Lemma 6.1.2 states the following *reversibility* result: the makespan does not change if the jobs traverse the flow shop in the opposite direction in reverse order.

Example 6.1.3 (Graph Representations and Reversibility). Consider the instance of Example 6.1.1. The dual of this instance is given in the table below.

jobs	j_1	j_2	j_3	j_4	j_5
p_{1,j_k}	5	2	3	6	3
p_{2,j_k}	1	4	3	4	4
p_{3,j_k}	4	4	2	4	4
p_{4,j_k}	3	6	3	5	5

The corresponding directed graph, its critical paths and the Gantt charts are depicted in Figure 6.3. It is clear that the critical paths are determined by the same set of processing times and that the makespan, therefore, is 34 as well. ||

Consider now the $F2 \mid\mid C_{\max}$ problem: a flow shop with two machines in series with unlimited storage in between the two machines. There are n jobs and the processing time of job j on machine 1 is p_{1j} and its processing time on machine 2 is p_{2j}. This was one of the first problems to be analyzed in the early days of Operations Research and led to a classical paper in scheduling theory by S.M. Johnson. The rule that minimizes the makespan is commonly referred to as Johnson's rule.

An optimal sequence can be generated as follows. Partition the jobs into two sets with Set I containing all jobs with $p_{1j} < p_{2j}$ and Set II all jobs with $p_{1j} > p_{2j}$. The jobs with $p_{1j} = p_{2j}$ may be put in either set. The jobs in Set I go first and they go in increasing order of p_{1j} (SPT); the jobs in Set II follow in decreasing order of p_{2j} (LPT). Ties may be broken arbitrarily. In what follows such a schedule is referred to as an *SPT(1)-LPT(2)* schedule. Of course, multiple schedules may be generated this way.

Theorem 6.1.4. *Any SPT(1)-LPT(2) schedule is optimal for* $F2 \mid\mid C_{\max}$.

Proof. The proof is by contradiction. Suppose another type of schedule is optimal. In this optimal schedule there must be a pair of adjacent jobs, say job j followed by job k, that satisfies one of the following three conditions:

(i) job j belongs to Set II and job k to Set I;

(ii) jobs j and k belong to Set I and $p_{1j} > p_{1k}$;

(iii) jobs j and k belong to Set II and $p_{2j} < p_{2k}$.

It suffices to show that under any of these three conditions the makespan is reduced after a pairwise interchange of jobs j and k. Assume that in the original

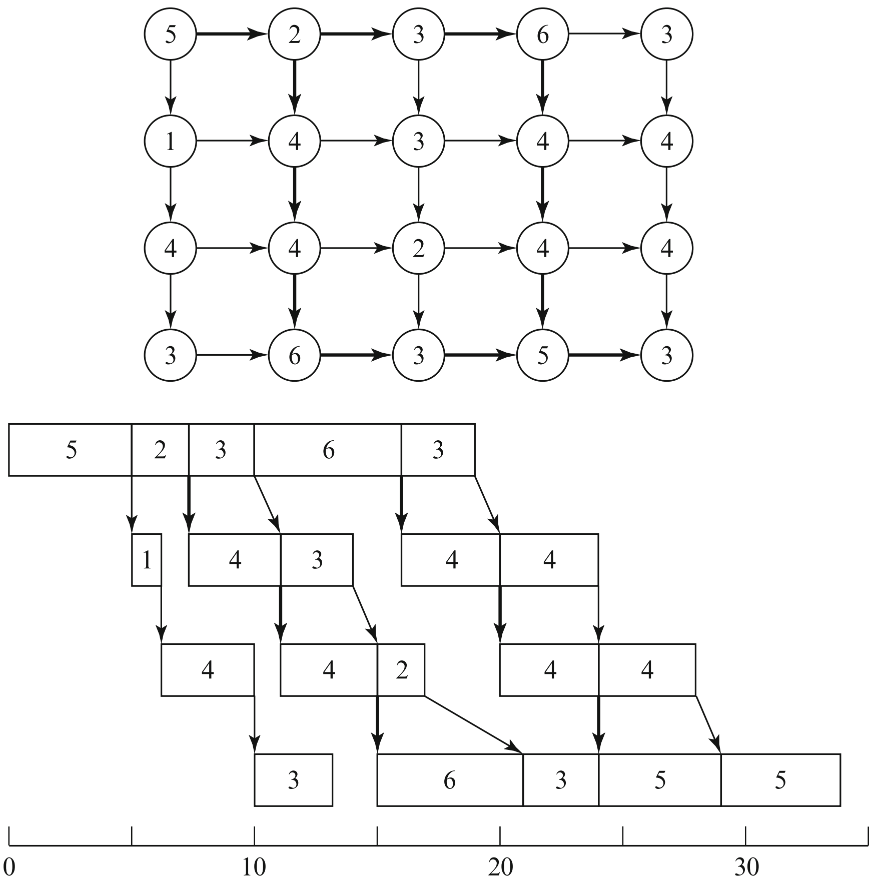

Fig. 6.3 Directed graph, critical paths, and Gantt chart (the numerical entries represent the processing times of the jobs and not the job indexes)

schedule job l precedes job j and job m follows job k. Let C_{ij} denote the completion of job j on machine i under the original schedule and let C'_{ij} denote the completion time of job j on machine i after the pairwise interchange. Interchanging jobs j and k clearly does not affect the starting time of job m on machine 1, as its starting time on machine 1 equals $C_{1l} + p_{1j} + p_{1k}$. However, it is of interest to know at what time machine 2 becomes available for job m. Under the original schedule this is the completion time of job k on machine 2, i.e., C_{2k}, and after the interchange this is the completion time of job j on machine 2, i.e., C'_{2j}. It suffices to show that $C'_{2j} \leq C_{2k}$ under any one of the three conditions described above.

The completion time of job k on machine 2 under the original schedule is

$$\begin{aligned} C_{2k} &= \max\Big(\max\,(C_{2l},\ \ C_{1l} + p_{1j}) + p_{2j},\ \ C_{1l} + p_{1j} + p_{1k}\Big) + p_{2k} \\ &= \max\Big(C_{2l} + p_{2k} + p_{2j},\ \ C_{1l} + p_{1j} + p_{2j} + p_{2k},\ \ C_{1l} + p_{1j} + p_{1k} + p_{2k}\Big), \end{aligned}$$

whereas the completion time of job j on machine 2 after the pairwise interchange is

$$C'_{2j} = \max\Big(C_{2l} + p_{2k} + p_{2j},\ \ C_{1l} + p_{1k} + p_{2k} + p_{2j},\ \ C_{1l} + p_{1k} + p_{1j} + p_{2j}\Big).$$

Under condition (i) $p_{1j} > p_{2j}$ and $p_{1k} < p_{2k}$. It is clear that the first terms within the max expressions of C_{2k} and C'_{2j} are identical. The second term in the last expression is smaller than the third in the first expression and the third term in the last expression is smaller than the second in the first expression. So, under condition (i) $C'_{2j} \leq C_{2k}$.

Under condition (ii) $p_{1j} < p_{2j}$, $p_{1k} < p_{2k}$ and $p_{1j} > p_{1k}$. Now the second and the third term in the last expression are smaller than the second term in the first expression. So, under condition (ii) $C'_{2j} \leq C_{2k}$ as well.

Condition (iii) can be shown in a similar way as the second condition. Actually, condition (iii) follows immediately from the reversibility property of flow shops. □

These SPT(1)-LPT(2) schedules are by no means the only schedules that are optimal for $F2 \mid\mid C_{\max}$. The class of optimal schedules appears to be hard to characterize and data dependent.

Example 6.1.5 (Multiple Optimal Schedules). Consider a set of jobs with one job that has a very small processing time on machine 1 and a very large processing time on machine 2, say K, with $K \geq \sum_{j=1}^{n} p_{1j}$. It is clear that under the optimal sequence this job should go first in the schedule. However, the order of the remaining jobs does not affect the makespan. ||

Unfortunately, the SPT(1)-LPT(2) schedule structure cannot be generalized to characterize optimal schedules for flow shops with more than two machines. However, minimizing the makespan in a permutation flow shop with an arbitrary number of machines, i.e., $Fm \mid prmu \mid C_{\max}$, can be formulated as a Mixed Integer Program (MIP).

In order to formulate the problem as an MIP a number of variables have to be defined: The decision variable x_{jk} equals 1 if job j is the kth job in the sequence and 0 otherwise. The auxiliary variable I_{ik} denotes the idle time on machine i between the processing of the jobs in the kth position and $(k+1)$th position and the auxiliary variable W_{ik} denotes the waiting time of the job in the kth position in between machines i and $i+1$. Of course, there exists a strong relationship between the variables W_{ik} and the variables I_{ik}. For example, if $I_{ik} > 0$, then $W_{i-1,k+1}$ has to be zero. Formally, this relationship can be established by considering the difference between the time the job in the $(k+1)$th position starts on machine $i+1$ and the time the job in the kth position completes its processing on machine i. If Δ_{ik} denotes this difference and if $p_{i(k)}$ denotes the processing time on machine i of the job in the kth position in the sequence, then (see Figure 6.4)