Robust scheduling of task graphs under execution time uncertainty

Michele Lombardi
Michela Milano
Luca Benini
Main topic: **reliable scheduling** of multitask applications on multiprocessor systems in the presence of **execution time uncertainty**

Some assumptions:
- Tasks have **temporal dependencies**
- **No preemption** support
- **Hard deadline constraints** are specified

**Context:**
- Real time applications
- Clustered, domain specific multicore platforms (e.g. Cell, Nomadik...)
Off-line allocation and scheduling, PROs and CONs:

- **Improves application performance**
- **Requires a lot of a-priority knowledge**
- **Requires conservative assumptions (e.g. force WCET durations & force fixed start times to avoid anomalies)**

A fixed start time, WCET schedule:
Off-line allocation and scheduling, PROs and CONs:

- **Improves application performance**
- **Requires a lot of a-priority knowledge**
- **Requires conservative assumptions (e.g. force WCET durations & force fixed start times to avoid anomalies)**

A fixed start time, WCET schedule:
Off-line allocation and scheduling, PROs and CONs:

- **Improves application performance**
- **Requires a lot of a-priority knowledge**
- **Requires conservative assumptions (e.g. force WCET durations & force fixed start times to avoid anomalies)**

A fixed start time, WCET schedule:

(IDLE Time insertion) Timers and interrupts
Precedence Constraint Posting

Provide a schedule as a set of additional precedence relations:

- Those are added to avoid resource contention => no anomalies
- Tasks are started when all predecessors are over => little idle time
Precedence Constraint Posting

Provide a schedule as a set of **additional precedence relations**

- Those are avoid to avoid resource contention => **no anomalies**
- Tasks are started when all predecessors are over => **little idle time**

![Diagram showing task scheduling and precedence constraints](image)
Precedence Constraint Posting is a hybrid scheduling method:

- A graph augmentation is computed off-line
- But it is executed by an on-line scheduler

In detail, the output of the on-line process is:

- A set of precedence relations
- A set of restricted time windows

The on-line scheduler must:

- Start a task when the predecessors are over
- Start a task within the restricted time window

The on-line scheduler can be smarter than this:

- E.g. a task as soon as possible
• Preventing resource conflicts requires some knowledge on duration variability
• We use **duration bounds rather than probability distributions**

[Diagram with variables: $T_i$, $d_{min}$, $d_{max}$]

• Easier to obtain than probability distributions
• More computationally tractable
• Poor expected completion time approximation
The target problem

Task Graph

Resources:

\[
\begin{array}{c|cccccc}
\text{C} & \text{T0} & \text{T1} & \text{T2} & \text{T3} & \text{T4} & \text{T5} & \text{T6} \\
\hline
\text{R0} & 2 & -- & 2 & -- & 1 & 2 & 1 & -- \\
\text{R1} & 3 & -- & 1 & 2 & -- & 2 & 1 & -- \\
\hline
\text{d}_{\text{min}} & 0 & 1 & 2 & 1 & 1 & 1 & 0 \\
\text{d}_{\text{max}} & 0 & 2 & 3 & 3 & 2 & 2 & 0 \\
\end{array}
\]

Variable durations:

\[
\begin{align*}
\delta_{\text{min}}/\delta_{\text{max}} \\
\text{T5}
\end{align*}
\]

Time lags:

\[
\begin{align*}
\delta_{\text{min}} \\
\delta_{\text{max}}
\end{align*}
\]

- Setup times
- max-latency csts
The target problem

Task Graph

- Deadline and release times can be specified for each task
- We assume a resource allocation is provided by an external tool
Base algorithm: **Depth First Search**

Each node represents a **partial schedule** (i.e. set of precedence relations and time windows):

- Temporally feasible
- Resource infeasible

---

**Diagram:**

```
ENFORCE TEMPORAL FEASIBILITY

 IDENTIFY A RESOURCE INFEASIBILITY

 exit

 RESOLVE RESOURCE INFEASIBILITY
```
Issue #1: temporal consistency of the augmented network

Temporal model $\subseteq$ STN with Uncertainty [Vidal,Fargier ‘99]

- Event variables:
  - $[l,u]$  
  - $\exists d \in [l,u], l \geq 0$
- Free constraint:
  - $\forall d \in [l,u], l \geq 0$
- Contingent constraint:
  - $[l:u]$

Domain = time window
**Issue #1:** temporal consistency of the augmented network

Temporal model ≤ STN with Uncertainty [Vidal, Fargier ‘99]
Constraint propagation:

- When the domain of an event variable is modified, filtering algorithms remove probably infeasible values from the domain of other event variables.
Temporal Consistency

Constraint propagation:
- When any domain gets empty, the search process has met a dead-end and a backtrack occurs.
Resource consistency: Minimal Critical Sets

Critical Sets
\{A_1, A_5\}, \{A_3, A_4, A_5\},
\{A_3, A_4\}, \{A_4, A_5\}

A Minimal Critical Set is **resolved** by posting a new precedence cst.

No MCS = resource consistency

**Minimal Critical Set** = minimal set of possibly overlapping tasks, collectively overusing a resource
Issue #2: efficient MCS identification

the **MCSs are Independent Sets** on the augmented graph =>
in exponential number

=> **In principle**, NP-complete identification

**In practice**, the graph has a transitive orientation

=> Finding the max. weight I.S. Set is ∈ P [Golumbic04]
Issue #2: efficient MCS identification

Cast to Minimum Flow problem:

- Solve via inverse Ford-Fulkerson

\[
\text{Flow} = W(\text{MaxWIS}) = W(\text{MaxCS})
\]

\[
\text{S/T cut} = \text{MaxWIS} = \text{MaxCS}
\]

Simple and fast:
- Based on the temporal graph
- No need to check usage at specific time instants
- **Incremental**: min flow used to prime the next search node
Issue #2: efficient MCS identification

the **MCSs are Independent Sets** on the augmented graph =>
in exponential number

=> **In principle**, **NP-complete** identification

**In practice**, the graph has a transitive orientation

=> Finding the max. weight I.S. Set is ∈ P [Golumbic04]

Additionally:
We managed to apply some filtering algorithms for resource csts.
- Insufficient to check consistency on their own
- Sufficient to remove part of the probably infeasible values
Issue #2: efficient MCS identification

the **MCSs are Independent Sets** on the augmented graph => in exponential number

=> **In principle**, NP-complete identification

**In practice**, the graph has a transitive orientation

=> Finding the max. weight I.S. Set is ∈ P [Golumbic04]

Still an issue:

The MCS is not minimal and may be bad for branching

- **Greedy minization**: iteratively remove activities according to a heuristic quality estimate
- Chosen heuristics: preserved space [Laborie ‘95]
Given a **target MCS**, branch over **possible resolvers**

- **Left branch:** post a resolver
- **Right branch:** forbid a resolver

**Search stops** as soon as a resource **feasible** schedule is found (no optimization is performed at this stage)

\[
\text{MCS} = \{T_4, T_5\}
\]
• Optimization is performed by solving a **sequence of feasibility problems**
• A constraint on the chosen objective is iteratively tightened following a binary search scheme
Experimental Setting

Implemeted on IBM-ILOG Solver 6.7, time limit 600s

Testing problem:
• For every activity $d_{\text{min}} \leq d_{\text{max}} \leq 2 \times d_{\text{min}}$
• Platform:
  • 16 cores (unary, i.e. cap 1)
  • 2 communication ports per core
• Each task requires 1 unary resource
• Inter-task communications require 10% to 90% of the bandwidth of an input port and of an output port
• Instances with $\approx 80$ activities, growing branching factor

Objective function: worst case execution time
Solution time, best WCET

- Solution time [sec]
- Branching factor

- Branching factor:
  - 2-4
  - 3-5
  - 4-6
  - 5-7
  - 6-8

- Solution times:
  - 2-4: 2.25 sec
  - 3-5: 1.5 sec
  - 4-6: 1.5 sec
  - 5-7: 1.5 sec
  - 6-8: 1.5 sec
Objective function: the completion time corresponding to the scenario where all tasks have average duration

The minimum application WCET was used as a global deadline

- Goal: assess the quality of the PCP schedule in terms of average completion time
- Our PCP approach was compared against:
  - Fixed Priority Scheduling, optimized via Tabu Search
  - On-line FIFO scheduling
- Experiments on 10 instances from the previous benchmark
- Evaluation via Monte-Carlo simulation
Average completion time minimization

![Chart showing average completion time for PCP, FPS, and FIFO algorithms across different branching factors.](chart-image)
Average completion time minimization
Average completion time minimization
Robust scheduling of task graphs under execution time uncertainty

Michele Lombardi
Michela Milano
Luca Benini
More advanced on-line schedulers

The on-line scheduler must:
• Start a task when the predecessors are over
• Start a task within the restricted time window

The on-line scheduler can be smarter than this:
• The scheduler can decide how much to delay a task (or not to delay it at all)
• The scheduler can resolve resource conflicts on-line

One more ingredient: Self Timed Execution (STE)
“activities are started by the dispatcher as soon as they are ready”
• Reduces the number of possibly occurring conflicts
• Introduces some complications in the solution algorithm
Some MCS should not be taken into account

Conflict Sets (no S.T.E.):
\{A_1, A_5\}, \{A_3, A_4, A_5\},
\{A_4, A_5\}, \{A_4, A_5\}

Conflict Sets (with S.T.E.):
\{A_3, A_4, A_5\}, \{A_4, A_5\}, \{A_4, A_5\}

Conflict \{A_1, A_5\} never occurs

BUT:
Some MCS should not be taken into account

Conflict Sets (no S.T.E.):
\{A_1, A_5\}, \{A_3, A_4, A_5\},
\{A_4, A_5\}, \{A_4, A_5\}

Conflict Sets (with S.T.E.):
\{A_3, A_4, A_5\}, \{A_4, A_5\}, \{A_4, A_5\}

BUT: Suppose:
1. A_1 uses 2 units of R_1
2. A_2 \rightarrow A_1

new conflicts may arise due to search decisions

Conflict \{A_1, A_5\} never occurs
The graph for the min flow problem is **augmented with the implied precedence relations**, due to the self timed execution policy.

- **The flow value** (maximum weight of a Conflict Set) may reduce.
- Implied arcs may be removed due to search decisions.
- At each search node, the flow value is repaired to accommodate for removed arcs.

**red = req.**
**green = flow**
A1 uses 2 units of R1

- Suppose A₂ is scheduled at 0
- Propagation shifts A₁

A schedule with time gaps cannot be executed:

\[ \text{start}(T_j) = \max_{T_i < T_j} \text{end}(T_i) \]

- The schedule can be fixed by adding a new precedence constraint

Issue 3: **stopping search** requires no conflicts, and no time gaps
• A partial schedule can now be time infeasible
• Time gaps are resolved by adding precedence constraints, similarly to MCSs
Robust scheduling of task graphs under execution time uncertainty

Michele Lombardi
Michela Milano
Luca Benini