Constraint Integer Programming

A New Approach To Integrate CP and MIP

Timo Berthold

Zuse Institute Berlin

joint work with T. Achterberg, S. Heinz, T. Koch, K. Wolter

DFG Research Center MATHEON

Mathematics for key technologies

Paris, 05/21/2008

CP vs. MIP

Constraint Programming (CP)

- Domains of variables are (arbitrary) sets
- Constraints are (arbitrary) subsets of domain space
- High flexibility in modeling, natural but very general concept

Mixed Integer Programming (MIP)

- Domains are intervals in $\mathbb{Q}$ or $\mathbb{Z}$
- Constraints and objective function are linear
- Highly structured, specialized algorithms, restricted modeling

Constraint Integer Programming (CIP)

- Linear objective function
- Arbitrary constraints, but ...
- fixing all integer variables always leaves LP (as in MIP)

Relation to CP and MIP

- Every MIP is a CIP.
- Every CP over a finite domain space is a CIP.

CP-formulation:

$$\begin{array}{lll} \min & \text{length}(x) & \\ s.t. & \text{alldiff}(x_1, \dots, x_n) & \\ & x \in \{1, \dots, n\}^n & \end{array}$$

MIP-formulation:

$$\begin{array}{lll} \min & \sum_{e \in E} d_e \, x_e & \\ s.t. & \sum_{e \in \delta(v)} x_e = 2 & \forall\, v \in V \\ & \sum_{e \in \delta(S)} x_e \geq 2 & \forall\, S \subset V, S \neq \varnothing \\ & x_e \in \{0, 1\} & \forall\, e \in E \end{array}$$

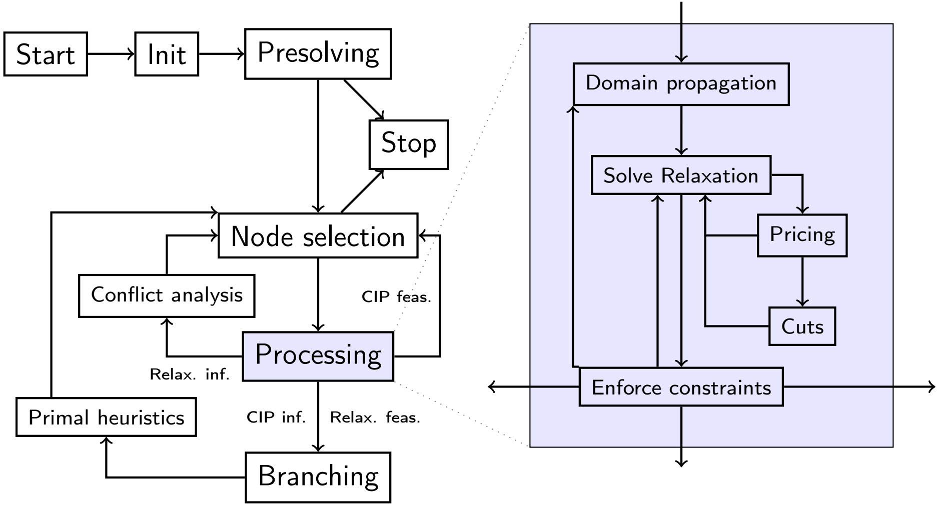
CIP-formulation:

$$\begin{array}{lll} \min & \sum_{e \in E} d_e \, x_e & \\ s.t. & \sum_{e \in \delta(v)} x_e = 2 & \forall\, v \in V \\ & \text{nosubtour}(x) & \\ & x_e \in \{0, 1\} & \forall\, e \in E \end{array}$$

Single nosubtour constraint rules out subtours (e.g. by domain propagation).
may also separate subtour elimination inequalities.

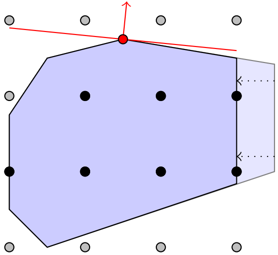
SCIP (Solving Constraint Integer Programs) . . .

- is a branch-and-bound framework,
- is constraint based,
- incorporates
  - CP features (domain propagation),
  - MIP features (cutting planes, LP relaxation), and
  - SAT-solving features (conflict analysis, restarts),
- has a modular structure via plugins,
- provides a full-scale MIP solver,
- is free for academic purposes,
- and is available in source-code under `http://scip.zib.de` !


Start
Init
Presolving
Stop
Node selection
Conflict analysis
CIP feas.
Processing
Relax. inf.
Primal heuristics
CIP inf.
Relax. feas.
Branching
Domain propagation
Solve Relaxation
Pricing
Cuts
Enforce constraints

Task

- Simplify model, remove redundant parts
- Strenghten formulation
- Extract information, recognize structure

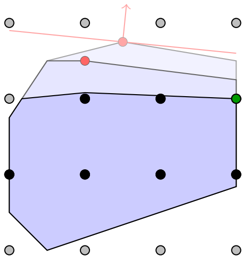


Techniques

- Variables: dual fixing, bound strengthening
- Constraints: coefficient tightening, upgrading
- Restarts: abort search, reapply global presolving

Task

- Strengthen relaxation
- Add valid constraints
- Generate on demand

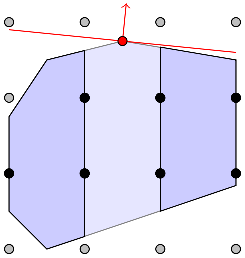


Techniques

- General (for MIP): Gomory, c-MIR, strong Chvátal-Gomory, implied bounds, $\{0, \frac{1}{2}\}$-cuts, ...
- Problem Specific: clique, knapsack, flow cover, MCF, ...

Task

- Divide into subproblems
- Improve local dual bounds
- Early branchings most important!

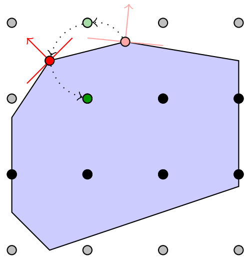


Techniques

- Branching on Variables: most infeasible, pseudocost, strong, reliability, inference branching
- Branching on Constraints: SOS1, SOS2 branching

Task

- Improve primal bound
- Effective on average, but no warranty
- Solutions guide remaining search



Techniques

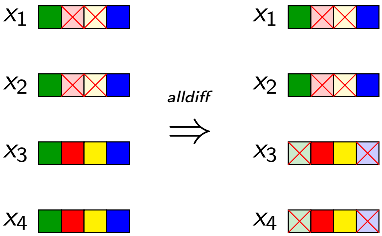
- Rounding: set fractional variables to feasible integral values
- Diving: simulate DFS in the branch-and-bound tree
- Objective diving: manipulate objective function
- Large Neighborhood Search: solve some sub-CIP

Further Components for Solving CIPs

- Node selection: which subproblem should be considered next?
- Propagation: simplifies problem, improves dual bound locally
- Pricing: allows dynamic generation of variables
- Conflict analysis: learns from infeasible subproblems

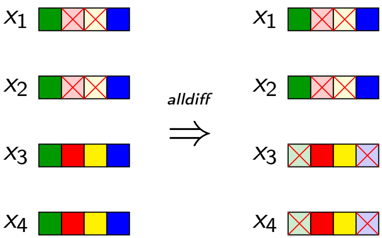
Further Components for Solving CIPs

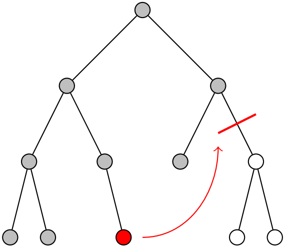
- Node selection: which subproblem should be considered next?
- Propagation: simplifies problem, improves dual bound locally
- Pricing: allows dynamic generation of variables
- Conflict analysis: learns from infeasible subproblems

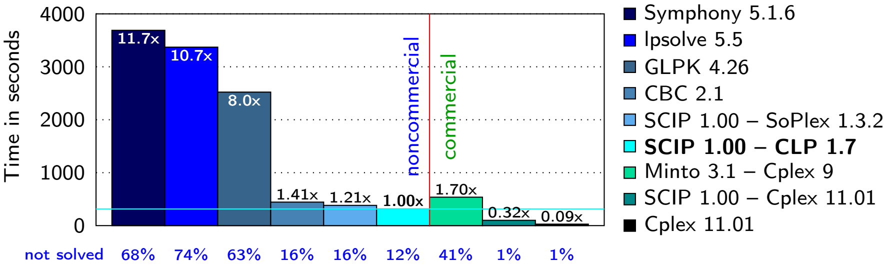


Further Components for Solving CIPs

- Node selection: which subproblem should be considered next?
- Propagation: simplifies problem, improves dual bound locally
- Pricing: allows dynamic generation of variables
- Conflict analysis: learns from infeasible subproblems







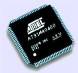
Results taken from Hans Mittelmann (04/19/2008)
`http://plato.asu.edu/ftp/milpf.html`

Application: Chip Design Verification

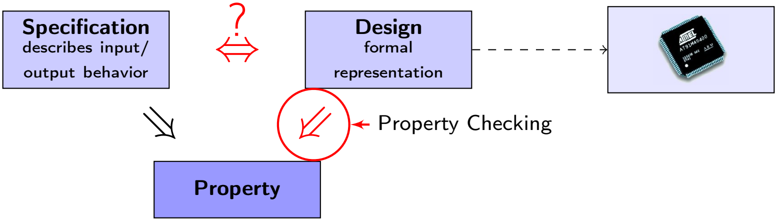
Specification
describes input/
output behavior

$\overset{?}{\Leftrightarrow}$

Design
formal
representation



Application: Chip Design Verification

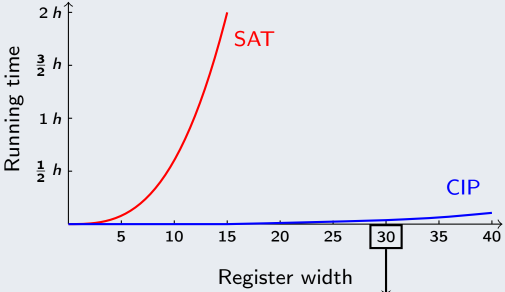


Property checking

- Derive certain properties from specification
- Check whether they hold for the design
- Leads to feasibility problems
- Can be modeled as SAT instance or as CIP

CIP versus SAT

- CIP has constraints for standard operations:
  - addition
  - subtraction
  - multiplication
  - shift left / right
  - ...
- SAT has only one constraint type



Constraints	422	152026
Variables	3714	50756

Constraint Integer Programming
A New Approach To Integrate CP and MIP

Timo Berthold
Zuse Institute Berlin

joint work with T. Achterberg, S. Heinz, T. Koch, K. Wolter

DFG Research Center MATHEON
Mathematics for key technologies