Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-Time Systems

Robert Davis and Alan Burns

Real-Time Systems Research Group, University of York

Research scope

- **Homogeneous Multiprocessor Real-Time Systems**

↓

- **Global scheduling**
  - Single global run-queue
  - Pre-emption and migration

↓

- **Fixed priority scheduling**
  - All jobs of a task have the same fixed priority

↓

- **How should priorities be assigned?**

Motivation

- **Aim to close the gap between existing sufficient schedulability tests for sporadic tasksets with constrained deadlines and what may be possible as indicated by infeasibility tests**
  - RTA test for global FP scheduling (DMPO) shown to be more effective than state-of-the-art tests for global EDF and EDZL (Bertogna [17])
  - Deadline Monotonic Priority Ordering (DMPO) can have poor performance in the multiprocessor case (Dhall [25])
- **Hypothesis:**
  - Priority assignment is key to the effectiveness of FP scheduling
  - Possible to improve schedulability test performance by using more effective priority assignment policies

Outline of presentation

- **System model, terminology, and definitions**
- **Recap on schedulability tests for global FP scheduling**
- **Optimal Priority Assignment (OPA) algorithm**
  - Which schedulability tests are OPA-compatible?
- **Heuristic priority assignment policies**
- **Taskset generation**
  - Parameter independence, UUnifast-Discard algorithm
- **Empirical results**
- **Summary and conclusions**

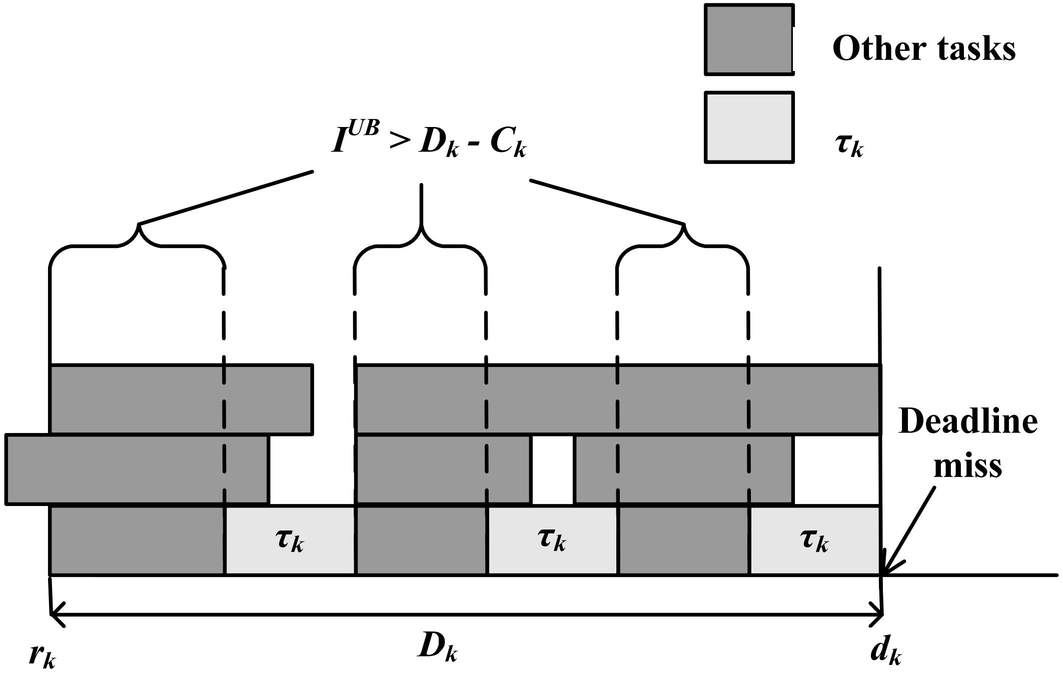
System model

- **Multiprocessor system**
  - *m* identical processors
  - Global fixed priority pre-emptive scheduling
  - Migration is permitted, but a job can only execute on one processor at a time
- **Sporadic task model**
  - Static set of *n* tasks τ_i with priorities 1..*n*
  - Bounded worst-case execution time C_i
  - Sporadic/periodic arrivals: minimum inter-arrival time T_i
  - Relative deadline D_i (Constrained deadlines $\leq T_i$)
  - Independent

Definitions

- **Feasibility and Optimality**
  - A taskset is said to be feasible on a multiprocessor system if there exists some scheduling algorithm that can schedule the taskset without missing a deadline
  - A scheduling algorithm is said to be optimal if it can schedule all feasible tasksets
- **Schedulability tests**
  - Sufficient – if all tasksets / priority ordering combinations deemed schedulable are in fact schedulable
  - Necessary – if all tasksets / priority ordering combinations deemed unschedulable are in fact unschedulable
  - Exact – both sufficient and necessary
- **No exact tests are known for global FP scheduling of sporadic tasksets**

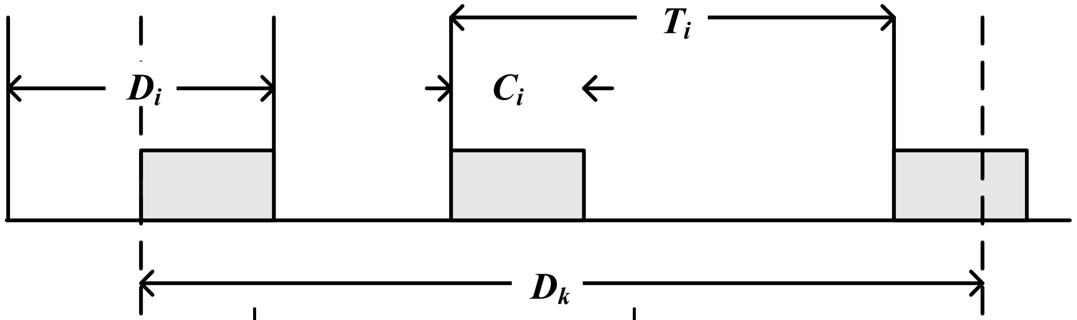
Sufficient schedulability tests



- **Fundamental approach (Baker [8])**
  - Problem window in which deadline is missed (e.g. D_k)
  - Necessary condition for deadline miss: m processors all occupied for more than D_k - C_k
  - Derive upper bound on interference I^{UB}
  - Negate the un-schedulability condition to form a sufficient schedulability test

Sufficient schedulability tests

- **Polynomial time test: Deadline Analysis ("DA test") (Bertogna et al. [18])**



$$D_k \geq C_k + \left\lfloor \frac{1}{m} \sum_{\forall i \in hp(k)} I_i^D(D_k) \right\rfloor$$

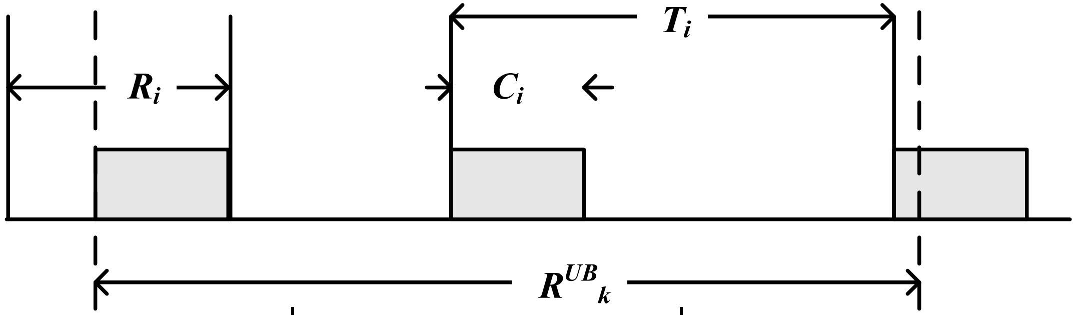
$$I_i^D(D_k) = \min(W_i^D(D_k), D_k - C_k + 1)$$

$$W_i^D(D_k) = N_i(D_k)C_i + \min(C_i, D_k + D_i - C_i - N_i(D_k)T_i)$$

$$N_i(D_k) = \left\lfloor \frac{D_k + D_i - C_i}{T_i} \right\rfloor$$

Sufficient schedulability tests

- **Pseudo-polynomial time Response Time Analysis ("RTA test") (Bertogna and Cirinei [16])**



$$R_k^{UB} \leftarrow C_k + \left\lfloor \frac{1}{m} \sum_{\forall i \in hp(k)} I_i(R_k^{UB}) \right\rfloor$$

$$I_i(R_k^{UB}) = \min(W_i^R(R_k^{UB}), R_k^{UB} - C_k + 1)$$

$$W_i^R(L) = N_i^R(L)C_i + \min(C_i, L + \boxed{R_i^{UB}} - C_i - N_i^R(L)T_i)$$

$$N_i^R(L) = \left\lfloor \frac{L + R_i^{UB} - C_i}{T_i} \right\rfloor$$

Optimal Priority Assignment

- **Definition of Optimal Priority Assignment**
  - In FP scheduling, a priority assignment policy *P* is referred to as optimal with respect to some schedulability test *S* if there are no tasksets that are deemed schedulable by the test using any other priority assignment policy *Q*, that are not also deemed schedulable using policy *P*.
- **Definition applicable to sufficient tests as well as exact tests**
  - Sufficient test + optimal priority assignment cannot schedule all tasksets that are feasible under global FP
  - Exact test + optimal priority assignment can schedule all tasksets that are feasible under global FP, but not all feasible tasksets (global FP is not an optimal multiprocessor scheduling algorithm)

Optimal Priority Assignment

- **Single processor:**
  - Constrained-deadline tasksets: DMPO is optimal
  - Arbitrary-deadline tasksets / tasks with offsets – Audsley's Optimal Priority Assignment algorithm [6], [7] is optimal

Optimal Priority Assignment (OPA) Algorithm

```
for (each priority level k, lowest first){
   for (each unassigned task τ){
        if (τ is schedulable at priority k
         according to schedulability test S){
            assign τ to priority k
                   break (continue outer loop)
        }
   }
   return unschedulable
}
return schedulable
```

Optimal Priority Assignment

- **Multiprocessor:**
  - B. Andersson and Jonsson [1] observed that for periodic tasksets scheduled using global FP scheduling

    "There exist task sets for which the response time of a task depends not only on C_i and T_i of its higher-priority tasks, but also on the relative priority ordering of those tasks"

    Conclusion

    It isn't possible to use Audsley's OPA algorithm to determine the optimal priority ordering for periodic tasks when using an exact schedulability test ✓

    Possibly led to a general misconception

    OPA algorithm cannot be used for priority assignment in multiprocessor global FP scheduling ✗

Optimal Priority Assignment

- **OPA algorithm provides optimal priority assignment w.r.t. any schedulability test *S* for global FP scheduling provided that 3 conditions are met…**

Condition 1: Schedulability of a task may, according to the test, be dependent on the set of higher priority tasks, but not on their relative priority ordering

Condition 2: Schedulability of a task may, according to the test, be dependent on the set of lower priority tasks, but not on their relative priority ordering

Condition 3: When the priorities of any two tasks of adjacent priority are swapped, the task being assigned the higher priority cannot become unschedulable according to the test, if it was previously deemed schedulable at the lower priority

Tests meeting these conditions referred to as **OPA-compatible**

Optimal Priority Assignment

- **OPA-Compatible tests**
  - Deadline Analysis (DA test) of Bertogna et al. [18]
  - Simple Response Time test of B. Andersson and Jonsson [1]

$$R_k^{ub} \leftarrow C_k + \frac{1}{m} \sum_{\forall i \in hp(k)} \left(\left\lceil \frac{R_k^{ub}}{T_i} \right\rceil C_i + C_i \right)$$

- **OPA-Incompatible tests**
  - Any exact schedulability test for periodic tasksets (e.g. Cucu and Goossens [20], [21])
  - Response time analysis (RTA test) of Bertogna and Cirinei [16]
  - Improved RTA test of Guan et al. that limits carry-in interference

Heuristic Priority Assignment Policies

- **Deadline Monotonic Priority Ordering (DMPO)**
  - Optimal for single processor case when tasks have constrained deadlines (Leung & Whitehead [31])
  - Assumed in the majority of research on global FP scheduling
  - Suffers from the "Dhall effect" (Utilisation bound close to 1 for *m* processors)
- **Deadline minus Computation time Monotonic Priority Ordering (DCMPO)**

$$D_k - C_k \geq \left\lfloor \frac{1}{m} \sum_{\forall i \in hp(k)} I_i^D(D_k) \right\rfloor$$

RHS increases slowly for large *m* suggesting that DCMPO might be an effective heuristic priority assignment policy

Heuristic Priority Assignment Policies

- **TkC policy (B. Andersson and Jonsson [2])**
  - Devised for tasksets with implicit deadlines
  - Aimed at circumventing the Dhall effect
  - Priority based on T_i - k C_i

$$k = \frac{m - 1 + \sqrt{5m^2 - 6m + 1}}{2m}$$

  - k varies from 1 to ≈1.62
- **DkC policy**
  - Simple extension of TkC to the constrained deadline case
  - Priority based on D_i - k C_i

Taskset Generation

- **Randomly generated tasksets used to examine schedulability test and priority assignment policy effectiveness**
- **Requirements for Taskset generation:**
  - Unbiased distribution of task utilisation values
  - Able to generate tasksets with different parameter settings (e.g. number of tasks, total utilisation) independent of one another
  - Fast – able to generate many tasksets in a reasonable time frame
- **UUnifast algorithm (Bini & Buttazzo [19])**
  - Achieves this for single processor case
  - De-facto standard for empirical investigation of uniprocessor schedulability tests

Taskset Generation

- **UUnifast algorithm (Bini & Buttazzo [19])**

```
UUnifast(n,Ut)
{
     SumU = Ut;
     for (i = 1 to n-1) {
         nextSumU = SumU * pow(rand(), 1/(n-i));
         U[i] = SumU - nextSumU;
         sumU = nextSumU;
     }
     U[n] = SumU;
}
```

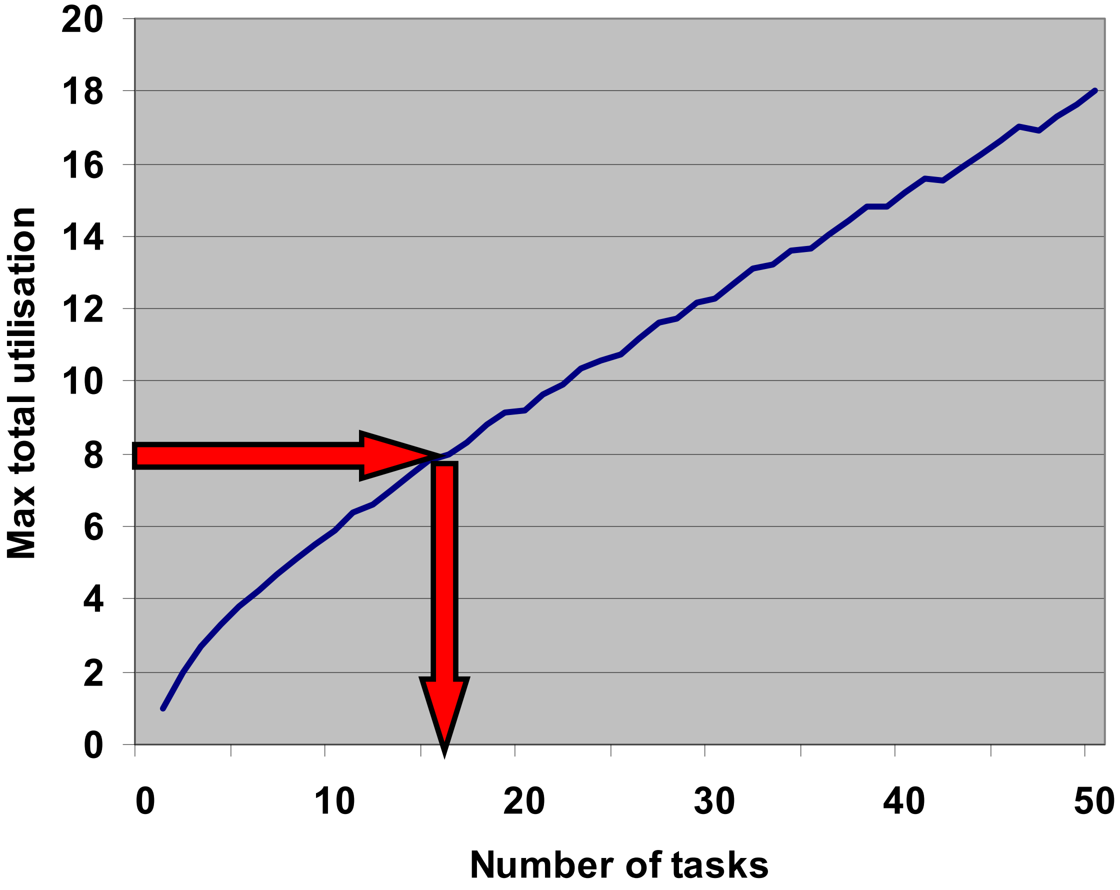
- **UUnifast is scale invariant**
  - Can be used to generate tasksets with total utilisation > 1 ✓
  - **BUT** some tasks may be given a utilisation > 1 ✗

Taskset Generation

- **UUnifast-Discard**
  - Use UUnifast and simply discard any taskset generated with an invalid task (utilisation > 1)
  - Set a pragmatic discard limit on how many tasksets we are willing to discard per valid taskset generated (e.g. 1000)
- **Advantages of UUnifast-Discard**
  - Unbiased distribution of utilisation values
  - Can vary number of tasks and taskset utilisation independently – avoids problem of confounding variables
- **Disadvantages of UUnifast-Discard**
  - Does not cover all of the problem space

    In practice UUnifast-Discard covers enough of the problem space for some useful experiments

Taskset Generation

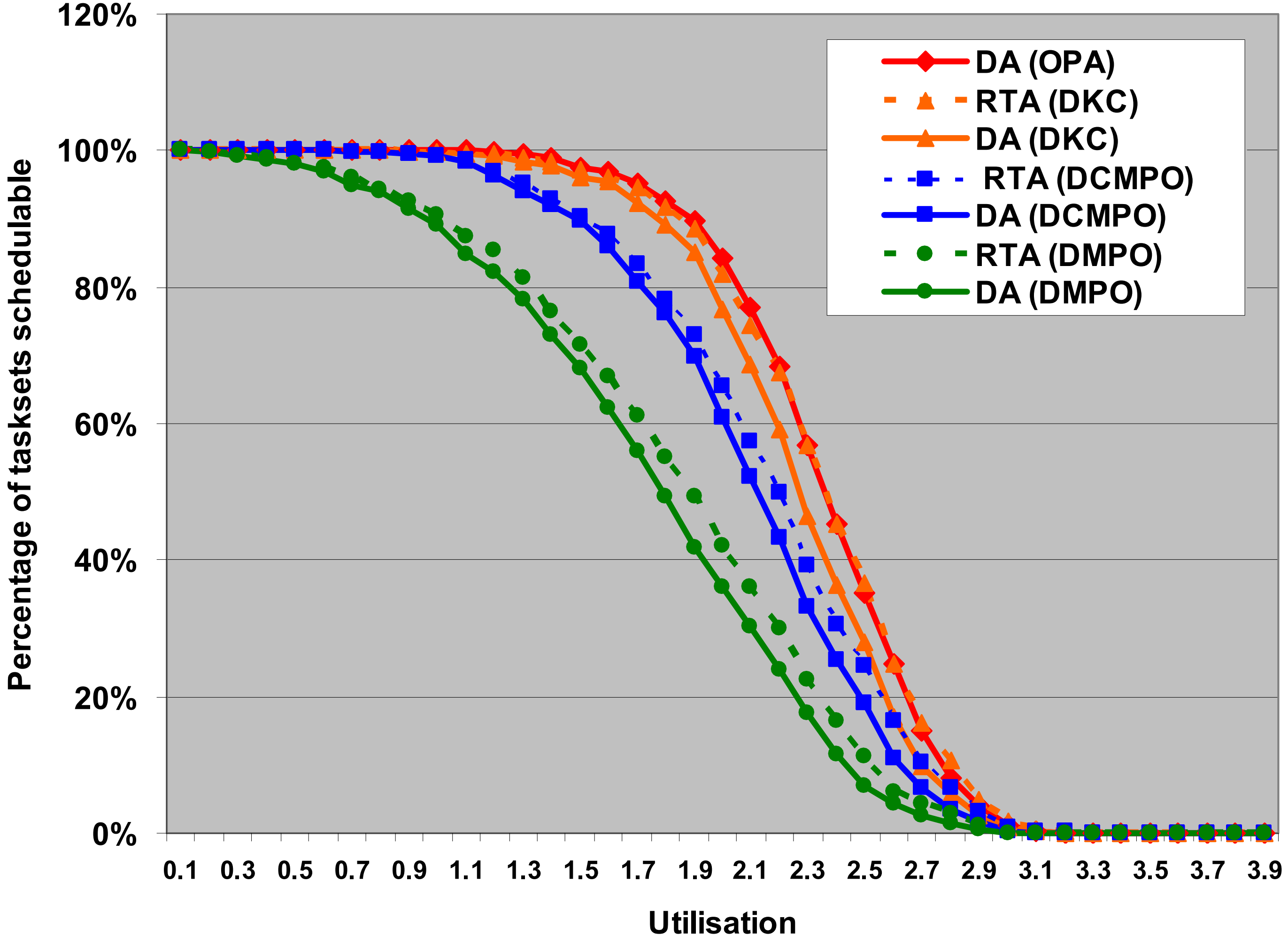


- **UUnifast-Discard**
  - Graph assumes a discard limit of 1000
  - Algorithm is effective for wide range of parameters used in most experiments

Empirical Investigation

- **Taskset parameters**
  - Task utilisations generated via UUnifast-Discard
  - Task periods chosen from a log-uniform distribution with a range from min to max period of 1000 (e.g. 1ms to 1 sec)
  - Execution times set from task utilisation and period values
  - Task deadlines chosen from a uniform distribution between execution time and period
  - Total utilisation varied from 0.025*m* to 0.975*m* in steps of 0.025*m*
  - 1000 tasksets generated for each total utilisation level
  - Graphs plot the percentage of tasksets that are schedulable according to each schedulability test / priority assignment policy, against total utilisation

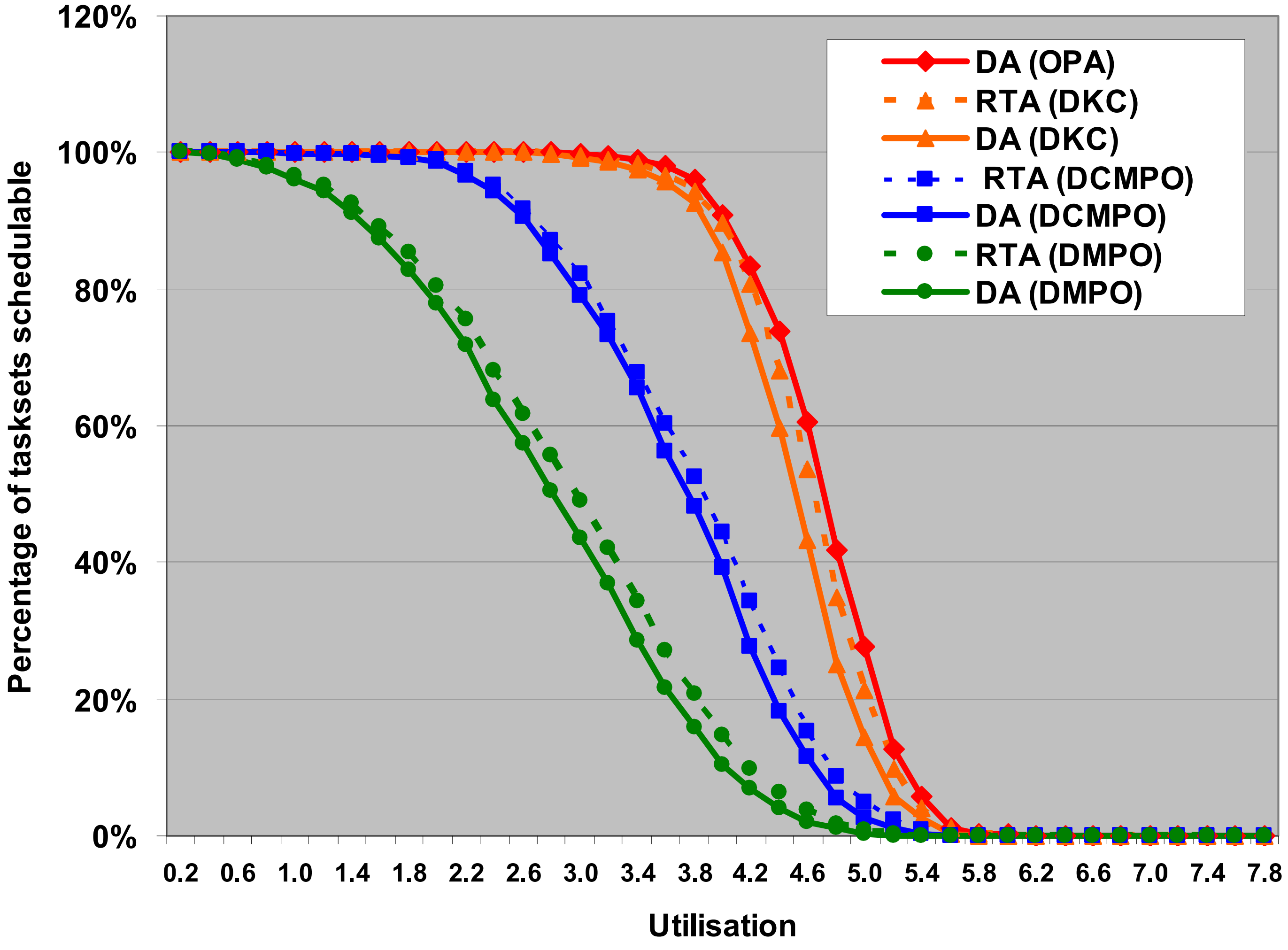
Expt 1: Priority Assignment



4 Processors

20 tasks

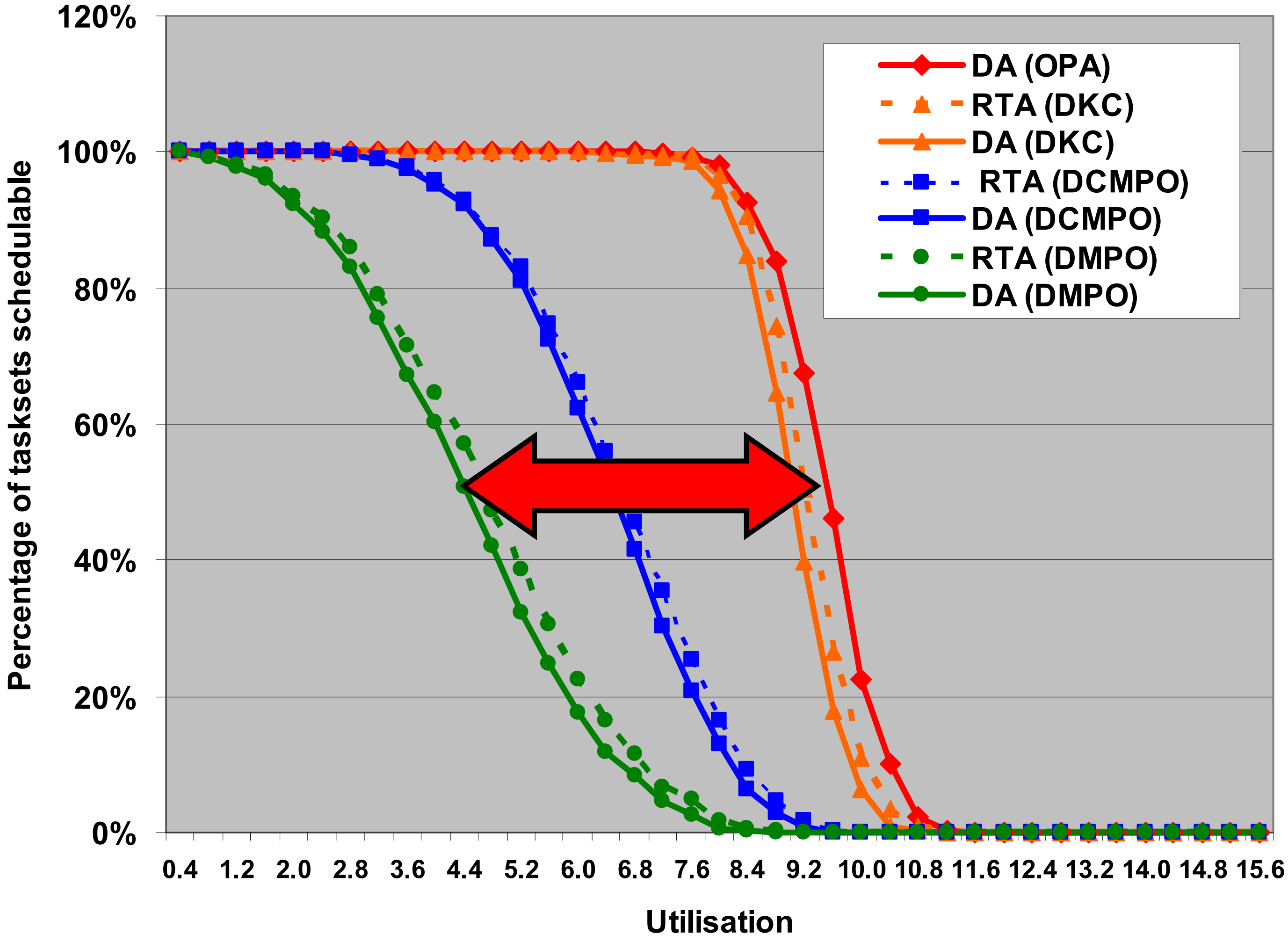
Expt 1: Priority Assignment



8 Processors

40 tasks

Expt 1: Priority Assignment



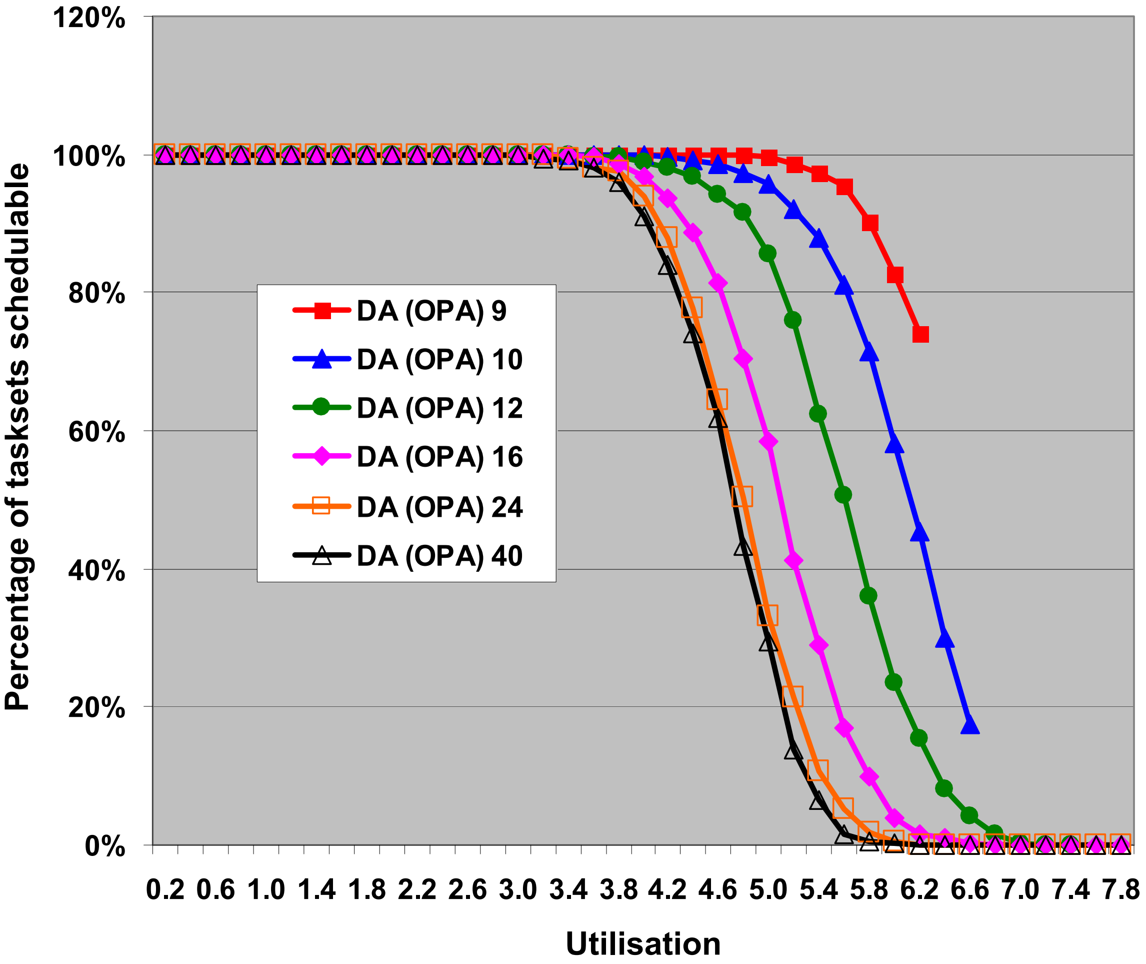
16 Processors

80 tasks

Total number of schedulable tasksets increased from ≈10,000 with DMPO to ≈23,000 with OPA

>100% more

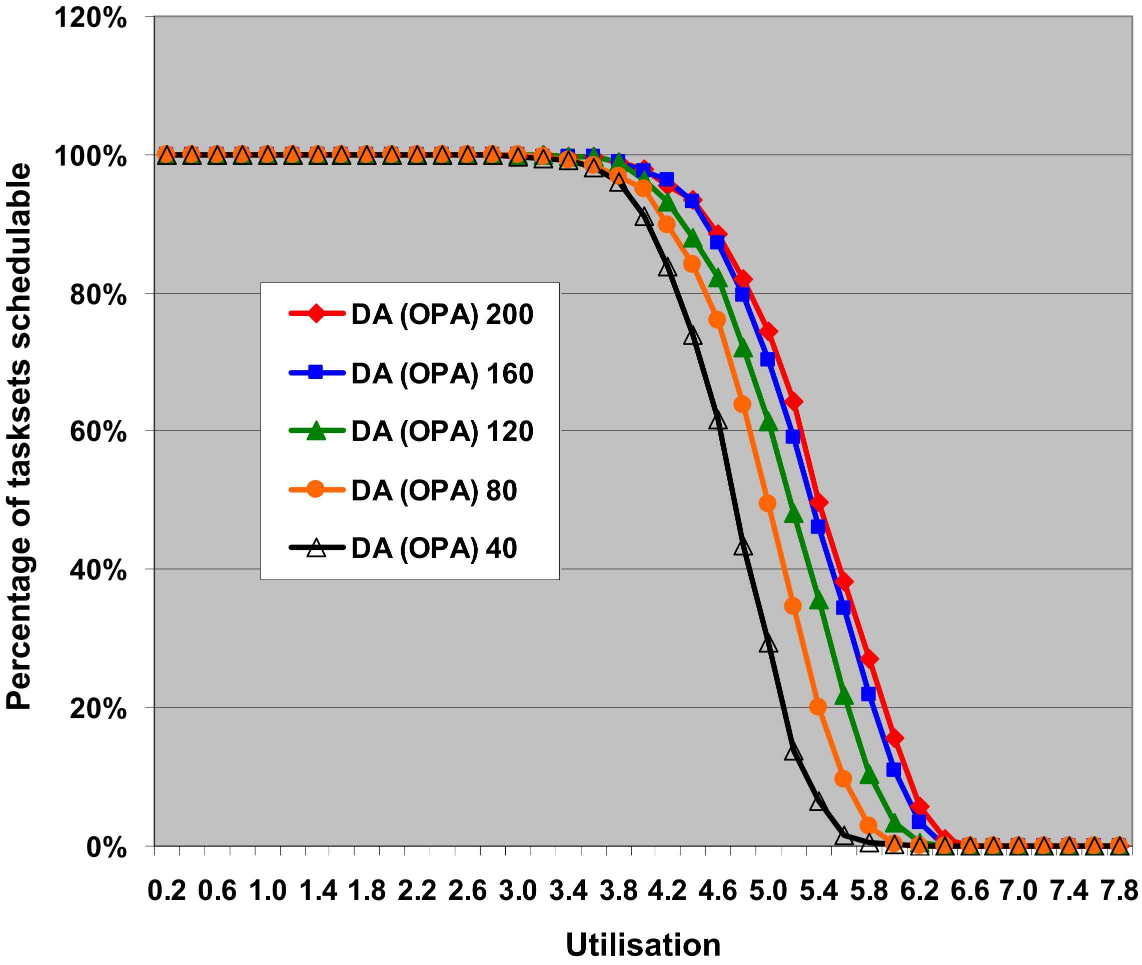
Expt 2: Number of tasks



8 processor system

- Note UUnifast-Discard can't generate tasksets with n = 9, U > 6.6
- Becomes **harder** to schedule tasksets as the number of tasks increases from 9 to 40
- With a small number of tasks, each high utilisation task effectively occupies a single processor and can be scheduled

Expt 2: Number of tasks



8 processor system

- Becomes **easier** to schedule tasksets as number of tasks increases from 40 to 200
- With a large number of tasks, average task utilisation is small, reducing the pessimism in the assumption that all other processors are idle when the task of interest executes

Summary

- **Motivation**
  - To improve on current state-of-the-art in terms of practical techniques that enable the efficient use of processing capacity in hard real-time systems based on multiprocessors.
  - Drawn to this area of research by the results of Bertogna et al. [18] showing that schedulability tests for global FP scheduling using DMPO outperformed those for global EDF and EDZL
- **Hypothesis**
  - Priority assignment is of fundamental importance in global FP scheduling
  - Possible to improve schedulability test performance by using more effective priority assignment policies

Summary

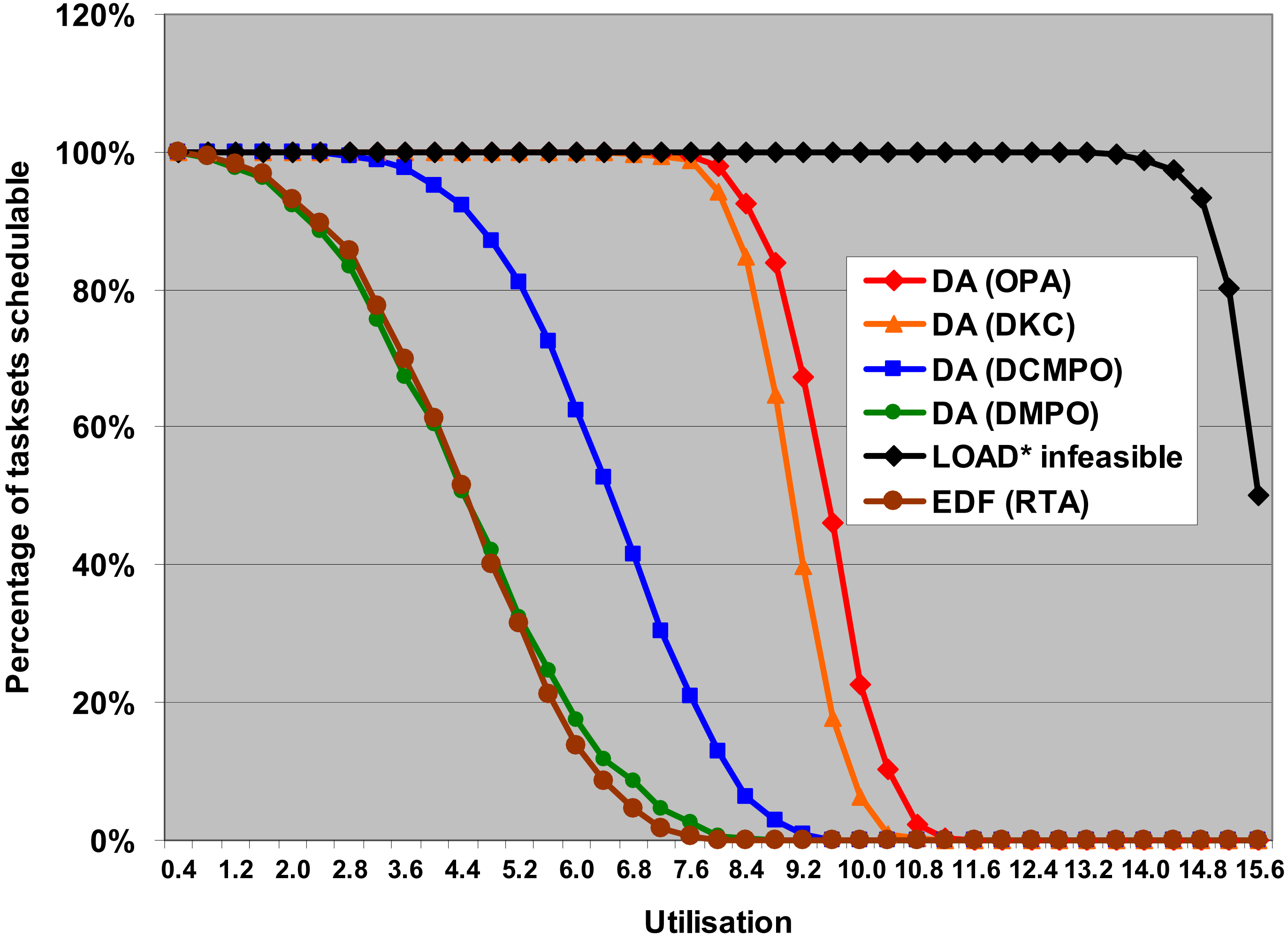
- **Contribution**
  - Proof that the **OPA algorithm** (Audsley [6], [7]) provides optimal priority assignment for global FP scheduling tests that meet 3 simple conditions: **OPA-compatible** tests
  - **DkC** priority assignment policy for constrained deadline tasksets
    - Trivial extension of TkC
    - Effective policy to use with **OPA-incompatible** tests
  - **UUnifast-Discard**
    - Adaptation of UUnifast algorithm to multiprocessor case
    - Generates unbiased distribution of task utilisation values and avoids the problem of confounding variables
    - Area of future work: covering all of the problem space

Conclusions

- **Empirical Evaluation**
  - Shows that the OPA algorithm and DkC priority assignment are highly effective at improving the schedulability of constrained-deadline tasksets under global FP scheduling
  - In the 16 processor case using OPA rather than DMPO
    - More than doubled the number of schedulable tasksets
    - Effectively more than doubled the processor capacity that could be used by hard real-time tasks
  - Made a significant contribution to closing the gap between sufficient schedulability tests for global FP scheduling and what might be possible as indicated by infeasibility tests

Questions ?

Expt1: EDF and LOAD*



16 Processors

80 tasks

Survey paper

"A Survey of Hard Real-Time Scheduling Algorithms and Schedulability Analysis Techniques for Multiprocessor Systems"

Now available as Technical Report YCS-2009-443