On the Scalability of Constraint Solving for Static/Offline Real-Time Scheduling

R. Gorcitz[1] **E. Kofman**[2] D. Potop-Butucaru[2]
R. De Simone[2] Thomas Carle[3]

[1]CNES, France

[2]INRIA, France

[3]Brown University, USA

September 4, 2015 - FORMATS - Madrid

Outline

Motivation

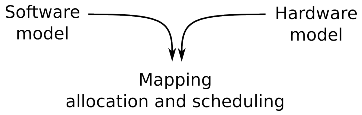


Figure: Y-chart design methodology

Allocating and scheduling tasks and messages over a network of resources is NP-Complete (for most of the problem instances).

Motivation

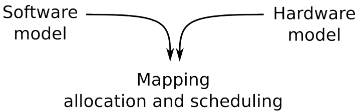


Figure: Y-chart design methodology

Allocating and scheduling tasks and messages over a network of resources is NP-Complete (for most of the problem instances).

Two main approaches:

- Avoiding NP-Completeness: heuristics
- Accepting it: general solvers (Integer Linear Programming / SAT Modulo Theory / Constraint Programming)

Motivation

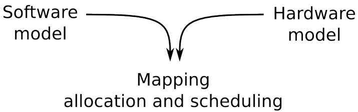


Figure: Y-chart design methodology

Allocating and scheduling tasks and messages over a network of resources is NP-Complete (for most of the problem instances).

Two main approaches:

- Avoiding NP-Completeness: heuristics
- Accepting it: general solvers (Integer Linear Programming / SAT Modulo Theory / Constraint Programming)
- How far?

Motivation

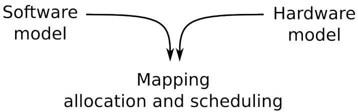


Figure: Y-chart design methodology

Allocating and scheduling tasks and messages over a network of resources is NP-Complete (for most of the problem instances).

Two main approaches:

- Avoiding NP-Completeness: heuristics
- Accepting it: general solvers (Integer Linear Programming / SAT Modulo Theory / Constraint Programming)
- How far? (with "up to date" solvers)
  - Small size problems (can still be interesting)
  - Design for scalability

Motivation

- SAT solvers can handle large problems
  functional model checking (time model checking?)
- It has practical interests in static scheduling (meta-programming) and compilation problems.

We need an empirical evaluation of static distributed real-time scheduling problems. When can we apply exact solving techniques?

Motivation

- SAT solvers can handle large problems
  functional model checking (time model checking?)
- It has practical interests in static scheduling (meta-programming) and compilation problems.

We need an empirical evaluation of static distributed real-time scheduling problems. When can we apply exact solving techniques?

Contributions

- Encoding of a variety of scheduling problems as SMT/ILP/CP problems
- Synthetic and realistic test cases
- Empirical hardness related to various parameters (problem size, preemtiveness, satisfiability or optimization target, system load, dependencies, pipelining, resources homogeneity, unique or multiple task periods ...).

Motivation

Two real-life test cases (FFT and Platooning applications) and synthetic benchmarks.

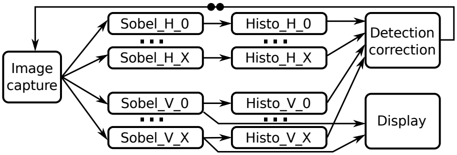


Figure: Platooning dataflow graph application

Motivation

Two real-life test cases (FFT and Platooning applications) and synthetic benchmarks.

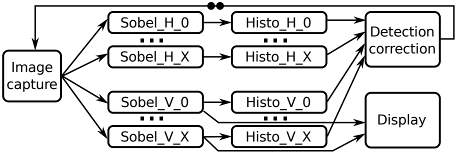
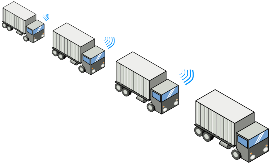


Figure: Platooning dataflow graph application



Reduces traffic jams (and fuel consumption).

Motivation

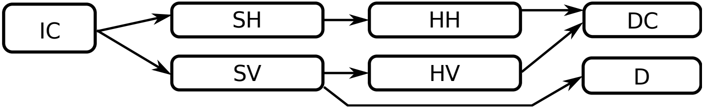


Figure: Platooning dataflow graph application, no pipelining, no data-parallelism (X=0) and simplified labels...

Motivation

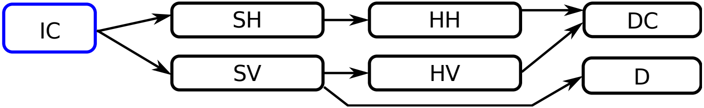


Figure: Platooning dataflow graph application, no pipelining, no data-parallelism (X=0) and simplified labels...

Application constraints:

- $IC_{Stop} = IC_{Start} + IC_{Duration}$

Motivation

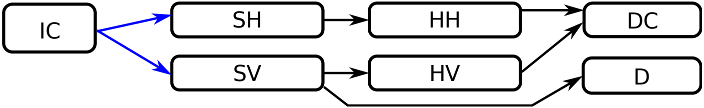


Figure: Platooning dataflow graph application, no pipelining, no data-parallelism (X=0) and simplified labels...

Application constraints:

- $IC_{Stop} = IC_{Start} + IC_{Duration}$
- $SH_{Start} \geq IC_{Stop}$
- $SV_{Start} \geq IC_{Stop}$

Motivation

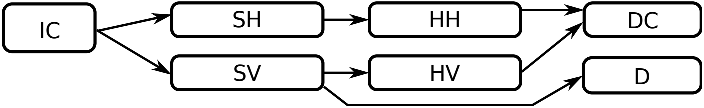


Figure: Platooning dataflow graph application, no pipelining, no data-parallelism (X=0) and simplified labels...

Application constraints:

- $IC_{Stop} = IC_{Start} + IC_{Duration}$
- $SH_{Start} \geq IC_{Stop}$
- $SV_{Start} \geq IC_{Stop}$
- ...

Motivation

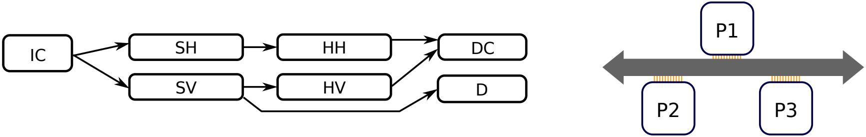


Figure: Platooning dataflow graph application single period, no data-parallelism (X=0) and simplified labels...

Motivation

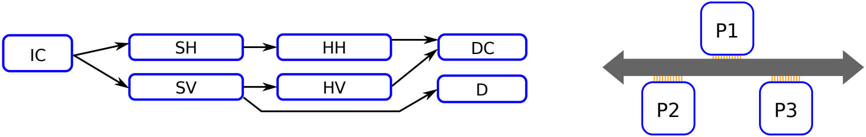


Figure: Platooning dataflow graph application single period, no data-parallelism (X=0) and simplified labels...

Resource constraints:

- $IC_{Map} \in \{P1, P2, P3\}, SH_{Map} \in \{P1, P2, P3\}, ...$

Motivation

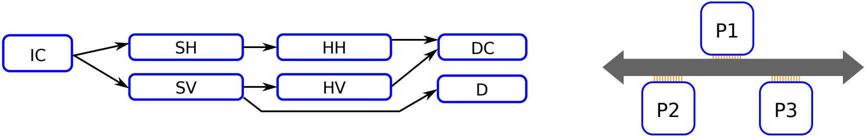


Figure: Platooning dataflow graph application single period, no data-parallelism (X=0) and simplified labels...

Resource constraints:

- $IC_{Map} \in \{P1, P2, P3\}, SH_{Map} \in \{P1, P2, P3\}, \ldots$
  $SH_{Map} == SV_{Map} \rightarrow SH_{Start} \geq SV_{Stop} \oplus SH_{Stop} \geq SV_{Start}$
- ...

Motivation

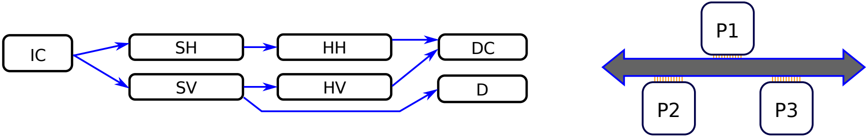


Figure: Platooning dataflow graph application single period, no data-parallelism (X=0) and simplified labels...

Resource constraints:

- $IC_{Map} \in \{P1, P2, P3\}, SH_{Map} \in \{P1, P2, P3\}, \ldots$
  $SH_{Map} == SV_{Map} \rightarrow SH_{Start} \geq SV_{Stop} \oplus SH_{Stop} \geq SV_{Start}$
- ...

The encoding is a set of linear constraints and a set of decision variables.

Motivation

Realistic modeling: We also associated a time with messages (communication between tasks) with the following rules:

- Messages cannot overlap (because of the single Bus)
- Messages have no cost if the source task and destination tasks are allocated on the same resource (shared memory, you only need to pass a pointer, disputable)

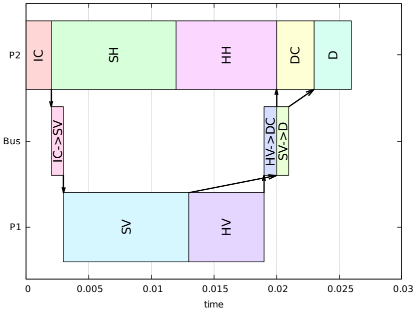


Figure: Gantt diagram (optimal non-preemptive schedule)

Modeling and solving

in plain English

1. Each task is allocated on exactly one processor
2. If two tasks are ordered, the second starts after the first ends
3. If two tasks are allocated on the same processor, they must be ordered
4. The source and destination of a dependency must be ordered
5. The bus communication associated with a dependency (if any) must start after the source task ends and must end before the destination task starts
6. When two dependencies require both a bus communication, these communications must be ordered
7. If two dependencies are ordered, the first must end before the second starts
8. All tasks must end at a date smaller or equal than the schedule length
9. ...

Modeling and solving

We look at:

- Average solving time
- Timeout (1 hour) ratio

When we vary:

Scheduling problem single or multi period, preemptive or not, resource homogeneity

Objective schedulability analysis vs optimization

Average system loads 25%, 75%, 87.5%,125%

Pipelining depth

Onto:

- Synthetic test cases
- Two realistic test cases (FFT and Platooning applications)

Modeling and solving

Why merge the two ?

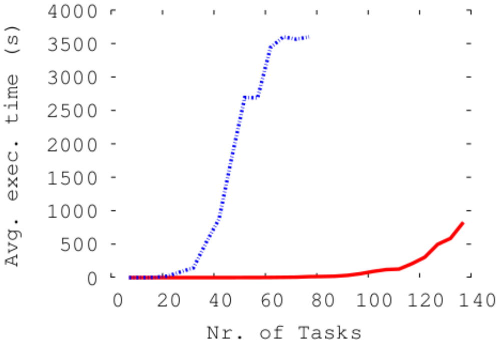


Figure: Single Period Non-Preemptive vs Multi Period Preemptive

Modeling and solving

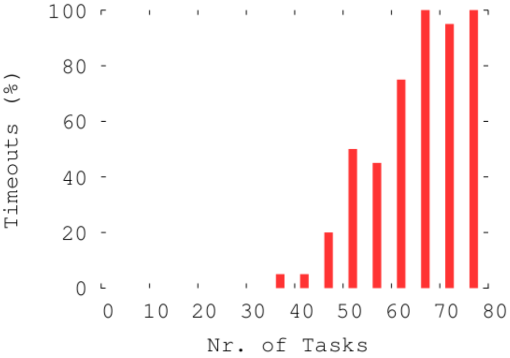


Figure: Timeout (1h) rate for preemptive multi-periodic

→ Preemptive problems are of greater complexity

→ Non-preemptive scheduling is actually one of the first motivations of this work.

Modeling and solving

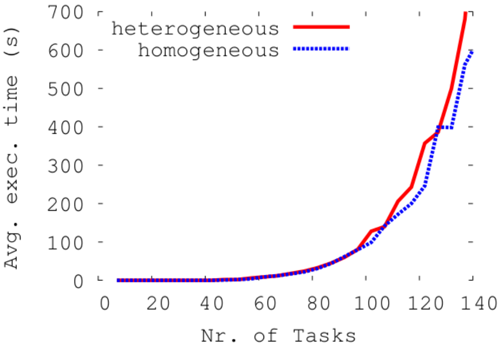


Figure: Homogeneous resources VS Heterogeneous resources

→ Not so clear

Modeling and solving

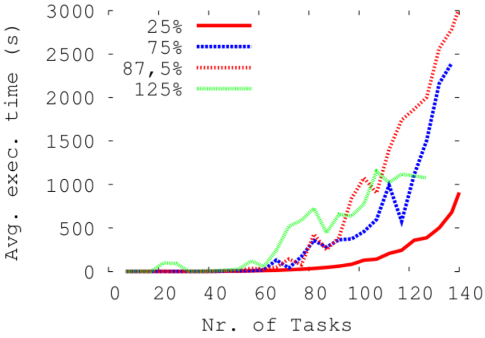


Figure: System load

→ Underloaded and overloaded systems are easier to handle.

Modeling and solving

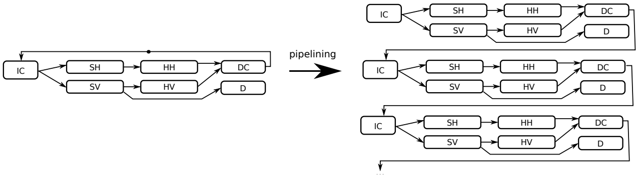


Figure: Pipelining: Cyclic dataflow graph to DAG

Pipelining raises the depth of the graph

Modeling and solving

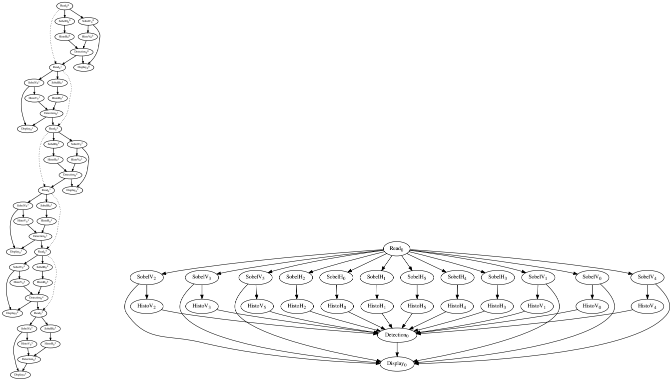


Figure: “Aspect” of the graph

$\rightarrow$ Deep graphs will scale much better than large graphs

Modeling and solving

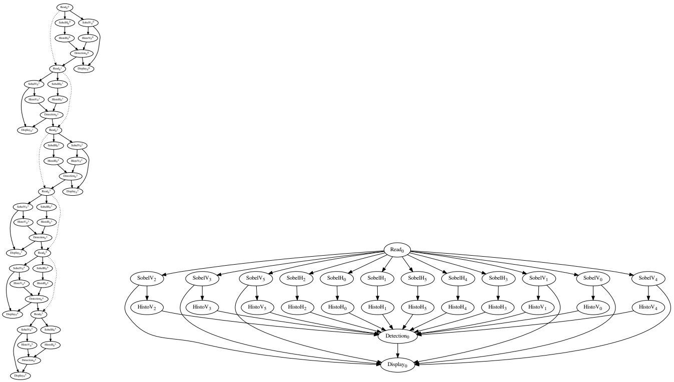


Figure: "Aspect" of the graph

$\rightarrow$ Deep graphs will scale much better than large graphs
Even with symmetry breaking techniques

Modeling and solving

Good practices for encoding scheduling problems:

- The order of the variables matters: somehow ask the solver to select the variables (eq. the tasks) in the topological order of the tasks, beginning with the small values of the variable state.
- Non-synthetic graphs can exhibit a lot of symmetry (e.g. split merge graph). Associating variables (T_{Map} and T_{Start}) to each task of a split/merge is certainly not scalable $\rightarrow$ Symmetry breaking techniques
- Symmetries can also occur in architecture graph (if homogeneous resources) $\rightarrow$ Symmetry breaking techniques (cumulative scheduling).

Modeling and solving

Thank you

Modeling and solving

Thank you

Questions ?