Decomposition approaches for a large-scale scheduling problem

Elina Rönnberg
Department of Mathematics, LiU

What?

A challenging scheduling problem

- Electronic systems in aircraft
- Multiprocessor scheduling with precedence relations between tasks
  (lb and ub on time lag)
  & communication network
- Feasibility: find a schedule or prove that none exists

Of importance for development of future aircraft

Why?

- Make sure software functions are assigned the hardware resources they need: run applications & pass data
- Configurable system—each configuration needs a schedule
- Part of design process:
  - iterative development and changes
  - changes → new schedule
  - updates during whole life-time (decades)

Why?

- Make sure software functions are assigned the hardware resources they need: run applications & pass data
- Configurable system—each configuration needs a schedule
- Part of design process:
  - iterative development and changes
  - changes → new schedule
  - updates during whole life-time (decades)

- Scheduling fails → costly design changes

Why?

- Make sure software functions are assigned the hardware resources they need: run applications & pass data
- Configurable system—each configuration needs a schedule
- Part of design process:
  - iterative development and changes
  - changes → new schedule
  - updates during whole life-time (decades)

- Scheduling fails → costly design changes
- Heuristic approaches not suitable to determine feasibility

Why?

- Make sure software functions are assigned the hardware resources they need: run applications & pass data
- Configurable system—each configuration needs a schedule
- Part of design process:
  - iterative development and changes
  - changes → new schedule
  - updates during whole life-time (decades)

- Scheduling fails → costly design changes
- Heuristic approaches not suitable to determine feasibility
- Large-scale problem: Out of reach for generic DO solvers

Who?

Close collaboration between Linköping University and Saab

- LiU side: Optimisation perspective
- Saab side: Technical perspective
- Joint team
  - Elina Rönnberg (research leader, LiU & Saab)
  - Robert Petersson (technical leader at Saab)
  - Emil Karlsson (PhD student, LiU & Saab)
  - Andreas Stenberg (software developer, Saab)
  - Hannes Uppman (algorithm developer, Saab)
  - some more software developers / system engineers at Saab
  - steering group with managers and technical fellows

How?

Pre-runtime scheduling tool = method development

- Modelling + decomposition approaches that exploit
  - problem structure
  - power of generic solvers
- Exploring data ⇒
  - preprocessing
  - adapt decomposition

How?

Pre-runtime scheduling tool = method development

- Modelling + decomposition approaches that exploit
  - problem structure
  - power of generic solvers
- Exploring data ⇒
  - preprocessing
  - adapt decomposition

Today's talk: Overview of two decomposition approaches, the use of MIP vs. CP, and the importance of understanding the data

Outline

Introduction

Technical background

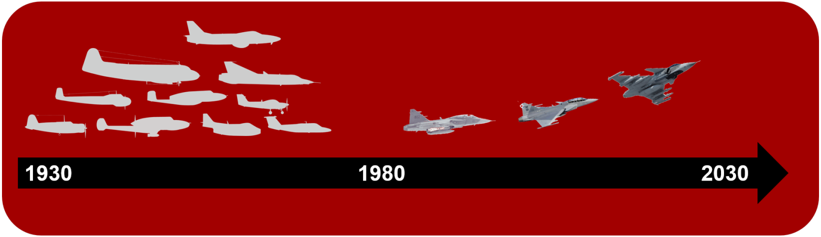
Problem formulation

Decomposition approaches

Concluding comments

Aircraft development

From conquering the laws of physics—
to becoming computers with wings



Avionics

Electronics in an aircraft

- sensors that collect information
- units where the information is processed
- actuators that control the aircraft
- equipment that presents information to the pilot

Prescribe—down to the nanosecond—what the electronics does

Avionics design

Making sure that the system can be trusted is key

Avionics design

Making sure that the system can be trusted is key

Examples of aspects:

- Create subsystems that can be validated independently
- Prevent faults from propagating between functions
- All possible scenarios are covered and evaluated
- Information is correct and protected from unauthorized access

Extensive documentation, testing and certification processes

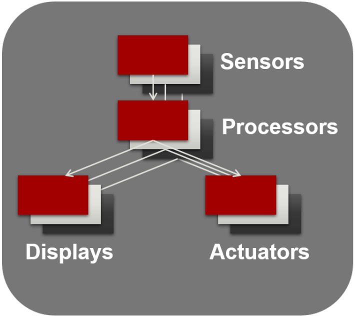
Classic design

Federated system

- Each function has a separate hardware
- A hard-wired system

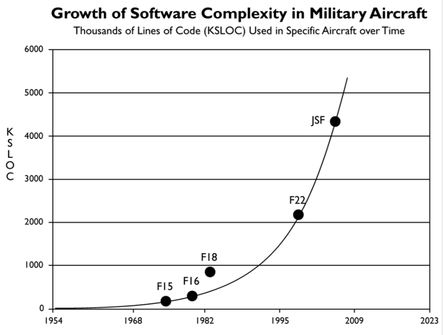
+ Simple integration and verification

– Limited synergy and system integration



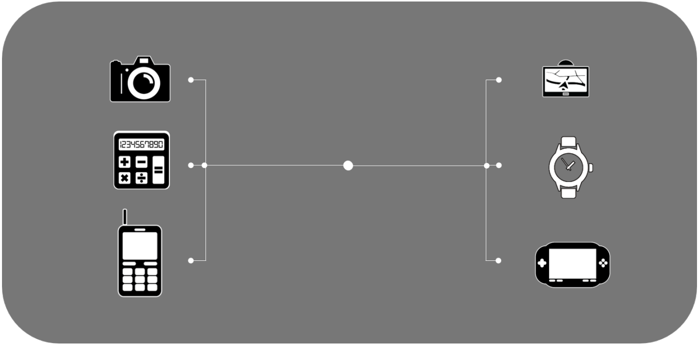
Times are changing

- Digital systems and computers introduced new possibilities
- System complexity increases over time
- New needs:
  - upgradable
  - adaptable
  - reconfigurable

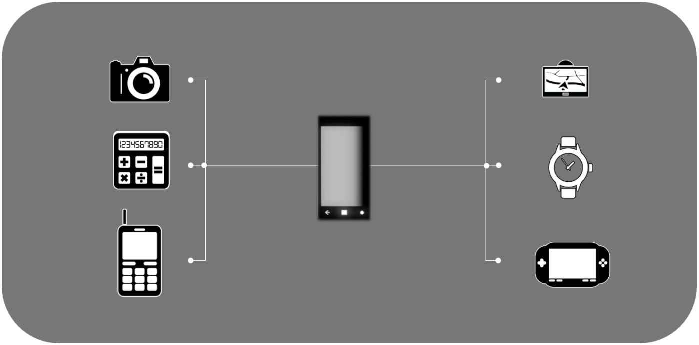


https://savi.avsi.aero/about-savi/savi-motivation/exponential-system-complexity/

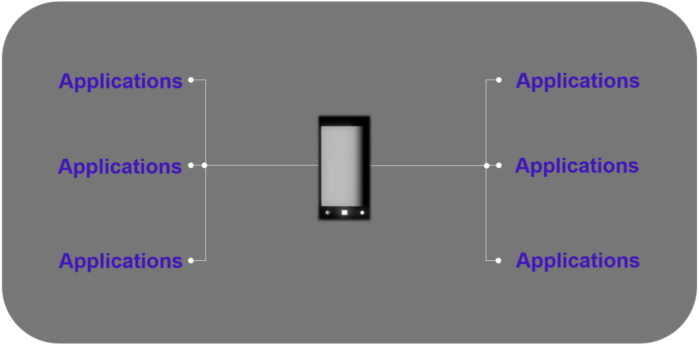
Change of design philosophy—an analogy



Change of design philosophy—an analogy



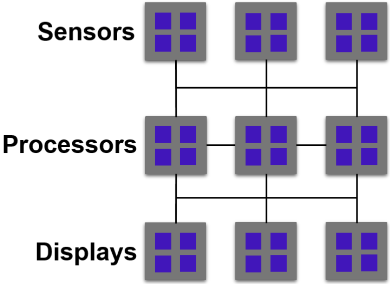
Change of design philosophy—an analogy



Modern design

IMA: Integrated modular avionics

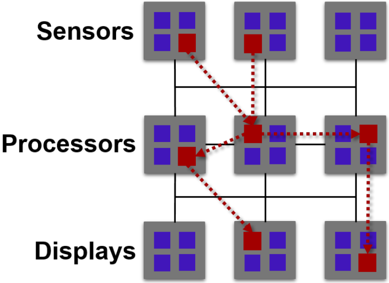
- Shared hardware



Modern design

IMA: Integrated modular avionics

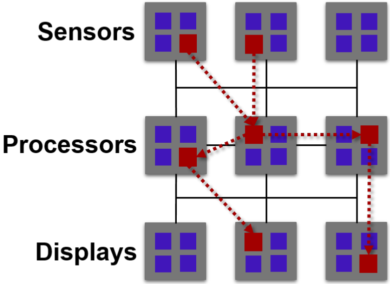
- Shared hardware
- Software defines the functionality



Modern design

IMA: Integrated modular avionics

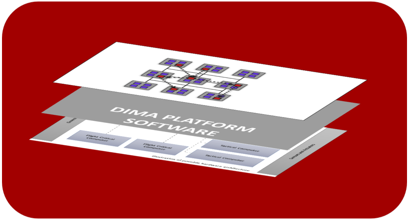
- ▶ Shared hardware
- ▶ Software defines the functionality
- + Facilitates synergies and integration
- – Complex integration and verification



Separation of software and hardware

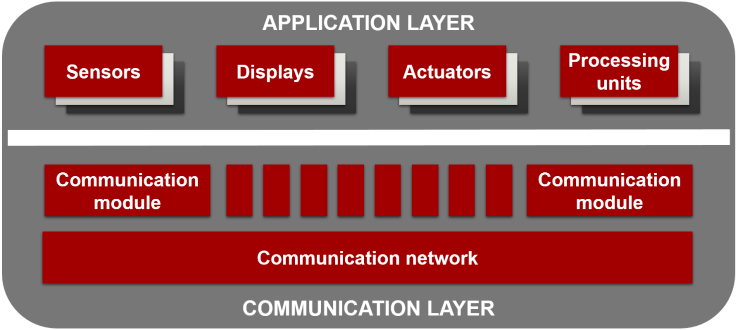
Three independent layers

- Software
- "The glue"
  Code, tools, tests
- Hardware



One responsibility of "the glue" is to allocate hardware resources to the software processes—and make sure the system can be trusted

SAAB avionics design case



APPLICATION LAYER

Sensors

Displays

Actuators

Processing units

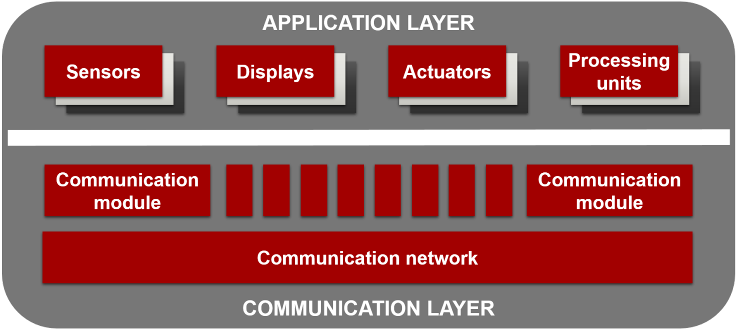
Communication module

Communication module

Communication network

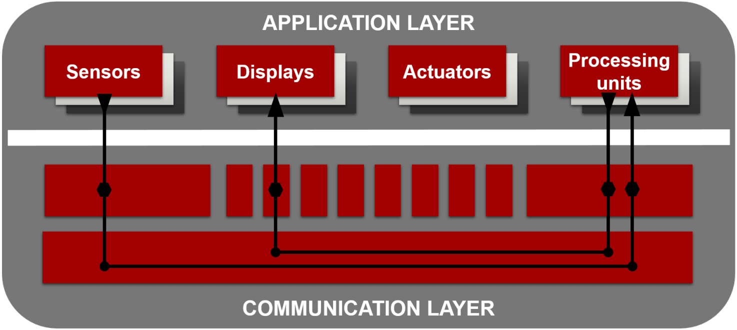
COMMUNICATION LAYER

SAAB avionics design case



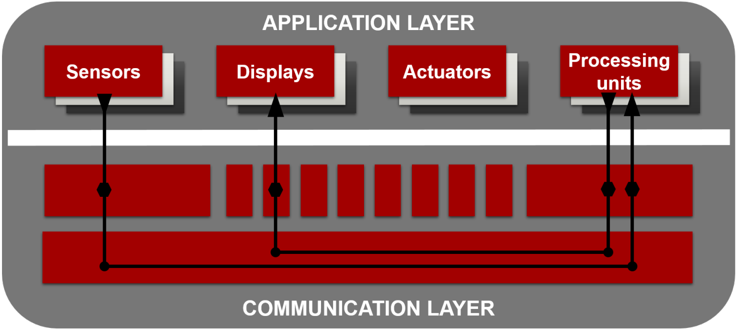
- Application layer: Application development view
- Communication layer:
  Infrastructure to provide communication

SAAB avionics design case



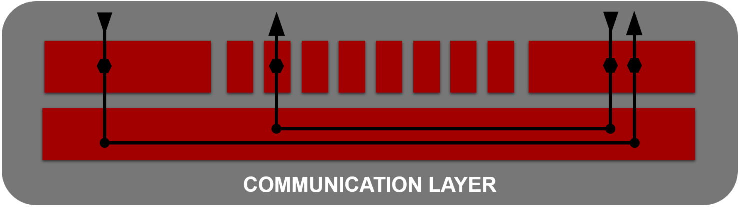
- Application layer: Application development view
- Communication layer:
  Infrastructure to provide communication

SAAB avionics design case



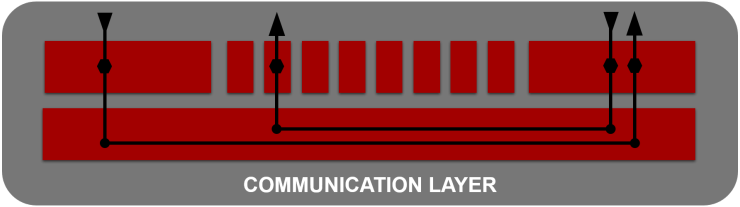
- Completely synchronous system
- Each communication activity is explicitly scheduled and data is available at a determined point in time

SAAB avionics design case: communication details



- Communication network:
  Switched Ethernet with messages sent in discrete time slots
  - Access to the full bandwidth at that instant
  - Can guarantee fast communication
  - Schedule determine when data will arrive
  - Mulitcast

SAAB avionics design case: communication details



- Communication network:
  Switched Ethernet with messages sent in discrete time slots
  - Access to the full bandwidth at that instant
  - Can guarantee fast communication
  - Schedule determine when data will arrive
  - Mulitcast
- Communication modules: Chains of tasks to be scheduled

SAAB avionics design case: characteristics

- Independence between different applications by
  - a pre-runtime schedule with start times for all activities
  - known worst-case execution times for all activities
  - spatial partitioning (not part of scheduling)

SAAB avionics design case: characteristics

- Independence between different applications by
  - a pre-runtime schedule with start times for all activities
  - known worst-case execution times for all activities
  - spatial partitioning (not part of scheduling)
- Applications developed, verified, and simulated in isolation

SAAB avionics design case: characteristics

- Independence between different applications by
  - a pre-runtime schedule with start times for all activities
  - known worst-case execution times for all activities
  - spatial partitioning (not part of scheduling)
- Applications developed, verified, and simulated in isolation
- Adaptable by design: Easy to upgrade and reconfigure

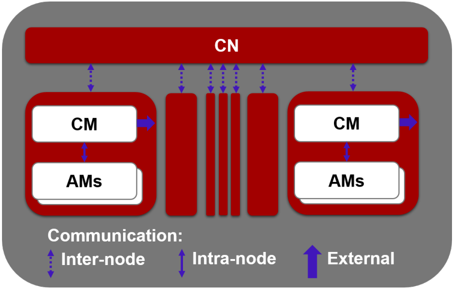
SAAB avionics design case: characteristics

- Independence between different applications by
  - a pre-runtime schedule with start times for all activities
  - known worst-case execution times for all activities
  - spatial partitioning (not part of scheduling)
- Applications developed, verified, and simulated in isolation
- Adaptable by design: Easy to upgrade and reconfigure
- Highly advanced ”glue” between software and hardware

SAAB avionics design case: characteristics

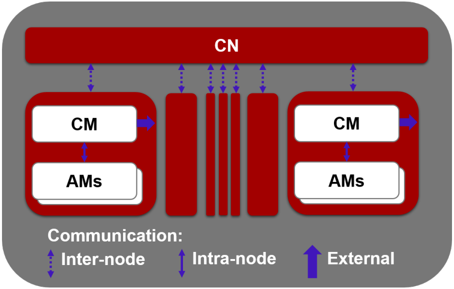
- Independence between different applications by
  - a pre-runtime schedule with start times for all activities
  - known worst-case execution times for all activities
  - spatial partitioning (not part of scheduling)
- Applications developed, verified, and simulated in isolation
- Adaptable by design: Easy to upgrade and reconfigure
- Highly advanced ”glue” between software and hardware
- Complex scheduling
  - All activities at once
  - Communication scheduling intricate and detailed

From system design to scheduling



- Communication network (CN)
  Communication between nodes
- Application module (AM)
  Run software processes
- Communication module (CM)
  Handle communication

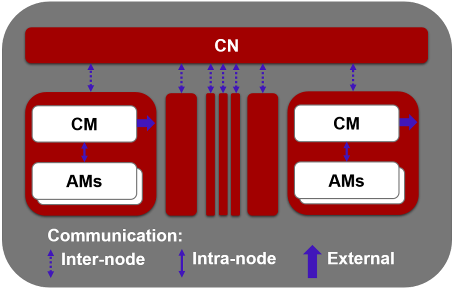
From system design to scheduling



- Communication network (CN)
  Communication between nodes
- Application module (AM)
  Run software processes
- Communication module (CM)
  Handle communication

- Cyclic schedule
- One cycle = a major frame (system period) ∼1s long

From system design to scheduling



- Communication network (CN)
  Communication between nodes
- Application module (AM)
  Run software processes
- Communication module (CM)
  Handle communication

- Cyclic schedule
- One cycle = a major frame (system period) $\sim$1s long
- Time resolution in nanoseconds
- Find a **feasible** solution or deduce that none exists

Multi-processor scheduling with ...

Scheduling of periodic tasks

- AMs: Few tasks, several instances per major frame
- CMs: Huge number of tasks, one instance per major frame

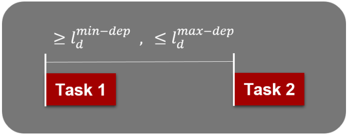
Multi-processor scheduling with ...

Scheduling of periodic tasks

- AMs: Few tasks, several instances per major frame
- CMs: Huge number of tasks, one instance per major frame
- Dependencies

Dependency:

- Precedence relation with time lag
- On the same or on different modules



Multi-processor scheduling with …

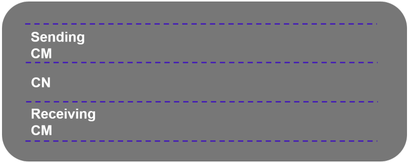
Scheduling of periodic tasks

- AMs: Few tasks, several instances per major frame
- CMs: Huge number of tasks, one instance per major frame
- Dependencies

Sequence and assign start time

Multi-processor scheduling with ...

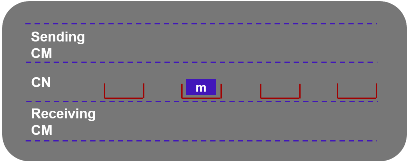
Scheduling of communication



Multi-processor scheduling with …

Scheduling of communication

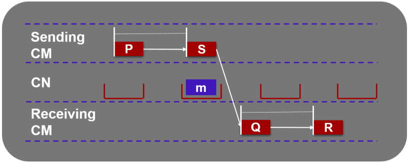
- Messages are sent in discrete time slots



Multi-processor scheduling with …

Scheduling of communication

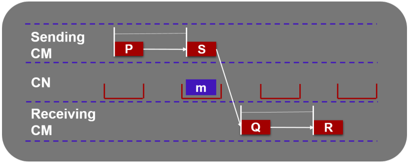
- Messages are sent in discrete time slots
- A message involves tasks restricted by dependencies



Multi-processor scheduling with ...

Scheduling of communication

- Messages are sent in discrete time slots
- A message involves tasks restricted by dependencies

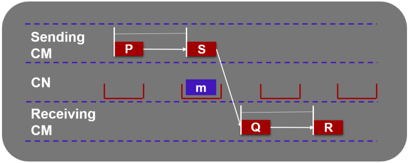


Choose a time slot for each message

Multi-processor scheduling with …

Scheduling of communication

- Messages are sent in discrete time slots
- A message involves tasks restricted by dependencies

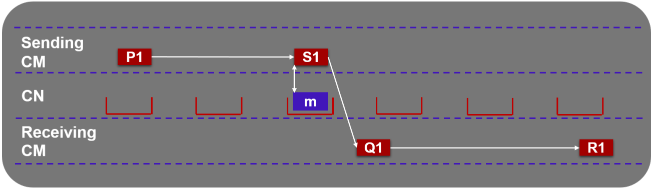


Choose a time slot for each message

BUT choice of time slot ⇒
additional restrictions on the involved tasks …

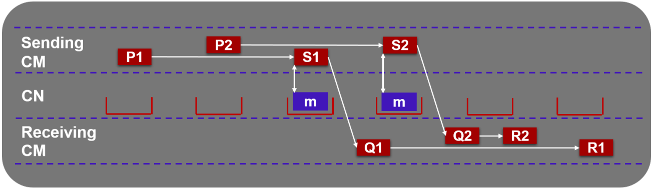
Interaction between task and communication scheduling

- The choice of time slot impacts the release times and deadlines of some tasks



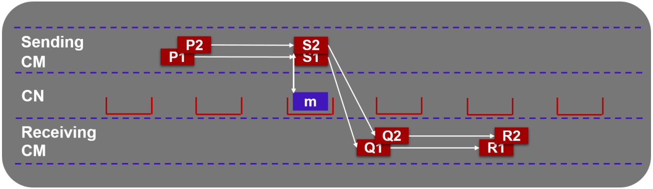
Interaction between task and communication scheduling

- The choice of time slot impacts the release times and deadlines of some tasks
- Same relative order between messages and some of the tasks

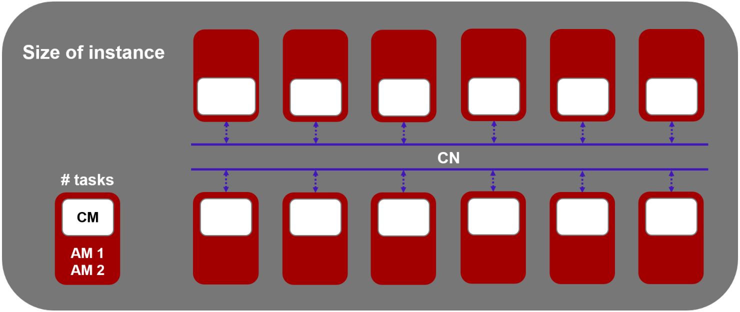


Interaction between task and communication scheduling

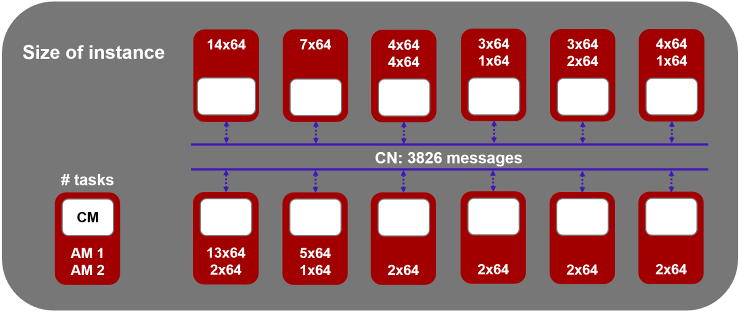
- The choice of time slot impacts the release times and deadlines of some tasks
- Same relative order between messages and some of the tasks
- Co-allocation of messages ⇒ merging of tasks



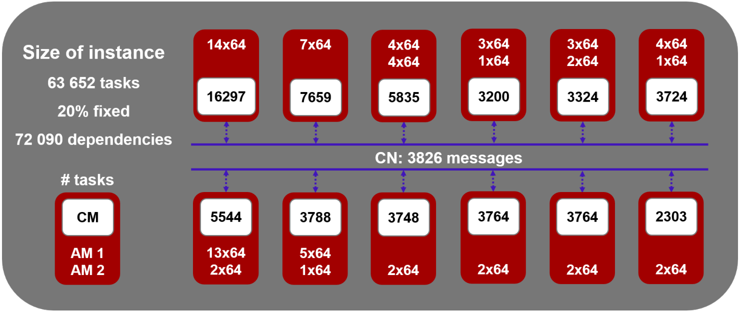
Goal: instance of practical relevance for future design



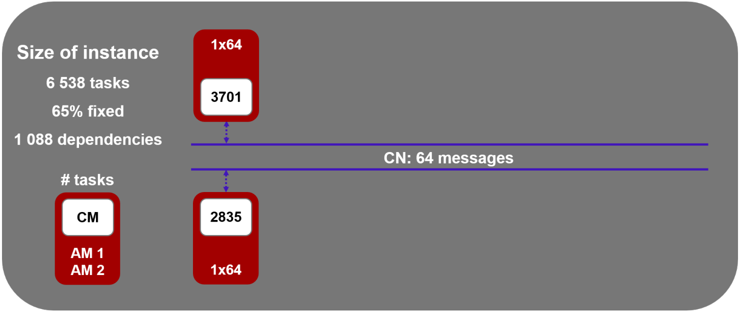
Goal: instance of practical relevance for future design



Goal: instance of practical relevance for future design



Point of departure



Full MIP model: not solved within a week

Tried different MIP and CP approaches for small scale problems

Problem analysis

Main computational challenges

- Interaction between task and communication scheduling
  - ”Messy” part of the model
  - Impact on the task sequencing

Problem analysis

Main computational challenges

- Interaction between task and communication scheduling
  - ”Messy” part of the model
  - Impact on the task sequencing
- Sequence tasks on the CMs:

Problem analysis

Main computational challenges

- Interaction between task and communication scheduling
  - "Messy" part of the model
  - Impact on the task sequencing
- Sequence tasks on the CMs:
  - Huge number of tasks: $> 15,000$ on a single module

    A magnitude of 100 million "Task *i* before *j*" decisions

Problem analysis

Main computational challenges

- Interaction between task and communication scheduling
  - ''Messy'' part of the model
  - Impact on the task sequencing
- Sequence tasks on the CMs:
  - Huge number of tasks: $> 15,000$ on a single module
    A magnitude of 100 million ''Task *i* before *j*'' decisions
  - Long scheduling horizon: 10^9 time points
    A magnitude of 10^{12} ''Task *i* start at time *t*'' decisions

Problem analysis

Important design considerations

- Find a feasible schedule or prove that none exists

Problem analysis

Important design considerations

- Find a feasible schedule or prove that none exists
- Interaction between communication & tasks

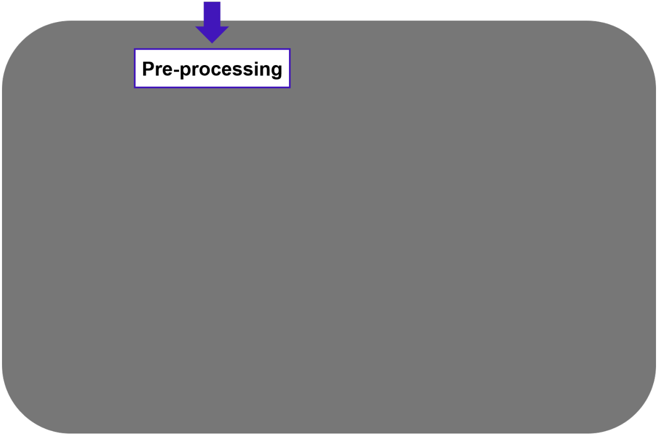
Problem analysis

Important design considerations

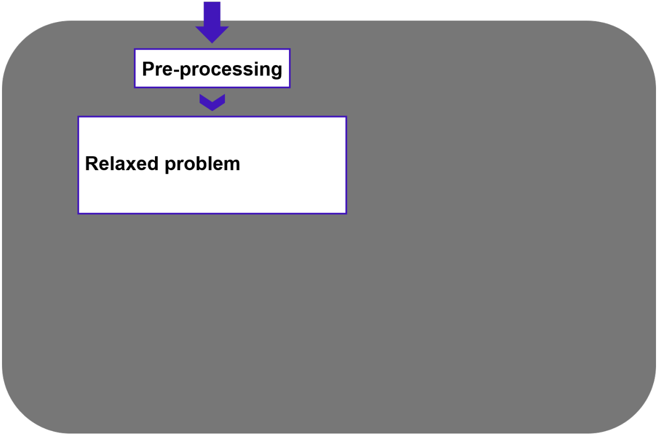
- Find a feasible schedule or prove that none exists
- Interaction between communication & tasks
- Need to handle the sequencing
  MIP: Time-indexed formulations are not viable,
  does order-based formulations stand a chance?
  CP: Better equipped for such sequencing?

Good news: Problem structure to exploit!

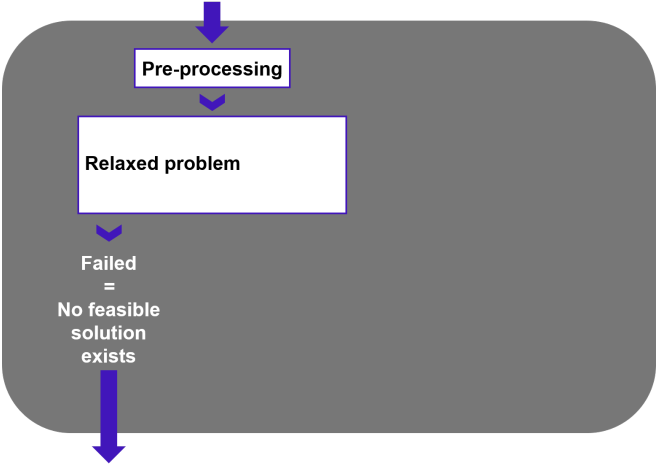
Promising type of decomposition scheme


Pre-processing

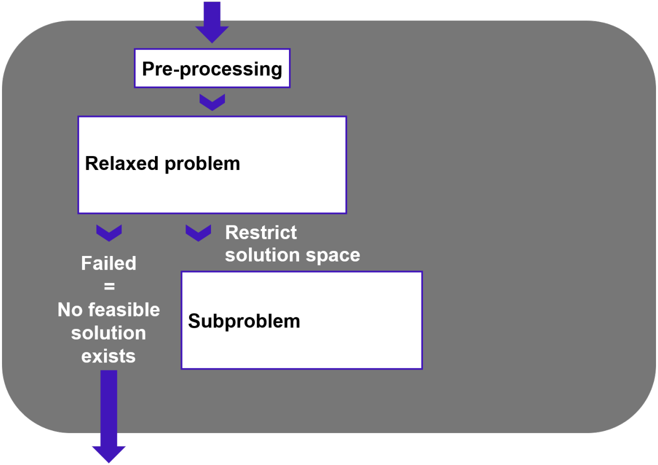
Promising type of decomposition scheme



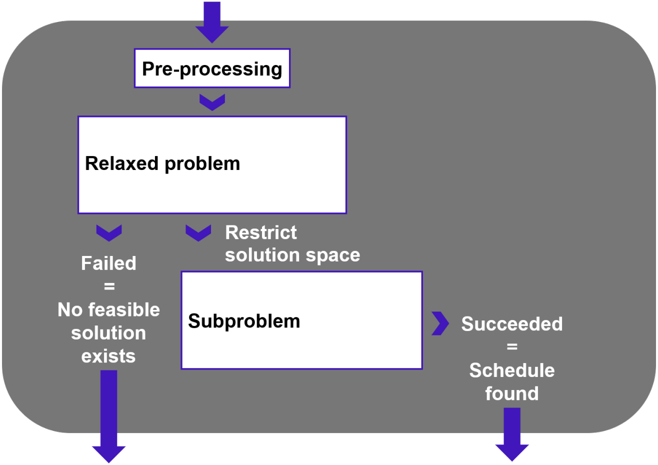
Promising type of decomposition scheme



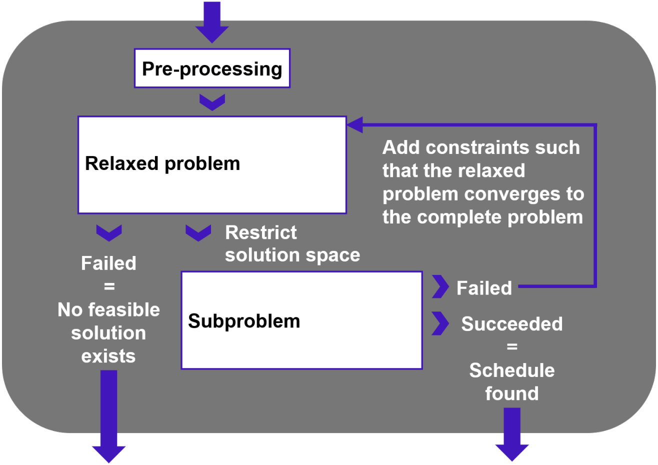
Promising type of decomposition scheme



Promising type of decomposition scheme


Pre-processing
Relaxed problem
Restrict solution space
Failed = No feasible solution exists
Subproblem
Succeeded = Schedule found

Promising type of decomposition scheme



Pre-processing

Relaxed problem

Add constraints such that the relaxed problem converges to the complete problem

Restrict solution space

Failed = No feasible solution exists

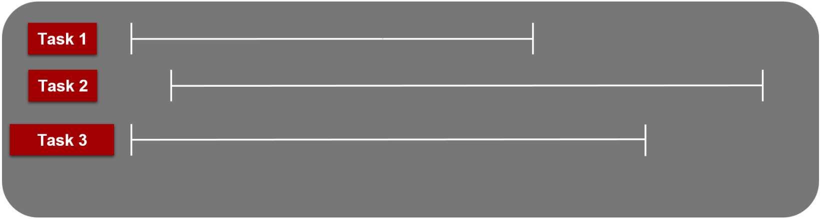
Subproblem

Failed

Succeeded = Schedule found

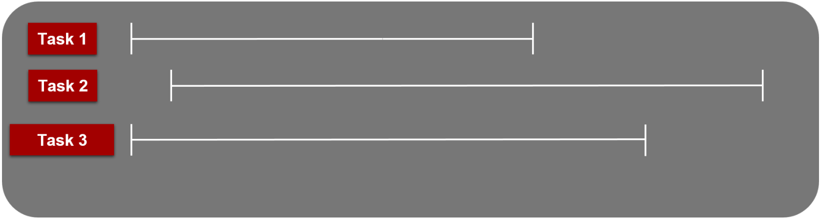
Problem structure: sequencing of tasks on CMs

Original data: Each task has a release time and deadline



Problem structure: sequencing of tasks on CMs

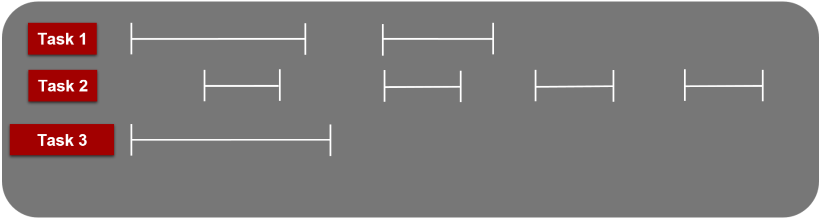
Original data: Each task has a release time and deadline



$$\left.\begin{array}{c}\text{Technical restrictions: Discrete times, fixed tasks, ...}\\ +\\ \text{Pre-processing: Propagate precedence relations and time lags}\end{array}\right\}$$

Problem structure: sequencing of tasks on CMs

Original data: Each task has a release time and deadline



Technical restrictions: Discrete times, fixed tasks, ...

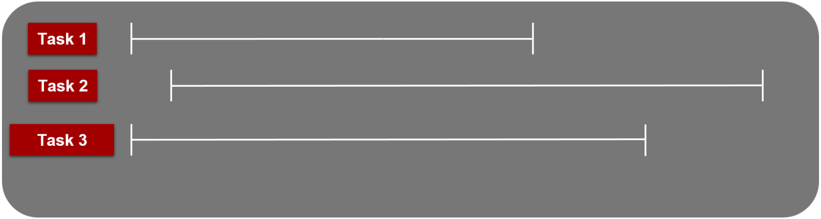
+

Pre-processing: Propagate precedence relations and time lags }

⇒ Only some sub-intervals within the interval between release time and deadline are feasible

Problem structure: sequencing of tasks on CMs

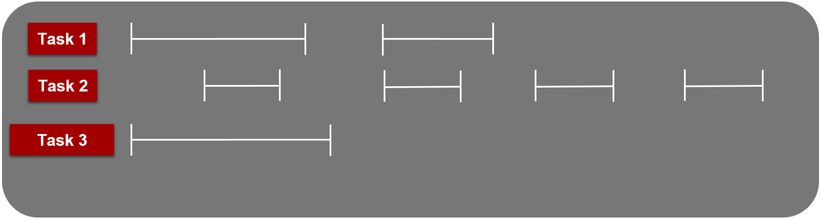
Impact of these sub-intervals?



"Task *i* before *j*" decisions

Problem structure: sequencing of tasks on CMs

Impact of these sub-intervals?

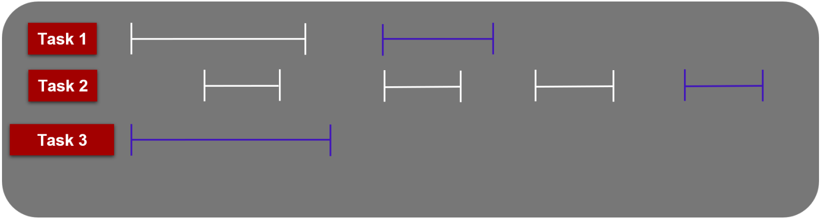


"Task i before j" decisions

- Reduces to $\sim 1/10$ when considering the sub-intervals

Problem structure: sequencing of tasks on CMs

Impact of these sub-intervals?

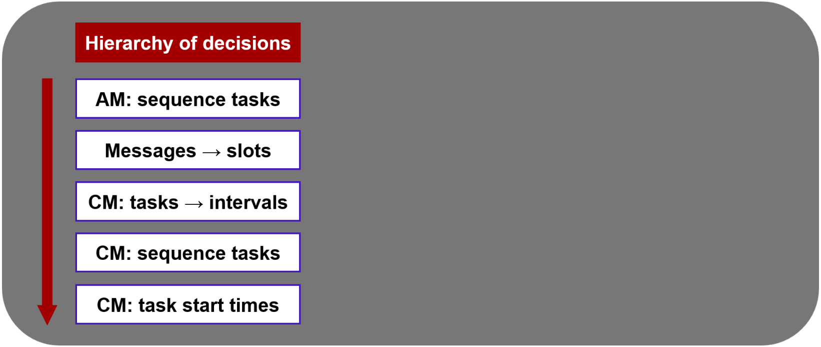


"Task *i* before *j*" decisions

- Reduces to $\sim 1/10$ when considering the sub-intervals
- Reduces to $\sim 1/10$ for a fixed assignment of sub-intervals

Model structure

Including the knowledge about the task sub-intervals— the model can be seen as:

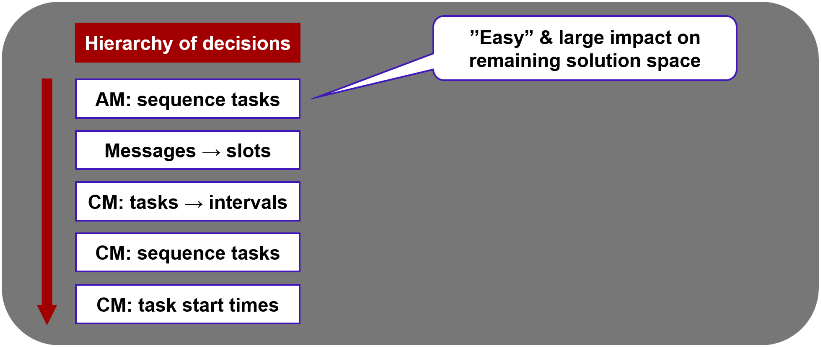


Hierarchy of decisions

- **AM: sequence tasks**
- **Messages → slots**
- **CM: tasks → intervals**
- **CM: sequence tasks**
- **CM: task start times**

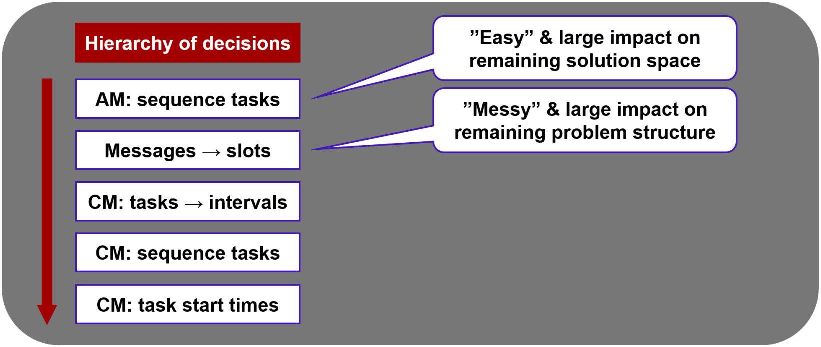
Model structure

Including the knowledge about the task sub-intervals—the model can be seen as:



Model structure

Including the knowledge about the task sub-intervals—
the model can be seen as:



Hierarchy of decisions

AM: sequence tasks

Messages → slots

CM: tasks → intervals

CM: sequence tasks

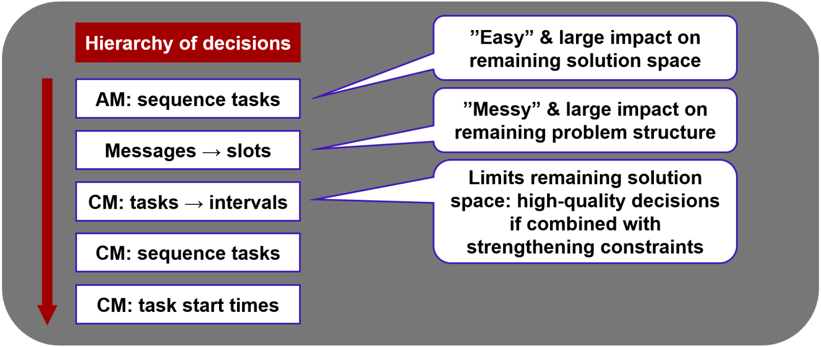
CM: task start times

”Easy” & large impact on remaining solution space

”Messy” & large impact on remaining problem structure

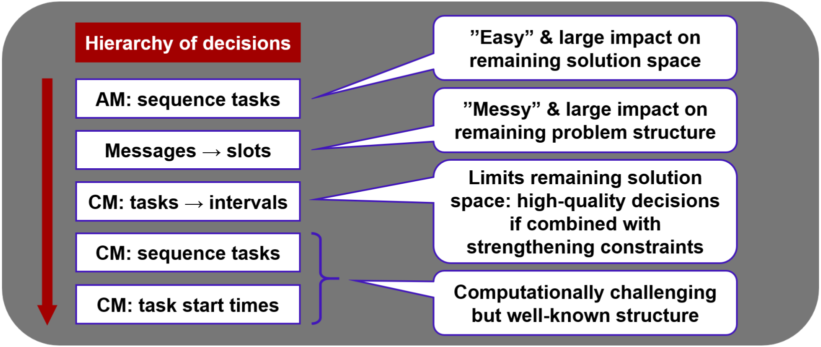
Model structure

Including the knowledge about the task sub-intervals—
the model can be seen as:


Hierarchy of decisions
AM: sequence tasks
Messages → slots
CM: tasks → intervals
CM: sequence tasks
CM: task start times
”Easy” & large impact on remaining solution space
”Messy” & large impact on remaining problem structure
Limits remaining solution space: high-quality decisions if combined with strengthening constraints

Model structure

Including the knowledge about the task sub-intervals—the model can be seen as:



Hierarchy of decisions

AM: sequence tasks

Messages → slots

CM: tasks → intervals

CM: sequence tasks

CM: task start times

”Easy” & large impact on remaining solution space

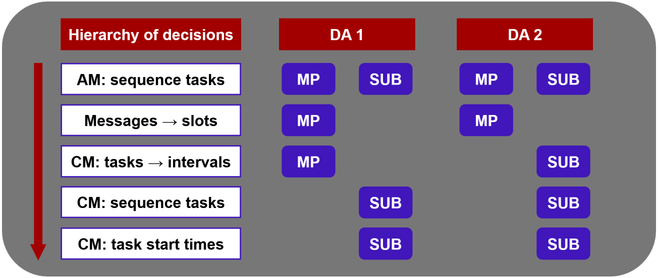
”Messy” & large impact on remaining problem structure

Limits remaining solution space: high-quality decisions if combined with strengthening constraints

Computationally challenging but well-known structure

Two decomposition approaches

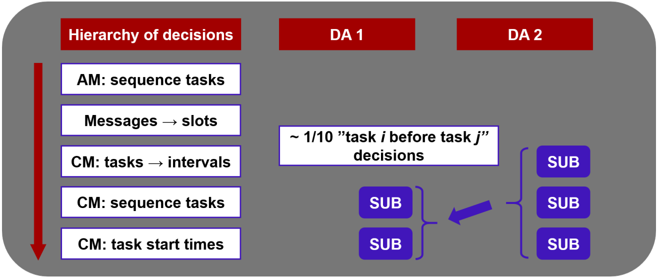
- Differs in decisions made by master problem and subproblem
- Different type of feedback information to the master problem



Hierarchy of decisions	DA 1		DA 2	
AM: sequence tasks	MP	SUB	MP	SUB
Messages → slots	MP		MP	
CM: tasks → intervals	MP			SUB
CM: sequence tasks		SUB		SUB
CM: task start times		SUB		SUB

Two decomposition approaches

- Differs in decisions made by master problem and subproblem
- Different type of feedback information to the master problem



Hierarchy of decisions

DA 1

DA 2

AM: sequence tasks

Messages → slots

CM: tasks → intervals

CM: sequence tasks

CM: task start times

~ 1/10 "task *i* before task *j*" decisions

SUB

SUB

SUB

SUB

SUB

DA1 characteristics

- Master problem:
  - Place tasks in sub-intervals and messages in slots
  - Messy and challenging MIP-model (Gurobi)

DA1 characteristics

- Master problem:
  - Place tasks in sub-intervals and messages in slots
  - Messy and challenging MIP-model (Gurobi)
- Subproblem:
  - Sequence tasks, precedence relations w. lb and ub on time lags
  - Penalise if tasks are not included in the sequence
  - Objective:
    find subsets of tasks that causes conflicts wrt sequencing
  - Order-based MIP-formulation (Gurobi)

DA1 characteristics

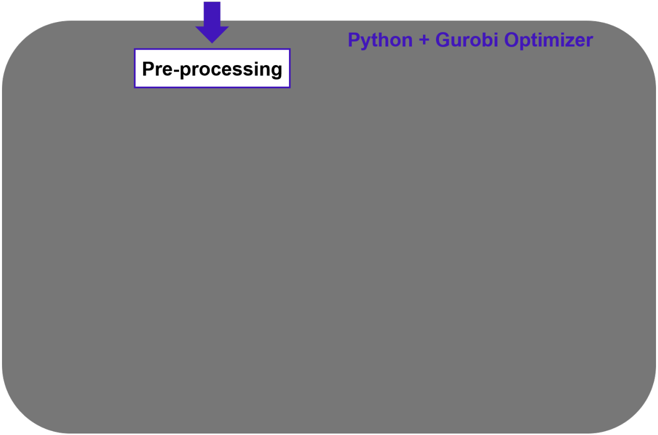
- Master problem:
  - Place tasks in sub-intervals and messages in slots
  - Messy and challenging MIP-model (Gurobi)
- Subproblem:
  - Sequence tasks, precedence relations w. lb and ub on time lags
  - Penalise if tasks are not included in the sequence
  - Objective:
    find subsets of tasks that causes conflicts wrt sequencing
  - Order-based MIP-formulation (Gurobi)
- Feedback: subsets of tasks that the master problem sequences

DA1 characteristics

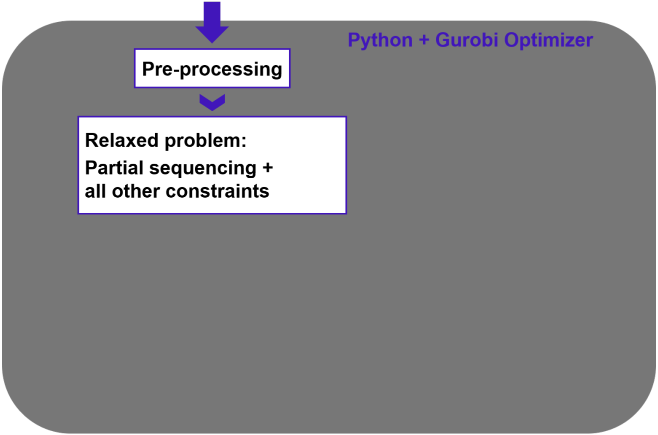
- Master problem:
  - Place tasks in sub-intervals and messages in slots
  - Messy and challenging MIP-model (Gurobi)
- Subproblem:
  - Sequence tasks, precedence relations w. lb and ub on time lags
  - Penalise if tasks are not included in the sequence
  - Objective:
    find subsets of tasks that causes conflicts wrt sequencing
  - Order-based MIP-formulation (Gurobi)
- Feedback: subsets of tasks that the master problem sequences

Mindset: Strong relaxation that can discover infeasibility ...
... and if the problem seems feasible—find a schedule

DA1: A constraint generation approach



DA1: A constraint generation approach

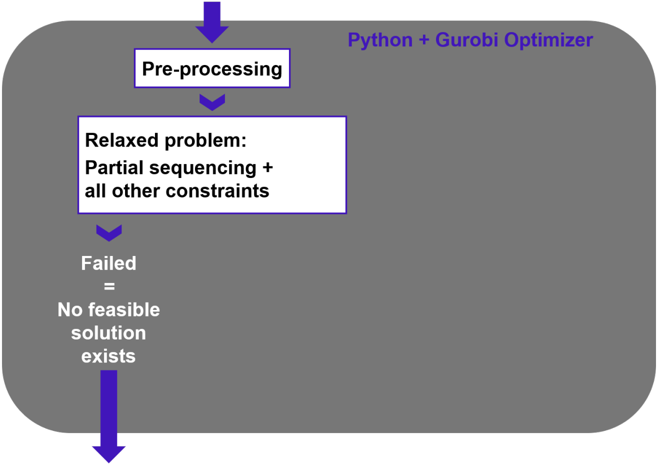


Python + Gurobi Optimizer

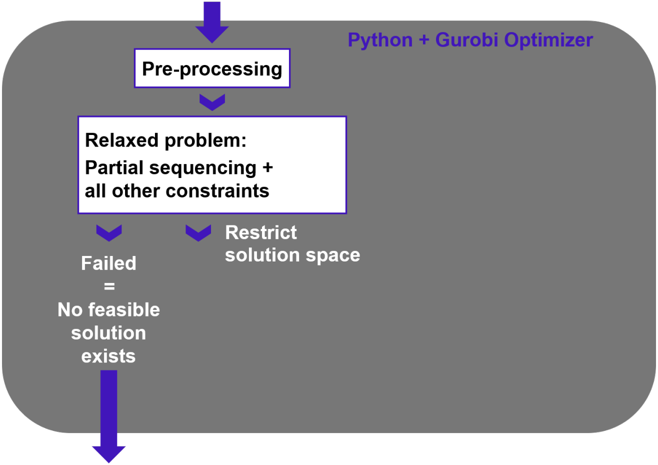
Pre-processing

Relaxed problem:
Partial sequencing +
all other constraints

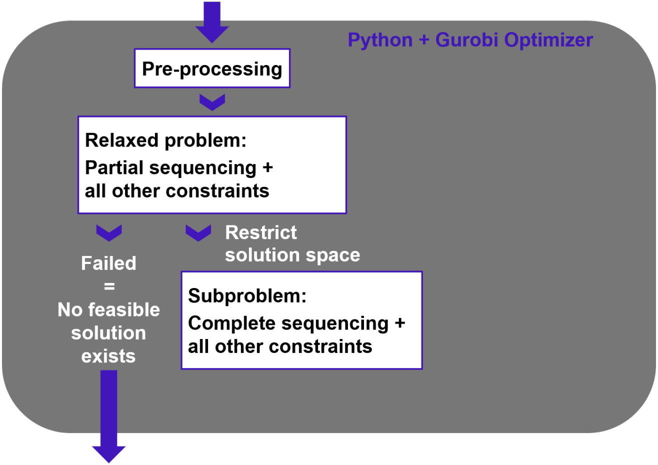
DA1: A constraint generation approach



DA1: A constraint generation approach



DA1: A constraint generation approach



Python + Gurobi Optimizer

Pre-processing

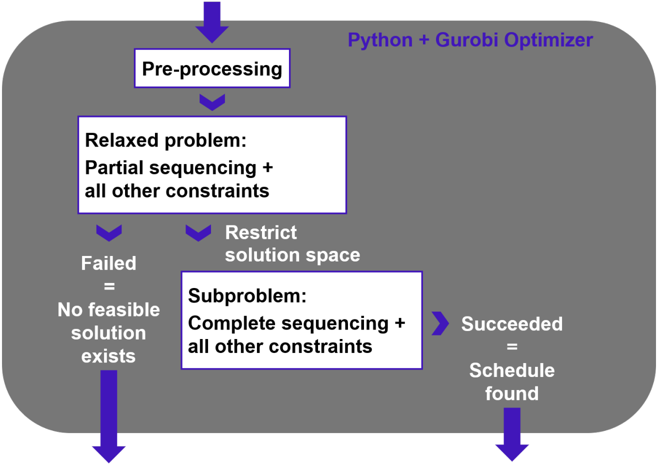
Relaxed problem:
Partial sequencing +
all other constraints

Failed
=
No feasible
solution
exists

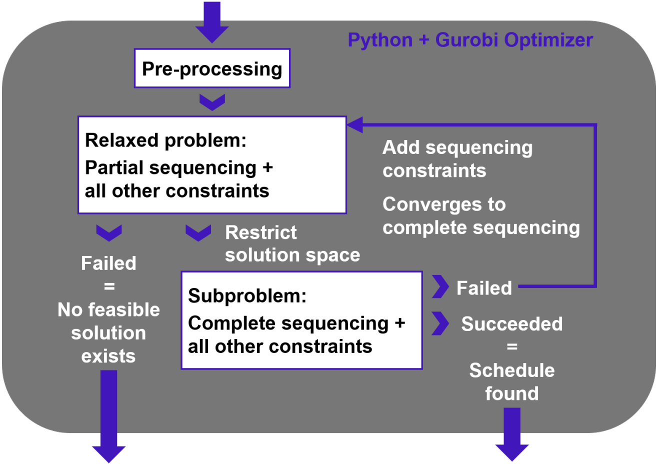
Restrict
solution space

Subproblem:
Complete sequencing +
all other constraints

DA1: A constraint generation approach



DA1: A constraint generation approach



Python + Gurobi Optimizer

Pre-processing

Relaxed problem:
Partial sequencing +
all other constraints

Add sequencing constraints

Converges to complete sequencing

Restrict solution space

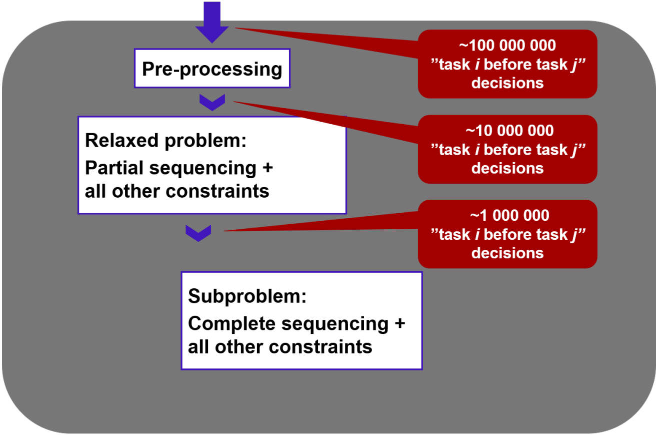
Failed = No feasible solution exists

Subproblem:
Complete sequencing +
all other constraints

Failed

Succeeded = Schedule found

DA1: A constraint generation approach



Pre-processing

Relaxed problem:
Partial sequencing +
all other constraints

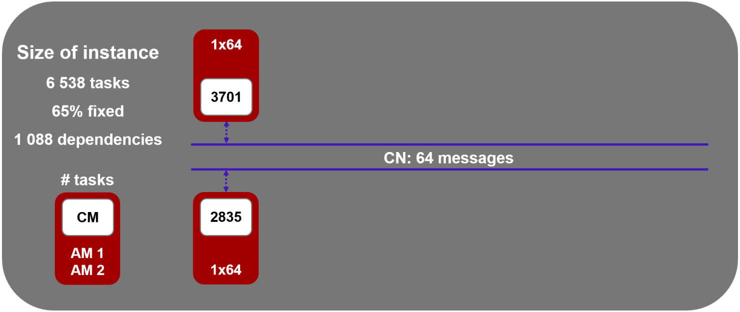
Subproblem:
Complete sequencing +
all other constraints

~100 000 000
"task *i* before task *j*"
decisions

~10 000 000
"task *i* before task *j*"
decisions

~1 000 000
"task *i* before task *j*"
decisions

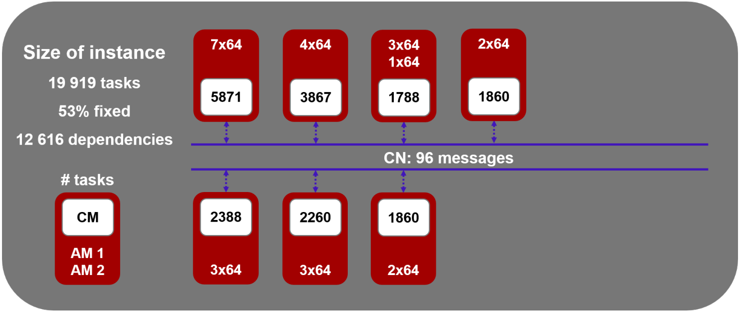
Results of DA1



Full model: not solved within a week

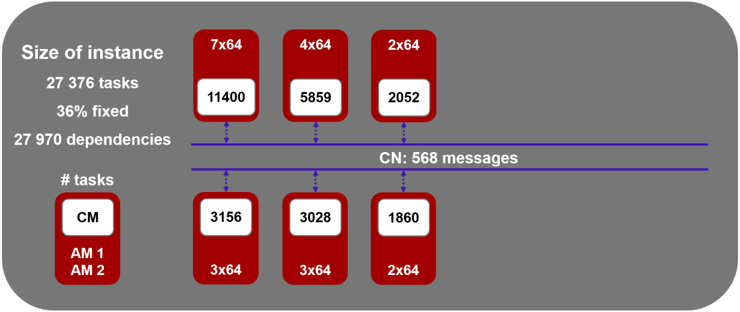
DA1: solved within < 1 minute

Results of DA1



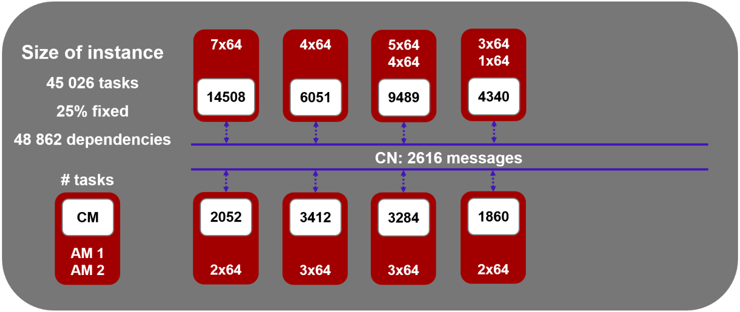
DA1: solved within 2 minutes

Did DA1 take us all the way?



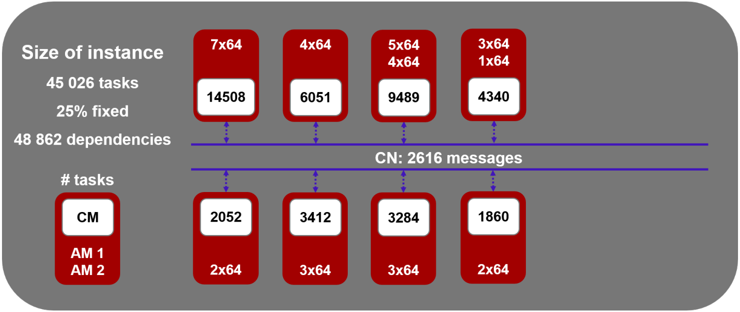
Relaxed model: solved in $11\frac{1}{2}$h

Did DA1 take us all the way?



Relaxed model: not solved in 24h

Did DA1 take us all the way? No ...



Relaxed model: not solved in 24h

A CM with more than 10 000 tasks ⇒ Relaxed problem challenging

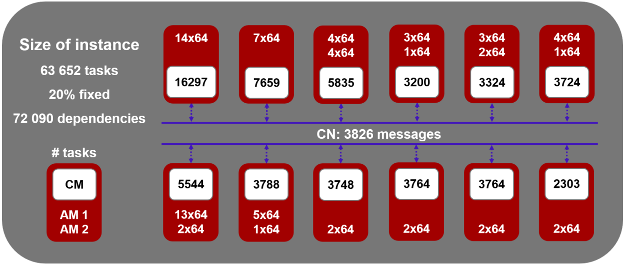
Conclusions & insights from DA1

- Really strong relaxation of the problem, it serves it purpose!
- *A bit* too expensive—want results for larger instances

Conclusions & insights from DA1

- Really strong relaxation of the problem, it serves it purpose!
- *A bit* too expensive—want results for larger instances

MIP-based Adaptive Large Neighbourhood Search (ALNS)
for solving the relaxed problem



Size of instance

63 652 tasks

20% fixed

72 090 dependencies

tasks

CM

AM 1
AM 2

14x64 — 16297

7x64 — 7659

4x64
4x64 — 5835

3x64
1x64 — 3200

3x64
2x64 — 3324

4x64
1x64 — 3724

CN: 3826 messages

5544 — 13x64
2x64

3788 — 5x64
1x64

3748 — 2x64

3764 — 2x64

3764 — 2x64

2303 — 2x64

Computational time: $\sim$ 3 days

DA2: Logic-Based Benders Decomposition (LBDD)

- Master problem:
  - Assign: messages to slots
  - Messy (but not very challenging) MIP-model (Gurobi)

DA2: Logic-Based Benders Decomposition (LBBD)

- Master problem:
  - Assign: messages to slots
  - Messy (but not very challenging) MIP-model (Gurobi)
- Subproblem:
  - Sequence tasks, not placed in intervals (respect dependencies)
  - No objective:
    Determine if feasible or not
  - CP model (IBM ILOG CP Optimizer)

DA2: Logic-Based Benders Decomposition (LBBD)

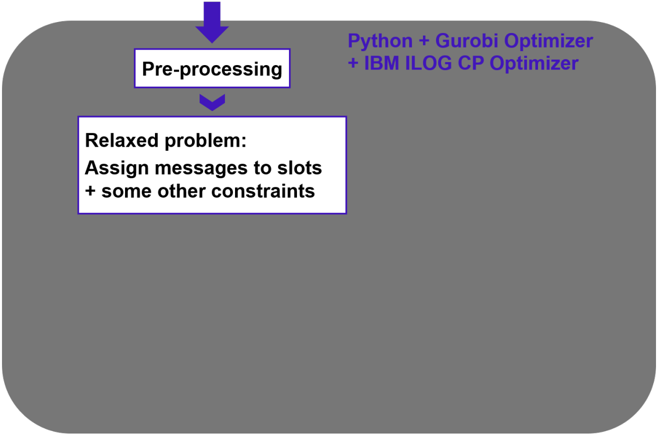
- Master problem:
  - Assign: messages to slots
  - Messy (but not very challenging) MIP-model (Gurobi)
- Subproblem:
  - Sequence tasks, not placed in intervals (respect dependencies)
  - No objective:
    Determine if feasible or not
  - CP model (IBM ILOG CP Optimizer)
- Feedback: Infeasible message to slot assignments (no-goods)

DA2: Logic-Based Benders Decomposition (LBBD)

- Master problem:
  - Assign: messages to slots
  - Messy (but not very challenging) MIP-model (Gurobi)
- Subproblem:
  - Sequence tasks, not placed in intervals (respect dependencies)
  - No objective:
    Determine if feasible or not
  - CP model (IBM ILOG CP Optimizer)
- Feedback: Infeasible message to slot assignments (no-goods)

Mindset: Less focus on discover infeasibility ...
... and find a schedule faster?

DA2: A LBBD approach



Pre-processing

Relaxed problem:

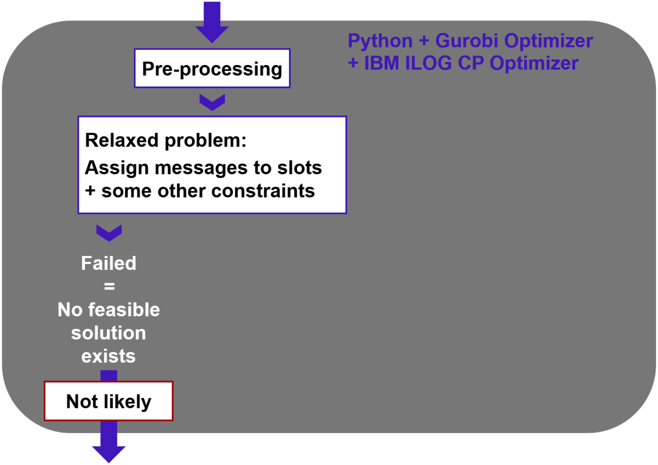
Assign messages to slots

+ some other constraints

Python + Gurobi Optimizer

+ IBM ILOG CP Optimizer

DA2: A LBBD approach



Python + Gurobi Optimizer
+ IBM ILOG CP Optimizer

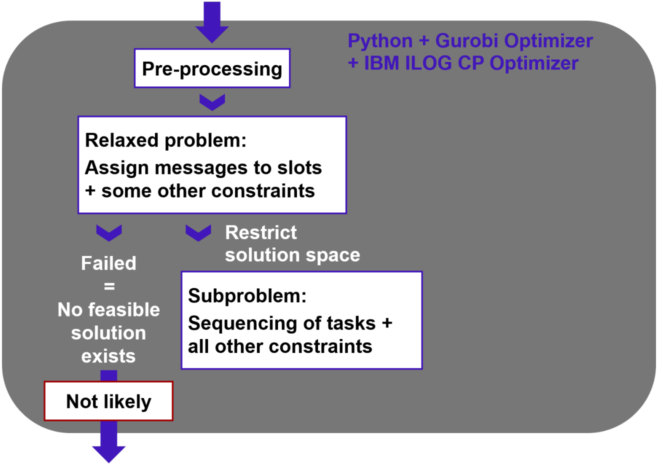
Pre-processing

Relaxed problem:
Assign messages to slots
+ some other constraints

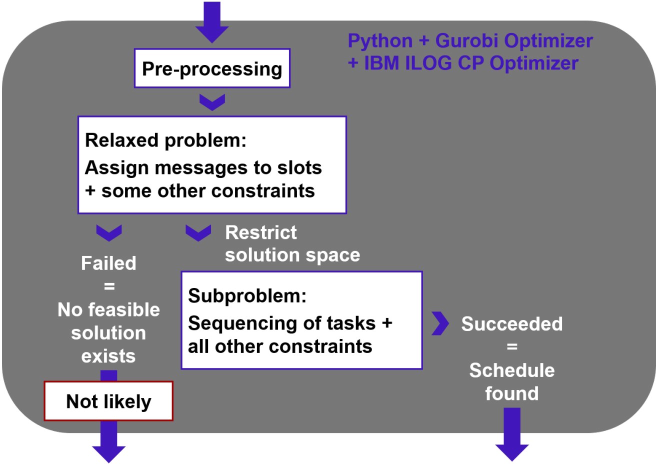
Failed
=
No feasible
solution
exists

Not likely

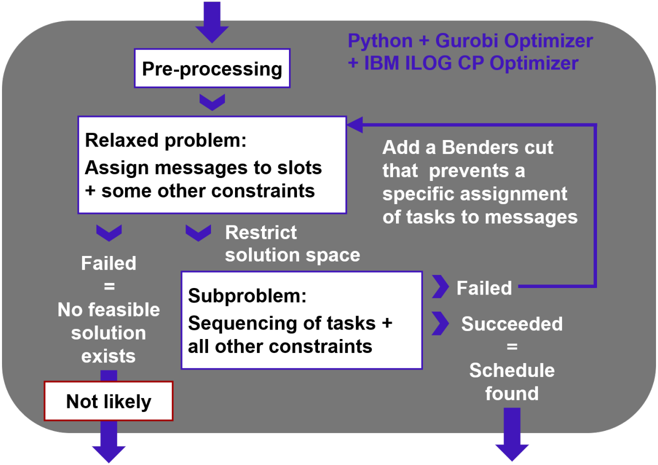
DA2: A LBBD approach



DA2: A LBBD approach



DA2: A LBBD approach



Acceleration techniques for LBBD

- Cuts from complete assignments of messages to slots are weak
  $\Rightarrow$ Apply cut-strengtheing techniques

Acceleration techniques for LBBD

- Cuts from complete assignments of messages to slots are weak $\Rightarrow$ Apply cut-strengtheing techniques
- Strengthen master problem by a subproblem relaxation
- Find a good initial solution

Acceleration techniques for LBBD

- Cuts from complete assignments of messages to slots are weak
  $\Rightarrow$ Apply cut-strengtheing techniques
- Strengthen master problem by a subproblem relaxation
- Find a good initial solution
- New partial assignment acceleration technique
  - Based on common cut-strengthening:
    Systematic search over subsets of variables
    Master problem variable $1 \to 0$ = relaxation of subproblem

Acceleration techniques for LBBD

- Cuts from complete assignments of messages to slots are weak $\Rightarrow$ Apply cut-strengtheing techniques
- Strengthen master problem by a subproblem relaxation
- Find a good initial solution
- New partial assignment acceleration technique
  - Based on common cut-strengthening:
    Systematic search over subsets of variables
    Master problem variable $1 \rightarrow 0$ = relaxation of subproblem
  - Our extension:
    restriction to explore in the search to find feasible solutions

Computational comparisons

Set of public instances:
https://gitlab.liu.se/eliro15/avionics_inst/tree/master

Instance category D, 30 instances with ranges:

Modules	Tasks	Messages	Fixed tasks	Dependencies
14–21	30,000–55,000	1200–2800	5000–8000	60,000–120,000

Computational comparisons

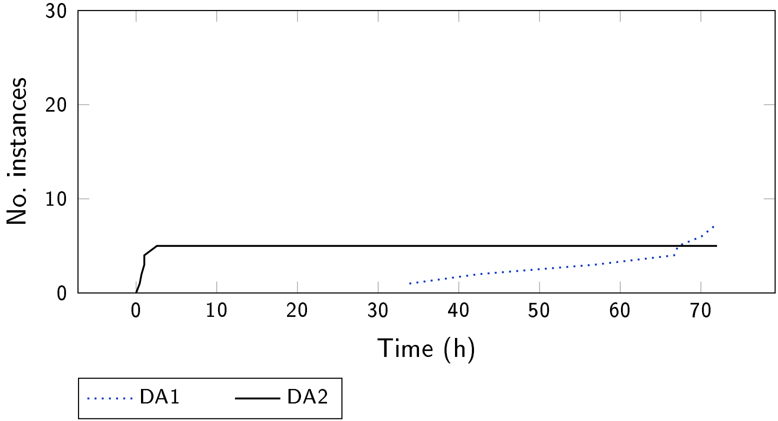
Set of public instances:
https://gitlab.liu.se/eliro15/avionics_inst/tree/master

Instance category D, 30 instances with ranges:

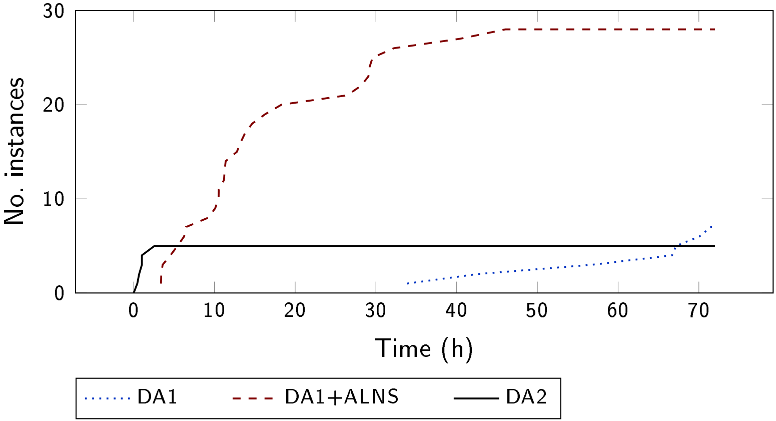
Modules	Tasks	Messages	Fixed tasks	Dependencies
14–21	30,000–55,000	1200–2800	5000–8000	60,000–120,000

- Two Intel Xeon Gold 6130 Processors (16 cores, 2.1 GHz)
- Unfair comparison wrt memory usage:
  - DA1: 384 GB RAM
  - DA2: 96 GB RAM

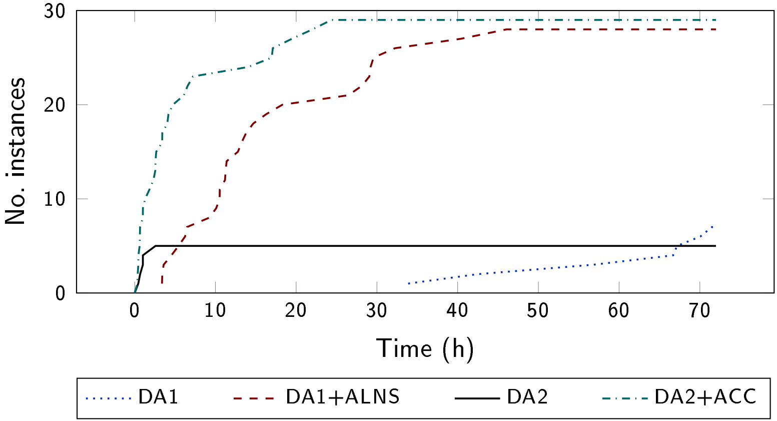
Computational results: "Pure methods"



With matheuristic component in DA1



With new acceleration technique in DA2



Mathematical programming & constraint programming

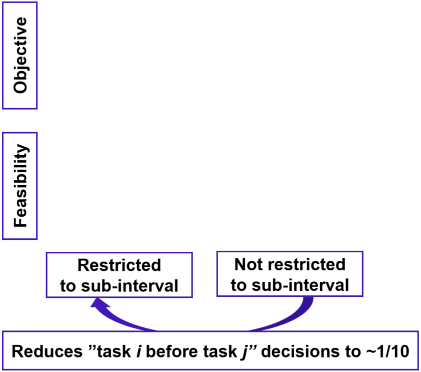
Both subproblems:

- Sequencing of tasks on modules (up to 15,000 per module)
- Release time and deadline for each task
- Precedence relations with lb and ub on time lags
- Some more ”details”

Mathematical programming & constraint programming

Both subproblems:

- Sequencing of tasks on modules (up to 15,000 per module)
- Release time and deadline for each task
- Precedence relations with lb and ub on time lags
- Some more "details"



Objective

Feasibility

Restricted to sub-interval

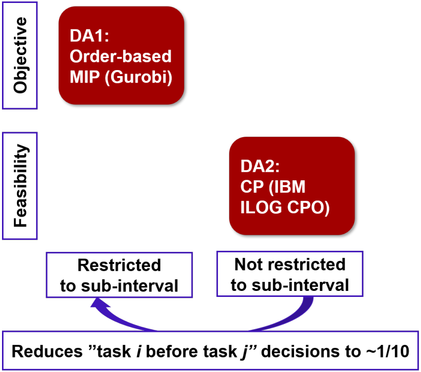
Not restricted to sub-interval

Reduces "task *i* before task *j*" decisions to ~1/10

Mathematical programming & constraint programming

Both subproblems:

- Sequencing of tasks on modules (up to 15,000 per module)
- Release time and deadline for each task
- Precedence relations with lb and ub on time lags
- Some more "details"



Objective

Feasibility

DA1: Order-based MIP (Gurobi)

DA2: CP (IBM ILOG CPO)

Restricted to sub-interval

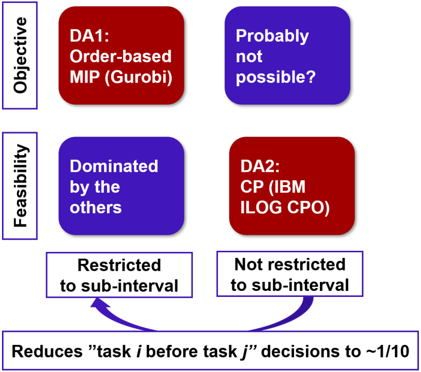
Not restricted to sub-interval

Reduces "task *i* before task *j*" decisions to ~1/10

Mathematical programming & constraint programming

Both subproblems:

- Sequencing of tasks on modules (up to 15,000 per module)
- Release time and deadline for each task
- Precedence relations with lb and ub on time lags
- Some more "details"


Objective
DA1: Order-based MIP (Gurobi)
Probably not possible?
Feasibility
Dominated by the others
DA2: CP (IBM ILOG CPO)
Restricted to sub-interval
Not restricted to sub-interval
Reduces "task *i* before task *j*" decisions to ~1/10

Models & data

- Technical requirements → mathematical modelling and decomposition
- Understanding data and engineering assumptions → preprocessing and decomposition

Learning on this higher level possible?

Models & data

- Technical requirements →
  mathematical modelling and decomposition
- Understanding data and engineering assumptions →
  preprocessing and decomposition

Learning on this higher level possible?

On a lower level: very active area of research

- Select methods or decompositions, boost methods, ...
- Our current work
  - Oberweger F.F., Raidl G.R., Rönnberg E., Huber M.: *A Learning Large Neighborhood Search for the Staff Rerostering Problem.* CPAIOR 2022.
  - Ongoing: Learning in logic-based Benders decomposition

Acknowledgements & references

- Center for Industrial Information Technology (CENIIT)
- Research School in Interdisciplinary Mathematics at Linköping University
- Computational experiments: the Swedish National Infrastructure for Computing (SNIC) at National Supercomputer Centre (NSC)

DA2+ACC Karlsson E., Rönnberg E.: *Logic-based Benders decomposition with a partial assignment acceleration technique for avionics scheduling*, Computers & Operations Research, 146:105916, 2022.

DA1+ALNS Karlsson E., Rönnberg E., Stenberg A., and Uppman H.: *A matheuristic approach to large-scale avionic scheduling*, Annals of Operations Research, 302:425–459, 2021.

LBBD Karlsson E., Rönnberg E.: *Strengthening of feasibility cuts in logic-based Benders decomposition*, CPAIOR 2021.

DA1 Blikstad, M., Karlsson, E., Lööw, T., Rönnberg, E.: *An optimisation approach for preruntime scheduling of tasks and communication in an integrated modular avionic system*, Optimization and Engineering 19:977–1004, 2018.

Acknowledgements & references

- Center for Industrial Information Technology (CENIIT)
- Research School in Interdisciplinary Mathematics at Linköping University
- Computational experiments: the Swedish National Infrastructure for Computing (SNIC) at National Supercomputer Centre (NSC)

DA2+ACC Karlsson E., Rönnberg E.: *Logic-based Benders decomposition with a partial assignment acceleration technique for avionics scheduling*, Computers & Operations Research, 146:105916, 2022.

DA1+ALNS Karlsson E., Rönnberg E., Stenberg A., and Uppman H.: *A matheuristic approach to large-scale avionic scheduling*, Annals of Operations Research, 302:425–459, 2021.

LBBD Karlsson E., Rönnberg E.: *Strengthening of feasibility cuts in logic-based Benders decomposition*, CPAIOR 2021.

DA1 Blikstad, M., Karlsson, E., Lööw, T., Rönnberg, E.: *An optimisation approach for preruntime scheduling of tasks and communication in an integrated modular avionic system*, Optimization and Engineering 19:977–1004, 2018.

Thanks for listening!

LINKÖPING UNIVERSITY