Deadline Monotonic Scheduling :

Dms is a fixed-priority preemptive algorithm.it is used in a real time system. Priorities are assigned to each task based on their relative deadline. It follows shortest deadline is the highest priority. It assumes that tasks are periodic and the deadline is equal or less than its period. Task can be represented **T(Q,P,E,D)**

Where P= Period; Q=release time; E= Execution Time and D= deadline ;

Advantages :

- Optimal for Static Priority Scheduling.
- Performs well in case of availability of tasks having longer period but shorter deadlines.
- Good performance in case of overload.

Limitation :

- Implementation is complex.
- It is a time taking process.
- It uses fixed priority.
- DMS has fixed priorities, which means it isn't very flexible when it comes to adjusting to changes in tasks. This can make it hard for DMS to handle tasks that have strict or uneven deadlines, especially if the deadlines don't match up well or if the timing of the tasks varies a lot.

For reference:
https://www.geeksforgeeks.org/deadline-monotonic-cpu-scheduling/

Least Laxity First (LLF)

Least laxity first is a job level dynamic priority scheduling algorithm. It calculates laxity(time). Laxity is something that defines the difference between remaining time of deadline and the time required to complete the task. A task with the smallest laxity is the highest priority and schedule first. Priority may change rapidly and frequently during execution time.

Laxity = Deadline -Current Time - Remaining Execution Time

Advantages :

It provides highly responsiveness to deadline urgency.

LLF is considered an optimal algorithm as it guarantees scheduling for task sets that pass the utilization test .

Adapts changes in execution time and makes it suitable for dynamic systems.

Limitations:

LLF can suffer from thrashing when multiple tasks have the same least laxity, resulting in frequent context switching.

As LLF changes priority frequently and rapidly, resulting in high overhead and reduced systems efficiency.

For reference:
https://microcontrollerslab.com/least-laxity-first-llf/#:~:text=In%20conclusion%2C%20the,frequent%20context%20switching

Minimal Slack Scheduling (MSS)

Minimal Slack Scheduling (MSS) is a dynamic scheduling . It builds on the ideas of Least Laxity First (LLF). It looks at not only the slack time of each task but also how these tasks affect the system as a whole. MSS focuses on scheduling the task that is most likely to miss its deadline first, which helps reduce the risk for all tasks that are currently running. This method allows for smarter scheduling choices and reduces unnecessary preemptions compared to LLF.

Advantages

Smartly handles tasks based on execution time and deadlines.

This approach allows for more strategic scheduling decisions and can reduce unnecessary preemptions compared to LLF.

Limitation:

However, MSS is more complicated because it needs constant checking of slack times and how tasks are progressing.

MSS implementation is complex in real-time environment as well as expensive.

It works especially well in systems that are overloaded or when resources need to be used carefully.

Every scheduling algorithm has its own unique benefits and limitations, which can vary based on the situation in which they are applied.

Deadline Monotonic Scheduling works best for systems that have regular, predictable tasks where assigning fixed priorities is effective. Its straightforward nature and reliability make it a popular option in environments where safety is crucial, but it may not perform well with tasks that are unpredictable or happen at irregular intervals.

On the other hand, Least Laxity First provides better responsiveness and can be the best choice for scheduling in many situations. However, it is difficult to use in practice because it requires a lot of resources because it frequently changes tasks and the need for accurate time estimates. LLF is more suitable for situations where meeting deadlines is very important and where task times can be closely tracked. In a sentence we can say Dynamic systems with varying deadlines.

Minimal Slack Scheduling aims to balance quick responses with keeping the system stable. It does this by reducing unnecessary interruptions while still being flexible to the needs of different tasks. It works well in systems that experience varying loads or where it's important to manage overload smoothly. However, it requires more computing power and is more complex to implement than Deadline Monotonic Scheduling.

Aspect	**DMS**	**LLF**	**MSS**
Priority Assignment	**Static, based on deadline**	**Dynamic, based on laxity**	**Dynamic, based on slack time**
Adaptability	**Low**	**High**	**High**
Overhead	**Low**	**High**	**High**
Best Use Case	**Predictable systems with fixed tasks**	**Dynamic systems with varying deadlines**	**Systems with tight timing constraints**
Limitations	**Fails with high utilization**	**Task thrashing and high overhead**	**High overhead and resource requirements**

In conclusion, choosing between these scheduling algorithms depends on the specific needs of the real-time system, including how often tasks occur, the system's workload, the importance of deadlines, and how complex the solution can be.