c}^U \). Lastly, \( c_{n+1}^T - c_{0,0,0}^S = c_{a,b,n+1}^V - c_{0,0,0}^U \) since \( c_{0,0,0}^S = c_{0,0,0}^U \) = 0 by (??) \( c_{n+1}^T = c_{n+1,c,d}^V = p_{3,n+1} \).

Now we know that the increase of makespan from schedule S to schedule T is less than or equal to the increase of makespan from schedule U to schedule V, that is \( c_{max}^T - c_{max}^S \leq c_{max}^V - c_{max}^U \) and
by the inductive hypothesis, $c_{\text{max}}^S \leq c_{\text{max}}^U$. Therefore $c_{\text{max}}^T \leq c_{\text{max}}^V$, but this is a contradiction since schedule $V$ was assumed to have a minimum makespan.

Case 2. job $a = \text{job } n$.

We see that schedule $U = (\ldots, n, b, c, d, \ldots)$ and so schedule $V = (\ldots, n, n + 1, c, d, \ldots)$. If we assume job $b \neq \text{job } n - 1$ then by lemma (??) schedule $V$ does not have a minimum makespan, a contradiction. Therefore, assume job $b = \text{job } n - 1$, but now by lemma (??) schedule $V$ still does not have a minimum makespan, a contradiction.

Therefore, by case 1 and case 2, job $a = \text{job } n - 1$ and job $b = \text{job } n$. So we can conclude that schedule with a minimum makespan has the property that job $n$ followed job $n - 1$ and preceded job $n + 1$. This does not yet show that the natural schedule has a minimum makespan as any schedule $V = (\ldots, n - 1, n, n + 1, c, d, \ldots)$ with this form is still a valid candidate. To this end, we see the cycle changes due to the insertion of job $n + 1$ would be:

$$c_{\{n-1,n,c\}}^U \rightarrow c_{\{n-1,n,n+1\}}^V$$
$$c_{\{n,c,d\}}^U \rightarrow c_{\{n,n+1,c\}}^V$$
$$c_{\{0,0,0\}}^U \rightarrow c_{\{n+1,c,d\}}^V$$

We compare the increases in makespan here and see that they are the same. In fact, $c_{\{n-1,n,n+1\}}^V - c_{\{n-1,n\}}^S = c_{\{n-1,n,n+1\}}^U - c_{\{n-1,n,c\}}^U$ since $c < n$ and by (??) $p_{1,c} \leq p_{2,n}$ and so $c_{\{n-1,n\}}^S = c_{\{n-1,n,c\}}^U = \max\{p_{3,n-1},p_{2,n}\}$ and $c_{\{n-1,n,n+1\}}^V = c_{\{n-1,n,n+1\}}^U$ and so the above holds. In a similar manner, $c_{\{n,n+1\}}^T - c_{\{n\}}^S = c_{\{n,n+1,c\}}^V - c_{\{n,c,d\}}^U$ and $c_{\{n+1\}}^T - c_{\{0,0,0\}}^S = c_{\{n+1,c,d\}}^V - c_{\{0,0,0\}}^U = p_{3,n+1}$. So $c_{\text{max}}^T - c_{\text{max}}^S = c_{\text{max}}^U - c_{\text{max}}^V$ but by the inductive hypothesis $c_{\text{max}}^S \leq c_{\text{max}}^U$. Therefore $c_{\text{max}}^T \leq c_{\text{max}}^V$ as required.

\[\square\]

3 Ordered Cycle Blocking Flow Shop: Machine Order (132)

3.1 Conditions

In this case the machine order is (132). There are still $n$ ordered jobs and 3 machines. The conditions are formally stated as:

$$p_{2,i} \geq p_{3,i} \geq p_{1,i} \quad \text{for } i = 1, 2, 3, \ldots, n$$
(12)

$$p_{j,i} \geq p_{j,l}, \quad \forall \ l > i$$
(13)
It turns out that under the above conditions a schedule of \( n \) ordered jobs will have a minimum makespan only if the jobs in the schedule increase in job number up to the largest job then decrease in job number to the end of the schedule. We say the schedule takes on a pyramid shape. We prove this in 2 parts and both proofs are by induction.

**Proposition 3.1.** Given \( n \) ordered jobs, with machine ordering (132), if the largest job, job \( n \), must be the last job in the schedule, then the natural schedule has a minimum makespan.

**Proof.** For a job set containing 1 job, the natural schedule is the only schedule. Therefore it has a minimum makespan. Assume for \( n \) ordered jobs that if the largest job, job \( n \), must be the last job in the schedule, then the natural schedule has a minimum makespan. Now for a job set of \( n+1 \) ordered jobs we remove the second largest job, job \( n \), from the job set. When we remove job \( n \) from the job set, the resulting job set has \( n \) ordered jobs and this allows us to use inductive hypothesis if we fix the largest job, job \( n+1 \), as the last job in a schedule. Let \( S = (1, 2, 3, \ldots, n−3, n−2, n−1, n+1) \) be a schedule with increasing job numbers of the \( n \) jobs without job \( n \). Since job \( n+1 \) is the last job in the schedule, \( S \) may considered the natural schedule of its job set by renaming job \( n+1 \) if we please. Let \( T = (1, 2, 3, \ldots, n−2, n−1, n+1) \) be the resulting schedule when job \( n \) is inserted into schedule \( S \) between job \( n−1 \) and job \( n+1 \). Fix another schedule \( U = (\ldots, a, b, c, d, \ldots) \) with job \( n+1 \) as the last job of the \( n+1 \) jobs without job \( n \), such that when job \( n \) is inserted between job \( b \) and job \( c \) the resulting schedule \( V = (\ldots, a, b, n, c, d, \ldots) \) has a minimum makespan. We will proceed to show that schedule \( T \) and schedule \( V \) must be the same.

Suppose job \( c \neq \) job \( n+1 \). From here there are 2 cases either \( p_{2, n} \geq p_{1, n+1} \) or \( p_{2, n} < p_{1, n+1} \).

**Case 1.** \( p_{2, n} \geq p_{1, n+1} \).

The cycle changes that occur as a result of inserting job \( n \) into schedule \( S \) and schedule \( U \) are:

\[
\begin{align*}
& c^U_{(a,b,c)} \rightarrow c^V_{(a,b,n)} \\
& c^U_{(0,0,0)} \rightarrow c^V_{(b,n,c)} \\
& c^U_{(b,c,d)} \rightarrow c^V_{(n,c,b)} \\
& c^S_{(n−2,n−1,n+1)} \rightarrow c^T_{(n−2,n−1,n)} \\
& c^S_{(0,0,0)} \rightarrow c^T_{(n−1,n,n+1)} \\
& c^S_{(n−1,n+1)} \rightarrow c^T_{(n,n+1)}
\end{align*}
\]

We now compare the increase of makespan from schedule \( S \) to schedule \( T \) to the increase of makespan from schedule \( U \) to schedule \( V \).

- **First**, \( c^T_{(n−2,n−1,n)} − c^S_{(n−2,n−1,n+1)} \leq c^V_{(a,b,n)} − c^U_{(a,b,c)} \). In fact, \( c^T_{(n−2,n−1,n)} − c^S_{(n−2,n−1,n+1)} = \max\{p_{3,n−2}, p_{2,n−1}, p_{1,n}\} − \max\{p_{3,n−2}, p_{2,n−1}, p_{1,n+1}\} \leq 0 \) since \( p_{1,n} \leq p_{1,n+1} \) by (??). Also, \( c^V_{(a,b,n)} − c^U_{(a,b,c)} = \max\{p_{3,a}, p_{2,b}, p_{1,n}\} − \max\{p_{3,a}, p_{2,b}, p_{1,c}\} \geq 0 \) since \( c < n \) and by (??) \( p_{1,c} \leq p_{1,n} \). Secondly, \( c^T_{(n−1,n,n+1)} − c^S_{(0,0,0)} = c^V_{(b,n,c)} − c^U_{(0,0,0)} = p_{2,k} \) since \( p_{2,k} \geq p_{1,k+1} \geq p_{1,c} \) by our assumption and (??). Lastly, \( c^T_{(n,n+1)} − c^S_{(n−1,n+1)} \leq c^V_{(n,c,b)} − c^U_{(b,c,d)} \). In fact, \( c^T_{(n,n+1)} − c^S_{(n−1,n+1)} = 0 \) since \( b < n < n+1 \)

12
by (??) $p_{3,b} \leq p_{3,n-1} \leq p_{2,k}$ and since $c < n + 1$ by (??) and (??) $p_{1,c} \leq p_{2,k}$.

Now we know that the increase of makespan from schedule $S$ to schedule $T$ is less than or equal to the increase of makespan from schedule $U$ to schedule $V$, that is $c^T_{\text{max}} - c^S_{\text{max}} \leq c^V_{\text{max}} - c^U_{\text{max}}$ and by the inductive hypothesis, $c^S_{\text{max}} \leq c^U_{\text{max}}$. Therefore $c^T_{\text{max}} \leq c^V_{\text{max}}$, but this is a contradiction since schedule $V$ was assumed to have a minimum makespan.

Case 2. $p_{2,n} < p_{1,n+1}$.

Here we again compare the increase of makespan from schedule $S$ to schedule $T$ to the increase of makespan from schedule $U$ to schedule $V$. However, we compare the cycles differently. The cycle changes to be compared are:

$$
c^U_{\{a,b,c\}} \rightarrow c^V_{\{a,b,n\}} \quad \quad \quad c^S_{\{n-2,n-1,n+1\}} \rightarrow c^T_{\{n-1,n,n+1\}}
$$

$$
c^U_{\{0,0,0\}} \rightarrow c^V_{\{b,n,c\}} \quad \quad \quad c^S_{\{0,0,0\}} \rightarrow c^T_{\{n-2,n-1,n\}}
$$

$$
c^U_{\{b,c,d\}} \rightarrow c^V_{\{n,c,b\}} \quad \quad \quad c^S_{\{n-1,n+1\}} \rightarrow c^T_{\{n,n+1\}}
$$

First, $c^T_{\{n-1,n,n+1\}} - c^S_{\{n-2,n-1,n+1\}} \leq c^V_{\{a,b,n\}} - c^U_{\{a,b,c\}}$. In fact, $c^T_{\{n-1,n,n+1\}} - c^S_{\{n-2,n-1,n+1\}} = p_{1,n+1} - p_{1,n+1} = 0$ since $n - 1 < n$ by (??) and (??) and our assumption, $p_{3,n-2} \leq p_{3,n-1} \leq p_{2,n} \leq p_{1,n+1}$. From case 1 $c^V_{\{a,b,n\}} - c^T_{\{n-1,n,n+1\}} \leq 0$. Secondly, $c^T_{\{n-2,n-1,n\}} - c^S_{\{0,0,0\}} \leq c^V_{\{b,n,c\}} - c^U_{\{0,0,0\}}$ since $n - 2 < n - 1 < n$ by (??) and (??) $p_{3,n-2} \leq p_{2,n-1} \leq p_{2,n}$ and $p_{1,n} \leq p_{2,n}$. Lastly, $c^T_{\{n,n+1\}} - c^S_{\{n-1,n+1\}} \leq c^V_{\{n,c,b\}} - c^U_{\{b,c,d\}}$ as proved in case 1.

Again, $c^T_{\text{max}} - c^S_{\text{max}} \leq c^V_{\text{max}} - c^U_{\text{max}}$ and by the inductive hypothesis, $c^S_{\text{max}} \leq c^U_{\text{max}}$. Therefore $c^T_{\text{max}} \leq c^V_{\text{max}}$, but this is a contradiction since schedule $V$ was assumed to have a minimum makespan.

This shows that job $c = \text{job } n + 1$, that is, job $n$ must have preceded job $n + 1$ and since job $n$ is fixed as the second last job at the end of the schedule by the inductive hypothesis, the natural schedule has a minimum makespan. Although job $n$ is not the last job in schedule, we may ignore job $n + 1$ because the only cycle time it may affect is $c^T_{\{n-1,n,n+1\}}$ and only if $c^T_{\{n-1,n,n+1\}} = p_{1,n+1}$, but in this situation we change one of the processing times on job $n$ to one on job $n+1$, $p'_{2,n} = p_{1,n+1}$ as $p'_{1,n+1} = p_{2,n}$. After this change job $n + 1$ may be ignored and job $n$ may be considered the last job in the schedule and the inductive hypothesis may be use. This switch with the processing times is valid only because it does not change the cycle time, that is $c^T_{\{n-1,n,n+1\}} = p_{1,n+1}$ still, and we know this cycle will exist because job $n + 1$ was assumed to be the last job preceded by job $n$ was. This completes the proof.

We now prove the reverse of the natural schedule gives a minimum makespan when the largest job must be the first job. This will help prove that after the largest job a schedule with a minimum makespan must have jobs with decreasing job numbers. This is essentially the other half of the
pyramid shape. We prove this part in similar manner as done in the previous proof.

**Proposition 3.2.** Given \( n \) ordered jobs, with machine ordering (132), if the largest job, job \( n \), must be the first job in the schedule, then the natural schedule has a minimum makespan.

**Proof.** For a job set containing 1 job, the reverse of the natural schedule is the only schedule. Therefore it has a minimum makespan. Assume for \( n \) ordered jobs that if the largest job, job \( n \), must be the first job in the schedule, then the reverse of the natural schedule has a minimum makespan. Now for a job set of \( n+1 \) ordered jobs we remove the second largest job, job \( n \), from the job set. When we remove job \( n \) from the job set, the resulting job set has \( n \) ordered jobs and this allows us to use inductive hypothesis if we fix the largest job, job \( n+1 \), as the first job in a schedule. Let \( S = (n+1, n-1, n-2, \ldots, 3, 2, 1) \) be a schedule with decreasing job numbers of the \( n \) jobs without job \( n \). Since job \( n+1 \) is the first job in the schedule, \( S \) may be considered the reverse of the natural schedule of its job set by renaming job \( n+1 \) if we please. Let \( T = (n+1, n, n-1, n-2, \ldots, 3, 2, 1) \) be the resulting schedule when job \( n \) is inserted into schedule \( S \) between job \( n+1 \) and job \( n-1 \). Fix another schedule \( U = (\ldots, a, b, c, d, \ldots) \) with job \( n+1 \) as the first job of the \( n \) jobs without job \( n \), such that when job \( n \) is inserted between job \( b \) and job \( c \) the resulting schedule \( V = (\ldots, a, b, n, c, d, \ldots) \) has a minimum makespan. We will proceed to show that schedule \( T \) and schedule \( V \) must be the same.

Suppose job \( b \neq \text{job } n+1 \). From here there are 2 cases either \( p_{2,n} \geq p_{1,n+1} \) or \( p_{2,n} < p_{1,n+1} \).

**Case 1.** \( p_{2,n} \geq p_{3,n+1} \).

The cycle changes that occur as a result of inserting job \( n \) into schedule \( S \) and schedule \( U \) are:

\[
\begin{align*}
\cV_{\{a,b,c\}} &\rightarrow \cV_{\{a,b,n\}} \\
\cV_{\{0,0,0\}} &\rightarrow \cV_{\{b,n,c\}} \\
\cV_{\{b,c,d\}} &\rightarrow \cV_{\{n,c,b\}}
\end{align*}
\]

\[
\begin{align*}
\cS_{\{0,n+1,n-1\}} &\rightarrow \cS_{\{0,n+1,n\}} \\
\cS_{\{0,0,0\}} &\rightarrow \cS_{\{n+1,n-1\}} \\
\cS_{\{n+1,n-1,n-2\}} &\rightarrow \cS_{\{n-1,n-2\}}
\end{align*}
\]

We now compare the increase of makespan from schedule \( S \) to schedule \( T \) to the increase of makespan from schedule \( U \) to schedule \( V \).

First, \( \cS_{\{0,n+1,n-1\}} - \cS_{\{0,n+1,n-1\}} \leq \cV_{\{a,b,n\}} - \cV_{\{a,b,c\}} \). In fact, \( \cS_{\{0,n+1,n\}} - \cS_{\{0,n+1,n-1\}} \geq 0 \) since \( c < n \) and by (??) \( p_{1,c} \leq p_{1,n} \). Also, \( \cV_{\{a,b,n\}} - \cV_{\{a,b,c\}} = p_{2,n+1} - p_{2,n+1} = 0 \) since \( n-1 < n < n+1 \) and by (??) \( p_{1,n-1} \leq p_{1,n} \leq p_{2,n+1} \). Secondly, \( \cV_{\{n+1,n,n-1\}} - \cS_{\{0,0,0\}} \leq \cV_{\{b,n,c\}} - \cV_{\{0,0,0\}} \) since \( \cS_{\{n+1,n,n-1\}} = \cV_{\{b,n,c\}} = p_{2,n} \) since \( b < n, c < n \) and \( n-1 < n \) by (??) and (??) \( p_{3,b} \leq p_{2,n}, p_{1,c} \leq p_{2,n}, p_{1,n-1} \leq p_{2,n} \) and \( p_{3,n+1} \leq p_{3,n} \) by our assumption. Lastly, \( \cS_{\{n,n-1,n-2\}} - \cS_{\{n+1,n-1,n-2\}} \leq \cV_{\{n,c,b\}} - \cV_{\{b,c,d\}} \). In fact, \( \cS_{\{n,n-1,n-2\}} - \cS_{\{n+1,n-1,n-2\}} \leq 0 \) since \( n < n+1 \) \( p_{3,n} \leq p_{3,n+1} \) by (??). Also, \( \cV_{\{n,c,b\}} - \cV_{\{b,c,d\}} \geq 0 \) since \( b < n \) \( p_{3,b} \leq p_{3,n} \) by (??).

Now we know that the increase of makespan from schedule \( S \) to schedule \( T \) is less than or equal to the increase of makespan from schedule \( U \) to schedule \( V \), that is \( \cS_{max} - \cS_{max} \leq \cV_{max} - \cV_{max} \) and
by the inductive hypothesis, $c^S_{\max} \leq c^U_{\max}$. Therefore $c^T_{\max} \leq c^V_{\max}$, but this is a contradiction since schedule $V$ was assumed to have a minimum makespan.

Case 2. $p_{2,n} < p_{3,n+1}$.

Here we again compare the increase of makespan from schedule $S$ to schedule $T$ to the increase of makespan from schedule $U$ to schedule $V$. However, we compare the cycles differently. The cycle changes to be compared are:

\[
\begin{align*}
&c^U_{\{a,b,c\}} \longrightarrow c^V_{\{a,b,n\}} & c^S_{\{n-2,n-1,n+1\}} \longrightarrow c^T_{\{n-1,n,n+1\}} \\
&c^U_{\{0,0,0\}} \longrightarrow c^V_{\{b,n,c\}} & c^S_{\{0,0,0\}} \longrightarrow c^T_{\{n,n+1\}} \\
&c^U_{\{b,c,d\}} \longrightarrow c^V_{\{n,c,b\}} & c^S_{\{n-1,n+1\}} \longrightarrow c^T_{\{n-2,n-1,n\}}
\end{align*}
\]

First, $c^S_{\{n-2,n-1,n+1\}} - c^U_{\{n-1,n,n+1\}} \leq c^V_{\{a,b,n\}} - c^U_{\{a,b,c\}}$ as shown in case 1. Secondly, $c^T_{\{n,n+1\}} - c^S_{\{0,0,0\}} \leq c^V_{\{b,n,c\}} - c^U_{\{0,0,0\}}$ since $p_{3,n} \leq p_{2,n} \leq p_{2,n-1} \leq p_{2,n}$ and $p_{1,n-2} \leq p_{2,n}$ by (??) and (??). Lastly, $c^T_{\{n-2,n-1,n\}} - c^S_{\{n-1,n+1\}} \leq c^V_{\{n,c,b\}} - c^U_{\{b,c,d\}}$. In fact, $c^T_{\{n-2,n-1,n\}} - c^S_{\{n-1,n+1\}} = p_{3,n+1} - p_{3,n+1} = 0$ since $p_{1,n-2} \leq p_{1,n-1} \leq p_{2,n-1} \leq p_{2,n} \leq p_{3,n+1}$ by our assumption, (??) and (??).

Again, $c^T_{\max} - c^S_{\max} \leq c^V_{\max} - c^U_{\max}$ and by the inductive hypothesis, $c^S_{\max} \leq c^U_{\max}$. Therefore $c^T_{\max} \leq c^V_{\max}$, but this is a contradiction since schedule $V$ was assumed to have a minimum makespan.

This shows that job $b = \text{job } n+1$, that is, job $n$ must have followed job $n+1$ and since job $n$ is fixed as the second job at the beginning of the schedule by the inductive hypothesis, the natural schedule has a minimum makespan. Although job $n$ is not the first job in schedule, we may ignore job $n+1$ because the only cycle time it may effect is $c^T_{\{n+1,n,n-1\}}$ and only if $c^T_{\{n+1,n,n-1\}} = p_{3,n+1}$, but in this situation we change one of the processing times on job $n$ to one on job $n+1$, $p'_{3,n+1} = p_{3,n+1}$ as $p'_{3,n+1} = p_{2,n}$. After this change job $n+1$ may be ignored and job $n$ may be considered the last job in the schedule and the inductive hypothesis may be use. This switch with the processing times is valid only because it does not change the cycle time, that is $c^T_{\{n-1,n,n+1\}} = p_{1,n+1}$ still, and we know this cycle will exist because job $n+1$ was assumed to be the first job followed by job $n$ was . This completes the proof.

This solves the other half of the pyramid shape. The only concern is how does the schedule behaves at the top of the pyramid since we have only dealt with one half of the pyramid at a time. But we will soon see that jobs on one side of the pyramid cannot affect the cycle times associated with jobs on the other side of pyramid. This will complete the pyramid shape of optimal schedules.

**Lemma 3.1.** Given $n$ ordered jobs, with machine ordering (132), in any schedule jobs that are scheduled before the largest job, job $n$, can not affect cycle times associated with jobs after job $n$, and jobs that are scheduled after job $n$ can not affect cycle times associated with jobs before job $n$.

**Proof.** Let $S = (..., a, n, b, ...)$ be a schedule of $n$ ordered jobs. We see the only cycle time that involves both jobs that are schedule before and after job $n$ is $c^S_{a,n,b}$. Now we see that jobs $a$ and job
b can not affect this cycle time because \( c_{a,n,b}^S = p_{2,n} \) since \( a < n \) and \( b < n \) and by (??) \( p_{3,a} \leq p_{2,n} \) and \( p_{1,b} \leq p_{2,n} \).

\[ \square \]

**Theorem 3.1.** Given \( n \) ordered jobs, with machine ordering (132), an schedule \( S \) of the \( n \) ordered jobs with a minimum makespan is pyramid shaped.

**Proof.** Let \( S \) be a schedule of \( n \) ordered jobs with a minimum makespan. Assume \( S \) is not pyramid shaped. Then either the schedule does not increase in job number up to the largest job or the job numbers do not decrease after the largest job or both. If the former is true, then we take all the jobs that were scheduled before job \( n \) and re-arrange them so that the job numbers increase up to job \( n \), by proposition (??) and lemma (??), the resulting schedule has a smaller makespan, a contradiction. If the job numbers do not decrease after job \( n \) then we use proposition (??) and lemma (??) in a similar manner and generate a contradiction. This covers all the cases. Therefore, a schedule \( S \) is pyramid shaped.

\[ \square \]