

INTEGRATION OF MATHEMATICAL PROGRAMMING AND CONSTRAINT PROGRAMMING FOR COMBINATORIAL OPTIMIZATION PROBLEMS

Michela Milano

Università di Bologna - ITALY

`mmilano@deis.unibo.it`

`http://www-lia.deis.unibo.it/Staff/MichelaMilano/`

Tutorial CP'2000: Singapore, September 2000

OVERVIEW

- Preliminaries:
 - Combinatorial Optimization Problems
 - CP and OR techniques
- Unifying frameworks
- Integration:
 - *problem modelling*
 - *problem solving:*
 - *Optimization*
 - *branch & bound*
 - *branch & cut*
 - *column generation*
 - Search

OVERVIEW

- Preliminaries:
 - Combinatorial Optimization Problems
 - CP and OR techniques
- Unifying frameworks
- Integration:
 - *problem modelling*
 - *problem solving*:
 - *Optimization*
 - *branch & bound*
 - *branch & cut*
 - *column generation*
 - Search

COMBINATORIAL OPTIMIZATION PROBLEMS

- We consider discrete NP-hard problems
- Minimizing (Maximizing) a function of many variables subject to
 - *Mathematical constraints*
 - *Symbolic constraints*
 - *Integrality restrictions on some or all variables*
- Many application areas:
 - *resource allocation, scheduling, planning, routing, sequencing, design, configuration....*

FINITE DOMAIN CP

- Problem modelling
 - Variables range on a finite domain of objects of arbitrary type
 - Constraints among variables
 - mathematical constraints
 - symbolic constraints
- Problem solving
 - Propagation algorithms embedded in constraints
 - Arc consistency as standard propagation
 - More sophisticated propagation for global constraints
 - Search strategies
 - Branch & Bound for optimization

CP:OPTIMIZATION

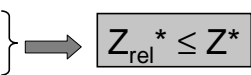
- In some applications, we are not interested in a feasible solution but in the OPTIMAL solution according to a given criterion
- ENUMERATION \Rightarrow inefficient
 - find all feasible solutions
 - chose the best one
- Constraint Programming tools in general embed a simple form of Branch & Bound
 - each time a solution is found whose cost is C^* , impose a constraint on the remaining search tree, stating that further solutions (whose cost is C) should be better than the best one found so far

$$C < C^*$$

OR: BRANCH & BOUND

- Different from CP Branch & Bound
- Two step tree search procedure:
 - solving a relaxation of the original problem
 - splitting the problem into subproblems
- Relaxation:
 - consider the original problem P at a given node
 - relax some constraints and generate P_{rel} easier than P
- Branching:
 - select a variable Var
 - impose additional constraints on Var

OR: BRANCH & BOUND

- The relaxation P_{rel} of a problem P provides a lower bound for P in minimization problems, an upper bound for P in maximization problems
- For minimization problems
 - Z_{rel}^* = optimal solution of P_{rel}
 - Z^* = optimal solution of P
$$Z_{rel}^* \leq Z^*$$
- At a given node, if the lower bound is greater than the best solution found so far, the corresponding subtree can be pruned.

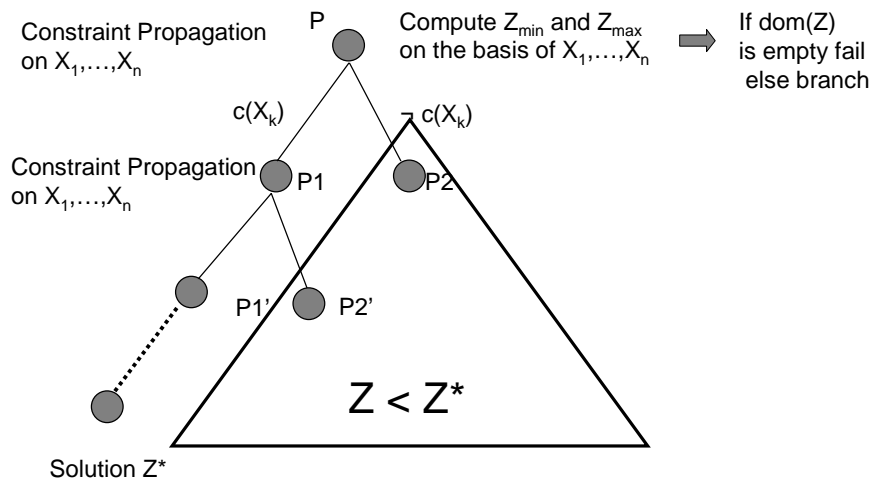
OR: BRANCH & BOUND

- Use of upper bounds in minimization problems:
 - we need a solution Z_{up}
 - solve the problem with a heuristic algorithm
- For minimization problems
 - Z_{up}^* = heuristic solution of P
 - Z^* = optimal solution of P } \rightarrow $Z^* \leq Z_{up}^*$
- At a given node, if the lower bound is greater than the upper bound, the corresponding subtree can be pruned.
- Upper bound updated during search

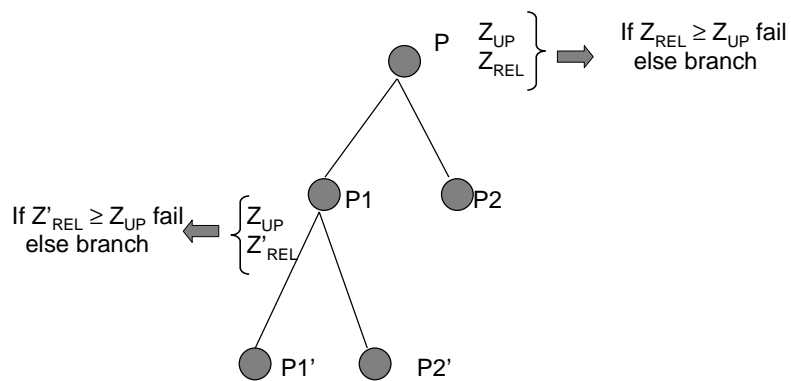
COMPARISON

- Optimization:
 - In CP each time a feasible solution is found Z^* , a constraint is added on the objective function variable $Z < Z^*$. Since Z is linked to problem variables, propagation is performed. At each node, when variables are instantiated and propagation is performed, bounds of Z are updated.
 - In MIP, at each node the LP relaxation is solved providing a lower bound on the problem. If the lower bound is worse than the current upper bound, the node is fathomed. Otherwise, a non integral variable x is selected and the branching is performed on its bounds. An initial upper bound can be in general computed.

CP BRANCH & BOUND



OR BRANCH & BOUND



INTEGER PROGRAMMING

- Standard form of Combinatorial Optimization Problem (IP)
[Nemhauser Wolsey: *Integer and Combinatorial Optimization 88*]

$$- \min z = \sum_{j=1}^n c_j x_j$$

- subject to

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad i = 1..m$$

$$x_j \geq 0 \quad j = 1..n$$

x_j integer ← May make the problem NP complete

- Inequality $y \geq b$ recasted in $y - s = b \quad s \geq 0$
- Maximization expressed by negating the objective function

LINEAR RELAXATION

- General form of Combinatorial Optimization Problem (IP)

$$- \min z = \sum_{j=1}^n c_j x_j$$

- subject to

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad i = 1..m$$

$$x_j \geq 0 \quad j = 1..n$$

x_j integer ← Removed

← Linear Relaxation

- The linear relaxation is solvable in POLYNOMIAL TIME
- The SIMPLEX ALGORITHM is the technique of choice even if it is exponential in the worst case

0-1 INTEGER PROGRAMMING

- Many Combinatorial Optimization Problem can be expressed in terms of 0-1 variables (IP)

$$- \min z = \sum_{j=1}^n c_j x_j$$

- subject to

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad i = 1..m$$

$$x_j \in [0, 1] \quad \leftarrow \text{May make the problem NP complete}$$

0-1 LINEAR PROGRAMMING

- Many Combinatorial Optimization Problem can be expressed in terms of 0-1 variables (IP)

$$- \min z = \sum_{j=1}^n c_j x_j$$

- subject to

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad i = 1..m$$

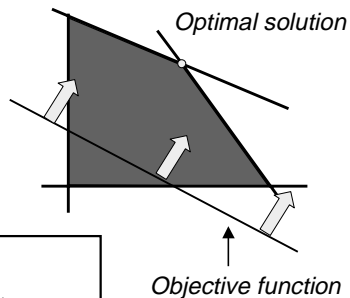
$$0 \leq x_j \leq 1 \quad \leftarrow \text{Relaxed}$$

\leftarrow Linear Relaxation

GEOMETRIC PROPERTIES OF LP

- The set of constraints defines a polytope
- The optimal solution is located on one of its vertices

$$\begin{aligned} & - \min z = \sum_{j=1}^n c_j x_j \\ & - \text{subject to} \\ & \quad \sum_{j=1}^n a_{ij} x_j = b_i \quad i = 1..m \\ & \quad x_j \geq 0 \quad j = 1..n \end{aligned}$$

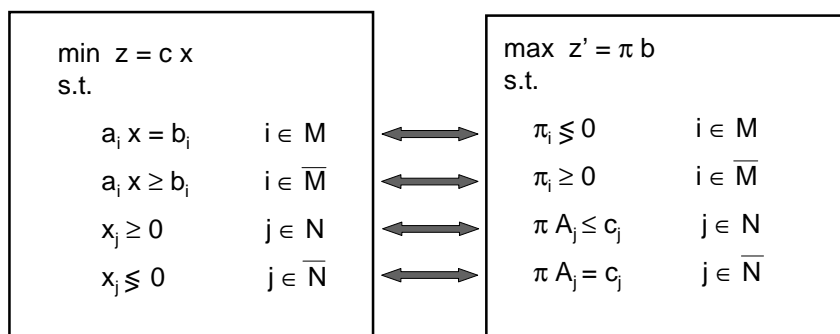


The simplex algorithm starts from one vertex and moves to an adjacent one with a better value of the objective function

PROPERTIES OF LP SOLUTION

- The optimal solution of the relaxation is an assignment of values to variables such that all linear inequalities are satisfied and the objective function is minimized
- The optimal LP solution is in general fractional: violates the integrality constraint
- Each LP (primal) has an associated dual problem where dual variables correspond to constraints and dual constraints correspond to primal variables

DUAL PROBLEM



- The optimal LP solution provides reduced costs (through dual variables) representing the cost to be paid if a given value is in the solution

EXAMPLE

- Produce and sell two kinds of products: A and B
- Both products have to be processed on two machines M1 and M2
 - Product A process lasts 12 min. on M1 and 30 min. on M2
 - Product B process lasts 24 min. on M1 and 24 min. on M2
 - The resource availability of M1 is 400 hours and that of M2 is 490 hours
 - Profit of selling product A is \$12 and for product B is \$20.
- Goal: produce at least 100 units of each product and maximize the profit

EXAMPLE: MODEL

- Decision variables **prodA** and **prodB** representing the quantity to be produced. Convert minutes to hours

$\max 12 \cdot \text{prodA} + 20 \cdot \text{prodB}$ (*Profit to be maximized*)

s.t.

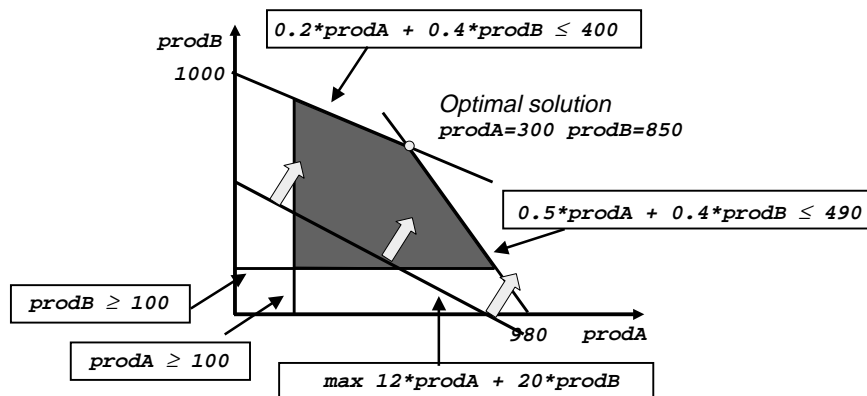
$0.2 \cdot \text{prodA} + 0.4 \cdot \text{prodB} \leq 400$ (*Availability M1*)

$0.5 \cdot \text{prodA} + 0.4 \cdot \text{prodB} \leq 490$ (*Availability M2*)

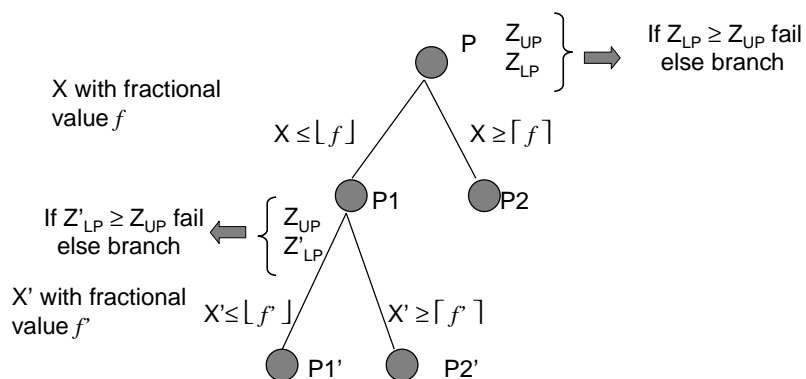
$\left. \begin{array}{l} \text{prodA} \geq 100 \\ \text{prodB} \geq 100 \end{array} \right\}$ (*minimal quantity required*)

$\text{prodA}, \text{prodB}$ integer

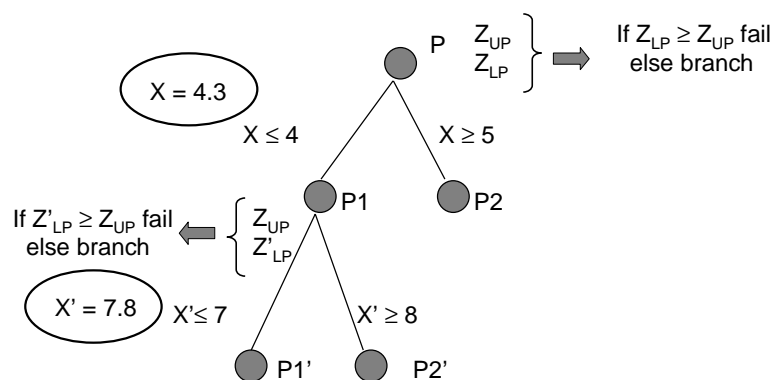
EXAMPLE: MODEL



BRANCH & BOUND based on LP



BRANCH & BOUND based on LP: EXAMPLE

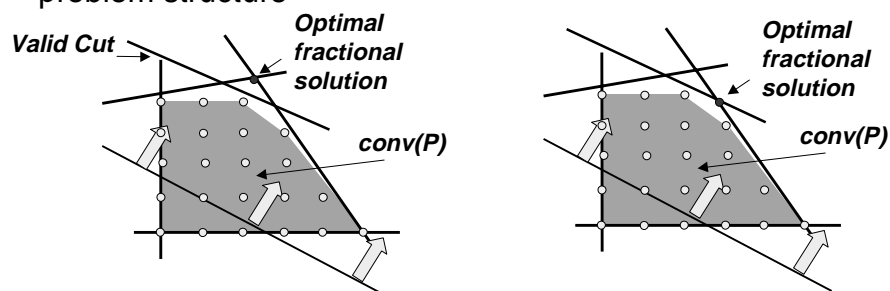


OR: CUTTING PLANES ALGORITHM

- Iterative procedure:
 - solving a linear relaxation of the problem P, x^* optimal solution
 - add cutting planes when the optimal solution of the relaxation is not integral
- Cutting Planes: [Gomory, 63]
- linear inequalities $\alpha x \leq \alpha_0$
 - should cut off the optimal solution of the Linear Relaxation
 - $\alpha x^* > \alpha_0$
 - should not remove any integer solution \implies valid cut
 - $\alpha x \leq \alpha_0 \quad \forall x \in \text{conv}(P)$ where $\text{conv}(P)$ is the convex hull of P

OR: CUTTING PLANES ALGORITHM

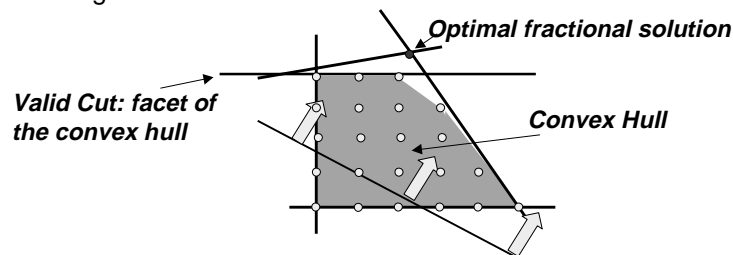
- Cutting Planes: syntactic cuts: do not exploit the problem structure



- Convergence is not guaranteed in general
- For some cases, i.e., Gomory cuts, the process converges but it can be too expensive

POLYHEDRAL CUTS

- Problem structure dependent
- Given an Integer Problem: S is the set of its solutions
 - $\text{conv}(S)$: convex hull of S
 - if we have a constraint representation of the convex hull we can optimally solve the IP with Linear Programming
 - impossible to find the $\text{conv}(S)$ efficiently
- Idea: generate cuts that are facets of the convex hull



CP2000 Tutorial - Singapore September 2000

27

OR: BRANCH & CUT

- Integrates Branch & Bound and Cutting Planes
- Two step tree search procedure: at each node
 - solving a relaxation of the original problem
 - add cuts when the optimal solution of the relaxation is not integral in order to improve the bound
- Branch when cuts are no longer effective
- In Branch & Cut in general we have a unique pool of cuts globally valid

CP2000 Tutorial - Singapore September 2000

28

OR: COLUMN GENERATION

- Method for solving large scale problems [Dantzig-Wolfe *Econometrica* 61] [Gilmore, Gomory *OR61*]
- Avoid considering all variables (say X) of the problem, but consider only a subset $X' \subset X \implies$ *MASTER PROBLEM*
- Once the master problem is solved, search for new variables in $X \setminus X'$ which can improve the solution \implies *SUBPROBLEM*
- From duality theory variables with negative reduced costs improve the solution

OR: COLUMN GENERATION-EXAMPLE

- Partition a Directed Acyclic Graph into the minimal number of paths.
- *MASTER PROBLEM* \implies Choose paths

V : set of variables (n nodes $O(2^n)$)
 $x_j \in \{0,1\}$ $x_j = 1$ if Path j is chosen
 $\min \sum_{x_j \in V} x_j$
 $\sum_{x_j \in V} a_{ij} x_j = 1$ each node belongs to one path

\implies Provides dual values λ_i

- Solve the master problem on $V' \subset V$, then add variables by solving the *SUBPROBLEM* \implies find a path with negative reduced costs

$y_j \in \{0,1\}$ $y_j = 1$ if node i is on that path
 z cost of the path
 $z - \sum_{i \in V} \lambda_i y_i \leq 0$

\implies Provides new columns x_{ji}

SPECIAL PURPOSE ALGORITHMS

- Beside the general methods (branch and bound, branch and cut, column generation), we have special purpose algorithms suited for solving a particular structured problem:
 - *network flow algorithms*
 - *graph-based algorithms*
 - ...

OVERVIEW

- Preliminaries:
 - Combinatorial Optimization Problems
 - CP and OR techniques
- Unifying frameworks
- Integration:
 - *problem modelling*
 - *problem solving*
 - *branch & bound*
 - *branch & cut*
 - *column generation*
 - *Search*

COMPARISONS

- Many studies on:
 - comparisons on the same practical application
 - [Smith et al. Constraints 96], [Puget, De Backer APMOD'95],[Darby-Dowman et al. Constraints 98], [Proll, Smith INFORMS JOC98], [Darby-Dowman, Little INFORMS JOC98]
 - comparisons for defining general similarities and differences in abstract ways
 - [Heipcke Annals of OR99],[Darby-Dowman, Little INFORMS JOC98], [Van Hentenryck, CP95], [Williams Tut.EURO XVI],
 - works aimed at defining unifying frameworks
 - [Bockmayr, Kasper INFORMS JOC 98], [Heipcke PhD99]

UNIFYING FRAMEWORKS

- Purpose: define general concepts aimed at capturing the basic concepts of CP and OR techniques
- Define basis for understanding correspondences, similarities and differences between the two approaches
- Define the basis for possible integrations

UNIFYING FRAMEWORK

- Branch and Infer [*Bockmayr Kasper INFORMS JoC98*]
 - identifies common concepts to ILP and CP
- Primitive and non primitive constraints
 - CP primitive constraints: $\{X \leq u, X \geq b, X \neq v, X = Y, \text{integer}(X)\}$
 - ILP primitive constraints: linear equalities and inequalities

- Branch and Infer based on transition rules

$$\bullet \text{ rule_name : } \frac{\langle P, S \rangle}{\langle P', S' \rangle} \quad \text{if Cond} \quad \begin{array}{l} P: \text{disjunctive subproblems} \\ S: \text{feasible solution set} \end{array}$$

UNIFYING FRAMEWORK

- Goal: derive primitive from non primitive constraints

$$\text{– } \text{bi_infer} : \frac{\langle (c \cup C) \cup P, S \rangle}{\langle (p \cup (c \cup C)) \cup P, S \rangle} \quad \begin{array}{l} p: \text{primitive, } c: \text{non primitive} \\ \text{Prim}(C) \not\rightarrow p \quad \text{Prim}(C) \wedge c \rightarrow p \end{array}$$

- CP: global constraints
 - declarative abstractions embedding powerful filtering algorithms:
derive primitive constraints $X \leq u, X \geq b, X \neq v$
- MIP: addition of cuts
 - cuts are primitive constraints

UNIFYING FRAMEWORK

- Branching: splitting the problem into subproblems

$$\begin{aligned}
 - \text{bi_branch} : & \frac{\langle C \cup P, S \rangle}{\langle \{c_1 \cup C, \dots, c_k \cup C\} \cup P, S \rangle} & C \equiv C \wedge (\forall c_i) \\
 & & \text{if } c_i \text{ primitive } \text{Prim}(C) \not\rightarrow c_i
 \end{aligned}$$

- Branching can be avoided if a subproblem is infeasible

$$- \text{bi_clash} : \frac{\langle C \cup P, S \rangle}{\langle P, S \rangle} \quad \text{Prim}(C) \rightarrow \perp$$

- *The infeasibility test is performed only on the relaxation generated by primitive constraints*

UNIFYING FRAMEWORK

- Solving Combinatorial Problems:

$$\begin{aligned}
 - \text{bi_sol} : & \frac{\langle C \cup P, S \rangle}{\langle P, S \cup S^* \rangle} & S^* = \text{extract}(\text{Prim}(C)) \\
 & & S^* \rightarrow C
 \end{aligned}$$

- Solving Combinatorial Optimization Problems: B&B *a-la* CP

$$\begin{aligned}
 - \text{bi_climb} : & \frac{\langle \{C_1, \dots, C_n\}, \{s\} \rangle}{\langle \{C \cup C, C \cup C_1, \dots, C \cup C_n\}, \{s^*\} \rangle} & s^* = \text{extract}(\text{Prim}(C)) \\
 & & f(s^*) > f(s), c \equiv (f(x) > f(s^*))
 \end{aligned}$$

- *extract: responsible of extracting a feasible solution of the relaxation*

FINAL REMARKS on COMPARISONS

- There are no general guidelines to know in advance which technique is the most appropriate
- Unifying frameworks and problem class features can help in deciding the best technique
- The integration can lead to exploit advantages of both sides.

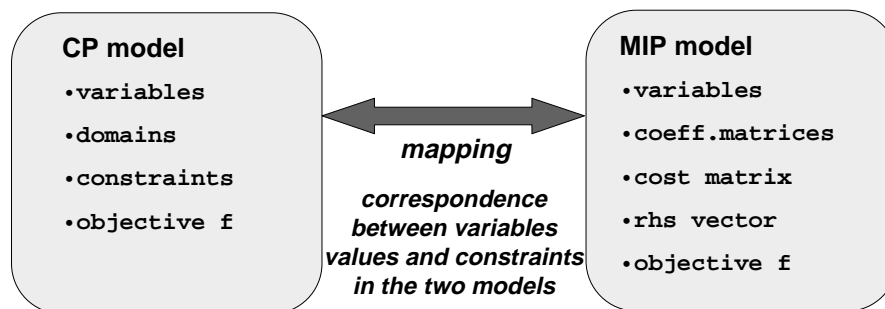
OVERVIEW

- Preliminaries:
 - Combinatorial Optimization Problems
 - CP and OR techniques
- Unifying frameworks
- Integration:
 - *problem modelling*
 - *problem solving*
 - *branch & bound*
 - *branch & cut*
 - *column generation*
 - *Search*

INTEGRATION: MOTIVATIONS

- The main motivations for integrating modeling and solving techniques from CP and MIP are:
 - Combine the advantages of the two approaches
 - CP: modelling capabilities, interaction among constraints
 - MIP: global reasoning on optimality, solution methods
 - Overcome the limitations of both
 - CP: poor reasoning on the objective function
 - MIP: not flexible models, no symbolic constraints
- Integration directions:
 - Problem modelling
 - Problem solving

INTEGRATION: MODELLING

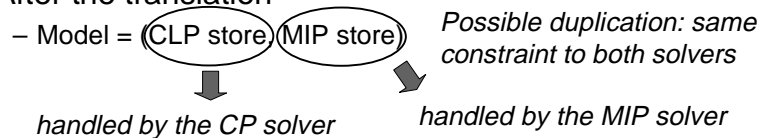


INTEGRATION: MODELLING

- CP and MIP models should coexist, cooperate or even merge in a single language.
- Different levels of integration can be achieved:
 - approaches which provide the user only the CP language and hide the OR model and the mapping between the CP and OR models
 - approaches which provide the user both models and allow to state both constraints on the CP part and the OR part
 - approaches which merge the two models in order to provide a unique model embedding both the OR and CP side

TRANSPARENT MODEL INTEGRATION

- The user works with a CP language
- The mapping and the MIP model are hidden
- Need of a translation of the CP model in terms of MIP model [Rodosek, Wallace, Hajian AnnalsOR98, Refalo CP2000]
 - automatic transformation of CLP programs in non-disjunctive form through auxiliary binary variables
 - optimization on the number of binary variables
 - mapping of global constraints
- After the translation



TRANSPARENT MODEL INTEGRATION

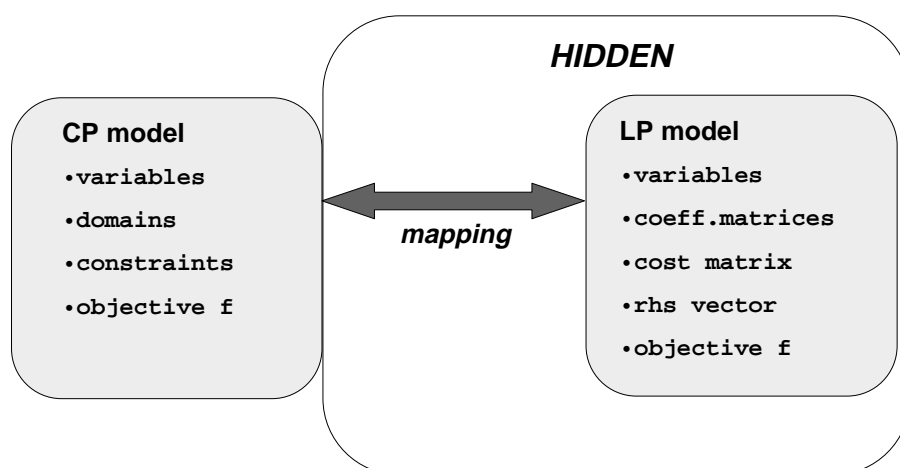
- Automatic translating constraints

`Var::[minV..maxV]` \longleftrightarrow `binary(BminV), ..., binary(BmaxV)`
 $B_{\min V} + \dots + B_{\max V} = 1$

if `Vari = j` the corresponding $B_{ij} = 1$

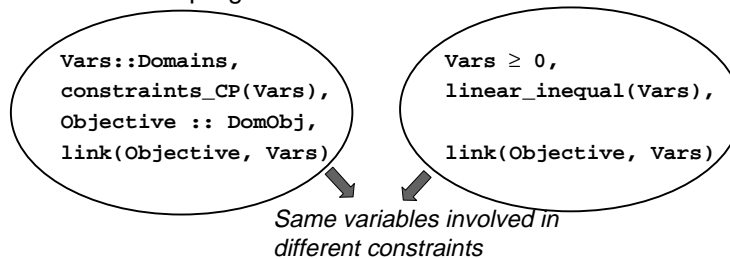
`alldifferent([V1, ..., Vn])` \longleftrightarrow $B_{11} + \dots + B_{n1} \leq 1$
.....
 $B_{1k} + \dots + B_{nk} \leq 1$

INTEGRATION: MODELLING

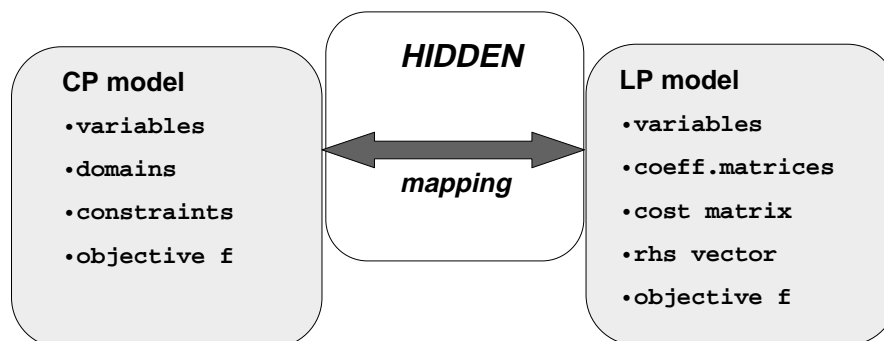


EXPLICIT MODEL COOPERATION

- The user explicitly states constraints to the two solvers. More complex programs. [Beringer, De Backer 95] [Heipcke PhD99]
- The user does not control the mapping between the two models which is in general achieved through shared variables
- In the same program:

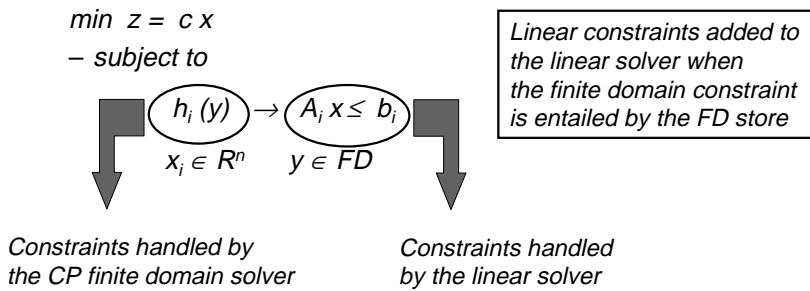


INTEGRATION: MODELLING



UNIQUE MODEL

- The user designs a unique model suitable for the hybrid solver:
MLLP: Mixed Logical/Linear Programming [Hooker et al. AAAI99]
- More complex model but closer to an hybrid architecture



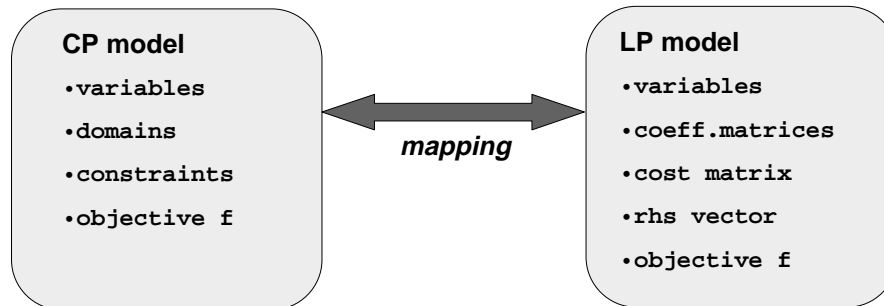
UNIQUE MODEL: GLOBAL CONSTRAINTS

- Global constraints can be introduced in MLLP: example on Variable Subscripts in Linear Constraints [Ottoson, Thorsteinsson CPAIOR00]

- $z \geq \sum_j c_j y_j$ sum of costs of assigning worker y_j to job j
- $c_j y_j$ can be substituted by z_j $z \geq \sum_j z_j$
 $y_j = k \rightarrow z_j = c_{jk} \quad \forall j \in \{1 \dots n\}, k \in \{1 \dots m\}$

or alternatively $z \geq \sum_j z_j$
 $\text{element}(y, [c_{11}, \dots, c_{1n}], z_1), \text{element}(y, [c_{21}, \dots, c_{2n}], z_2),$
 , $\text{element}(y, [c_{m1}, \dots, c_{mn}], z_m).$

INTEGRATION: MODELLING



WHICH PARTS OF THE PROBLEM ?

- The mapping defines a correspondence between the CP model and the IP model.
- Which parts of the problem are involved ?
 - the whole problem is represented in both models
 - transparent model integration [*Rodosek, Wallace, Hajian AnnalsOR98*]
 - explicit model integration of the whole problem
 - only some parts of the problem are represented in both models:
 - translation of global constraints [*Refalo CP2000*]
 - LP/IP model within global constraints [*Focacci, Lodi, Milano CP99*]
 - explicit model integration of some parts of the problem
 - some parts of the problem are represented in the CP model and other parts in the IP/LP model
 - problem decomposition [*El Sakkout, Wallace Constraints 2000*]

TRANSLATION OF THE WHOLE PROBLEM AUTOMATIC OR EXPLICIT TRANSLATION

CP solver

```
Vars::Domains,  
constraints_CP(Vars),  
Objective :: DomObj,  
link(Objective, Vars)
```

LP solver

```
Vars ≥ 0,  
linear_inequal(Vars),  
link(Objective, Vars)
```

*Exchange information on the
same entities (variables and
objective function)*

TRANSLATION OF PART OF THE PROBLEM AUTOMATIC OR EXPLICIT TRANSLATION

*For example: non linear
parts are not translated*

CP solver

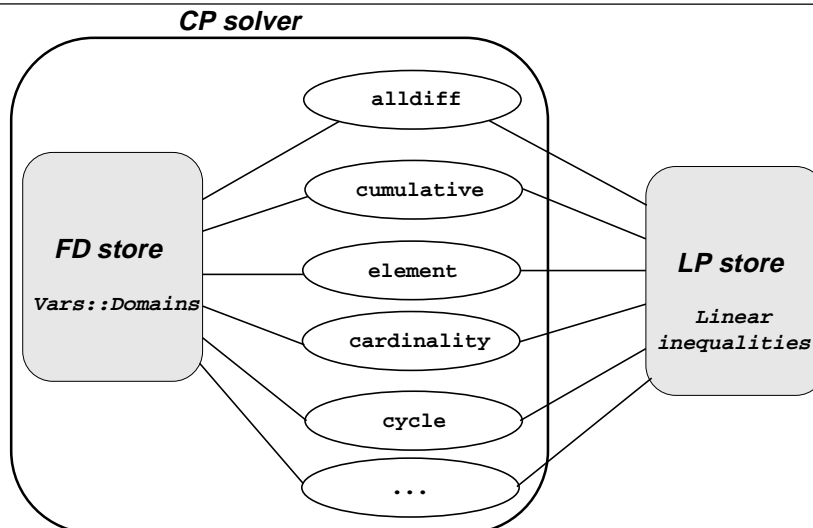
```
Vars::Domains,  
constraints_CP(Vars),  
Objective :: DomObj,  
link(Objective, Vars)
```

LP solver

```
Vars' ≥ 0,  
linear_inequal(Vars'),  
link(Objective', Vars')
```

Vars' \subset Vars
*Exchange information on the
common entities (variables and
objective function)*

TRANSLATION OF PART OF THE PROBLEM TRANSLATION OF GLOBAL CONSTRAINTS

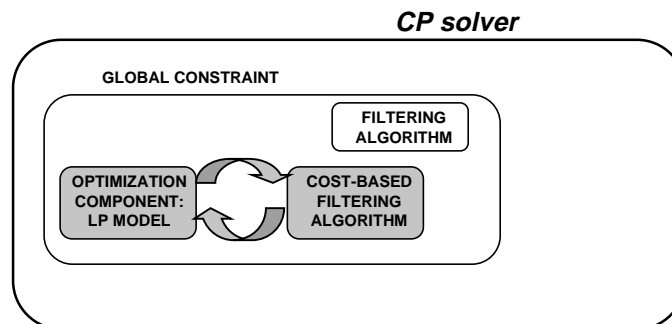


CP2000 Tutorial - Singapore September 2000

57

TRANSLATION OF PART OF THE PROBLEM LP/IP MODEL WITHIN GLOBAL CONSTRAINTS

- The user works with a CP language
 - global constraints embed a linear model [*Focacci, Lodi, Milano CP99, Regin CP99*] which allows to perform a propagation based on costs

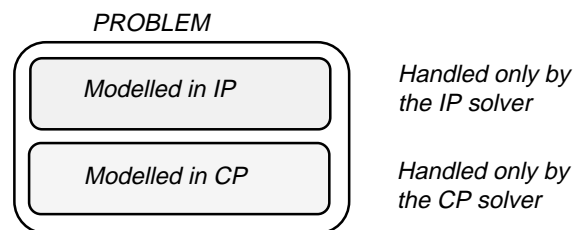


CP2000 Tutorial - Singapore September 2000

58

PROBLEM DECOMPOSITION

- If a problem can be decomposed in subproblems choose the best technique/solver to solve it. Interaction among subproblems should be handled
 - CP: subproblems are single constraints
- Application to Dynamic Scheduling [*EI Sakkout, Wallace Constraints 2000*]



OVERVIEW

- Preliminaries:
 - Combinatorial Optimization Problems
 - CP and OR techniques
- Unifying frameworks
- Integration:
 - *problem modelling*
 - *problem solving*
 - *branch & bound*
 - *branch & cut*
 - *column generation*
 - *Search*

INTEGRATION: SOLVING

- CP solving is based on interleaved search and inference (constraint propagation)
 - constraint propagation rules in global constraints exploit global problem-dependent knowledge to perform pruning
 - different constraints interact through shared variables
 - problem dependent branching strategies
- MIP solving is based on interleaved search and the computation of the optimal solution of a relaxation
 - pruning is performed on nodes for which the relaxation is infeasible or proven sub-optimal
 - branching is guided by relaxation information (relaxation provides a point in space where search should be centered)

INTEGRATION: SOLVING

- Integration on solving: two aspects
 - feasibility
 - optimality
- Concerning feasibility, CP global constraints embed powerful filtering algorithms that prune the search space
- Many of them coming from OR
 - Edge Finder [*Carlier Pinson 90*] [*Baptiste, Le Pape, Nuijten, IJCAI95*]
 - Network flow based algorithms [*Regin AAAI96*]

GLOBAL CONSTRAINTS

- References:
 - Global constraints in CHIP [*Beldiceanu Contejean, Math.Comp.Mod.94*]
 - Task Intervals [*Caseau Laborthe ICLP94*] [*Caseau Laborthe LNCS1120*] [*Caseau Laborthe JICSLP96*] [*Caseau Laborthe LNCS1120*]
 - alldifferent [*Regin AAAI94*] Symmetric alldifferent [*Regin IJCAI99*]
 - Edge Finder [*Baptiste Le Pape Nuijten, IJCAI95*], [*Nuijten Le Pape JoH98*][*Nuijten Aarts Eur.J.OR96*]
 - Sequencing [*Regin Puget CP97*]
 - Cardinality [*Regin AAAI96*]
 - Sortedness [*Bleuzen Colmerauer Constraints 2000*]

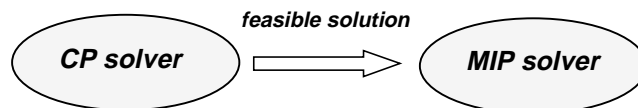
INTEGRATION: SOLVING

- Optimization Problems
- CP and MIP/LP solvers should interact and cooperate by exchanging information
- Different levels of integration can be achieved:
 - approaches which keep the two solvers separate and independent exchanging information
 - sequential computations
 - interleaved computations
 - approaches which integrate an optimization component (LP solver) in global constraints

LOOSE INTEGRATION

SEQUENTIAL COMPUTATION

- CP and MIP solvers are used in sequence:
 - the CP solver computes the first feasible solution (rather quickly)
 - the MIP solver uses this solution as warm start for a Simplex based Branch & Bound
- Approach used in the British Airways Fleet Assignment [*Hajjaan et al. TR95/09-01*]



TIGHTER INTEGRATION

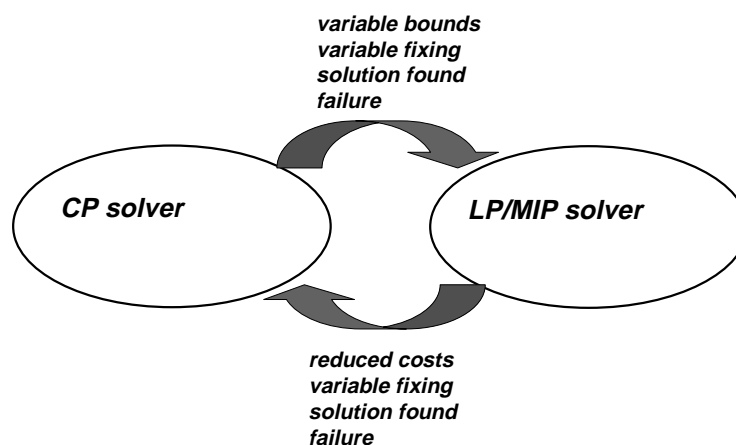
INTERLEAVED COMPUTATION

- CP and LP solvers are interleaved:
 - the CP solver performs propagation
 - LP solver optimally solves the linear relaxation
- CP and LP solvers exchange information:
 - variable bounds
 - variable fixing
 - optimal solution of LP/reduced costs
- Branching performed on the CP or on the MIP side
 - CP branching based on problem structure or on var. domains
 - MIP branching based on optimal solution of the relaxation

TIGHTER INTEGRATION INTERLEAVED COMPUTATION

- Solving technique for modelling approaches where the CP and the LP model are kept separate
 - the user can explicitly impose which constraints are handled by the CP solver and which ones are handled by the LP solver
[Beringer-De Backer95] [Heipcke PhD99]
 - the automatic linearization (of global constraints) is sent to the LP solver while the propagation of global constraints is performed as usual by the CP solver
[Rodosek, Wallace, Hajian AnnalsOR98], [Refalo CP2000], [Focacci Lodi Milano CP99]

TIGHTER INTEGRATION INFORMATION EXCHANGES



TIGHTER INTEGRATION

EXPLOITATION OF RESULTS

- Where results of the CP solver can be used
 - domain bound reduction \Rightarrow sent to the LP solver which adds the new bounds to the problem formulation (cut)
 - domain value removal \Rightarrow the corresponding binary variable fixed to 0
 - variable instantiation \Rightarrow the corresponding binary variable fixed to 1
 - solution found \Rightarrow upper bound available

TIGHTER INTEGRATION

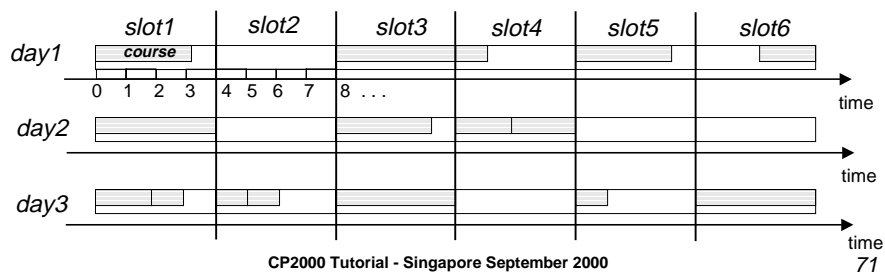
EXPLOITATION OF RESULTS

- Where results of the LP solver can be used
 - variable fixing $\left\{ \begin{array}{l} \text{to 0} \Rightarrow \text{corresponding value removed} \\ \text{to 1} \Rightarrow \text{corresponding variable instantiated} \end{array} \right.$
 - reduced costs \Rightarrow domain filtering
 - solution found \Rightarrow lower bound on the objective function variable

*Suggestions for
guiding the search
more on this later...*

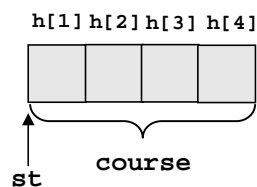
EXAMPLE

- Timetabling problem from [Caseau Laborthe CP97]
 - 4-Hours Slots - 1 to 4 Hours Courses
 - Two courses cannot overlap
 - A course must be contained in a single slot
 - Preferences are associated with: Course-Slot assignments
 - Maximize Sum of Preferences



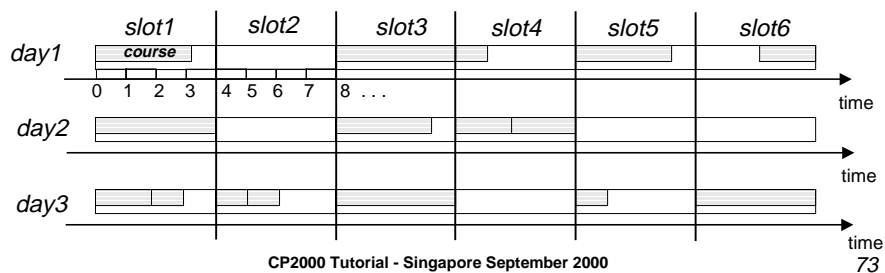
EXAMPLE

- Variables associated to each course lasting Nh hours
 - Start time st : domain contains possible starting time for the course
 - Single hours $h[i]$ $i=1..Nh$: domain contains hours along the time line
 - Course $course$: domain contains slots
- Constraints link the different variables



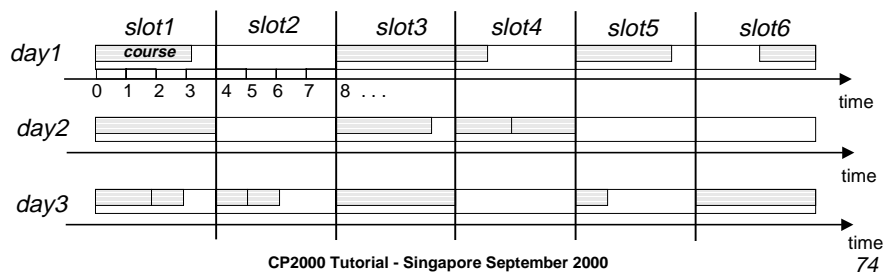
EXAMPLE

- Course lasting $N_h=3$ hours
 - `st::[0,1,4,5,8,9,...]`
 - `h[1]::[0,1,4,5,8,9,...]` `h[2]::[1,2,3,4,9,10,...]`
 - `h[3]::[2,3,6,7,10,11,...]`
 - `course::[day1slot1,day1slot2,...,day2slot1,...,daynslotm]`



EXAMPLE

- Constraints
 - `TimeTable(startVars, durations);`
 - `// relaxing the contiguity constraint`
 - `AllDiffCost(singleHours, objective, costsMtx);`
 - `// subproblem defined by 3 and 4 hour courses`
 - `AllDiffCost(courses34Hours, objective1, costsMtx1);`



MODEL MAPPING

`AllDiffCost(singleHours, objective, costsMtx);`
 - `singleHours`: array of SumOfDurations variables
 whose domain contains single hours

`singleHours[i]=j` \leftrightarrow `xij=1`
`singleHours[i]≠j` \leftrightarrow `xij=0`

`costMtx[i][j]` \leftrightarrow `cij`
`costMtx[i][j] = ∞` if `xij=0`
`costMtx[i][j] = -pref[i][j]/dur`

ILP-Model

$$\min z = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1..n \quad (A)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1..n \quad (B)$$

`xij` integer



LB



reduced costs

INTERACTIONS AMONG SOLVERS

- LB-based Propagation

from *LB* towards objective function $Z::[Z_{min}..Z_{max}]$

$$LB \leq Z$$

- Reduced Cost-based Propagation

from *reduced costs* towards decision variables

$x_i::[i_1, i_2, \dots, i_m]$ each i_j has a gradient function
 $\text{grad}(x_i, i_j)$ measuring the cost to pay if $x_i = i_j$

$$\text{if } LB + \text{grad}(x_i, i_j) \geq Z_{max} \quad \text{then } x_i \neq i_j$$

TRIGGERING EVENTS

- In CP constraint propagation is triggered each time the domain of one variable appearing in it is modified
 - Variable domains are changed both due to
 - other constraint propagation
 - variable fixing from the linear solver
 - reduced cost based propagation
 - In particular, the reduced cost based propagation is triggered when the LP computes a new solution or when the upper bound of the objective function changes
- LP solver triggered each time a value in the solution of the LP is deleted from the variable domain

SPECIAL PURPOSE ALGORITHMS

- Instead of using an LP solver, we can embed in global constraints special purpose algorithms when the linear relaxation of the constraint represents a well structured problem
- Characteristics of the algorithm
 - polynomial time complexity
 - incremental behaviour
 - should provide the optimal solution of the relaxed problem
 - should (possibly) provide reduced costs for enhancing propagation

GLOBAL CONSTRAINTS for OPTIMIZATION

- References:
 - Global constraint for TSP [*Caseau Laborthe ICLP97*] [*Focacci Lodi Milano Elect.Notes on DM 99*], TSPTW [*Focacci Lodi Milano ICLP99*]
 - Matching problems [*Caseau Laborthe CP97*] [*Focacci Lodi Milano CP-AI-OR 99*]
 - Reduced cost fixing in global constraints [*Focacci Lodi Milano CP99*]
 - Cardinality constraints + costs [*Regin CP99*]

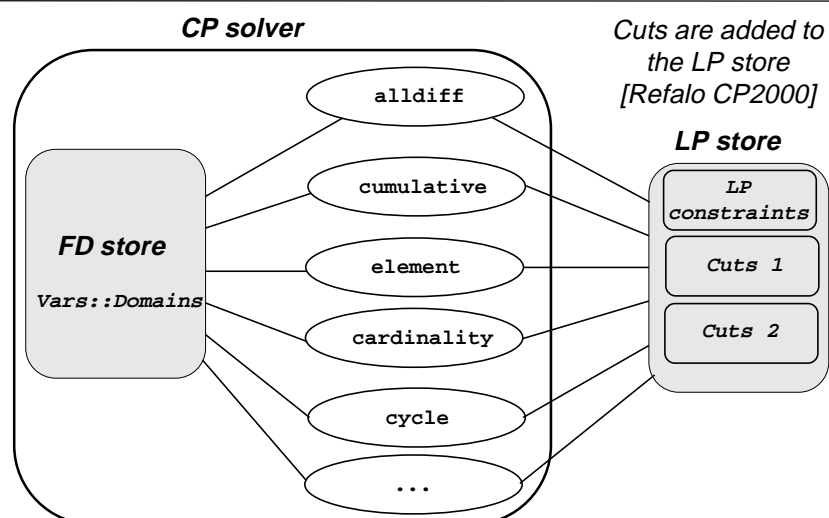
OVERVIEW

- Preliminaries:
 - Combinatorial Optimization Problems
 - CP and OR techniques
- Unifying frameworks
- Integration:
 - *problem modelling*
 - *problem solving*
 - *branch & bound*
 - *branch & cut*
 - *column generation*
 - *Search*

CUTTING PLANES IN CP

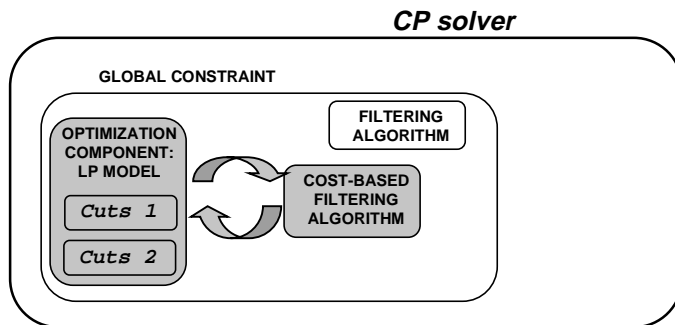
- When the LP is used, the linear relaxation of constraints can be enhanced with the addition of cutting planes:
 - Global cuts
 - Local cuts
- New LP, called LP^{cut} which provides:
 - more precise bound: $Sol(LP^{cut}) \geq Sol(LP)$
 - reduced costs
- Different ways of adding cuts to the LP formulation:
 - to the LP store
 - within each constraint

CUTTING PLANES IN CP



CUTTING PLANES IN CP

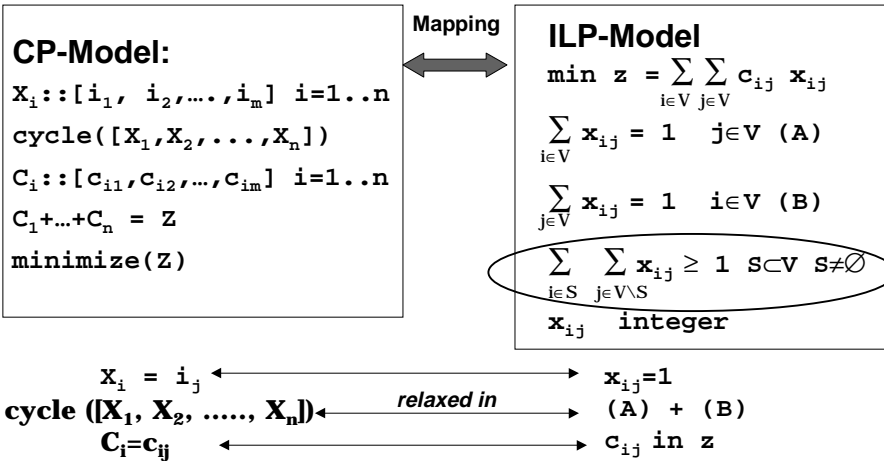
Cuts are added within each constraint [Focacci, Lodi, Milano CP2000]



EXAMPLE: **cycle** CONSTRAINT

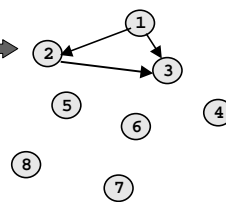
- Relaxation of the constraint: Assignment Problem
 - the AP finds a set of (possibly) disjoint subtours that minimizes the sum of costs
 - the CP optimal solution satisfies the integrality constraints
- Sub-tour constraints are relaxed
- Sub-tour elimination cuts (SECs) can be separated in polynomial time [Padberg, Rinaldi 90]
 - find the subtour polytope: in general provides a fractional solution where no subtour are present

EXAMPLE: cycle CONSTRAINT



EXAMPLE: cycle CONSTRAINT

- Example: subtour elimination cut
- suppose the optimal solution of the AP \rightarrow
- provides the following subtour:
- we can add to the AP formulation the following cutting plane that removes the subtour



$$x_{21} + x_{12} + x_{31} + x_{13} + x_{32} + x_{23} \leq 2$$

The new problem is LP^{cut}

- $\text{Sol}(LP^{\text{cut}}) \leq \text{Sol}(LP)$
- when all cuts are satisfied $\text{Sol}(LP^{\text{cut}}) = \text{Sol}(LP)$

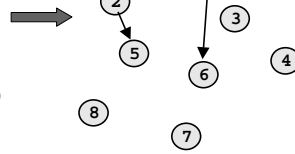
EXAMPLE: cycle CONSTRAINT

- During search

- at each node compute valid cuts and add to the LP
- subtour elimination cuts are globally valid
- can be removed upon backtracking

– Example: subtour elimination cut

$$x_{21} + x_{12} + x_{31} + x_{13} + x_{32} + x_{23} \leq 2$$



when during search x_2 is assigned to

5 and x_1 is assigned to 6 the cut is

trivially satisfied but another can be computed

$$x_{25} + x_{26} + x_{21} + x_{52} + x_{56} + x_{51} + x_{62} + x_{65} + x_{61} + x_{16} + x_{15} + x_{12} \leq 3$$

LOCALLY VALID CUTS

- Cuts added during search can exploit information on variable domains.
- Tighter integration of cutting planes in CP [RefaloCP99]
 - Definition of the convex hull of the LP + variable domain bounds.

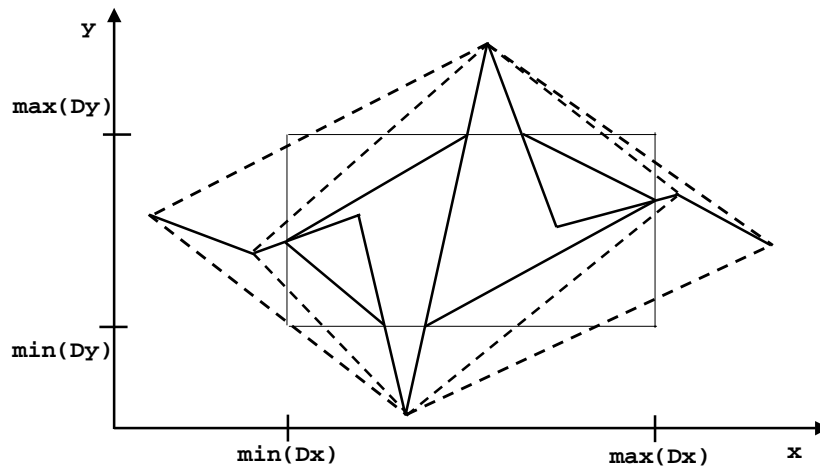
Cartesian product of domains

Solution set for c

$$T = \text{convex_hull}(D_X) \cap \left\{ \bigcap_{c \in P} \text{convex_hull}(S(c)) \right\}$$

$$T' = \bigcap_{c \in P} \text{convex_hull}(S(c) \cap D_X) \longrightarrow \text{When added during search depend on constraint propagation LOCALLY VALID}$$

EXAMPLE: **piecewise** linear function



CP2000 Tutorial - Singapore September 2000

89

EXAMPLE: **piecewise** linear function

- Generated Cuts are facets of the convex hull
 - from $(\min(Dx), f(\min(Dx)))$ to $(u, f(u))$ $u \in Dx$, $f(u) \in Dy$ and the slope s_1 is *maximal*
 - from $(\min(Dx), f(\min(Dx)))$ to $(u, f(u))$ $u \in Dx$, $f(u) \in Dy$ and the slope s_2 is *minimal*
 - from $(\max(Dx), f(\max(Dx)))$ to $(u, f(u))$ $u \in Dx$, $f(u) \in Dy$ and the slope s_3 is *maximal*
 - from $(\max(Dx), f(\max(Dx)))$ to $(u, f(u))$ $u \in Dx$, $f(u) \in Dy$ and the slope s_4 is *minimal*
- Other 4 cuts for Dy
- Locally valid cuts must be removed in backtracking

CP2000 Tutorial - Singapore September 2000

90

LOGICAL CUTS

- Hooker recognized the equivalence of cutting plane techniques with resolution for propositional logic
 - linear inequalities can be seen as clauses
 - valid cuts can be derived by logical inference: a cut is nothing other than a logical implication of the constraint set.

$4x_1 + 2x_2 + x_3 \geq 3$ *equivalent* $x_1 \vee x_3$
 \longleftrightarrow $x_1 \vee x_2$

x_1	x_2	x_3	Value
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

In fact the linear inequality is satisfied iff x_1 is set to 1 or x_2 and x_3 are both 1

LOGICAL CUTS

- Resolution provides logical cuts:

$x_1 + x_2 + x_3 \geq 1$ $(1-x_1) + x_2 + (1-x_4) \geq 1$		$x_1 \vee x_2 \vee x_3$ $\neg x_1 \vee x_2 \vee \neg x_4$
\Downarrow	<i>equivalent</i>	\Downarrow
$x_2 + x_3 + (1-x_4) \geq 1$ <i>Rank 1 cut</i>	\longleftrightarrow	$x_2 \vee x_3 \vee \neg x_4$ <i>Resolvent</i>

- Connections between k-resolution (k-rank cuts) and k-consistency
 - k-resolution generates all resolvent of less than k literals not already absorbed by other clauses

LOGICAL CUTS IN CLP(PB)

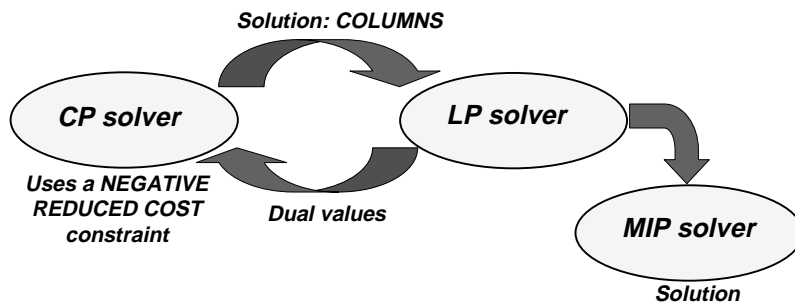
- Integration of logical cuts in CLP(PB) [*Barth Bockmayr ICLP95*] where variables are finite domains and range on $[0,1]$
 - interpret a CLP(PB) program as an Integer Problem in the form $Ax \leq b \quad x \in \{0,1\}^n$. Then consider the linear relaxation LP
 - add valid inequalities
- Advantages:
 - constraints are translated in a solved form
 - entailment decided earlier
 - unfeasibility can be detected before enumeration
 - use of lower/upper bounds

OVERVIEW

- Preliminaries:
 - Combinatorial Optimization Problems
 - CP and OR techniques
- Unifying frameworks
- Integration:
 - *problem modelling*
 - *problem solving*
 - *branch & bound*
 - *branch & cut*
 - *column generation*
 - *Search*

COLUMN GENERATION and CP

- General Framework [Junker et al. CP99]
- Master Problem: MIP
- Subproblem: CSP



COLUMN GENERATION and CP

```
v' := getInitialColumns()
repeat
  λ := solveLP(v')
  {xj1, ..., xjk} := solveSubProblem(λ)
  v' := v' ∪ {xj1, ..., xjk}
until {xj1, ..., xjk} = ∅
```

- Applied to crew rostering application [Junker et al. CP99]
 - Path constraint based on set variables uses dynamic programming techniques for propagation
 - uses shortest path algorithm for Acyclic graphs
- Advantage of using CP: the subproblem can be formulated by considering many problem dependent constraints

OVERVIEW

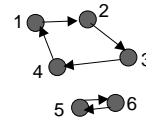
- Preliminaries:
 - Combinatorial Optimization Problems
 - CP and OR techniques
- Unifying frameworks
- Integration:
 - *problem modelling*
 - *problem solving*
 - *branch & bound*
 - *branch & cut*
 - *column generation*
 - *Search*

GUIDING SEARCH

- Exploiting results from the optimal solution of the relaxation in order to guide the search
 - the search is focussed on more “promising” values
- Example of search strategies based on relaxation:
 - violated constraints
 - subtour elimination strategy
 - branching on variables with fractional values
 - notion of regret
 - probe backtracking

SUBTOUR ELIMINATION STRATEGY

- Example on TSP:
 - relaxation: Assignment Problem
 - optimal solution of AP is integer, but possibly presents subtour
 - violation of subtour constraints in TSP



- Start from the optimal AP solution
- Select a subtour and break it
 - Example: subtour 1 2 3 4
 - $\text{Next}_1 \neq 2 \vee (\text{Next}_1=2, \text{Next}_2 \neq 3) \vee (\text{Next}_1=2, \text{Next}_2=3, \text{Next}_3 \neq 4) \vee (\text{Next}_1=2, \text{Next}_2=3, \text{Next}_3=4, \text{Next}_4 \neq 1)$
 - $1 > 2 \vee (1 < 2, 2 > 3) \vee (1 < 2, 2 < 3, 3 > 1) \vee (1 < 2, 2 < 3, 3 < 4, 4 > 1)$ more suitable for CP
 - [Focacci Lodi MilanoCP98 Wrk. on LSCO] applies subtour generation in CP

BRANCHING ON FRACTIONAL VARs

- LP provides a solution with fractional values
- Select variables that are assigned to fractional values
 - Example: optimal LP solution $X_i = 4.2$
 - Branching
 - $X_i \leq 4 \vee X_i \geq 5$
- Used in OR Branch and Bound

NOTION OF REGRET

- Regret: difference between the best value and the second best
- Select the variable with maximal regret
 - should be assigned first to its best values otherwise the solution would be worsened too much
- Computation of regret on the cost matrix [*Caseau Laborthe CP97*]
- Computation of regret on reduced costs [*Focacci Lodi MilanoCP99*]
 - select the variable with higher minimal reduced cost
 - assign the value suggested by the relaxation

NOTION OF REGRET

- Computation of regret on the cost matrix [*Caseau Laborthe CP97*]

$$\text{Regret}_i = c_{i,\text{best}} - c_{i,\text{second}}$$

- Computation of regret on reduced costs [*Focacci Lodi MilanoCP99*]
 - select the variable with higher minimal reduced cost
 - assign the value suggested by the relaxation

$$\text{Regret}_i = \min_{\substack{k=1..n \\ k \neq \text{opt}(i)}} \{\bar{c}_{ik}\}$$

PROBE BACKTRACKING

- Backtrack search supported by lookahead procedures (*probe generators*) which dynamically generate potentially good assignments (*probes*). [*EI Sakkout, Wallace Constraints 2000*] [*Purdom Haven SIAM J. on Computing97*]
 - the probe generator assign each variable a tentative value
 - focus the backtrack search on regions where the probe violates constraints
 - probe generator should provide good solutions (super-optimal)
- Unimodular Probing Algorithm: the LP finds optimal integer solutions on unimodular subproblems which are considered as probes

INCOMPLETE APPROACHES

- Integration of CP and Local Search
 - CP global search is used to find an initial (feasible) solution and LS is applied to improve this solution;
 - within a CP framework, the exhaustive exploration is stopped at a chosen level of the search tree and the leafs are reached through LS;
 - within a LS (metaheuristic) framework, CP global search is used to exhaustively explore the neighborhood, or to complete in optimal way a partial solution.
- References: [*Li et al. "Modern Heuristic Search Methods", John Wiley96*], [*Shaw CP98*], [*Pothos, Richards, Cp98 Wks.on really hard problems*], [*Michel, Van Hentenryck CP97*], [*Psarras et al. European JOR97*], [*Pesant, Gendrau CP96*], [*Nuijten LePape JOH98*], [*Caseau et al. CP99*] [*DeBacker, Furnon, Shaw CPAIOR99*], [*Caseau, Laburthe CPAIOR99*]

SYSTEMS

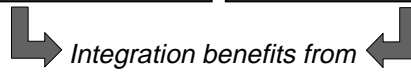
- Prolog III [*Colmerauer CACM(33) 90*],
- CHIP [*Dincbas et al., JICSLP88*],
- 2LP [*McAloon, Tretkoff PPCP93*]
- CLP(PB) and COUPE [*Barth, Bockmayr ICLP95*] [*Kasper PhD98*],
- OOPDB [*Barth PhD 96*]
- Eclipse^e [*Wallace et al.97*]
- ILOG [*Puget SPICIS94*], [*Puget, Leconte ILPS95*]
 - *Solver Planner Dispatcher Scheduler*
- OPL [*Van Hentenryck MIT Press99*]
- SCHEDEns [*Colombani PhD97*]
- COME [*Heipcke PACT96*]
- CLAIRE [*Caseau Laburthe JICSLP Wks on Multi Paradigm Logic98*],
- SALSA [*Caseau Laburthe CP98*],
- LOCALIZER [*Michel, Van Hentenryck CP97*]
- *many others.....*

TO KNOW MORE...

- Conferences and Journals
 - Conferences on Constraints: CP, PACLP, Constraints...
 - Conferences on AI: ECAI, IJCAI, AAAI, AIJ...
 - Conferences on OR: IFORS, INFORMS, Annals of OR
- INFORMS Journal on Computing (Special issue vol.10, No.3 1998)
- Journal of Heuristics (Special issue on CP-AI-OR99, to appear)
- CHIC2 Deliveries
- Workshops on the subject:
 - *CP98 and CP99 Workshop on Large Scale Combinatorial Optimisation and Constraints*
 - *CP-AI-OR99 and CP-AI-OR2000 Workshop on integration of AI and OR techniques in CP for Combinatorial Optimization*
 - *CP-AI-OR2001 will be held in London IC PARC*
 - *AAAI2000 Workshop on integration of AI and OR techniques for Combinatorial Optimization*

CONCLUSION & FUTURE DIRECTIONS

- | | |
|--|--|
| <ul style="list-style-type: none">• CP advantages:<ul style="list-style-type: none">– easy modeling (less bugs)– constraint propagation– guide search– flexibility• CP drawbacks:<ul style="list-style-type: none">– no global reasoning– no optimality reasoning | <ul style="list-style-type: none">• OR advantages:<ul style="list-style-type: none">– global problem view (LP)– exploitation of structure– optimization problems• OR drawbacks:<ul style="list-style-type: none">– models are less flexible– software engineering poor |
|--|--|


*Integration benefits from
the advantages of both*

CONCLUSION & FUTURE DIRECTIONS

- CP Problem modelling
 - composition of basic components (constraints)
- CP Problem solving
 - interaction of constraint propagation algorithms
- OR provides new components/algorithms
 - relaxation (LP, specific algorithms)
 - cut generation
 - dynamic programming
 - filtering algorithms (e.g. Edge finder)
- Collaboration through the constraint store

CONCLUSION & FUTURE DIRECTIONS

- Integration can exploit
 - problem decomposition
 - problem abstraction
 - problem special (sub)structure
 - problem *perspective* (cooperative solvers)
 - different solvers capabilities
 - exchange results
- Guidelines
 - each solver manages the sub-problem it is more suitable for
 - cheap solvers first, lower solver later
 - define events triggering solver computations

CONCLUSION & FUTURE DIRECTIONS

- Integration: the other way round
 - Can OR benefit from the CP paradigm ?
- Software engineering: modelling tools
- Constraint propagation during search
- Global constraints in IP [*Bockmayr Kasper INFORMS J. Comp.98*]
 - TSP structure
 - Assignment structure