# Basic concepts of SCIP

## CO@Work Berlin

Stefan Heinz

MATHEON/ZIB

09/22/2009

# Outline

## SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

Solving Constraint Integer Programs

Summary and Outlook

# Outline

## SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

Solving Constraint Integer Programs

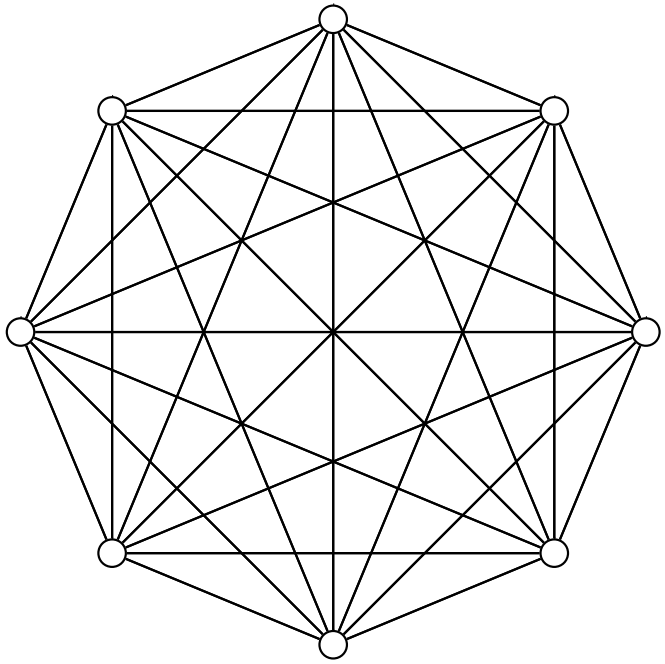
Summary and Outlook

# Traveling Salesman Problem (TSP)

## Definition (TSP)

Given a complete graph $G = (V, E)$ and distances $d_e$ for all $e \in E$:

Find a Hamiltonian cycle (cycle containing all nodes, tour) of minimum length.



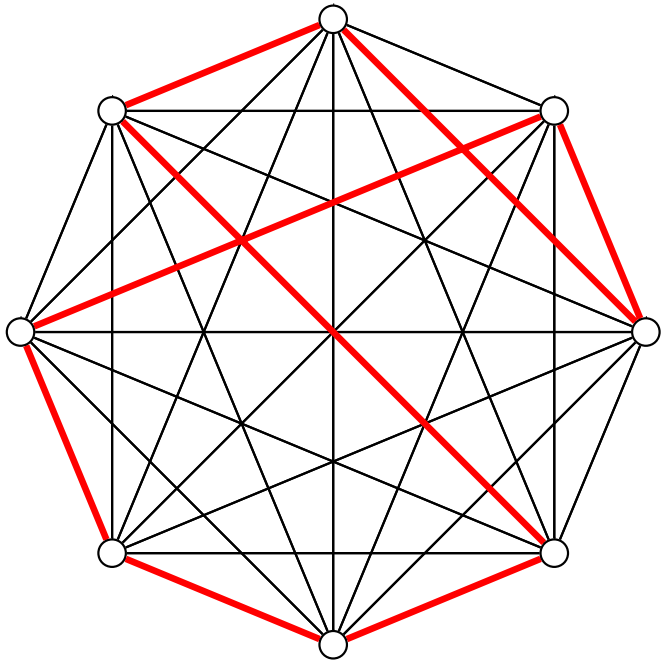
$K_8$

# Traveling Salesman Problem (TSP)

## Definition (TSP)

Given a complete graph $G = (V, E)$ and distances $d_e$ for all $e \in E$:

Find a Hamiltonian cycle (cycle containing all nodes, tour) of minimum length.



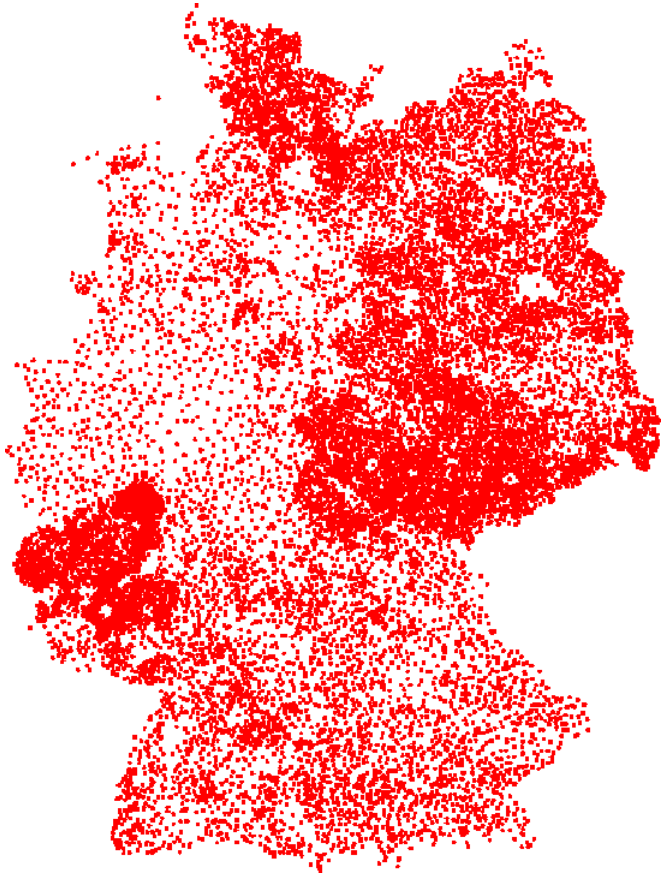
$K_8$

# Traveling Salesman Problem (TSP)

## Definition (TSP)

Given a complete graph $G = (V, E)$ and distances $d_e$ for all $e \in E$:

Find a Hamiltonian cycle (cycle containing all nodes, tour) of minimum length.

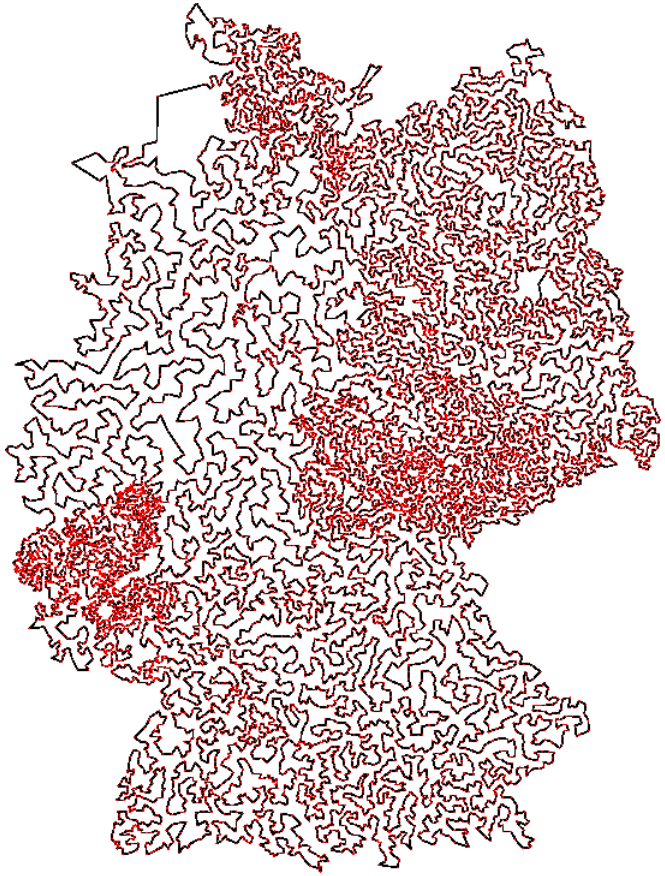


# Traveling Salesman Problem (TSP)

## Definition (TSP)

Given a complete graph $G = (V, E)$ and distances $d_e$ for all $e \in E$:

Find a Hamiltonian cycle (cycle containing all nodes, tour) of minimum length.

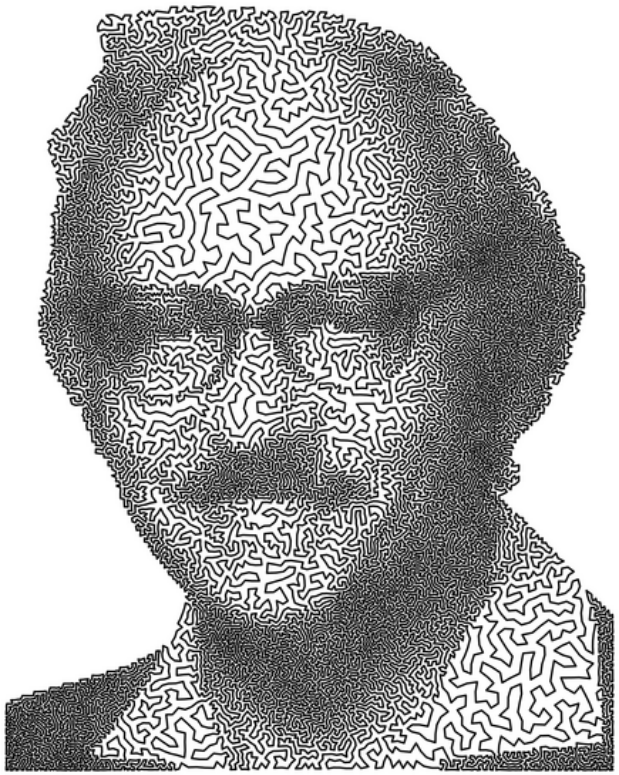


# Traveling Salesman Problem (TSP)

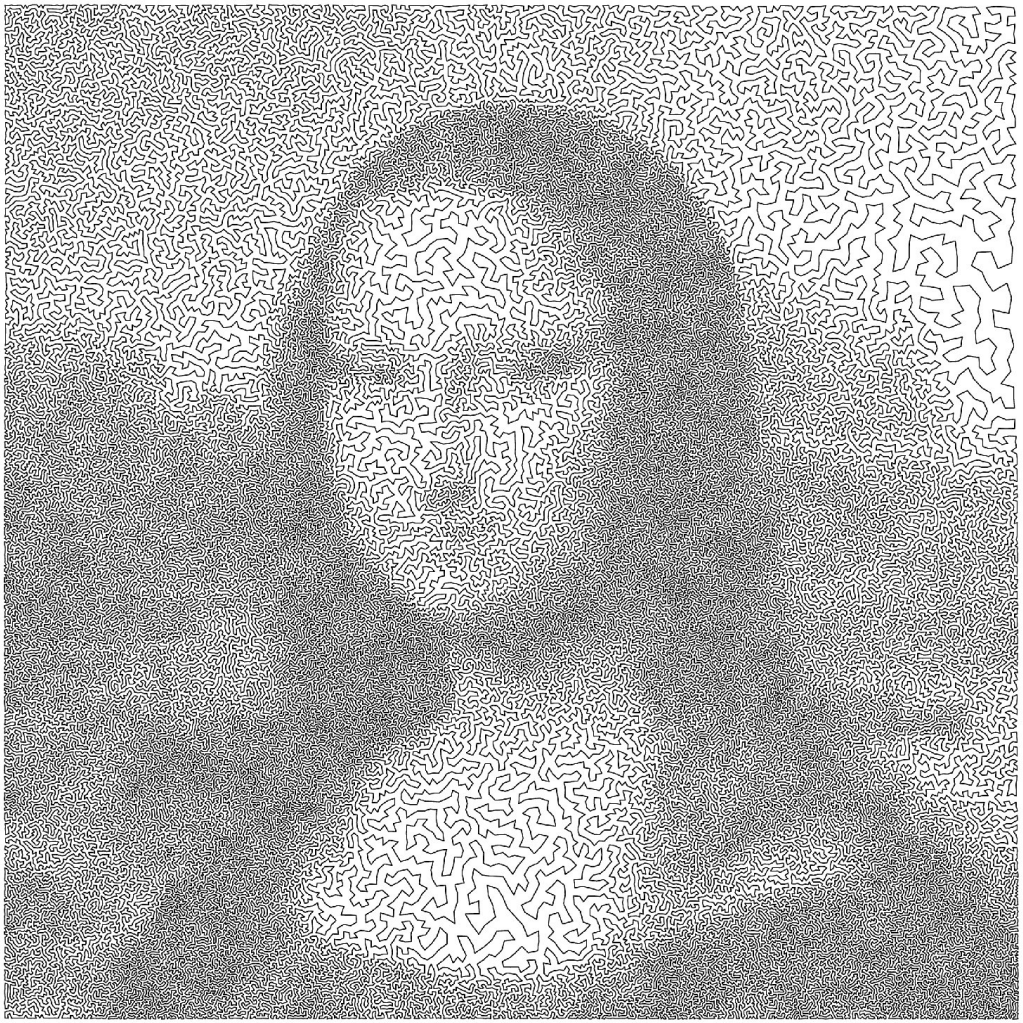
## Definition (TSP)

Given a complete graph $G = (V, E)$ and distances $d_e$ for all $e \in E$:

Find a Hamiltonian cycle (cycle containing all nodes, tour) of minimum length.



# Mona Lisa TSP Challenge



- 100,000 nodes
- Best solution:
  Tour: 5,757,191
  Bound: 5,757,025
  Gap: 166 (0.0029%)

$100 prize to the first person to find a tour shorter than 5,757,191.

http://www.tsp.gatech.edu/data/ml/monalisa.html

# Mixed Integer Program (MIP)

## Characteristics

Objective function:

- ▷ linear function

Feasible region:

- ▷ described by linear constraints

Variable domains:

- ▷ real or integer values

$$
\begin{aligned}
\min \quad & c^T x \\
s.t. \quad & Ax \leq b \\
& (x_I, x_C) \in \mathbb{Z}^I \times \mathbb{R}^C
\end{aligned}
$$

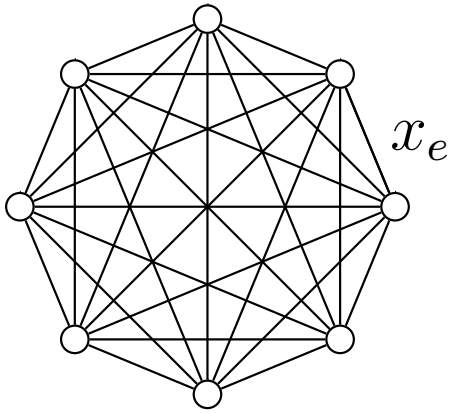
# TSP – Integer Programming Formulation

Given

- Complete graph $G = (V, E)$
- for each $e \in E$ a distance $d_e > 0$

Binary variables

- $x_e = 1$ if edge $e$ is used




$K_8$

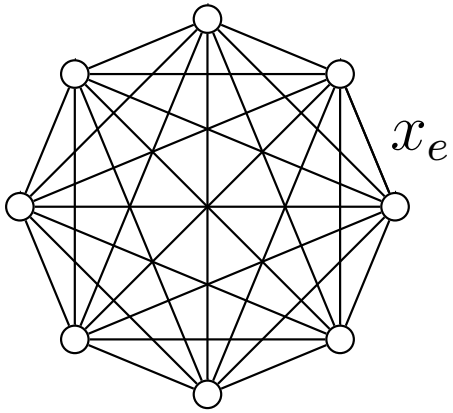
# TSP – Integer Programming Formulation

## Given

- Complete graph $G = (V, E)$
- for each $e \in E$ a distance $d_e > 0$

## Binary variables

- $x_e = 1$ if edge $e$ is used




$K_8$

$$
\begin{aligned}
\min \quad & \sum_{e \in E} d_e \, x_e & \\
\text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 2 & \forall v \in V \\
& \sum_{e \in \delta(S)} x_e \geq 2 & \forall S \subset V, S \neq \varnothing \\
& x_e \in \{0, 1\} & \forall e \in E
\end{aligned}
$$

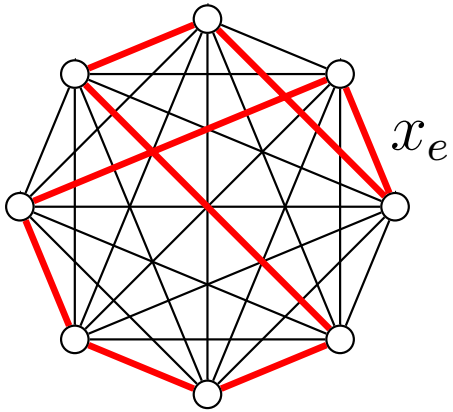
# TSP – Integer Programming Formulation

## Given

- Complete graph $G = (V, E)$
- for each $e \in E$ a distance $d_e > 0$

## Binary variables

- $x_e = 1$ if edge $e$ is used




$K_8$

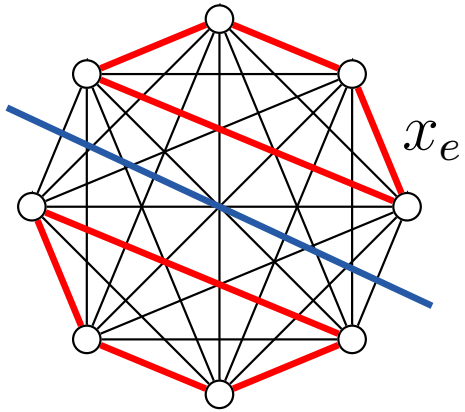
$$\min \sum_{e \in E} d_e \, x_e$$

| | | | |
|---|---|---|---|
| subject to | $\sum_{e \in \delta(v)} x_e = 2$ | $\forall v \in V$ | Degrees |
| | $\sum_{e \in \delta(S)} x_e \geq 2$ | $\forall S \subset V, S \neq \varnothing$ | |
| | $x_e \in \{0, 1\}$ | $\forall e \in E$ | |

# TSP – Integer Programming Formulation

## Given

- Complete graph $G = (V, E)$
- for each $e \in E$ a distance $d_e > 0$




$K_8$

## Binary variables

- $x_e = 1$ if edge $e$ is used

$$\min \quad \sum_{e \in E} d_e \, x_e$$

$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2 \qquad \forall v \in V$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall S \subset V, S \neq \varnothing$$

NoSubtours

$$x_e \in \{0, 1\} \qquad \forall e \in E$$

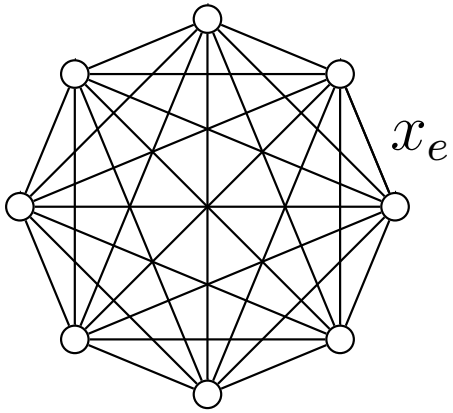
# TSP – Integer Programming Formulation

## Given

- Complete graph $G = (V, E)$
- for each $e \in E$ a distance $d_e > 0$

## Binary variables

- $x_e = 1$ if edge $e$ is used




$K_8$

$$\min \sum_{e \in E} d_e \, x_e \qquad \text{Distance}$$

$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2 \qquad \forall v \in V$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall S \subset V, S \neq \varnothing$$

$$x_e \in \{0, 1\} \qquad \forall e \in E$$

# Constraint Program (CP)

## Characteristics

Objective function:

- arbitrary function

Feasible region:

- described by arbitrary constraints

Variable domains:

- discrete values

$$\begin{aligned} \min \quad & c(x) \\ s.t. \quad & x \in F \\ & x \in \mathbb{Z}^n \end{aligned}$$

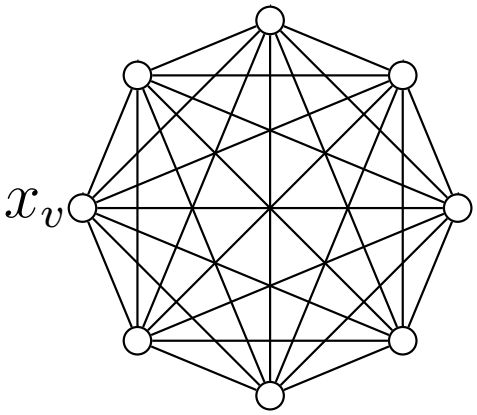
# TSP – Constraint Programming Formulation

Given

- Complete graph $G = (V, E)$
- for each $e \in E$ a distance $d_e > 0$

Integer variables

- $x_v$ position of $v \in V$ in tour




$K_8$

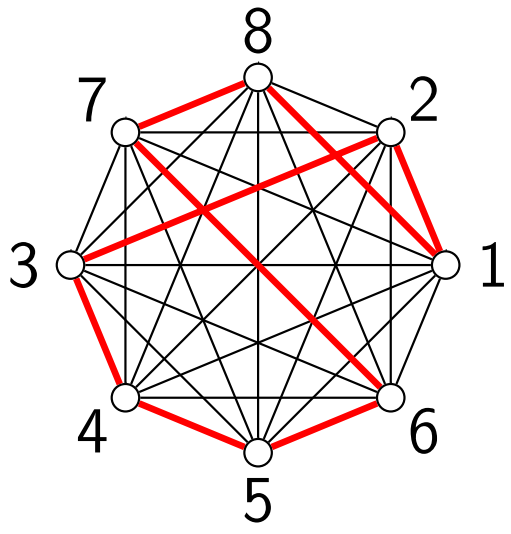
# TSP – Constraint Programming Formulation

Given

- Complete graph $G = (V, E)$
- for each $e \in E$ a distance $d_e > 0$

Integer variables

- $x_v$ position of $v \in V$ in tour




$$\begin{aligned} \min \quad & \text{length}(x_1, \ldots, x_n) \\ \text{subject to} \quad & \text{alldifferent}(x_1, \ldots, x_n) \\ & x_v \in \{1, \ldots, n\} \qquad \forall v \in V \end{aligned}$$

# Traveling Salesman Problem

- Given a complete graph $G = (V, E)$ and distances $d_e$.
- Find a Hamiltonian cycle (cylce containing all nodes, tour) with minimum length.

## MIP Formulation

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

## CP Formulation

$$
\begin{array}{ll}
\min & \text{length}(x_1, \ldots, x_n) \\
s.t. & \text{alldifferent}(x_1, \ldots, x_n) \\
 & x_v \in \{1, \ldots, n\}
\end{array}
$$

# Constraint Integer Program (CIP)

## Characteristics

Objective function:

- linear function

Feasible region:

- described by arbitrary constraints
- after fixing all integer vars: CIP becomes an linear program (LP)

Variable domains:

- real or integer values

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & x \in F \\ & (x_I, x_C) \in \mathbb{Z}^I \times \mathbb{R}^C \end{aligned}$$

# Constraint Integer Program (CIP)

## Characteristics

Objective function:

- ▷ linear function

Feasible region:

- ▷ described by arbitrary constraints
- ▷ after fixing all integer vars: CIP becomes an linear program (LP)

Variable domains:

- ▷ real or integer values

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

(CIP formulation of TSP)

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

# Computational Results

- TSPLIB

http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

- 108 Instances
- Time limit 1800 seconds
- Memory limit 6 GB
- Flow based IP formulation versus CIP

| | CPLEX | SCIP |
|---|---|---|
| #Solved | 33 | 64 |
| Total Time [h] | 40 | 34 |
| Geom. Time [s] | 475.0 | 164.7 |

# Computational Results

- TSPLIB

http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

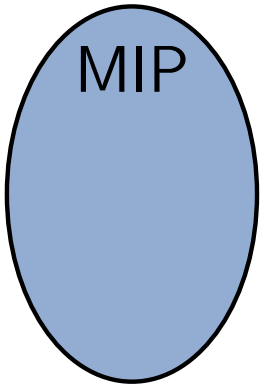
- 108 Instances
- Time limit 1800 seconds
- Memory limit 6 GB
- Flow based IP formulation versus CIP

| | CPLEX | SCIP | Concorde |
|---|---|---|---|
| #Solved | 33 | 64 | 108 |
| Total Time [h] | 40 | 34 | <2 |
| Geom. Time [s] | 475.0 | 164.7 | 3.0 |

http://www.tsp.gatech.edu/concorde.html

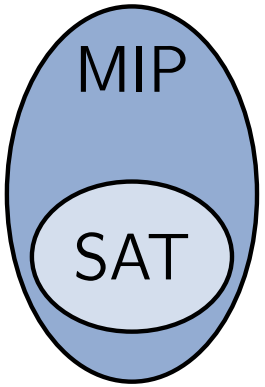
# Constraint Integer Programming

- Mixed Integer Programs



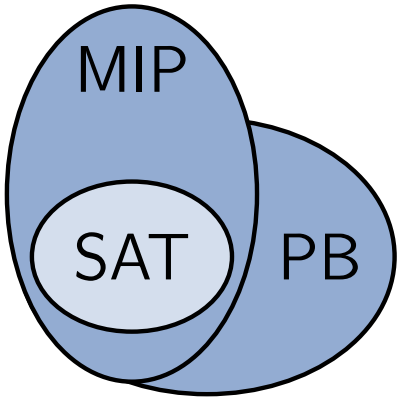
# Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability



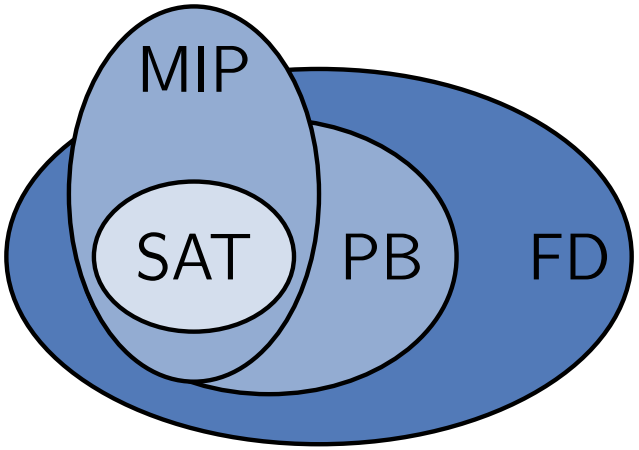
# Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean



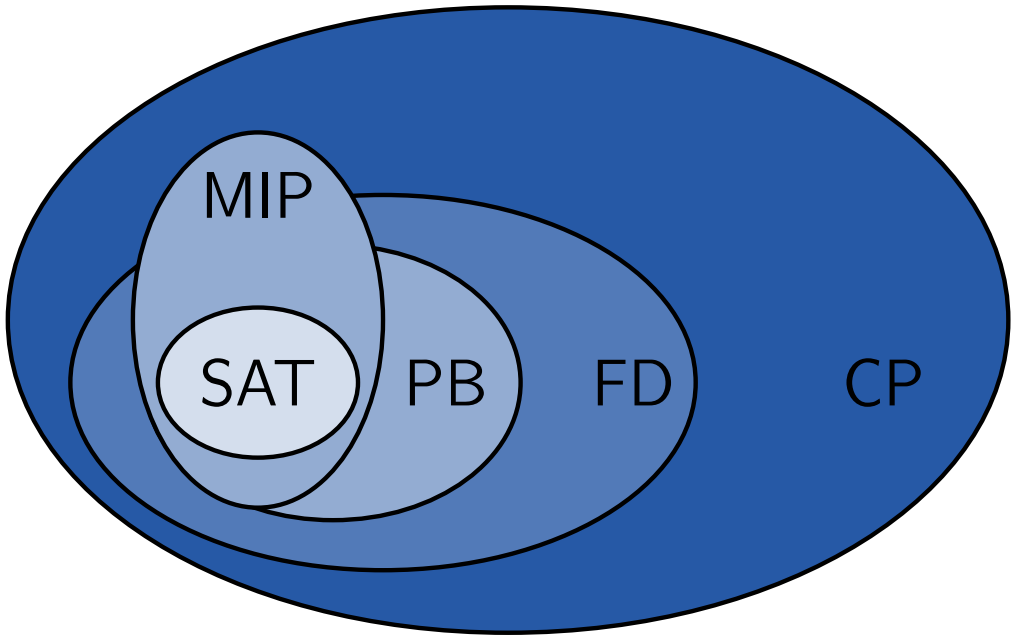
# Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean
- Finite Domain



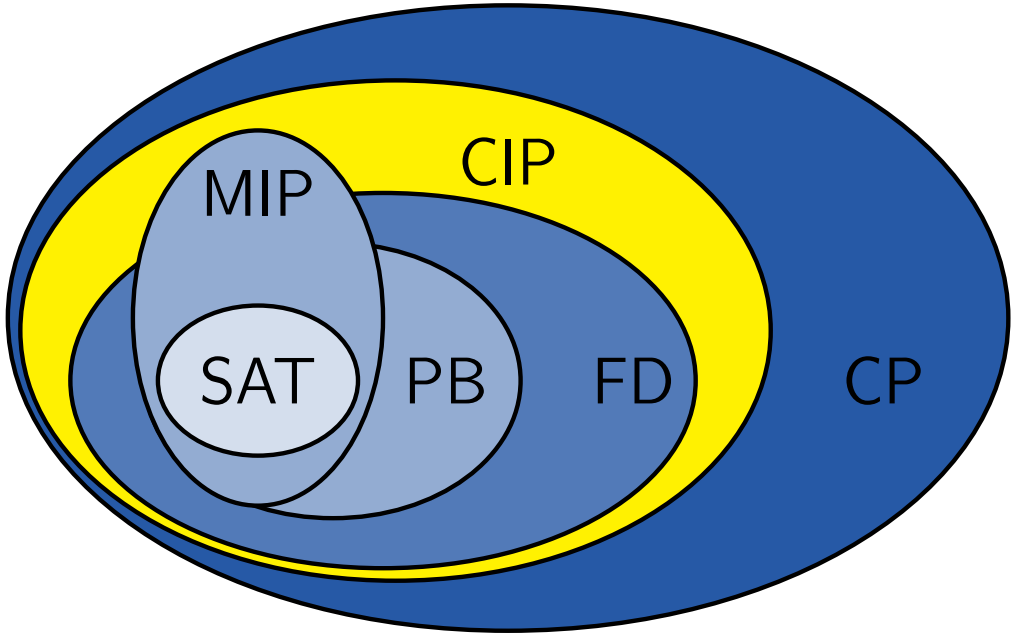
# Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean
- Finite Domain
- Constraint Programming



# Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean
- Finite Domain
- Constraint Programming
- Constraint Integer Programming



# Outline

## SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

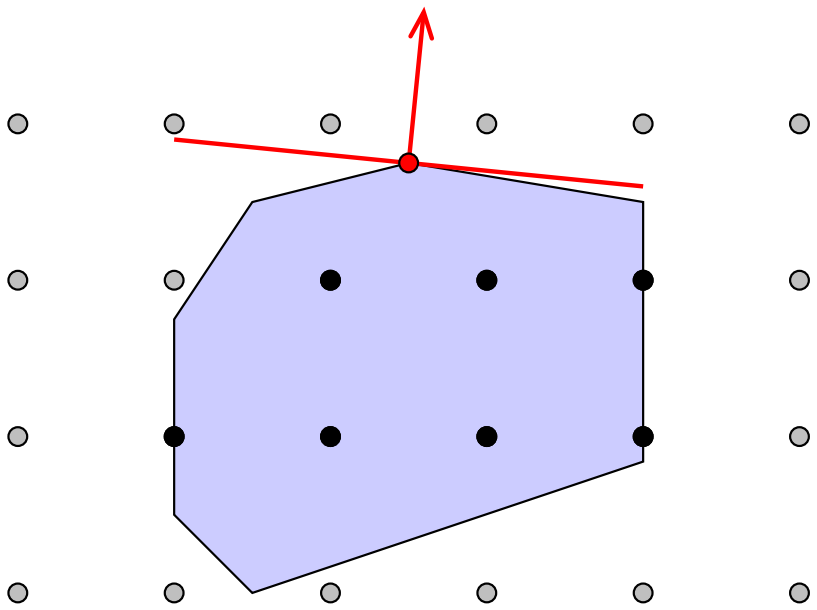
Solving Constraint Integer Programs

Summary and Outlook

# MIP Solving Techniques

## Combination of ...

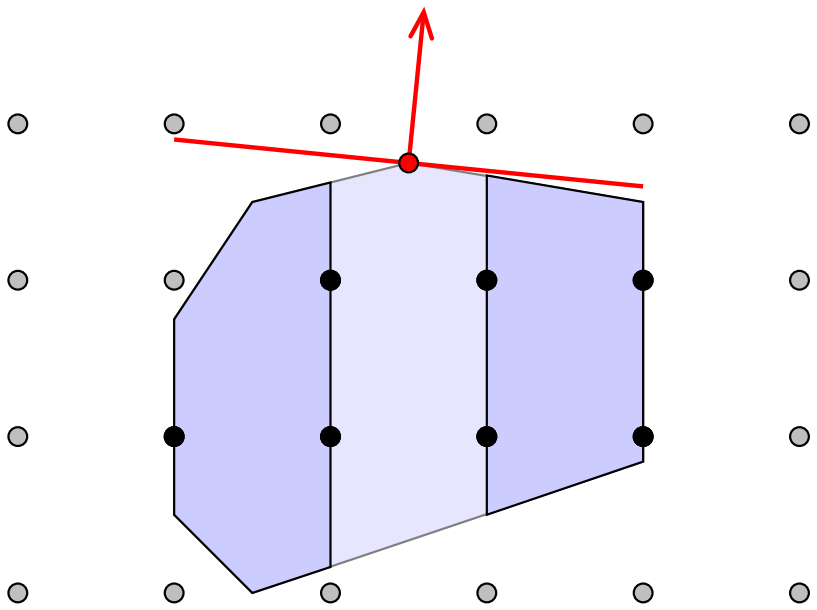
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

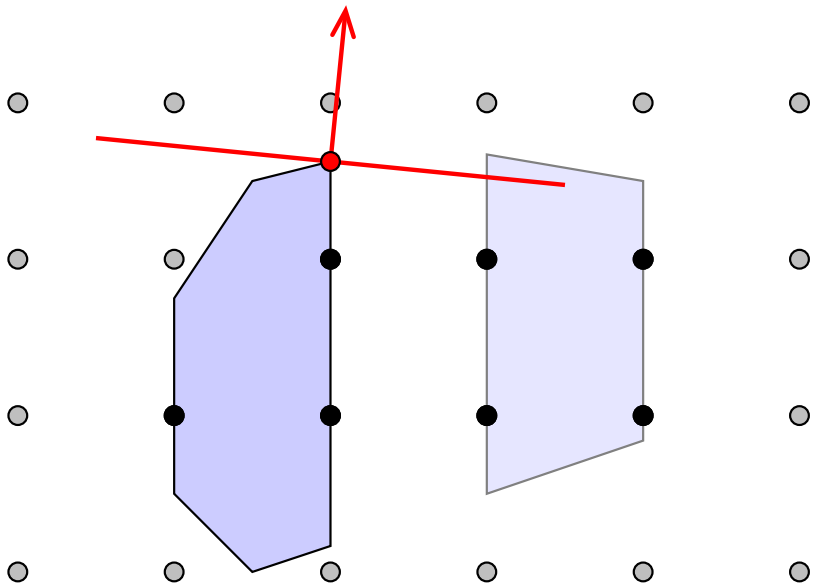
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

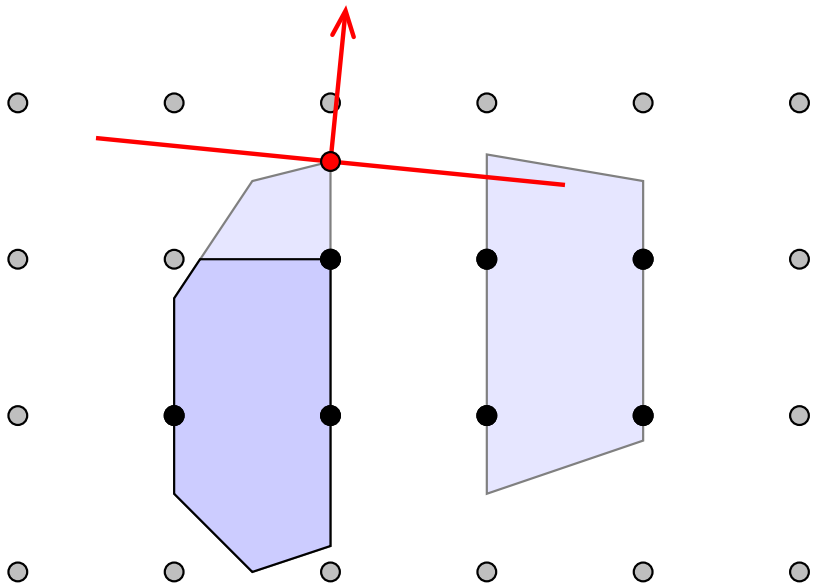
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

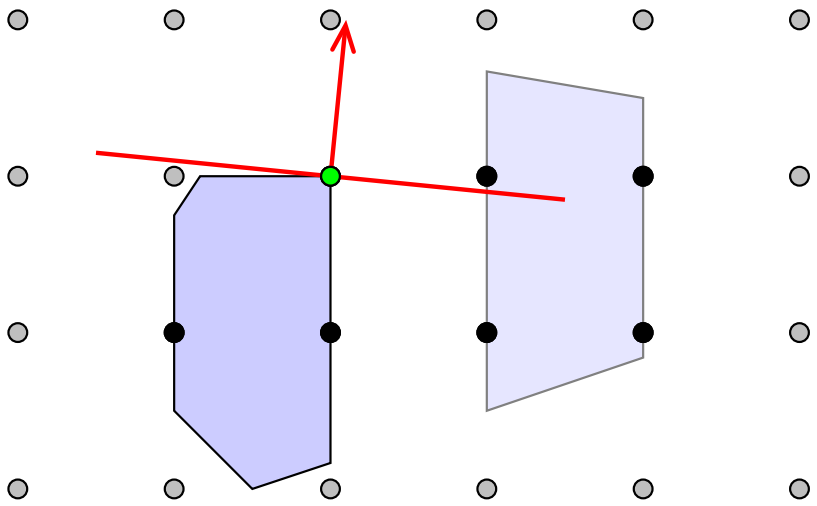
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

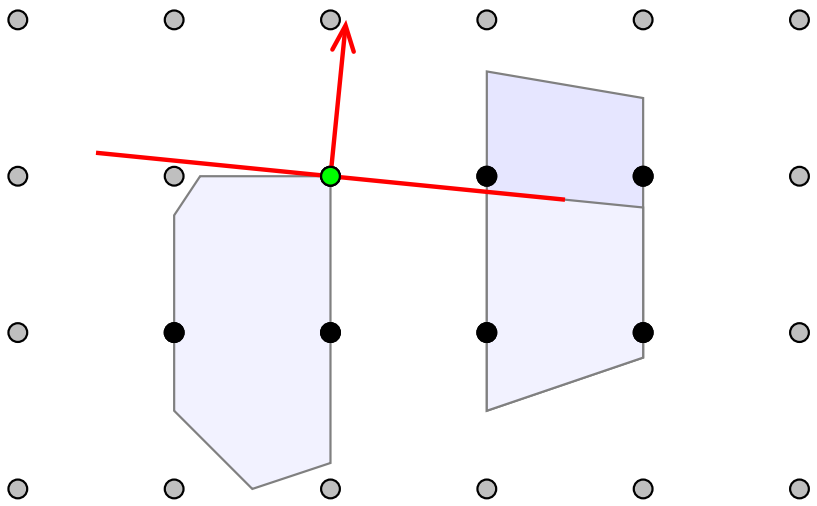
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

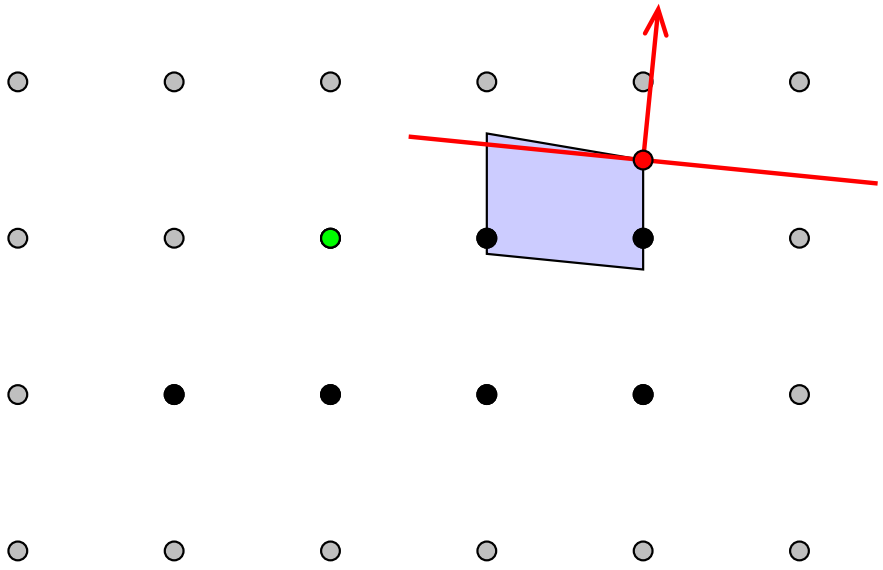
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

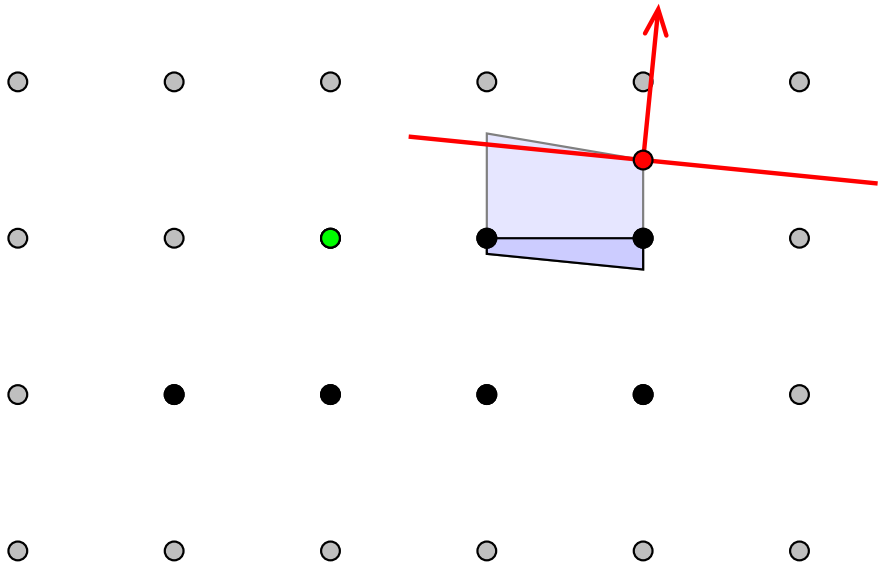
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

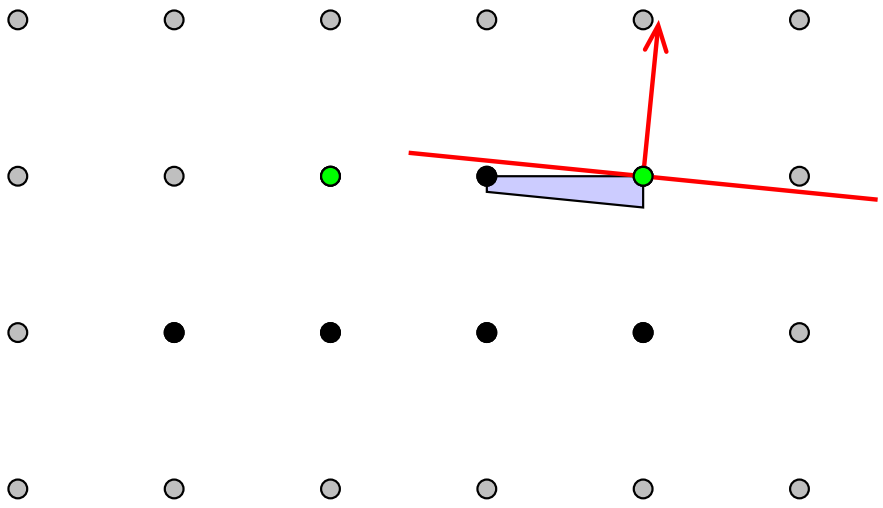
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

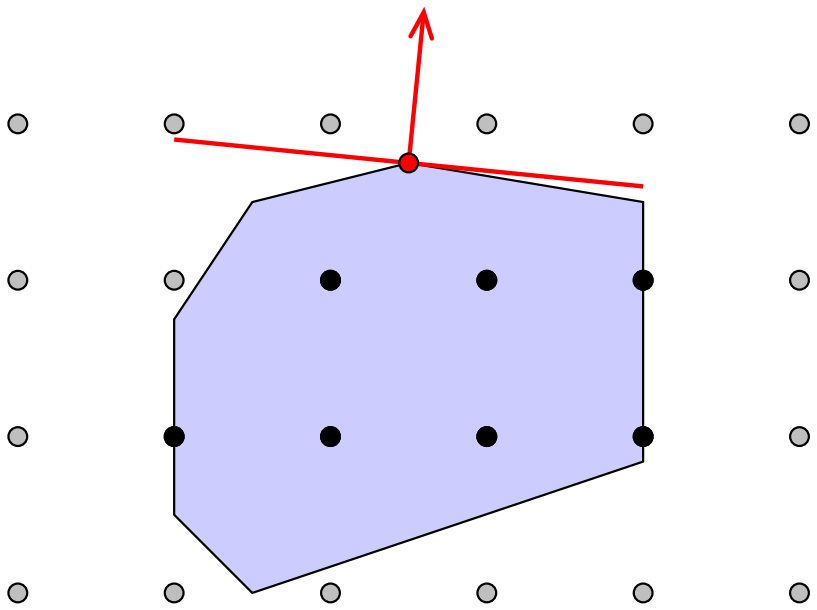
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

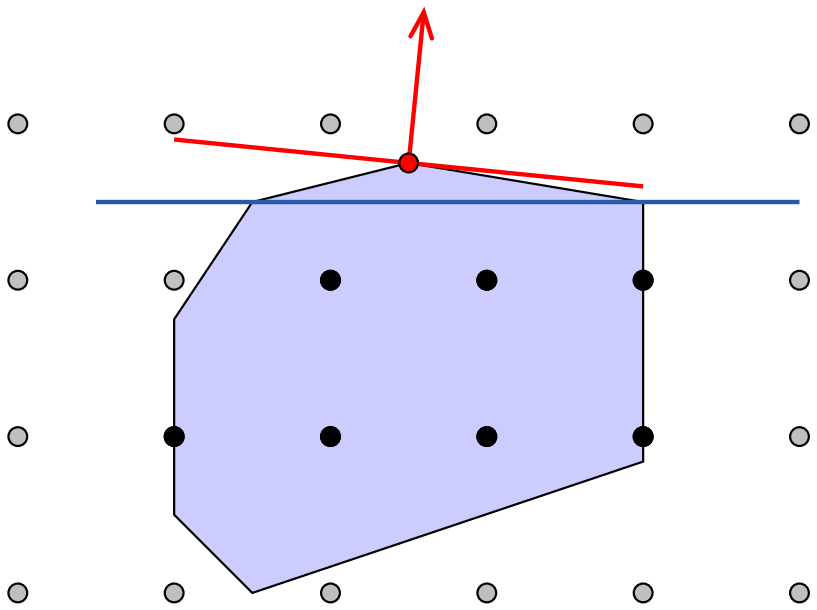
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

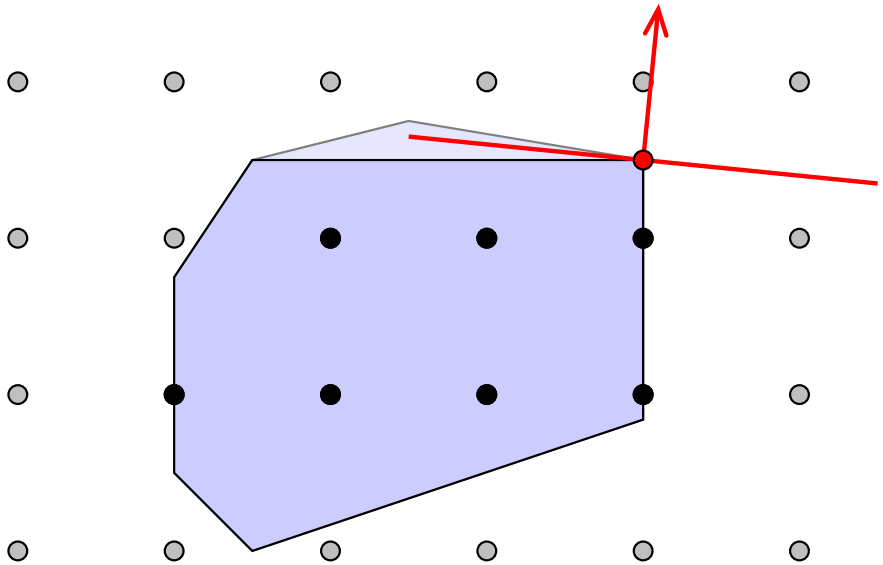
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

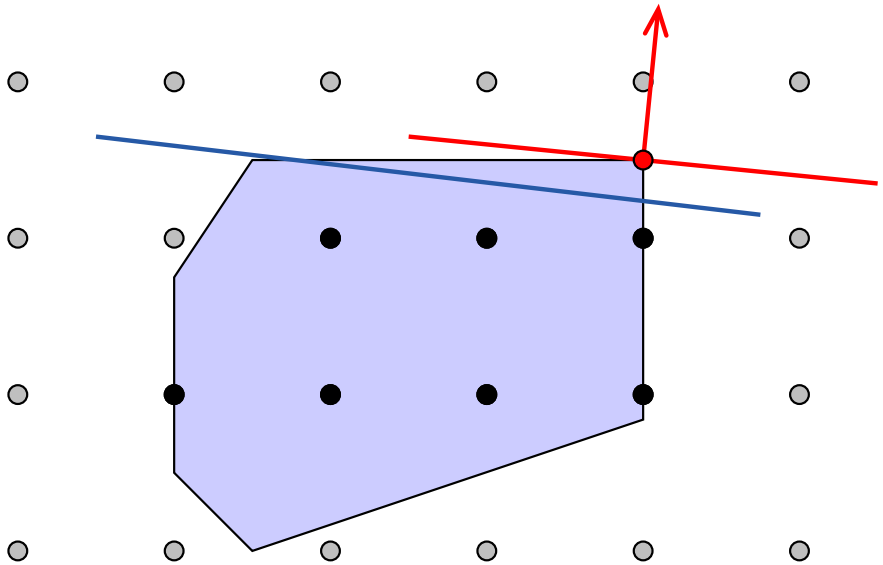
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

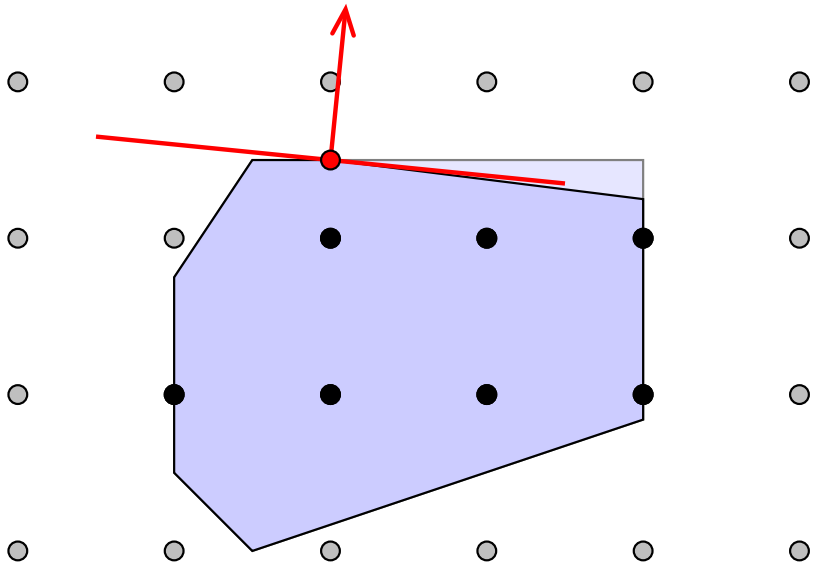
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

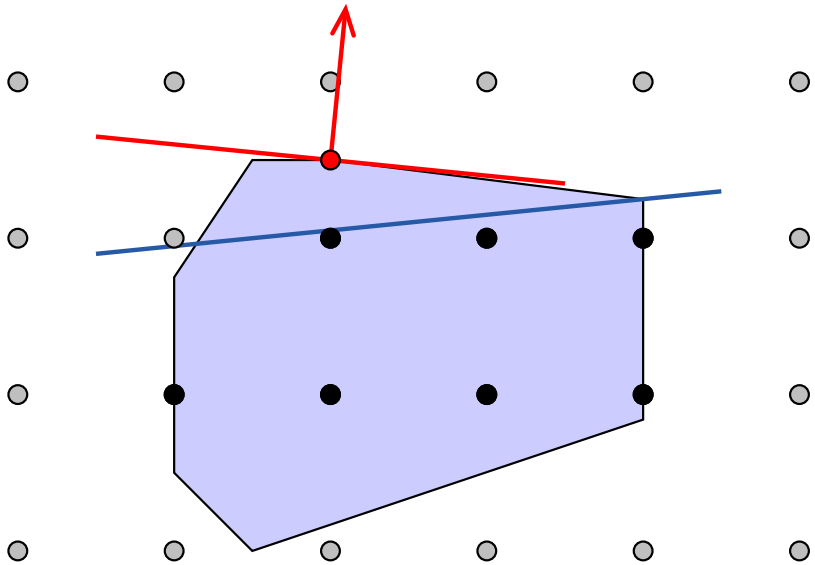
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

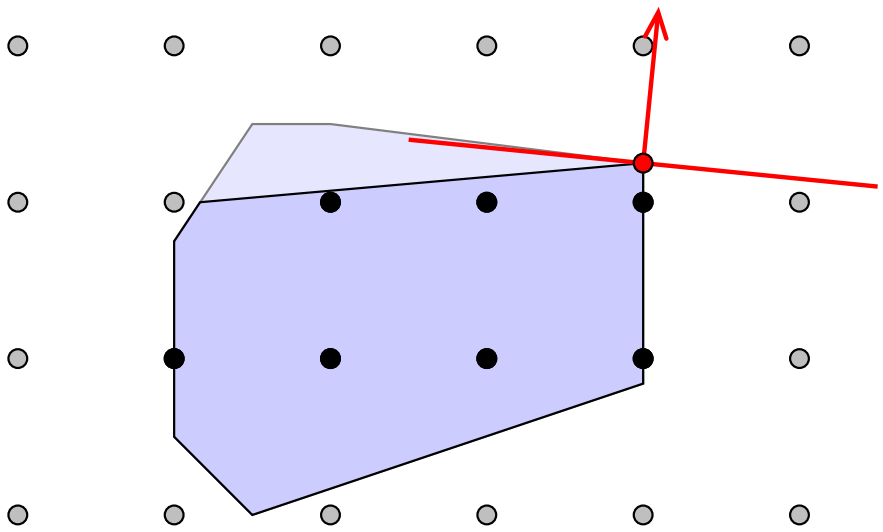
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

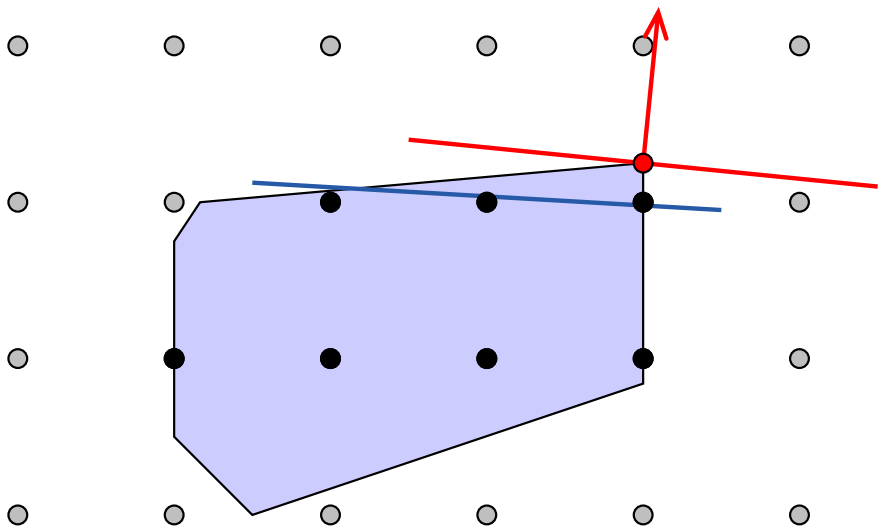
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

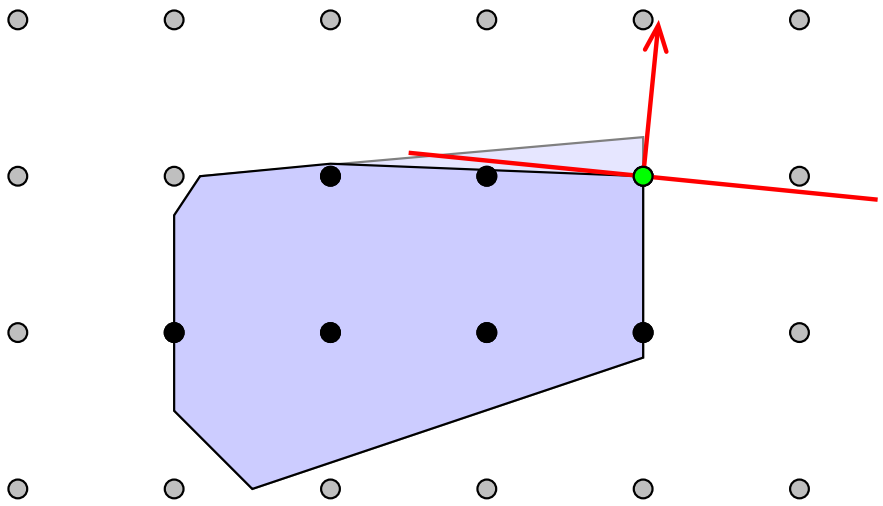
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

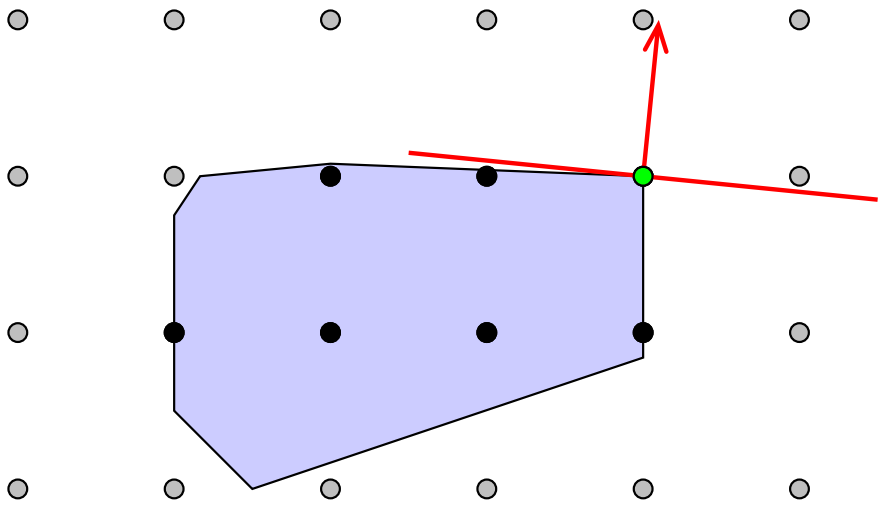
- LP relaxation
- branch-and-bound
- cutting planes



# MIP Solving Techniques

## Combination of ...

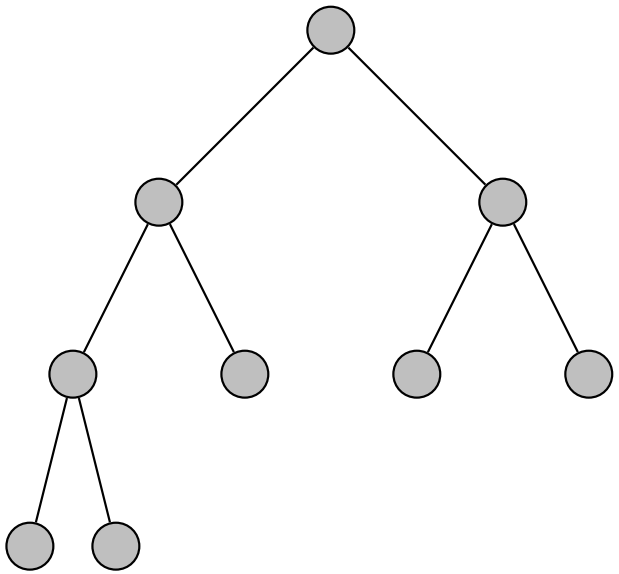
- LP relaxation
- branch-and-bound
- cutting planes



# CP Solving Techniques

## Combination of ...

- branch-and-bound
- domain propagation



# CP Solving Techniques

## Combination of ...

- branch-and-bound
- domain propagation

Analyze ...

- constraints of subproblem
- current domains of variables

in order to infer domain reductions.
($\longrightarrow$ restrict the search space)

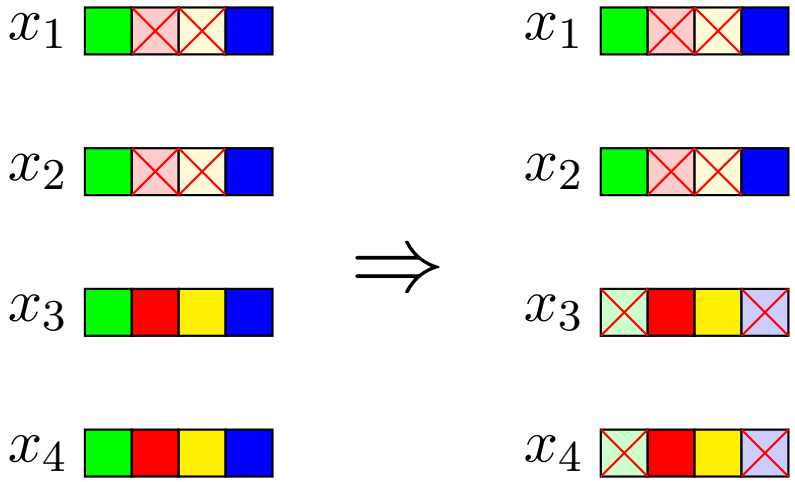
# CP Solving Techniques

## Combination of ...

- branch-and-bound
- domain propagation

Alldifferent constraint:

$\{x_1, x_2, x_3, x_4\} \in \{1, 2, 3, 4\}$
pairwise different.



# CIP Solving Techniques

## MIP

- LP relaxation
- Cutting planes

## MIP, CP, and SAT

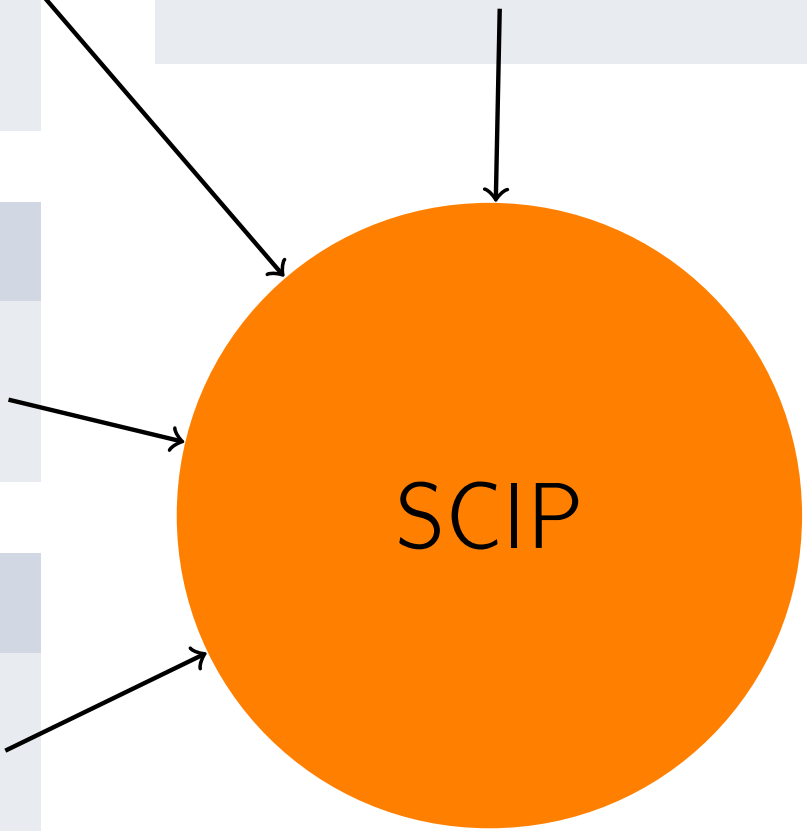
- Branch-and-bound

## CP

- Domain propagation

## SAT

- Conflict analysis
- Periodic restarts



# Outline

## SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

Solving Constraint Integer Programs

Summary and Outlook

# What is SCIP?

SCIP is a framework for Constraint Integer Programming oriented towards the needs of Mathematical Programming experts who want to have total control of the solution process and access detailed information down to the guts of the solver.

- Framework to solve constraint integer programs
- Branch-and-bound framework
- Branch-and-cut framework
- Branch-and-propagate framework
- Branch-and-price framework
- Black box MIP solver

# History of SCIP

| | |
|---|---|
| 1998 | SIP – Solving Integer Programs (Alexander Martin) |
| 10/2002 | Start of SCIP development (Tobias Achterberg) |
| 02/2003 | First version to solve MIPs |
| 03/2004 | Knapsack cover separator (Kati Wolter) |
| 01/2005 | Objective feasibility pump (Timo Berthold) |
| 09/2005 | First public version 0.80 |
| 09/2006 | Version 0.90 |
| 07/2007 | Tobias Achterberg finished his PhD. |
| 09/2007 | Version 1.00 |
| 01/2008 | Solution counting (Stefan Heinz) |
| 09/2008 | Version 1.1.0 |
| 10/2008 | Nonlinear support (Stefan Vigerske) |
| 09/2009 | Version 1.2.0 |

# SCIP Developers

- Tobias Achterberg (IBM)
- Marc Pfetsch (TU Braunschweig)
- Thorsten Koch

- Timo Berthold
- Stefan Heinz
- Stefan Vigerske
- Kati Wolter

- Gerald Gamrath
- Robert Waniek
- Michael Winkler

# SCIP Contributers

- Andreas Bley (TU Berlin) — VRP example
- Daniel Espinoza (U. Chile) — Interface to QSopt
- John Forrest (IBM) — Interface to CLP
- Thorsten Gellermann (TU Darmstadt) — Generic NLP interface
- Ambros Gleixner (ZIB) — GAMS Reader, SoPlex support
- Bo Jensen (Mosek) — Interface to MOSEK
- Manuel Kutschka (RWTH Aachen) — Separator for 0,1/2-cuts
- Michael Perregaard (FICO/Xpress) — Interface to FICO/Xpress
- Christian Raack (ZIB) — Separator for MCF cuts
- Jörg Rambau (U. Bayreuth) — SamplePricer example
- Cornelius Schwarz (U. Bayreuth) — Queens example
- Andreas Tuchscherer (ZIB) — SamplePricer example

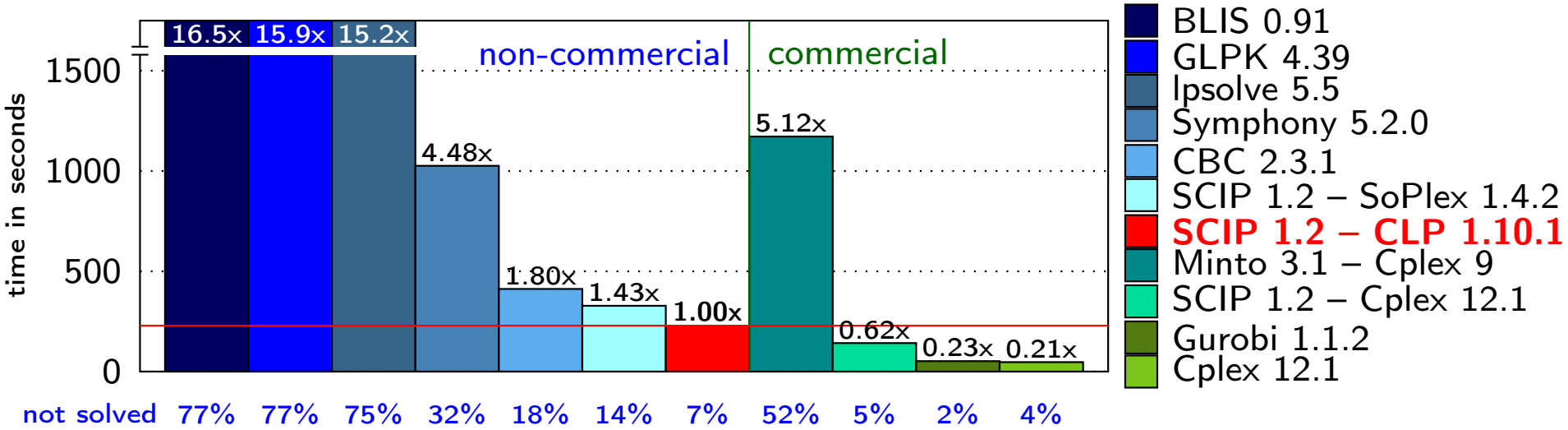
# SCIP Contributers

- Andreas Bley (TU Berlin) — VRP example
- Daniel Espinoza (U. Chile) — Interface to QSopt
- John Forrest (IBM) — Interface to CLP
- Thorsten Gellermann (TU Darmstadt) — Generic NLP interface
- Ambros Gleixner (ZIB) — GAMS Reader, SoPlex support
- Bo Jensen (Mosek) — Interface to MOSEK
- Manuel Kutschka (RWTH Aachen) — Separator for 0,1/2-cuts
- Michael Perregaard (FICO/Xpress) — Interface to FICO/Xpress
- Christian Raack (ZIB) — Separator for MCF cuts
- Jörg Rambau (U. Bayreuth) — SamplePricer example
- Cornelius Schwarz (U. Bayreuth) — Queens example
- Andreas Tuchscherer (ZIB) — SamplePricer example

YOU?

# Performance of SCIP

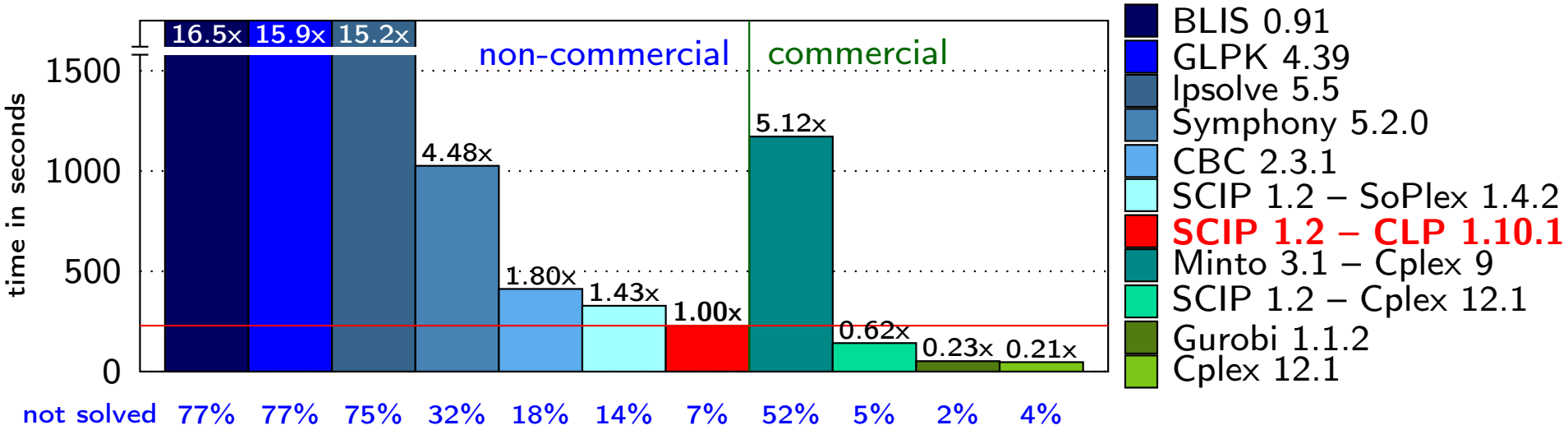
- One of the fastest noncommercial MIP solver




results taken from H. Mittelmann (09/17/2009)

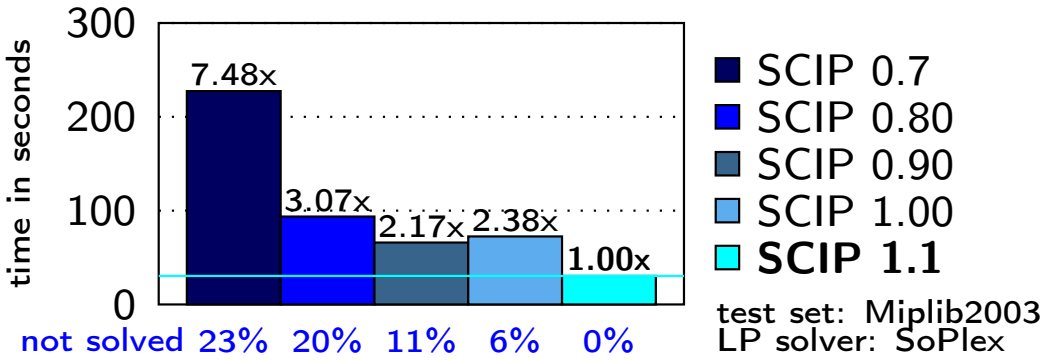
# Performance of SCIP

- One of the fastest noncommercial MIP solver




results taken from H. Mittelmann (09/17/2009)

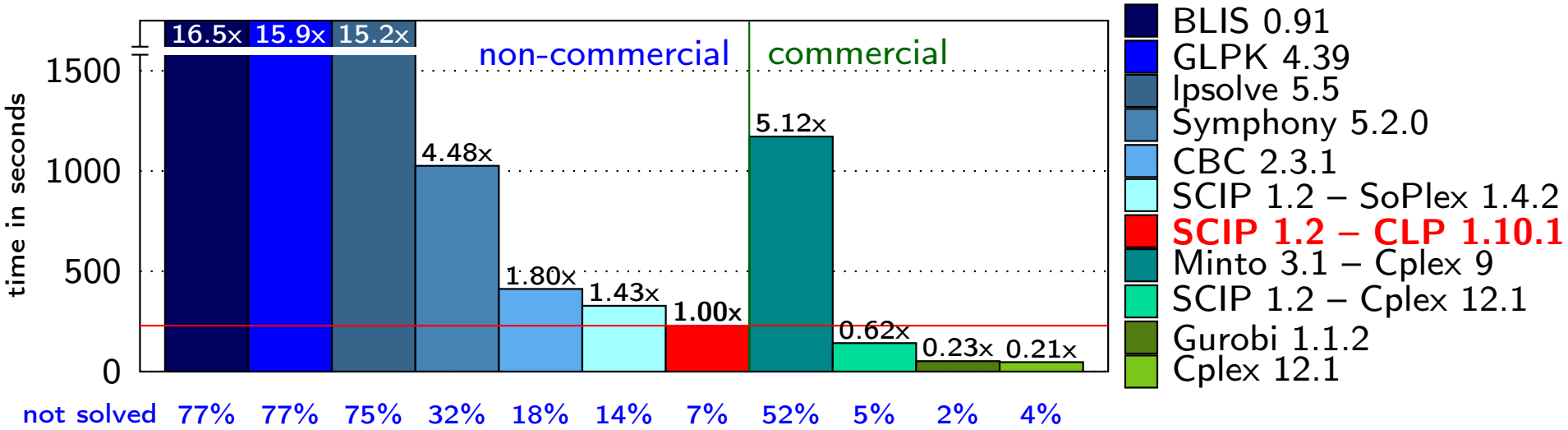
- Performance development




test set: Miplib2003
LP solver: SoPlex

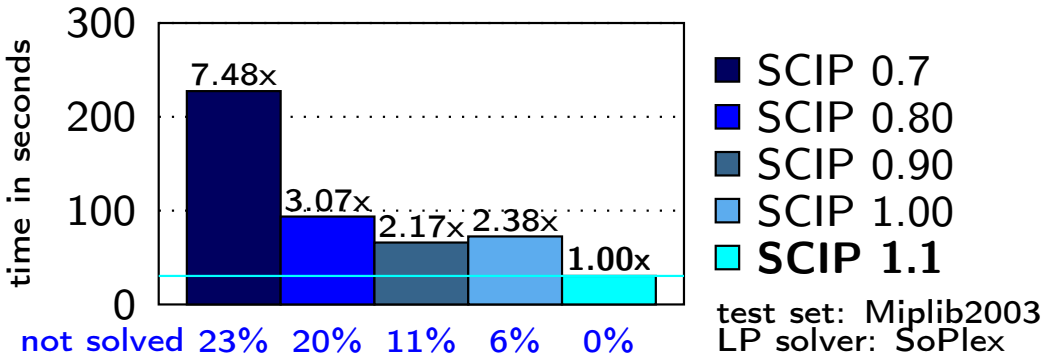
# Performance of SCIP

- One of the fastest noncommercial MIP solver




results taken from H. Mittelmann (09/17/2009)

- Performance development




- Winner of the "Pseudo-Boolean Evaluation 2009"

  http://www.cril.univ-artois.fr/PB09/

# SCIP Facts

- Approx. 275.640 lines of C code
  → 18% documentation
  → 20% assertions
- 7 examples illustrating the use of SCIP
- HowTos: each plugin type, debugging, automatic testing, ...
- C++ wrapper classes
- 7 interfaces to external linear programming solvers
  → CLP, CPLEX, Gurobi, Mosek, QSopt, SoPlex, XPRESS
- 6 different input formats
  → cnf, flatzinc, lp, mps, opb (pseudo-boolean), zimpl
- 876 parameters to play with
- Part of the ZIB Optimization Suite http://zibopt.zib.de

# ZIP Optimization Suite = SCIP + SoPlex + ZIMPL

Tool for generating and solving constraint integer programs

## ZIMPL

- A mixed integer programming modeling language
- Easily generating linear programs and mixed integer programs

## SCIP

- A mixed integer programming solver and constraint programming
- ZIMPL models can directly be loaded into SCIP and solved

## SoPlex

- A linear programming (LP) solver
- Solution process SCIP may use SoPlex as underlying LP solver

All three tools are available in source code and free for academic use

# Outline

## SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

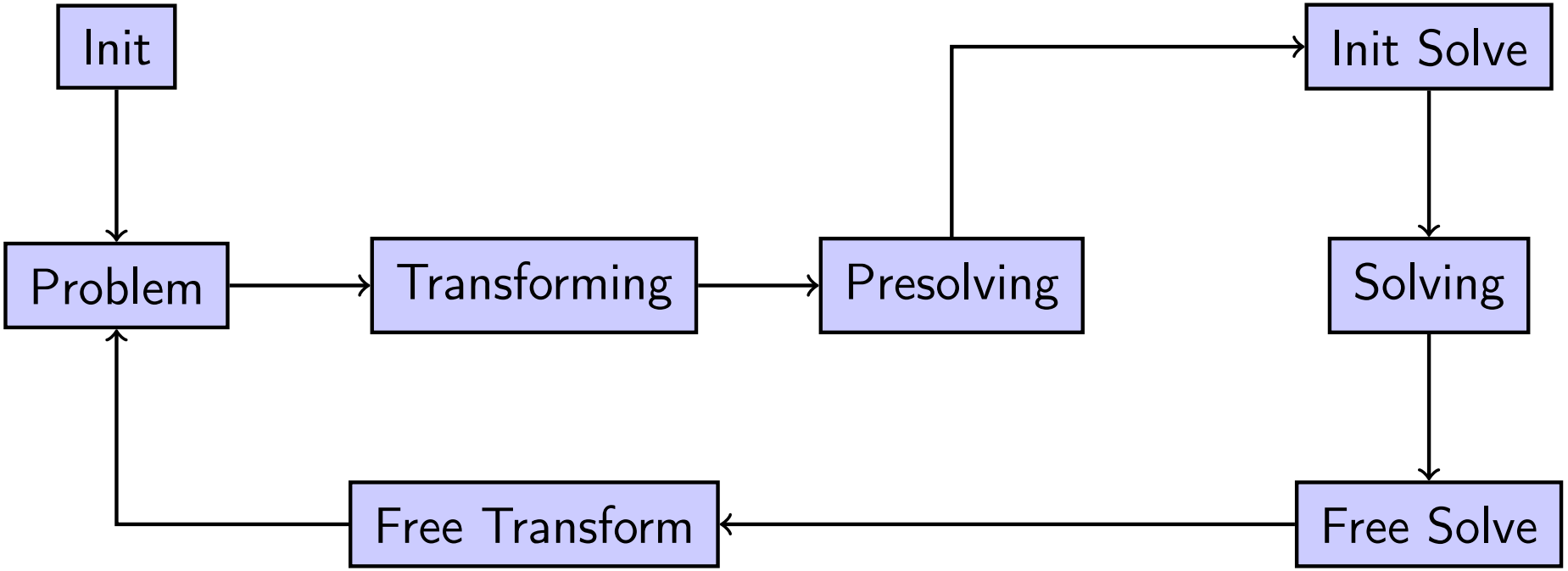
Solving Constraint Integer Programs

Summary and Outlook

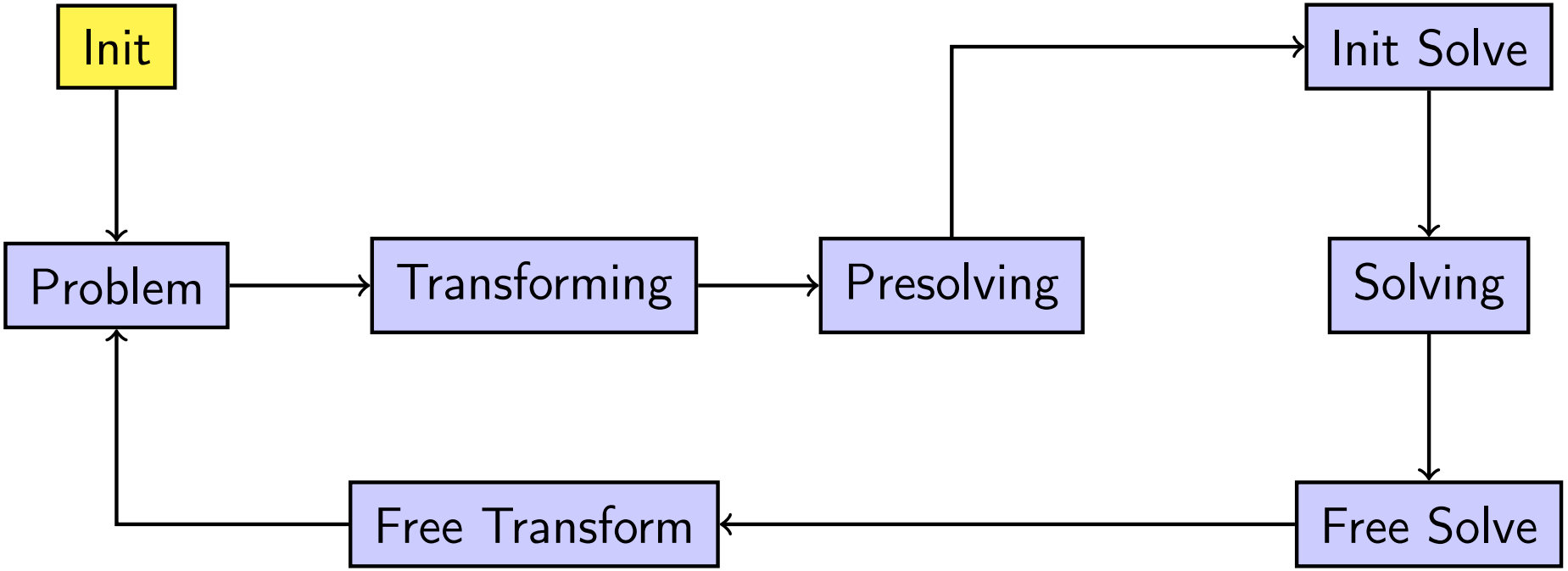
# Important Aspects

- SCIP is constraint based
  - Advantage: flexibility
  - Disadvantage: limited global view
- A constraint knows its variables, but a variable does not know the constraints it appears in.
- A single constraint may represent a whole set of inequalities, not only a single one.
- From the constraint programming perspective:
  - LP-relaxation is only an add-on!
  - Many constraints do not separate inequalities at all.

# Operational Stages



# Operational Stages




- Basic data structures are allocated and initialized.
- User includes required plugins (or just takes default plugins).

# SCIP Interface

Main SCIP interface: plugins.

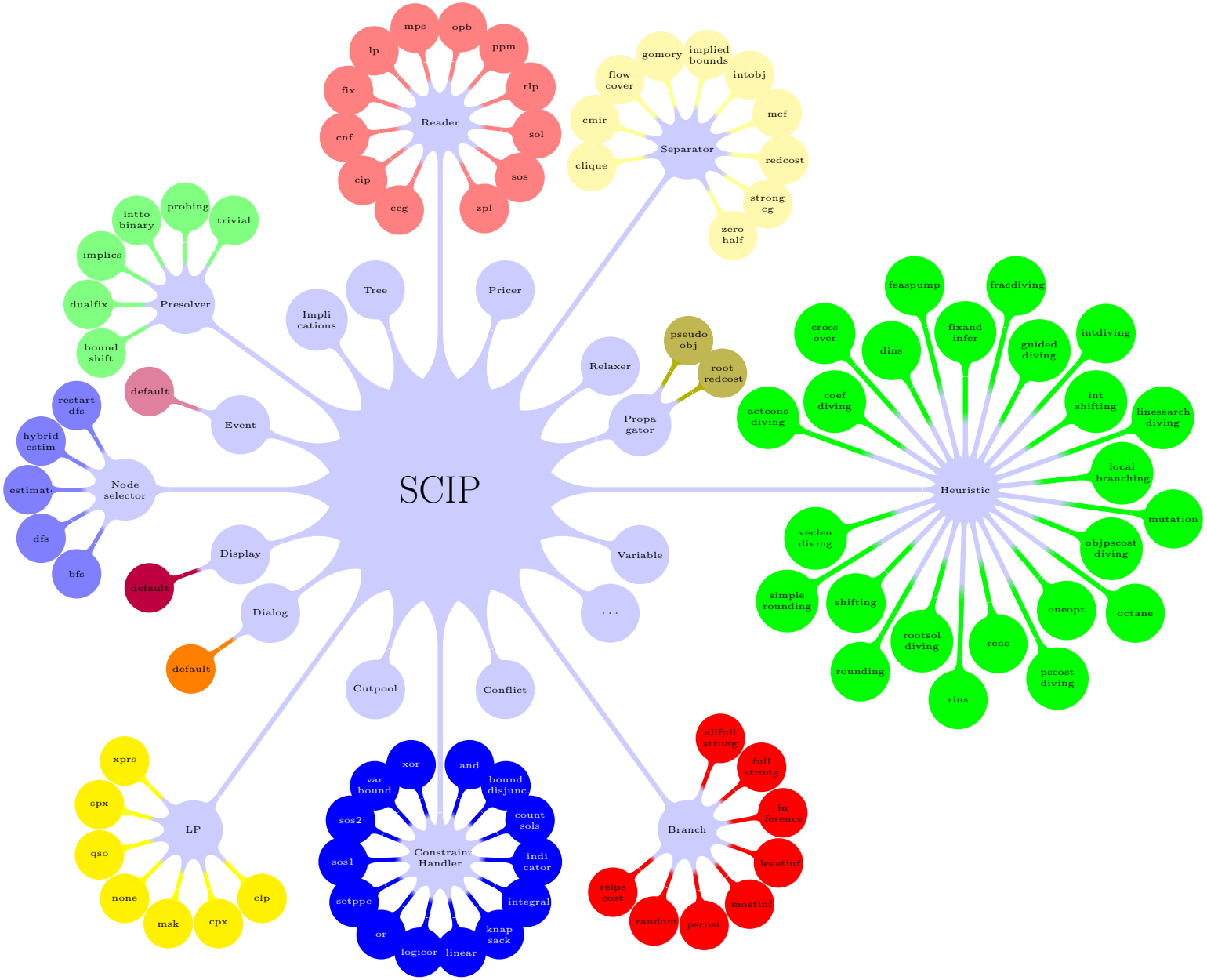
- Perform all problem specific actions.
- Each step calls user defined plugins.
- SCIP knows of plugins through "include" functions.
- Plugins may have private data.
- User defined callback functions (virtual functions in C++-interface).
- Yields modular structure.
- The MIP solver is realized through plugins.

Everything SCIP knows, it knows through plugins.

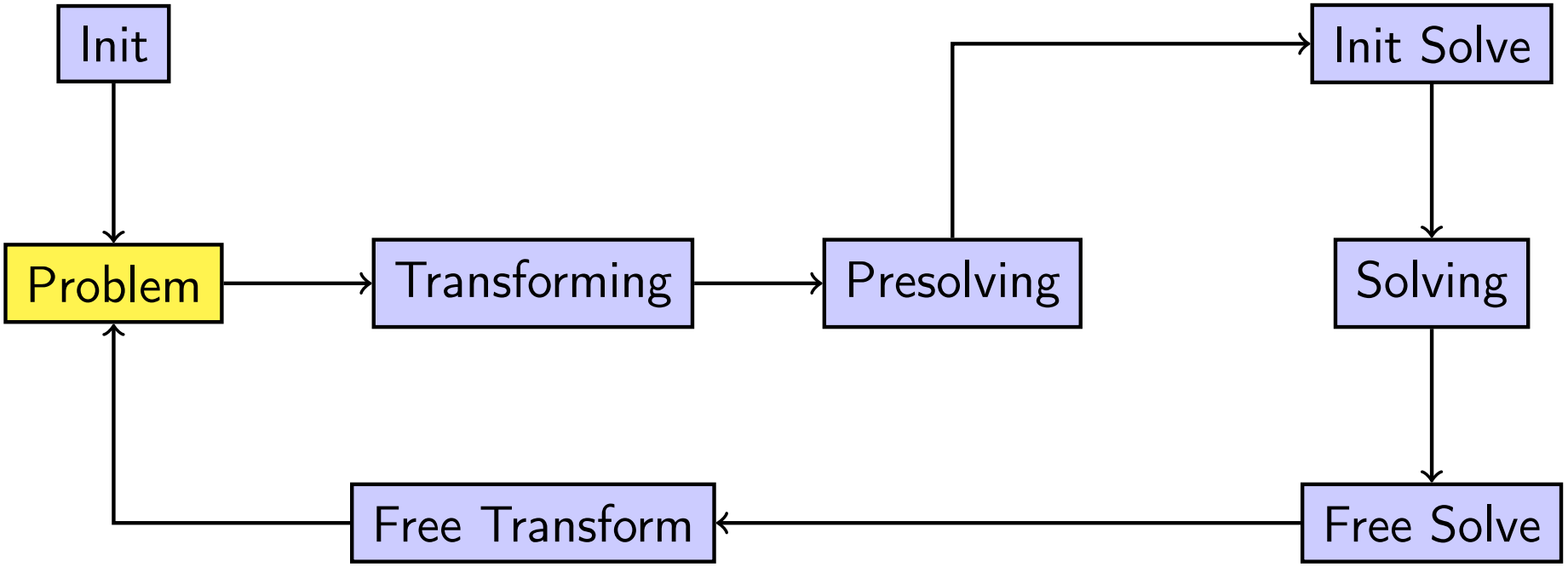
# Types of Plugins

- Constraint handler: assures feasibility, strengthens formulation
- Separator: adds cuts, improves dual bound
- Pricer: allows dynamic generation of variables
- Heuristic: searches solutions, improves primal bound
- Branching rule: how to divide the problem?
- Node selection: which subproblem should be regarded next?
- Presolver: simplifies the problem in advance, strengthens structure
- Propagator: simplifies problem, improves dual bound locally
- Reader: reads problems from different formats
- Event handler: catches events (e.g., bound changes, new solutions)
- Display: allows modification of output
- ...

# SCIP Structures

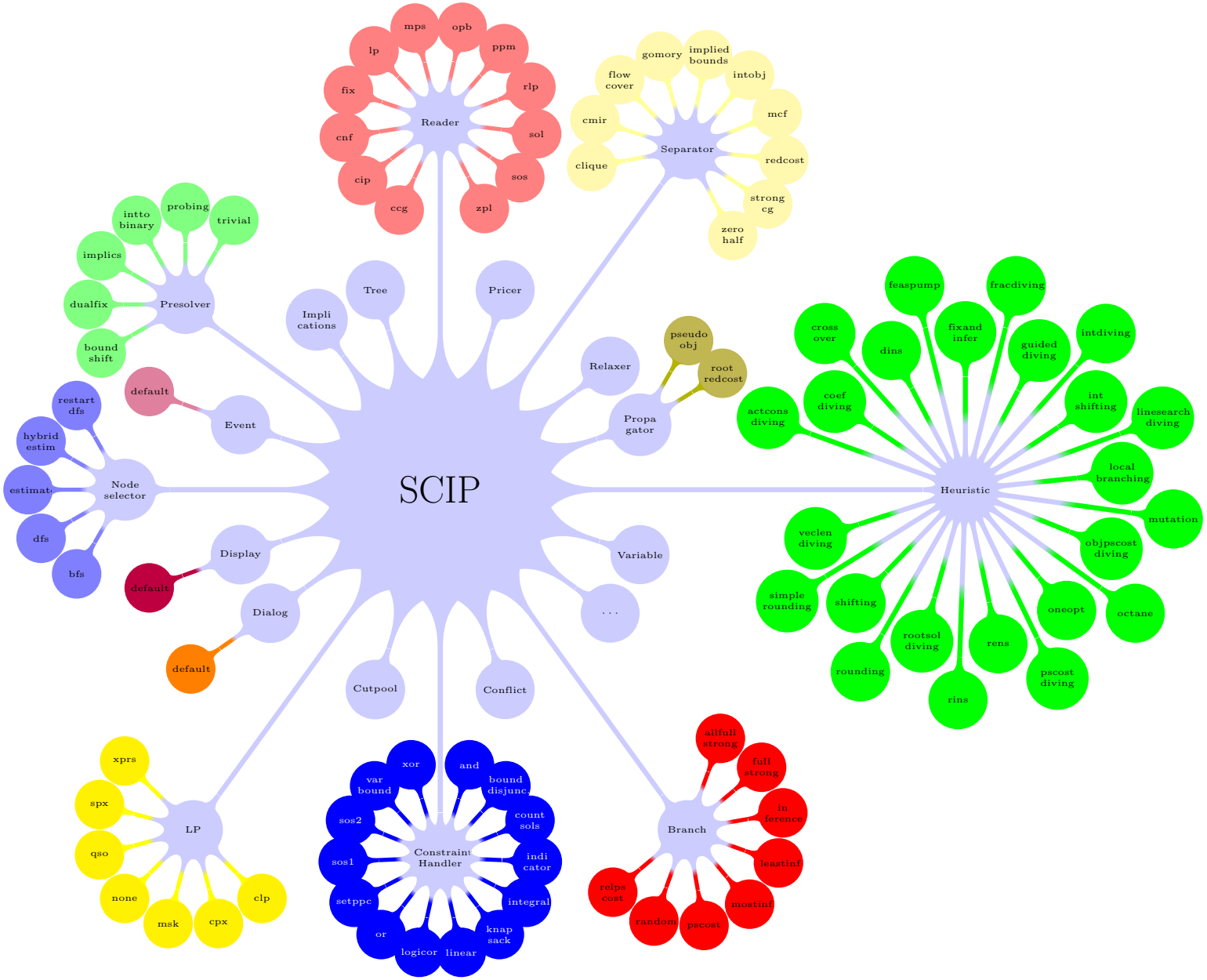


# Problem Specification

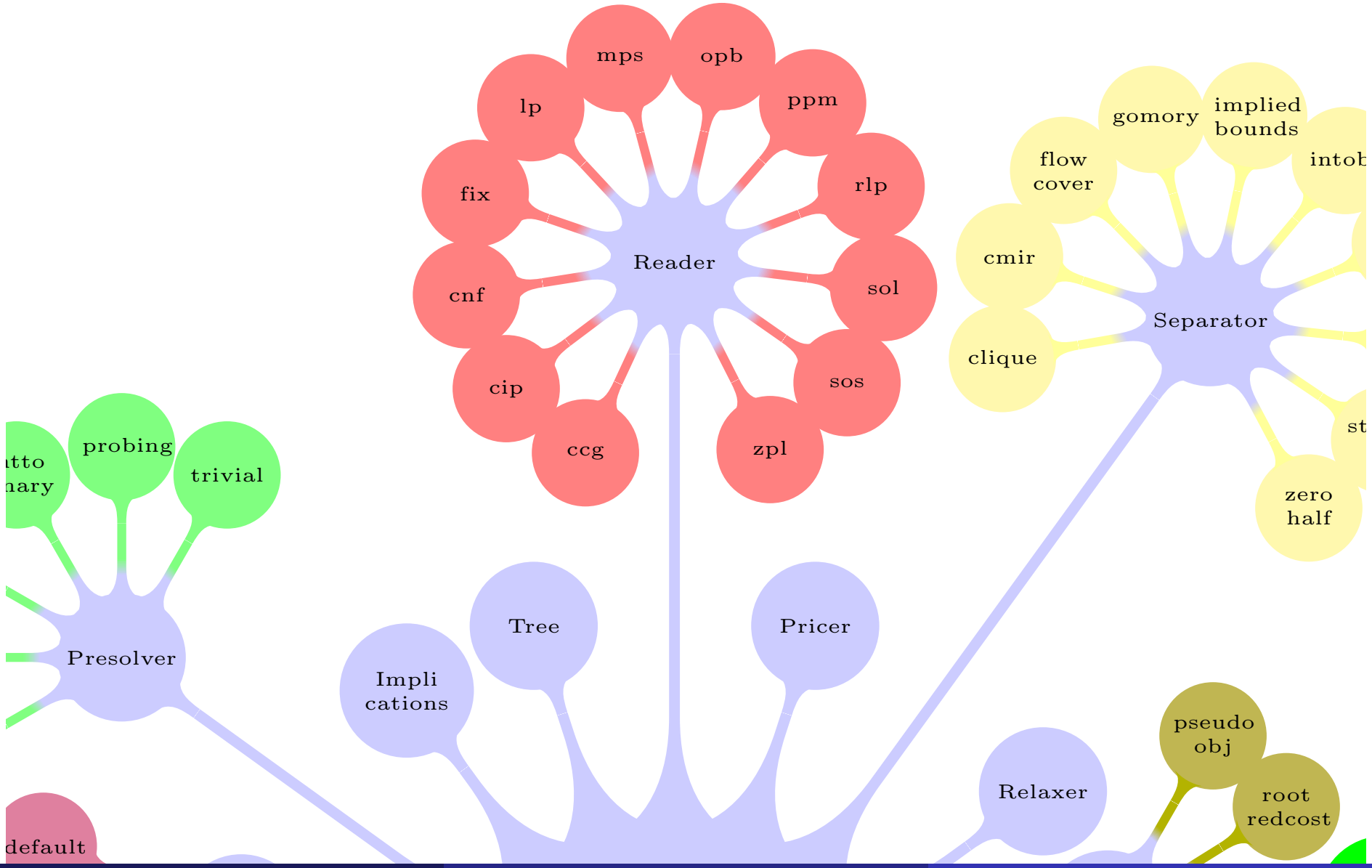



- User creates and modifies the original problem instance.
- Problem creation is usually done in file readers (`SCIPreadProb()`).

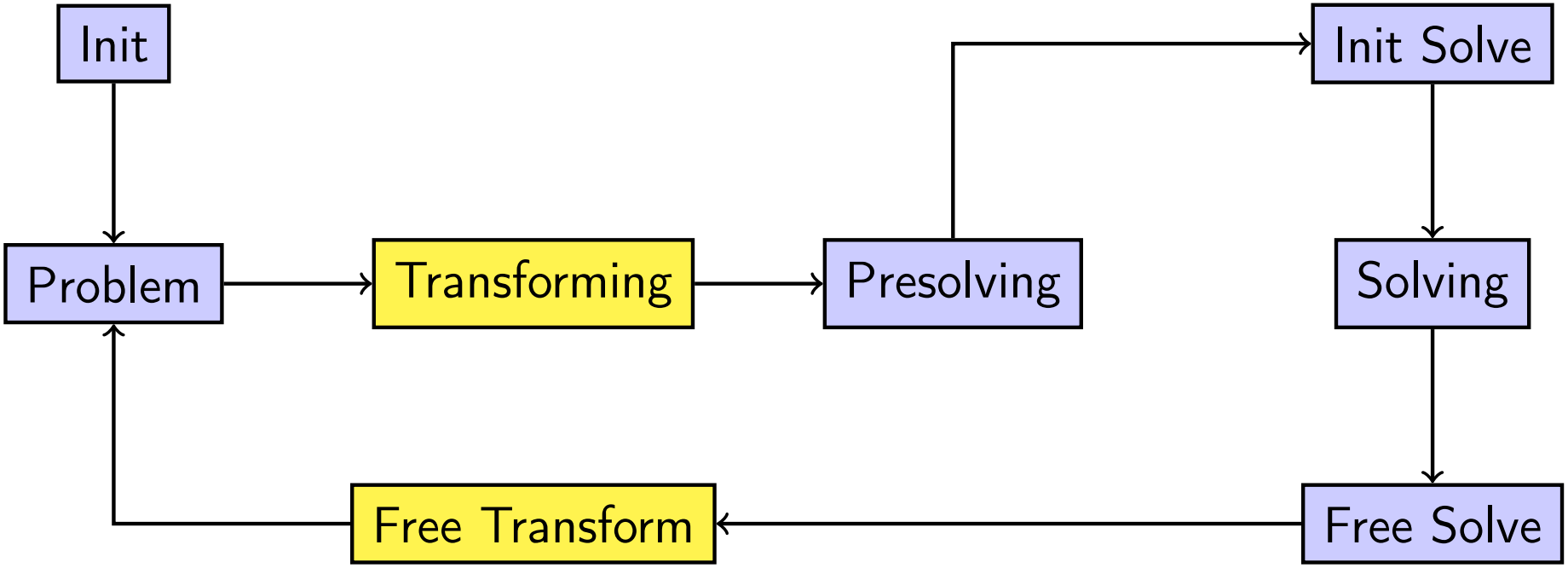
# SCIP Structures



# SCIP Structures

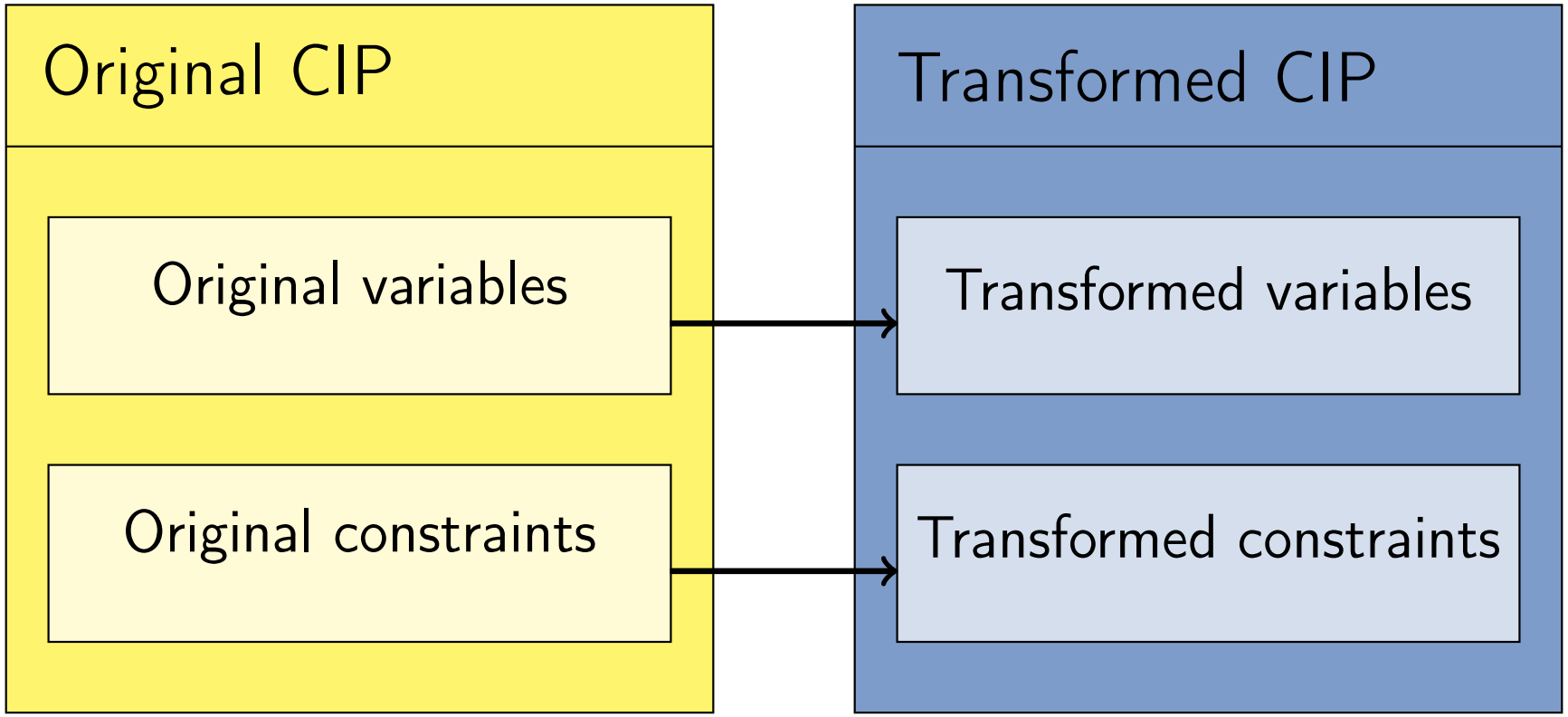


# Transformation



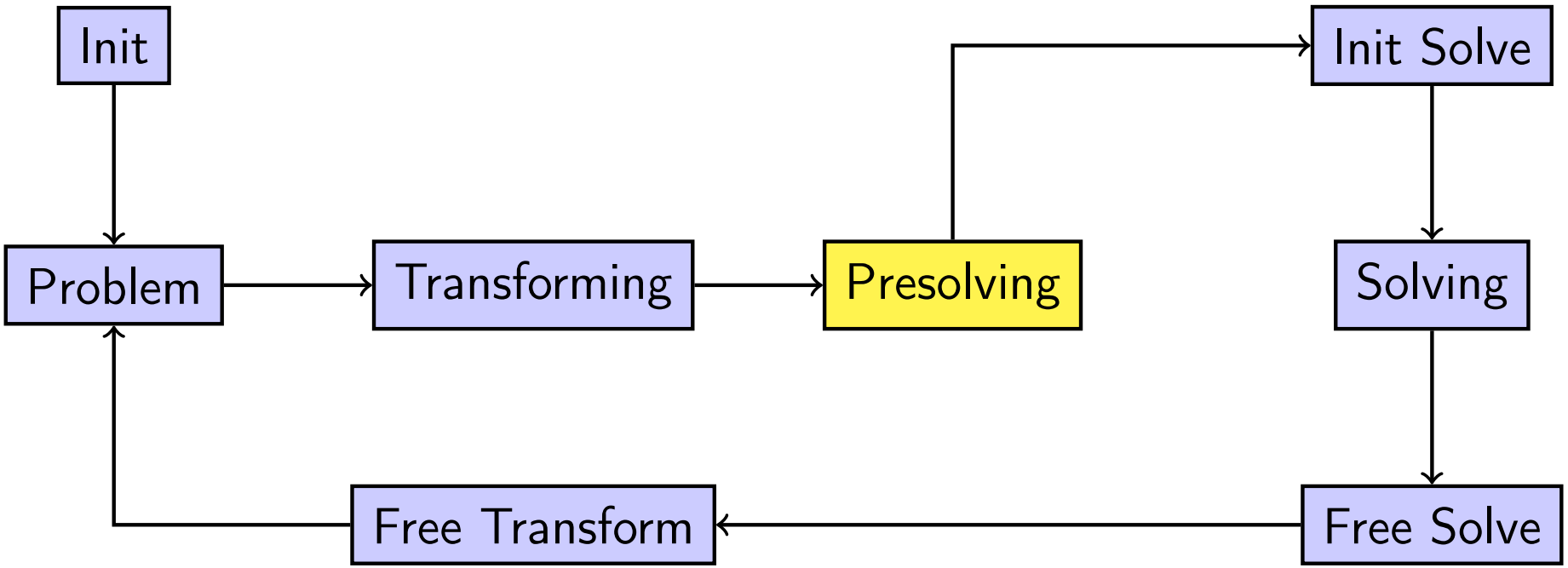

- Creates a working copy of the original problem.

# Original and Transformed Problem

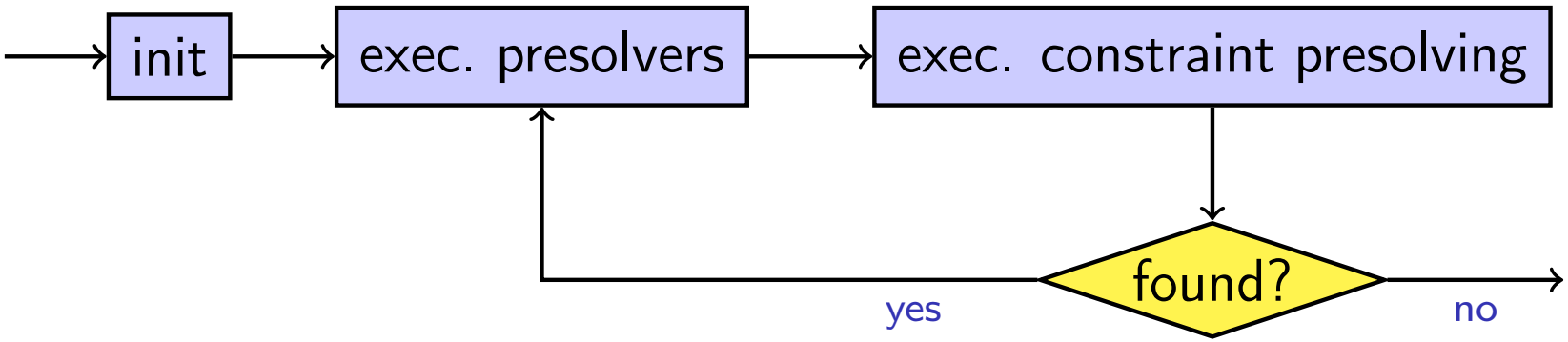



- data are copied into separate memory area
- presolving and solving operate on transformed problem
- original data can only be modified in problem modification stage

# Presolving

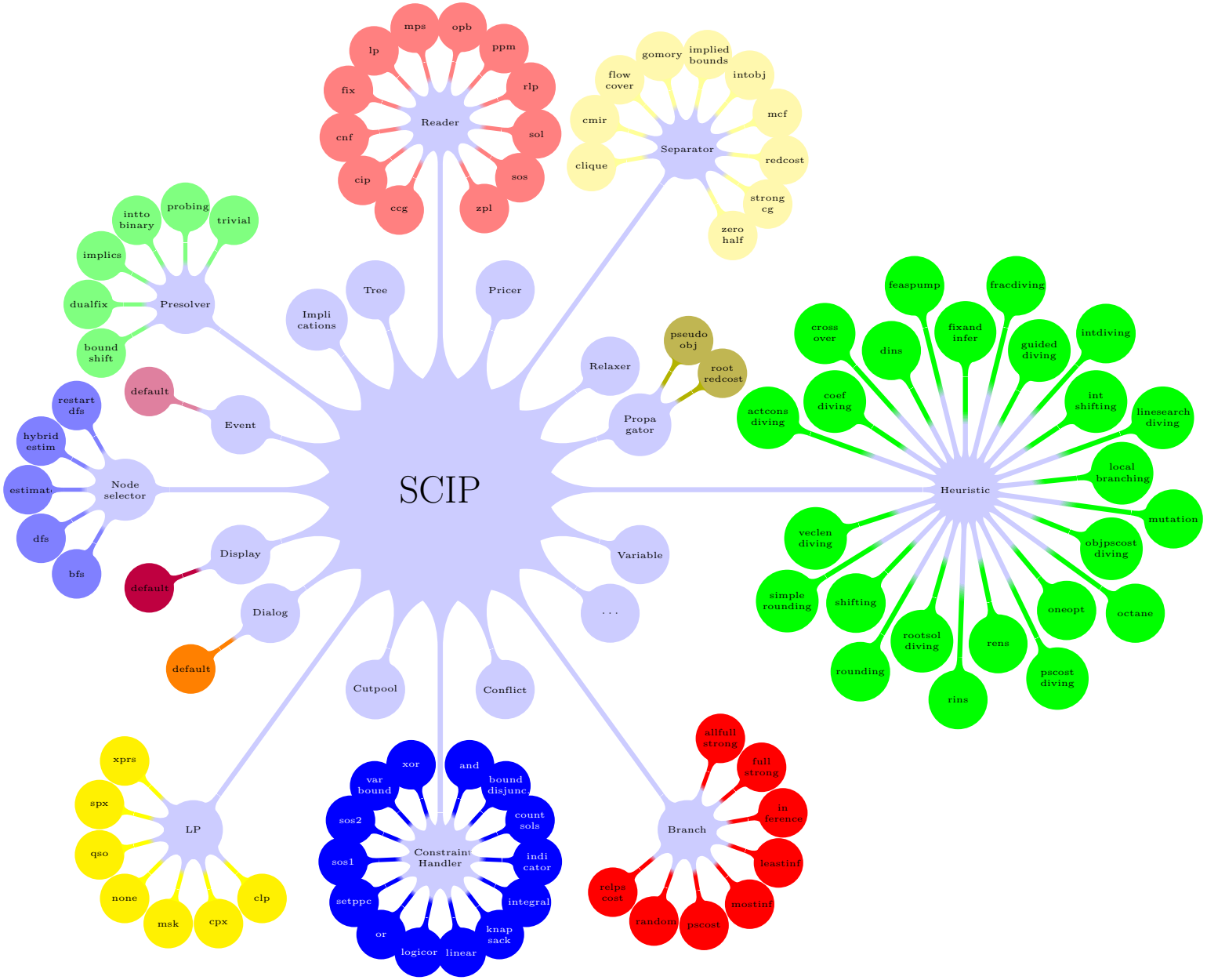

Init
Problem
Transforming
Presolving
Init Solve
Solving
Free Solve
Free Transform

# Presolving

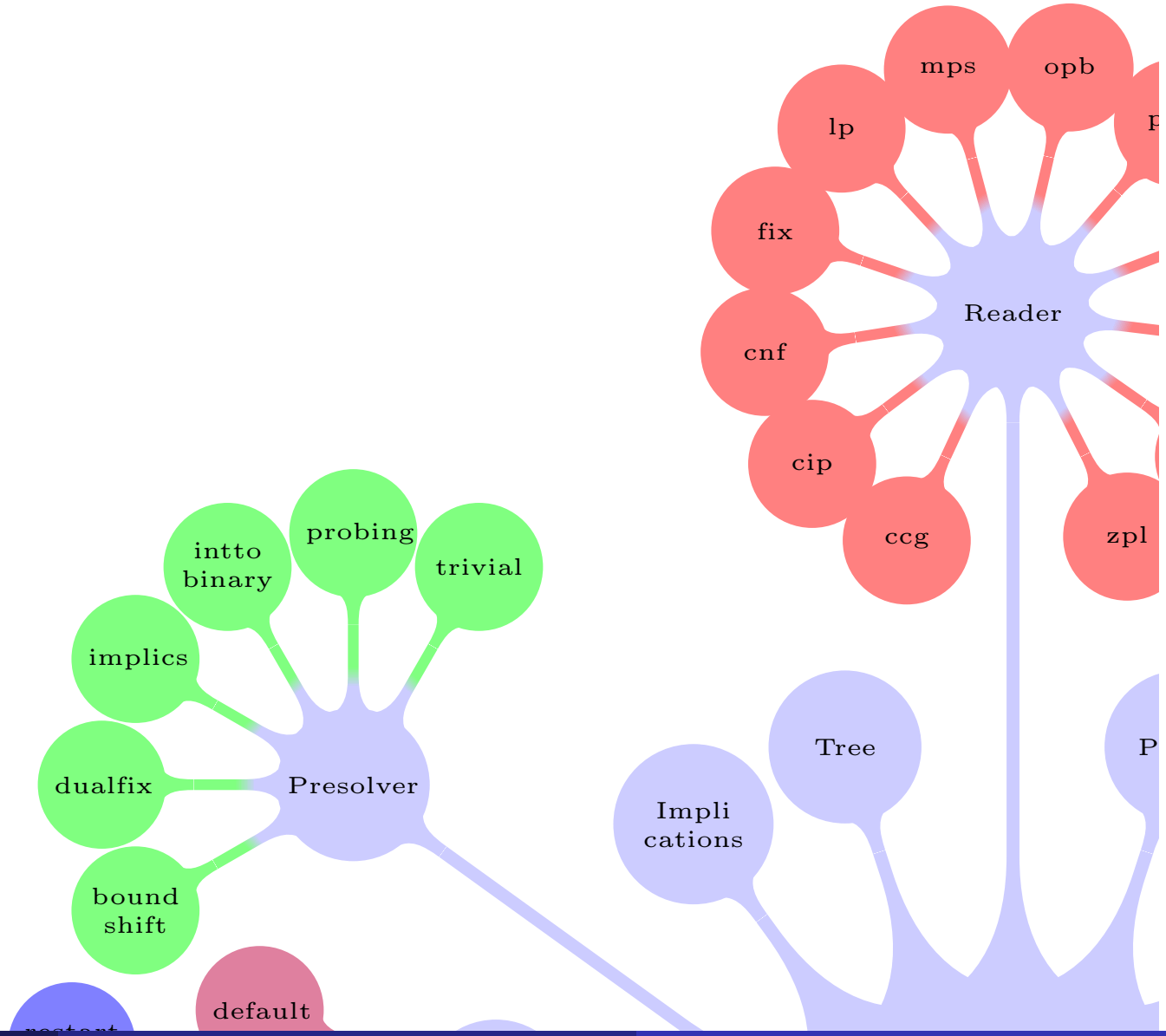



- Presolvers provide global presolving (e.g. dual fixing)
- Constraint Handlers provide constraint specific presolving, e.g.:
  - Domain tightening,
  - Coefficient modification,
  - Deletion of redundant constraints,
  - Constraint upgrading.
  - ...

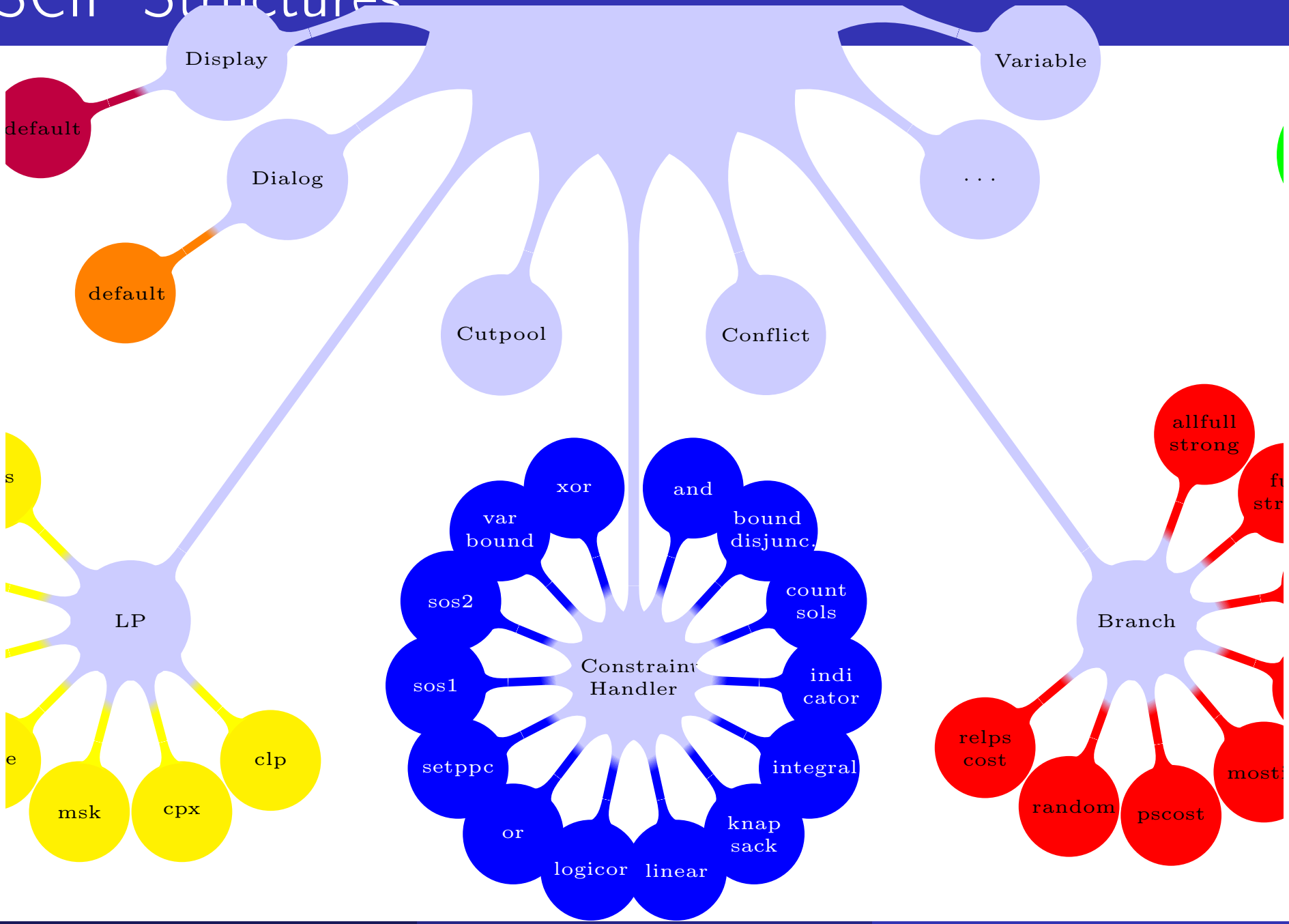
# SCIP Structures



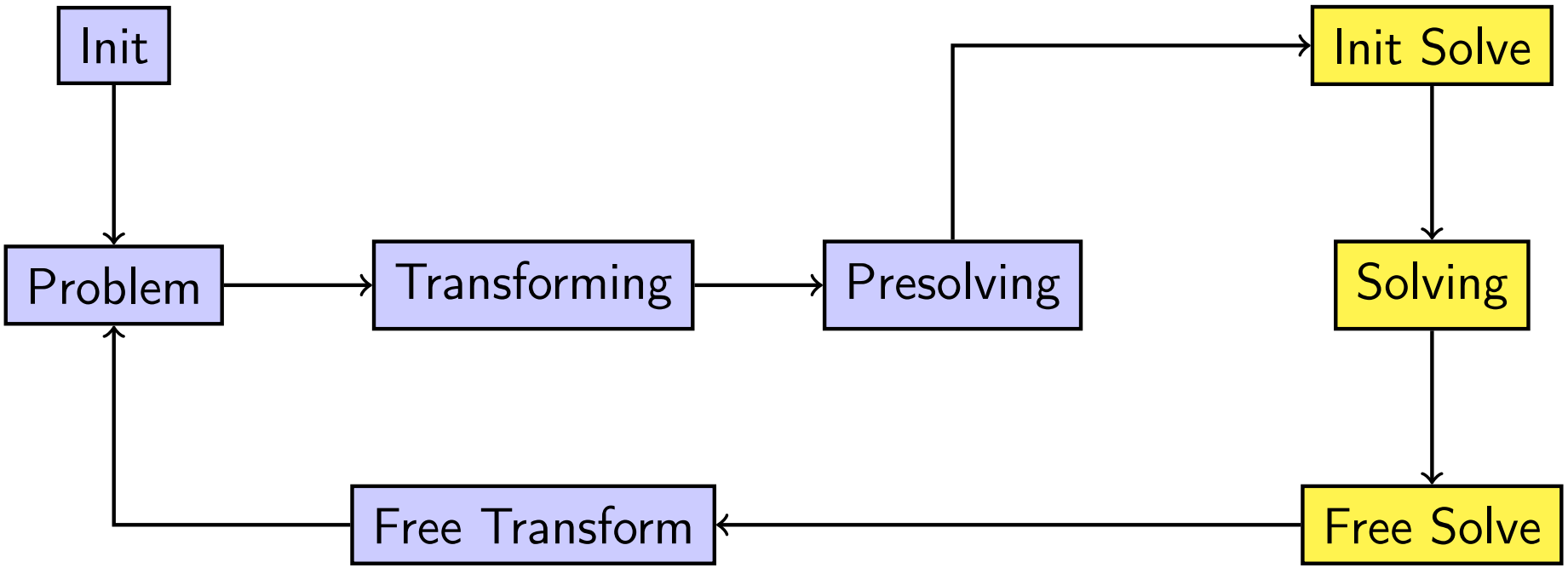
# SCIP Structures



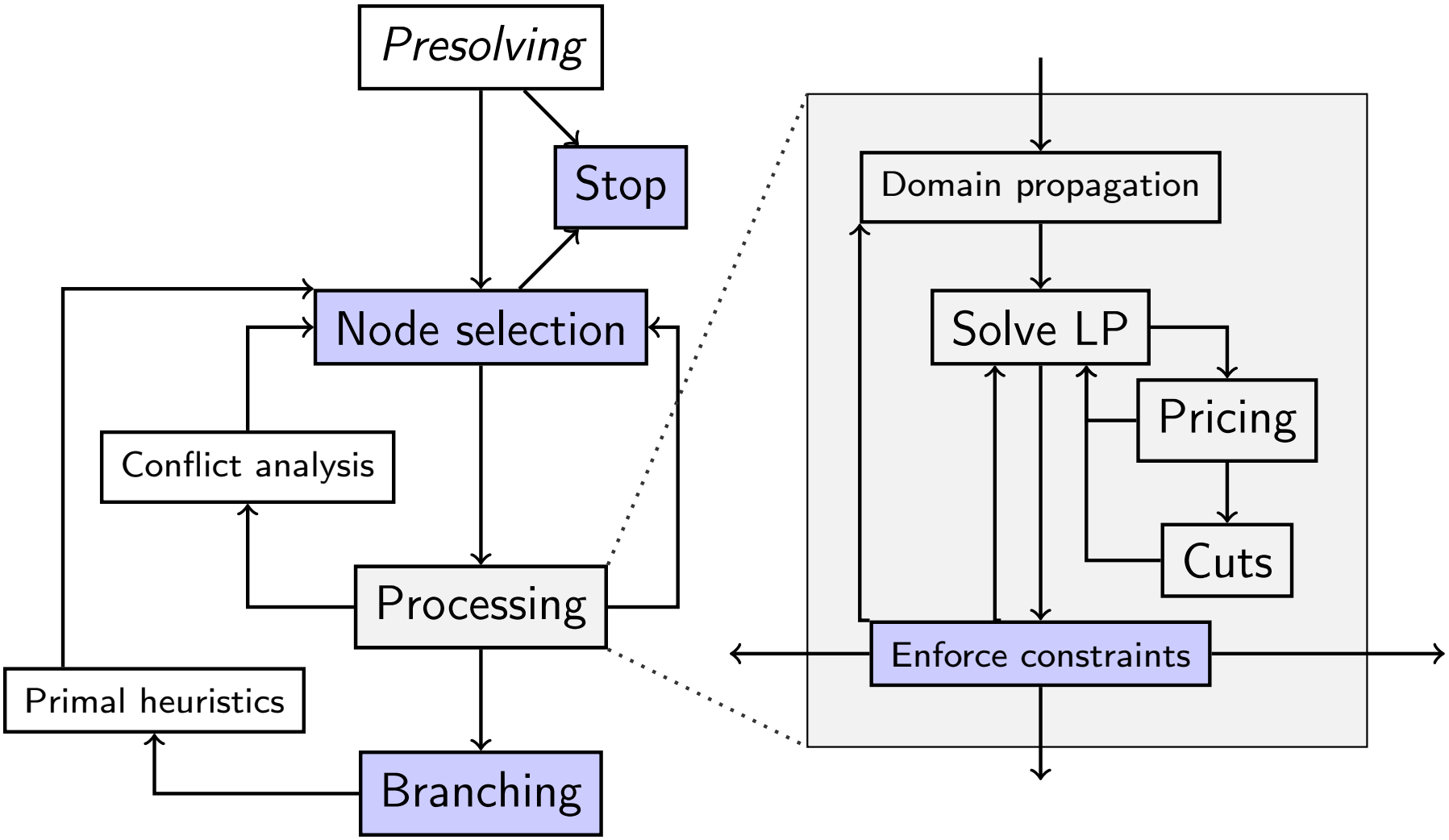
# SCIP Structures



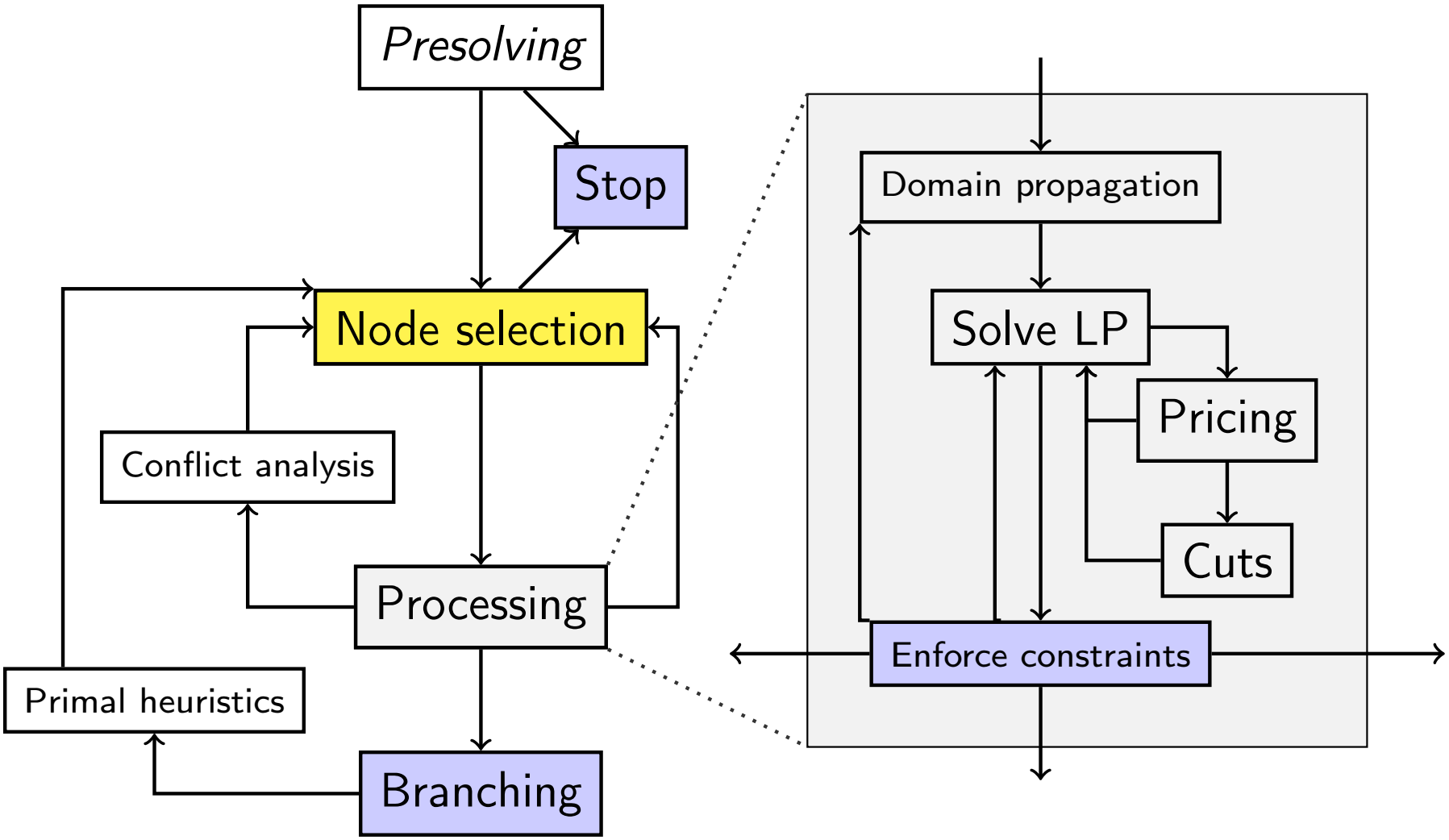
# Solving


Init
Problem
Transforming
Presolving
Init Solve
Solving
Free Solve
Free Transform

# Flow Chart SCIP



# Flow Chart SCIP



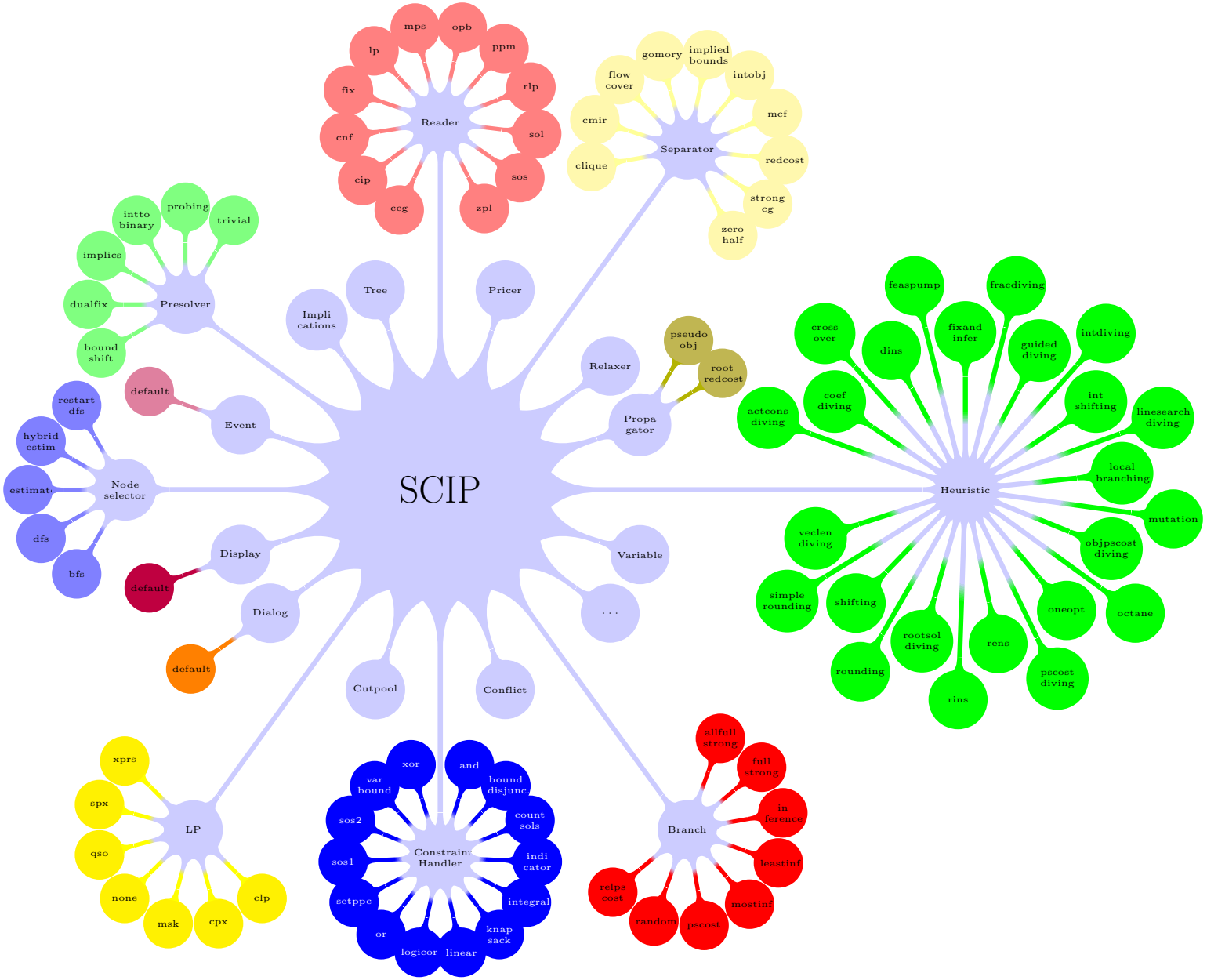
# Node Selection

- Node Selectors called in order of priority
- Choose:
  - Child node → subproblem's processing is less expensive
  - Sibling node → subproblem's processing is less expensive
  - Other open leaf node
- Two operation modes:
  - Standard mode (default: best estimate)
  - Memory-saving mode (default: depth first search)

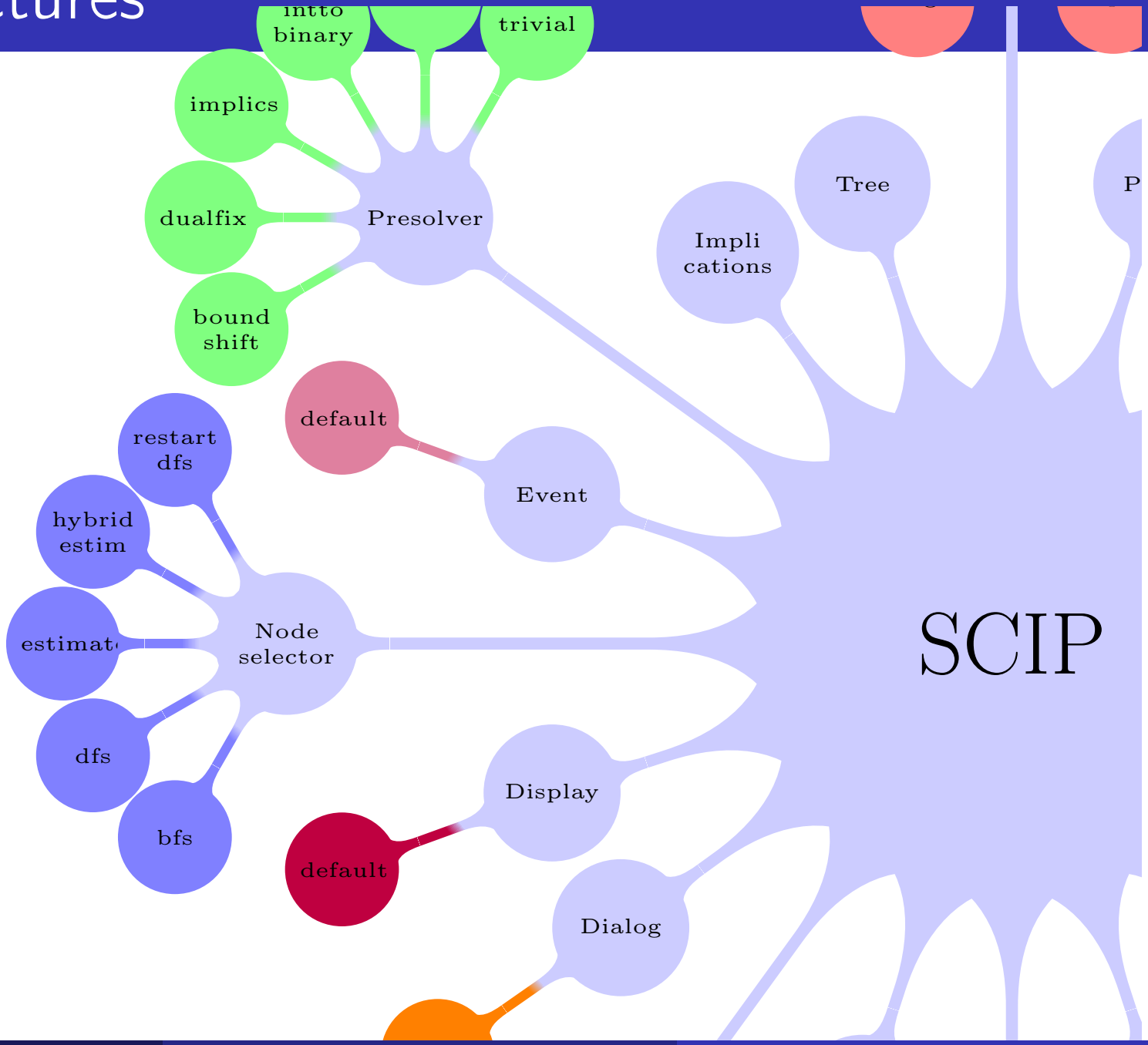
  switching between modes is automatic

```
SCIP> set limits memory 100
SCIP> set memory savefrac 0.8
```

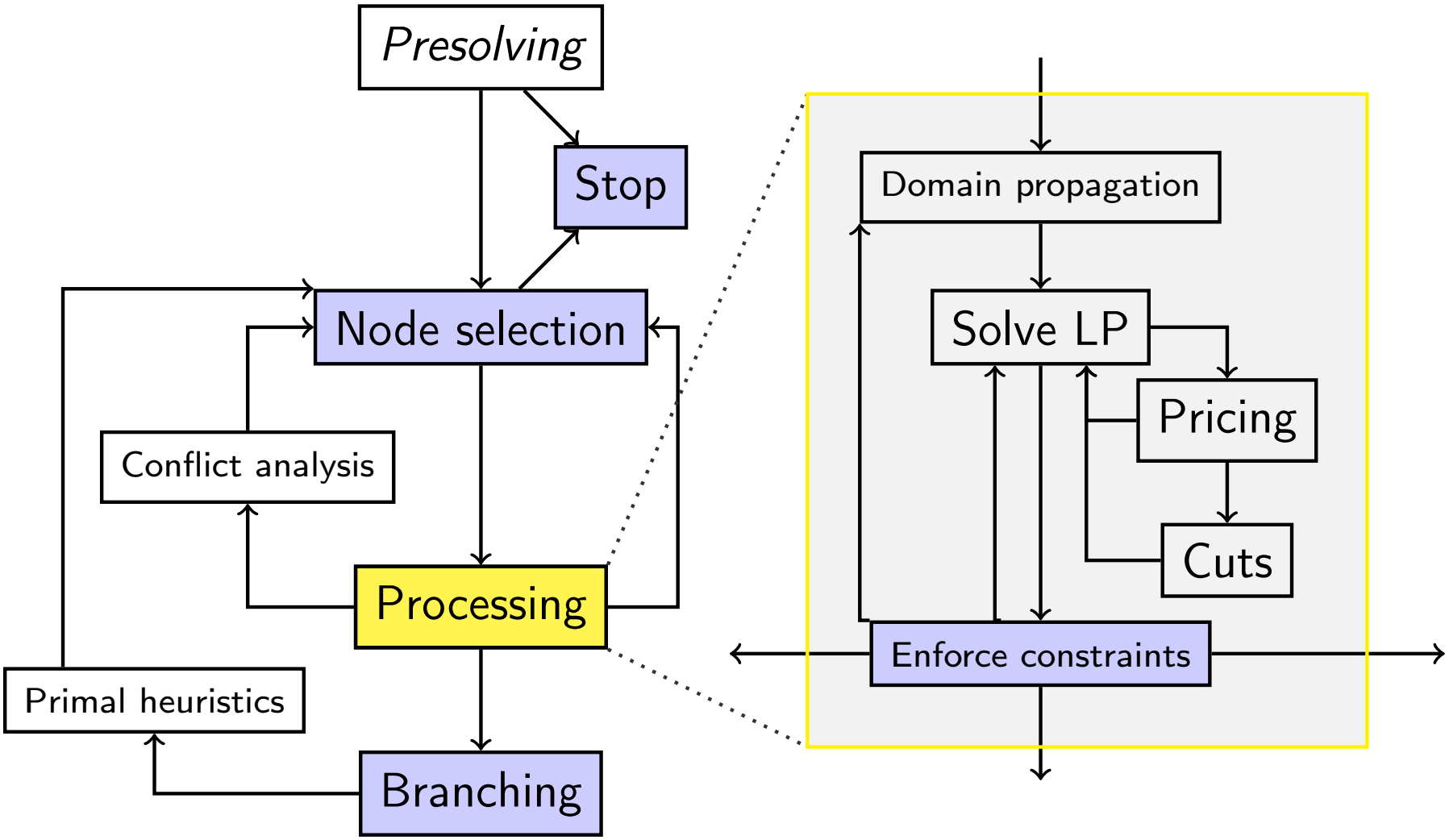
# SCIP Structures



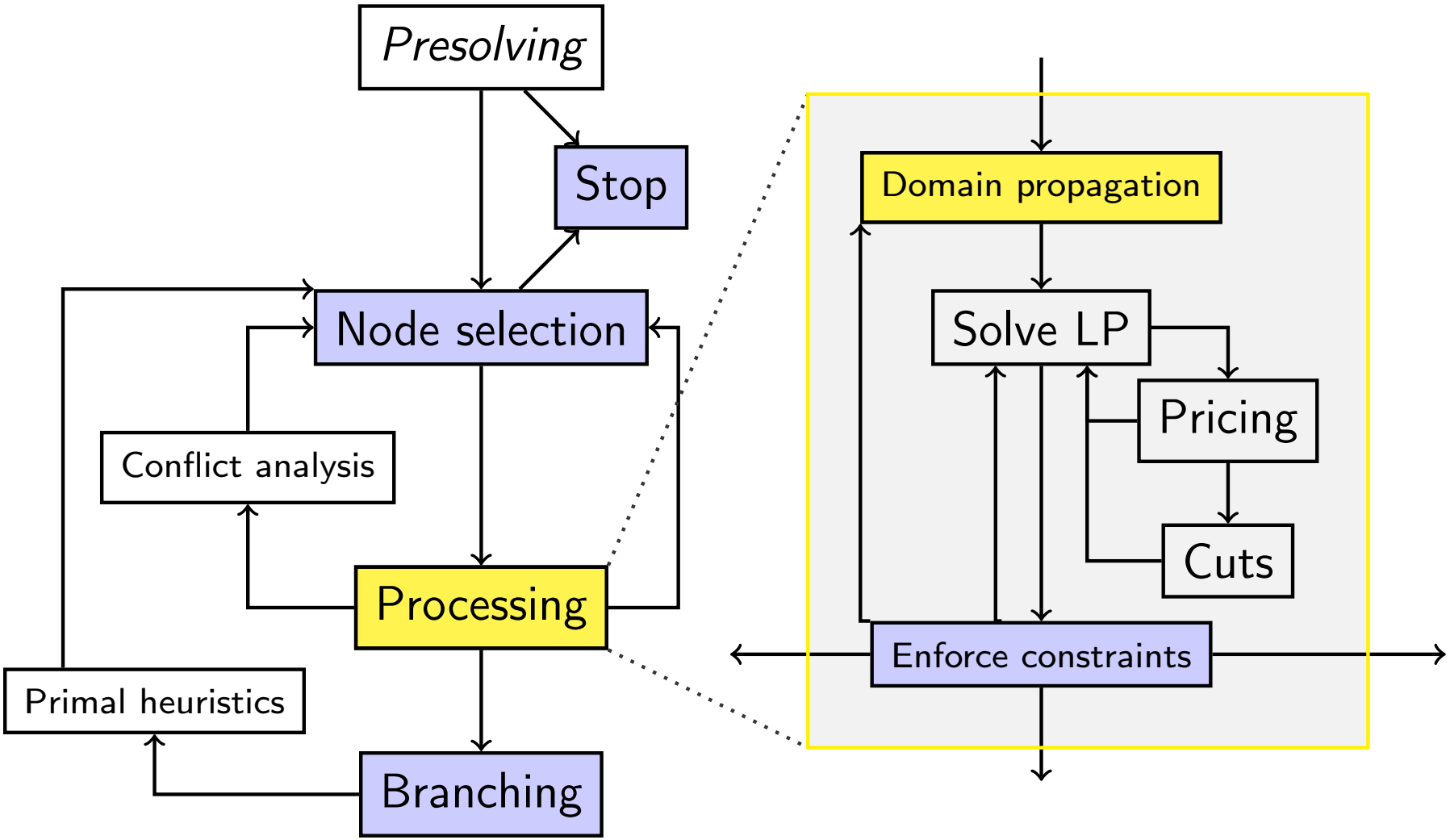
# SCIP Structures



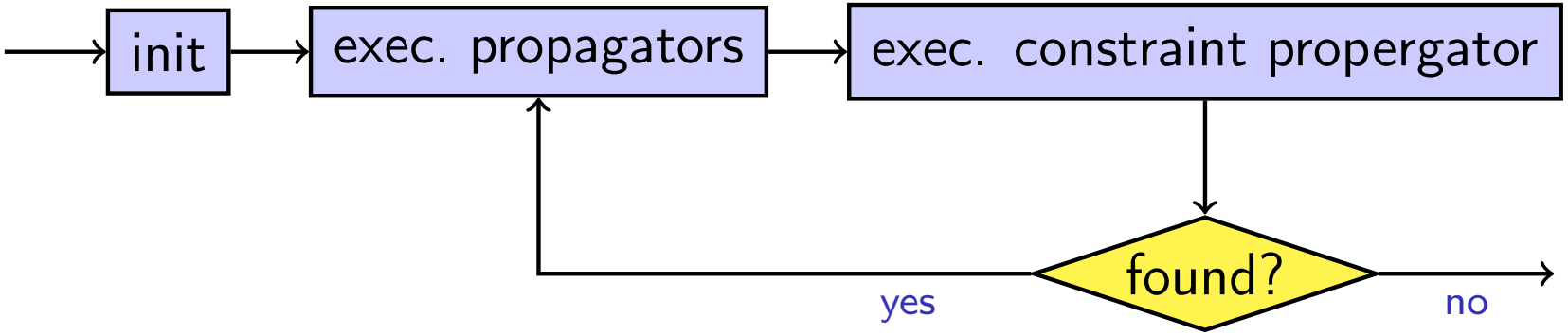
# Flow Chart SCIP



# Flow Chart SCIP

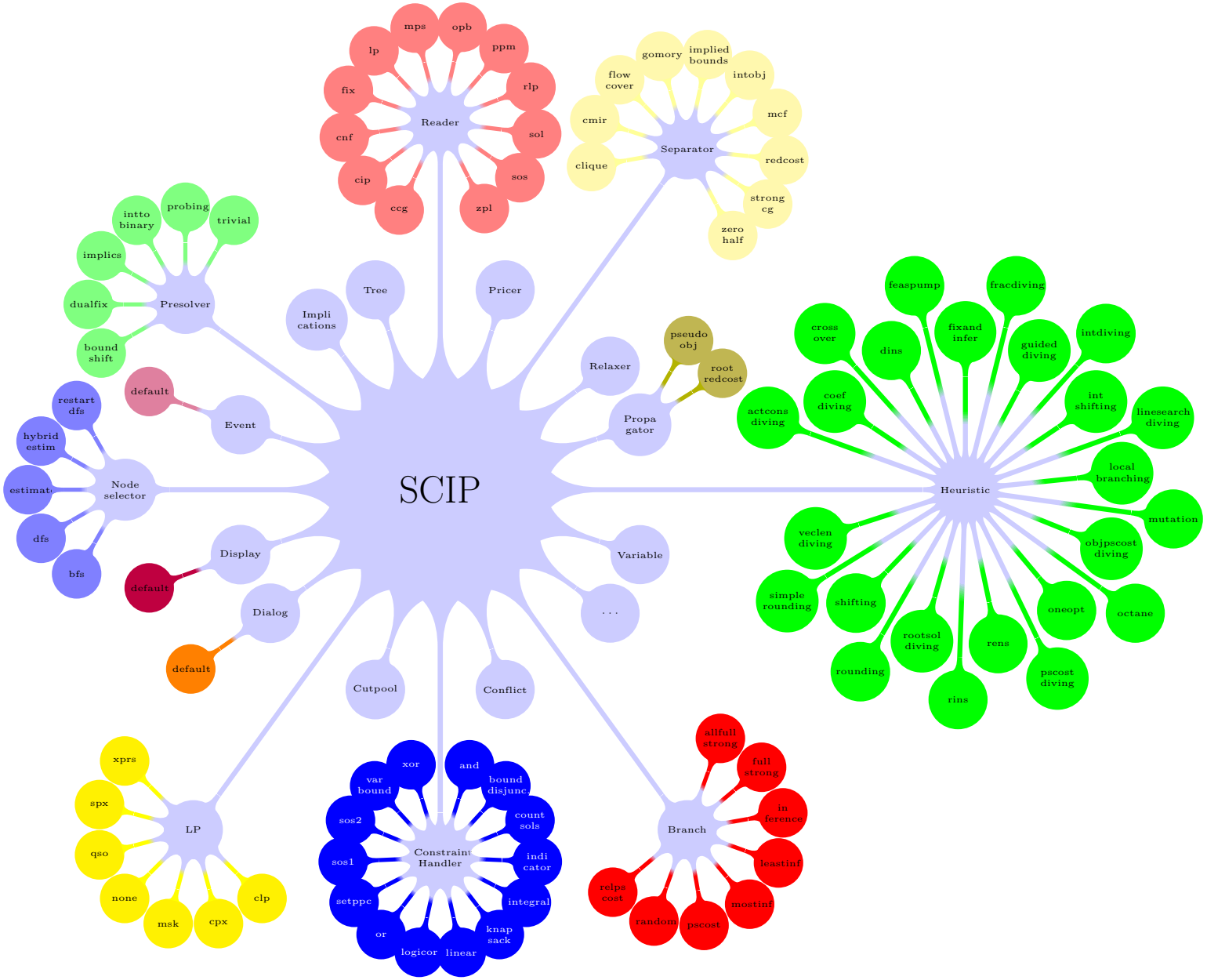


# Domain Propagation (Node Presolving)

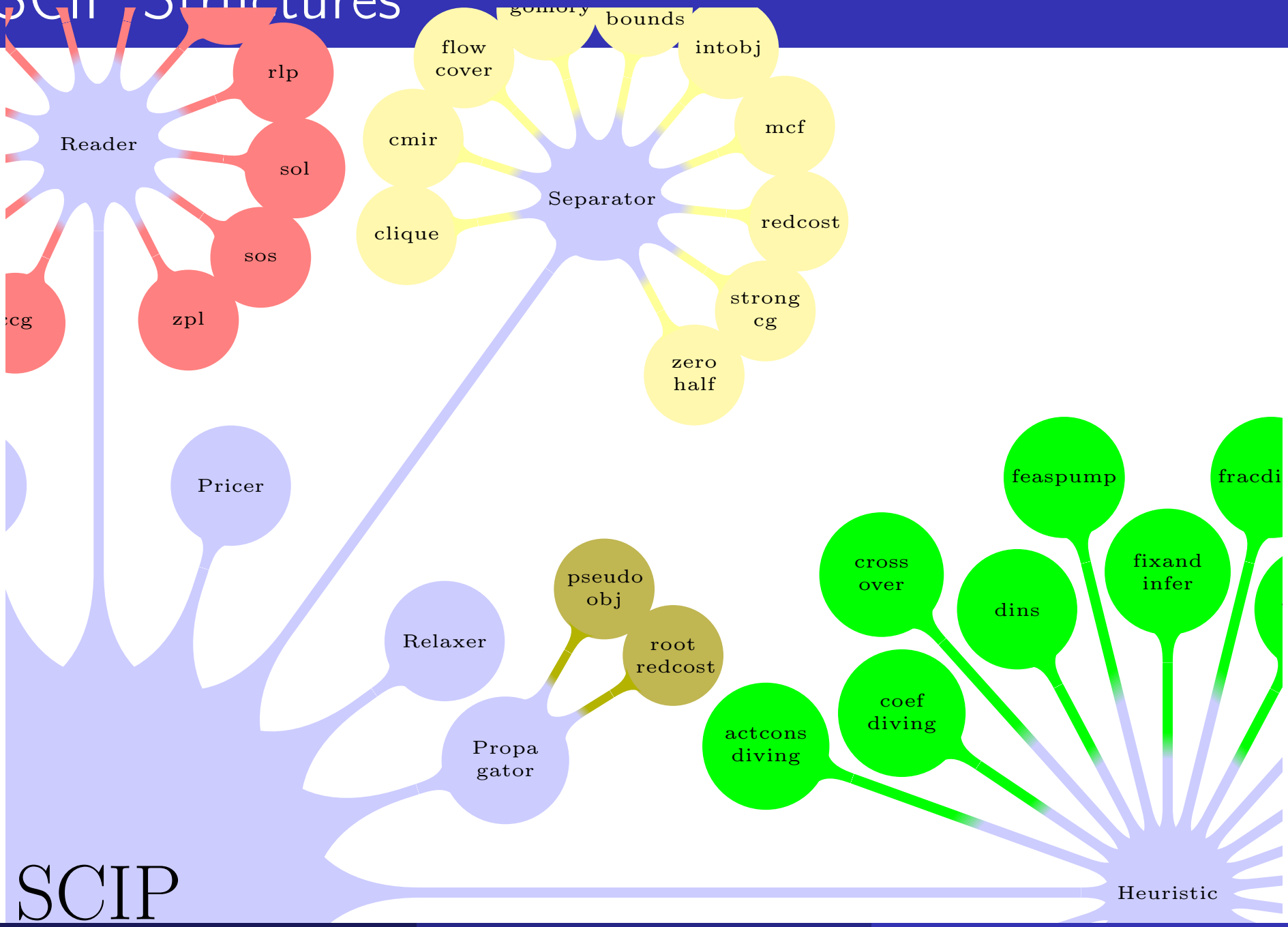



- Propagatore solvers provide local presolving
- Constraint Handlers provide constraint specific propagation, e.g.:
  - Domain tightening,
  - Coefficient modification,
  - Deletion of redundant constraints,
  - Constraint upgrading.
  - ...

# SCIP Structures

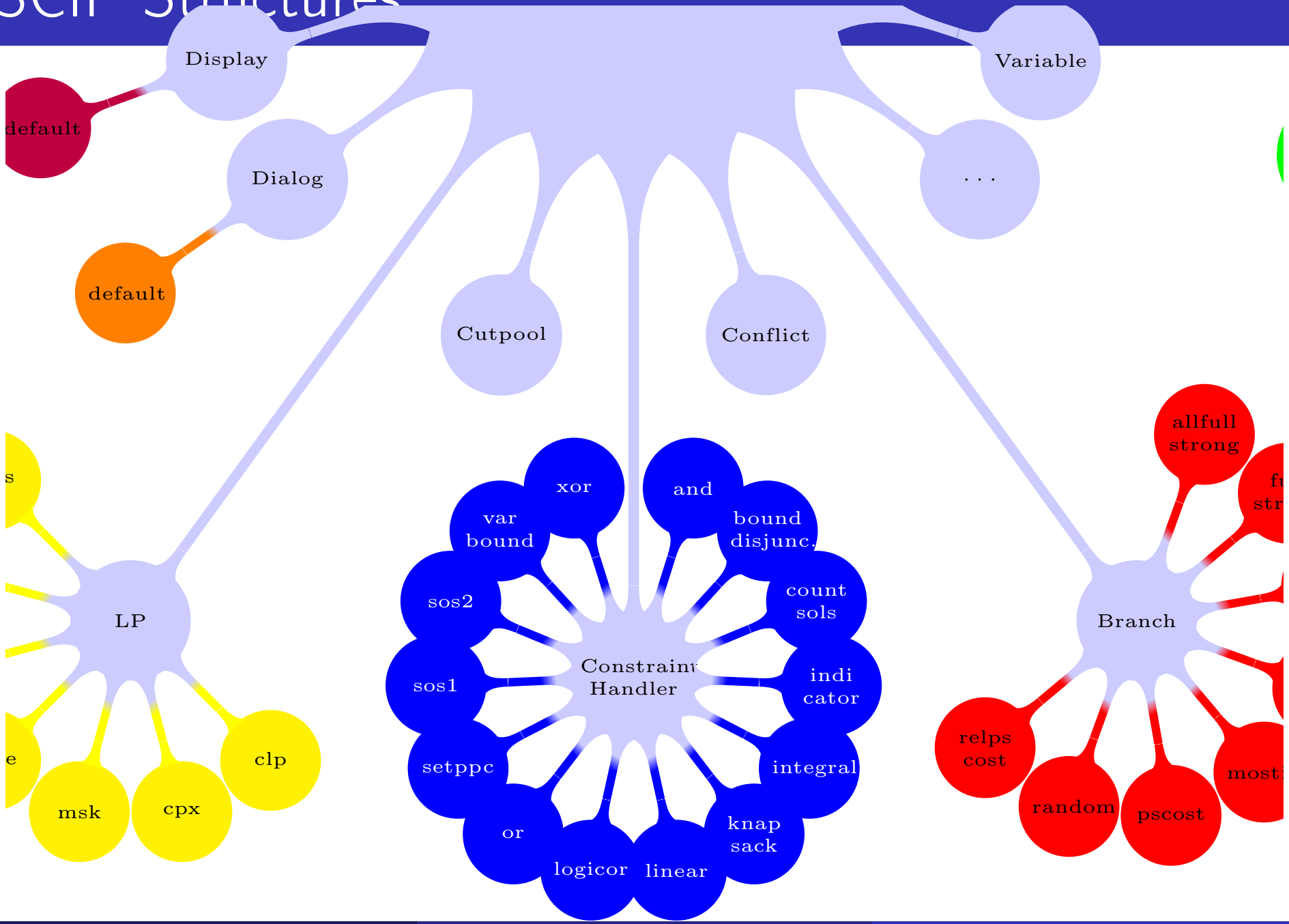


# SCIP Structures

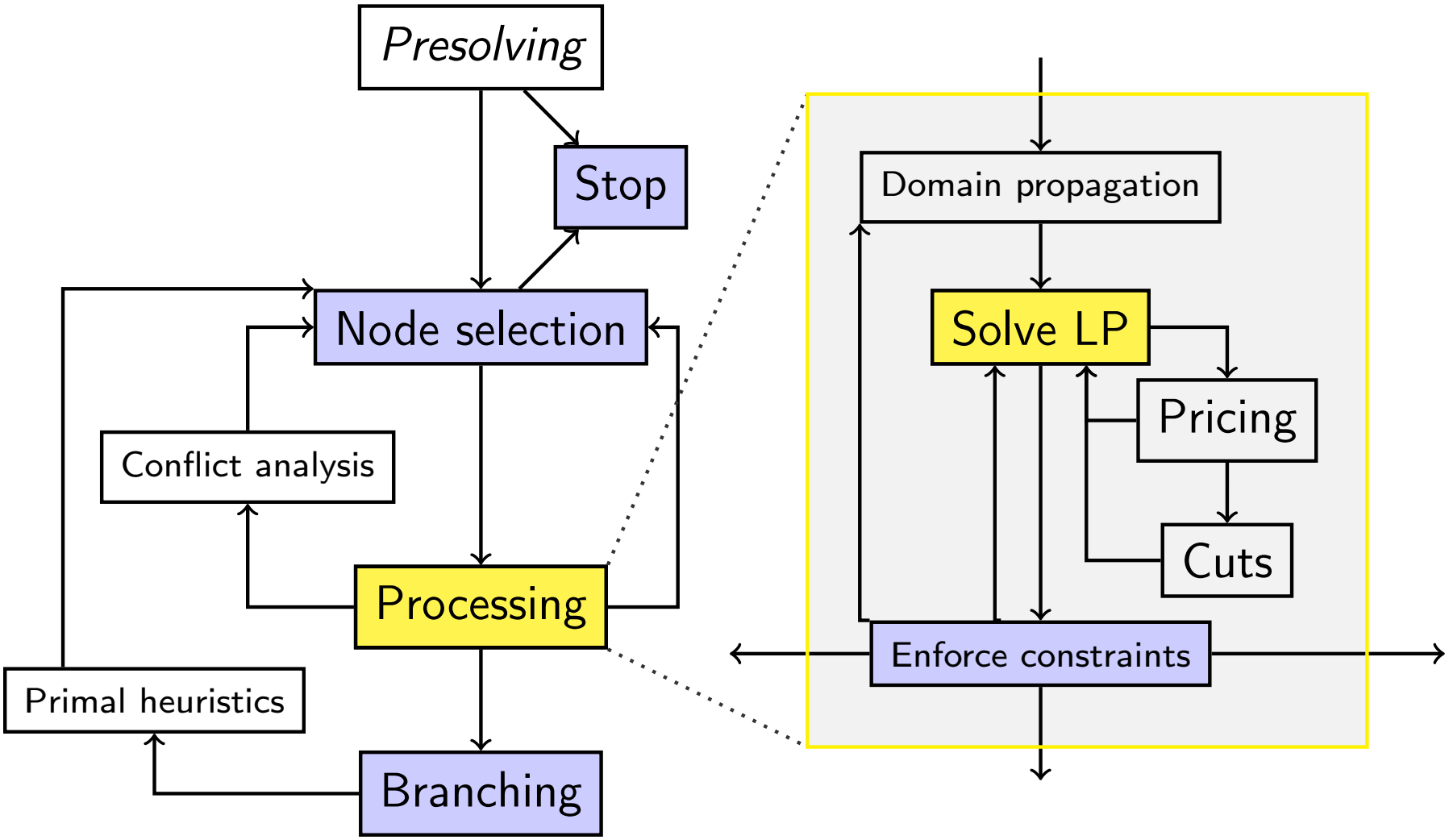



SCIP

# SCIP Structures



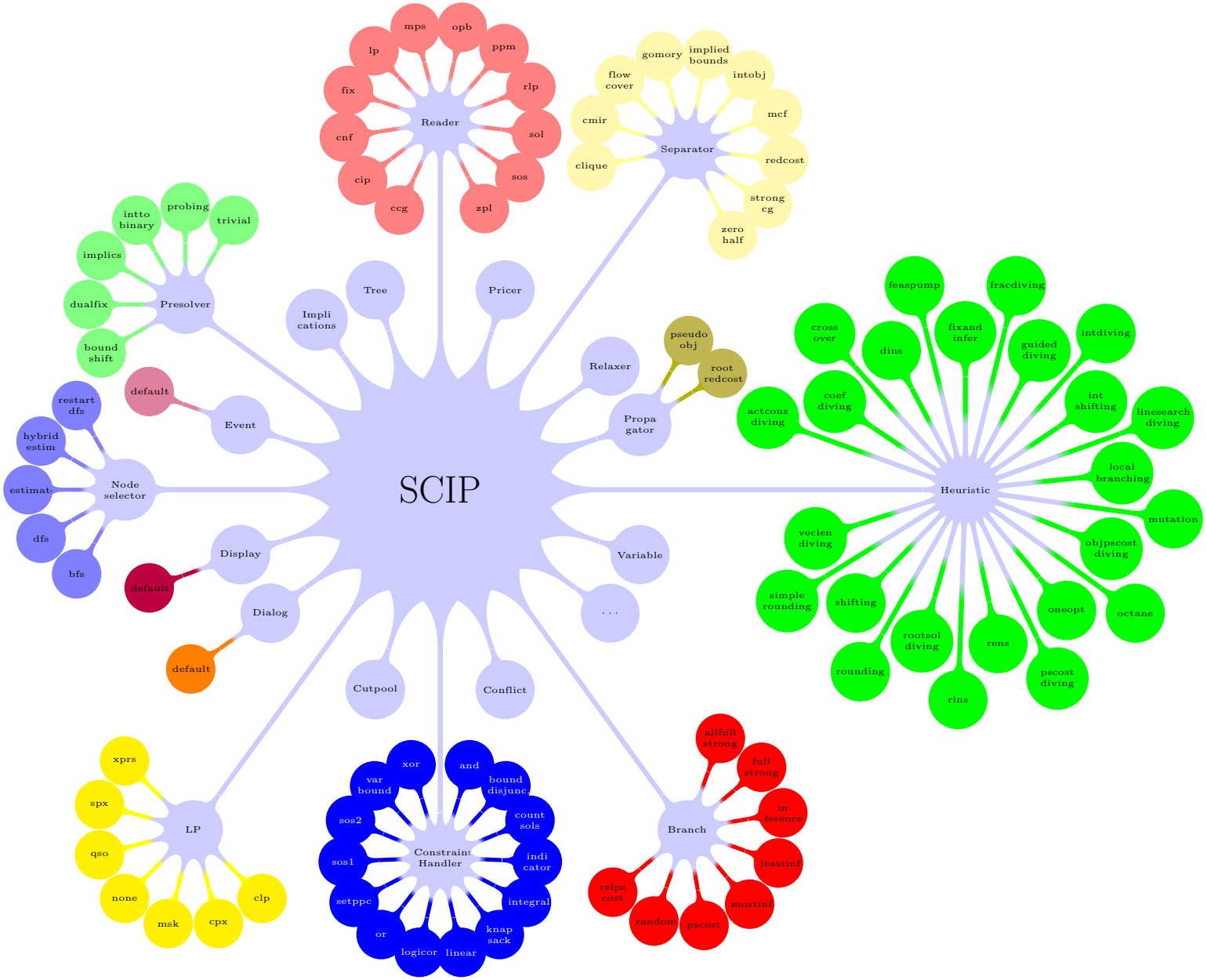
# Flow Chart SCIP



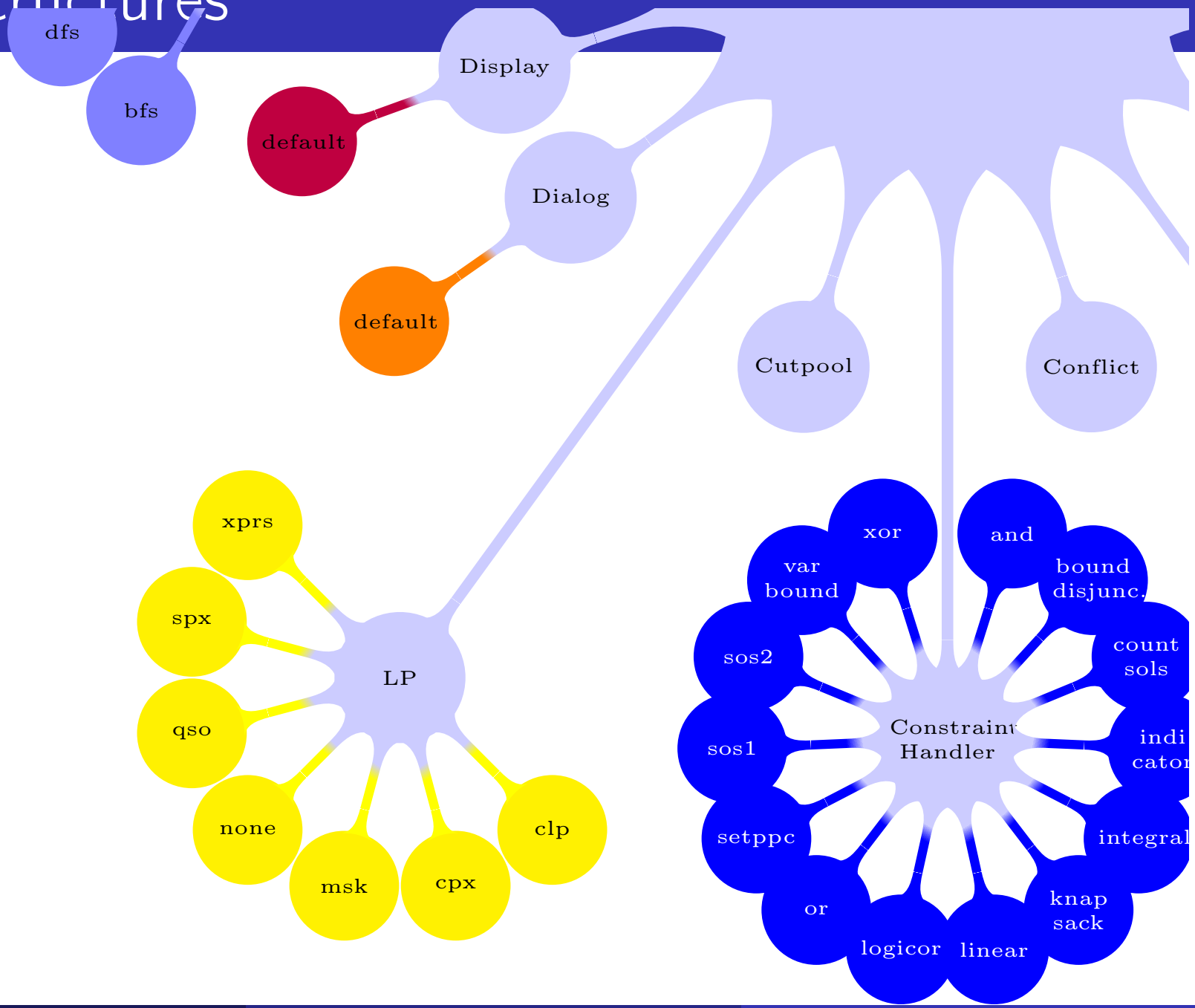
# Solve Linear Program (LP)

- LP Interface to various LP solvers:
  $\rightarrow$ CLP, CPLEX, Gurobi, Mosek, QSopt, SoPlex, XPRESS
- Primal or dual simplex
- Barrier with or without crossover (if supported)
- LP stability checked:
  resolution by changing parameters:
  scaling, tolerances, solving from scratch, other simplex method.

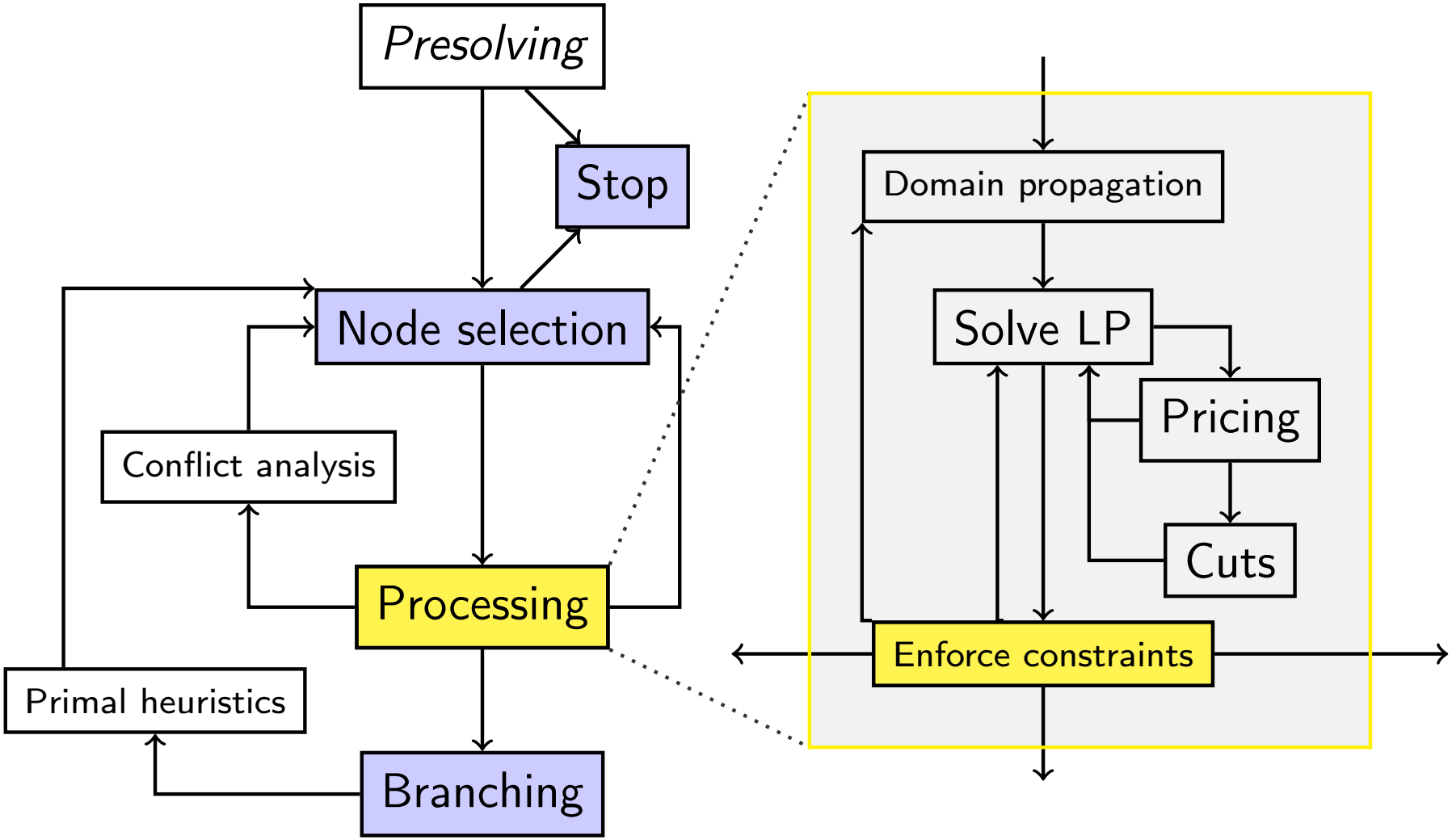
# SCIP Structures



# SCIP Structures



# Flow Chart SCIP



# Constraint Enforcement

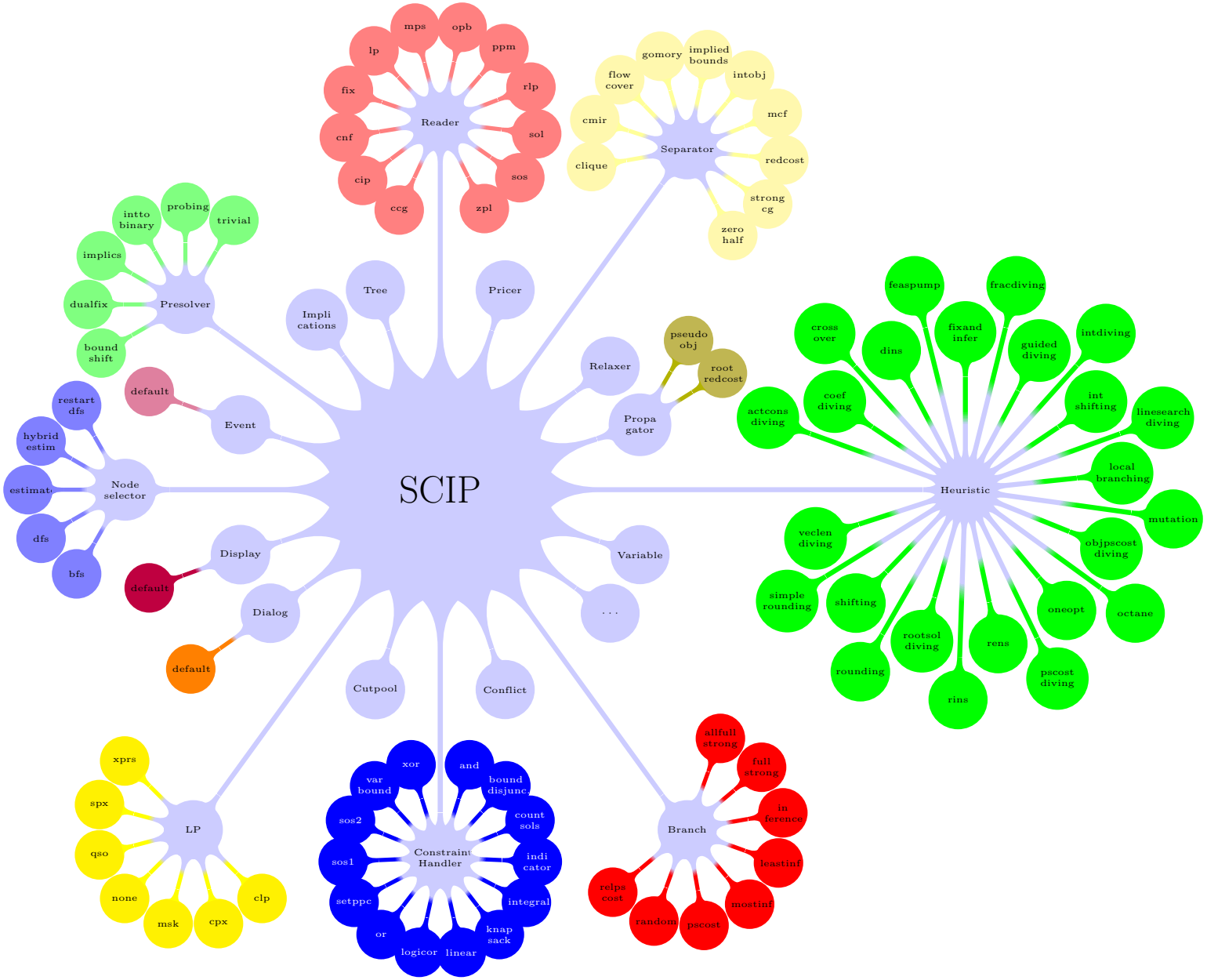
LP solution may violate a constraint not contained in the relaxation.

Enforcing is necessary for a correct implementation!

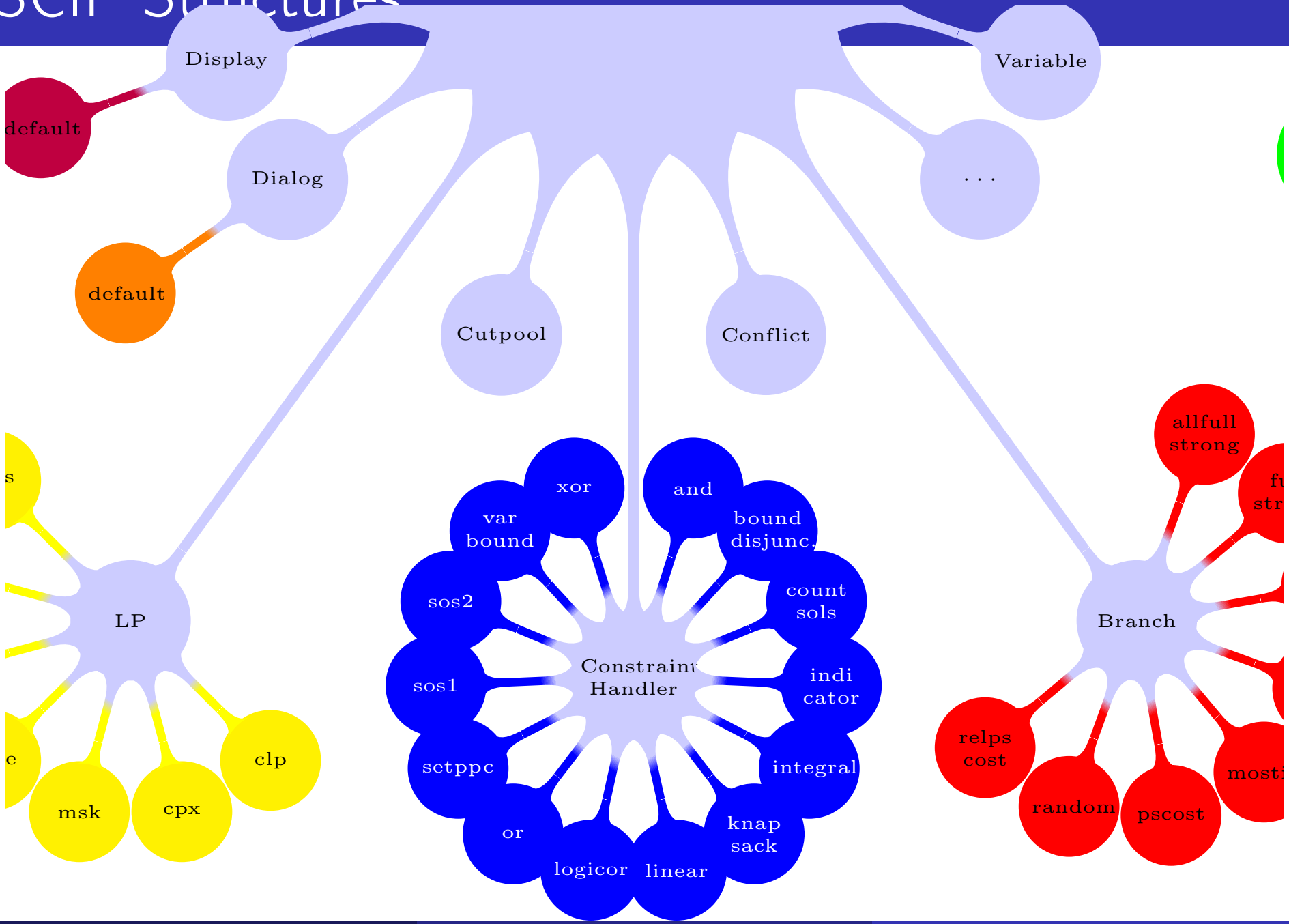
Constraint handler resolves the infeasibility by ...

- Reducing a variable's domain,
- Separating a cutting plane (may use integrality),
- Adding a (local) constraint,
- Creating a branching,
- Concluding that the subproblem is infeasible and can be cut off, or
- Just saying "solution infeasible".

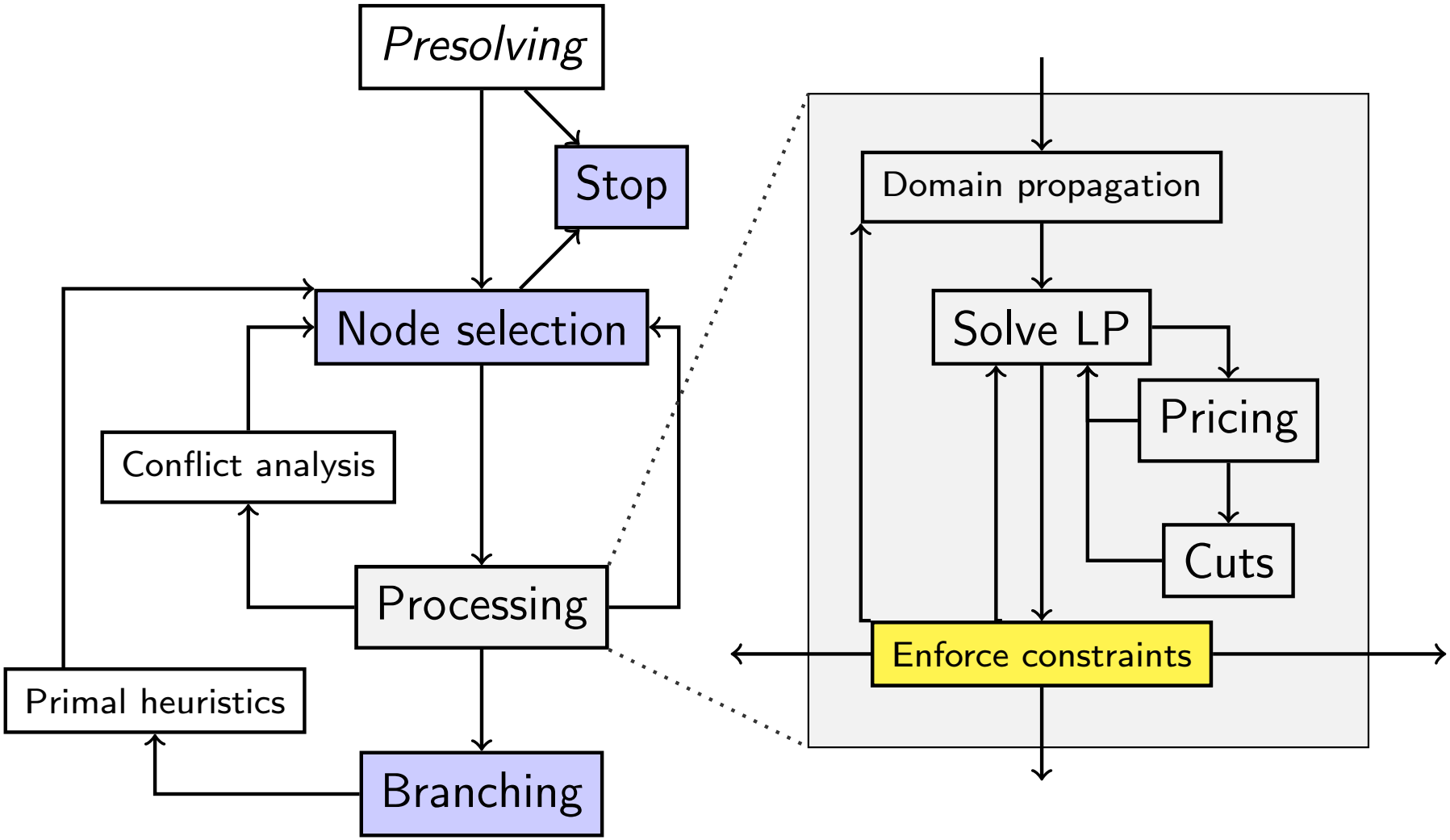
# SCIP Structures



# SCIP Structures

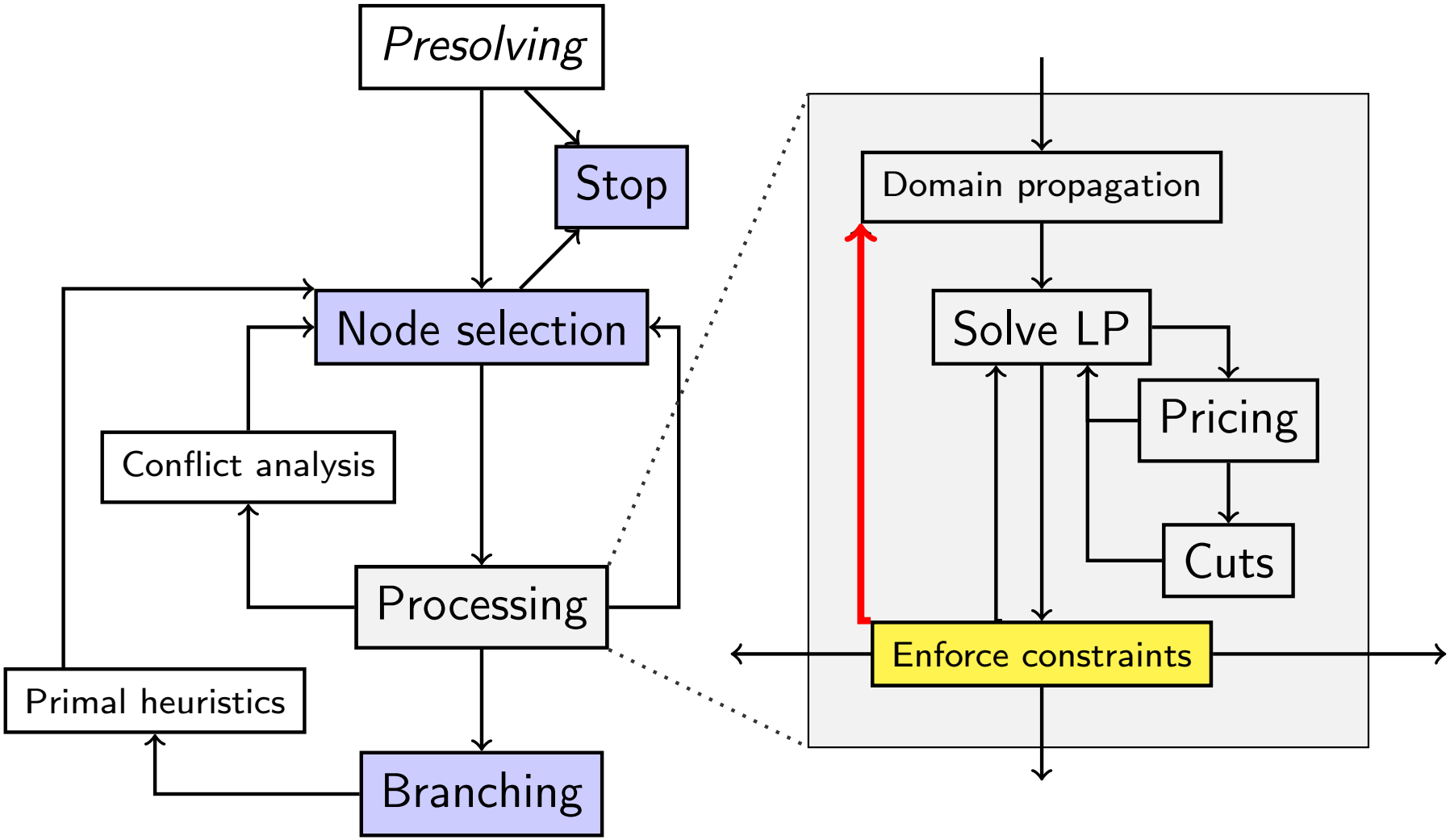


# Constraint Enforcement



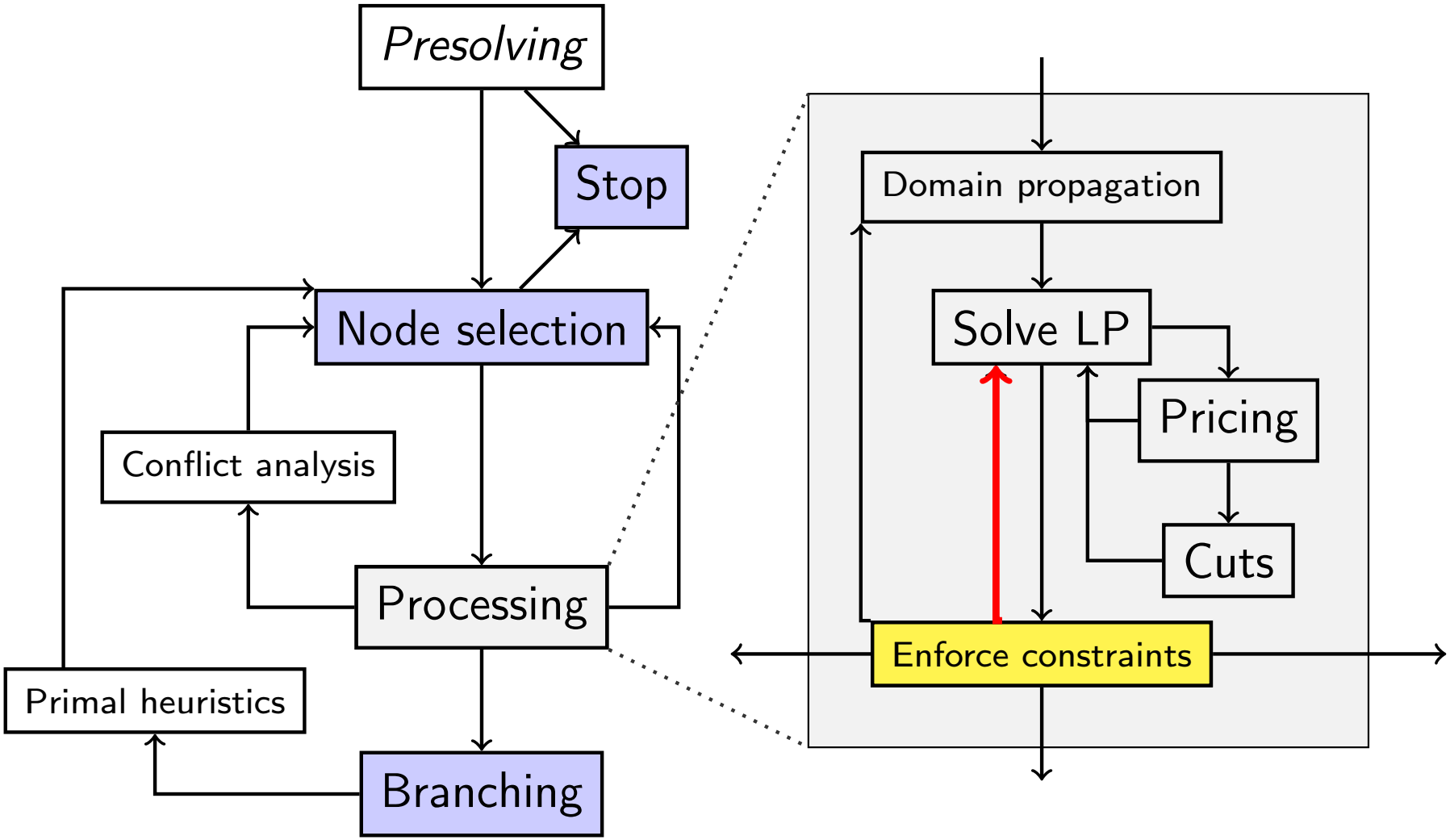

- Reduced domain
- Added constraint
- Added cut
- Branched
- Cutoff
- Infeasible
- Feasible

# Constraint Enforcement



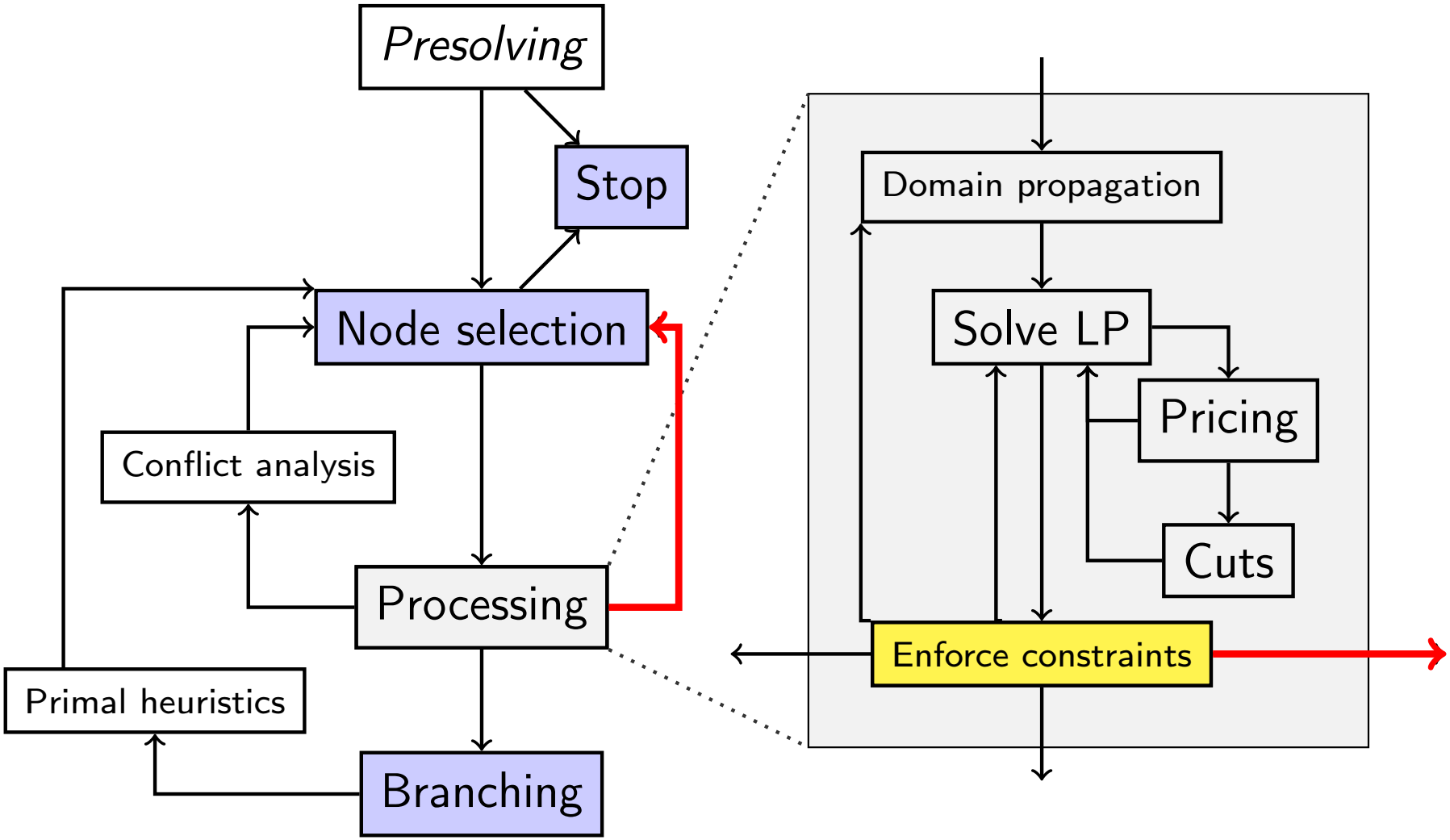

- Reduced domain
- Added constraint
- Added cut
- Branched
- Cutoff
- Infeasible
- Feasible

# Constraint Enforcement



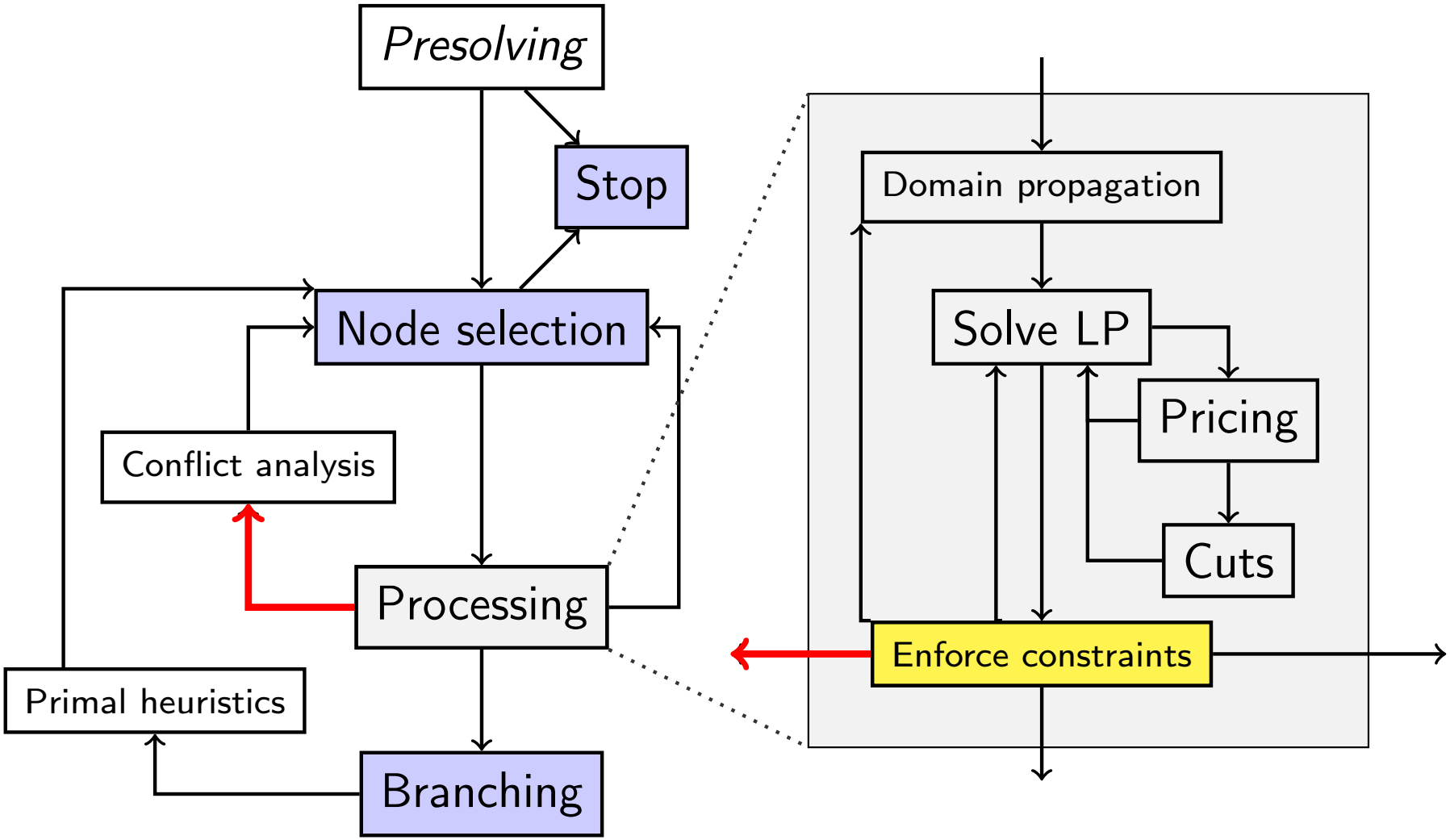

- Reduced domain
- Added constraint
- Added cut
- Branched
- Cutoff
- Infeasible
- Feasible

# Constraint Enforcement



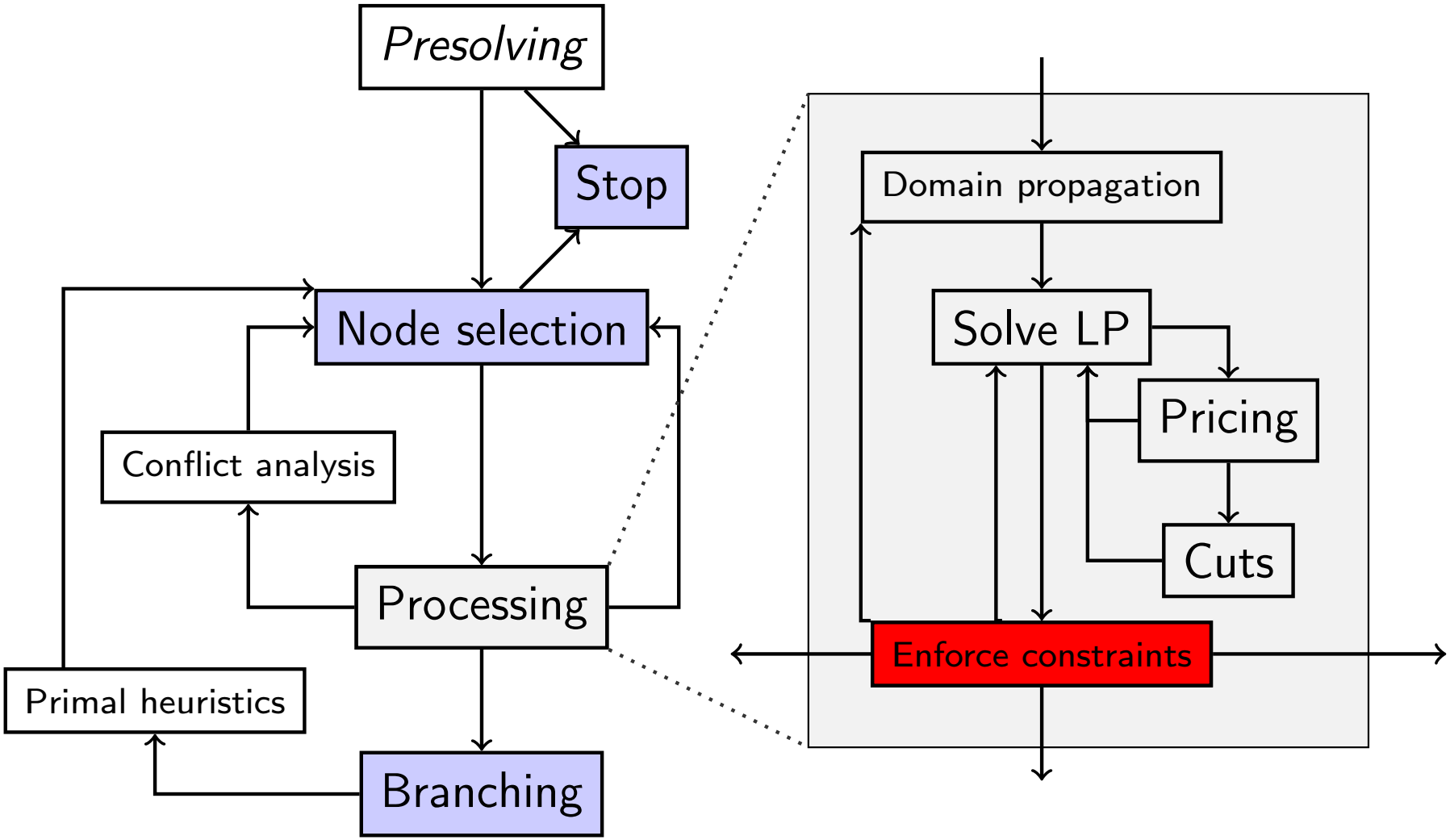

- Reduced domain
- Added constraint
- Added cut
- Branched
- Cutoff
- Infeasible
- Feasible

# Constraint Enforcement



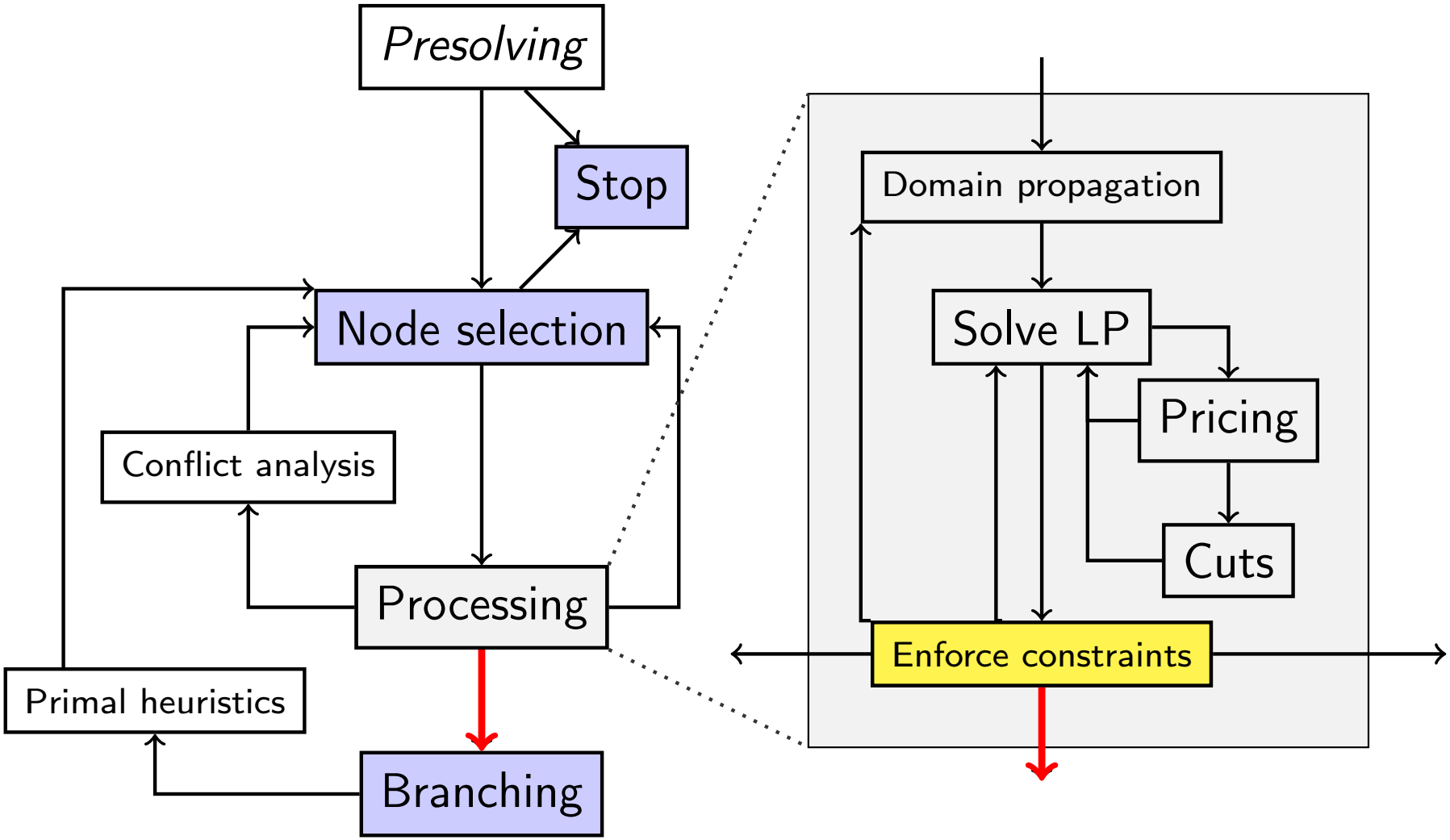

- Reduced domain
- Added constraint
- Added cut
- Branched
- Cutoff
- Infeasible
- Feasible

# Constraint Enforcement



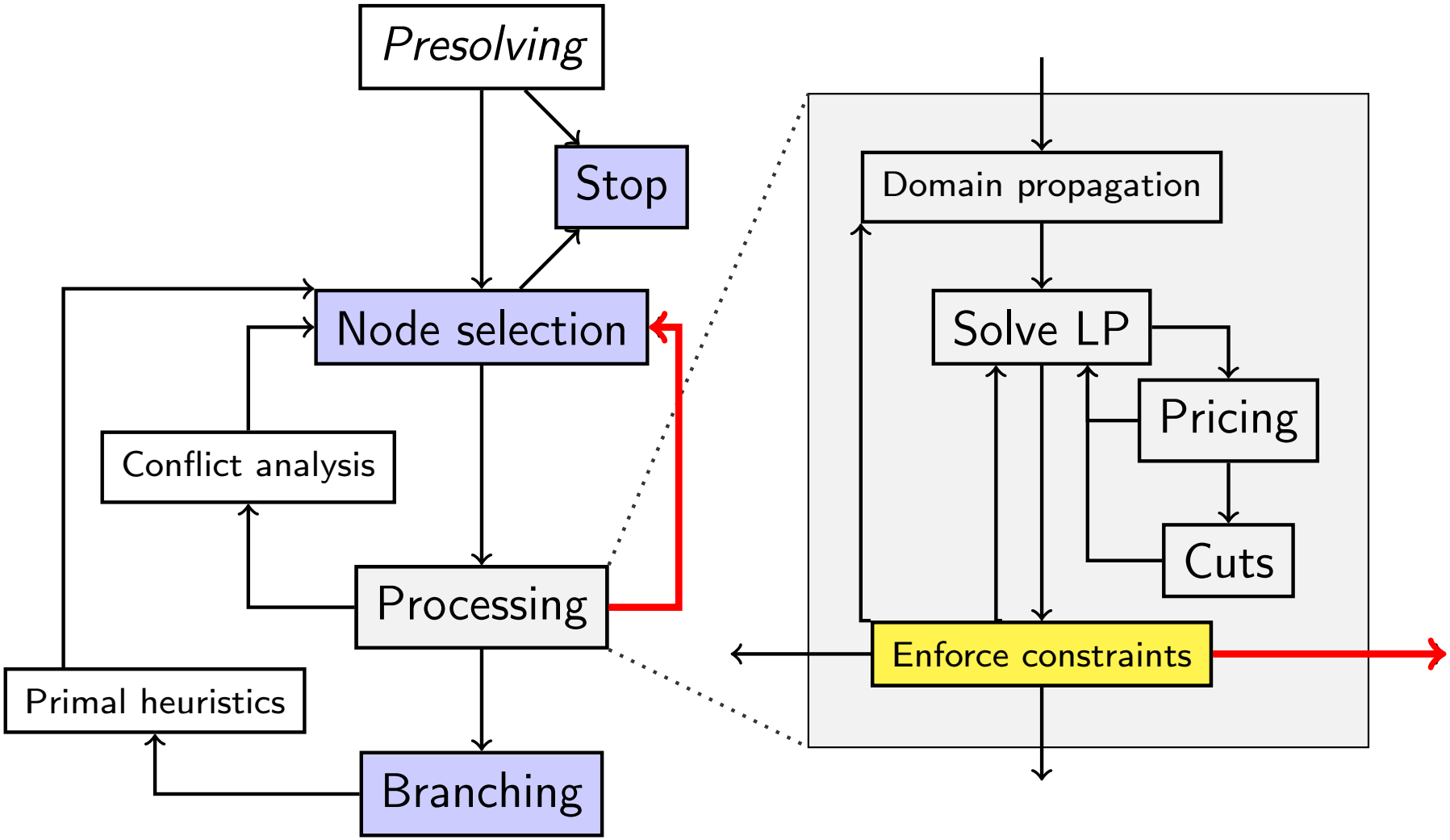

- Reduced domain
- Added constraint
- Added cut
- Branched
- Cutoff
- Infeasible
- Feasible

# Constraint Enforcement



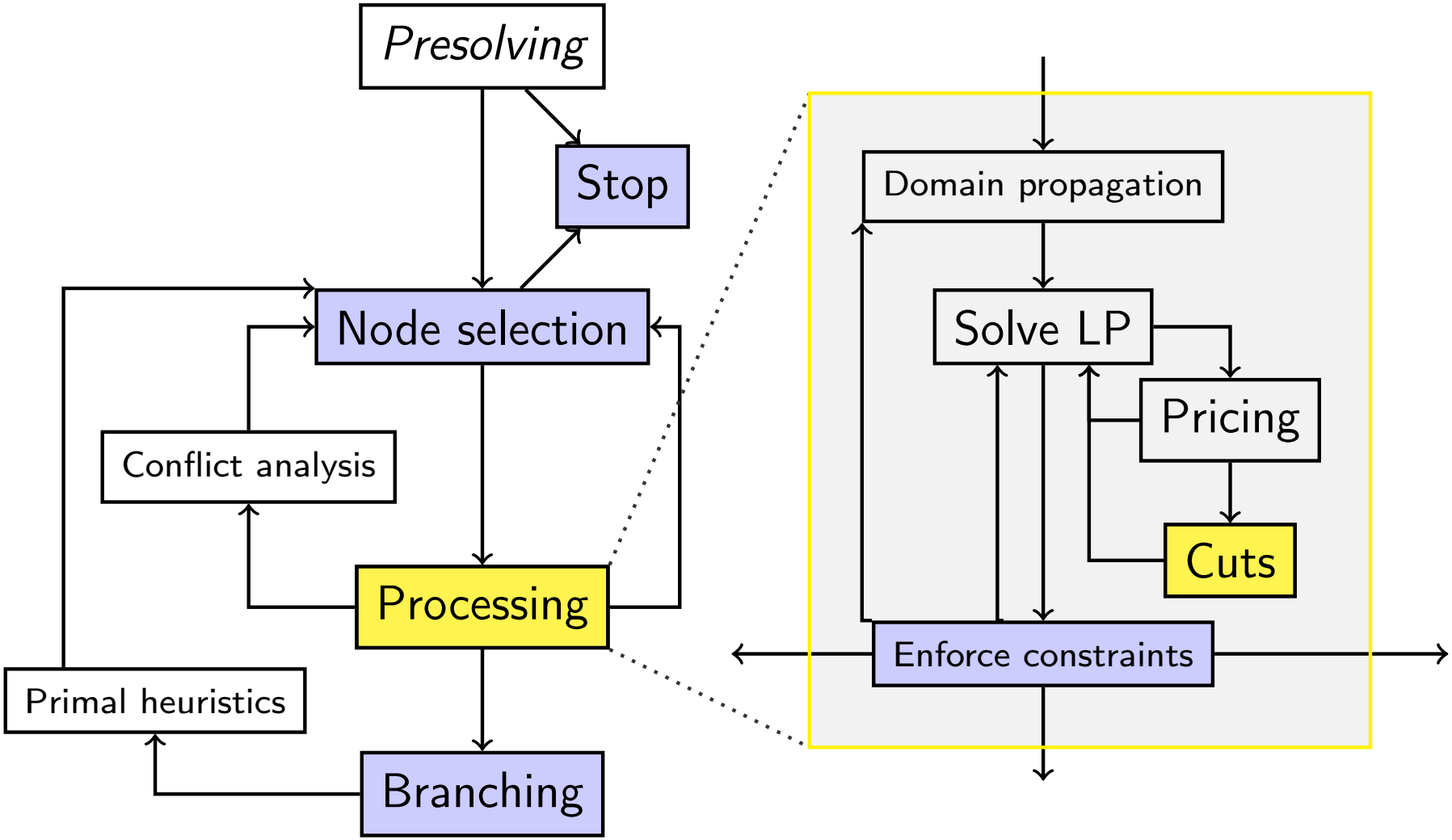

- Reduced domain
- Added constraint
- Added cut
- Branched
- Cutoff
- Infeasible
- Feasible

# Constraint Enforcement

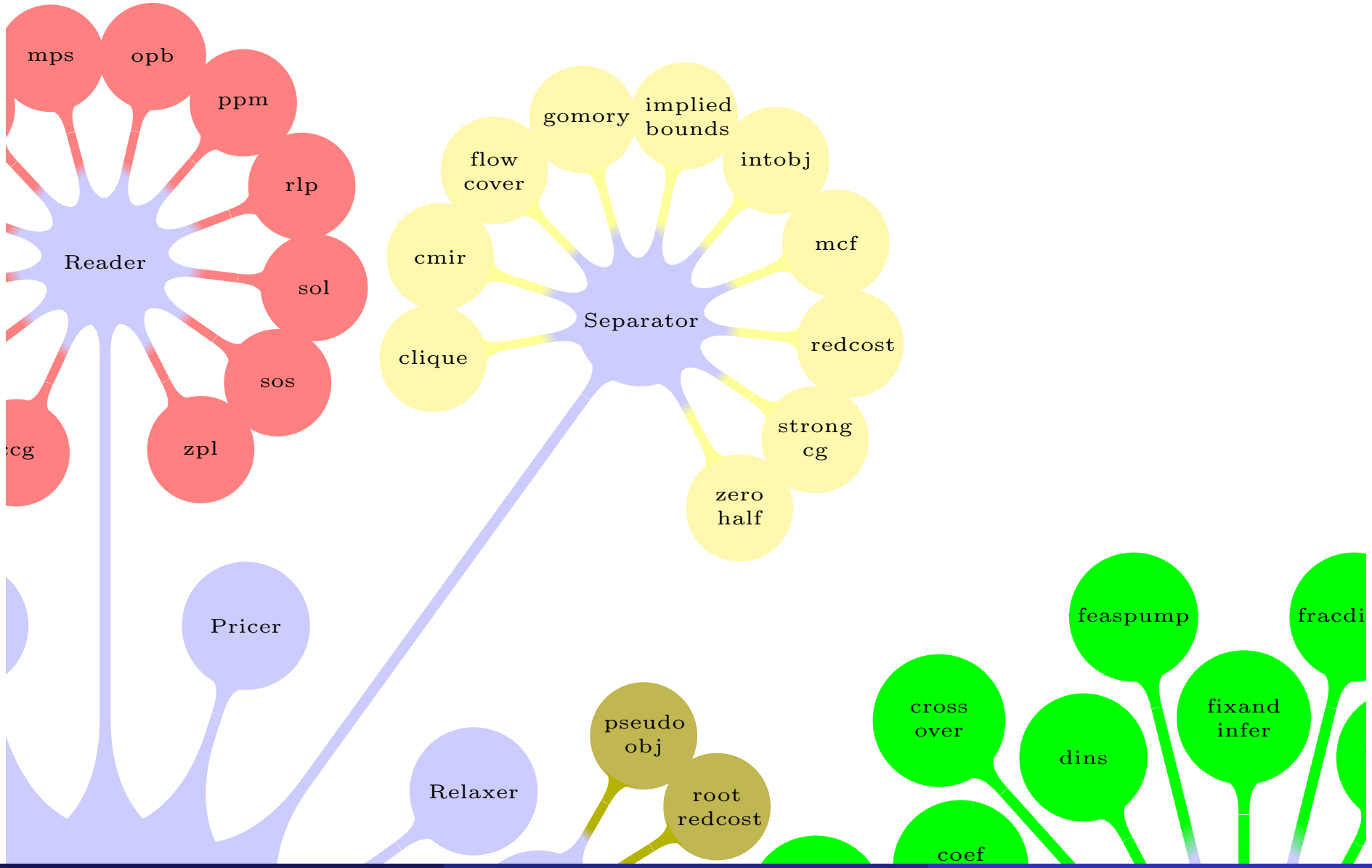



- Reduced domain
- Added constraint
- Added cut
- Branched
- Cutoff
- Infeasible
- Feasible

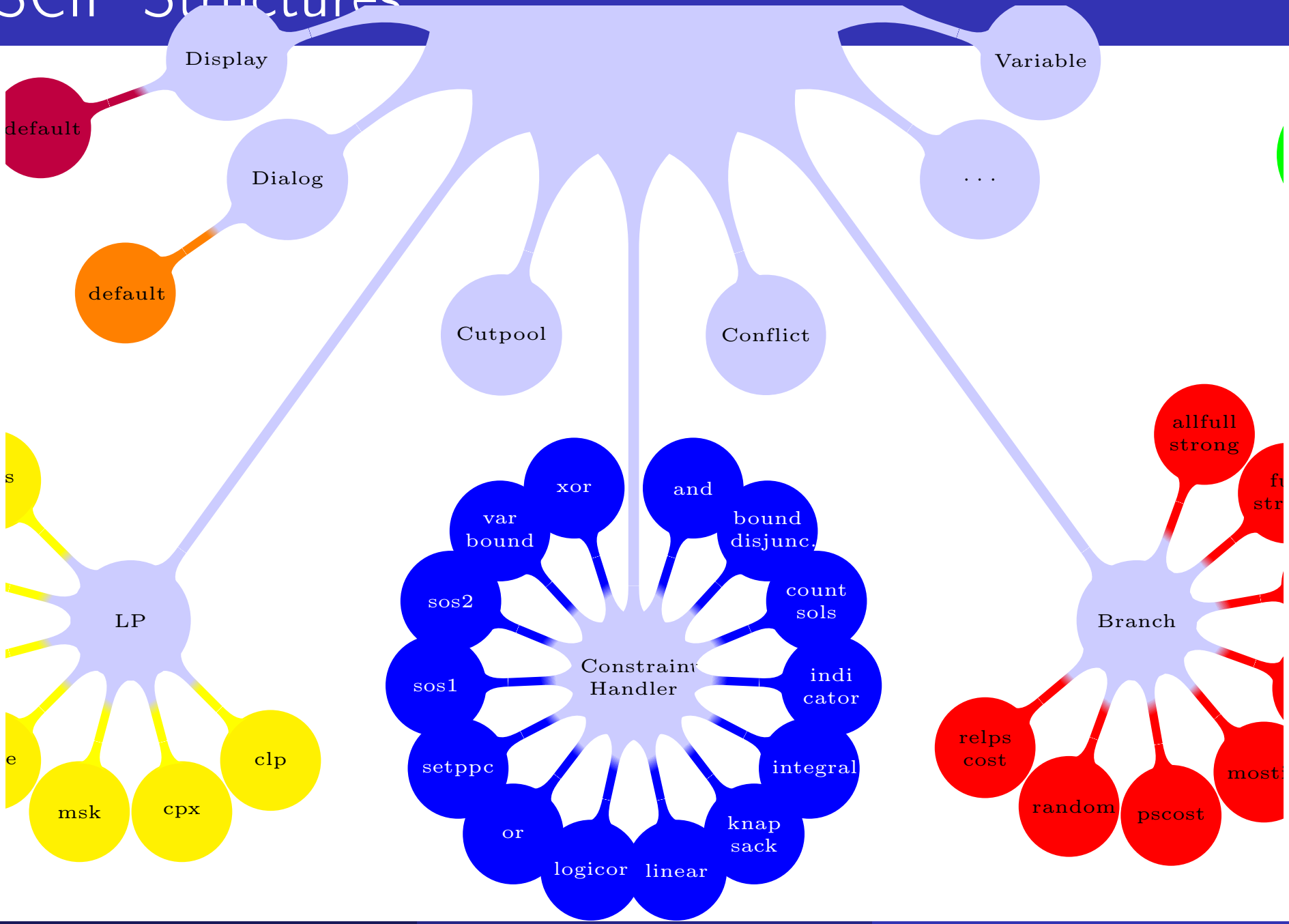
# Flow Chart SCIP



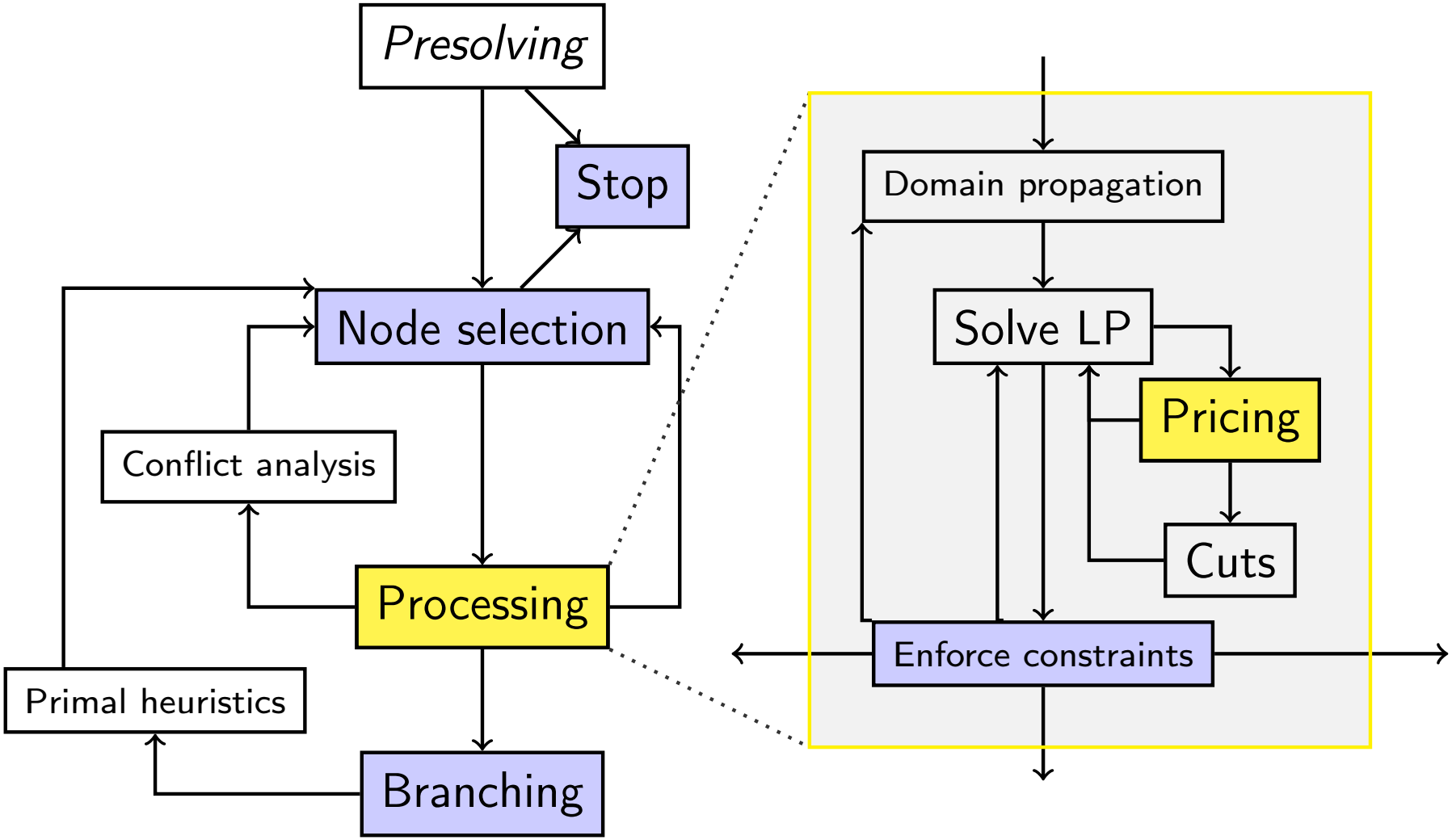
# SCIP Structures



# SCIP Structures

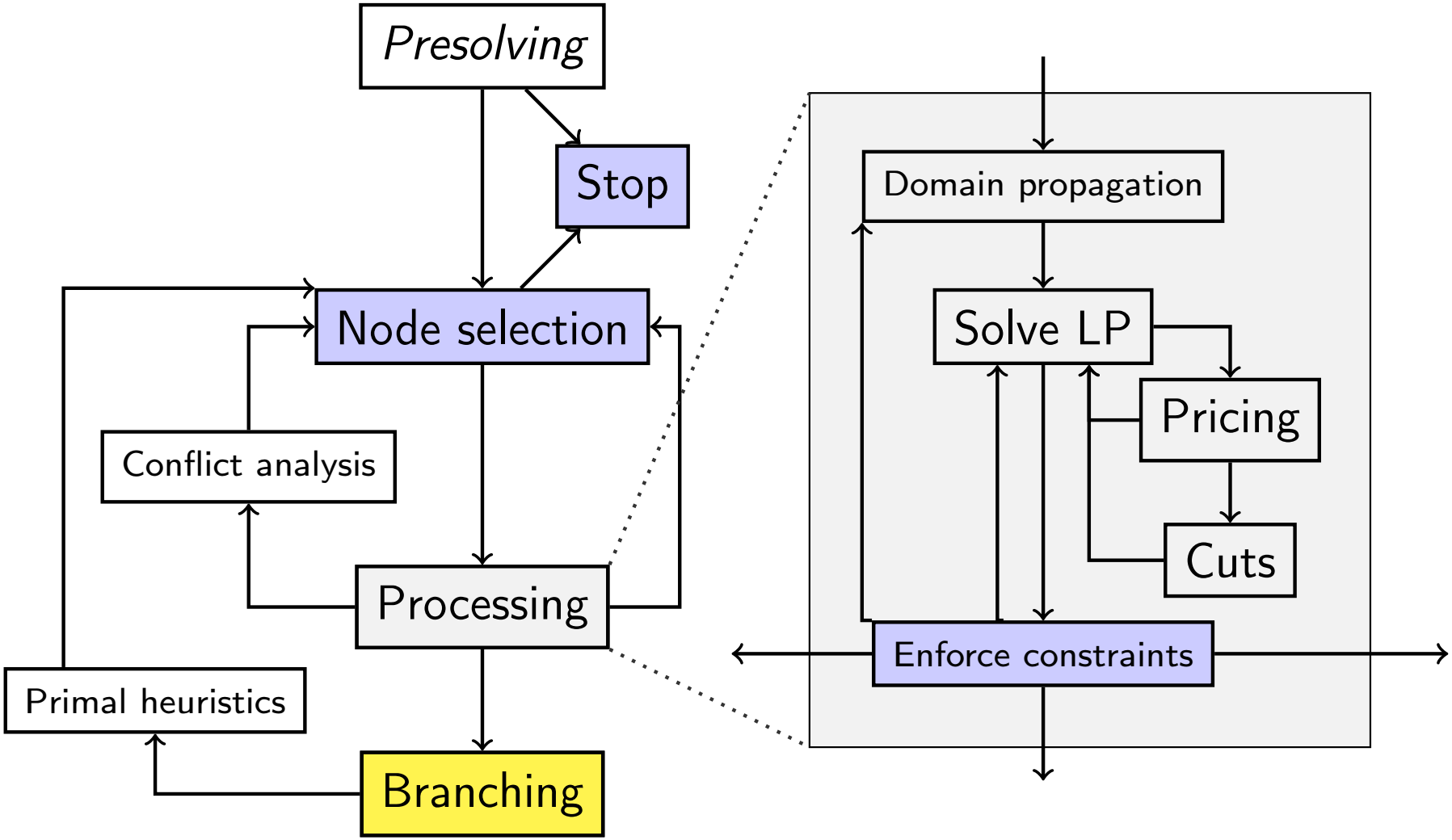


# Flow Chart SCIP



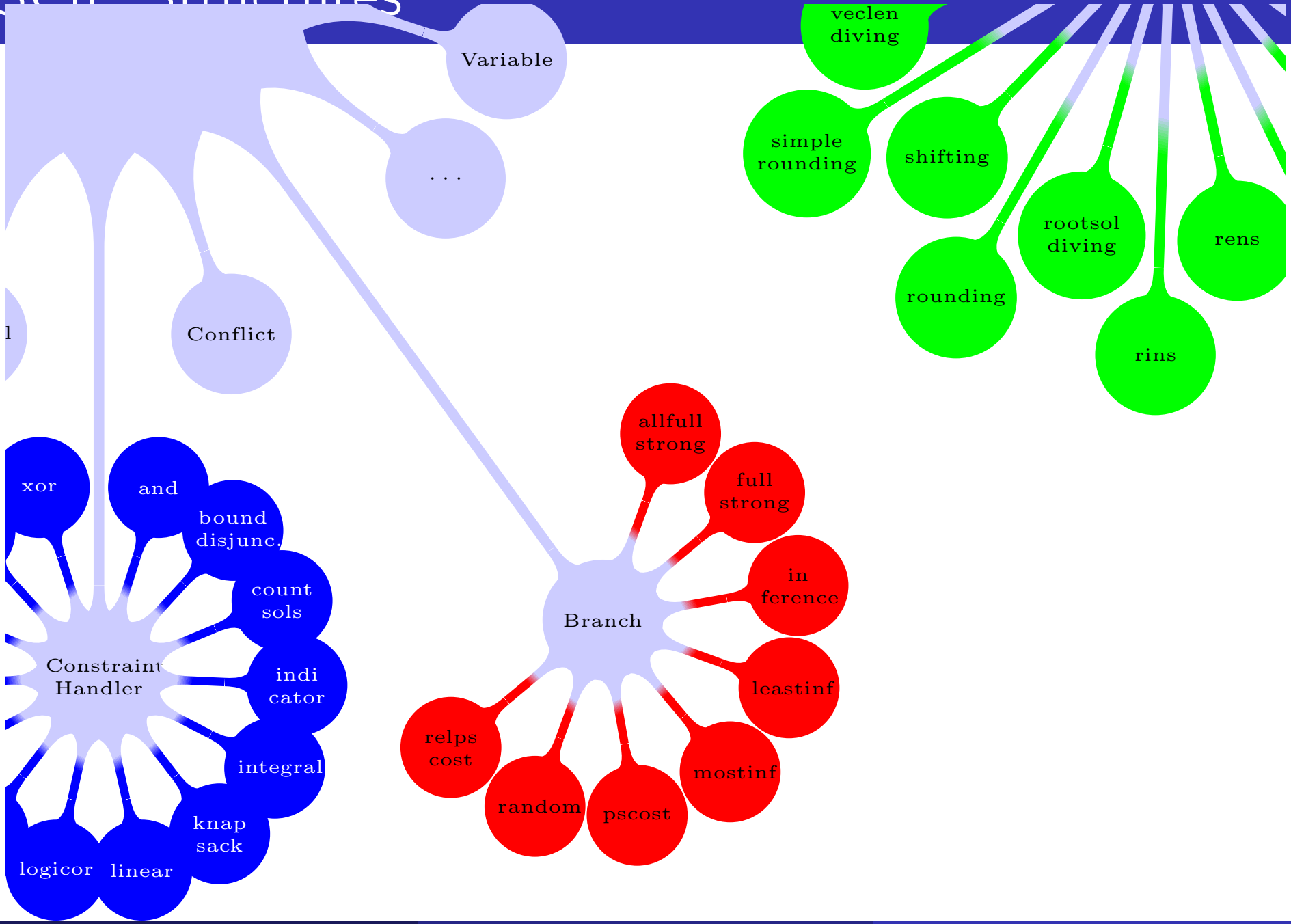

- Pricing: allows dynamic generation of variables

# Flow Chart SCIP

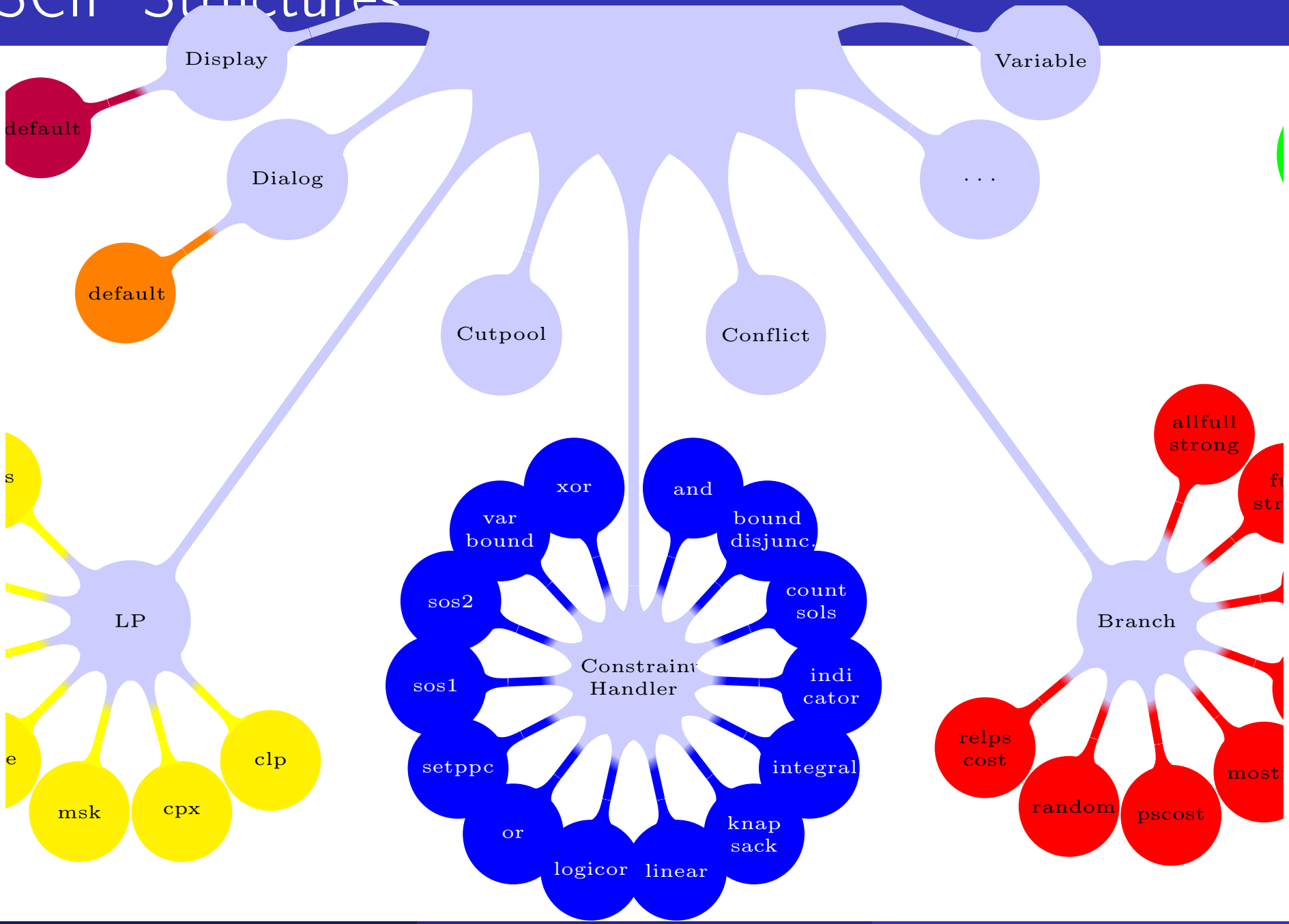



- Creates a branching with the infeasible solution no longer being feasible in the relaxations of the child nodes.

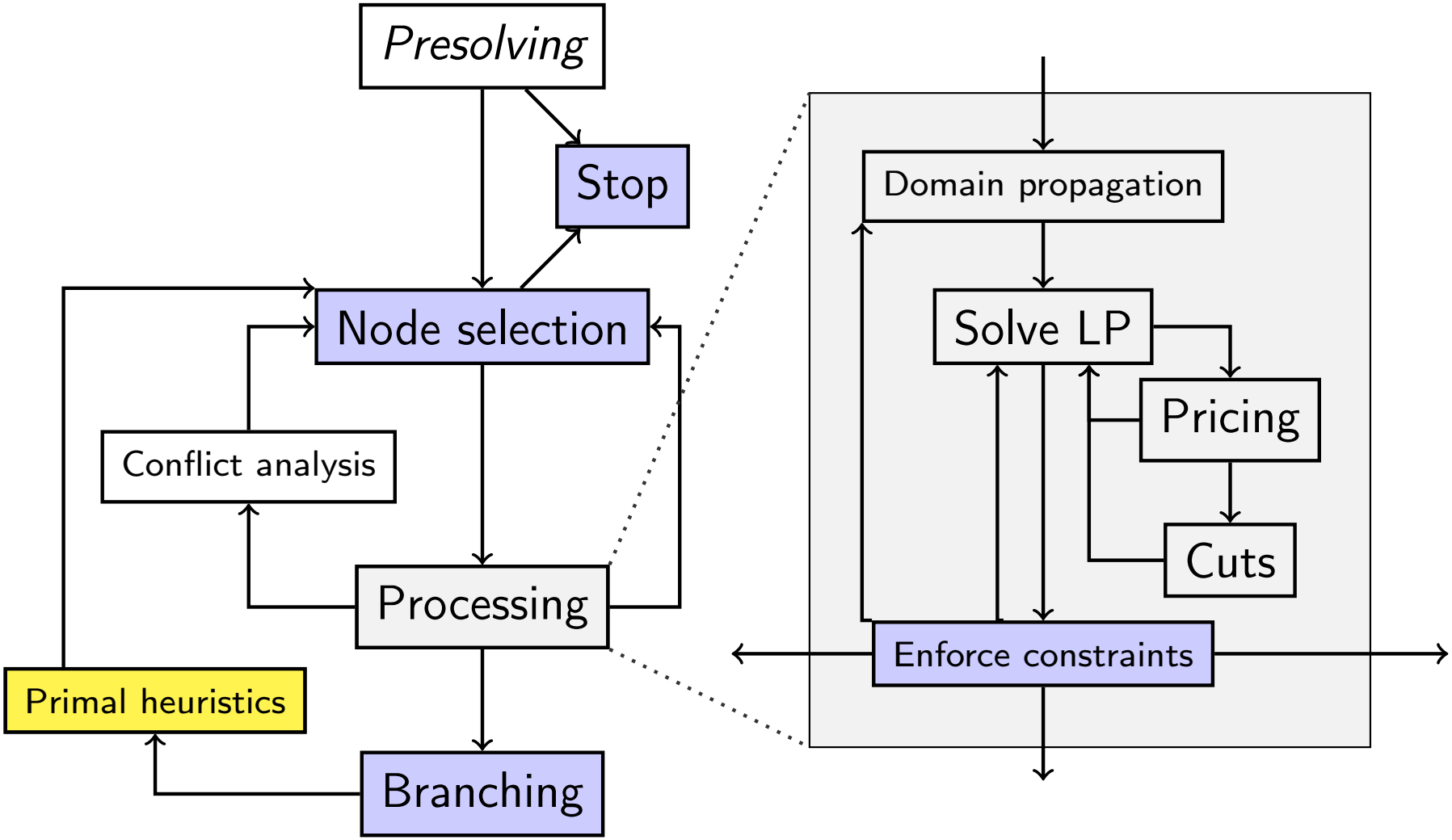
# SCIP Structures



# SCIP Structures

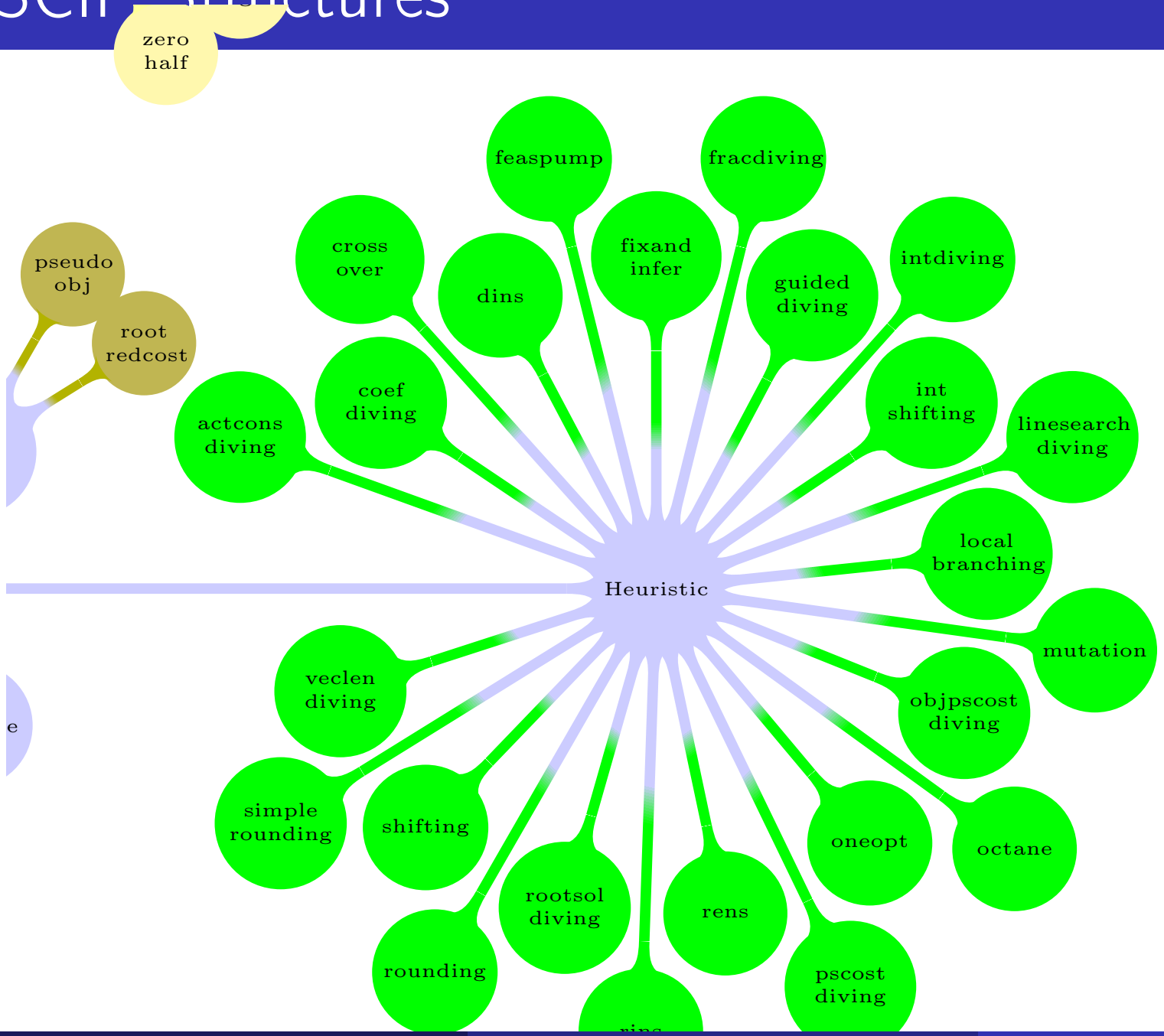


# Flow Chart SCIP

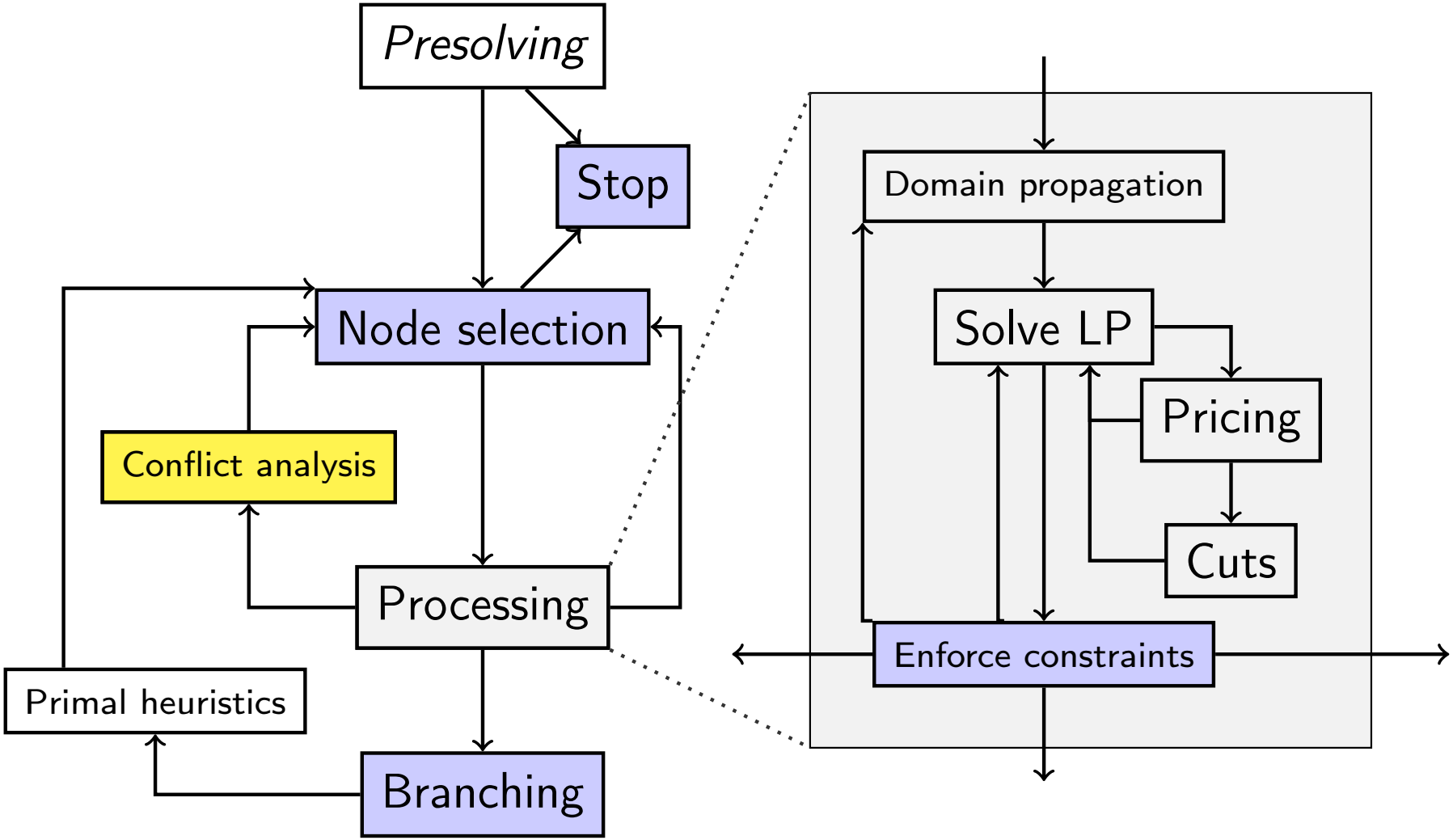



- Primal heuristics try to find feasible solutions (in addition to feasible LP solutions).

# SCIP Structures



# Flow Chart SCIP




- Conflict analysis learns from infeasible subproblems

# Outline

SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

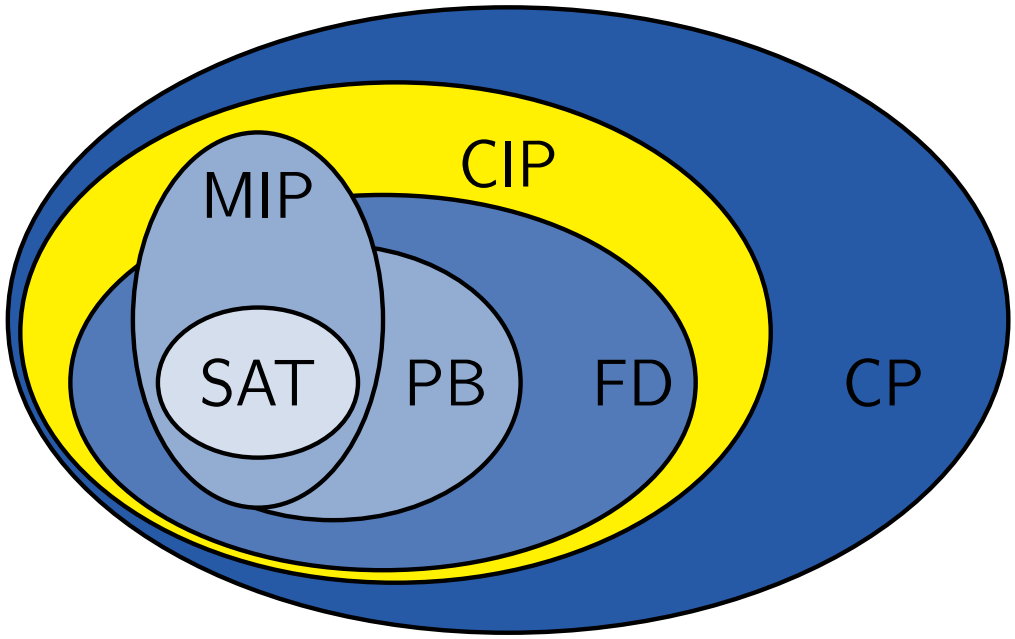
What is SCIP?

Solving Constraint Integer Programs

Summary and Outlook

# Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean
- Finite Domain
- Constraint Programming
- Constraint Integer Programming



# Advantages and Disadvantages of SCIP

## Advantages

- Can be used as a black box MIP solver

# Advantages and Disadvantages of SCIP

## Advantages

- Can be used as a black box MIP solver
- Hundreds of parameters to play with (876)

# Advantages and Disadvantages of SCIP

## Advantages

- Can be used as a black box MIP solver
- Hundreds of parameters to play with (876)
- Robust, fast, and well documented

# Advantages and Disadvantages of SCIP

## Advantages

- Can be used as a black box MIP solver
- Hundreds of parameters to play with (876)
- Robust, fast, and well documented
- Extendable in any direction since constraint based
  - Adding global constraints
  - Adding problem specific plugins

# Advantages and Disadvantages of SCIP

## Advantages

- Can be used as a black box MIP solver
- Hundreds of parameters to play with (876)
- Robust, fast, and well documented
- Extendable in any direction since constraint based
  - Adding global constraints
  - Adding problem specific plugins

## Disadvantages

- Hundreds of parameters to play with (876)
- In general variables do not know in which constraint they appear since constraint based

# Applications

Other universities and institutes using SCIP:

- Universität Bayreuth
- FU Berlin
- TU Berlin
- TU Braunschweig
- TU Chemnitz
- TU Darmstadt
- TU Dresden
- Fachhochschule Lausitz
- OvGU Magdeburg
- TU München
- Universität Osnabrück
- Universität Stuttgart
- Università dell'Aquila
- University College Cork
- Danmarks TU
- Rijksuniversiteit Groningen
- Københavens Universitet
- University of Melbourne
- Monash University
- University of Newcastle
- University of Nottingham
- IASI CNR
- Universidad SF de Quito
- Universidade de Sao Paulo
- ...

# Documentation

- Constraint Integer Programming
  PhD. dissertation of Tobias Achterberg, 2007
- Primal Heuristics for Mixed Integer Programs
  Timo Berthold, Diploma Thesis, TU Berlin, 2006
- Implementation of Cutting Plane Separators for MIPs
  Kati Wolter, Diploma Thesis, TU Berlin, 2006
- CIP: a New Approach to Integrate CP and MIP
  Achterberg, Berthold, Koch, and Wolter, Proceedings CPAIOR08
- CIP: Techniques and Applications
  Achterberg, Berthold, Heinz, Koch, and Wolter, submitted

- `http://scip.zib.de` Doxygen documentation, HowTos, FAQ, Additional Examples

# Automatically run test

SCIP comes with a set of scripts which allow automatic testing

```
scip> make test
```

- TEST state test set
- SETTINGS state setting file
- TIME state a time limit
- NODE state a node limit
- MEM state a memory limit
- ...

```
scip> make test TEST=miplib MEM=1024 TIME=3600
```

- Result file summarizing the test run
- Comparison of different runs

See Doxygen documentation for more information

# Visualize the Branch-and-Cut Tree

SCIP can visualize the branch-and-bound tree

```
SCIP> set vbc filename stein45.tree
```

- SCIP creates a file named "stein45.tree"
- File is readable with vbctool
  - vbc – visualization of branching and cut Algorithms
  - Drawing a tree during computational process
  - Emulating a tree growing process after the computation is finished
  - `http://www.informatik.uni-koeln.de/old-ls_juenger/projects/vbctool.html`
  - Just Google for "vbctool"
  - Group of Prof. Dr. Jünger in Cologne (Germany)

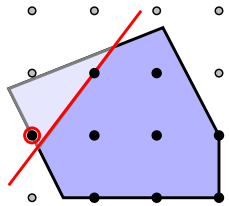
# Future Plans and Research

Future plans:

- More cuts (e.g., odd hole), more heuristics (combinatorial)
- Infeasibility analysis
- Improve column generation
- Scheduling capabilities
- CP constraints (alldifferent, element)
- extend nonlinear support

Research:

- Exact integer programming (Project within DFG program "Algorithm Engineering")
- Mixed integer nonlinear programming (MATHEON)



# Conclusion

Why should you use SCIP?

- You need a full-scale MIP solver
- You want to combine SAT, MIP, and CP features
- You need a solver which is free for academic uses
- You need a solver which is competitive to state-of-the-art solvers
- You want to have total control over the solving process
- You need a solver which is easy to use and well documented

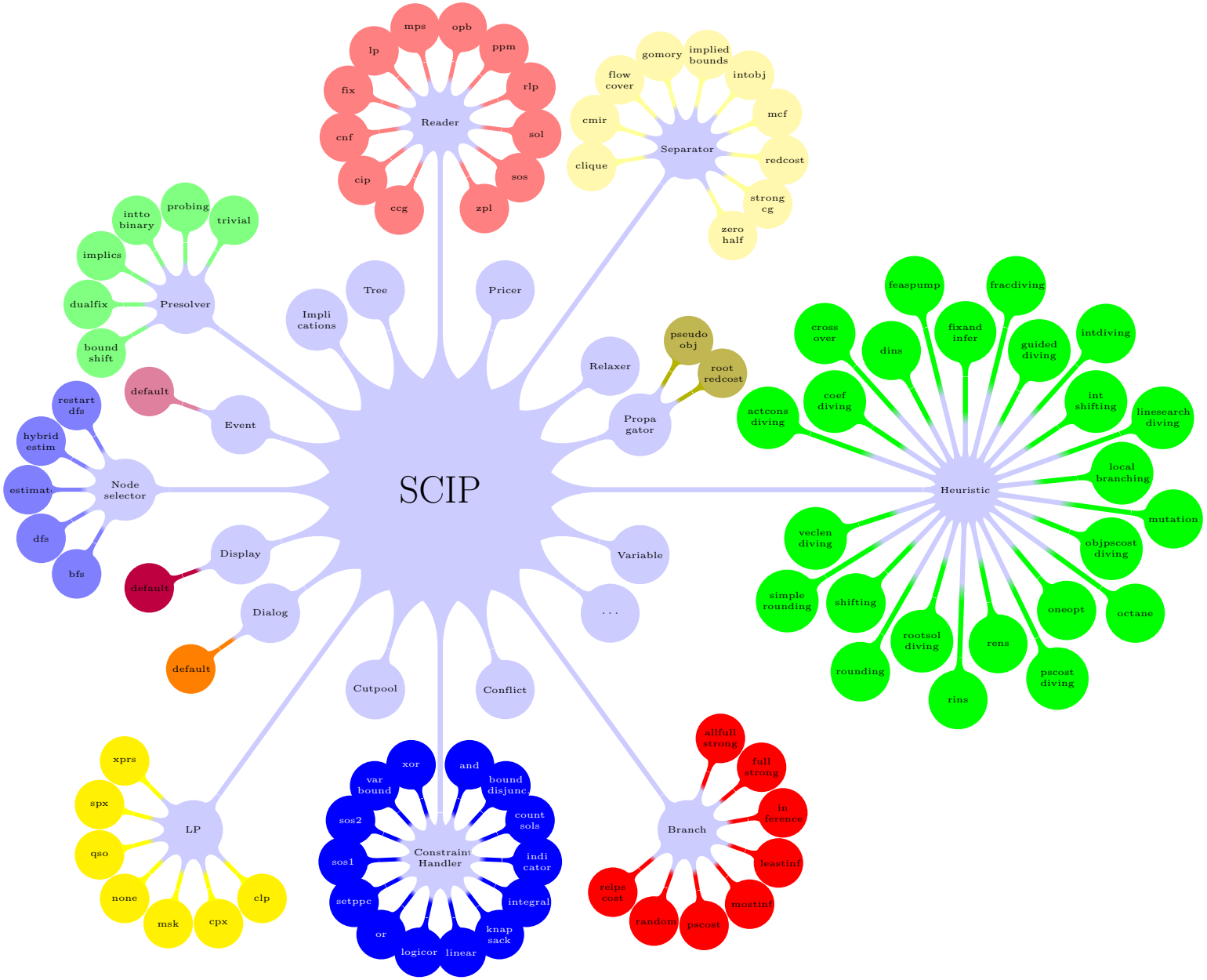
You can contribute by ...

- Using SCIP and report publications/projects,
- Sending bug-reports – better: bug-fixes,
- Giving feedback on documentation,
- Implementing plugins.

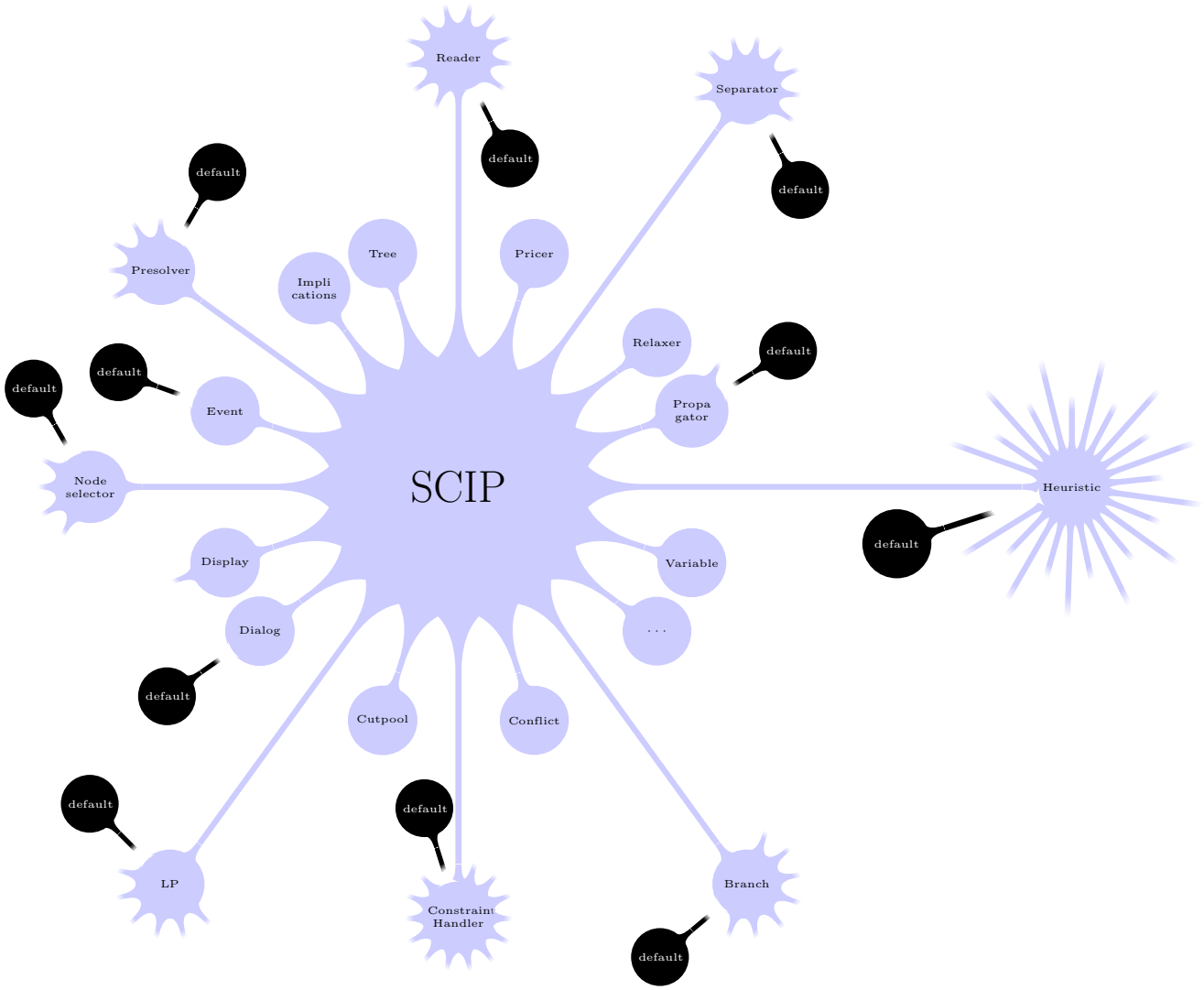
# SCIP at CO@Work

- Tuseday (today):
  - Get ZIB Optimization Suite running
  - Solve a ZIMPL model with SCIP
  - Create a SCIP project
- Saturday (09/26/09):
  - Talks: Node selections, branching rules, and constraint handlers
  - Implementing a constraint handler (TSP)
- Wendsday (09/30/09)
  - Talks: Cuts and heuristics
  - Implementing a heuristics (TSP)
  - Implementing a pricer (Binpacking)

# SCIP Structures



# SCIP Structures



# SCIP Structures