How much dynamic behaviour is needed for optimality?

Rob Davis
Single processor

- Liu & Layland task model
- Utilisation bound for FP-P

\[ U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N(2^{1/N} - 1) \quad U \leq 0.69 \text{ as } N \to \infty \]

- EDF-P

\[ \sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1 \]
Fixed priority with deferred pre-emption

- With FP-DS, each task $\tau_i$ has a final non-pre-emptive region of length $q_i$
  - Larger $q_i$ improves schedulability of $\tau_i$ but increases blocking on higher priority tasks
  
- FP-DS dominates both FP-P and FP-NP
  
- $q_i = 0$ (FP-P)
  - $q_i = C_i$ (FP-NP)
Optimality for FP-DS

Algorithm $A$ is optimal for FP-DS if for any taskset where there exists a priority ordering and set of $q_i$ that are schedulable, algorithm $A$ also provides a priority ordering and set of $q_i$ that is schedulable.
for each priority level \( k \), lowest first
{
    for each unassigned task \( \tau \)
    {
        binary search for the smallest value of \( q \) for which task \( \tau \) is schedulable at priority \( k \)
    }
    if no tasks are schedulable at priority \( k \)
    return unschedulable
    else
    {
        assign the schedulable task that tolerates the min \( q \) at priority \( k \) to priority \( k \) and use this min \( q \) value for its final non-preemptive region length
    }
}
return schedulable
FP-DS Optimal algorithm

- Optimal algorithm
  - Greedy: \( n(n+1)/2 \) binary searches and therefore tractable
  - Minimises blocking at every priority level

- Utilisation bound
  - Remains same as FP-P \( U \leq 0.69 \) as \( N \to \infty \)
Dual Priority scheduling

- After time $x_i \leq D_i$ task $\tau_i$ changes priority to a new level (higher or same).

- Open question: How to determine the dual priorities of each task and the $x_i$ values such that the system is schedulable?

- What is the utilisation bound?
  - Conjecture: It is 100% same as EDF
Current state

- For \( N=2 \) a proof has been obtained (so \( U=1 \) rather than \( U=0.83 \) for standard FP-P)
- No counter example found with extensive searches for \( N=3 \)
- Initial priorities are probably Rate Monotonic, possibly final ones too, and all higher than initial priorities
Multiprocessor

- L&L tasks on $m$ identical processors
- DP-Wrap approach: each task makes proportionate progress between any two adjacent deadlines

![Diagram showing progress on processors P1, P2, and P3]
Multiprocessor

- N-1 pre-emptions per job

- Question: What is the minimum number of pre-emptions per job necessary for optimality (U = 100%)? Algorithm to do this?

- For fixed job priority (single pre-emption per job) bound is 50% ((m+1)/2) – achieved by EDF-US and EDF(k).
Minimally dynamic algorithms

- Minimally dynamic – one change in priority per job => 2 pre-emption
  - EDZL, FPZL

- Utilisation bound for EDZL
  \[ U_{EDZL} \leq m(1 - 1/e) \approx 0.63m \]

- Utilisation bound for FPZL
  - Not known

- Can we do better with just one change in priority per job?