

# The AIMMS Interface to Constraint Programming

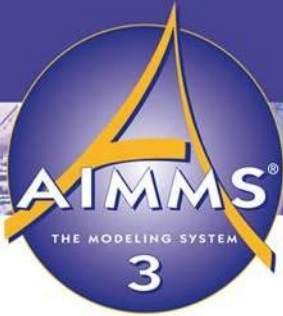


Willem-Jan van Hoeve<sup>1</sup>, Marcel Hunting<sup>2</sup>, and Chris Kuip<sup>2</sup>

<sup>1</sup> Tepper School of Business, Carnegie Mellon University

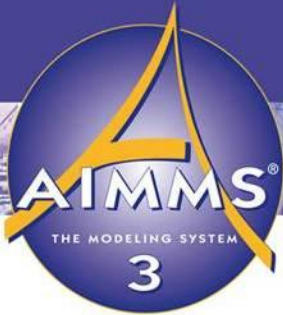
<sup>2</sup> Paragon Decision Technology

# Outline



- Motivation
- Language Extensions
  - Basic CP Functionality
  - Global Constraints
  - Scheduling Problems
- Graphical Objects
- Search and Solver Interface
- Demo

# Constraint Programming



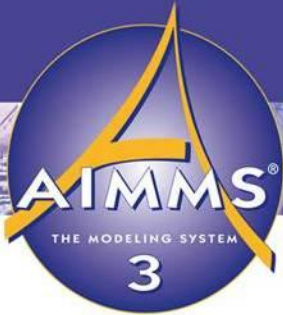
- Origins in Artificial Intelligence, Logic Programming, and Operations Research
- Modern CP (1994-)
  - Modular approach to combinatorial problem solving
  - Problem structure is modeled through high-level constraints
  - Systematic search combined with inference methods

minimize  $\max( i, \text{End}(i) )$

s.t.  $\text{SequentialSchedule}( i, \text{Start}(i), \text{Dur}(i), \text{End}(i) )$

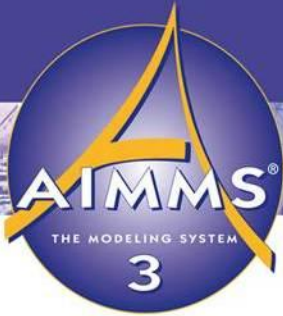
(minimize makespan for disjunctive scheduling over activities  $i$  )

# Why CP in AIMMS?



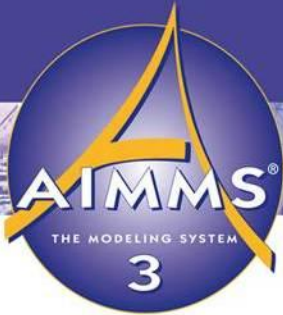
- CP has intuitive and expressive **modeling** syntax
  - arbitrary logical and algebraic expressions
  - constraints can be nonlinear, non-convex, non-continuous, ...
  - special 'global constraints' such as alldifferent, scheduling, ...
- CP has powerful **solving** methodology
  - particularly effective for highly combinatorial problems, e.g., rostering, scheduling, ...
- Increased need for **integrated** methods (e.g., MIP+CP)
  - constraint-based column generation
  - logic-based Benders decomposition
  - large neighborhood search

# CP modeling examples



- Variables range over a finite domain:  
 $v \in \{a,b,c,d\}, \text{ start} \in \{0,1,2,8,9,10\}$
- Algebraic expressions:  
 $x^3(y^2 - z) \geq 25 + x^2 \cdot \max(x,y,z)$
- Variables as subscripts:  
 $y = \text{cost}[x]$  (here  $y$  and  $x$  are variables, 'cost' is an array of parameters)
- Reasoning with meta-constraints:  
 $\sum_i (x_i > T_i) \leq 5$
- Logical relations in which (meta-)constraints can be mixed:  
 $((x < y) \text{ OR } (y < z)) \Rightarrow (c = \min(x,y))$
- Global constraints (a.k.a. symbolic constraints):  
 $\text{AllDifferent}(x_1, x_2, \dots, x_n)$

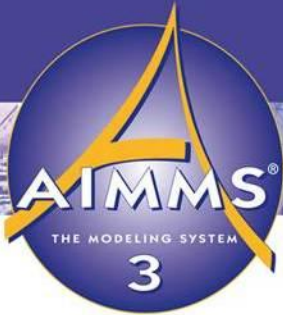
# Why AIMMS for CP



- CP is popular among dedicated professionals, but
  - considerable learning curve
  - CP solvers typically have low-level interface (Prolog, C++, scripting)
- AIMMS offers more gentle access to CP technology
  - modeling style is similar to existing AIMMS syntax
  - AIMMS is solver (i.e., vendor) independent
  - easy integration of MIP, QP, NLP, and now CP solvers
  - AIMMS offers advanced graphical interface

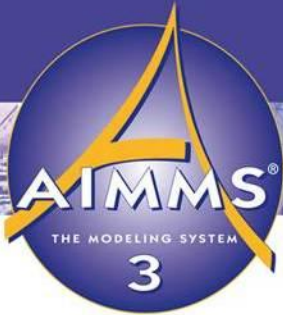


# Why CP for CAPD audience?



- Industrial design and operations problems often involve
  - planning
  - scheduling
  - optimization
- CP/MILP Decomposition methods can be particularly effective
  - column generation [Junker et al., 1999]
  - logic-based Benders [Jain & Grossmann, 2001] [Hooker, 2007]
  - multi-stage scheduling [Harjunkoski & Grossmann, 2002]
  - ...and many more (see, e.g., CPAIOR conference)
- AIMMS naturally facilitates the implementation of such hybrid methods, now also including CP

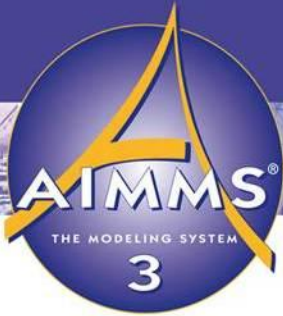
# Selected Related work



- Existing modeling languages from OR community
  - **AMPL**: extension with CP [Fourer & Gay, 2002] (not released though)
  - **XPRESS Mosel**: CP extension, but solver dependent (CP Kalis)
- Existing modeling languages/systems from CP community
  - **OPL**: great for professionals, but solver dependent (IBM-ILOG), and perhaps not as gentle as AIMMS
  - **Comet**: CP, MIP and LS technology, but solver dependent
  - **Zinc**: solver independent, no NLP, no graphical interface, academic release
  - **SIMPL**: mix between model and algorithm development, academic release

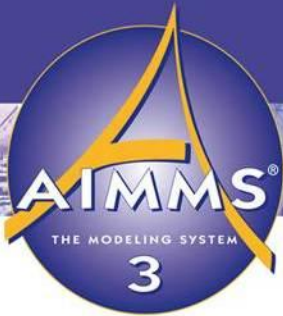


# Language Extensions



- Basic CP functionality
- Global constraints
- Scheduling problems

# Basic CP functionality in AIMMS: Variables



- Variable types

- continuous: not available for CP
- integer: available for MIP, MINLP, and CP
- **element variables**: new, available only for CP

ELEMENT VARIABLE:

```
identifier      : SupplyingWarehouse  
index domain   : f  
range          : Warehouses ;
```

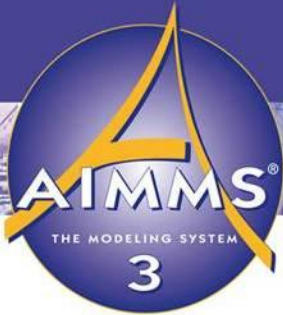
VARIABLE:

```
identifier      : SupplyCost  
definition     : sum(f, SupplyCost(f, SupplyingWarehouse(f))) ;
```

set-based type checking for  
element variables



# Basic CP functionality in AIMMS: Constraints



- Existing AIMMS syntax
  - standard arithmetic, logical, and set related operators
  - for MIP, these can be applied to condition an index set
- CP extension
  - semantic change: allow variables to appear in these expressions
  - minimal change to the AIMMS user

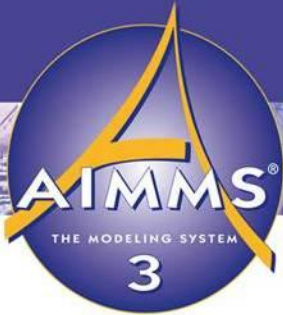
CONSTRAINT:

```
identifier : MyTableConstraint
```

```
definition : (Var1, Var2, Var3) in MyThreeDimRelation ;
```

MyThreeDimRelation represents a set of allowed tuples  
(this is called a 'Table' constraint)

# Global Constraints in AIMMS



- Global constraint catalog contains 354 global constraints from the literature...
- AIMMS offers all common global constraints (plus specific constraints for scheduling)

`cp::AllDifferent`

`cp::BinPacking`

`cp::Cardinality`

`cp::Channel`

`cp::Count`

`cp::Lexicographic`

`cp::ParallelSchedule`

`cp::Sequence`

`cp::SequentialSchedule`

*cumulative*

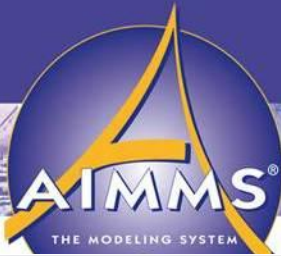
*unary*

CONSTRAINT:

identifier : DifferentColors

definition : `cp::AllDifferent(i, ColorOfItem(i)) ;`

# Representing Scheduling Problems

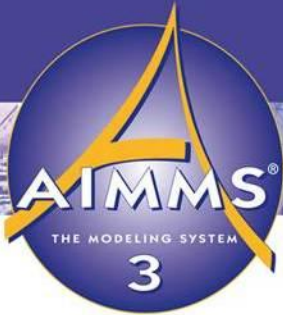


The screenshot shows the AIMMS software interface. The title bar reads "AIMMS". The menu bar includes "File", "Edit", "View", "Data", "Run", "Settings", "Tools", "Window", and "Help". The toolbar contains various icons for file operations and modeling. The "Model Explorer" on the left shows a project named "Main SchedulingExample" with a "Declaration" folder containing "Tasks", "StartTime(t)", "Duration(t)", "EndTime(t)", and "noTasksCanOverlap". Below this are "MainInitialization", "MainExecution", and "MainTermination". The "noTasksCanOverlap" constraint is selected, and its details are shown in the right pane.

noTasksCanOverlap	
Type	Constraint
Identifier	noTasksCanOverlap
Index domain	
Text	
Unit	
Property	
Definition	<pre>cp::SequentialSchedule(     jobDomain      : t,     jobStart       : StartTime(t),     jobDuration    : Duration(t),     jobEnd         : EndTime(t))</pre>



# Representing Scheduling Problems: Advanced Models



- Advanced functionality
  - **Activities:** objects that must be scheduled
  - **Resources:** are impacted by the execution of activities
- Activities
  - Each activity `Act` has automatically associated variables
    - `Act.Begin`
    - `Act.End`
    - `Act.Length`
    - `Act.Size`
    - `Act.Present`
  - These can be used as normal variables elsewhere in the model

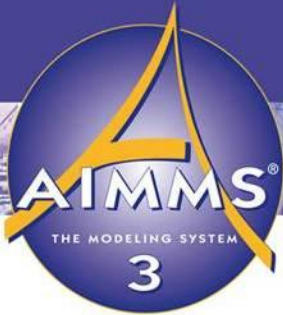
# Resource Representation

The screenshot displays the AIMMS software interface. The main window shows the 'Budget' resource definition. The 'Usage' field is set to 'Parallel', which is highlighted with a red box and an arrow pointing to the text 'Usage: Parallel or Sequential'. The 'Index domain' is empty, and the 'Schedule domain' is 'TimeHorizon'. The 'Activities' field lists 'Act (a), DepositAct (d)'. The 'Level range' is '{0..100}', and the 'Initial level' is '0'. The 'Begin change' and 'End change' fields are defined with activities and their corresponding data objects.

Property	Value
Identifier	Budget
Usage	Parallel
Index domain	
Text	
Schedule domain	TimeHorizon
Activities	Act (a), DepositAct (d)
Property	
Unit	
Level range	{0..100}
Initial level	0
Level change	
Begin change	DepositAct (d) : AmountDeposited (d), Act (a) : ActCost (a)
End change	Act (a) : Profit (a)
Comment	

Usage: Parallel or Sequential

# Global Scheduling Constraints



- For advanced scheduling models, AIMMS offers the following global constraints

`cp::BeginAtBegin`

`cp::EndBeforeBegin`

`cp::BeginAtEnd`

`cp::EndBeforeEnd`

`cp::BeginBeforeBegin`

`cp::Alternative`

`cp::BeginBeforeEnd`

`cp::Span`

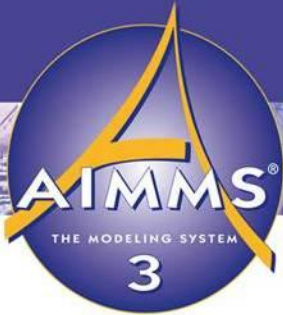
`cp::EndAtBegin`

`cp::Synchronize`

`cp::EndAtEnd`

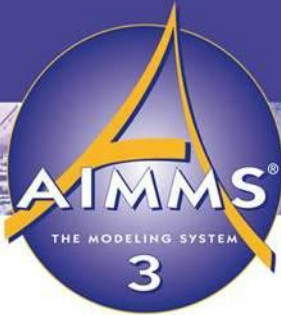
- Allows to build complex 'hierarchical' scheduling problems

# Graphical Objects



- AIMMS offer graphical user interface
  - display solution
  - what-if analysis (change data and resolve from output page)
  - end-user application (deployable)
- Also CP objects can be directly linked to graphical objects
- Examples:
  - Pivot tables
  - Network objects
  - Gantt charts
  - ...

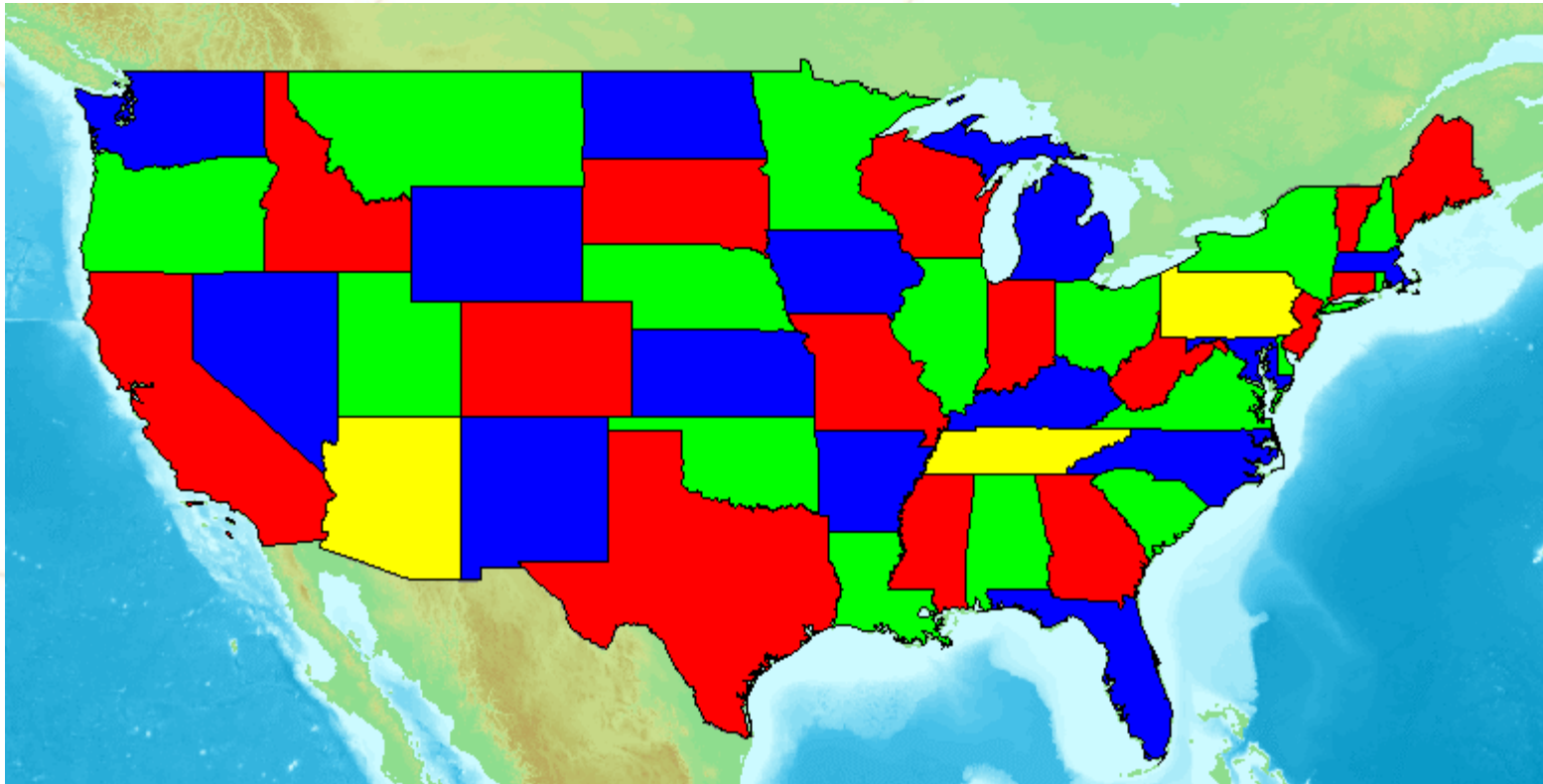
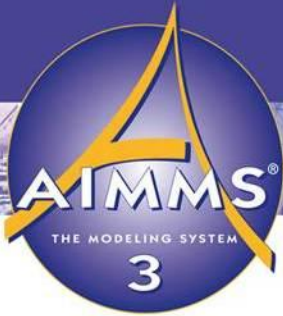
# Pivot Table for Crew Scheduling



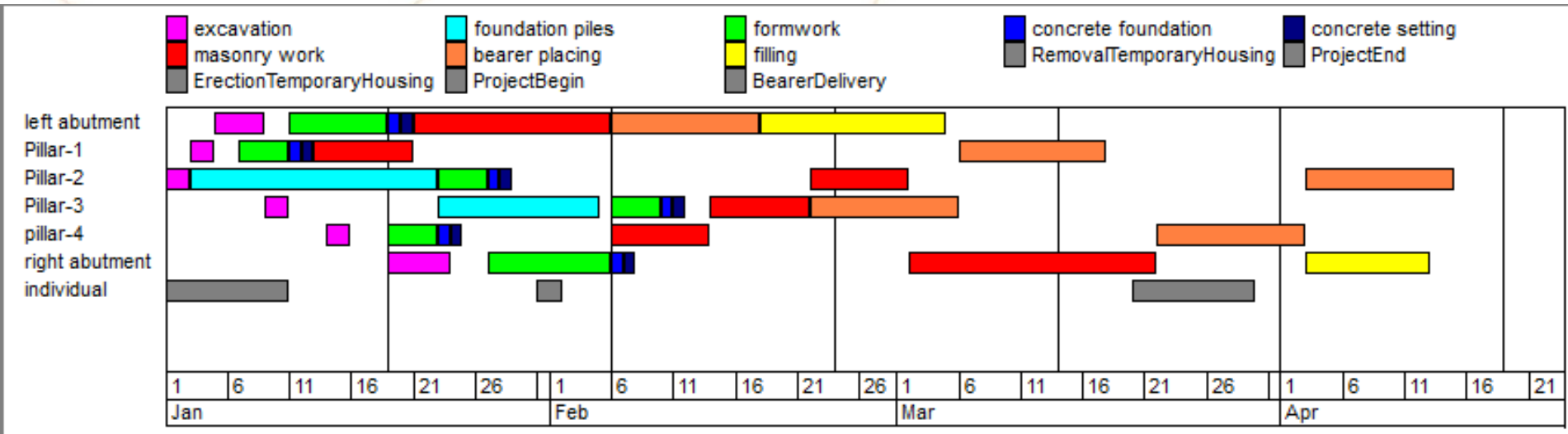
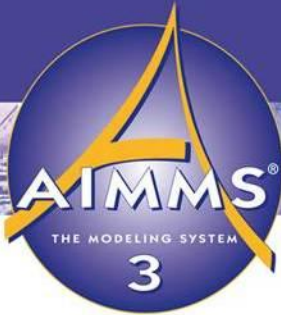
tf		Flight-01	Flight-02	Flight-03	Flight-04	Flight-05	Flight-06	Flight-07	Flight-08	Flight-09	Flight-10
Identifier	st_m										
CrewOnFlight	david										
	jeremy										
	ron										
	joe										
	bill										
	fred										
	bob										
	mario										
	ed										
	Count	1	3	2	3	3	2	4	2	3	4
RequiredCrew		1	1	1	2	3	1	1	1	2	3
CrewOnFlight	st_f										
	carol										
	janet										
	tracy										
	marilyn										
	carolyn										
	cathy										
	inez										
	jean										
	heather										
	juliet										
	Count	3	2	3	3	4	2	1	4	3	3
RequiredCrew		1	1	1	2	3	1	1	1	2	3



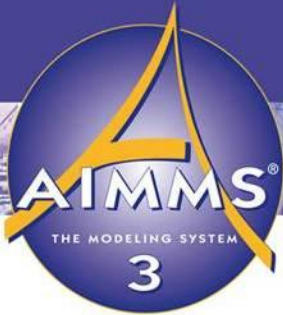
# Network Object for Map Coloring



# Gantt Chart for Construction Project Scheduling

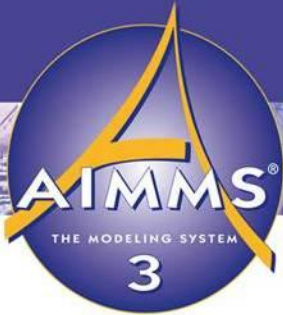


# Search Strategies

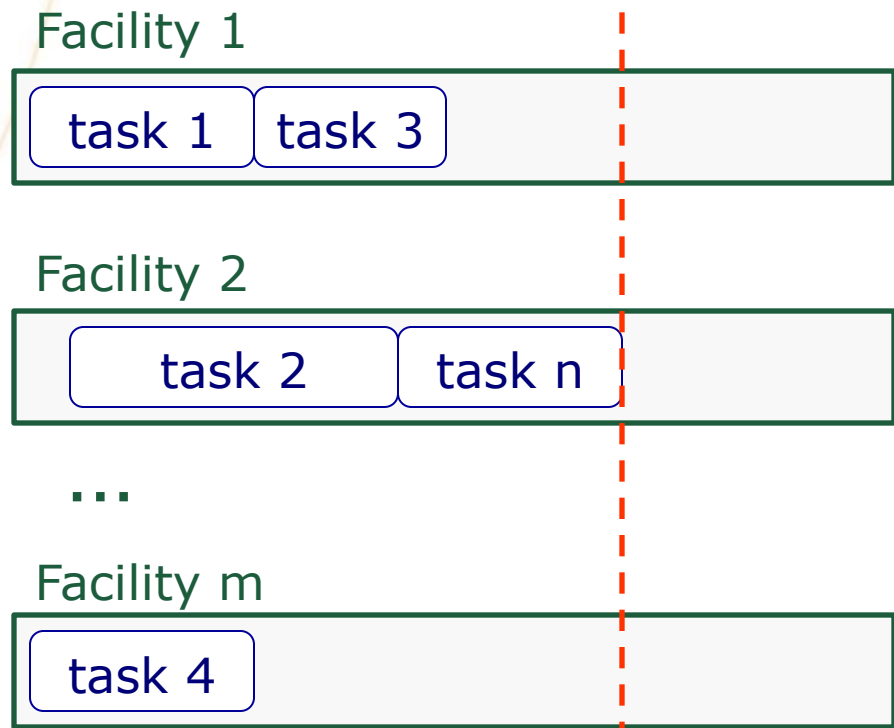
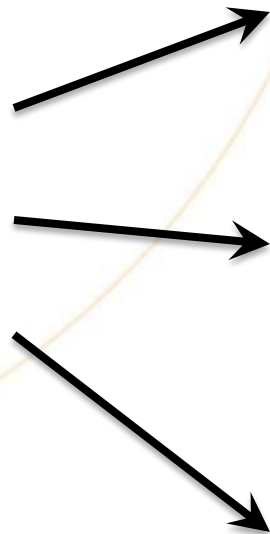
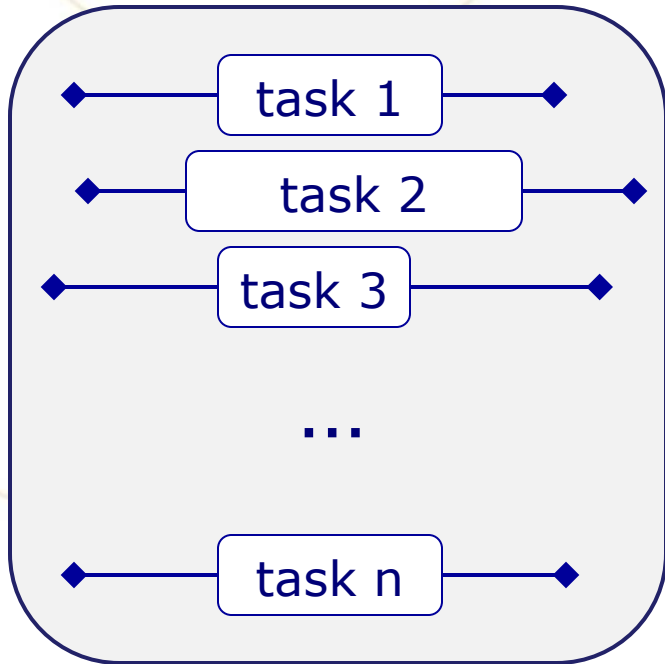


- Several CP solvers offer the possibility to declare a search strategy in the model
- At the modeling level, AIMMS supports search phases
  - 'priority' field for a variable (similar to MIP)
  - first branch on group of variables with the first priority
- Furthermore, all common search strategies can be selected as 'solver option'
  - depth-first, random restarts, impact-based, ...

# Demo: Logic-Based Benders for Task-Facility Allocation

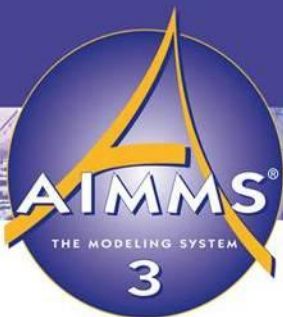


$r_i$        $p_i$        $d_i$



Makespan

# Benders Decomposition



Assign tasks to facilities (MIP)

min Makespan  
s.t. all tasks are assigned  
Makespan  $\geq$   
totalLoad(f)/Capacity(f) for all f  
Benders cuts; LBs and feasibility

task assignments  $T(f)$

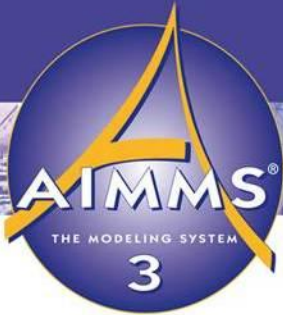
Benders cuts

Find schedule for each facility f (CP)

min Max( EndOf( $T(f)$ ) )  
s.t. ParallelSchedule(  $T(f)$ , Capacity(f) )



# Summary



- The AIMMS interface to Constraint Programming offers gentle access to state-of-the-art CP technology
- Minimal changes to existing syntax
- Basic and extended CP functionality
  - global constraints, scheduling algorithms, ...
- CP objects can be directly displayed in graphical interface
- Wide range of other solvers readily available
  - allows to easily integrate with MIP, NLP, ...
- Free academic license available
- <http://www.aimms.com/operations-research/mathematical-programming/constraint-programming>