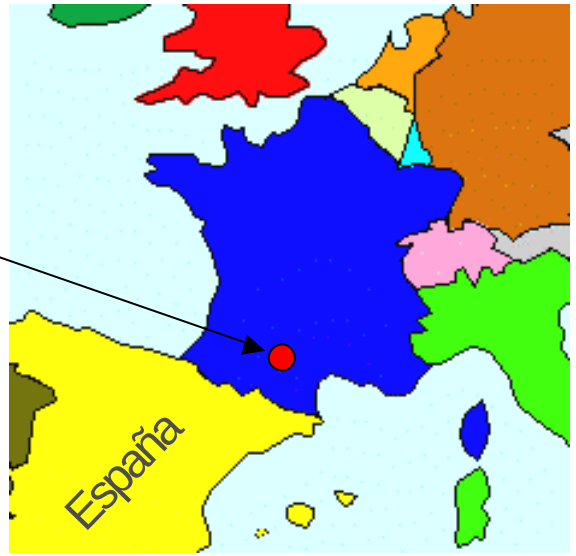
Pierre Lopez
LAAS, Toulouse
France
www.laas.fr



Constraint Programming
&
Scheduling

Objectives/Outline of the talk

- **overview of basis concepts related to constraint reasoning for decision problems (propagation/satisfaction/programming)**
- **relationships and interest for scheduling problems**

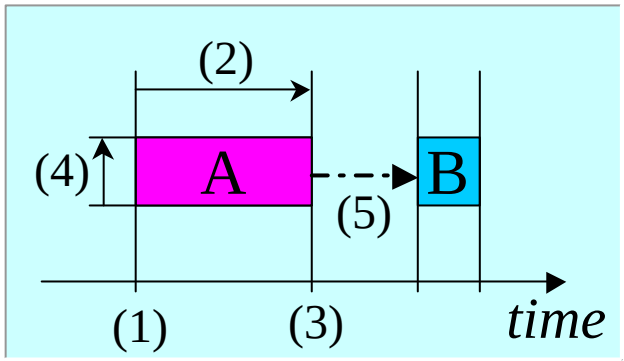
- **Combinatorial Optimization & Scheduling Problems**
  Definition / Resolution
- **Constraint Satisfaction Problems**
  Definition / Algorithms
- **Scheduling problems**
  - Temporal constraint propagation
  - Resource constraint propagation
  - Mixed scheduling-allocation problems
- **Propagation and searching**
- **Benefits/Shortcomings of Constraint Programming**

Framework: Combinatorial Optimization Problems

- **Combinatorial Optimization Problem = COP**
  - criterion (criteria) to optimize
  - under numerous, various constraints
- **Industrial examples**
  - Project management
  - Routing problems
  - Time tabling
  - Work organization in manufacturing
  - Network optimization (informatics, telecoms, spatial missions…)
  - Configuration, design problems
- **High complexity (*NP-hard*)**

Example of a COP: Scheduling Problem

- **Given:**
  - a set of *n* tasks, a set of *m* resources



Determine:

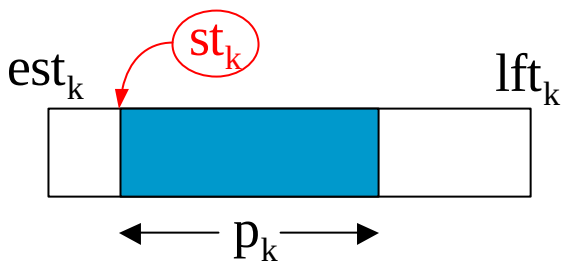
where the tasks are located in time ?

- start-times (1) and
- durations (2) or finish-times (3)

how resources are allocated to the tasks ?

- identity and intensity (4)
- sequencing (5) (e.g., A $\prec$ B)

- **Notations**



Solving a COP: Exact methods

- **Mathematical Programming**
  - dynamic programming
  - linear programming, MIP
- **Tree-search procedures**

  Search strategies
  - Depth-First Branch & Bound
  - Best-First Branch & Bound
  - Limited Discrepancy Search (branches with increasing discrepancies from an heuristic-driven solution)
- **Specific results**

  proper to each type of problem

  e.g., Johnson's algorithm for the $F2 \mid prmu \mid C_{max}$

Solving a COP: Approximation methods

- **Heuristics**
  - suited to the structure of the problem at hand, so as to find a solution of acceptable quality in a computation time as small as possible
  - e.g., dispatching rules

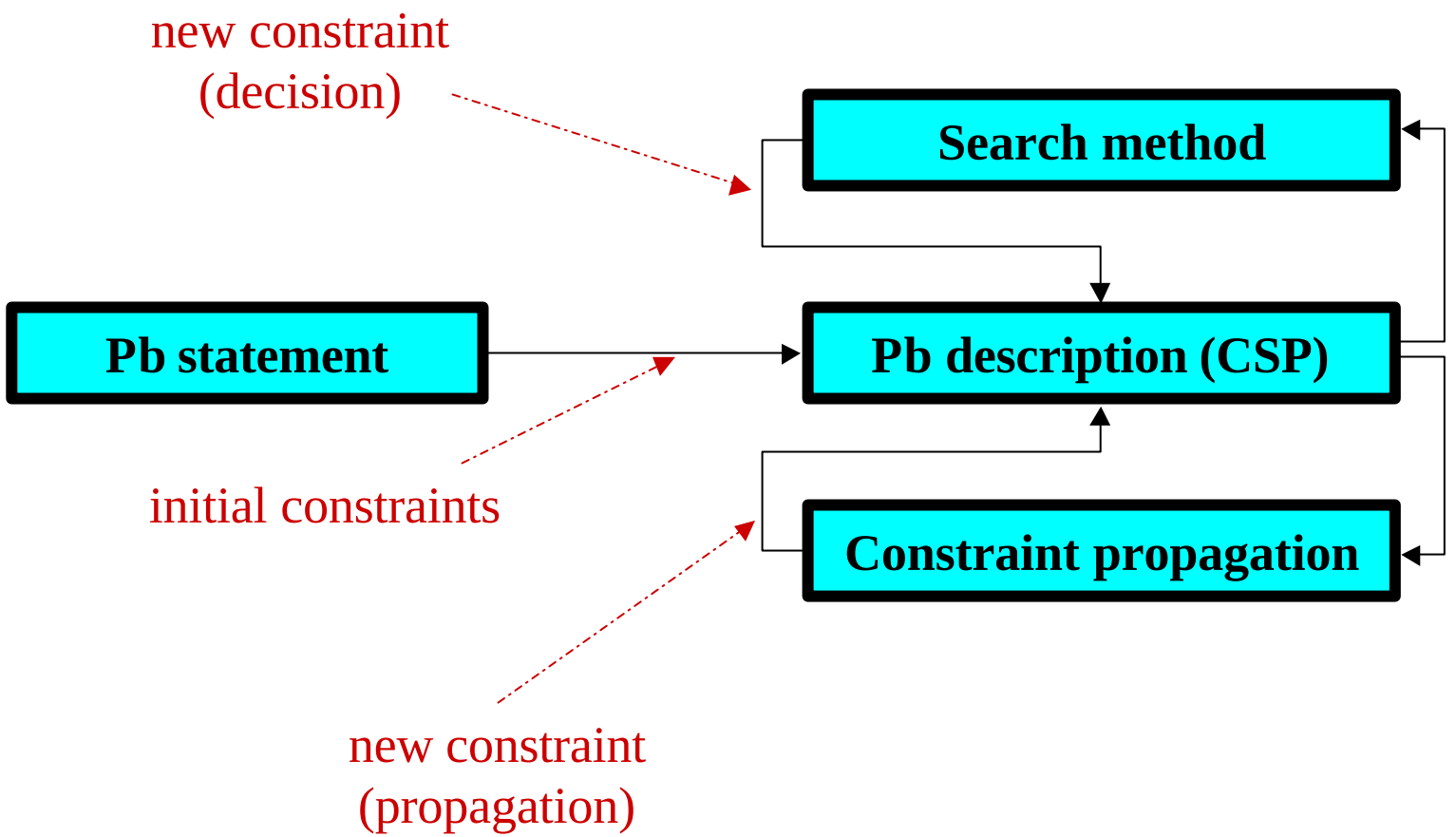
- **Metaheuristics**
  - general framework of resolution
  - unique solution
    - constructive methods: greedy/myopic algorithms
    - local search: Tabu search, simulated annealing
  - population of solutions
    - evolutionary methods: genetic algorithms, ant colony optimization

And the constraint programming...

Constraint programming paradigm (1)

- **Clear separation between**
  - problem statement in terms of variables and constraints to be fulfilled
  - logic deduction process over the constraints
  - problem solving procedures (instantiation of variables)

Constraint programming paradigm (2)



From Baptiste et al.

Constraint programming paradigm (3)

- **COP = < {variables}, {constraints}, [criteria] >**
  - a **constraint**
    - is a logic expression linking decision variables, each of them taking its values in a domain

      examples: $x < y$; $A \neq B$; $Z \in \{white, black\}$; $\alpha + \beta + \gamma = 180$
    - involves a restriction on the values the variables can take simultaneously
  - a **solution** is a set of instantiations (a **tuple** of values)…
    - … which satisfies all the constraints… (*coherent* or *consistent*)
    - … which reaches the extremum of possible criteria (*optimal*)

CSP

- **A COP can be defined as an instance of a Constraint Satisfaction Problem**
- **CSP = (X,D,C)**
  - X set of variables
  - D finite domains of values of the variables
  - C constraints linking variables of a given *arity*
- **The criterion is integrated as a constrained variable**

criterion_value < value_of_the_best_solution

(minimization problem)

  - restart the resolution
  - if an *inconsistency* arises, the latest found solution is optimal

CSP, COP: Example in scheduling

- **decision variables (X)**
  - start times st_k
- **domains (D)**
  - defined by the limit times (release dates, due dates) $D_k = [\ est_k\ ,\ lft_k - p_k\ ]$
- **constraints (C)**
  - routes job j: $j_1 \rightarrow j_2,\ (j_1,j_2) \in O_j$ and j_2 successor of j_1
  - time lags
  - resources with limited capacity (disjunctive/cumulative)

    machine i: $a \leftrightarrow b,\ (a,b) \in O^i$
- **criteria**
  - minimization of the makespan
  - minimization of the weighted number of late jobs
  - minimization of the maximum tardiness…

Questions associated to a CSP

- **Is a CSP consistent ?**
  - Þ **satisfiability … NP-complete**

- **What is a solution – feasible or optimal – of a CSP ?**
  - Þ **satisfaction … NP-hard in the general case**
    - ➔ resolution: Depth-First Branch & Bound
      - *test and generate* (backtrack algorithm): check a constraint violation after instantiation of all the variables inherent to the constraint
      - Main shortcoming: exponential complexity $O(md^n)$

        CSP of n variables, domains of cardinal d, m constraints:

        d^n terminal states, evaluation of md^n constraints

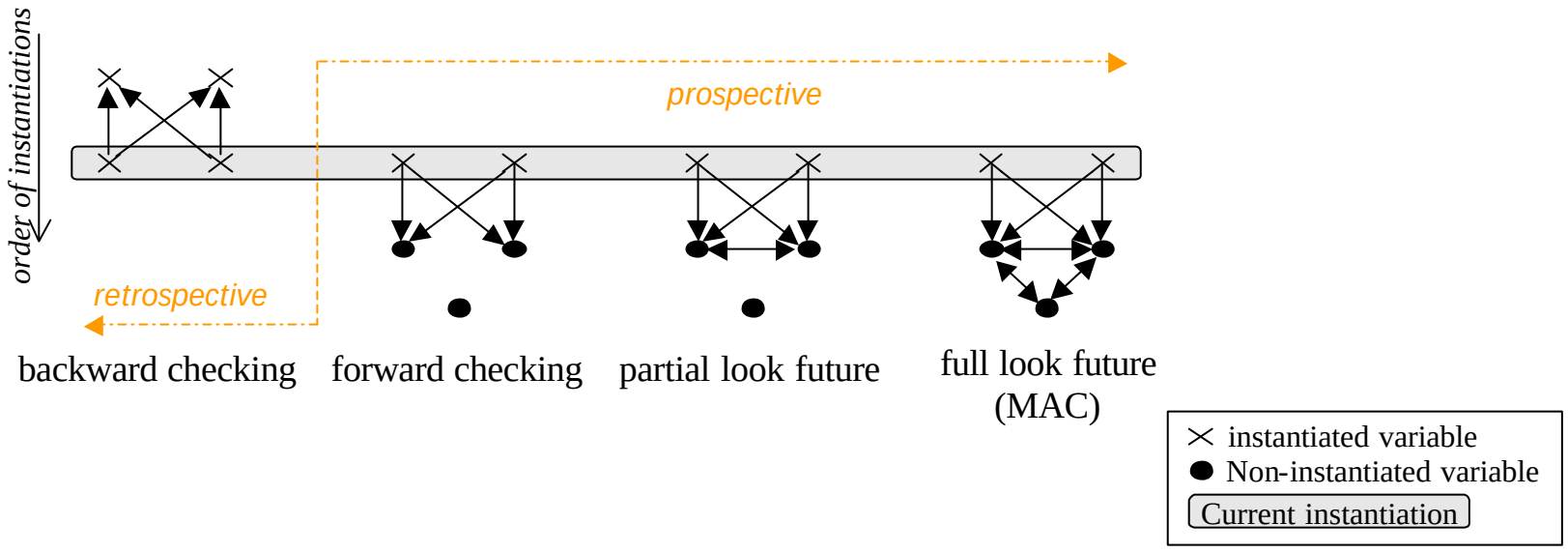
Improving Backtrack algorithm

- **definition of ordering heuristics: order on the instantiation of variables (*first-fail*, *most-constrained*, *smallest*...), values, constraints…**
- **reduction of the search space (using properties of local consistency or filtering algorithms or constraint propagation techniques) before – or during – the search for a solution**
- **modification of the algorithm in order to detect, at a lower cost and as soon as possible, the inconsistency of the current instantiation**

Retrospective vs. prospective schemes in Backtrack

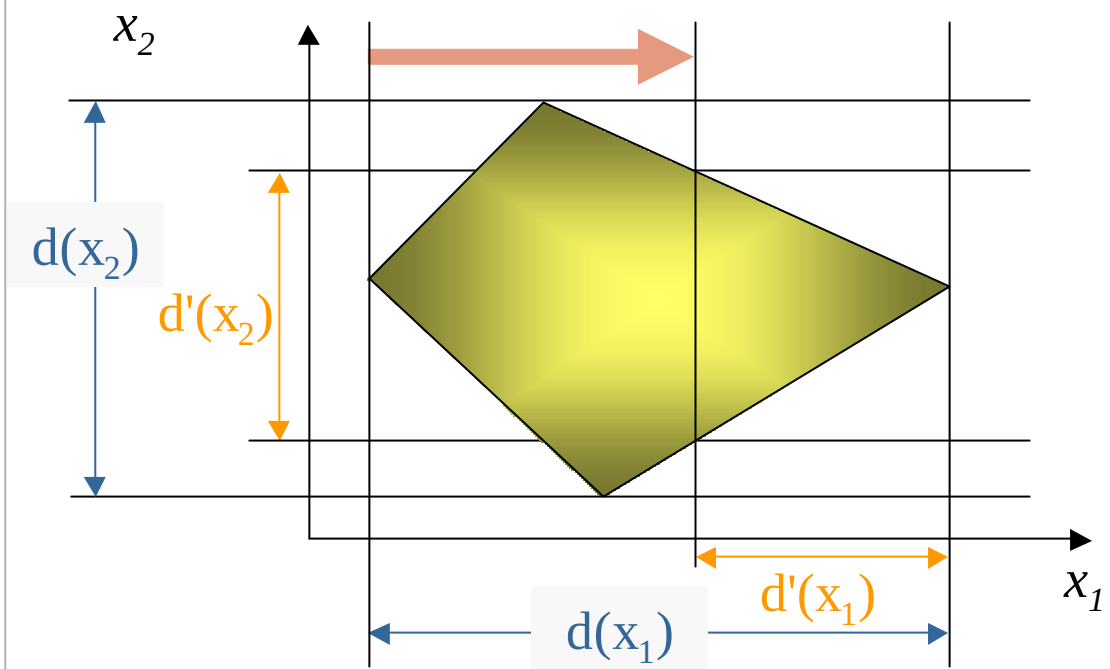
– **retrospective scheme: intelligent backtracking, *e.g.*:**
  - *backjumping* (go back to the closest variable connected with a constraint to the variable in conflict)
  - *nogoods recording* (recording of a set of instantiations that cannot lead to any solution)

\+ **prospective scheme (*look-ahead schemes*): uses results of filtering exploiting them during the search, *e.g.* FC or MAC**

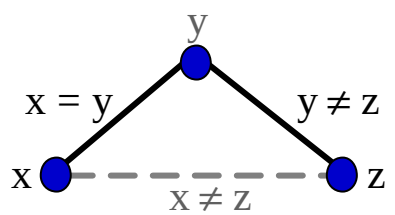


Constraint propagation

- **Set of techniques allowing**
  - the consistency checking of a solution
  - a consistency enforcing by *filtering* the values of the variables which do not belong to any solution
    - domain reduction



- deduction of new constraints



  - detection of a global inconsistency

Example of propagation (1)

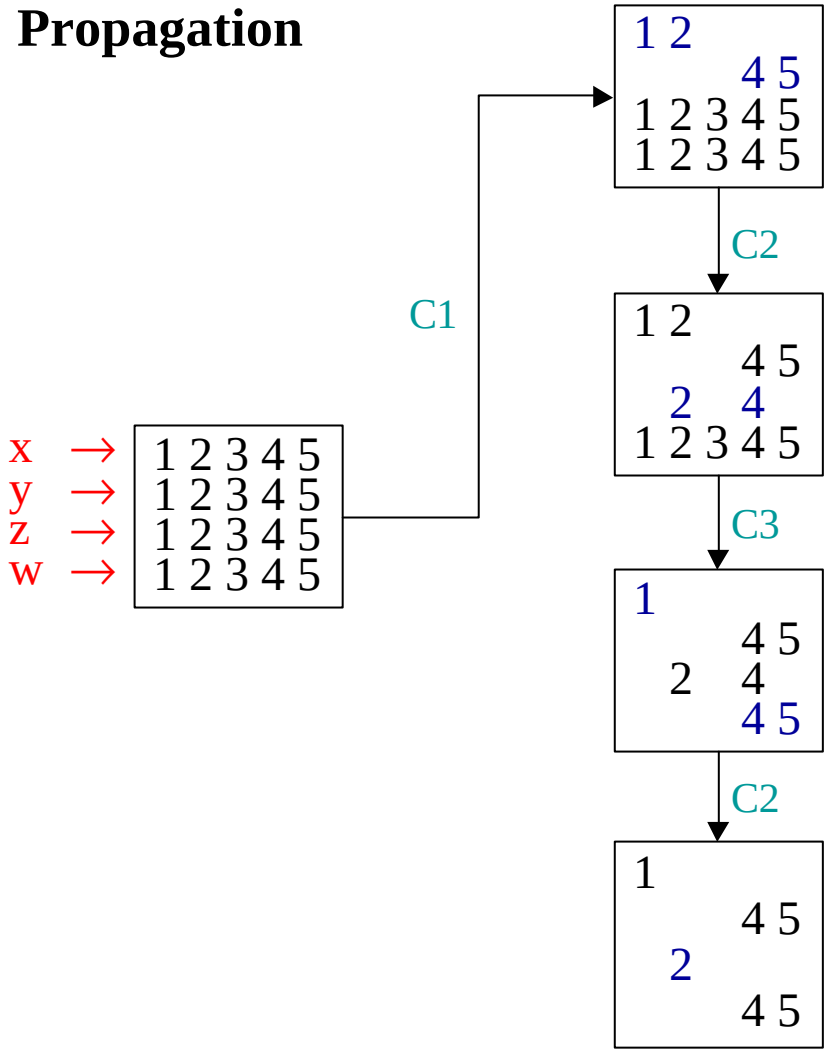
- **Initial CSP**

$D_x = D_y = D_z = D_w = \{1,2,3,4,5\}$

C1: $x + 3 \leq y$
C2: $z = 2 * x$
C3: $w = x * y$

a constraint is non-directional

- **Propagation**



Arc-consistency

- **Context: *binary* CSP**
  P=(X,D,C) with X={x}, D={D_x}, C={$C_{xx'}$}

- **(x,y) arc-consistent iff for each value of D_x there exists a compatible value in D_y**

AC-3 algorithm

```
L ← {(i,j) | ∃C_ij ∈ C}
while L ≠ ∅
    remove pair (k,m) from L
    if Revise(k,m)
        L ← L ∪ {(i,k) | ∃ C_ik ∈ C, i ≠ k, i ≠ m}
```

```
Revise(i,j): Boolean
    modif ← false
    for v ∈ D_i
        if ∄ v' ∈ D_j s.t. {i ← v, j ← v'} consistent
            D_i ← D_i – {v}
            modif ← true
    return modif
```

$O(md^3)$
[$O(md^2)$ for AC-4]

Time and resources...

Example of propagation in scheduling

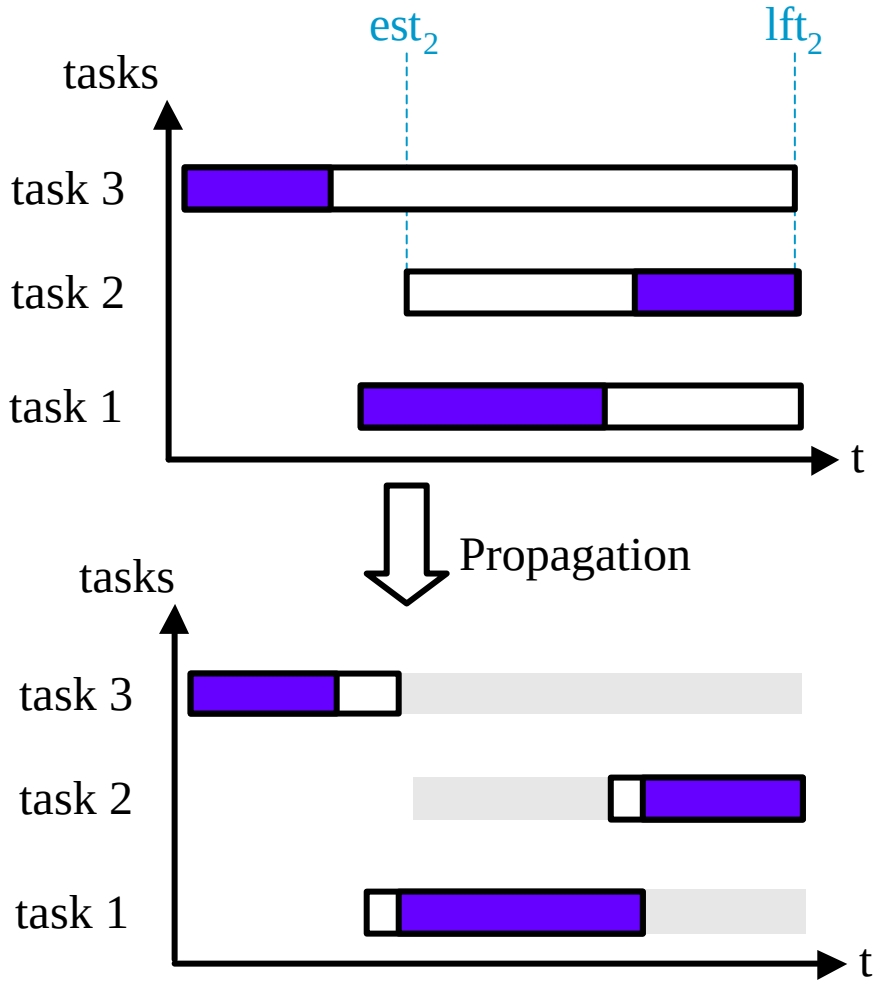
- **deduction rule dedicated to disjunctive pairs**

Example:

$lft_j - est_i < p_i + p_j \Rightarrow$ i not before j

disjunctive problem $\Rightarrow$ j before i

$\Rightarrow$ adjustments of est_i and lft_j

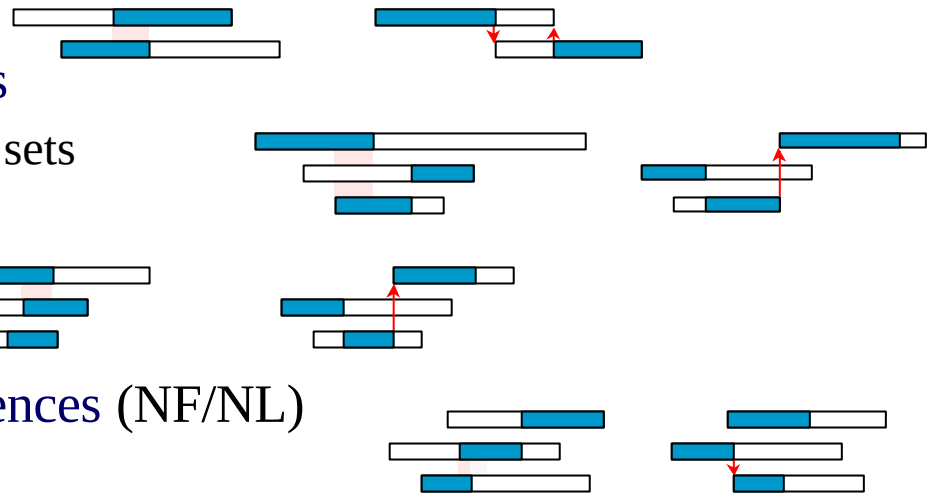


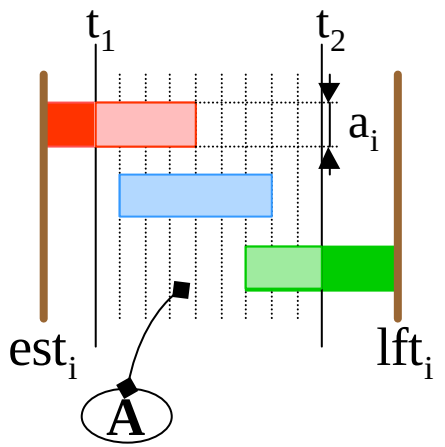
Propagation of time constraints

- **Consistency enforcing of a temporal CSP (TCSP)**
  - variables = times
  - constraints = distances (under the form of intervals) between 2 times (*binary* constraints)
- **Simple temporal problems (STP)**
  - constraint $\leftrightarrow$ 1 *simple* interval
  - graph algorithms (Bellman, Ford, Floyd-Warshall…)
  - polynomial and complete
- **General TCSPs**
  - constraint $\leftrightarrow$ disjunction of intervals
  - *path consistency*: PC2, ULT, LPC…
  - polynomial but incomplete

Propagation of resource constraints (1)

- **Local operations**
  - *disjunctive*
    - disjunctive pairs
    - conjunctive precedences
      - ascending/descending sets
      - EFF/LSL
    - non-conjunctive precedences (NF/NL)
    - energetic reasoning
      - integration of time and resource constraints (concept of "work") considering energetic balances
        - between tasks and resources
        - over time intervals
      - relative or absolute location of tasks





Propagation of resource constraints (2)

- **Global operations**
  - decision refutation
    by simple propagation on the global problem
  - shaving
  - singleton arc-consistency

Skeleton of a SAC algorithm

```
AC(X, D, C)
for x ∈ X and for v ∈ D_x
    x ← v
    AC(X, D, C ∪ {x ← v})
    if inconsistency
        C ← C ∪ {Ø (x ¬ v)}
        AC(X, D, C)
```

$st_i \in [est_i, \alpha]$, $\alpha < lft_i - p_i$

$st_i \in [\alpha+1, lft_i - p_i]$ (*i.e.*, $est_i \leftarrow \alpha+1$)

Some results on global operations

L. Péridy (1996)

Toàn Phan Huy (2000)

P. Torres & P. Lopez (2000)

- proof of optimality and optimal solution of FT10 in 1 node
- improving of best known lower bounds on very large instances (SWV, YN)
- reduction of the number of nodes in a factor up to 10000…
- reduction of CPU time in a factor up to 4
- optimal solving of open shop instances (Hurink, Taillard)

- **But…**
  - can also induce a very important increasing of CPU time

Mixed Scheduling-Allocation problems

- **The basis assumption**
  - durations and intensities may depend on the resource allocated
- **Principles of cross-propagation rules**
  - each time a task *i* is assigned to a resource *k*, try to tighten the time domains of tasks already assigned to *k*
  - each time the time domain of task *i* is reduced, remove any inconsistent allocation alternative
    - either because it is not time-feasible
      - the slack left by tasks already assigned to *k* is too small
    - or it is is not energy-feasible
      - the energy left by tasks already assigned to *k* is too small

Propagation and searching (1)

- **Chronological backtracking algorithm with a variable and value ordering heuristics**

```
stack of contexts = empty
select a good candidate variable to instantiate (e.g., the most constrained)
    select a good value (e.g., the less constraining)
        push the context on the stack                    (record)
        propagate time and resource constraints
        if the problem is no longer consistent
        then if the stack is not empty
                then pop the previous context            (restore)
until all variables have a value or the stack is empty   (inconsistency)
```

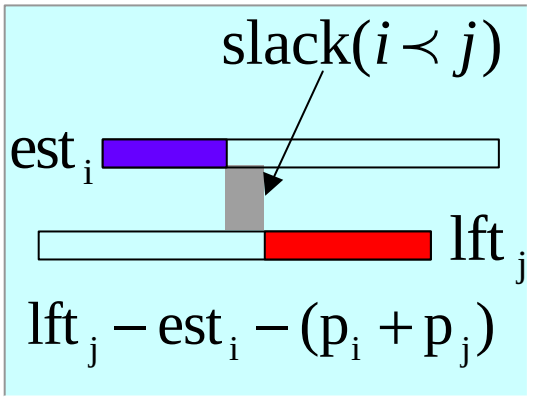
Propagation and searching (2)

- **Some branching schemes for minimizing makespan**

1/ Select the most constrained resource not yet entirely scheduled

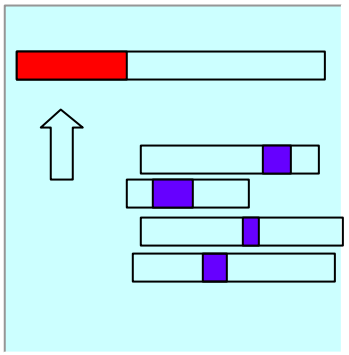
2/ Min-slack/Max-slack

- select an unordered pair with the smallest slack ($i \prec j$ or $j \prec i$)
- create two problems for each relation



2'/ Binary search

- select a task (*e.g.*, with the most outlying domain)
- create two problems for each half of the domain



3/ Propagate time and resource constraints

Repeat 1, 2-2', 3, until all resources are scheduled

Constraint Programming Systems

- **Pioneer: Constraint Logic Programming**

  1982-90: Prolog II, CLP(*R*), CHIP, Prolog III

- **[Fernandez & Hill, *JAIR*, 2000]: distinction of 2 types of systems:**
  - glass box (propagation scheme can be specified by the user)
    - clp(FD), SICStus, IF/Prolog, CHR
  - black box (control of propagation is beyond the user)
    - Oz, ECL^{i}PS^{e}, Ilog SOLVER, B-Prolog

- **CLAIRE, CHOCO**

- **Others… *see*:**
  - **Newsgroup:** **comp.constraints (FAQ)**
  - **Roman Barták:** **http://kti.ms.mff.cuni.cz/~bartak/constraints/**

Why the constraint programming ?

Benefits of constraint programming (1)

- **declarative nature of constraints**
- **representation close to original problem**
  - variables = problem entities
  - constraints: not necessarily translated in linear inequalities
- **importance of satisfiability problems**
- **modularity (distinction analysis/resolution)**
  - ➔ **flexibility of devised systems**
- **adaptation of algorithms to the types of constraints**
- **efficiency for some problems (scheduling, strongly symmetrical programs…)**

Benefits of constraint programming (2)

- **decision-aid**
  - limitation of possible actions, explicitation of available freedom degrees, characterization of sets of feasible solutions
  - reactive behavior (based on dynamic CSPs) suited to the model evolution
    - practical importance of side constraints not always integrated in the initial formulation because not easily formalizable, too much context-dependent…
    - adding a constraint → put in question the solutions but conservation of deductions (constraints additivity)
    - removing a constraint → put in question the deductions obtained from the propagation but conservation of solutions
    - *incremental* nature when accounting for new constraints (without considering neither existing constraints nor the search procedure)

Shortcomings of constraint programming

- **modeling**
  - ➔ modeler (OPL)
- **performances, efficiency of basis techniques to solve large scale problems**
  - ➔ determination of lower bounds and design of suited propagation mechanisms
- **completeness of deductions, genericity of reasonings**
  - ➔ structures to implement propagation rules (*lattice of task intervals*)
- **sacrifice of general characteristics of a programming language to favor sophisticated and dedicated primitives**
  - ➔ libraries

constraint programming, mathematical programming, metaheuristics… ?

the future is the hybridization and the cooperation
instead of the opposition between methods

Some related books...

- **Constraint-based Scheduling**
  P. Baptiste, C. Le Pape, & W. Nuijten
  Kluwer, 2001

- **Logic-based Methods for Optimization**
  J. Hooker
  Wiley, 2000

- **The OPL Optimization Programming Language**
  P. Van Hentenryck
  MIT Press, 1999

- **Intelligence artificielle et informatique théorique**
  J.-M. Alliot & T. Schiex
  Cépaduès, 1994
  (in French…)

- **Foundations of Constraint Satisfaction**
  E. Tsang
  Academic Press, 1993

Summary...

Constraint programming is based on:

- **Constraint Satisfaction Problems**
  - local consistency rules for temporal constraints
  - heuristic search strategies
- **Operations Research**
  - tree search solving procedures
  - lower and upper bounds for optimal solutions
  - propagation rules for resource constraints

Example of propagation (2)

- **Initial CSP**

  $D_x = D_y = D_z = \{1,2,3,\ldots,10\}$

  C1: $x + 3 \leq y$

  C2: $z = 2*x$

  C3: $y + 2 = z$

- **Propagation**

  C1: $D_x = \{1,\ldots,7\}$, $D_y = \{4,\ldots,10\}$

  C2: $D_y = \{4,\ldots,8\}$, $D_z = \{6,\ldots,10\}$

  C3: $D_z = \{6,8,10\}$, $D_x = \{3,4,5\}$

  C1: $D_y = \{6,7,8\}$

  C2: $D_z = \{8,10\}$

  C3: $D_x = \{4,5\}$

  C1: $D_y = \{7,8\}$

  C2: $z = 10$

  C3: $x = 5$

  C1: $y = 8$

Retrospective vs. prospective schemes in Backtrack