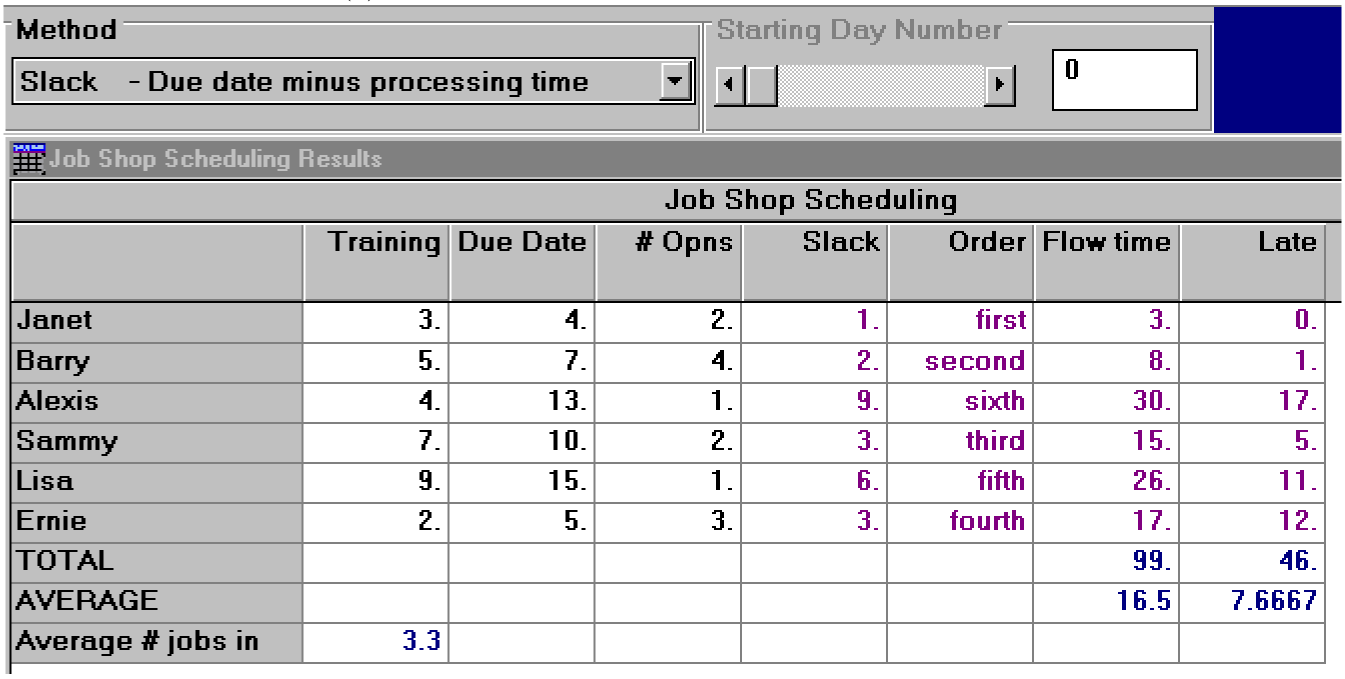
SLACK TIME

Example 3 - Schedule according to slack

Slack is defined as the due date minus the time required to process a job. In order to use slack the due date must be given. The slack column did not appear before but does now. It is the difference between the due column and the "training" column. For example, Janet must be trained by day 4 but it takes 3 days to train so there is one day of slack. The jobs are scheduled according to increasing order of slack. Janet has the least and is scheduled first while Alexis has the most (9) and is scheduled last.



Method: Slack - Due date minus processing time

Starting Day Number: 0

Job Shop Scheduling Results

Job Shop Scheduling

	Training	Due Date	# Opns	Slack	Order	Flow time	Late
Janet	3.	4.	2.	1.	first	3.	0.
Barry	5.	7.	4.	2.	second	8.	1.
Alexis	4.	13.	1.	9.	sixth	30.	17.
Sammy	7.	10.	2.	3.	third	15.	5.
Lisa	9.	15.	1.	6.	fifth	26.	11.
Ernie	2.	5.	3.	3.	fourth	17.	12.
TOTAL						99.	46.
AVERAGE						16.5	7.6667
Average # jobs in	3.3						

Example 4 - Slack time per operation

We have used the data in the number of operations column. The output (not shown) contains a new column titled slack/op which is generated by dividing the slack by the number of operations. For example the 1 day of slack for job 1 is divided by the 2 operations yielding a slack per operation value of .5. Jobs are scheduled according to increasing order of slack per operation. Therefore, Janet is first (.5) and Alexis is last (9). (Ties are broken arbitrarily.)

N Jobs, M Machines

List of Heuristics are as follows:

1. R (Random) – Pick any Job in Queue with equal probability. This rule is often used as
benchmark for other rules
2. FCFS (First Come First Serve) – Jobs are processed in the order in which they arrived at
the work center (also called earliest release date)
3. SPT (Shortest Processing Time) –This rule tends to reduce both work-in-process
inventory, the average job completion (flow) time, and average job lateness.
4. EDD (Earliest Due Date) – Choose Job that has earliest due date
5. CR (Critical Ratio) = Processing Time / Time until due (Due Date – Current Time).
Take the highest value.
6. LWR (Least Work Remaining) – This rule is an extension of SPT variant that considers
the number of successive operations
**7. ST (Slack Time) = Time until job is due - (Sum of processing time remaining). Take the
job with the smallest amount of slack time.**
8. ST/O (Slack Time per Remaining Operation) = slack time divided by number of
operations remaining. Take the job with the smallest amount of slack time per remaining
operation
When in Doubt, use SPT. Also, use SPT to break ties.

Least Slack Time (LST) scheduling is a scheduling algorithm. It assigns priority based on the *slack time* of a process. Slack time is the amount of time left after a job if the job was started now. This algorithm is also known as **Least Laxity First**. Its most common use is in embedded systems, especially those with multiple processors. It imposes the simple constraint that each process on each available processor possesses the same run time, and that individual processes do not have an affinity to a certain processor. This is what lends it a suitability to embedded systems.

Slack time

This scheduling algorithm first selects those processes that have the smallest "slack time". Slack time is defined as the temporal difference between the deadline, the ready time and the run time.
More formally, the *slack time* for a process is defined as:

$$(d - t) - c'$$

where d is the process deadline, t is the real time since the cycle start, and c' is the remaining computation time.

Suitability

LST scheduling is most useful in systems comprising mainly aperiodic tasks, because no prior assumptions are made on the events' rate of occurrence. The main weakness of LST is that it does not look ahead, and works only on the current system state. Thus, during a brief overload of system resources, LST can be sub-optimal. It will also be suboptimal when used with uninterruptible processes. However, like earliest deadline first, and unlike rate monotonic scheduling, this algorithm can be used for processor utilization up to 100%.