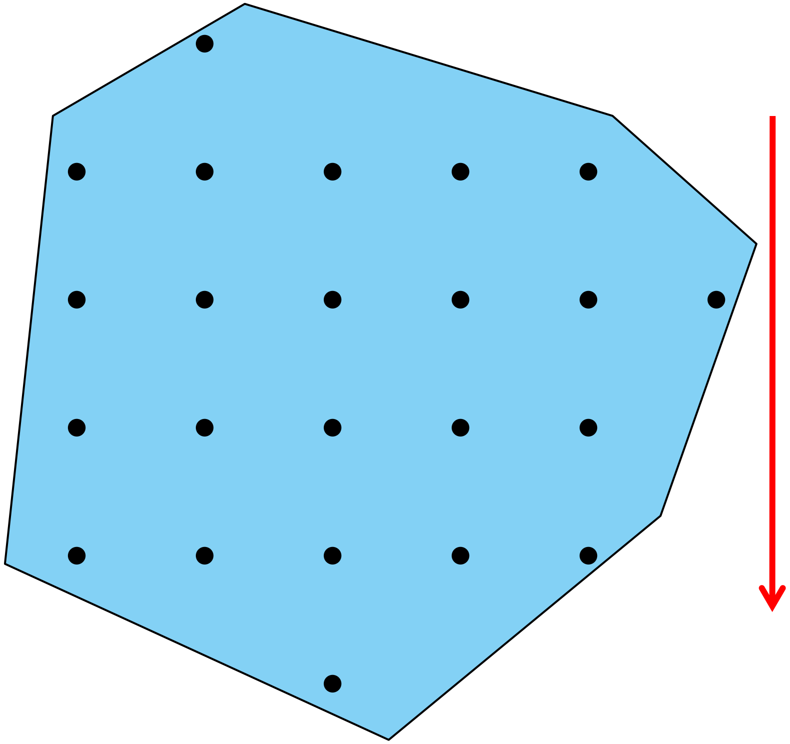
An Overview of CPLEX Mixed Integer Linear Programming Branch-and-Cut

@CPLEX school 2017, Montreal

Mixed Integer linear Programming

- A Mixed Integer (linear) Program (MIP) is a problem of the form

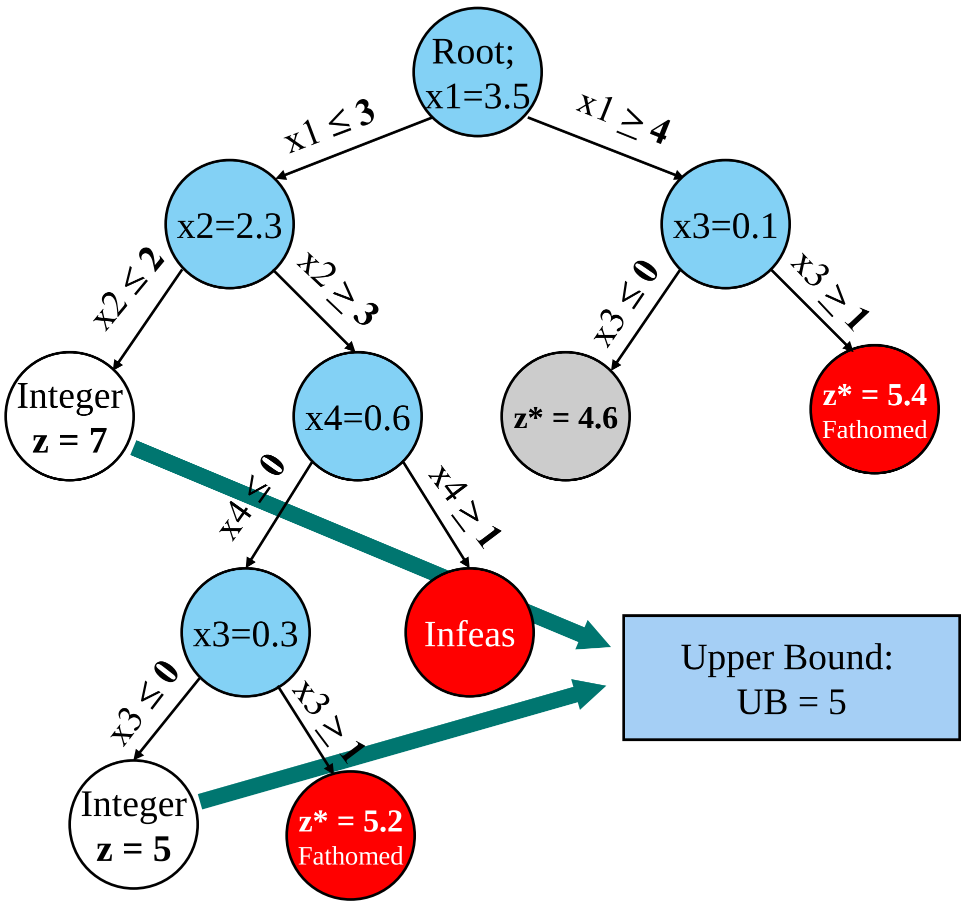
$$
\begin{aligned}
(MIP) \quad & Minimize \quad && z = c^T x \\
& Subject\ to && Ax = b \\
& && l \le x \le u \\
& && \text{some or all } x_j \text{ integer}
\end{aligned}
$$



Some MIP real-world applications

MANUFACTURING	TRANSPORTATION & LOGISTICS	UTILITIES, ENERGY & NATURAL RESOURCES	TELECOM	MULTIPLE/ OTHER
Inventory optimization	Depot/warehouse location	Supply portfolio planning	Network capacity planning	Workforce scheduling
Supply chain network design	Fleet assignment	Power generation scheduling	Routing	Advertising scheduling
Production planning	Network design	Distribution planning	Adaptive network configuration	Marketing campaign optimization
Detailed scheduling	Vehicle & container loading	Water reservoir management	Antenna and concentrator location	Revenue/Yield management
Shipment planning	Vehicle routing & delivery scheduling	Mine operations	Equipment and service configuration	Appointment & field service scheduling
Truck loading	Yard, crew, driver & maintenance scheduling	Timber harvesting		Combinatorial auctions for procurement
Maintenance scheduling	Inventory optimization			

Core of state of the art MIP solvers: LP-based Branch and Bound (B&B)



- B&B algorithm:
  - Enumerative solution scheme based on the LP relaxation of MIP
- **Bounding**:
  - Nodes with **z* > UB** can be **fathomed** without further ramification.
- Key points to avoid exponential explosion of B&B tree:
  - Strong LP relaxation
  - Effective branching rules
  - Primal heuristics

Agenda

- Main building blocks of state of the art MIP solvers
  - Presolve and probing
  - Cutting planes
  - Branching
  - Primal heuristics

- Performance analysis
  - Performance impact of main building blocks

Presolve

- **M.W.P. Savelsbergh**, Preprocessing and probing techniques for mixed integer programming problems, ORSA Journal of Computing 6, 445-454 (1994)

- Transform a problem **P** to a different but equivalent problem **P'**.
  - **Reduce problem size**
    - Speed-up linear algebra during the solution process
  - **Strengthen the LP relaxation**
  - Identify problem **sub-structures**
    - Cliques, implications, networks, disconnected components, …

- Primal reductions
  - Preserve the set of feasible solutions
  - Bound strengthening, coefficient strengthening, lifting of constraints, aggregation of variables, detection of implied integer variables and implied continuous variables, …

- Dual reductions
  - Preserve optimality, but can eliminate feasible and even optimal solutions
  - Dual fixings, fixing and aggregations based on symmetry, removal of parallel or dominated columns, detection of implied integer variable, …

Probing

- Tentatively set binary variable x to 0 and 1 and propagate fixing
- Inspect implications of $x = 0$ and $x = 1$ to derive globally valid information
  - detection of infeasibility and global fixing of probing variable:
    - $x = 0$ infeasible $\Rightarrow$ $x = 1$
  - global fixings and bound strengthening for implied variables:
    - $x = 0 \rightarrow y \leq u^0$, $x = 1 \rightarrow y \leq u^1 \Rightarrow y \leq \max\{u^0, u^1\}$
  - aggregations:
    - $x = 0 \rightarrow y = l_y$, $x = 1 \rightarrow y = u_y \Rightarrow y = l_y + (u_y - l_y)\, x$
  - implications and cliques:
    - $x = 0 \rightarrow y \leq u^0 \Rightarrow$ store implication in implication (y non-binary) or clique (y binary) table
  - lifting:
    - $ay \leq b$, $x = 1 \rightarrow ay \leq b - d \Rightarrow ay + dx \leq b$ is valid and dominates $ay \leq b$ (for $d > 0$)
- Applied during and after presolve, during root cut loop, and in node presolve
- Can be very time consuming but also very powerful
  - need to have good dynamically adjusted work limits

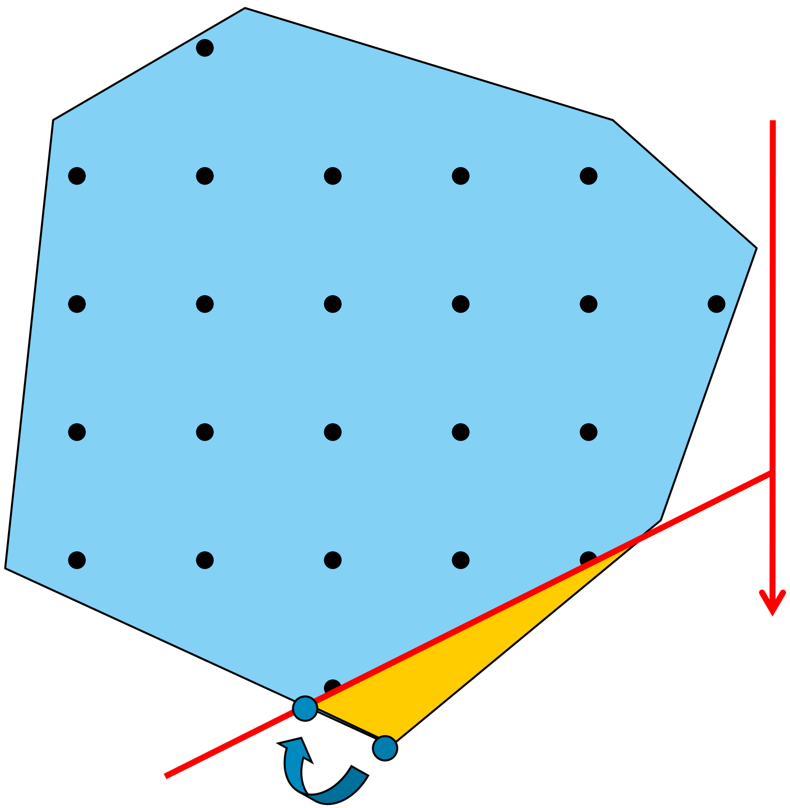
Cutting planes

- Valid inequalities for the MIP that cut off integer infeasible points of the LP relaxation

  – Iteratively separated on the fly to **strengthen the LP relaxation**

  – Separated:
    - At the root node (more aggressively)
    - In the tree (less aggressively)

  – Separation must be combined with **clever cut filtering and cut purging** to avoid
    - Numerical difficulties
    - Node throughput slowdown



The root cut loop in CPLEX

```
x* := optimal solution of the LP relaxation
while (x* not integer and cuts seem effective) {
  heuristics, probing, other secret stuff;
  cut separation;
  cut selection/filtering;
  reoptimization;
  cut purging;
}
```

- Cut separation

  Several families of cuts are separated at the same time.

- Cut filtering (inspired to Andreello et al., 2007)

  Only some of the separated cuts are selected and added to the current formulation:

  - Efficacy
  - Orthogonality

- Cut purging

  After reoptimization, some previously selected cuts may be deemed ineffective and may be discarded.

Cutting planes overview

- General purpose cutting planes in CPLEX
  - Gomory Mixed Integer (GMI) cuts
  - Mixed Integer Rounding (MIR) cuts
  - Lift and Project (L&P) cuts
  - Zero-half cuts
  - Flow cover cuts

- Structural cutting planes in CPLEX
  - Knapsack cover cuts
  - GUB cover cuts
  - Clique cuts
  - Implied bound cuts
  - Multi Commodity Flow (MCF) cuts
  - Flow path cuts

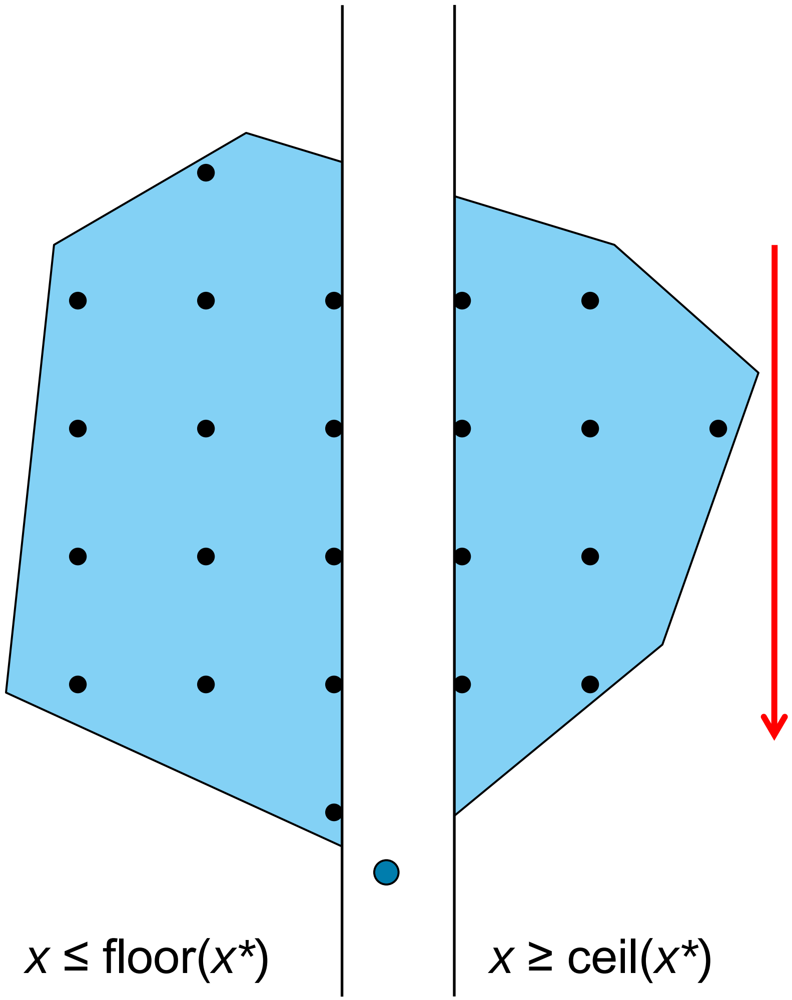
- Survey papers
  - H. Marchand, A. Martin, R. Weismantel, L. Wolsey, Cutting planes in integer and mixed integer programming, Discreate Applied Mathematics 123, 397-446, 2002
  - G. Cornuéjols, Valid inequalities for mixed integer linear programs, Mathematical Programming 112, 3-44, 2008

Separation of general purpose cutting planes

- Generate a **base inequality** by aggregating constraints of LP relaxation
- Apply **rounding formulas** to aggregated inequality to get a cut
- GMI cuts (Gomory, 1960)
  - Base inequality readily available from the **current tableau**
  - Resulting cut is violated by construction
- MIR cuts (Nemhauser & Wolsey, 1988)
  - Base inequalities **heuristically generated** (CPLEX implementation inspired to Marchand & Wolsey, 2001)
  - Resulting cut may be not violated
- L&P cuts (Balas, 1979, Balas et al., 1993)
  - **Solve Cut Generating LP (CGLP**) to get a tableau different from the one readily available
  - **Separate a GMI** cut from the different tableau
  - Resulting cut is violated by construction is CGLP succeeds finding the alternative tableau
  - CPLEX implementation inspired to Bonami (2012)
- They are “just” alternative strategies for separating **split cuts** (Cook et al., 1990)
  - GMI closure = MIR closure = Split closure (see e.g., Nehmauser & Wolsey, 1990)

Branching

- Divides the feasible region in a manner that all integer feasible solutions belong to one of the branches
  - Standard B&B:
    - up and down **branch on integer variables**
  - Branching on general disjunctions, e.g.:
    - Owen & Mehrotra (2001)
    - Mahajan & Ralphs (2009)
    - Karamanov & Cornéjols (2011)



A generic branching rule

- For each fractional variable **x = x***, compute
  - Down score **D(x)**
    - impact of branching down on **x ≤ floor(x*)**
  - Up score **U(x)**
    - impact of branching up on **x ≥ ceil(x*)**
  - Overall score **S (x) = f (D(x), U(x))**
  - Variables with large score **S (x)** are good branching candidates

- **What is an effective rule to compute down and up scores?**
  - Dual bound improvement
  - Child node infeasible or cut-off
  - Help propagation and bound tightening/fixing

- **How to combine down and up score in a single magic number?**
  - S = min {D, U} + μ max {D,U}
  - S = max {D, ϵ} * max {U, ϵ}
  - More elaborated strategies very recently investigated:
    - Le Bodic & Nemhauser (2015)

Famous branching rules

- Strong branching (Applegate et al., 1995)
  - Limited LP solve for each candidate variable
    - For each fractional variable **x = x***, tentatively branch down and up
    - **D(x)** and **U(x)** are the improvement in the objective function
  - Can lead to huge reduction in number of nodes
  - But generally too expensive in practice (two limited LP solve for each candidate)

- Pseudo cost branching (Bénichou et al., 1971)
  - Use historical data to predict impact of a branch
    - Record $\Delta obj/\Delta x$ for each branch
    - Maintain **D(x)** and **U(x)** as average of recorded values

Famous branching rules

- Pseudo cost with strong branching initialization (Linderoth & Savelsbergh, 1999)
  - If pseudo cost not available, initialize it with strong branching.

- Reliability branching (Achterberg et al., 2005)
  - Consider pseudo costs on **x** reliable only if **x** has been branched on **r** times
    - Among fractional variables, identify the ones with unreliable pseudo cost
    - Apply strong branching to some (possibly, all) unreliable candidates to update their pseudo cost and make them reliable
    - Apply **pseudo cost branching** to all candidates with **reliable pseudo cost**

Branching in CPLEX

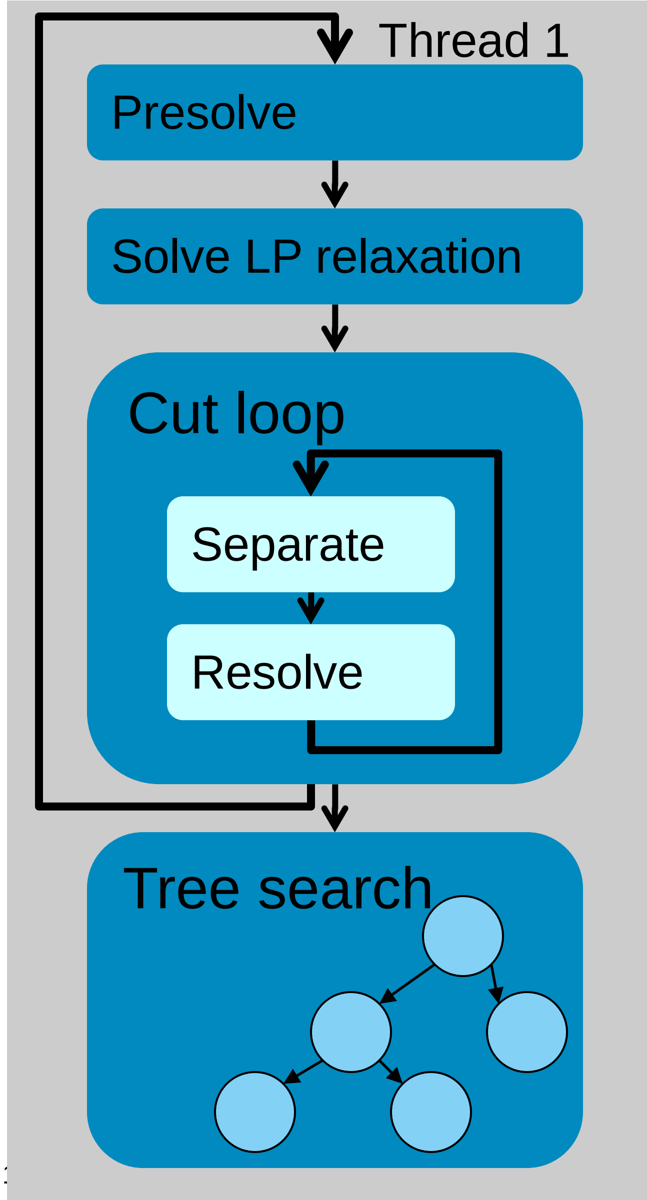
- Default CPLEX strategy is similar to Hybrid branching (Achterberg and Berthold, 2009):
  - Reliability branching
  - Conflict scores
    - Down and up score based on conflict table
    - Idea: prefer branching candidates that are more likely to yield infeasible nodes after branching.
  - Pseudo reduced cost branching (similar to Patel and Chinneck, 2007)
    - Estimate branching impact on the dual bound from the dual solution
  - Inference scores
    - Down and up scores based on clique table and implication table
    - Idea: prefer branching candidates that allow more propagation

Primal heuristics

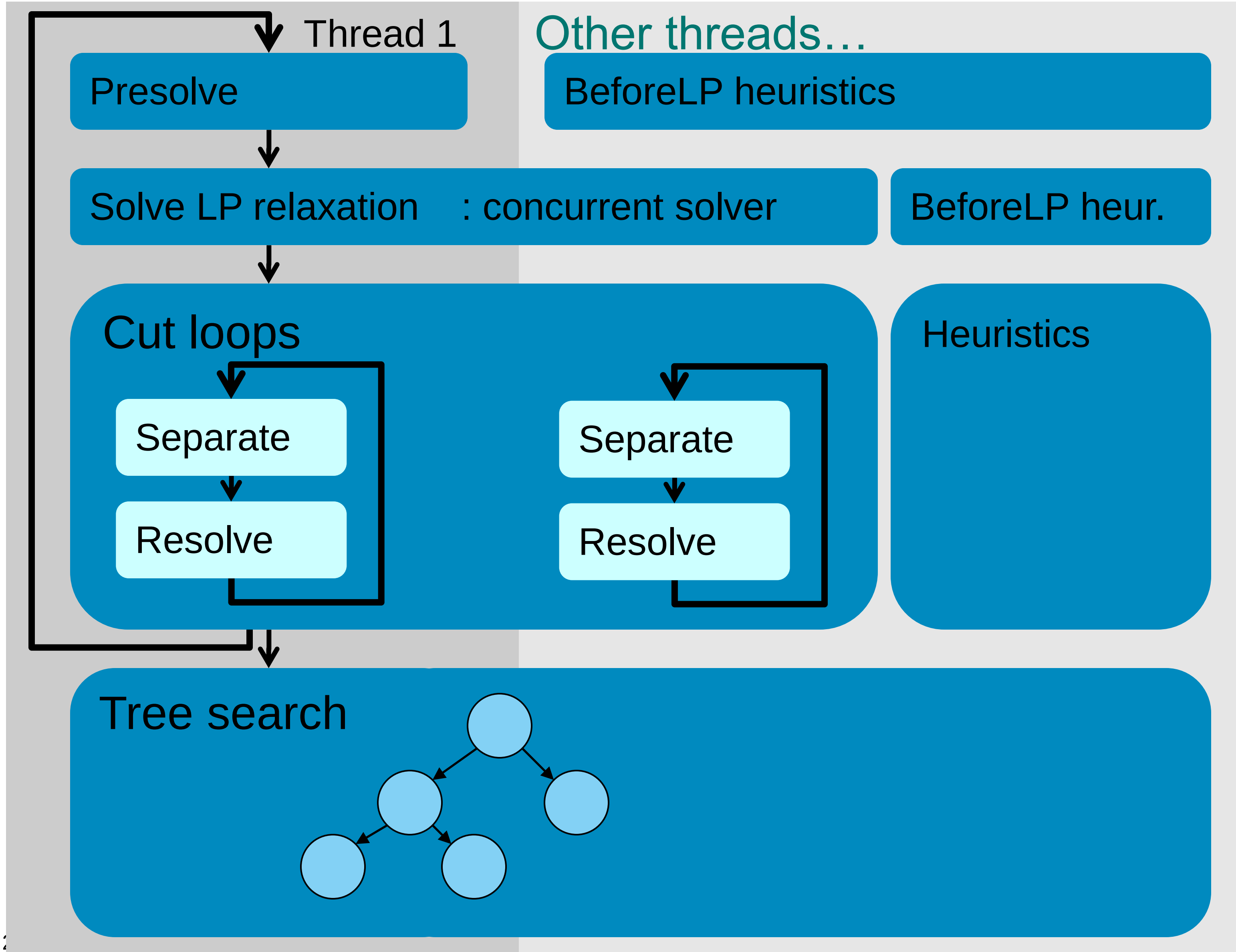
- Starting heuristics
  - Heuristics that do not need any LP solution available
    - before LP heuristics, …
  - Heuristics based on the current LP solution
    - Diving heuristics
      - Simulate depth-first-search with special branching strategy
      - Propagate and resolve LP
    - …
- Improving heuristics
  - Heuristics that do not need any LP solution available
    - Neighborood depends on incumbent solution only
  - Heuristic based on the current LP solution
    - E.g., RINS (Danna et al., 2005)
- Survey paper:
  - M. Fischetti, A. Lodi, Heuristics in mixed integer programming, in J.J. Cochran (ed.) Wiley Encyclopedia of Operations Research and Management Science, Vol. 8, pp. 738-747, Jonh Wiley & sons, 2011

In Summary

CPLEX MIP Solver

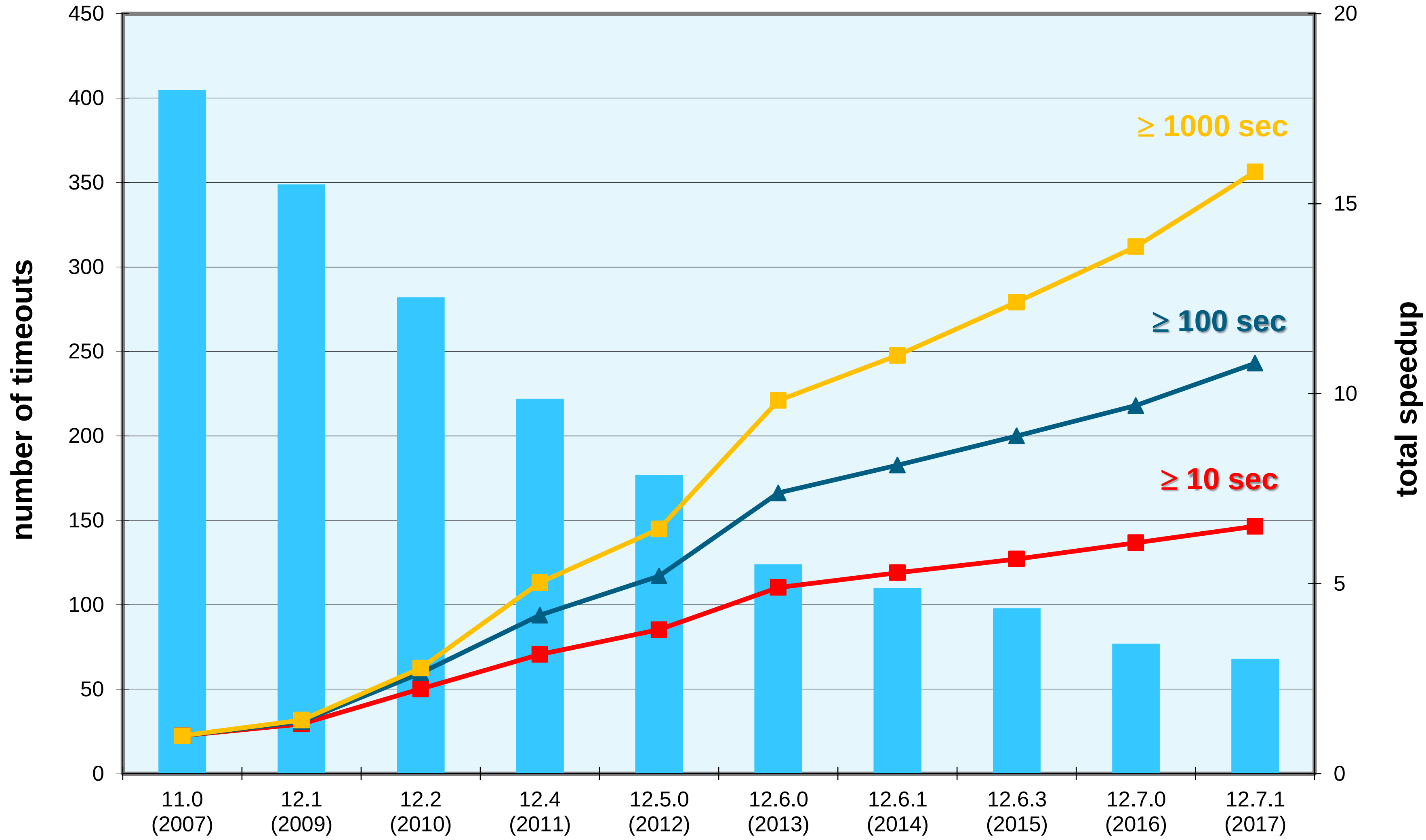


CPLEX parallel MIP Solver



MIP Performance Analysis

CPLEX MILP performance evolution



Date:	18 June 2017
Testset:	MILP: 4009 models
Machine:	Intel X5650 @ 2.67GHz, 24 GB RAM, 12 threads, deterministic
Timelimit:	10,000 sec

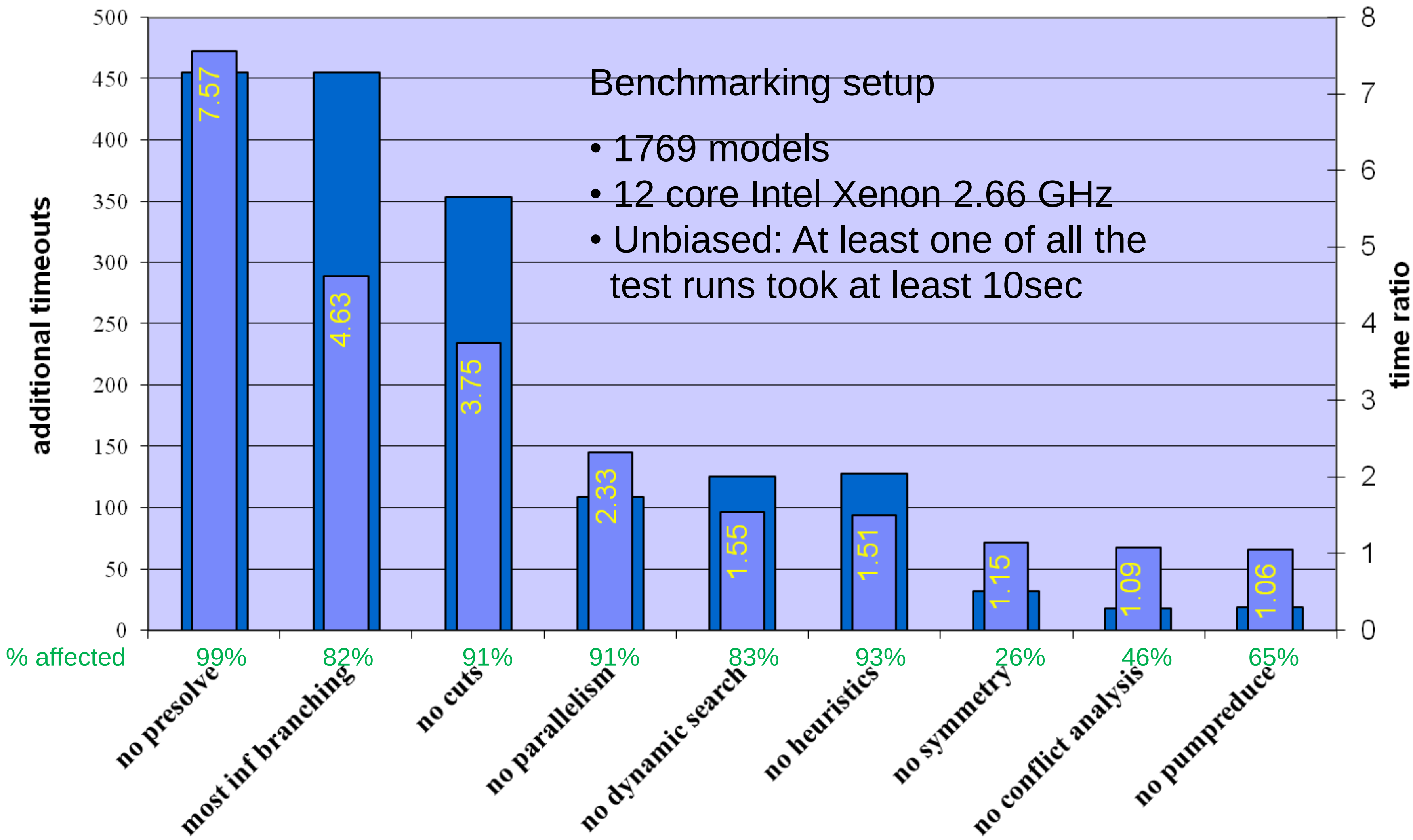
Main building blocks: Measuring performance impact

- How important is each component?
  Compare runs with feature turned on and off
  - Solution time degradation (geometric mean)
  - # of solved models
    - Essential or just speedup?
  - Number of affected models
    - General of problem specific?

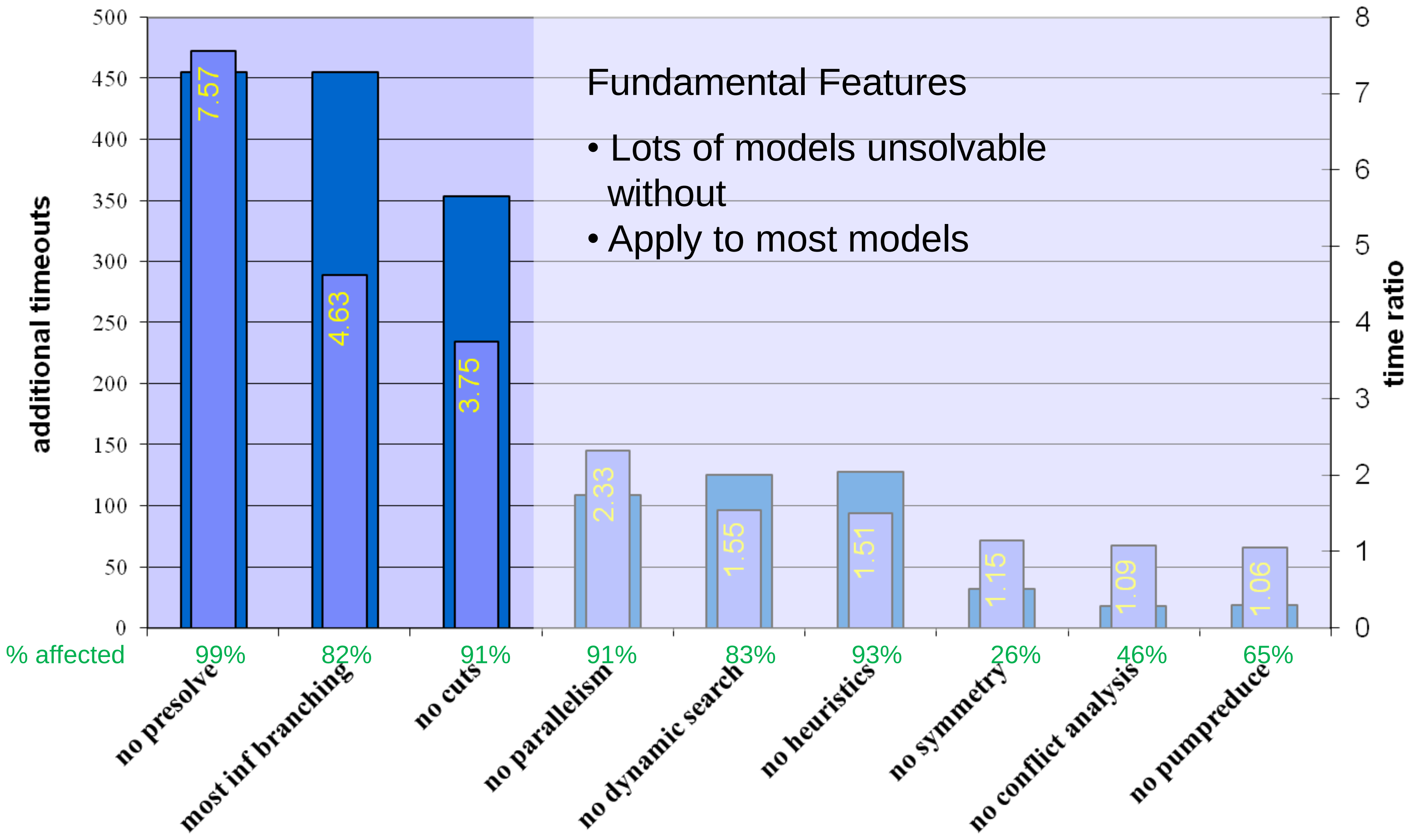
- Experiments conducted with CPLEX 12.5.0 (2012)
  - Several features not available yet, e.g.,
    - L&P cuts (added in CPLEX 12.5.1)
    - Parallel cut loop (added in CPLEX 12.5.1)

- More detailed analysis in:
  T. Achterberg and R. Wunderling, “Mixed Integer Programming: Analyzing 12 Years of Progress”, in: Jünger and Reinelt (eds.) Facets of Combinatorial Optimization, Festschrift for Martin Grötschel, pp.449-481, Springer, Berlin-Heidelberg (2013)

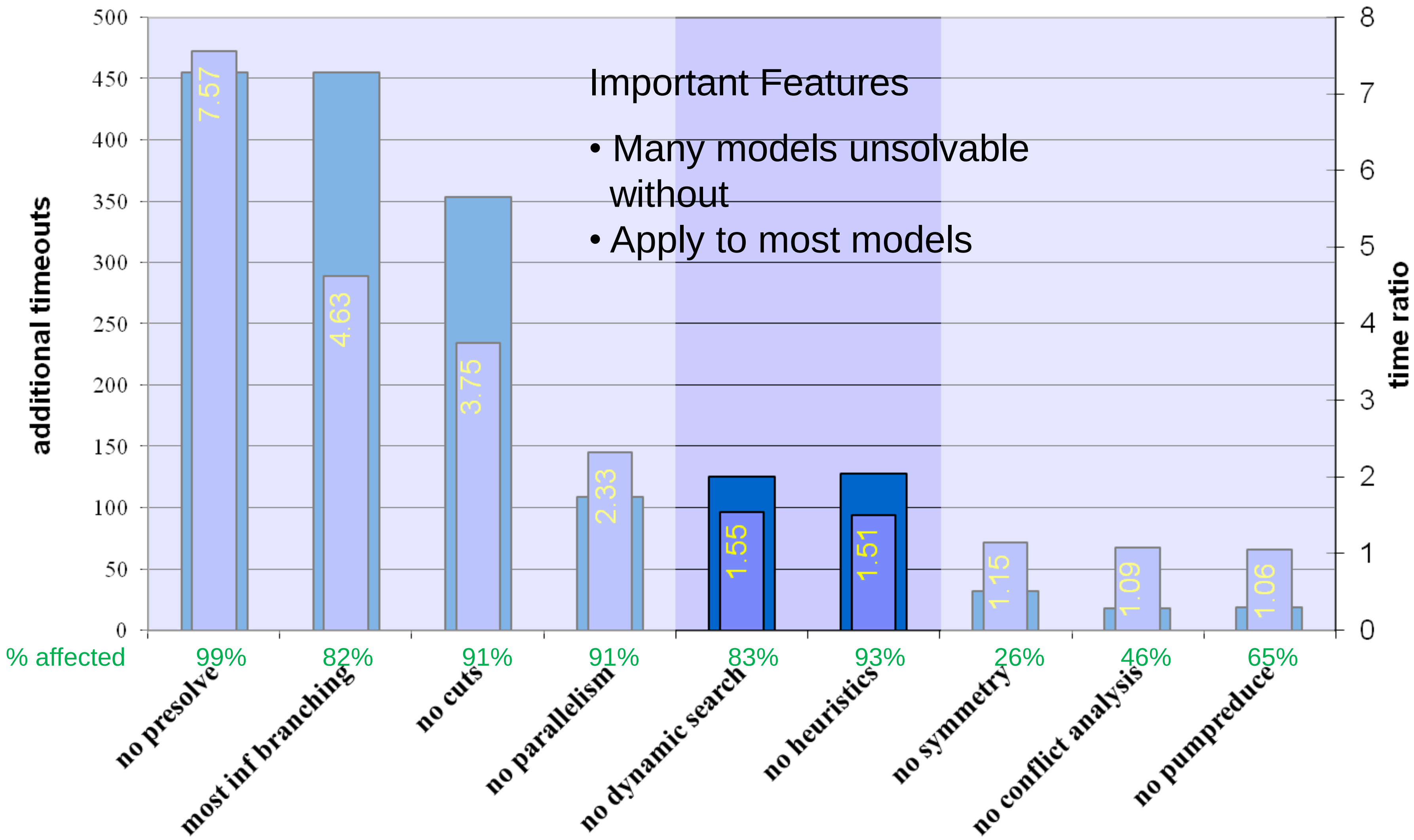
Component Impact CPLEX 12.5.0 - Summary



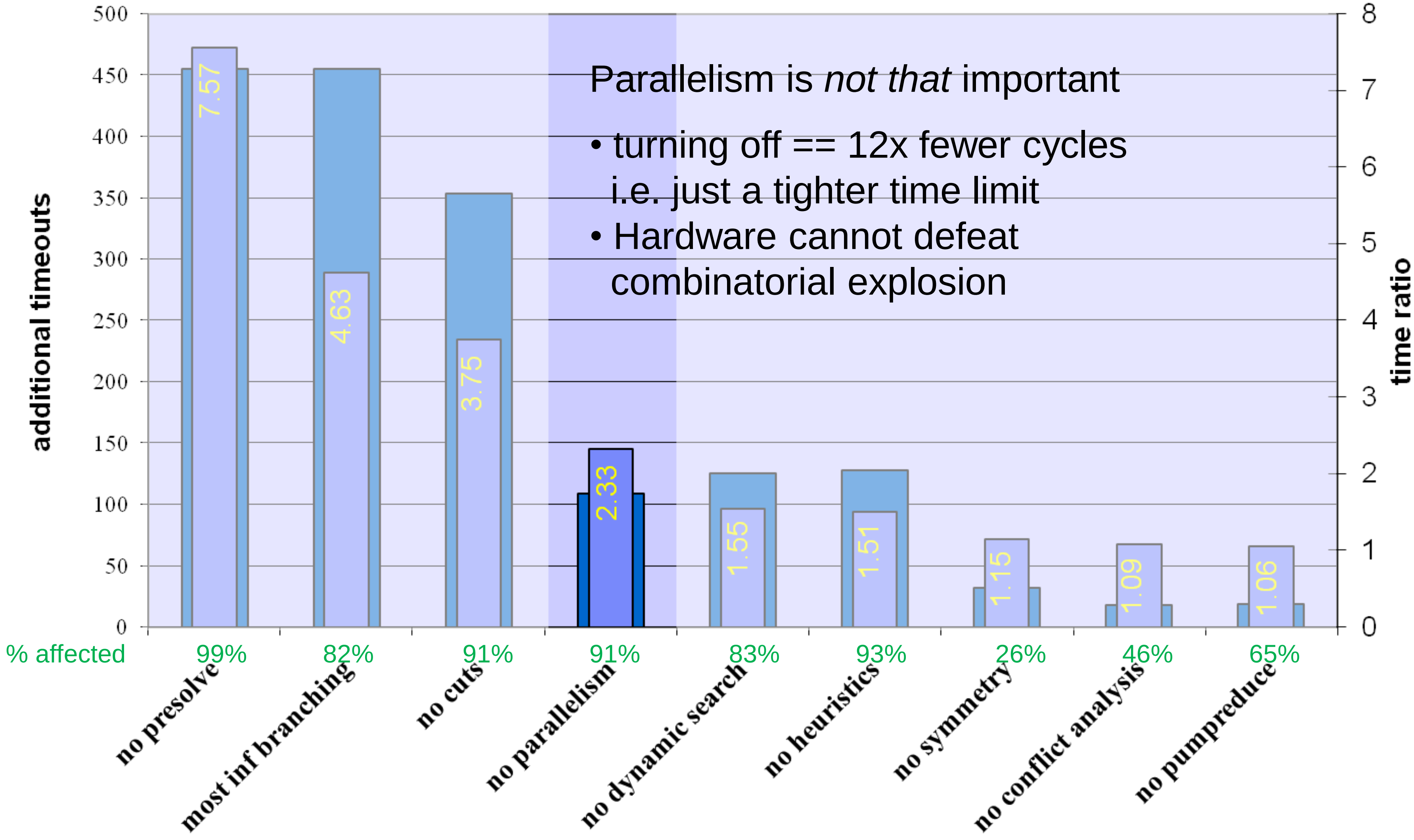
Component Impact CPLEX 12.5.0 - Summary



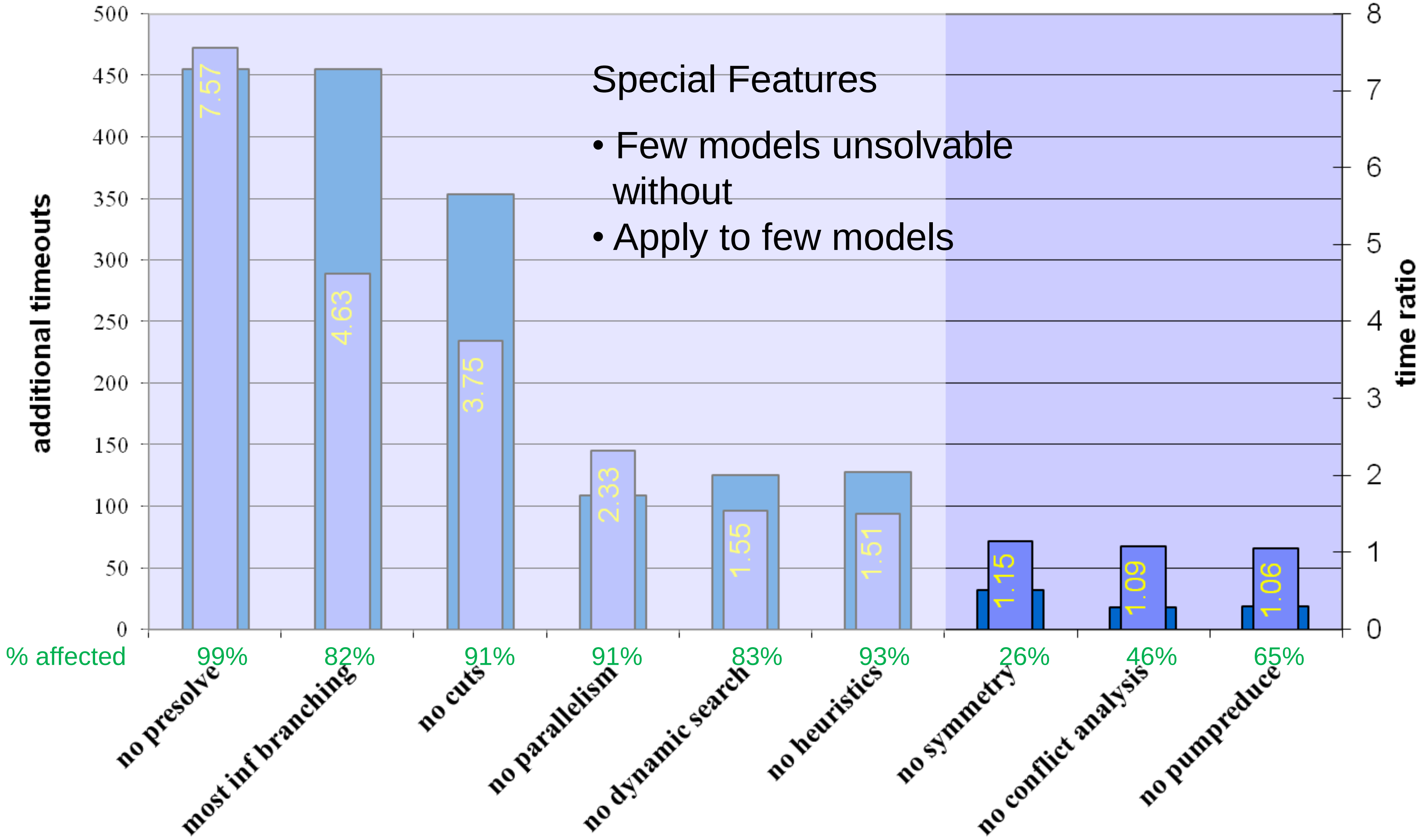
Component Impact CPLEX 12.5.0 - Summary



Component Impact CPLEX 12.5.0 - Summary



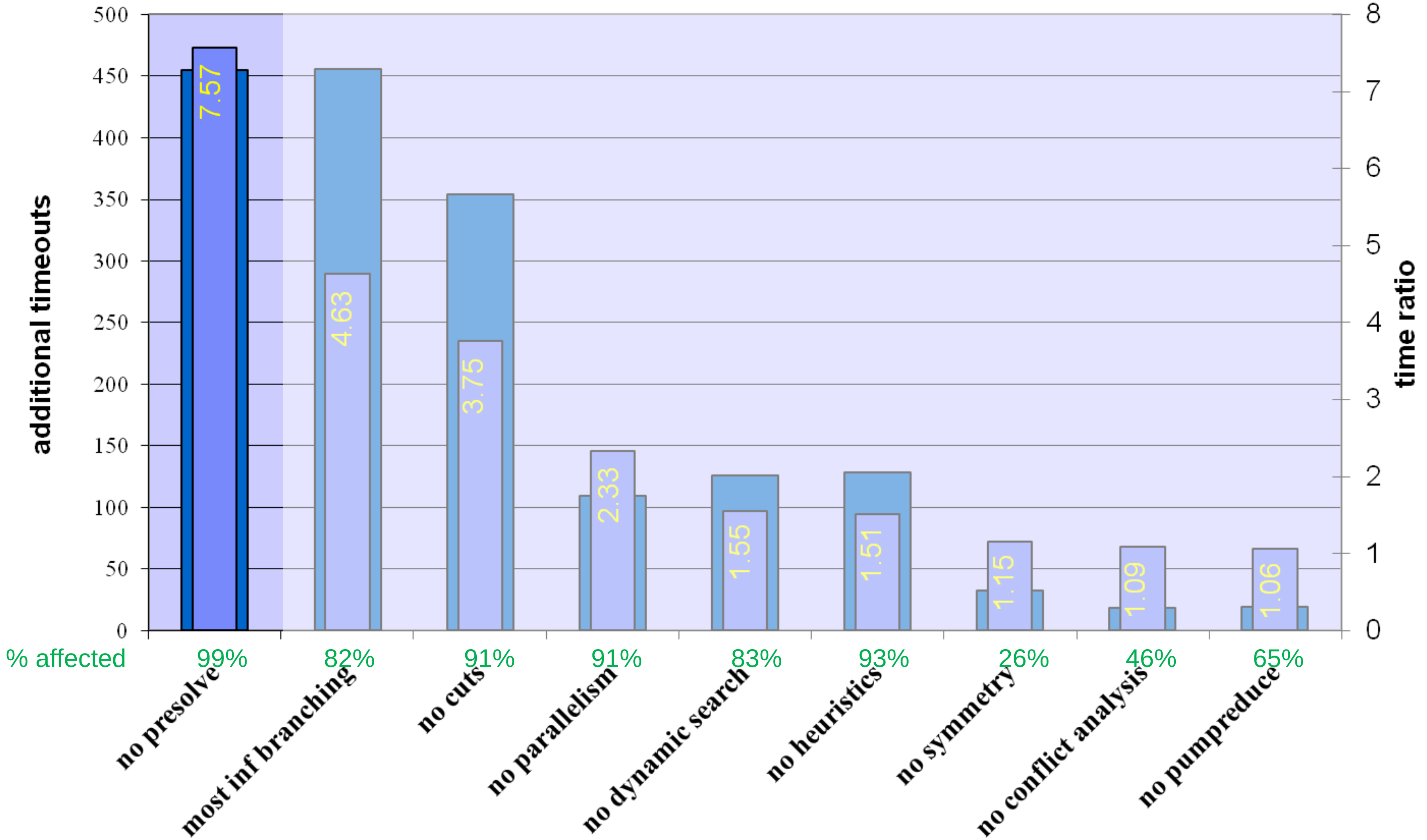
Component Impact CPLEX 12.5.0 - Summary



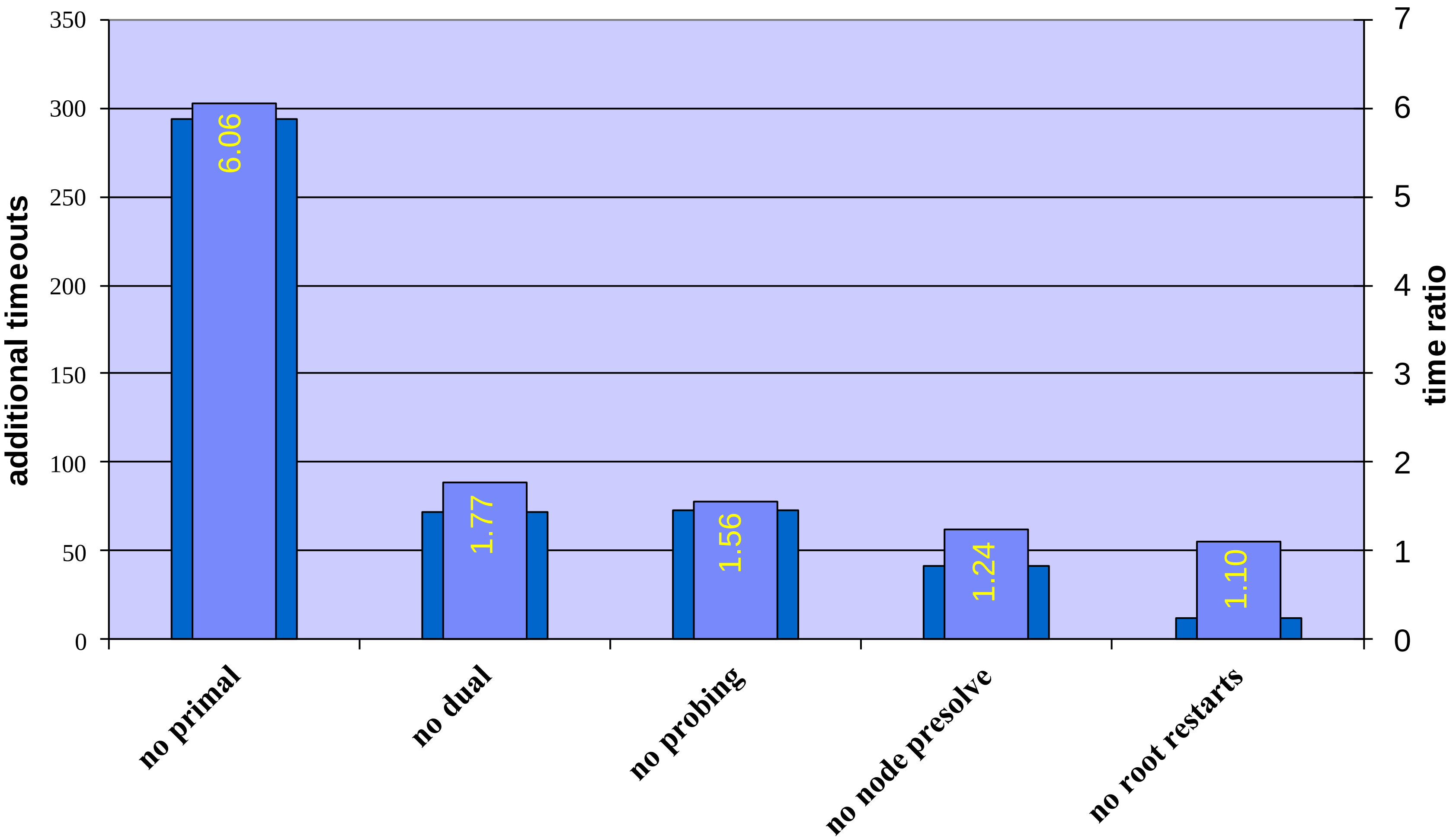
Special Features

- Few models unsolvable without
- Apply to few models

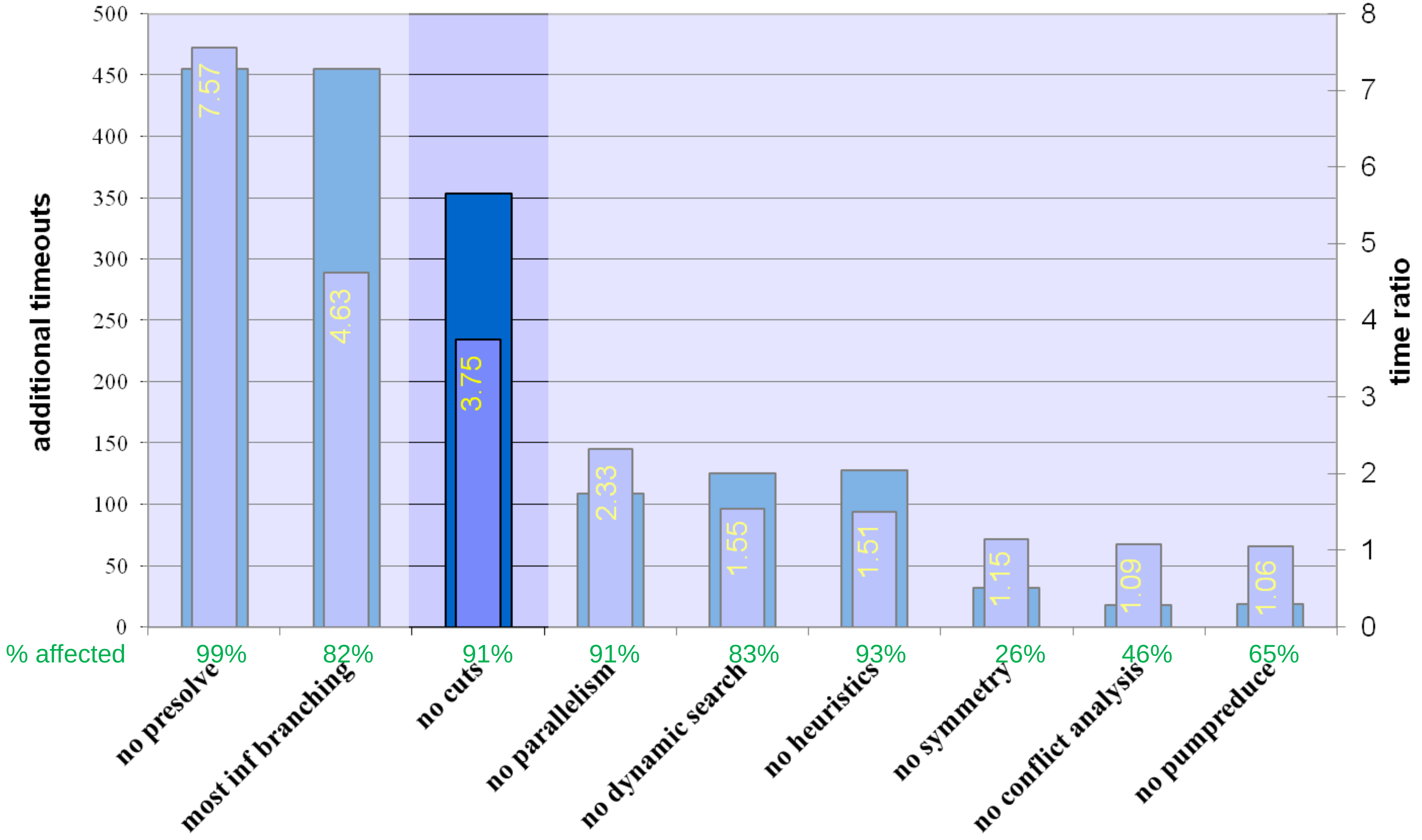
Component Impact CPLEX 12.5.0 - Summary



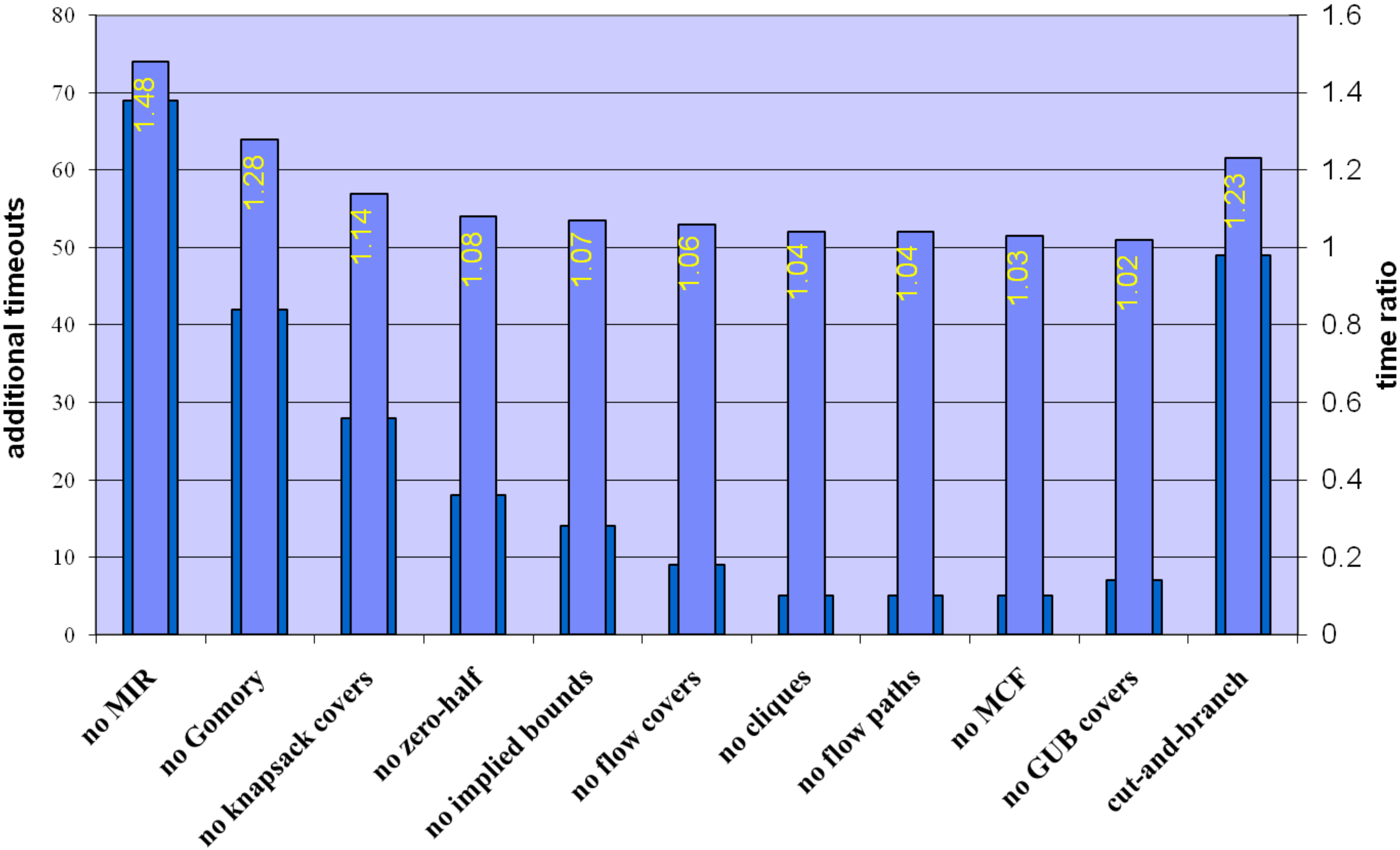
Component Impact CPLEX 12.5.0 – Presolve



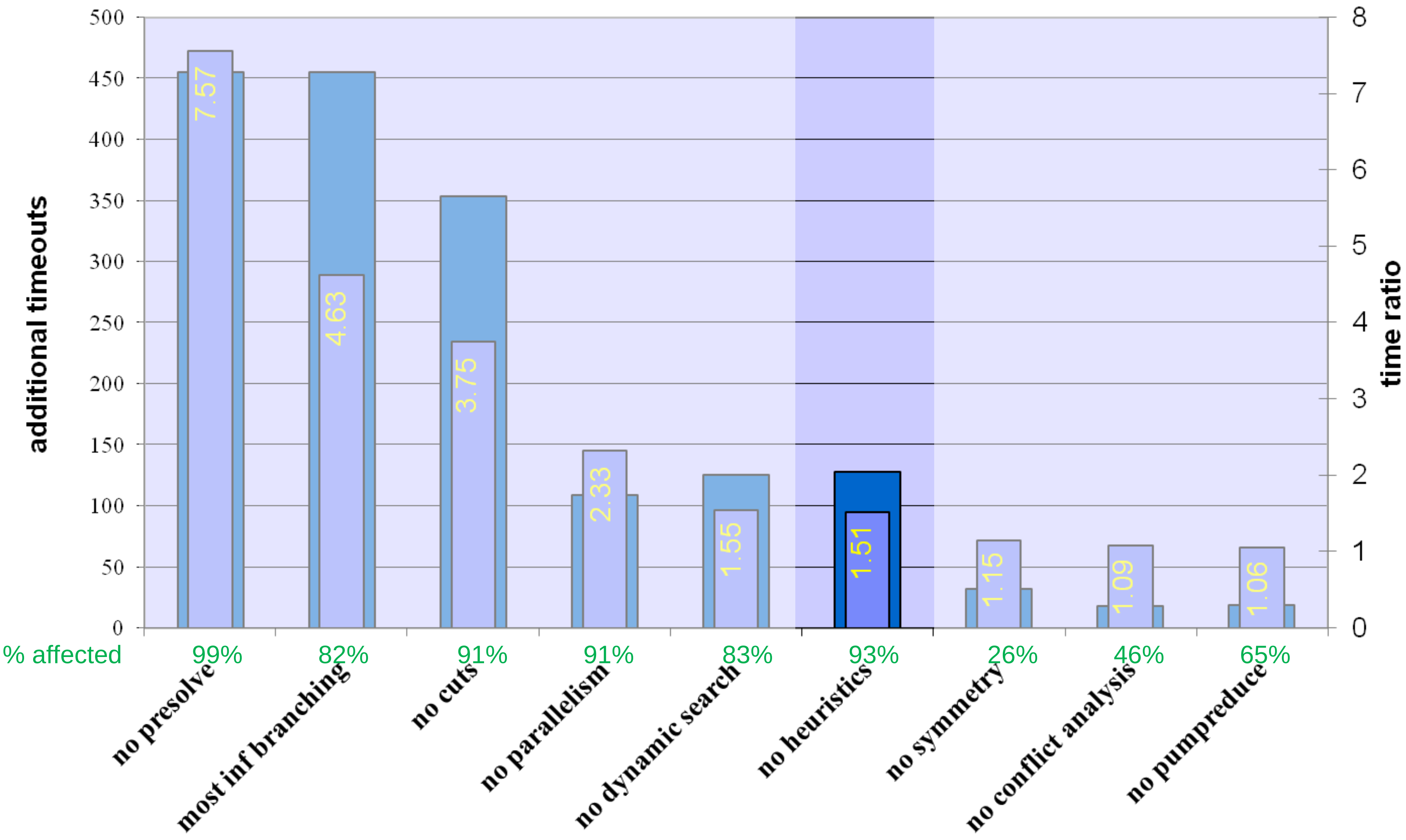
Component Impact CPLEX 12.5.0 - Summary



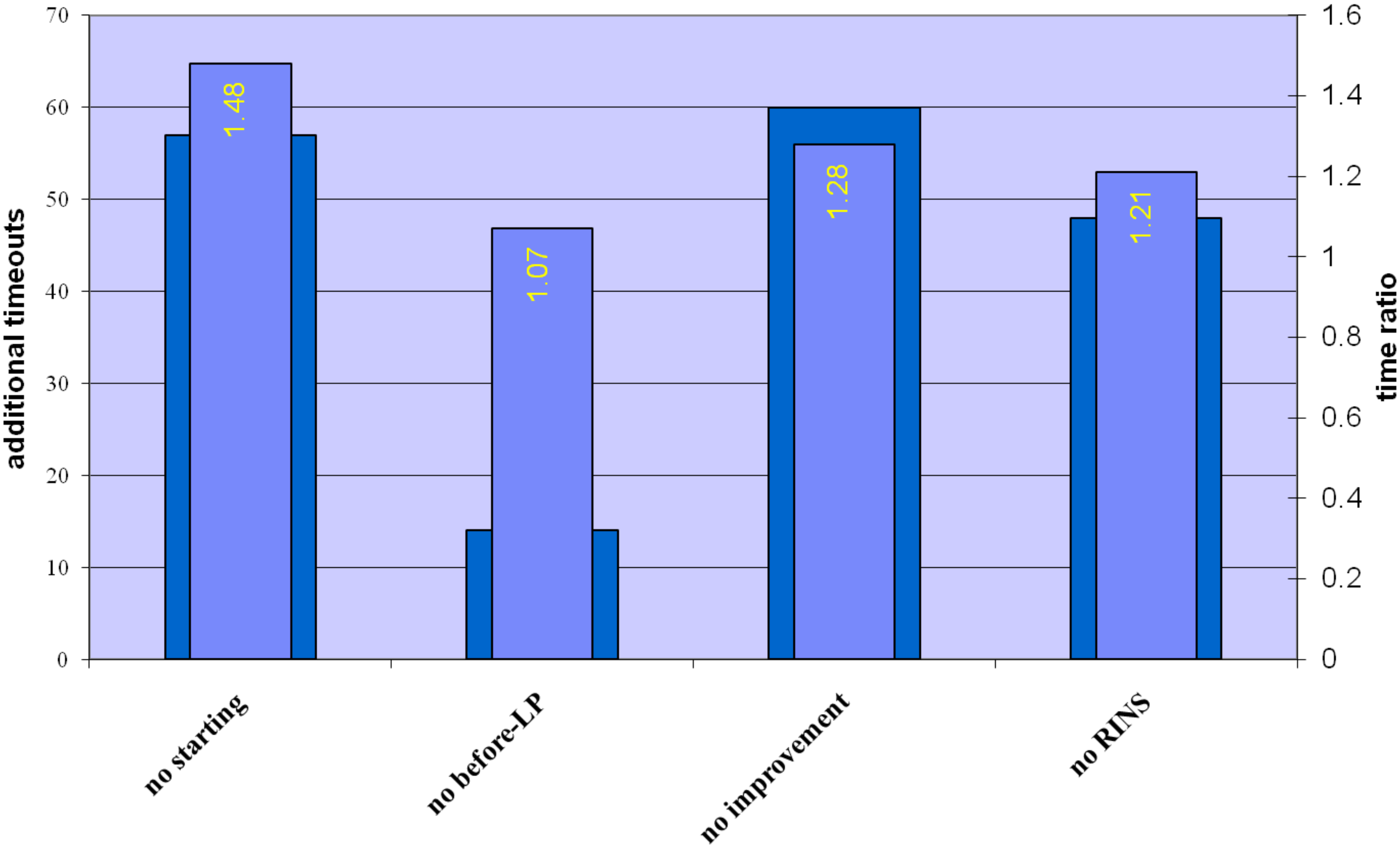
Component Impact CPLEX 12.5.0 – Cutting planes



Component Impact CPLEX 12.5.0 - Summary



Component Impact CPLEX 12.5.0 – Primal heuristics



Legal Disclaimer

-
- The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
- Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.

IBM®