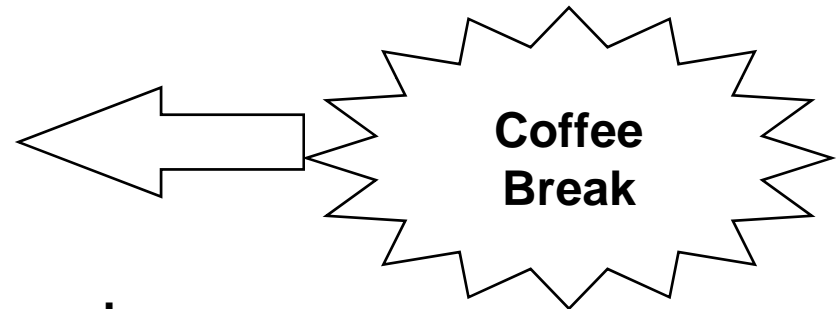


# **Scheduling and Planning with Constraint Logic Programming**

**Helmut Simonis  
COSYTEC SA**

# Overview

- **The problem**
  - what do we discuss
- **The methodology**
  - how do we express a problem
- **Applications**
  - examples of the use of the approach
- **To probe deeper**
  - how to continue



# Note

- **Tutorial with same name at PAP94**
- **Structure is kept**
- **Methods have changed**
- **New constraints**
  - diffn
  - cycle
  - more abstraction, more propagation
- **New methodology ideas**
  - machine choice
  - setup time problems
- **More applications**

# What we will discuss

- **Planning, scheduling, assignment**
  - Emphasis on detailed scheduling
- **Constraint logic programming (CLP)**
  - Emphasis on constraint
  - Modelling techniques
  - Use of constraints
- **Logic Programming**
  - Used as basis
  - Alone not sufficient

# What we will not discuss

- **System Considerations**
  - Object/Database Design
  - Integration
  - Graphical Interface Design
  - Quality Assurance
- **Details on constraint techniques**
  - Methods used inside constraint solvers
- **Classical Techniques**
  - Algorithms
  - Strategies

# Part 1: The problem

- **Terminology**
  - where we define the different terms and notations
- **Classification**
  - where we describe different classes of problems
- **Classical results**
  - where we present different results from Operational Research (OR) on scheduling and planning

# Terminology

- **What do we mean by**
  - Scheduling
  - Planning
  - Assignment
- **The main concepts**
  - Resources
  - Tasks
  - Jobs
  - Time
  - Cost

# Definitions

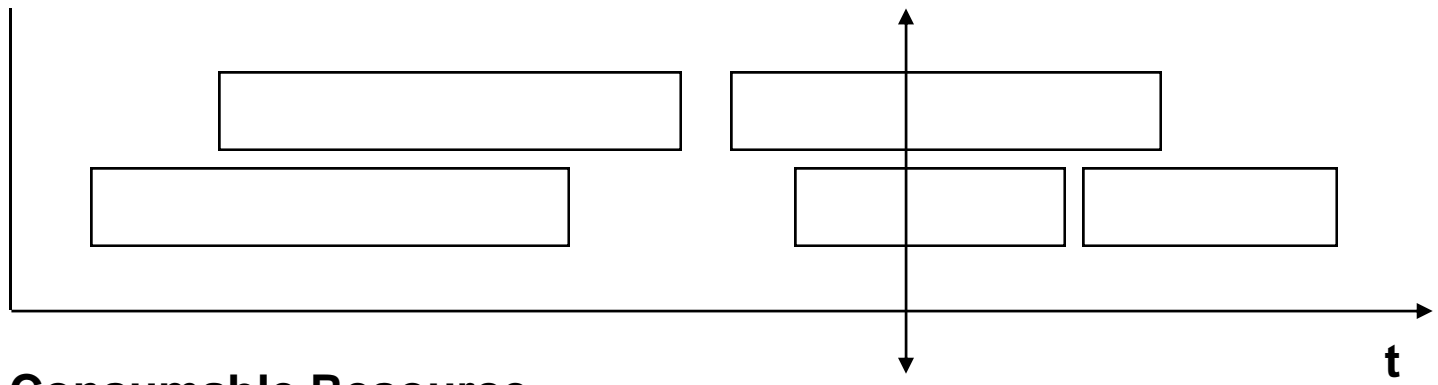
- **(Production) Planning**
  - *what to produce in which quantity in which period*
    - » decides which products will be produced in which period in which quantity
    - » driven by demand or forecasts
- **(Detailed) Scheduling**
  - *when to produce*
    - » detailed plan when to start production, when to stop
    - » driven by availability of resources and material
- **(Resource) Assignment**
  - *how to produce*
    - » which person (machine) will do what at which time
    - » often on ad hoc basis

# Resources

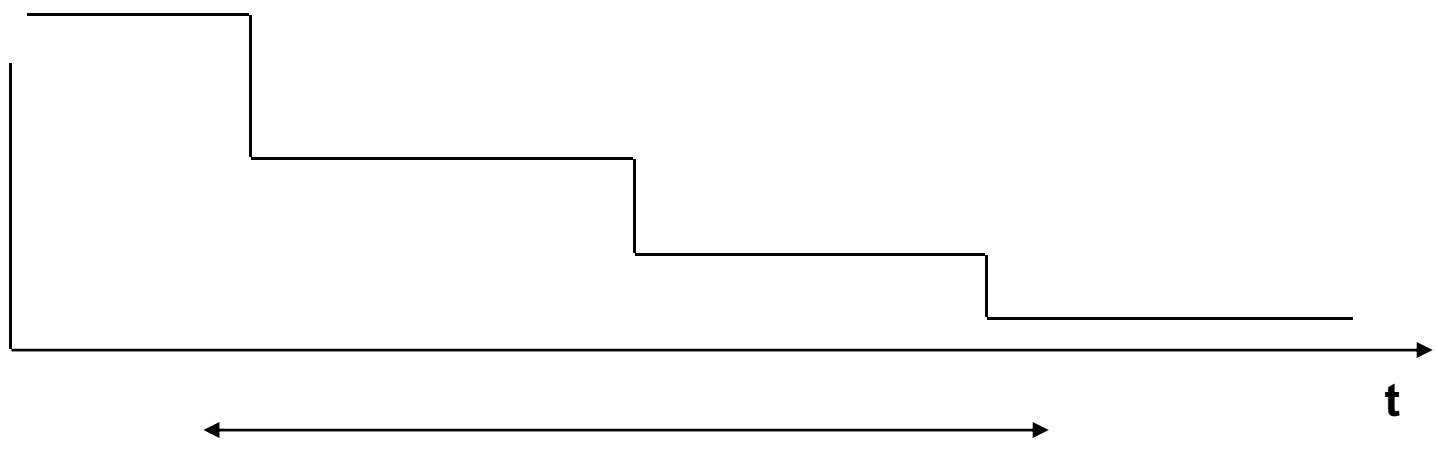
- **Renewable**
  - Typical examples:
    - » machines
    - » manpower
  - Used during time period
  - Constraints on overall use at a time point
- **Consumable**
  - Typical examples:
    - » raw material
    - » money
    - » utilities
  - Used at a certain time point
  - Constraints on overall use in a time period

# Resources (2)

**Renewable Resource**



**Consumable Resource**



# Tasks

- **Elementary operation during time period**
- **May or may not use resources**
  - full resource use
  - partial resource use
- **Tasks may be split (interrupted)**
  - preemptive scheduling

# Jobs

- **Set of tasks belonging together**
- **Sequence of tasks may be pre-determined**
  - precedence constraints between tasks
- **Jobs often relate to**
  - manufacturing of a product or
  - an order

# Time

- **Discrete/continuous**
- **Granularity**
- **Planning horizon**
- **Problem size: number of time points**

	<b>1 day</b>	<b>1 week</b>	<b>1 month</b>	<b>1 year</b>
<b>second</b>	<b>86400</b>	<b>604000</b>	<b>-</b>	<b>-</b>
<b>minute</b>	<b>1440</b>	<b>10800</b>	<b>44640</b>	<b>525600</b>
<b>hour</b>	<b>24</b>	<b>168</b>	<b>744</b>	<b>8760</b>
<b>shift</b>	<b>3</b>	<b>21</b>	<b>93</b>	<b>1095</b>

# Time and tasks

- **Start Date**  $t_i$ 
  - the actual starting time of a task
- **End Date**  $C_i$ 
  - the actual end time of a task
- **Duration**  $p_i$ 
  - $C_i - t_i$
  - may vary depending
    - » on the resource used
    - » on the start date
    - » etc

# Due/release date

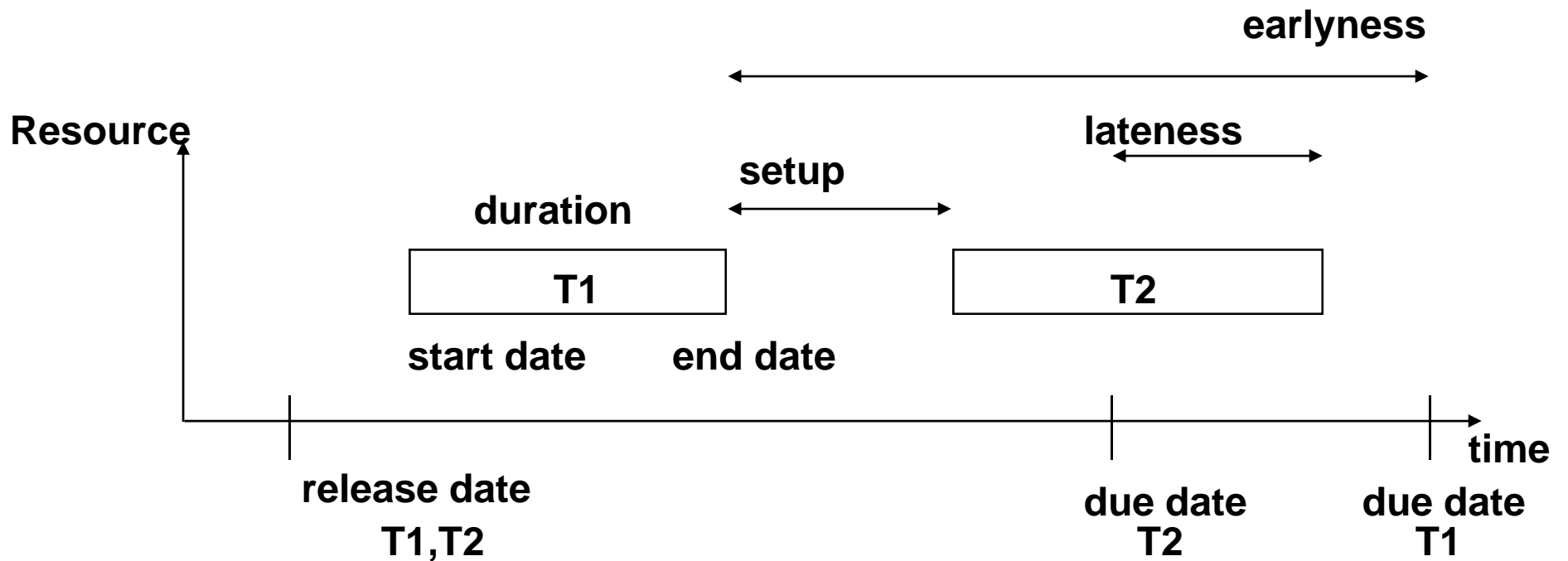
- **Due Date**  $d_i$ 
  - the time when the task should be finished
- **Release Date**  $r_i$ 
  - the timepoint after which the task can be started
- **These dates may be**
  - hard constraints (they must be respected)
  - soft constraints (they may be violated at a cost)

# Earliness/lateness

- **Earliness**
  - the amount of time a task is finished before its due date
  - $\max(0, d_i - t_i)$
- **Lateness**  $T_i$ 
  - the amount of time a task is finished after its due date
  - $\max(0, t_i - d_i)$
- **Latency**  $q_i$ 
  - the period between the end of the task and the end of the schedule
  - $\max(0, C_{\max} - C_i)$
- **Setup**
  - the time required between two consecutive tasks on the same machine

# Time points and periods

- Graphical view as seen in Gantt charts



# Cost

- **Overall Enddate**  $C_{\max}$ 
  - project duration
  - the overall end of the schedule
  - $C_{\max} = \max\{C_i\}$
- **Maximum Lateness**  $T_{\max}$ 
  - the maximum delay of a task
  - $T_{\max} = \max\{T_i\}$
- **Sum of Lateness**
  - a weighted sum of all delays
  - $\text{Sum } w_i T_i$

## **Cost (2)**

- **Earliness**
  - indicates stocks of finished products
- **Delay Indicator**
  - 0/1 variable indicating delay
  - important for jit (100% customer satisfaction)
- **Resource Utilization**
  - important when resource choice is possible
- **Setup Number or Duration**
  - cost of changing/cleaning machines
- **Machine Downtime**
  - times when a resource is idle
- **Stock Levels**

# Classification

- **Comes from OR methodology**
- **Classification of pure problems**
- **Often encountered in reality when considering only main constraints**
  - **Project Planning**
  - **Machine Shop**
    - » **Flow Shop**
    - » **Job Shop**
    - » **Open Shop**

# Project planning

- Precedence or sequence constraints
- No resources
- Cost: optimize  $C_{\max}$
- Very large problem instances (> 10000 tasks)
- Classical algorithms: CPM, PERT
- Deterministic algorithms

# Machine shop

- **Characteristics**
  - Different resources
  - Tasks grouped in jobs
  - Each tasks requires one resource during its duration
  - No choice of resources
  - Each resource can work on one task at a time
  - No preemption
- **Subclasses:**
  - Flow Shop
  - Job Shop
  - Open Shop

# Notation

- **Shorthand problem notation [n/m/F/D]**
  - n jobs
  - m machines
  - F; J Type e. g. Flow shop/ Job Shop
  - D optimization criteria, e. g.  $C_{\max}$

# Flow shop

- **Characteristics:**
  - Precedence constraints between tasks of a job
  - Resources are used by the tasks of all jobs in the same sequence
  - duration of tasks on resources may vary
- **Deterministic Subclasses**
  - $[n/2/F/C_{\max}]$ 
    - » Johnson's algorithm
  - $[n/3/F/C_{\max}]$ 
    - » polynomial under dominance criteria

# Job shop

- **Characteristics:**
  - Precedence constraints between tasks of one job
  - Sequence of resources used may vary
- **Deterministic subclasses**
  - $[n/2/J/C_{\max}]$ 
    - » Jackson's rule
  - Job-shop with 2 jobs

# Open shop

- **Characteristics:**
  - No precedence constraint between tasks
  - Each task uses some machine
- **Deterministic Subclasses**
  - $[n/2/O/C_{\max}]$ 
    - » Gonzalez & Sahni algorithm
- **Many standard methods do not work due to lack of precedence constraints**
  - complexity of choice functions

# Complexity results

- **Most machine shop problems are NP-hard**
  - difficult problem instances exist
  - no “general problem solver”
  - still, good solutions can be found
  - application specific knowledge is useful
- **Special cases for 2 machines**
  - basis for strategies for more complex problems
- **Complexity independent of cost function**
  - research centered on  $C_{\max}$  cost

# OR techniques

- **Linear/Integer Programming**
- **Heuristic Algorithms**
- **Decomposition methods**
  - Hierarchical
  - Structural
  - Temporal/spatial
  - Branch and bound
- **Neighborhood search**
  - Simulated annealing
  - Tabu search
  - Genetic algorithms
- **Relaxation methods**

# Linear/Integer Programming

- **Expression of scheduling problems with**
  - linear programming
  - mixed integer programming methods
- **Often poor result due to non-convexity of solution space**
  - bad initial cost
  - deep backtracking
  - poor cuts
- **Seldom used in practice for detailed scheduling**
- **Used with good results for production planning**

# Heuristic Algorithms

- **Progressive building solutions by adding tasks/jobs one at a time**
- **Items added chosen by heuristics**
- **Good solutions for weakly constrained problems**
- **Bad results for strongly constrained problems**
  - finding admissible solutions
- **Heuristics should take constraints into account**
  - dynamic, not static ordering

# Decomposition techniques

- **Cut problem into more manageable parts**
  - helps handle large/complex problems
- **Hierarchical**
  - projects and sub-projects
  - bottom-up and top-down
  - requires certain problem structure
- **Structural**
  - considering different degrees of freedom independently
  - example: separating scheduling/assignment
- **Temporal/Spatial**
  - solving sub problems for limited time period or limited number of resources

# Branch and bound

- **Create successive sub problems by enumeration on variables**
- **Exploration of search tree**
  - pruning of branches
  - lower bound approximation
- **Standard OR technique**
- **Search strategies must be defined carefully**
- **Very good results for complex problems**
- **High development effort**

# Neighborhood search

- **Search by finding initial solution and “improving” it**
  - feasible initial solution
  - modification function
  - cost evaluation
- **Allows different variations**
  - steepest ascent
  - hill climbing
  - simulated annealing
  - tabu search
  - genetic algorithms
- **Local optimization**

## Neighborhood search (2)

- **Constraint handling**
  - Constraints expressed in cost
  - Modification function checks constraints
- **Good for additive costs**
  - Local changes which improve costs
- **Difficult for very constrained problems**
  - Finding initial solution
  - Admissible modifications

# Relaxation methods

- Solving “simpler” problem helps finding solution to complex problems
- Ignoring/simplifying certain constraints
- Obtain lower/upper bounds
- Proof of optimality
- Initial solutions

# Importance to CLP

- **Defines areas in which CLP approach should (not) be used**
- **Techniques which can be used inside constraint programming**
- **Application use of techniques**
- **Reference results**

## **Part II: Methodology**

- **Constraints: the tools**
- **Solution approach**
- **Strategies**

# Constraints and how they work

- **Finite Domain Solver**
  - based on local consistency techniques
  - incomplete
  - incremental
  - requires enumeration
- **Continuous Solvers (not discussed)**
  - based on Simplex method
  - complete solver
  - solved form
- **Specialized Solvers (not discussed)**
  - Boolean Solver
  - List Constraints
  - Set Constraints

# Finite domains

- **Variables**
  - which range over sets of admissible values (integers)
- **Constraints**
  - which link variables and restrict the values which can be assigned to them
- **Search Strategy**
  - which determine
    - » the order in which variables are assigned
    - » the values which they take

# Finite domain basics

- **Domain definition**
- **Numerical constraints**
  - Inequality constraints
- **Global constraints**
  - Cumulative
  - Diffn
  - Cycle

# Domain definition

- **A variable or list of variables are defined**
- **Different types of domains exist**
- **Examples**
  - $X :: 0..100$ ,
  - $[X_1, X_2, X_3] :: [1,2,4,6,7,11]$
- **Operations on domains**
  - enumerate domain values
  - choose variables depending on their domain
  - get information about domains

# Inequality constraints

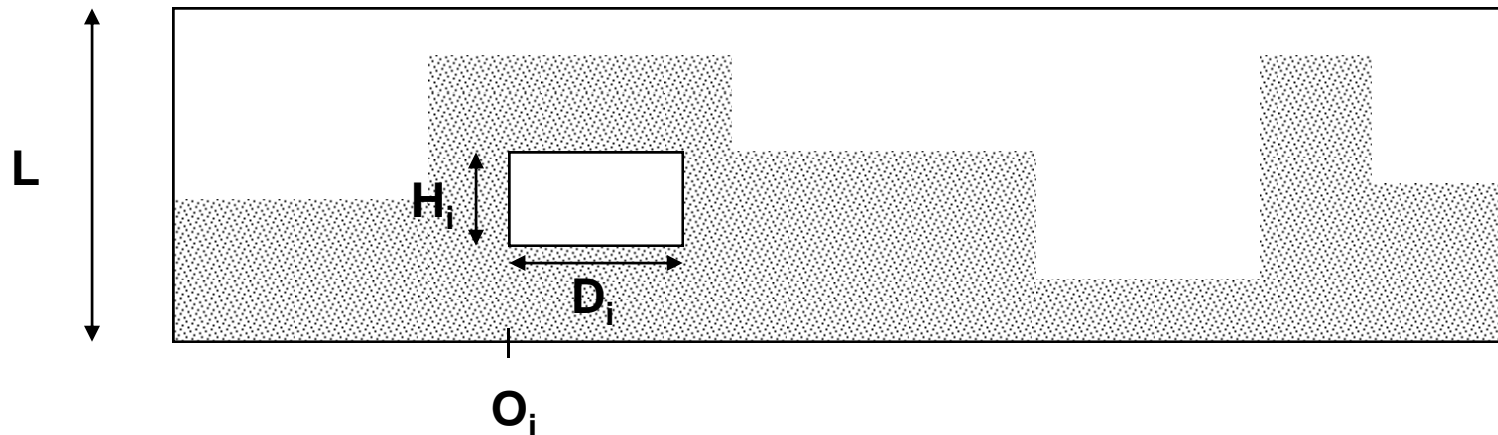
- **Link two variables (simple case)**
  - example:  $X \#>= Y+10$
  - works by updating minima and maxima of the domains
  - consistency technique partial lookahead
  - constraint is woken whenever one of the domains change
- **Link multiple variables (arithmetic constraints)**
  - example:  $X+Y+3*Z \#>= T+5*U$
  - propagation via domain changes
  - generalization of simple case

# Global Constraints

- very high level building blocks
- mix and match combination
- not *specialized* constraints, but general constraints
- integrate different methods and alorithms

# Cumulative constraint

- Graphical Introduction
  - $\text{cumulative}([O_1, O_2, \dots], [D_1, D_2, \dots], [H_1, H_2, \dots], L)$



## Cumulative (2)

- **Mathematical Form**

- $\min = \min\{O_i\}$
- $\max = \max\{O_i+D_i\}-1$
- $\forall i \in [\min, \max]$

$$\sum H_j \leq L \text{ for all } j \text{ with } O_j \leq i \leq O_j+D_j-1$$

- **Global constraint handling**

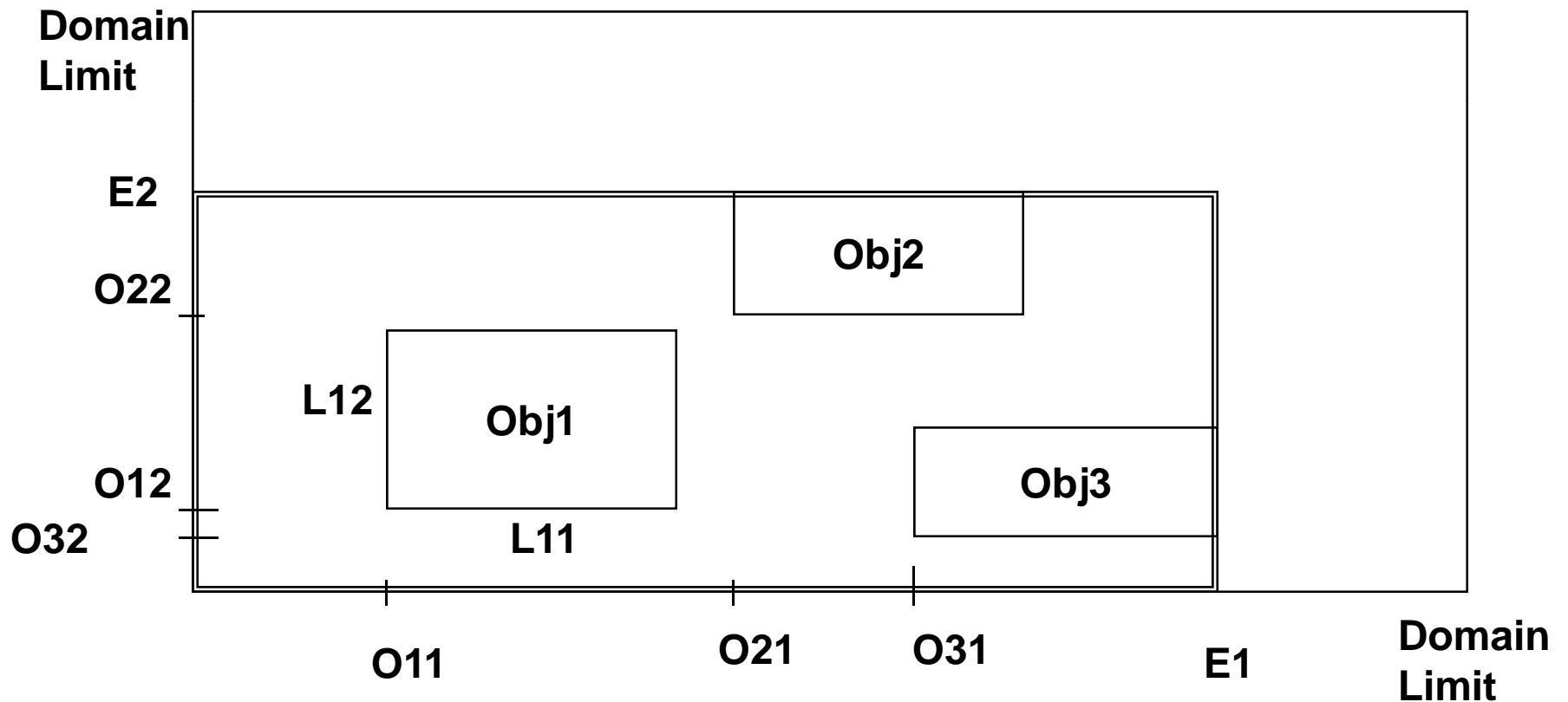
- ~ 20 different incremental algorithms
- 20000 lines in C

- **Only most simple form shown here**

# Diffn constraint

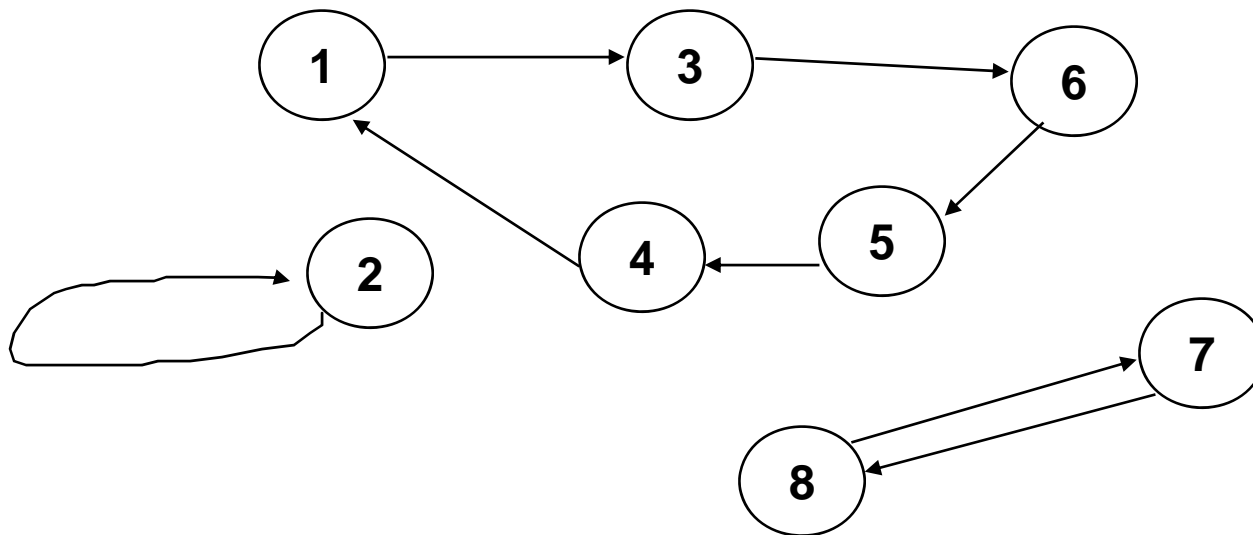
- **Constraint for n-dimensional rectangle overlap**
  - n-dimensional rectangles do not overlap in n-dimensional space and fit in a n-dimensional area
- **Form**
  - $\text{diffn}([\![O_{11}, O_{12}, O_{13}, \dots, L_{11}, L_{12}, L_{13}, \dots]\!], \dots, [E_1, E_2, \dots])$ 
    - » Origin  $O_{ij}$
    - » Length  $L_{ij}$
    - » End  $E_j = \max\{O_{ij} + L_{ij}\}$
- **Useful for**
  - 2D placement/ packing
  - 3D placement/ packing
  - Vehicle loading

# Two dimensional placement



# Cycle constraint

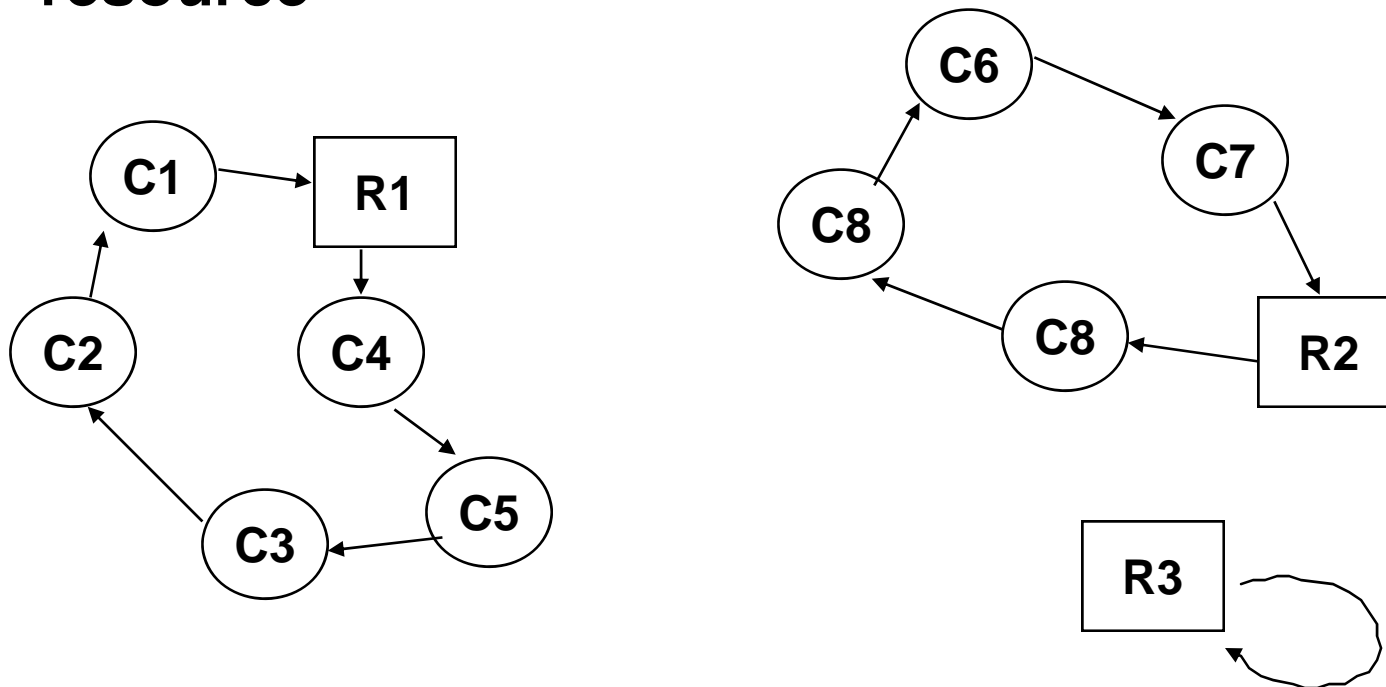
- Find cycles in directed graphs
- Example: 8 nodes, 3 cycles



`cycle(3,[3,2,6,1,4,5,8,7],...)` : cycles written as permutations

# Tour generation

- Nodes are cities and resources
- One resource per cycle
- All cities in one cycle are visited by one resource



# Cycle constraint in scheduling

- **Nodes = tasks and resources**
- **Domain variables = successor nodes = next task on resource**
- **Domains = possible successors**
- **Weight of nodes = duration of tasks or setup**
- **Weight of nodes = work time of resource**
- **Special assignment nodes = resources**
- **Origin of nodes = start times**

# Solution approach

- **Expressing scheduling problems with CLP**
- **Use of the finite domain solver**
  - continuous solver used for planning problems
- **Problem statement defines**
  - Variables
  - Constraints
  - Strategy
- **Modelling close to problem**
  - expressive power of constraints
  - search concept part of the language

# Variables

- **Decision variables**
  - variables correspond to decisions in the schedule
- **Time Granularity**
  - defines size of domains
  - large domain = large search space
- **Calendars**
  - include/exclude particular dates
    - » use of explicit domains
    - » mapping real dates on schedule dates
- **Release/ Due dates**
  - give smallest/largest value in domain

# Example

- Link between calendar and domains

Calendar

Monday											Tuesday												
0	2	4	6	8	10	12	14	16	18	20	22	0	2	4	6	8	10	12	14	16	18	20	22

Domain (explicit domain)

0	2	4	6	8	10	12	14	16	18	20	22	24	26	30	32	34	36	38	40	42	44	46	48
---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Domain (mapping)

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

# Constraint types in scheduling

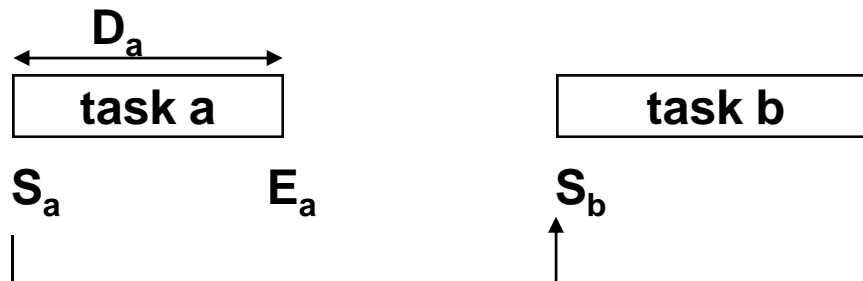
- **Precedence**
- **Sequence**
- **Disjunctive Machines**
- **Cumulative Machines**
- **Consumable Resources**
- **Machine Choice**
- **Setup**
- **Preemption**
- **Cost**

# Methodology

- **How to express different constraints**
- **Alternative models**
- **Use of global constraints**
- **Building blocks for applications**

# Precedence

- One task must be finished before another
- Expressed as inequalities
- Constraints
  - between variables denoting start dates or
  - between variables for start and end dates



$$S_b \geq S_a + D_a$$



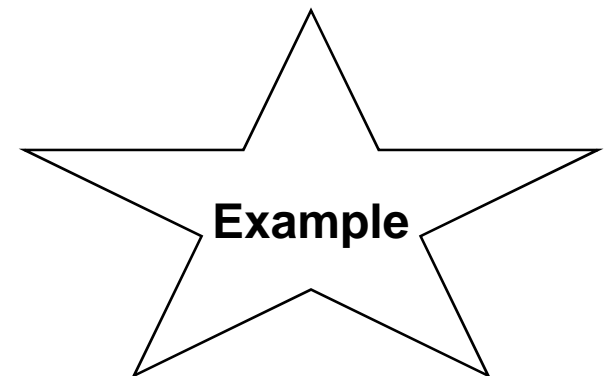
$$S_b \geq E_a$$

# Sequence

- **Sequence constraints link start or end dates**
  - Start-Start
  - Start-End
  - End-Start
  - End-End
- **Express Minimum/Maximum Limit**
- **Efficient handling by constraint propagation**
  - but problems with very large sets of constraints

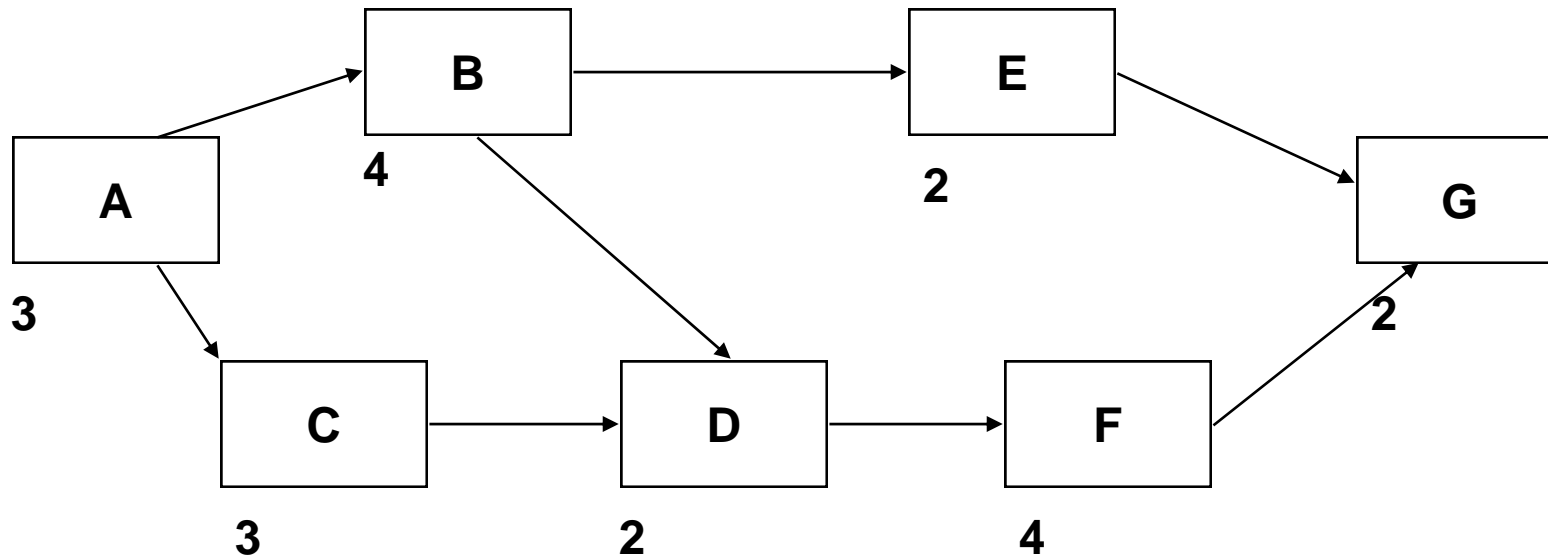
# Scheduling and planning examples

- **Example problems**
  - simple
  - small
  - not realistic
- **CHIP Modelling**
  - actual code
  - simple structure
- **Results**
  - to help understanding



# Project planning

- Simple project example

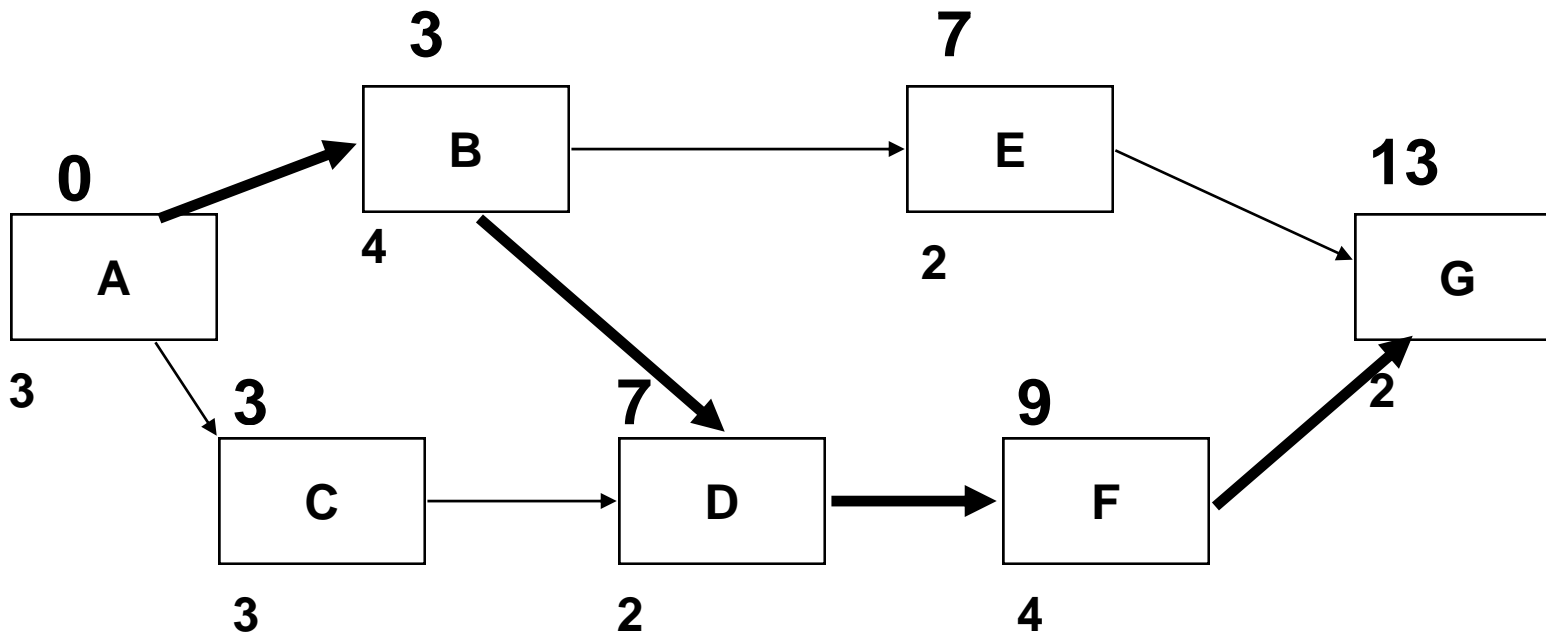


# Modelling

**[Sa,Sb,Sc,Sd,Se,Sf,Sg] :: 0..100,**  
**[Da,Db,Dc,Dd,De,Df,Dg] = [3,4,3,2,2,4,2],**  
**Sb #>= Sa + Da,**  
**Sc #>= Sa + Da,**  
**Sd #>= Sb + Db,**  
**Sd #>= Sc + Dc,**  
**Se #>= Sb + Db,**  
**Sf #>= Sd + Dd,**  
**Sg #>= Se + De,**  
**Sg #>= Sf + Df,**  
**indomain(Sg),**  
**labeling([Sa,Sb,Sc,Sd,Se,Sf,Sg]),**

# Project planning results

- Simple project example



# Disjunctive resources

- **Constraint Handled**
  - A task uses resource exclusively during its duration
  - Unavailability periods of resources
- **Modelling**
  - Constraint as Choices (too weak)
  - Conditional Propagation (too weak)
  - Cumulative constraint
- **Alternative Modelling**
  - Job sequencing

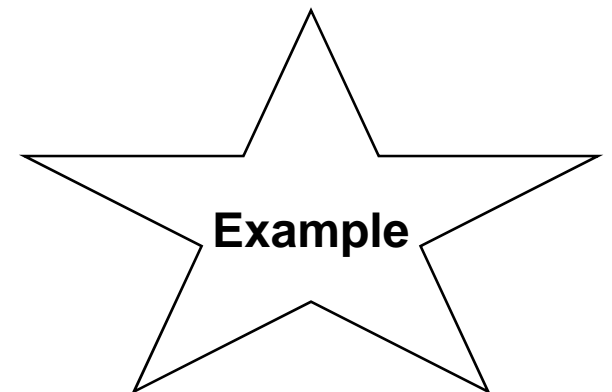
# Cumulative formulation

- **Express disjunctive with one cumulative constraint**
  - $\text{cumulative}([S_a, S_b, \dots], [D_a, D_b, \dots], [1, 1, \dots], 1, \dots)$
- **Active use of the constraint**
- **Recognition of capacity problems**
  - available time does not allow all jobs to be scheduled
  - bottleneck periods when some jobs have to be scheduled
  - possibility to link directly with project end via cumulative constraint parameter
- **Limited interaction with precedence constraints**

# Alternative formulation

- **Express disjunctive as sequence of tasks on a machine**
- **Instead of**
  - variable Task\_x, value startdate
  - variable first task, value task\_x
- **“Classical” formulation**
- **Problem when combining with other (e.g. precedence) constraints**

# Simple job-shop example



## Jobs and tasks

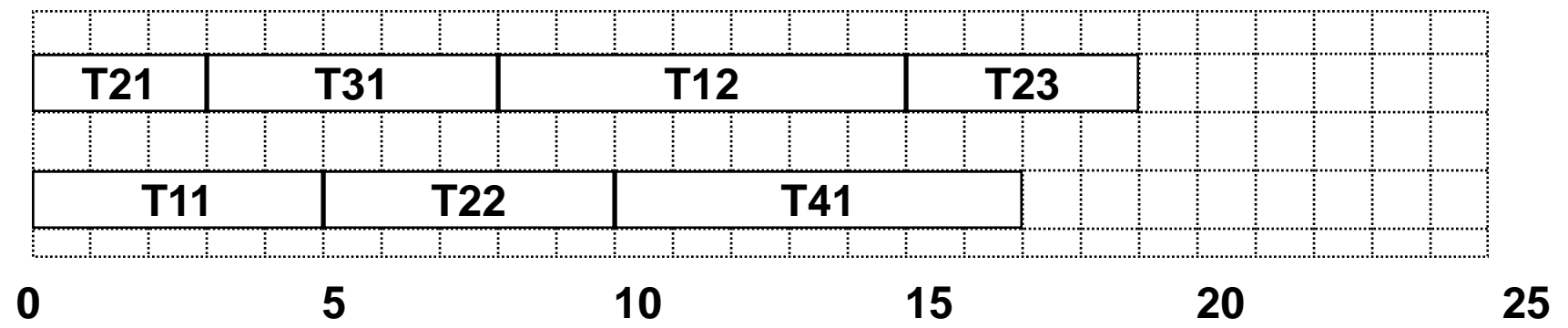
<b>Job</b>	<b>Task</b>	<b>Duration</b>	<b>Machine</b>
<b>J1</b>	<b>T11</b>	<b>5</b>	<b>M1</b>
<b>J1</b>	<b>T12</b>	<b>7</b>	<b>M2</b>
<b>J2</b>	<b>T21</b>	<b>3</b>	<b>M2</b>
<b>J2</b>	<b>T22</b>	<b>5</b>	<b>M1</b>
<b>J2</b>	<b>T23</b>	<b>4</b>	<b>M2</b>
<b>J3</b>	<b>T31</b>	<b>5</b>	<b>M2</b>
<b>J4</b>	<b>T41</b>	<b>7</b>	<b>M1</b>

# Job shop program

End = 100,  
[T11,T12,T21,T22,T23,T31,T41] :: 0..End,  
[E1,E2,E3,E4] :: 0..End,  
[D11,D12,D21,D22,D23,D31,D41] = [5,7,3,5,4,5,7],  
  
T12 #>= T11+D11, T22 #>= T21+D21, T23 #>= T22+D22,  
E1 #= T12+D12, E2 #= T23+D23,  
E3 #= T31+D31, E4 #= T41+D41,  
cumulative([T11,T22,T41],[D11,D22,D41],[1,1,1],1),  
cumulative([T12,T21,T23,T31],[D12,D21,D23,D31],[1,1,1,1],1),  
  
min\_max(labeling([T11,T12,T21,T22,T23,T31,T41]),  
[E1,E2,E3,E4]),

# Job shop solution

Minimize  $C_{max}$   
 Optimal solution 19



# Cumulative machines

- **Constraint Handled**
  - Alternative Machines
  - Manpower
  - Availability Profiles
- **Generalization of disjunctive case**
- **Constraint Modelling**
  - sets of 0/1 variables
  - cumulative formulation
- **Relation to Machine Assignment**

# Modelling

- **Required for each task**

- start date  $S_i$
- duration  $D_i$
- resource consumption  $R_i$

- **Global information**

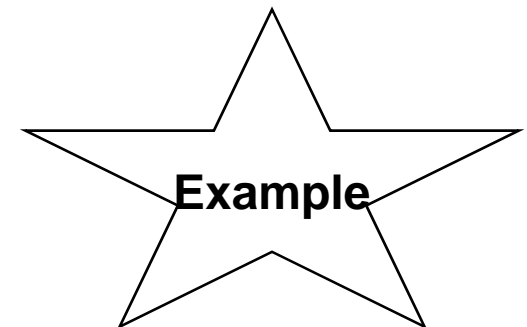
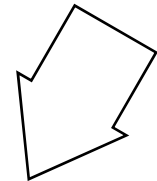
- Overall resource limit  $R$

$R_x :: 0..R,$

$\text{cumulative}([S_a, S_b, \dots], [D_a, D_b, \dots], [R_a, R_b, \dots], R_x, \dots)$

# Multiple machines

- End = 100,
- [T11,T12,T21,T22,T23,T31,T41] :: 0..End,
- [E1,E2,E3,E4] :: 0..End,
- [D11,D12,D21,D22,D23,D31,D41] = [5,7,3,5,4,5,7],
  
- T12 #>= T11+D11, T22 #>= T21+D21, T23 #>= T22+D22,
- E1 #= T12+D12, E2 #= T23+D23,
- E3 #= T31+D31, E4 #= T41+D41,
- cumulative([T11,T22,T41],[D11,D22,D41],[1,1,1],1),
- **cumulative([T12,T21,T23,T31],[D12,D21,D23,D31],[1,1,1,1],2),**
- 
- min\_max(labeling([T11,T12,T21,T22,T23,T31,T41]),
- [E1,E2,E3,E4]),

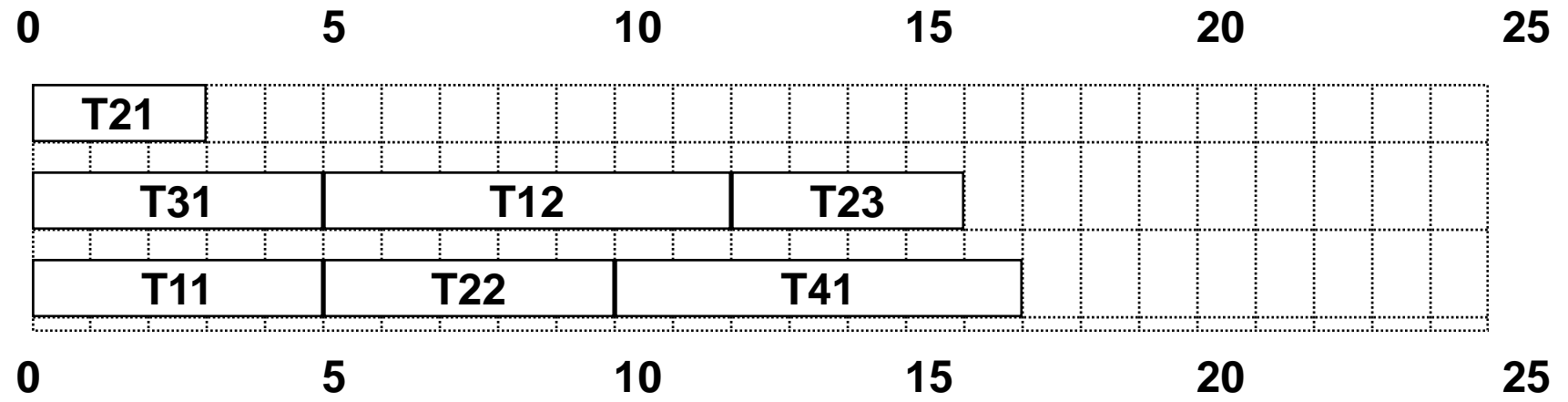


# Multiple machines

Two copies of machine 2

Cumulative solution - not assignment

Optimal solution cost 17



# Manpower use

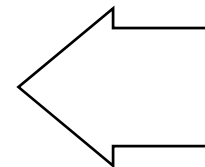
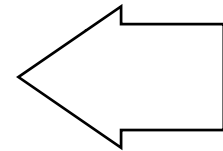
<b>Job</b>	<b>Task</b>	<b>Duration</b>	<b>Machine</b>	<b>Manpower</b>
<b>J1</b>	<b>T11</b>	<b>5</b>	<b>M1</b>	<b>2</b>
<b>J1</b>	<b>T12</b>	<b>7</b>	<b>M2</b>	<b>3</b>
<b>J2</b>	<b>T21</b>	<b>3</b>	<b>M2</b>	<b>4</b>
<b>J2</b>	<b>T22</b>	<b>5</b>	<b>M1</b>	<b>3</b>
<b>J2</b>	<b>T23</b>	<b>4</b>	<b>M2</b>	<b>3</b>
<b>J3</b>	<b>T31</b>	<b>5</b>	<b>M2</b>	<b>2</b>
<b>J4</b>	<b>T41</b>	<b>7</b>	<b>M1</b>	<b>4</b>

# Manpower model

```

End = 100,
[T11,T12,T21,T22,T23,T31,T41] :: 0..End,
[E1,E2,E3,E4] :: 0..End,
[D11,D12,D21,D22,D23,D31,D41] = [5,7,3,5,4,5,7],
Manpower = 6,
[U11,U12,U21,U22,U23,U31,U41] = [2,3,4,3,3,2,4],
T12 #>= T11+D11,T22 #>= T21+D21,T23 #>= T22+D22,
E1 #= T12+D12,E2 #= T23+D23,
E3 #= T31+D31,E4 #= T41+D41,
cumulative([T11,T22,T41],[D11,D22,D41],[1,1,1],1),
cumulative([T12,T21,T23,T31],[D12,D21,D23,D31],[1,1,1,1],2),
Manp :: 0..Manpower,
cumulative([T11,T12,T21,T22,T23,T31,T41],
           [D11,D12,D21,D22,D23,D31,D41],
           [U11,U12,U21,U22,U23,U31,U41],Manp),
min_max(labeling([T11,T12,T21,T22,T23,T31,T41]),
         [E1,E2,E3,E4]),

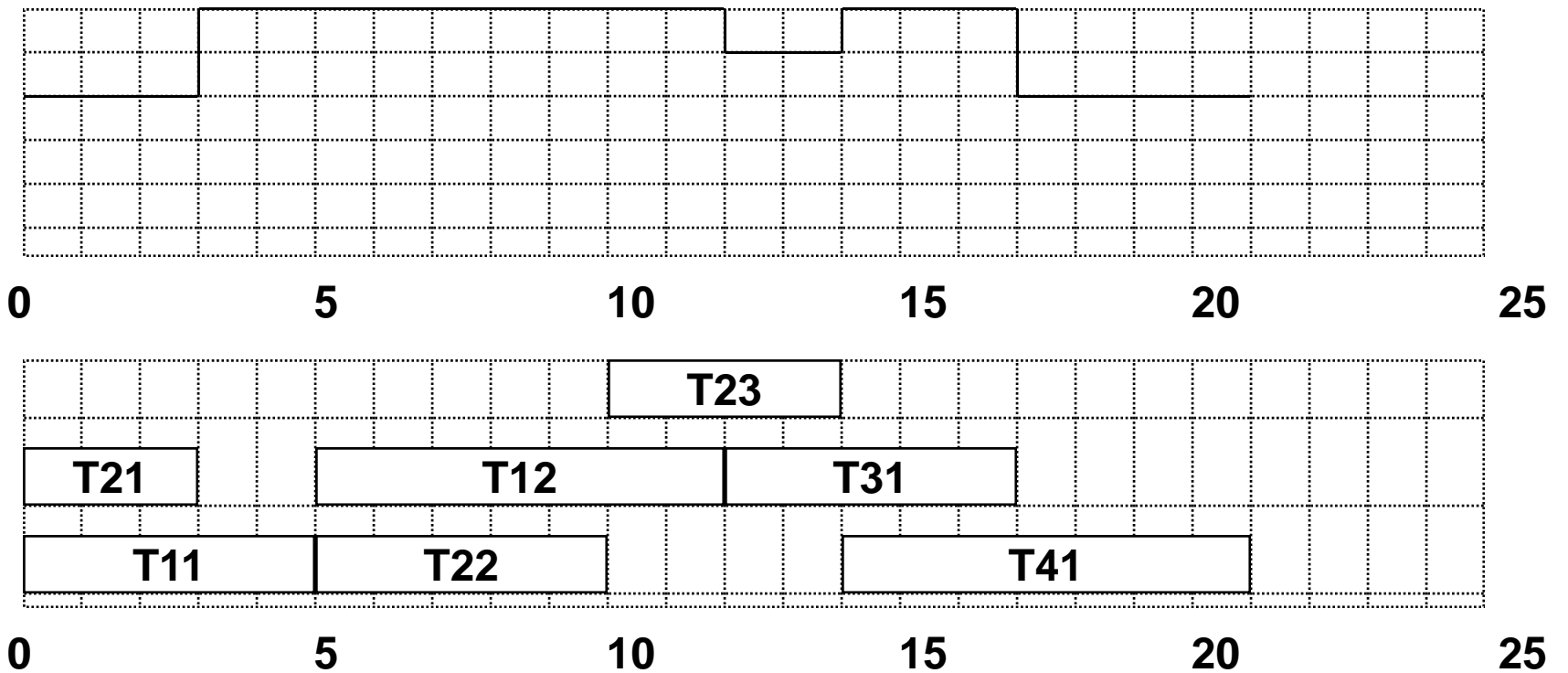
```



# Manpower result

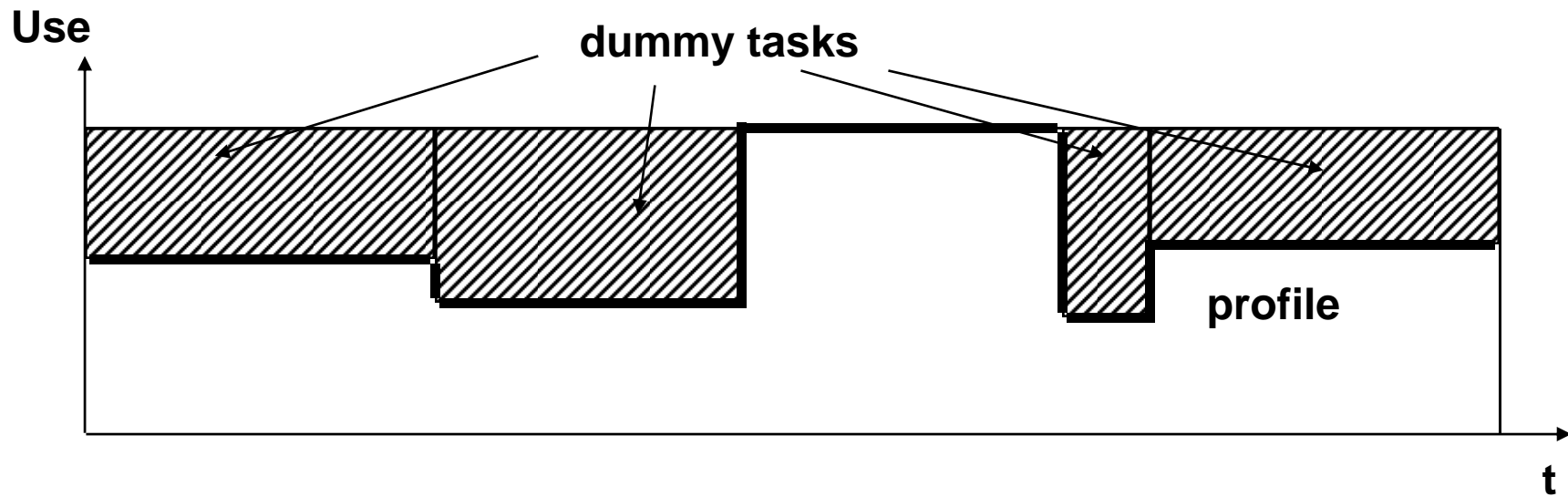
Optimum cost 21

Manpower use 6



# Unavailability

- **Handling of availability profiles**
  - introduction of fixed dummy tasks
- **resource use defined by max-availability**



# Relation to assignment

- **The cumulative problem states a**
  - necessary
  - but not sufficient
  - condition for solving the assignment problem
- **Assignment problem is trivial, if machine assignment can be preemptive**
- **Typically, assignment can be solved easily, if a few resource switches are allowed**
- **If assignment is very important, other constraints must be added**

# Consumable resources

- **Constraint Handled**
  - Raw Materials
  - Money
  - Stocks
  - Receivings
  - Transfers
  - Orders
- **Idea**
  - Handle consumable resources as a special type of renewable resource
  - Time points become periods
    - » from start to time point or
    - » from time point to end

# Producer constraint

- **Example: Finished Products**
- **Demand given for fixed time point and fixed quantities**
  - time points
  - quantity required
- **Determine the dates when to produce tasks in order to satisfy demand**
- **For each task which produces the product**
  - enddate
  - (fixed) quantity produced

# Consumer constraint

- **Example: Raw Materials**
- **Supply given for fixed time point and fixed quantities**
  - time points
  - quantity required
- **Determine the dates when to produce tasks in order to respect supply**
- **For each task which consumes supply**
  - enddate
  - (fixed) quantity consumed

# Producer consumer

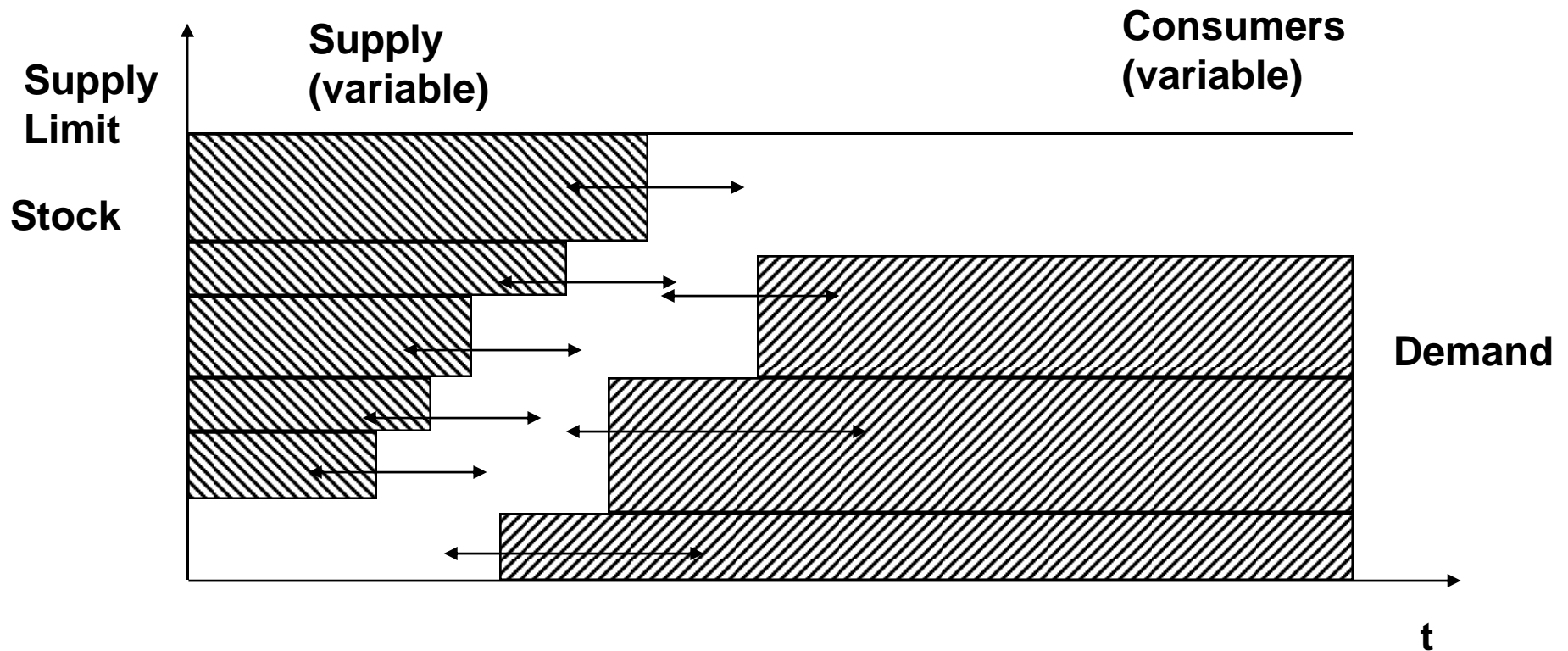
- **Example: Intermediate Products**
- **Material produced by Producer tasks**
  - enddates
  - (fixed) quantity
- **Material consumed by Consumer tasks**
  - start dates
  - (fixed) quantity
- **Additional initial stock**
- **Constraint:**
  - no negatives
  - stock must be produced before it is consumed

## **Producer consumer (2)**

- **Producer events are represented by**
  - tasks from overall start to end of task
- **Consumer events are represented by**
  - tasks from start of task to overall end
- **Resource limit is set as sum of all produced quantities + initial stock**

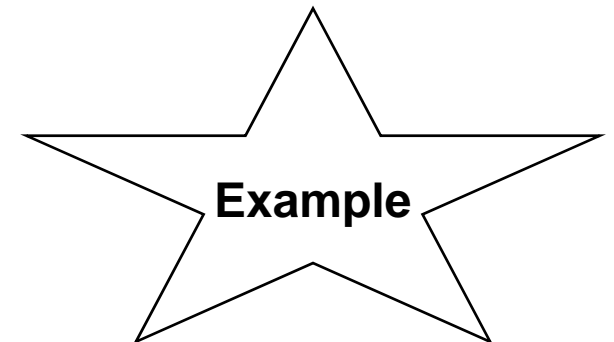
# Cumulative Modelling

- graphical representation of producer/ consumer constraints



# Product requirements

<b>Produce</b>	<b>Qty</b>	<b>Consume</b>	<b>Qty</b>
<b>J1</b>	<b>60</b>	<b>6</b>	<b>60</b>
<b>J2</b>	<b>70</b>	<b>12</b>	<b>40</b>
<b>J3</b>	<b>50</b>	<b>21</b>	<b>100</b>
<b>J4</b>	<b>40</b>		
<b>0</b>	<b>20</b>		



# Producer model

**End = 100,**

**Stock = 20,**

**[Q1,Q2,Q3,Q4] = [60,70,50,40],**

**Limit :: 0..1000,**

**Limit #<= Stock + Q1 + Q2 + Q3 + Q4,**

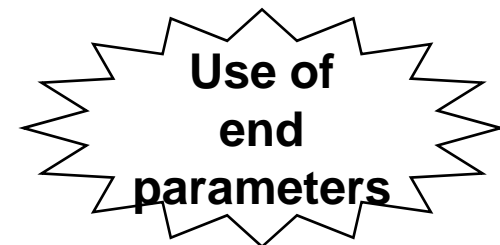
**cumulative([0,0,0,0,6,12,21],**

**[E1,E2,E3,E4,94,88,79],**

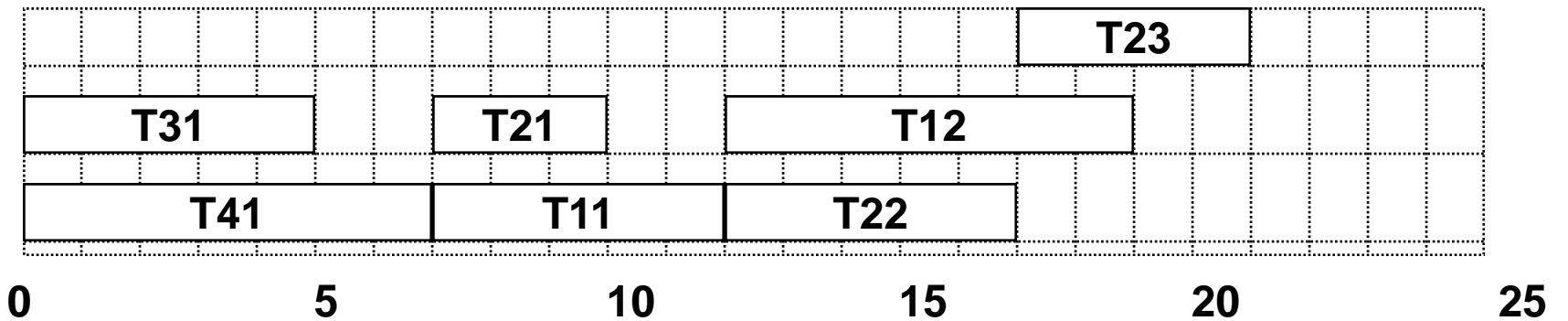
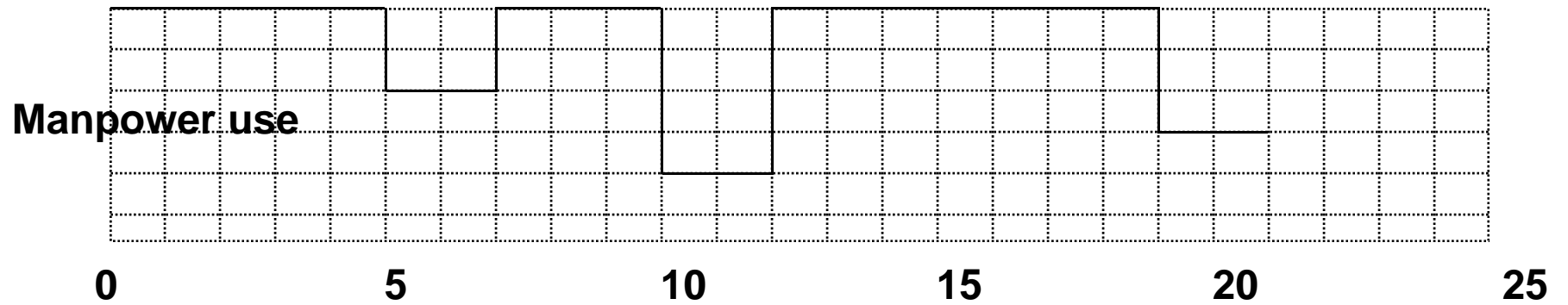
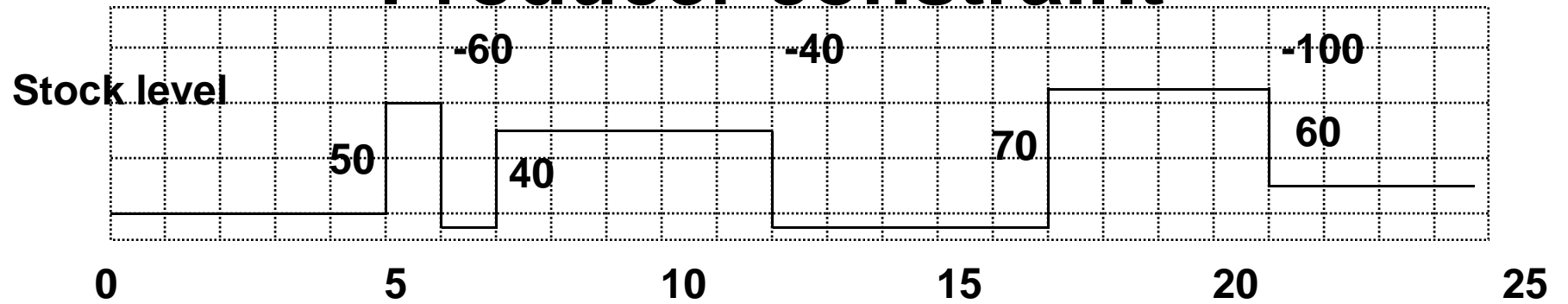
**[Q1,Q2,Q3,Q4,60,40,100],**

**[E1,E2,E3,E4,End,End,End],**

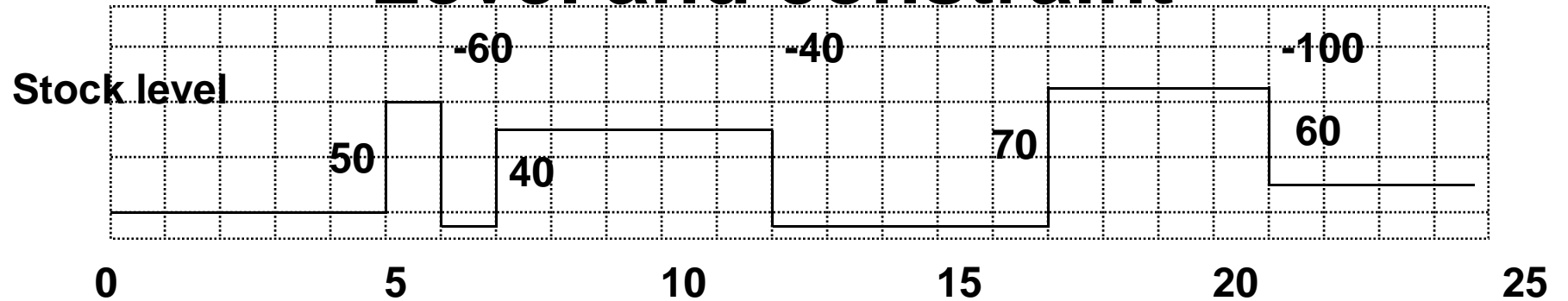
**unused,Limit,End,unused),**



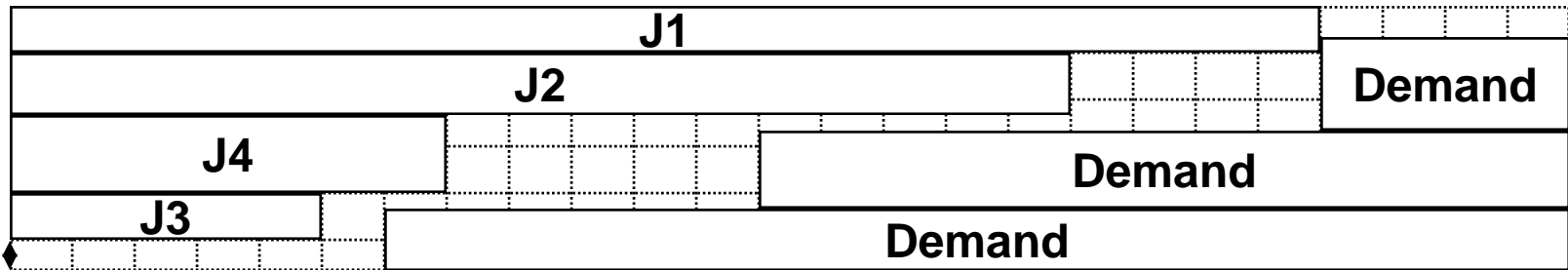
# Producer constraint



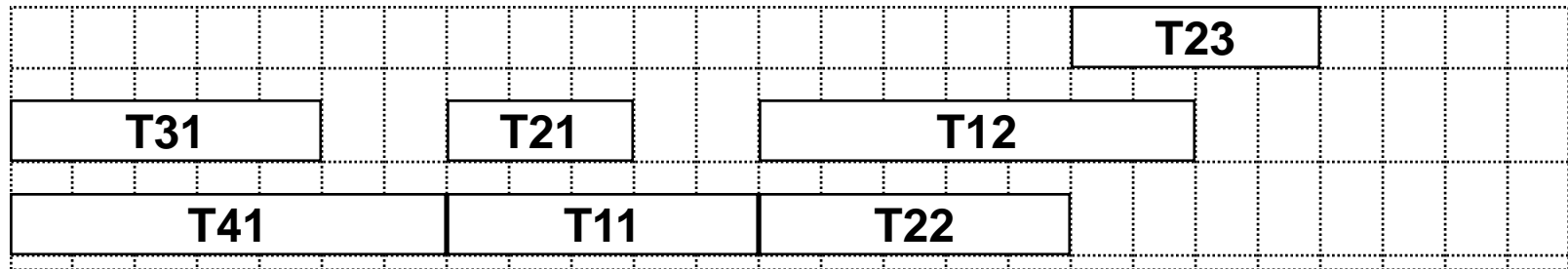
# Level and constraint



Producer constraint



Stock



# Extensions

- **Variable resource consumption/production**
  - variable height of rectangles + dummy tasks
- **Allow (limited) negative stock levels**
  - maximum negative stock at any time
  - total negative stock\*time amount
  - use of intermediate resource limit in cumulative
- **Splitting resource use**
  - consumer tasks do not need all material in the beginning
  - producer tasks produce some material before the end
  - pipelining operations
  - jit production
- **Safety margins**

# Limited stock levels

- **Problem**
  - only limited amount of stocks can be stored
  - production can not be arbitrarily long before consumption
- **Solution: another call to cumulative**
- **Idea:**
  - inverse roles of producer and consumer
  - add maximum stock level

# Machine choice

- **Intuition:**
  - tasks may run on different machines, but may not overlap each other
  - each machine is running at most one task at a time
  - sets of possible machines for different tasks may overlap partially
- **Constraint Handled:**
  - Alternative, not equivalent machines
  - different speed of machines (different capabilities)
  - machine dependent duration of tasks
  - flexible work cells

# Modelling

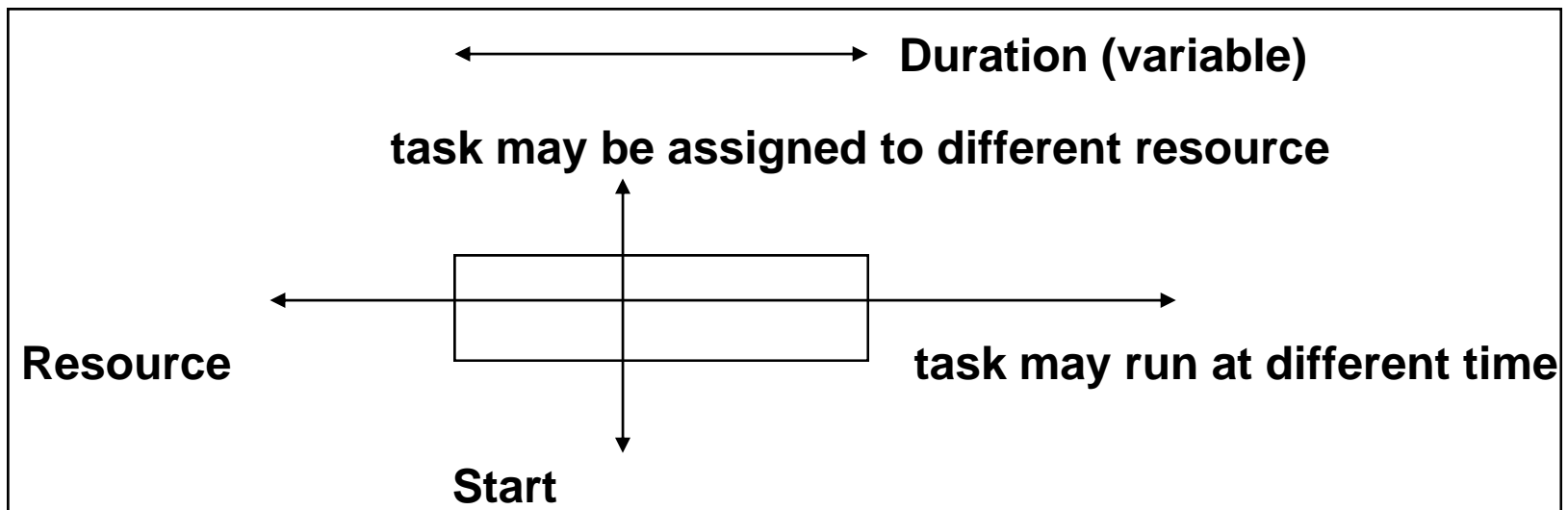
- **Additional degree of freedom**
  - dramatic increase of search space
- **New decision variable**
  - Machine assignment variable  $M_i$
- **element constraint (functional dependencies)**
  - link between machine and duration
  - $\text{element}(M_i, [\dots], p_i)$
- **Decomposition possible**
  - decide machine assignment beforehand
  - load balancing
  - select “best” machine for tasks

# Non overlapping constraint

- **Mathematical Formulation**
  - for two tasks  $i$  and  $j$  we have
    - »  $M_i \setminus M_j$  or
    - »  $t_i \geq t_j + p_j$  or
    - »  $t_j \geq t_i + p_i$
- **Modelling**
  - **Conditional Propagation**
    - » far too weak
  - **Rectangle Placement**
    - » diffn constraint
  - **Redundant cumulative constraint**
    - » Relaxation placement -> cumulative

# Diffn model

- Constraint expressed as 2D placement
- X-dimension = time
- Y-dimension = resource
- tasks are rectangles with height 1

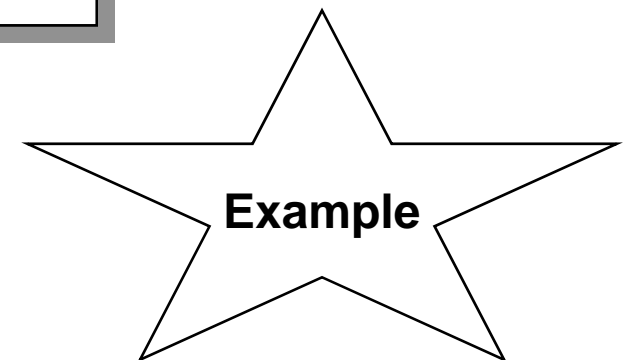


# Redundant cumulative

- **Relaxation**
  - each task has resource use one
  - each resource makes one resource unit available
    - » resource limit = number of machines
  - cumulative constraint on resource use
    - » over all tasks
    - » duration of tasks variable
  - good for “full” schedules
    - » all machines busy
    - » update of time periods of remaining tasks
- **Redundant constraint adds reasoning power**

# Machine speed

<b>Task</b>	<b>M1</b>	<b>M2</b>	<b>M3</b>
<b>T11</b>	<b>5</b>	<b>3</b>	<b>3</b>
<b>T12</b>	<b>7</b>	<b>7</b>	<b>7</b>
<b>T21</b>	<b>4</b>	<b>3</b>	<b>3</b>
<b>T22</b>	<b>5</b>	<b>7</b>	<b>7</b>
<b>T23</b>	<b>4</b>	<b>4</b>	<b>4</b>
<b>T31</b>	<b>6</b>	<b>5</b>	<b>5</b>
<b>T41</b>	<b>7</b>	<b>6</b>	<b>6</b>

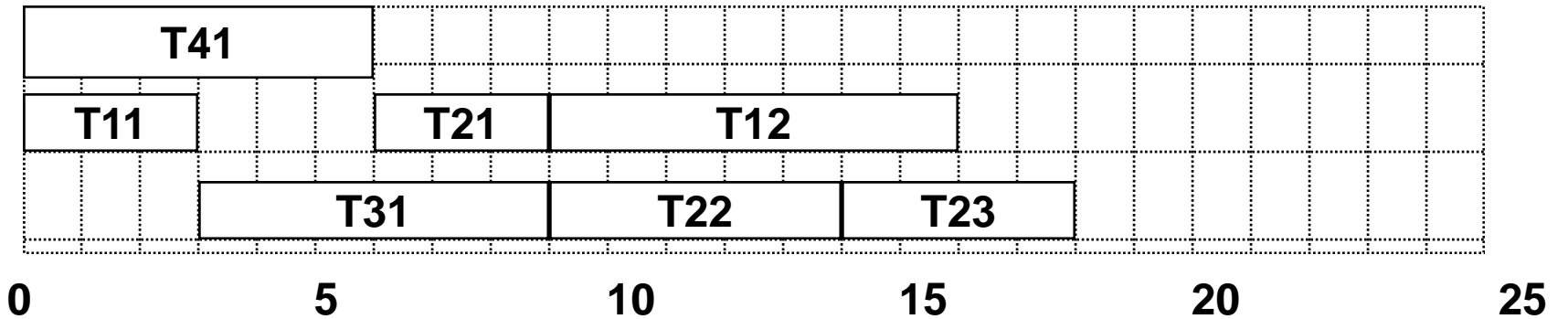
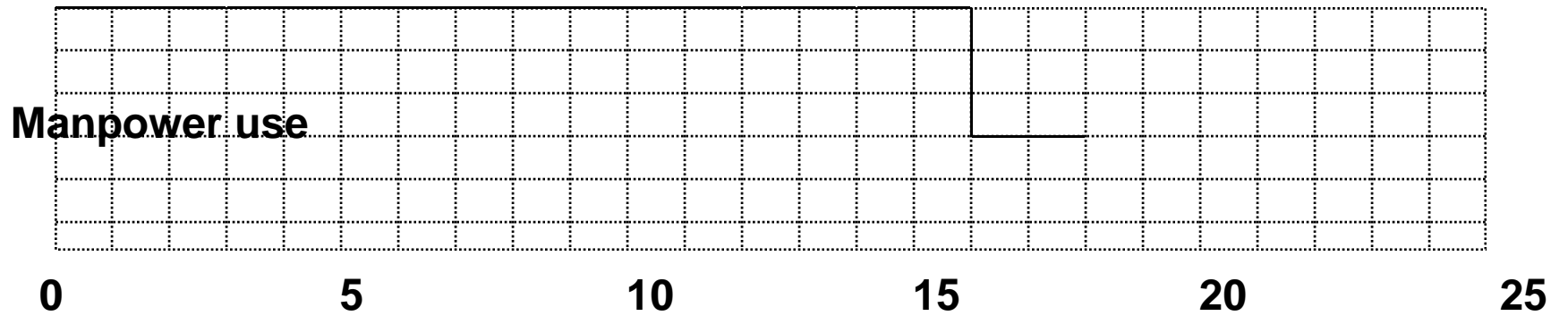
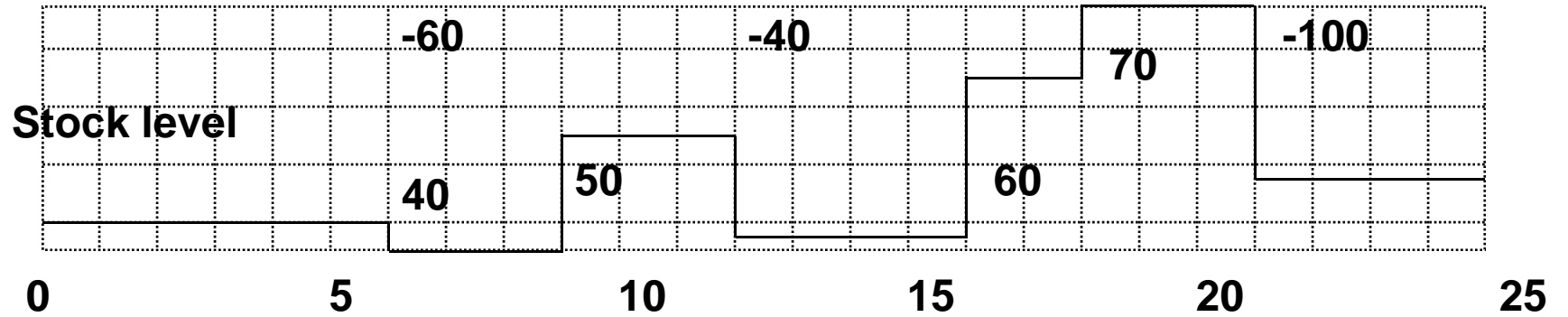


# Machine choice model

[M11,M12,M21,M22,M23,M31,M41] :: 1..3,  
machine\_duration([M11,M12,M21,M22,M23,M31,M41],  
[D11,D12,D21,D22,D23,D31,D41],  
[[5,3,3],[7,7,7],[4,3,3],[5,7,7],[4,4,4],[6,5,5],[7,6,6]]),  
diffn([[T11,M11,D11,1],[T12,M12,D12,1],[T21,M21,D21,1],  
[T22,M22,D22,1],[T23,M23,D23,1],  
[T31,M31,D31,1], [T41,M41,D41,1]]),  
min\_max(labeling([T11,T12,T21,T22,T23,T31,T41,M11,  
M12,M21,M22,M23,M31,M41]),[E1,E2,E3,E4]),



# Machine choice



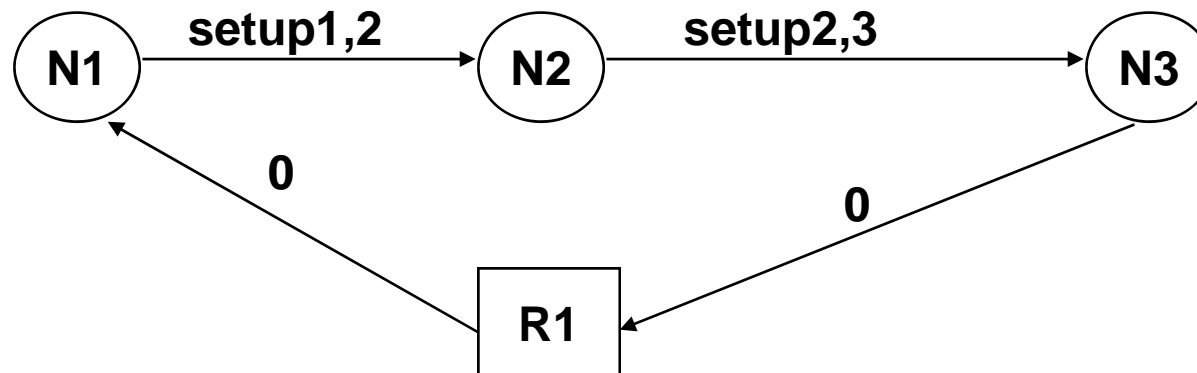
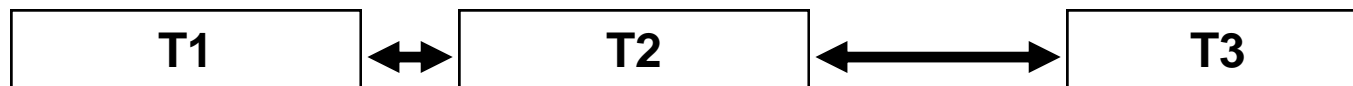
# Sequence dependent setup

- **Constraint Handled:**
  - Tooling
  - Cleaning
  - Adjustment Periods
- **Constraint Modelling**
  - Cycle constraint (difficult problems)
  - Diffn constraint (large problems)
    - » exclusion markers, not discussed here

# Setup cycles

- All tasks on one disjunctive resource form a cycle
- weight of each node = setup time (depends on successor)
- cycle with minimum weight = minimum setup

R1



# Setup relaxation

- **Problem:**
  - setup depends not just on one task; but on two
  - a priori unknown sequence of tasks
  - lower bound often zero
- **Relaxation of problem:**
  - Ignore setup times if not significant
    - » e.g. 5 min setup for 2 hour job
  - Include setup in duration of task if not sequence dependent
    - » extend duration of tasks
  - Over-estimate setup by worst case assumption

# Machine/sequence dependent setup

- **Constraint Handled:**
  - Setup on between tasks which are assigned to the same machine
  - Combination with machine choice
- **Modelling**
  - cycle constraint
  - multiple cycles in one constraint
  - generalisation of one machine case

# Preemption

- **Constraint Handled:**
  - Breaks
  - Shift Pattern
  - Weekend
  - Machine dependent calendars
  - Multi-tasking
- **Constraint Modelling**
  - Preemption by Downtime
    - » element constraints
    - » Inequalities
  - Preemption by task switch
    - » Paradigm switch

# Cost

- **Cost reporting**
  - A cost value that should be computed for a solution
  - Not updated inside the search routine
- **Cost constraint**
  - A cost which is handled as a constraint in the search
  - Updated inside the constraint propagation
- **Cost approximation**
  - A constraint which is not a cost of the application, but added to improve pruning in the search
- **Different cost constraints have different search behavior**

# Cost: maximum end date

- **Typical: Overall Project End**
  - related to machine utilization
- **Expression:**
  - explicit end task
    - » inequalities
  - general domain limit
- **Good cost estimation by constraints**
- **Good propagation from cost limit**

# Cost: sum of lateness

- **Scheduling with due-dates**
- **Expressed by**
  - computation of lateness
  - constraint approximation
    - »  $T_i \geq t_i + p_i - d_i$
  - arithmetic constraint on (weighted) sum
- **Reasonable cost estimation by constraints**
  - added estimate of individual lateness
- **Poor propagation from cost limit**
  - individual lateness only limited very late in search tree

# Cost: maximum lateness

- **Scheduling with due-dates**
- **Expressed by**
  - computation of lateness
  - constraint approximation
    - »  $T_i \geq t_i + p_i - d_i$
  - inequalities to each lateness
- **Good cost estimation by constraints**
  - estimate of individual lateness
- **Good propagation from cost limit**
  - individual lateness immediately

# Cost: preferred start date

- **JIT scheduling**
  - stock minimization
  - delay minimization
- **Expressed by**
  - two variables lateness and earliness
  - each approximated by inequality
- **Restricts domains to (small) intervals**
  - good propagation
  - possibility of failure

# Cost: resource usage

- **Optimizing factory throughput**
- **Machine choice**
- **Expressed by**
  - Sum of duration  $p_i$
- **Very weak constraint**
- **Use as a strategy**
  - run task on fastest machine

# Cost: stock level

- **Stock**
  - Stock Level
  - Maximum Stock Level
  - Stock Volume
  - Negatives
- **Expressed by**
  - cumulative constraints
- **Good estimates**
  - intermediate resource levels in cumulative
- **Poor propagation from constraint limit**
  - early use of all available amount
  - constraint too restrictive for remaining schedule

# Strategies

- **Need for strategies**
  - constraint solver not complete (complexity)
  - standard strategies give only average results
  - use of domain knowledge (intelligence)
- **Heuristic solutions**
  - find one (or few) solutions with a good quality
  - strategy used for finding good solution
- **Optimization procedures**
  - control search procedure to prove optimality
  - strategy used to limit search tree

# Heuristics

- **Degrees of Freedom**
  - **Inside application domain**
    - » **choice of model**
    - » **relaxation**
    - » **decomposition**
  - **Inside constraint Modelling**
    - » **variable assignment**
    - » **search strategy**

# Co-operation

- **Constraints and Heuristics should work together**
  - Constraints provide pruning which helps in selecting next choice
  - Dynamic heuristics
    - » choice re-evaluated at each step
  - Better than static heuristics
    - » based only on static picture
- **Constraints allow good heuristics**
  - Without constraints heuristic must simulate its effect

# Assignment strategy

- **Choose variable**
  - Depending on domains of variables
  - Depending on constraints
  - Depending on heuristics
- **Choose value**
  - Use of the restricted domain
  - Use of heuristics
- **Alternative:**
  - Choose set of alternative constraints
  - Choose among alternatives

# Labeling strategy

- **Two possible extremes; mixed forms possible**
- **Value choice**
  - Deterministic choice of variable
  - Non-deterministic choice of value
- **Variable choice**
  - Non-deterministic choice of variable
  - Deterministic choice of value
- **Heuristics version:**
  - Deterministic choice of variable
  - Deterministic choice of value

# Value choice

- **Often used for CSP problems**
  - First fail strategy
  - etc
- **Supported by primitives**
  - labeling
  - delete
  - indomain
- **Typical code**

```
labeling([]).  
labeling([H|T):-  
    delete(X,[H|T],R,0,first_fail),  
    indomain(X),  
    labeling(R).
```

# Variable choice

- **Often used in OR methods**
  - Left-most strategy
    - » choose a task and fix it on its earliest start
    - » all other tasks do not start earlier
    - » on backtracking, try another variable
- **Use of Prolog to write choice strategy**
  - Simple example delete/3

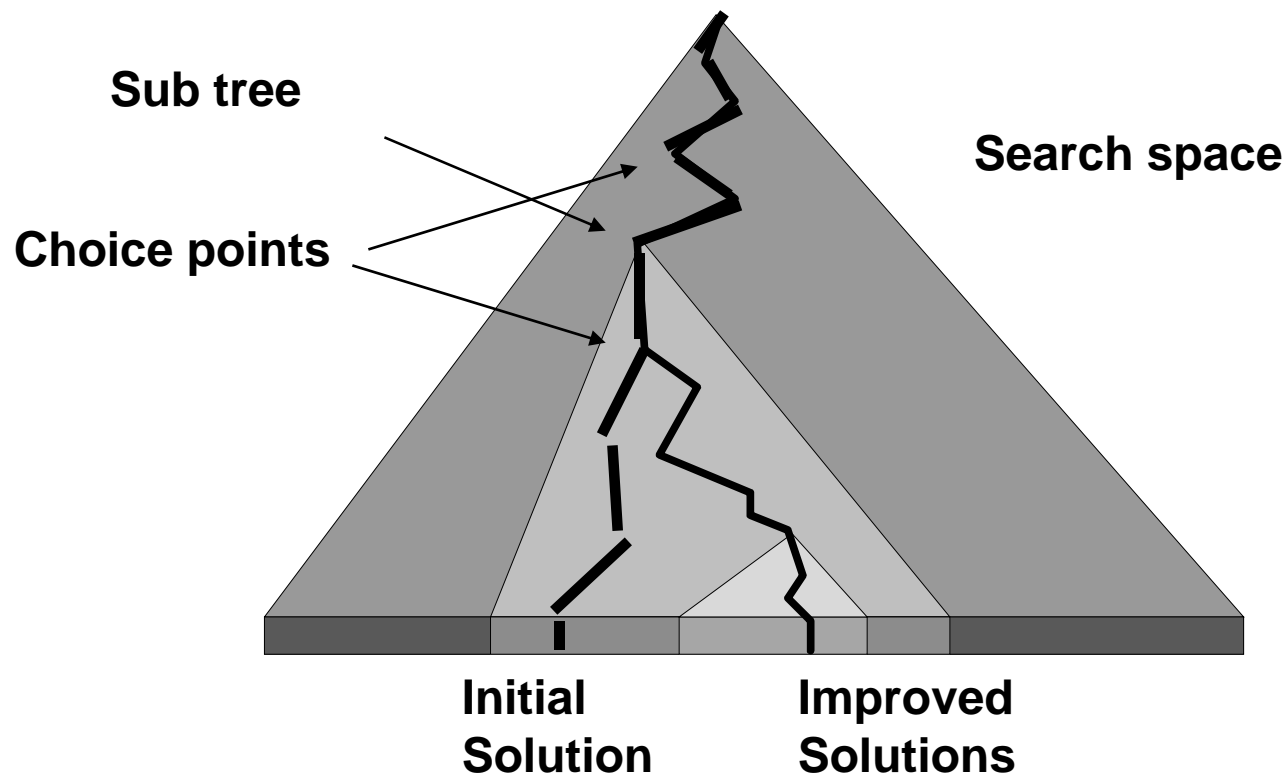
```
delete(X,[X|R],R).
delete(X,[Y|R],[Y|S]):-
    delete(X,R,S).
```
- **Support by primitives**
  - delete/5
  - domain\_info

# Redundant constraints

- **Add constraint to improve reasoning**
  - Consequence of incomplete solver
  - Inverse of LP techniques
- **Add constraints to restrict admissible solutions**
  - Strengthen constraints
  - Find only one solution, not all
  - Relaxation techniques

# Optimization

- Search trees



# Importance of lower bounds

- **Cut parts of search tree**
  - Complete enumeration impossible
- **Modelling must provide good estimates of cost**
  - The better the cost estimate, the earlier we can prune the search tree
- **Depends on cost function and constraints**
  - The pruning of the constraints may help to find good cost estimates
- **Avoid additive costs**
  - No direct influence between local decision and global cost

# Partial optimization

- **Not always necessary to explore complete search tree**
  - Conflicting cost functions
  - Imprecise data
- **Allow limited optimization around good heuristic solution**
  - Consider only one or two alternatives at each choice point
  - Choose the alternatives carefully
- **Consider decomposition to optimize sub problems**

# Methodology summary

- **Modelling of scheduling problems with constraints**
- **Heavy use of global constraints**
  - cumulative
  - diffn
  - cycle
- **Mixing of constraints**
  - toolbox
  - incremental problem Modelling
- **Role of strategies**
  - heuristics
  - optimisation

## **Part III: Applications**

- **Benchmarks and their limits**
- **ATLAS**
- **Other examples**

# Benchmark problems

- **Bridge**
  - Project planning with resource constraints
- **Muth/Thompson 10x10**
  - Difficult job-shop
- **Patterson**
  - 110 cumulative problems
- **Alvarez**
  - Difficult cumulative problems

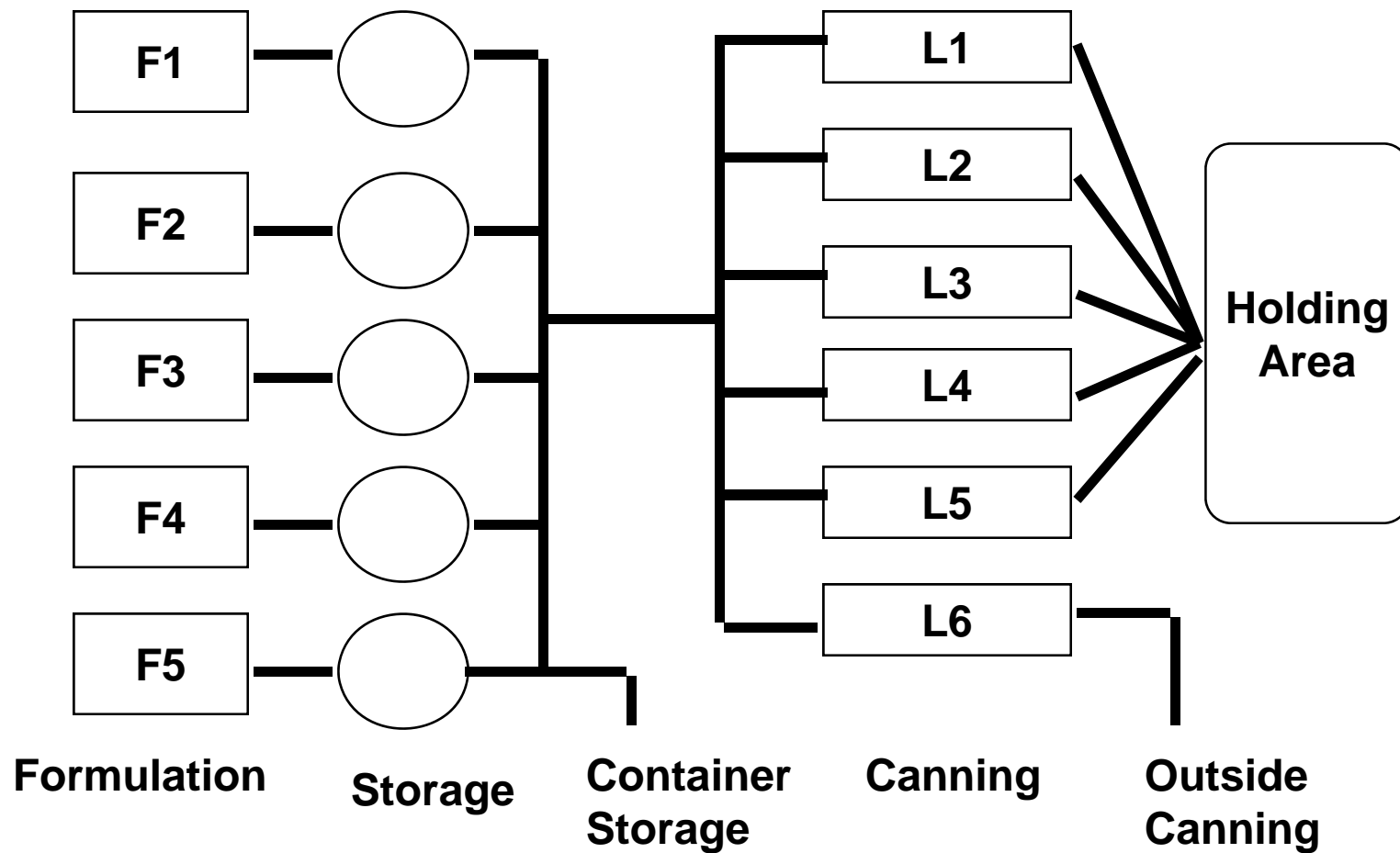
# Limits of benchmarks

- **Pure problems**
  - Non incremental techniques
  - Few types of constraints
- **Number of problem instances**
  - Over-specializing methods
  - Heuristics too particular
- **Performance measures**
  - Number of backtrack points
  - Run time
- **Problem sizes**
  - Complexity of heuristics  $> O(n^2)$

# ATLAS

- **Scheduling system in chemical factory**
  - herbicide manufacturing and packaging
  - integrated inventory control system
- **Live system**
  - first phase operational since July 93
  - second phase operational since March 94
- **Developed jointly by**
  - Beyers & Partners
  - COSYTEC
- **Multi-user PC/UNIX system**
  - multiple schedulers
  - co-operative behavior

# The process



# Batch part

- **Herbicide formulation**
- **Main constraints**
  - **Different production times for resources**
    - » **typical 4-8 hours**
  - **Batch sizes between min and max limits**
    - » **e.g. 20000 to 40000 liters**
    - » **no smaller batches allowed**
  - **Product dependent setup**
    - » **very long compared to production time**
  - **Limited choice of resources**
  - **Very limited storage capacity**
    - » **typical 1 batch**
  - **Exceptional use of auxiliary storage**

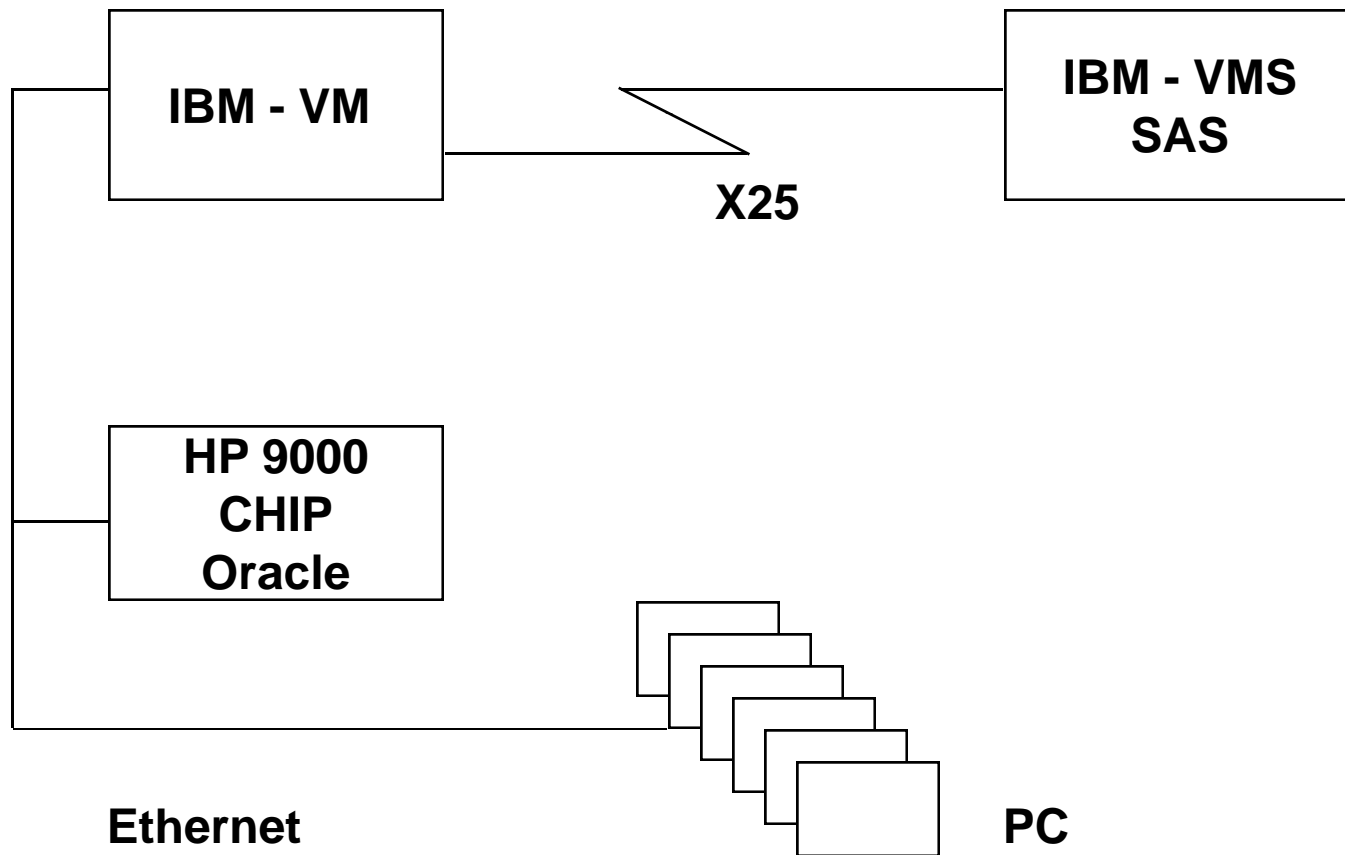
# Packaging part

- **Filling of product into containers of different sizes**
  - 0.5 l bottles to 200 l drums; granulate in sacks
- **Packaging lines vary in capability**
  - Speed
  - Material handled
  - Setup
  - Manpower required
- **Outside packaging scheduled manually**
  - But taken into account for stocks and materials
  - Operation can not be automated

# The system

- **Connection to other systems**
  - Data download/upload from/to local and remote mainframes
  - End-user connection via PCs
- **Data base**
  - Local on UNIX machine
  - ORACLE SQL standard
  - Forms/reports facilities
- **Multi-user tool**
  - Multiple schedulers for parts of production
  - Inventory control
  - Production
  - Multiple scenarios; one schedule

# System integration



# Graphical interface

- **Different views into data**
  - Graphical/textual display of information
- **Direct manipulation**
  - Moving tasks in time and on resources
- **Checks on constraints**
  - Control on manual modifications
  - Allows constraint violations
- **Manual override**
  - User can modify/change schedule at any time

# Views into data

- **Different views of data**
  - **Gantt**
  - **Manpower**
  - **Orders**
  - **Finished Products**
  - **Components**
  - **Raw Material**

# Constraint solver

- **Constraints handled**
  - Machine choice
  - Sequent/ machine dependent setup
  - Manpower limits
  - Component consumer constraints
  - Finished product producer constraint
  - Producer/consumer constraints for intermediate products
  - Production planning for batch part
    - » Number of batches for intermediate products
    - » Optimum batch sizing
  - Limited stock levels

# Machine choice

- **Tasks can be run on different lines**
  - Different production rates
  - Different manpower requirements
  - Different setups
- **Heuristic choice of preference to fastest line**
- **Offset by need to keep lines busy**

# Setup

- **Setup depends on characteristics of products**
  - Bottles
  - Caps
  - Formulated product
- **Setup varies very much**
  - No relaxation possible
- **Choice influences overall throughput**
  - Conflict with due dates
- **Uses strategy to minimize setup**
  - close to (theoretical) optimum

# Components

- **Component availability required for production**
  - Selected major components
- **Frequent changes in BOM**
  - BOM local to task
- **Constrains schedule up to lead time of component**
  - Typical up to three weeks
- **Delivery of material known up to lead time**
  - Delivery
  - Transfers
  - Material orders

## **Components (2)**

- **Tasks consume stocks on certain time points**
  - Not all material must be available at start date
  - Daily / per shift consumption possible
  - Large increase of constraint size
- **Consumer constraint on consumable resource**
  - Simple forms can be replaced by inequalities

# Finished products

- **Known delivery dates of finished products**
  - Orders of fixed quantity
- **Stock levels at beginning of schedule**
  - Data reconciliation problem
- **Finished product available before end of tasks**
  - Handled on a per day /shift basis
  - Increases problem size
  - Important to minimize stock levels
- **Producer constraint with cumulative constraint possible**

# Formulated product

- **Producer consumer constraint between**
  - Batch formulation (producer)
  - Canning line tasks (consumer)
  - Fixed parts: stocks, transfers, material orders etc
- **Limited overlap possible**
  - batches are available at end of formulation
  - canning tasks do not need all product at start date
- **Batch size dependent on actual use**

# Manpower

- **Manpower limits given from database**
  - Presented on per shift basis
- **Manpower requirements vary with placement of tasks**
  - Depends on canning line chosen
    - » Automated lines
    - » Manual lines
- **Additional manpower limits on number of lines in operation**
  - Supervisors/technicians

# Production planning

- **Tasks for canning lines given from database**
  - Related to orders/sales forecasts
  - Tasks can change in time (size/splitting)
- **Tasks for formulation generated by system**
  - Number of batches determined by system
  - Scheduled in main scheduler
  - Separate phase to tune batch sizes

# Multiple scenarios

- **Possibility to store/recall/delete multiple scenarios**
- **Evaluation via quality indicators**
- **Exchange of proposed solutions between operators**
- **What-if evaluation**
- **Comparison manual/automated scheduler**

# Quality indicators

- **Orders too late**
- **Tasks too late**
- **Tasks too early**
- **Stock level**
- **Negative stocks**
- **Production rate**
- **Productivity**
- **Setup periods/duration/manpower**
- **Cleaning time/ resources**

# ATLAS - Summary

- **End-user system**
  - Direct manipulation
  - Constraints not visible to user
  - Changes/strategies expressed via interface
- **Complex scheduling problem**
  - Real-life, not pure problem
  - Interaction/conflicts between constraints
  - Incremental expression with constraints
- **Significant development effort**
  - Integration
  - Data base reporting
  - Graphical interface
  - Problem solver

# Other applications

- **HIT (ICL)**
  - Scheduling/assignment for container harbor
- **PPO (Sligos/COSYTEC)**
  - Planning/scheduling of mail sorting
- **PLANE (Dassault Aviation)**
  - Production line scheduling
- **Workshop scheduler (Dassault Aviation)**
  - Layout and workcell scheduling
- **Saveplan (Sligos)**
  - MRP scheduling package
- **PROTOS (BIM, Hoechst, IBM, Sandoz)**
  - Scheduling project for chemical industry

## **Applications (2)**

- **CPLAN (COSYTEC)**
  - Toolbox for resource restricted project planning
- **Forward CHIP (Technip/COSYTEC)**
  - Scheduling planning/simulation for refineries
- **APACHE (COSYTEC)**
  - Assignment for airport stand allocation
- **PILOT (SAS Data/COSYTEC)**
  - Planning/assignment of air crew rotation
- **EVA (EDF/GIST)**
  - Transport schedule for nuclear fuel
- **TACT (COSYTEC)**
  - Transport schedule in food industry

## **Part IV: to probe deeper**

- **Comparison**
- **Pitfalls**
- **Part of the development process**
- **References**

# Comparison

- **Alternative Methods**
  - OR techniques
  - Heuristic approach
  - Iterative Improvement
- **Evaluation Criteria**
  - Effort
  - Expertise
  - Efficiency

# When to use constraints

- **Hard Problems**
  - Admissible solutions are not obvious
  - Many constraints are hard
  - Strong interaction of different constraints
- **Local Costs**
  - Costs are expressed locally, not globally
  - Lower bound estimated by constraints
- **Strategies are important**
  - User knowledge can be included
  - No blind search for an optimum

# Pitfalls

- **Modeling Everything**
  - Constraints must constrain
  - Relaxation of problem often easier
- **Soft/Hard Constraints**
  - Soft constraints do not help
  - Hard limits behind soft ones
- **Optimization with imprecise data**
  - Influence on problem size
  - Theoretical vs practical optimum
- **KISS principle applies**
  - Keep it simple, stupid!

# Relative importance of solver

- **Percentage of design**
  - Solver must be taken into account during design
  - Realistic design goals for automated solvers
- **Percentage of development**
  - Often small compared to integration work
  - Success not guaranteed
    - » Prototyping
    - » Experiments early in development cycle
  - Relatively high risk compared to other development
- **Importance to overall acceptance**
  - Solver alone is not enough
  - Without solver the rest is not useful

# References

- **References for the areas**
  - Constraint methods
  - Scheduling techniques
  - Applications of CLP
  - CLP systems
  - Overview articles
- **Other constraint tutorials**
- **Presentations at PAP 95**