Working with SCIP parameters
SCIP Workshop 2014

Gregor Hendel

Zuse Institute Berlin

SCIP – Working with SCIP parameters

SCIP – Working with SCIP parameters

SCIP Workshop 2014

There are parameters to affect almost every part of the solving process of SCIP.

- SCIP currently features more than 1600 parameters: boolean, integer, real valued, categorical.
- The default settings are an attempt to yield a good performance on a heterogeneous MIP benchmark set.
- In the past, no automated parameter tuning could improve the SCIP performance on a heterogeneous set of instances.

There are parameters to affect almost every part of the solving process of SCIP.

- SCIP currently features more than 1600 parameters: boolean, integer, real valued, categorical.
- The default settings are an attempt to yield a good performance on a heterogeneous MIP benchmark set.
- In the past, no automated parameter tuning could improve the SCIP performance on a heterogeneous set of instances.

How to identify promising parameters by hand? Where to start?

Goals of this exercise

- Get to know the parameter system of SCIP
- Learn to read the statistic output of SCIP
- Learn about emphasis (meta) settings of SCIP
- Start to emphasize/turn off individual plug-ins

SCIP – Working with SCIP parameters

Setting, saving, and loading parameter settings

- Setting a parameter is done via the `set` command in the interactive shell:

```
set limits time 1          # sets the solving time limit to 1 sec.
```

Setting, saving, and loading parameter settings

- Setting a parameter is done via the `set` command in the interactive shell:

```
set limits time 1          # sets the solving time limit to 1 sec.
```

- Two saving modes are available: `set diffsave` and `set save`:

```
set diffsave settings/myset.set   # only non-default params
set save settings/allmyset.set                    # all params
```

Setting, saving, and loading parameter settings

- Setting a parameter is done via the `set` command in the interactive shell:

```
set limits time 1          # sets the solving time limit to 1 sec.
```

- Two saving modes are available: `set diffsave` and `set save`:

```
set diffsave settings/myset.set   # only non-default params
set save settings/allmyset.set                  # all params
```

- Use the `set load`-command to load a settings file:

```
set load settings/myset.set          # loads params from file
```

Setting, saving, and loading parameter settings

- Setting a parameter is done via the `set` command in the interactive shell:

```
set limits time 1          # sets the solving time limit to 1 sec.
```

- Two saving modes are available: `set diffsave` and `set save`:

```
set diffsave settings/myset.set   # only non-default params
set save settings/allmyset.set                  # all params
```

- Use the `set load`-command to load a settings file:

```
set load settings/myset.set          # loads params from file
```

- In order to restore the default settings, type

```
set default                                     # restore default
```

Once you have found the ultimate setting for your instances,

- you can invoke SCIP either directly with

```
bin/scip -s settings/myset.set    # directly loads myset.set,
```

- or save them as `scip.set` in the working directory.

Once you have found the ultimate setting for your instances,

- you can invoke SCIP either directly with

```
bin/scip -s settings/myset.set      # directly loads myset.set,
```

- or save them as `scip.set` in the working directory.

Good practice

It is good practice to

- let settings files end with ".set",
- store them in the settings subdirectory of SCIP,

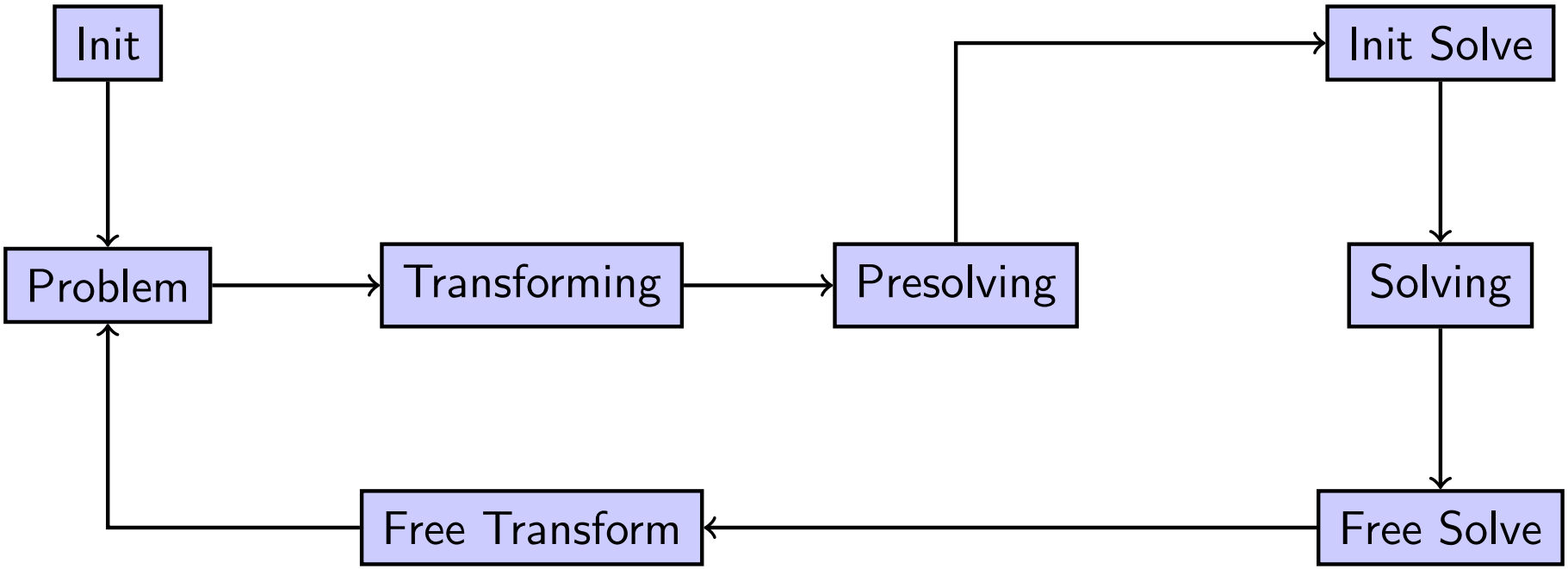
so that you can easily invoke tests by

```
make test SETTINGS=myset            # invoke test with your settings
```

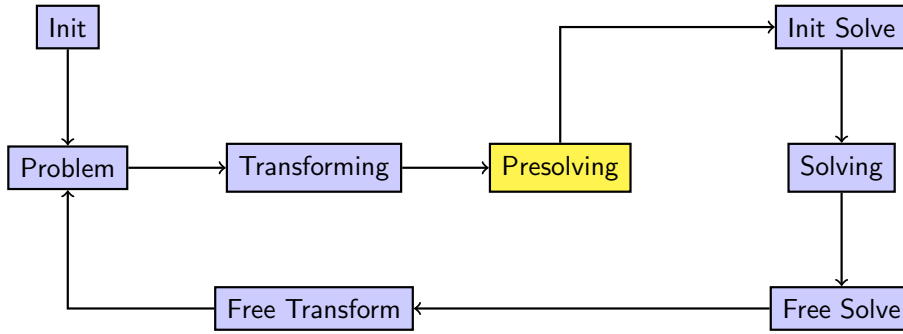
SCIP – Working with SCIP parameters

The SCIP output consists of 3 relevant parts:

- Output for presolving progress
- Periodic status line during branch-and-bound
- Statistics to be displayed after solving was terminated/interrupted



```
(round 33) 26 del vars, 48 del conss, 9 add conss, 269 chg bounds, 3 chg sides, 28 chg co
   (0.0s) probing cycle finished: starting next cycle
(round 34) 48 del vars, 66 del conss, 9 add conss, 269 chg bounds, 3 chg sides, 28 chg co
   (0.1s) probing cycle finished: starting next cycle
presolving (35 rounds):
 48 deleted vars, 66 deleted constraints, 9 added constraints, 269 tightened bounds, 0 ad
 276 implications, 0 cliques
presolved problem has 56 variables (14 bin, 16 int, 0 impl, 26 cont) and 34 constraints
     11 constraints of type <varbound>
      2 constraints of type <setppc>
     21 constraints of type <linear>
Presolving Time: 0.09
```

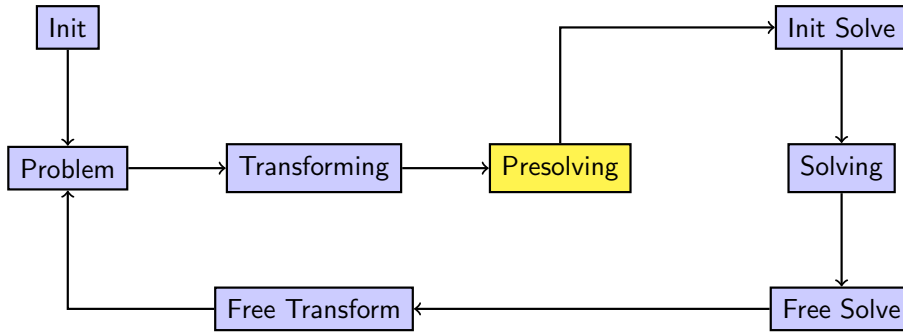


The presolving output consists of

- the number of deleted/added variables and constraints etc. for each round
- A problem summary at the end of presolving
- The presolving time

The presolving output

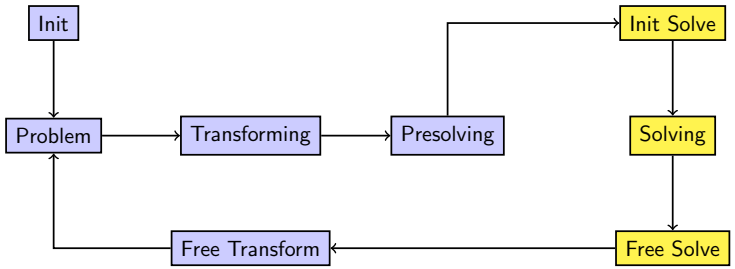
```
(round 33) 26 del vars, 48 del conss, 9 add conss, 269 chg bounds, 3 chg sides, 28 chg co
   (0.0s) probing cycle finished: starting next cycle
(round 34) 48 del vars, 66 del conss, 9 add conss, 269 chg bounds, 3 chg sides, 28 chg co
   (0.1s) probing cycle finished: starting next cycle
presolving (35 rounds):
 48 deleted vars, 66 deleted constraints, 9 added constraints, 269 tightened bounds, 0 ad
 276 implications, 0 cliques
presolved problem has 56 variables (14 bin, 16 int, 0 impl, 26 cont) and 34 constraints
     11 constraints of type <varbound>
      2 constraints of type <setppc>
     21 constraints of type <linear>
Presolving Time: 0.09
```




```
read check/instances/MIP/egout.mps          # read instance
presolve                                    # invoke presolving
```

```
 4191s| 49500 | 13653 |977710 |  19.7 |1004M| 292 |   9 |  23k|7292 |  23k|7337 |   8 | 104 |  15k|
 4197s| 49600 | 13679 |979624 |  19.7 |1005M| 292 |  26 |  23k|7293 |  23k|7337 |   8 | 105 |  15k|
 4209s| 49700 | 13713 |981720 |  19.7 |1007M| 292 |   8 |  23k|7294 |  23k|7337 |   8 | 106 |  15k|
 4215s| 49800 | 13739 |983852 |  19.7 |1009M| 292 |   4 |  23k|7294 |  23k|7337 |   8 | 107 |  15k|
 4223s| 49900 | 13765 |985852 |  19.7 |1011M| 292 |  16 |  23k|7294 |  23k|7337 |   8 | 108 |  15k|
 4230s| 50000 | 13793 |987604 |  19.7 |1012M| 292 |   - |  23k|7294 |  23k|7337 |   8 | 109 |  15k|
 time | node  | left  |LP iter|LP it/n| mem |mdpt |frac |vars |cons |cols |rows |cuts |confs|strbr|
 4237s| 50100 | 13815 |989993 |  19.7 |1015M| 292 |   - |  23k|7294 |  23k|7337 |   8 | 109 |  15k|
 4242s| 50200 | 13831 |991597 |  19.7 |1017M| 292 |  28 |  23k|7294 |  23k|7337 |   8 | 109 |  15k|
*4248s| 50271 |     0 |993090 |  19.7 | 209M| 292 |   - |  23k|7294 |  23k|7337 |   8 | 109 |  15k|

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 4247.93
Solving Nodes      : 50271 (total of 50272 nodes in 2 runs)
Primal Bound       : +5.85000000000000e+04 (319 solutions)
Dual Bound         : +5.85000000000000e+04
Gap                : 0.00 %
```



The solving output consists of periodic status lines showing, e.g.,

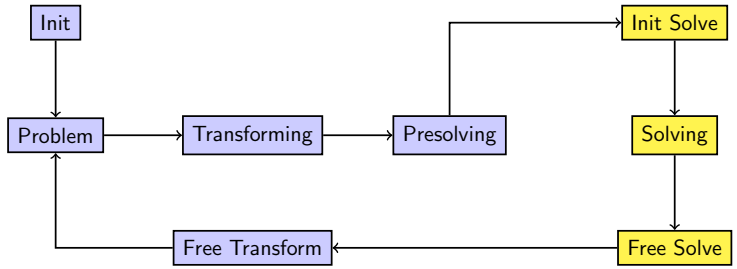
- the elapsed time in seconds
- the number of solved nodes, simplex iterations
- the primal and dual bound, and the gap

This output is customizable (via `set display ...`).

The solving output

```
 4191s| 49500 | 13653 |977710 |  19.7 |1004M| 292 |   9 |  23k|7292 |  23k|7337 |   8 | 104 |  15k|
 4197s| 49600 | 13679 |979624 |  19.7 |1005M| 292 |  26 |  23k|7293 |  23k|7337 |   8 | 105 |  15k|
 4209s| 49700 | 13713 |981720 |  19.7 |1007M| 292 |   8 |  23k|7294 |  23k|7337 |   8 | 106 |  15k|
 4215s| 49800 | 13739 |983852 |  19.7 |1009M| 292 |   4 |  23k|7294 |  23k|7337 |   8 | 107 |  15k|
 4223s| 49900 | 13765 |985852 |  19.7 |1011M| 292 |  16 |  23k|7294 |  23k|7337 |   8 | 108 |  15k|
 4230s| 50000 | 13793 |987604 |  19.7 |1012M| 292 |   - |  23k|7294 |  23k|7337 |   8 | 109 |  15k|
 time | node  | left  |LP iter|LP it/n| mem |mdpt |frac |vars |cons |cols |rows |cuts |confs|strbr|
 4237s| 50100 | 13815 |989993 |  19.7 |1015M| 292 |   - |  23k|7294 |  23k|7337 |   8 | 109 |  15k|
 4242s| 50200 | 13831 |991597 |  19.7 |1017M| 292 |  28 |  23k|7294 |  23k|7337 |   8 | 109 |  15k|
*4248s| 50271 |     0 |993090 |  19.7 | 209M| 292 |   - |  23k|7294 |  23k|7337 |   8 | 109 |  15k|

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 4247.93
Solving Nodes      : 50271 (total of 50272 nodes in 2 runs)
Primal Bound       : +5.85000000000000e+04 (319 solutions)
Dual Bound         : +5.85000000000000e+04
Gap                : 0.00 %
```



`optimize`

 # continue solving the instance

SCIP features a detailed statistical output.

Presolvers	:	ExecTime	SetupTime	Calls	FixedVars	AggrVars	ChgTypes	ChgBounds	AddHoles	DelCons	AddCons	ChgSides	ChgCoefs
boundshift	:	0.00	0.00	0	0	0	0	0	0	0	0	0	0
components	:	0.00	0.00	1	0	0	0	0	0	0	0	0	0
convertinttobin	:	0.00	0.00	0	0	0	0	0	0	0	0	0	0
domcol	:	0.86	0.00	140	68	0	0	0	0	0	0	0	0
dualinfer	:	0.00	0.00	0	0	0	0	0	0	0	0	0	0
gateextraction	:	0.00	0.00	0	0	0	0	0	0	0	0	0	0
implfree	:	0.00	0.00	0	0	0	0	0	0	0	0	0	0
implics	:	0.01	0.00	562	0	0	0	0	0	0	0	0	0
inttobinary	:	0.00	0.00	562	0	0	0	0	0	0	0	0	0
trivial	:	0.03	0.00	562	7284	0	0	0	0	0	0	0	0
dualfix	:	0.04	0.00	562	0	0	0	0	0	0	0	0	0
genvbounds	:	0.00	0.00	0	0	0	0	0	0	0	0	0	0
probing	:	8.31	0.00	3	3333	0	0	0	0	0	0	0	0
pseudoobj	:	0.00	0.00	1	0	0	0	2	0	0	0	0	0
varbound	:	0.00	0.00	557	0	0	0	0	0	0	0	0	0
knapsack	:	0.01	0.00	1	0	0	0	0	0	6	6	6	706
setppc	:	0.17	0.00	557	0	0	0	0	0	9	0	0	0
linear	:	1.69	0.02	183	3065	5	0	3065	0	930	837	97906	125126
logicor	:	0.00	0.00	0	0	0	0	0	0	0	0	0	0
root node	:	-	-	-	1732	-	-	1734	-	-	-	-	-
Constraints	:	**Number**	**MaxNumber**	**#Separate**	**#Propagate**	**#EnfoLP**	**#EnfoPS**	**#Check**	**#ResProp**	**Cutoffs**	**DomReds**	**Cuts**	**Applied**
integral	:	0	0	0	0	10	0	1	0	0	5	0	0
varbound	:	57	57	134	24016	0	0	0	3	0	0	0	0
setppc	:	30	30	134	24016	0	0	0	120	10	715	0	0
linear	:	387+	425	134	24006	0	0	0	1220	3327	15575	169	64
logicor	:	0+	1	0	21	0	0	0	0	0	0	0	0
countsols	:	0	0	0	0	0	0	0	0	0	0	0	0

- can be invoked at any time of the solving process via

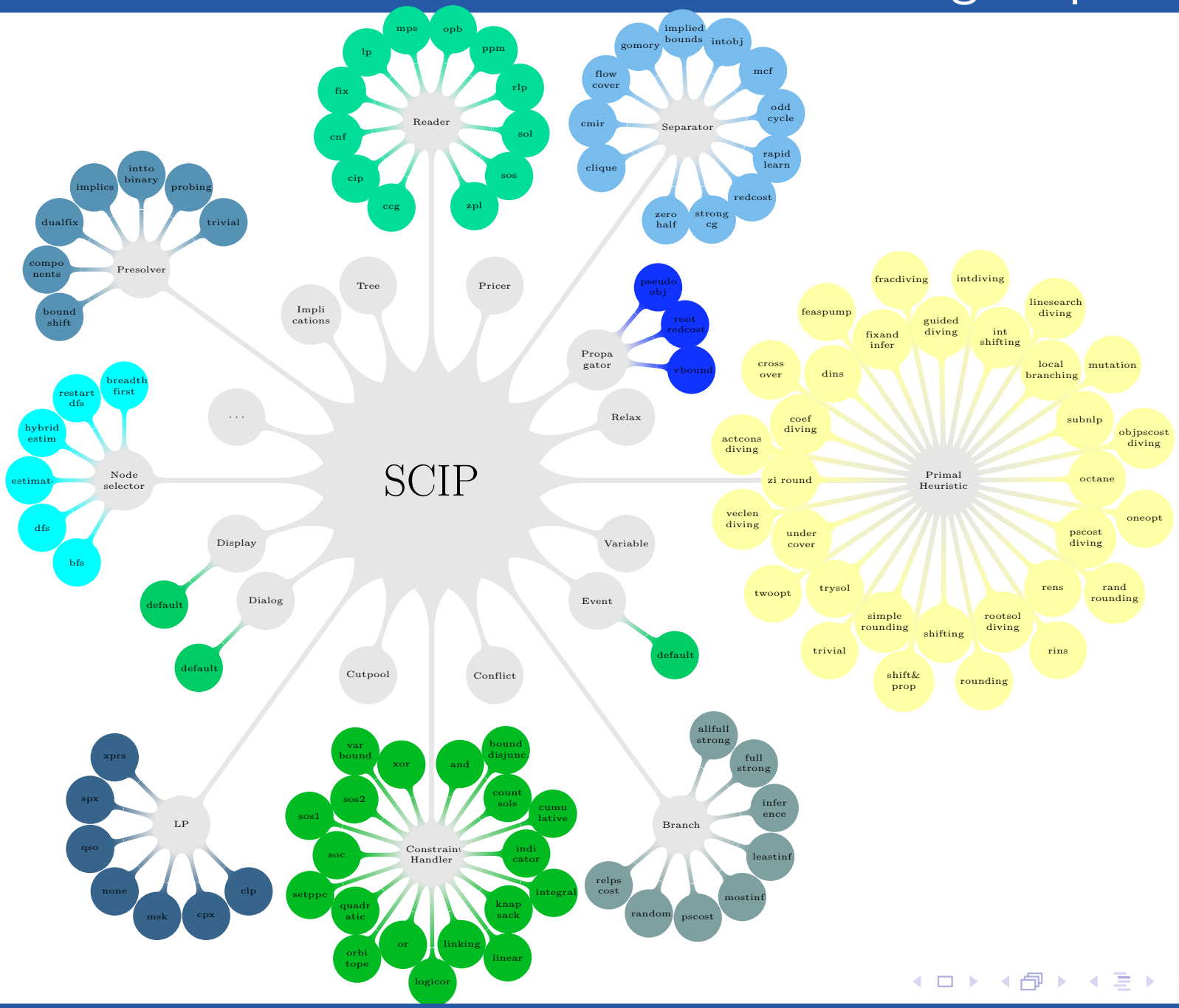
  `display statistics` # *displays solving statistics*

- sorted by plugin types
- displays output for all plugins that are relevant for your problem
- provides information about timing, success, and others

SCIP – Working with SCIP parameters

SCIP Workshop 2014

Presolving emphasis settings



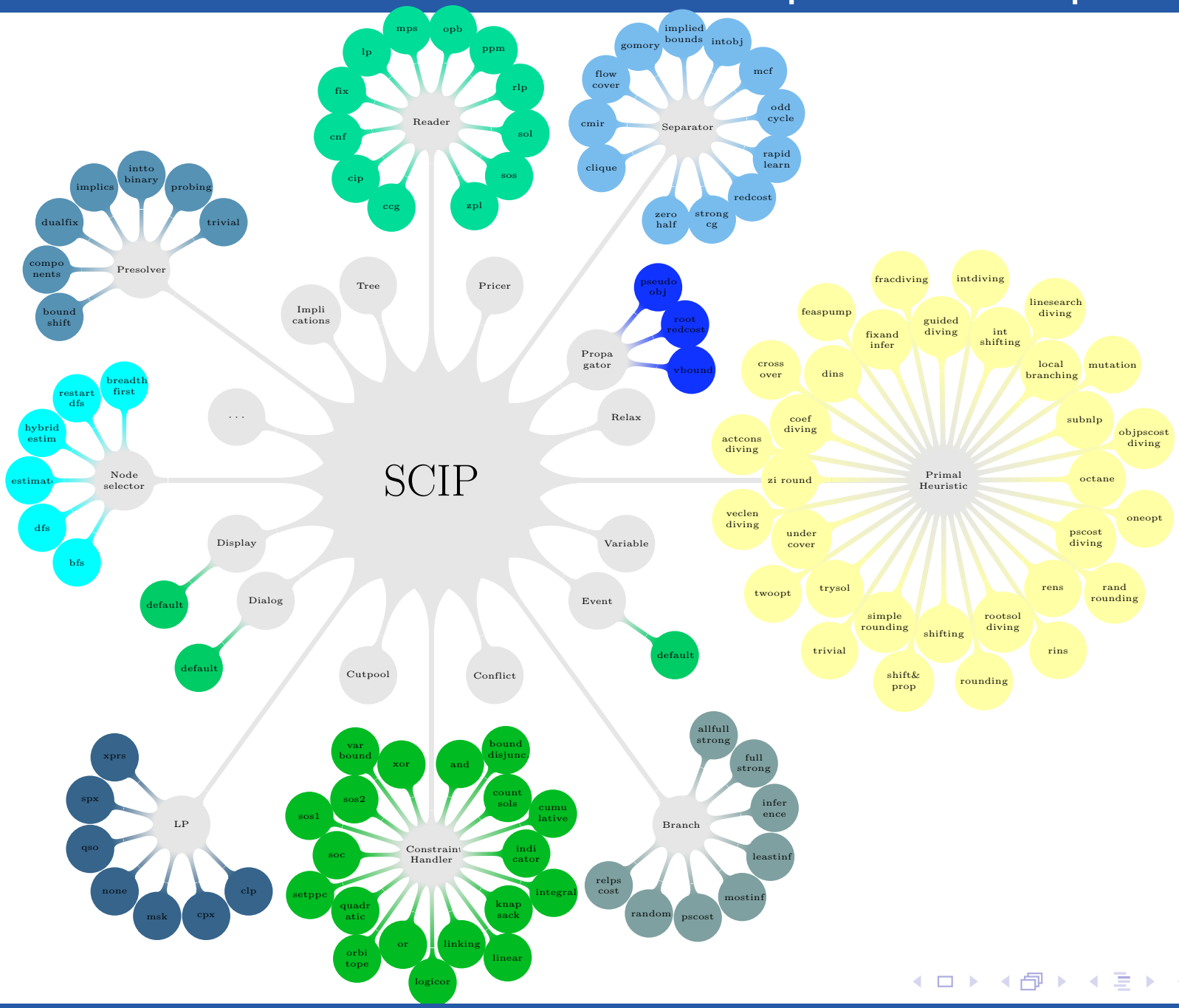
Important parameters for presolvers

- set presolving emphasis: meta-settings for presolving (aggressive, fast, off)
- presolving/maxrounds: maximum number of presolving rounds
- presolving/abortfac: minimum decrease of the problem size per round
- presolving/maxrestarts: maximum number of restarts
- presolving/xyz/maxrounds: maximum number of presolving rounds, the presolver participates in *(propagating/xyz/maxprerounds or constraints/xyz/maxprerounds)*
- presolving/xyz/priority: priority of the presolver (called in decreasing priority) *(propagating/xyz/presolpriority)*
- presolving/xyz/delay: should presolver be delayed if other presolvers found reductions? *(propagating/xyz/presoldelay or constraints/xyz/delaypresol)*

Presolving emphasis settings

- start the SCIP interactive shell:
  `scip-3.1.0.linux.x86_64.gnu.opt.spx`
- read instance: `read instances/MIP/bell5.mps`
  (models fiber optic network design problem)
- perform presolving: `presolve`
- SCIP shows reduction in number of constraints and variables
- more details: `display statistics`
- solve the problem: `optimize`
- disable presolving: `set presolving emphasis off`
- read and solve the problem again

Separation emphasis settings



Important parameters for Separators (1)

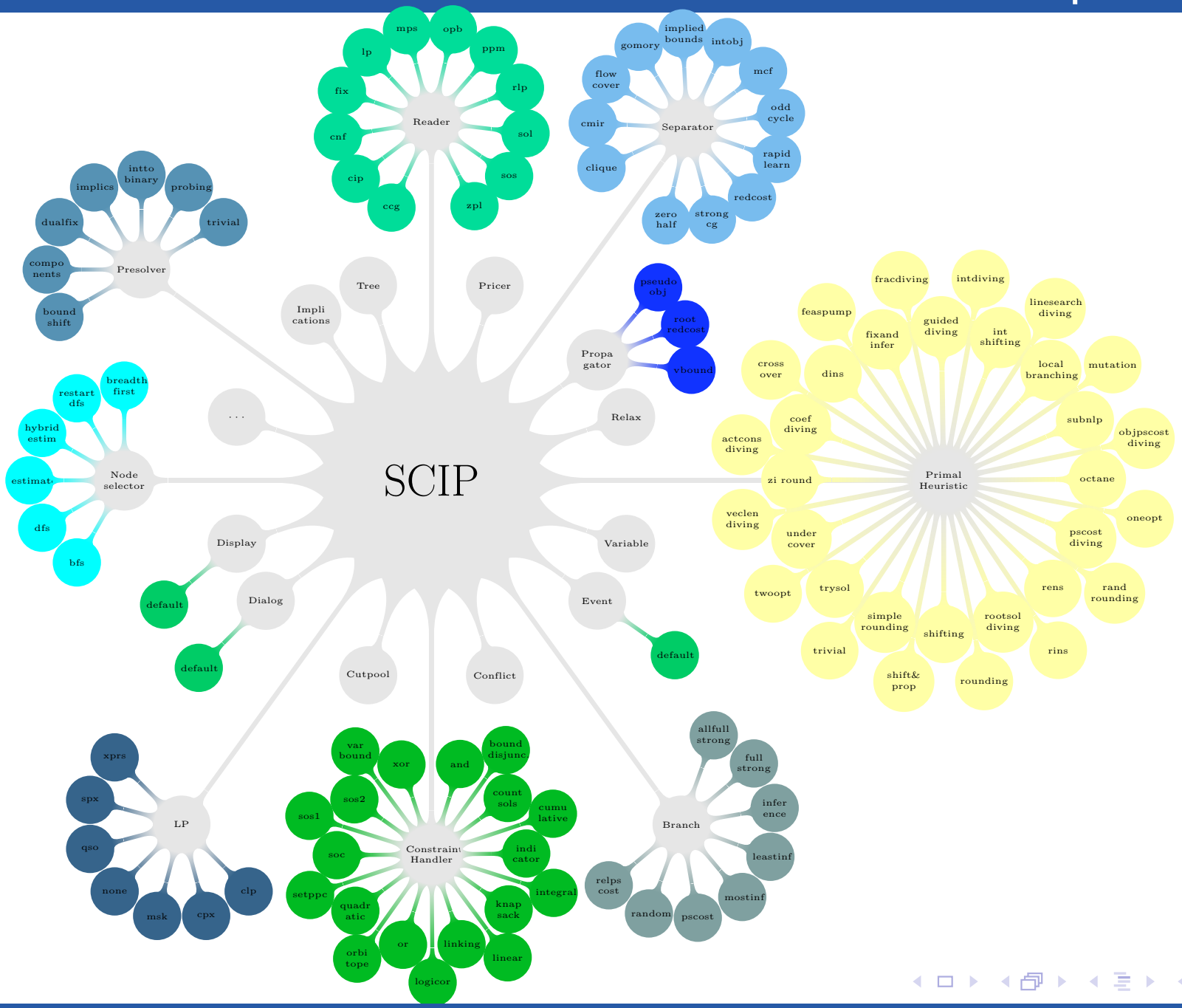
- set separating emphasis: meta-settings for separation (aggressive, fast, off)
- separating/maxrounds(root): max. number of separation rounds
- separating/maxcuts(root): max. number cuts separated per round
- separating/minefficacy(root): minimum efficacy of cut to enter the LP
- separating/minortho(root): minimum orthogonality of cut to enter the LP

Important parameters for Separators (2)

- separating/xyz/freq: frequency of the separator *(constraints/xyz/sepafreq(root))*
- separating/xyz/maxrounds(root): maximum number of separation rounds, the separator participates in *(constraints/xyz/maxrounds(root))*
- separating/xyz/maxsepacuts(root): maximum number of cuts separated by the separator per round *(constraints/xyz/maxsepacuts(root))*
- separating/xyz/priority: priority of the separator (called in decreasing priority)
- separating/xyz/delay: should separator be delayed if other separators found cuts? *(constraints/xyz/delaysepa)*

- read instance: `read instances/MIP/gt2.mps`
  (models truck routing problem)
- solve the problem: `optimize`
- disable cutting plane separation: `set separating emphasis off`
- read and solve the problem again
- compare nodes and solving time

Heuristic emphasis settings



Important parameters for heuristic

- set heuristics emphasis: meta settings for heuristics (aggressive, fast, off)
- heuristics/xyz/freq: frequency of the heuristic
- heuristics/xyz/freqofs: frequency offset of the heuristic
- heuristics/xyz/priority: priority of the heuristic (heuristics are called in decreasing order of priority)
- heuristics/xyz/nodequot: fraction of sub-MIP nodes allowed (LNS heuristics only)
- heuristics/xyz/maxlpiterquot: fraction of diving LP iterations compared to node LP iterations (diving heuristics only)

- read instance: `read check/instances/MIP/bell5.mps`
  (models fiber optic network design problem)
- solve the problem: `optimize`
- single characters at the beginning of a line indicate the heuristic which found a new incumbent
- with `display statistics`, we get a list of heuristics with
  - running time
  - number of calls
  - number of solutions found
- disable primal heuristics: `set heuristics emphasis off`
- read and solve the problem again

Remember

The default settings of SCIP are tuned to yield a good performance on a heterogeneous set of MIP instances.

For certain tasks, different settings might be more appropriate. SCIP has emphasis settings for some of them:

`cpsolver`	predefined parameter settings for CP like search
`easycip`	predefined parameter settings for easy problems
`feasibility`	predefined parameter settings for feasibility problems
`hardlp`	predefined parameter settings for problems with a hard LP
`optimality`	predefined parameter settings for proving optimality fast

SCIP – Working with SCIP parameters

SCIP Workshop 2014

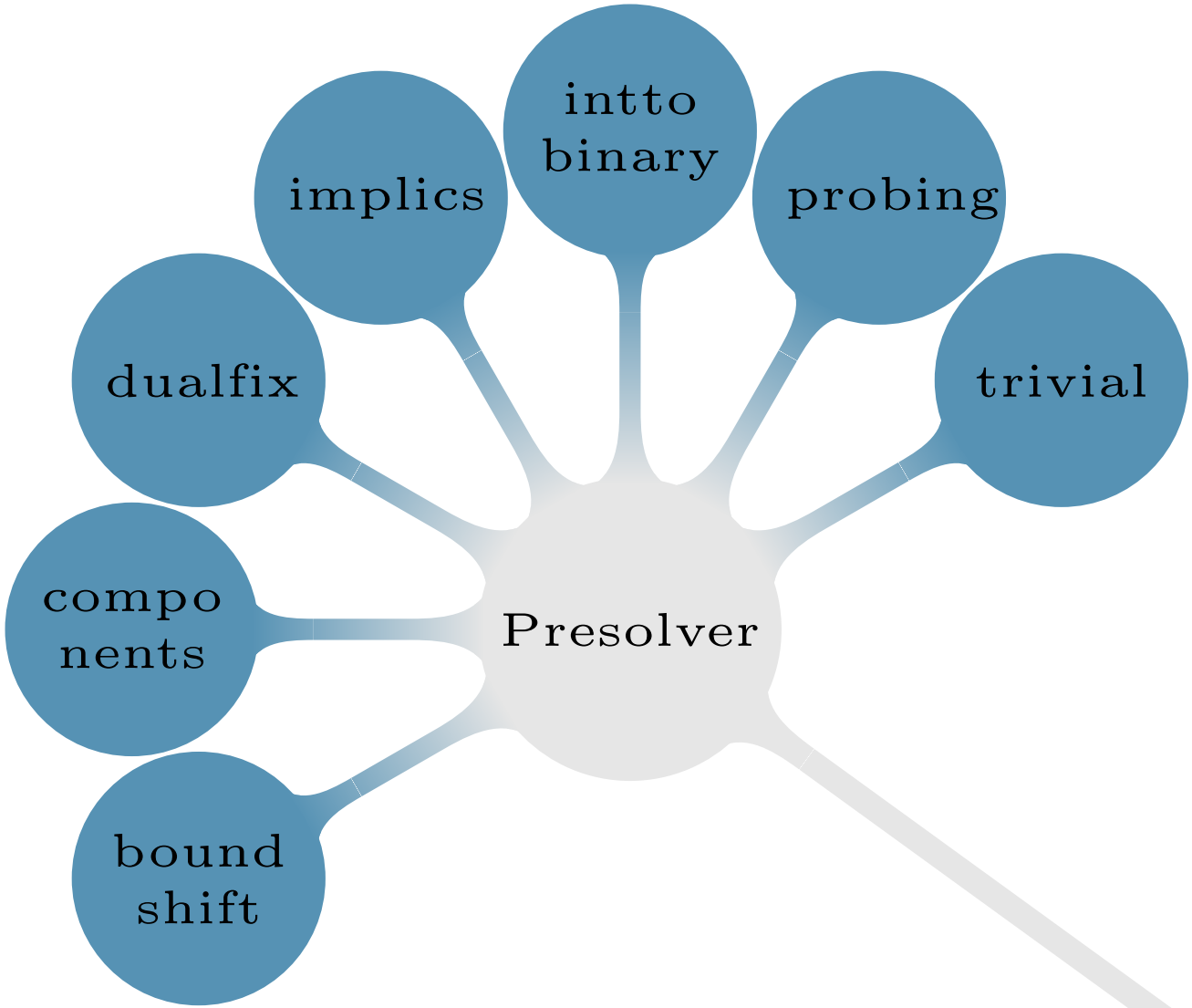
1. Get the two MIP instances `acc-tight5` and `ns1688347` from the MIPLIB 2010, either from one of our USB-devices, or online from `miplib.zib.de/download/acc-tight5.mps.gz`, `miplib.zib.de/download/ns1688347.mps.gz`.
2. For each of these two instances, try to solve them within a time limit of 3 minutes.
3. Can you spot why SCIP could not finish solving the instance?
4. Can you find an emphasis setting to speed up the solving process? Which settings did you use?

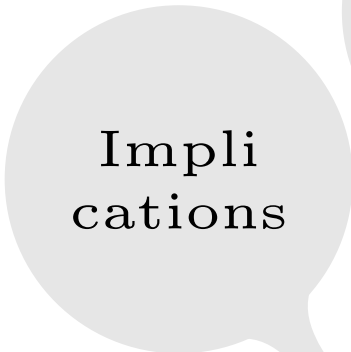
- SCIP comes with default parameters which yield a good performance on "the average instance"
- The output of SCIP can give hints which plugin type was particularly useful/useless.
- Emphasis settings are provided to control many parameters at once.

Thank you very much for your attention

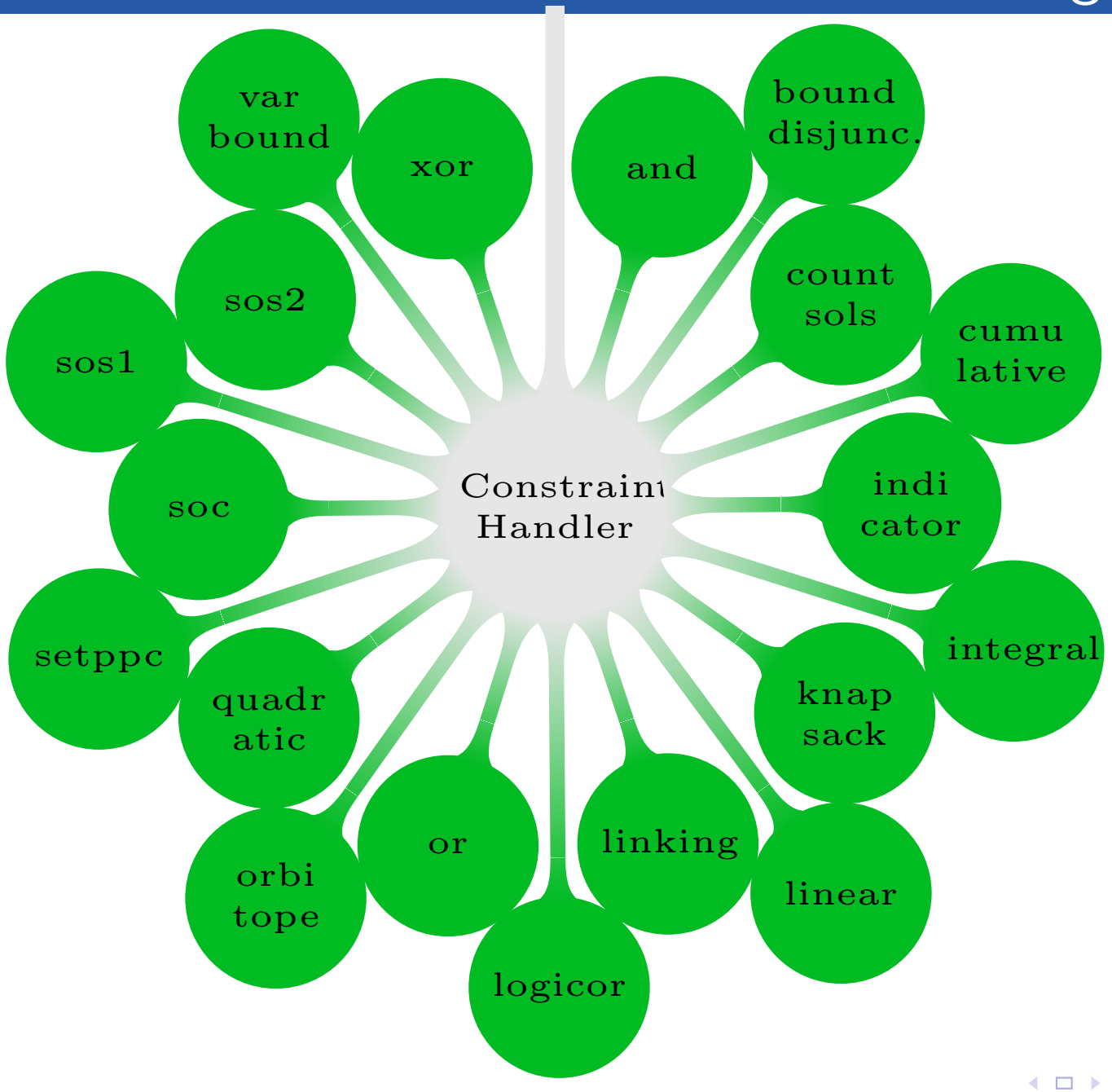
Next: Programming Exercise

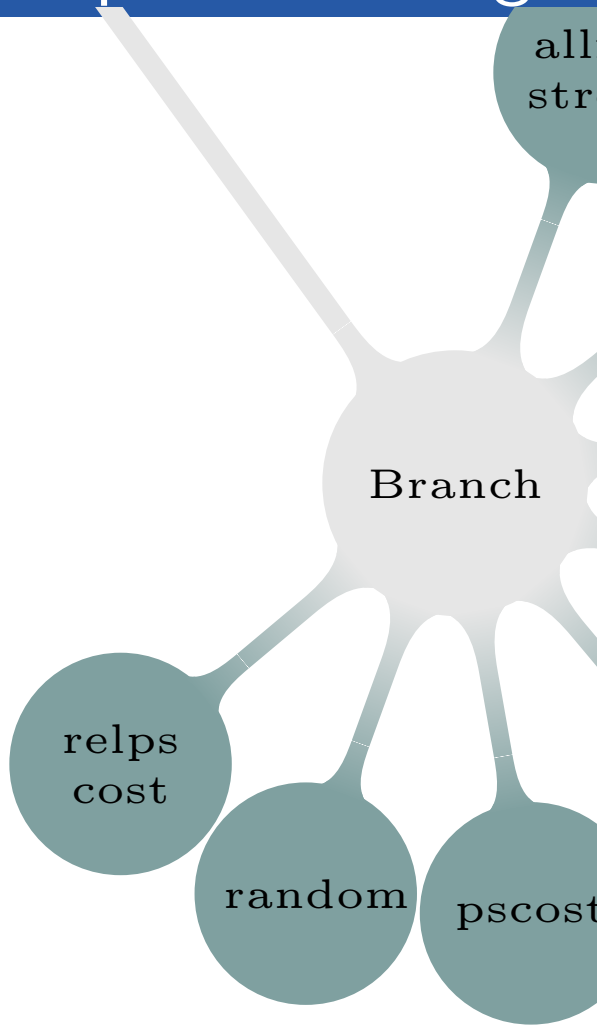
Presolving emphasis settings



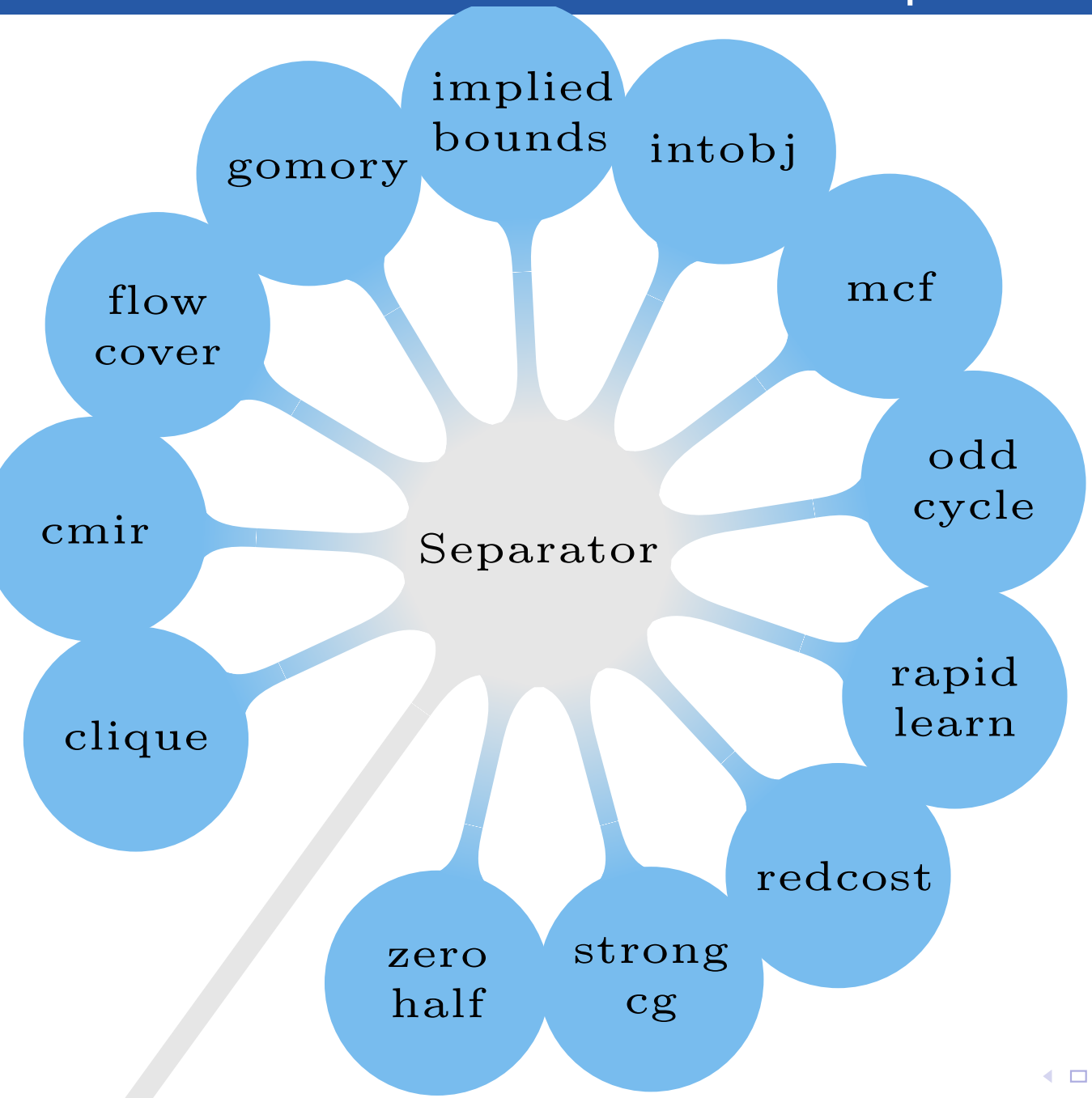


Presolving emphasis settings





Separation emphasis settings



Heuristic emphasis settings