Imperial College London

computational optimisation group

Integrating Mixed-Integer Optimisation & Satisfiability Modulo Theories

Application to Scheduling

Miten Mistry and Ruth Misener

Monday 15th May, 2017

EPSRC
Engineering and Physical Sciences Research Council

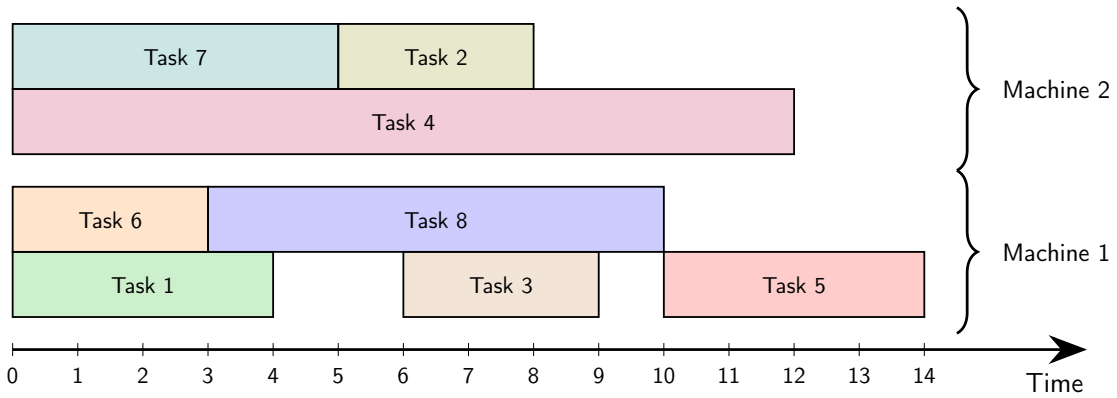
HiPEDS

Sources

- Mistry, Huth, Misener. Logical Modelling Using Satisfiability Modulo Theories to Manage Symmetric and Degenerate Mixed Integer Linear Programming Problems, 2017. `http://www.optimization-online.org/DB_FILE/2017/04/5978.pdf`
- Mistry, Misener. Integrating Mixed-Integer Optimisation and Satisfiability Modulo Theories: Application to Scheduling, 2017. `http://folk.ntnu.no/skoge/prost/proceedings/focapo-cpc-2017/FOCAPO-CPC%202017%20Invited%20Papers/36_FOCAPO_Invited.pdf`

Optimisation & Logic

- Possible modelling frameworks:
  - Generalised disjunctive programming, e.g. Grossmann & co-workers
  - Mixed logical-linear programming, e.g. Hooker & co-workers
- Exisiting solution frameworks (Maravelias and Sung, 2009):
  - Solve as a monolithic optimisation problem
  - Solve as a monolithic satisfiability problem (Bjørner and De Moura, 2011)
  - Decomposition methods
    - e.g., Logic-based Benders decomposition (Jain and Grossmann, 2001; Hooker and Ottoson, 2003)



Outline

Mixed Integer Optimisation

Mixed Integer *Linear* Programming (MILP)

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{R}^n \\ & x_j \in \mathbb{Z},\ j \in I \subseteq \{1, \ldots, n\} \end{aligned}$$

where $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$.

Mixed Integer *Nonlinear* Programming (MINLP)

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g(\boldsymbol{x}) \leq \boldsymbol{0} \\ & \boldsymbol{x} \in \mathbb{R}^n \\ & x_j \in \mathbb{Z},\ j \in I \subseteq \{1, \ldots, n\} \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$.

Solving a MILP

- Branch and cut
  - Relaxation - integer variables become continuous

$$z \in \{0, 1\} \twoheadrightarrow 0 \leq z \leq 1$$

  - Solve relaxed model with linear programming, usually simplex algorithm
  - Branch on non-integral solutions in relaxed problem
  - Add cutting planes - linear equalities.

Typically not numerically safe

Exception: Cook et al. (2013)

Propositional Satisfiability (SAT)

- Alphabet soup:

SAT	Propositional Satisfiability
CP	Constraint Programming
SMT	Satisfiability Modulo Theories
ASP	Answer set programming

Propositional Satisfiability (SAT) – Analogy to machine code

Given a propositional formula φ (built from $\wedge$, $\vee$ and $\neg$) over a set of propositional variables, p_i, $i \in \mathcal{I} = \{1, \ldots, n\}$, can we find an assignment on the p_i's such that φ evaluates to True (satisfied)?

Other Constraint Satisfaction Techniques – Analogy to high level language

CP, SMT, ASP

Modelling with Propositional Satisfiability

- Typically solved with a variant of the DPLL algorithm (tree search):
  - Expects a formula in *conjunctive normal form*,
  - Branches on Boolean variables,
  - Reduces size of constraints by unit propagation,
  - Derive conflict clauses.
- Variants of DPLL derive conflict clauses

Constraint Programming (CP)

- Special constraints exploit structure found in specific applications
- Example: cumulative constraint
  - mainly used in scheduling applications

$$\texttt{cumulative}(s, p, c, C)$$

  - s: start times
  - p: processing times
  - c: resource consumptions
  - C: resource consumption limit

Hybridisation with MILP

Bockmayr and Kasper (1998); Jain and Grossmann (2001); Hooker and Ottoson (2003); Maravelias and Grossmann (2004); Bockmayr and Kasper (2004); Hooker (2007), etc.

Satisfiability Modulo Theories (SMT)

- Reason about satisfiability (SAT) with respect to a theory
  - Encode the SMT formula into a propositional formula and use the theory solver with the SAT solver to assess satisfiability.
- Can result in:

SAT	Satisfiable
UNSAT	Unsatisfiable
UNKNOWN	For our applications usually a timeout

Theories of Interest

- Integer (Real) arithmetic - Interpret the symbols $(0, 1, +, -, =, \leq)$ in the usual way over $\mathbb{Z}$ ($\mathbb{R}$)
- Arrays
- Combined theories - e.g. combining integer arithmetic with arrays to reason about `a[i+j]`

SMT Example (Bjørner and De Moura, 2011)

Figure 1. Encoding job-shop scheduling.

$d_{i,j}$	Machine 1	Machine 2
Job 1	2	1
Job 2	3	1
Job 3	2	3

max $= 8$

Solution

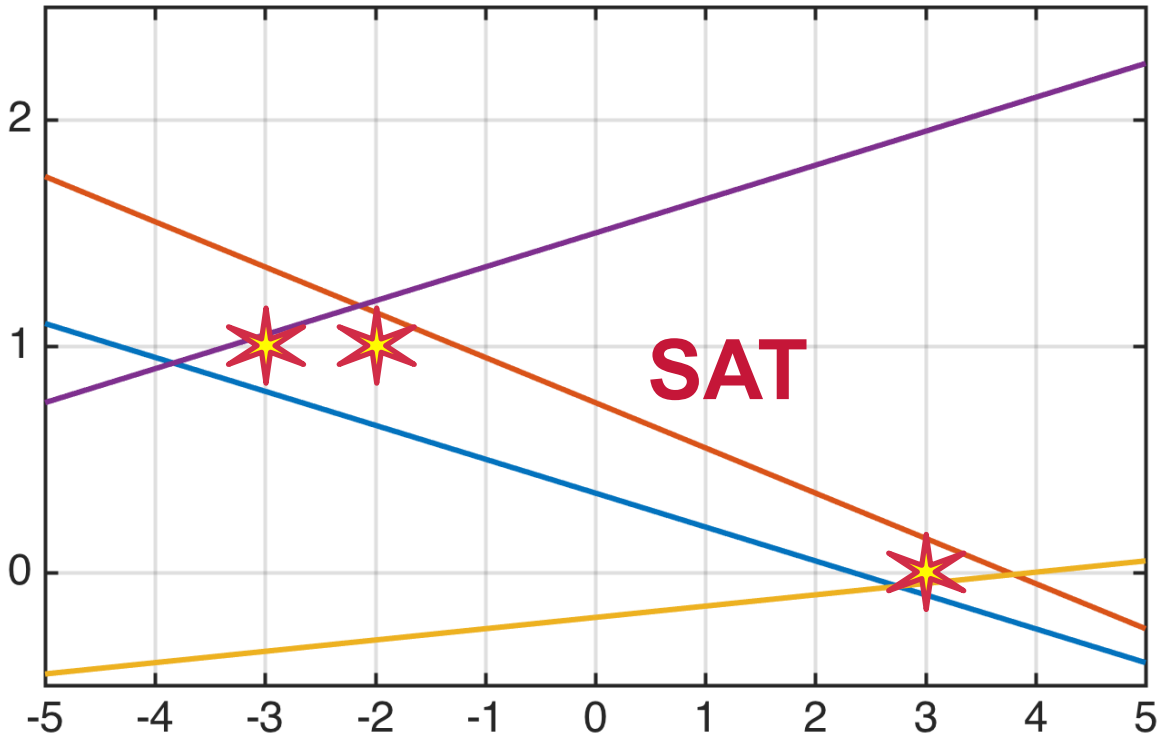
$t_{1,1} = 5, \quad t_{1,2} = 7,$
$t_{2,1} = 2, \quad t_{2,2} = 6,$
$t_{3,1} = 0, \quad t_{3,2} = 3$

Encoding

$$
\begin{aligned}
&(t_{1,1} \geq 0) \wedge (t_{1,2} \geq t_{1,1} + 2) \wedge (t_{1,2} + 1 \leq 8) \wedge \\
&(t_{2,1} \geq 0) \wedge (t_{2,2} \geq t_{2,1} + 3) \wedge (t_{2,2} + 1 \leq 8) \wedge \\
&(t_{3,1} \geq 0) \wedge (t_{3,2} \geq t_{3,1} + 2) \wedge (t_{3,2} + 3 \leq 8) \wedge \\
&((t_{1,1} \geq t_{2,1} + 3) \vee (t_{2,1} \geq t_{1,1} + 2)) \wedge \\
&((t_{1,1} \geq t_{3,1} + 2) \vee (t_{3,1} \geq t_{1,1} + 2)) \wedge \\
&((t_{2,1} \geq t_{3,1} + 2) \vee (t_{3,1} \geq t_{2,1} + 3)) \wedge \\
&((t_{1,2} \geq t_{2,2} + 1) \vee (t_{2,2} \geq t_{1,2} + 1)) \wedge \\
&((t_{1,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{1,2} + 1)) \wedge \\
&((t_{2,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{2,2} + 1))
\end{aligned}
$$

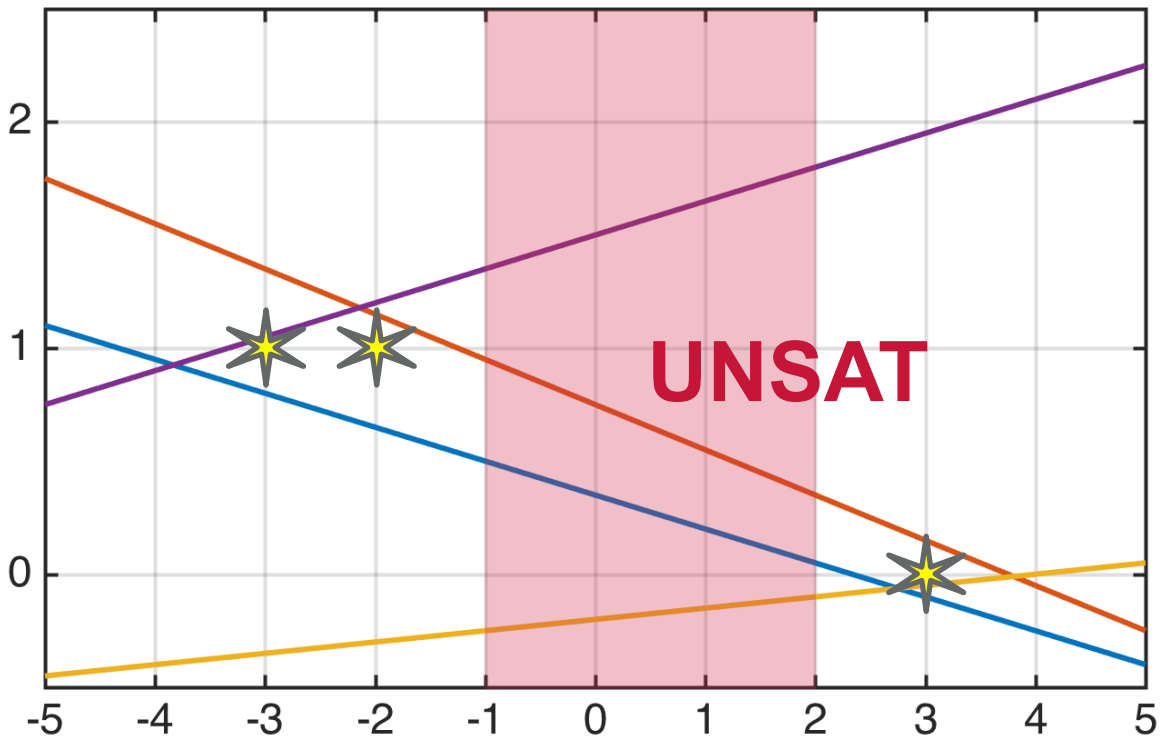
Example: Satisfiability Modulo Theories (SMT)

$$
\begin{aligned}
0.15 \cdot x + y &\geq 0.35 \\
-0.20 \cdot x - y &\geq -0.75 \\
-0.05 \cdot x + y &\geq 0.20 \\
0.15 \cdot x - y &\geq -1.50 \\
x, y &\in \mathbb{Z}
\end{aligned}
$$



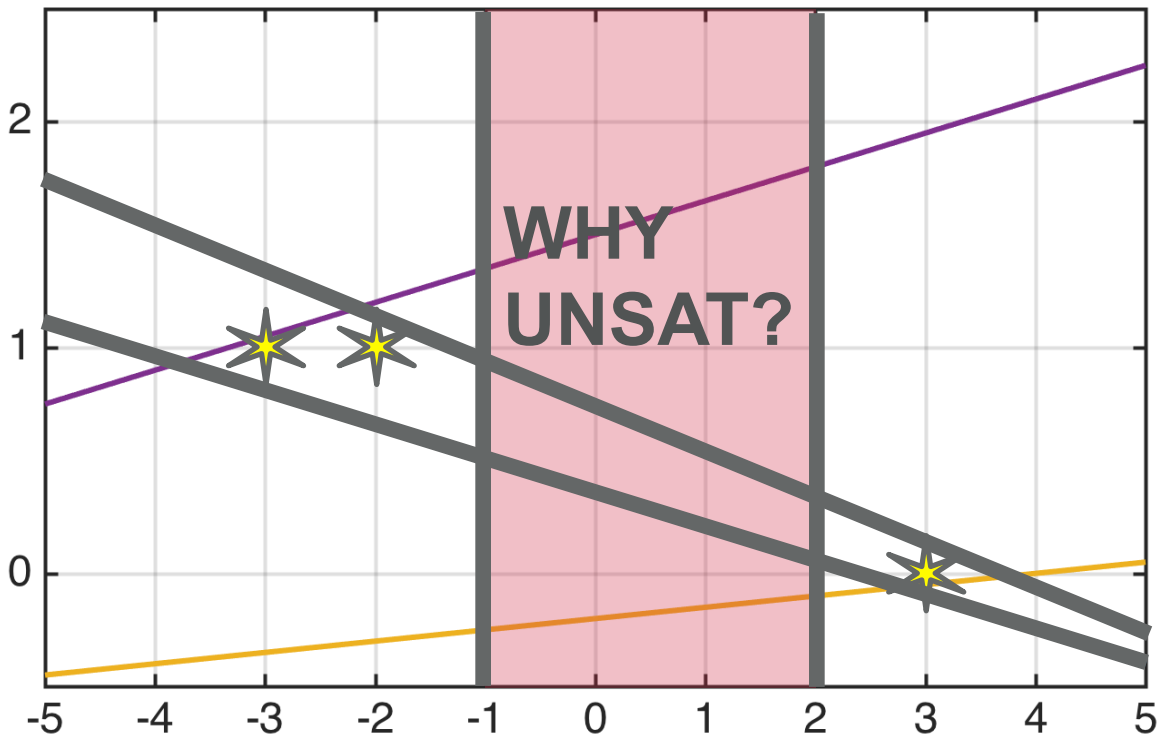
Example: Satisfiability Modulo Theories (SMT)

$$
\begin{aligned}
0.15 \cdot x + y &\geq 0.35 \\
-0.20 \cdot x - y &\geq -0.75 \\
-0.05 \cdot x + y &\geq 0.20 \\
0.15 \cdot x - y &\geq -1.50 \\
x, y &\in \mathbb{Z} \\
\mathbf{x} &\in \mathbf{[-1, 2]}
\end{aligned}
$$



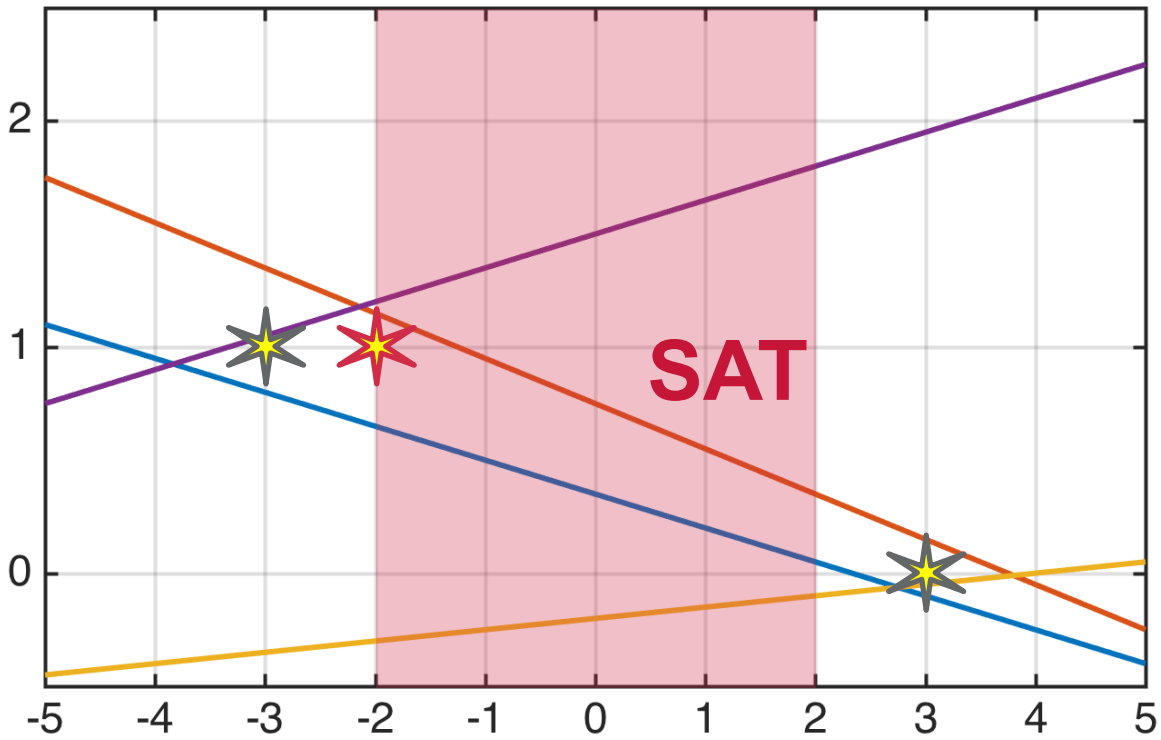
Example: Satisfiability Modulo Theories (SMT)

$$
\begin{aligned}
0.15 \cdot x + y &\geq 0.35 \\
-0.20 \cdot x - y &\geq -0.75 \\
-0.05 \cdot x + y &\geq 0.20 \\
0.15 \cdot x - y &\geq -1.50 \\
x, y &\in \mathbb{Z} \\
x &\in [-1, 2]
\end{aligned}
$$



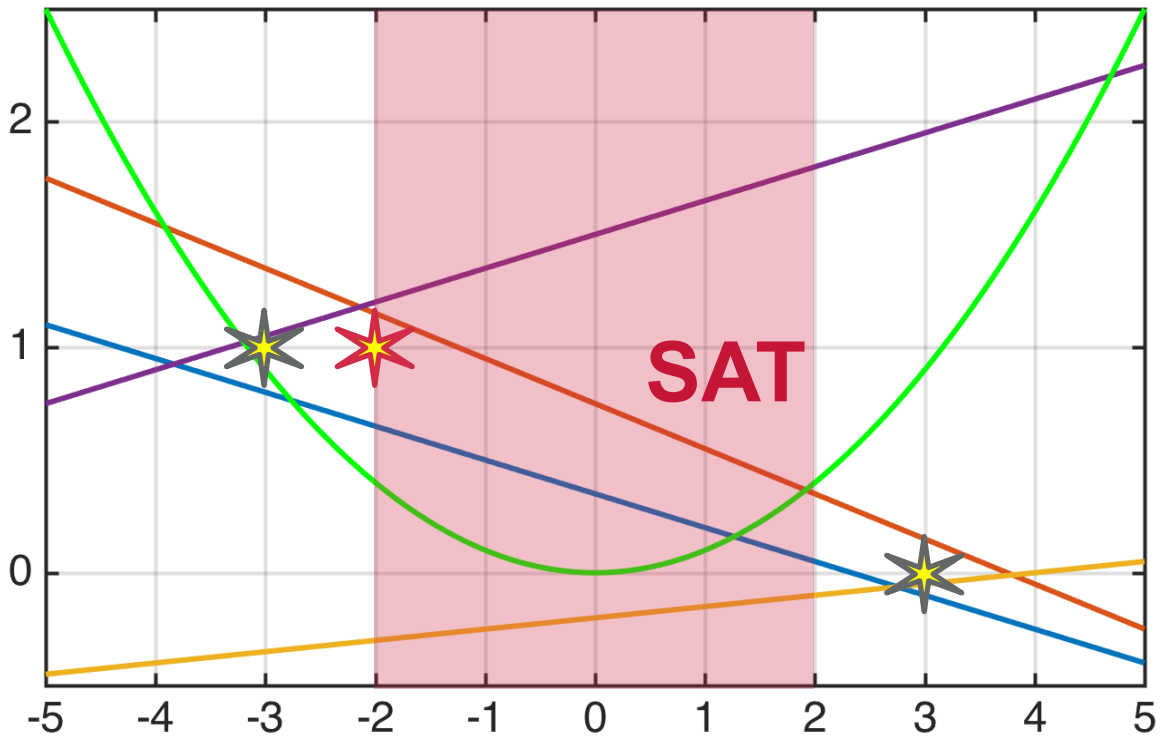
Example: Satisfiability Modulo Theories (SMT)

$$
\begin{aligned}
0.15 \cdot x + y &\geq 0.35 \\
-0.20 \cdot x - y &\geq -0.75 \\
-0.05 \cdot x + y &\geq 0.20 \\
0.15 \cdot x - y &\geq -1.50 \\
x, y &\in \mathbb{Z} \\
\cancel{x \in [-1, 2]} & \\
\mathbf{x \in [-2, 2]} &
\end{aligned}
$$



Example: Satisfiability Modulo Theories (SMT)

$$0.15 \cdot x + y \geq 0.35$$
$$-0.20 \cdot x - y \geq -0.75$$
$$-0.05 \cdot x + y \geq 0.20$$
$$0.15 \cdot x - y \geq -1.50$$
$$x, y \in \mathbb{Z}$$
~~$x \in [-1, 2]$~~
$$x \in [-2, 2]$$
$$\mathbf{-0.10 \cdot x^2 + y \geq 0.00}$$

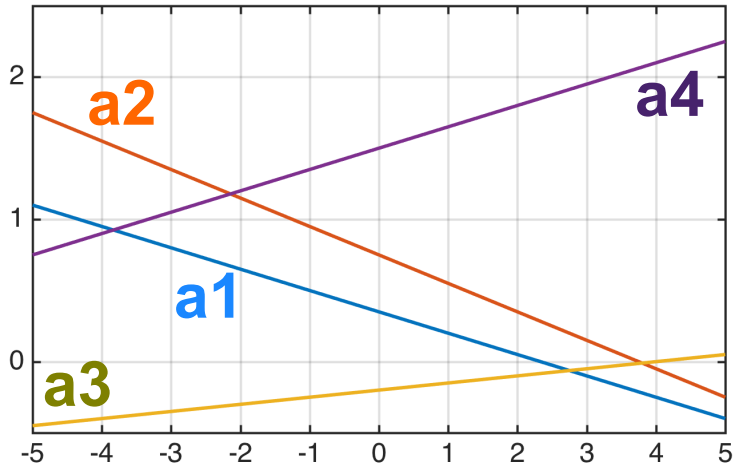


Example: Satisfiability Modulo Theories (SMT)

```
(set-option :produce-unsat-cores true)
(declare-const x Int)
(declare-const y Int)
(assert (! (>= (+ (*  0.15 x) (*  1 y))  0.35) :named a1))
(assert (! (>= (+ (* -0.20 x) (* -1 y)) -0.75) :named a2))
(assert (! (>= (+ (* -0.05 x) (*  1 y))  0.20) :named a3))
(assert (! (>= (+ (*  0.15 x) (* -1 y)) -1.50) :named a4))
(push)
(check-sat)
```

rise4fun.com/z3

→ **SAT**



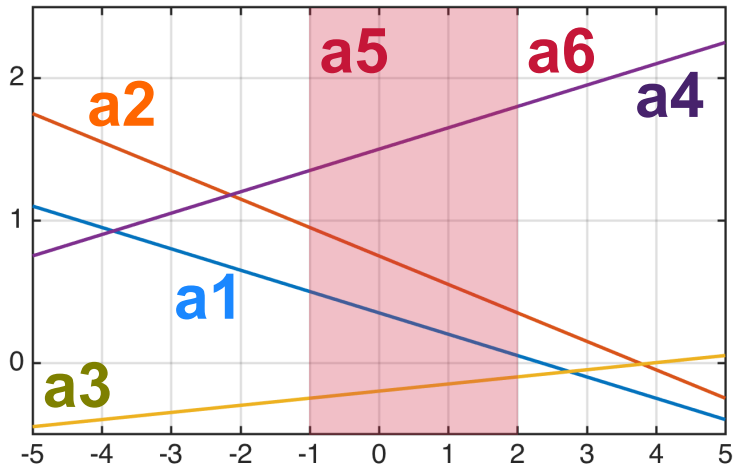
Example: Satisfiability Modulo Theories (SMT)

```
(set-option :produce-unsat-cores true)
(declare-const x Int)
(declare-const y Int)
(assert (! (>= (+ (*  0.15 x) (*  1 y))  0.35) :named a1))
(assert (! (>= (+ (* -0.20 x) (* -1 y)) -0.75) :named a2))
(assert (! (>= (+ (* -0.05 x) (*  1 y))  0.20) :named a3))
(assert (! (>= (+ (*  0.15 x) (* -1 y)) -1.50) :named a4))
(push)
(check-sat)
(assert (! (>= x -1) :named a5))
(assert (! (<= x  2) :named a6))
(check-sat)
```

rise4fun.com/z3

→ **SAT**

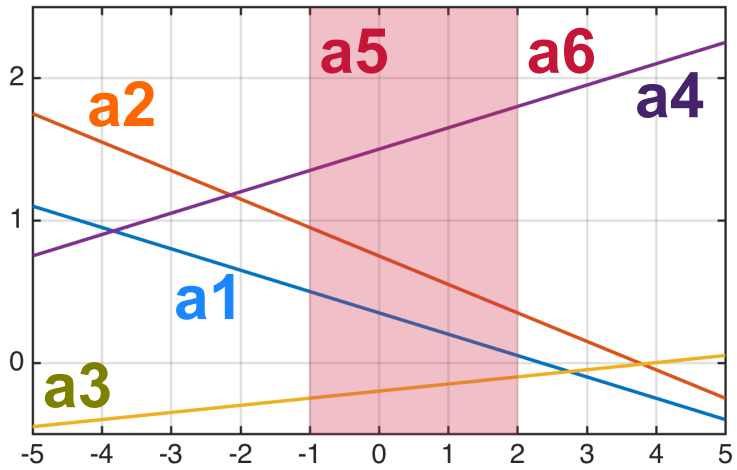
→ **UNSAT**



Example: Satisfiability Modulo Theories (SMT)

```
(set-option :produce-unsat-cores true)
(declare-const x Int)
(declare-const y Int)
(assert (! (>= (+ (*  0.15 x) (*  1 y))  0.35) :named a1))
(assert (! (>= (+ (* -0.20 x) (* -1 y)) -0.75) :named a2))
(assert (! (>= (+ (* -0.05 x) (*  1 y))  0.20) :named a3))
(assert (! (>= (+ (*  0.15 x) (* -1 y)) -1.50) :named a4))
(push)
(check-sat)          → SAT
(assert (! (>= x -1) :named a5))
(assert (! (<= x  2) :named a6))
(check-sat)          → UNSAT
(get-unsat-core)     → (a1 a2 a6 a5)
```

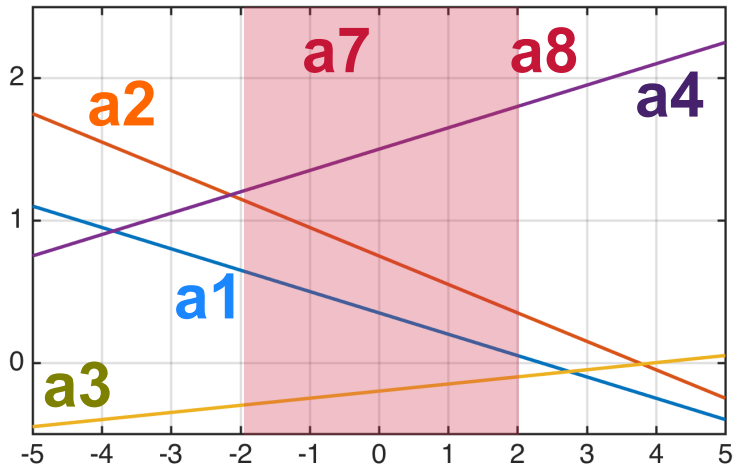
rise4fun.com/z3



Example: Satisfiability Modulo Theories (SMT)

```
(set-option :produce-unsat-cores true)
(declare-const x Int)
(declare-const y Int)
(assert (! (>= (+ (*  0.15 x) (*  1 y))  0.35) :named a1))
(assert (! (>= (+ (* -0.20 x) (* -1 y)) -0.75) :named a2))
(assert (! (>= (+ (* -0.05 x) (*  1 y))  0.20) :named a3))
(assert (! (>= (+ (*  0.15 x) (* -1 y)) -1.50) :named a4))
(push)
(check-sat)          → SAT
(assert (! (>= x -1) :named a5))
(assert (! (<= x  2) :named a6))
(check-sat)          → UNSAT
(get-unsat-core)     → (a1 a2 a6 a5)
(pop)
(assert (! (>= x -2) :named a7))
(assert (! (<= x  2) :named a8))
(check-sat)          → SAT
```

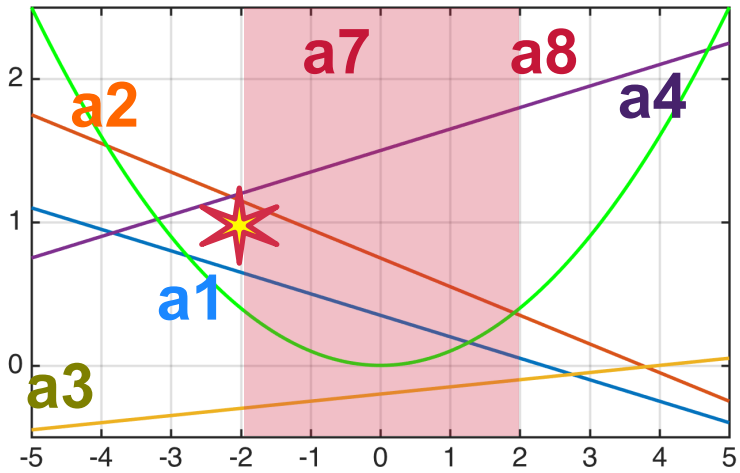
rise4fun.com/z3



Example: Satisfiability Modulo Theories (SMT)

```
(set-option :produce-unsat-cores true)
(declare-const x Int)
(declare-const y Int)
(assert (! (>= (+ (*  0.15 x) (*  1 y))  0.35) :named a1))
(assert (! (>= (+ (* -0.20 x) (* -1 y)) -0.75) :named a2))
(assert (! (>= (+ (* -0.05 x) (*  1 y))  0.20) :named a3))
(assert (! (>= (+ (*  0.15 x) (* -1 y)) -1.50) :named a4))
(push)
(check-sat)          → SAT
(assert (! (>= x -1) :named a5))
(assert (! (<= x  2) :named a6))
(check-sat)          → UNSAT
(get-unsat-core)     → (a1 a2 a6 a5)
(pop)
(assert (! (>= x -2) :named a7))
(assert (! (<= x  2) :named a8))
(check-sat)          → SAT
(assert (! (<= (* 0.1 (* x x)) y) :named a9))
(check-sat)          → SAT
```

rise4fun.com/z3

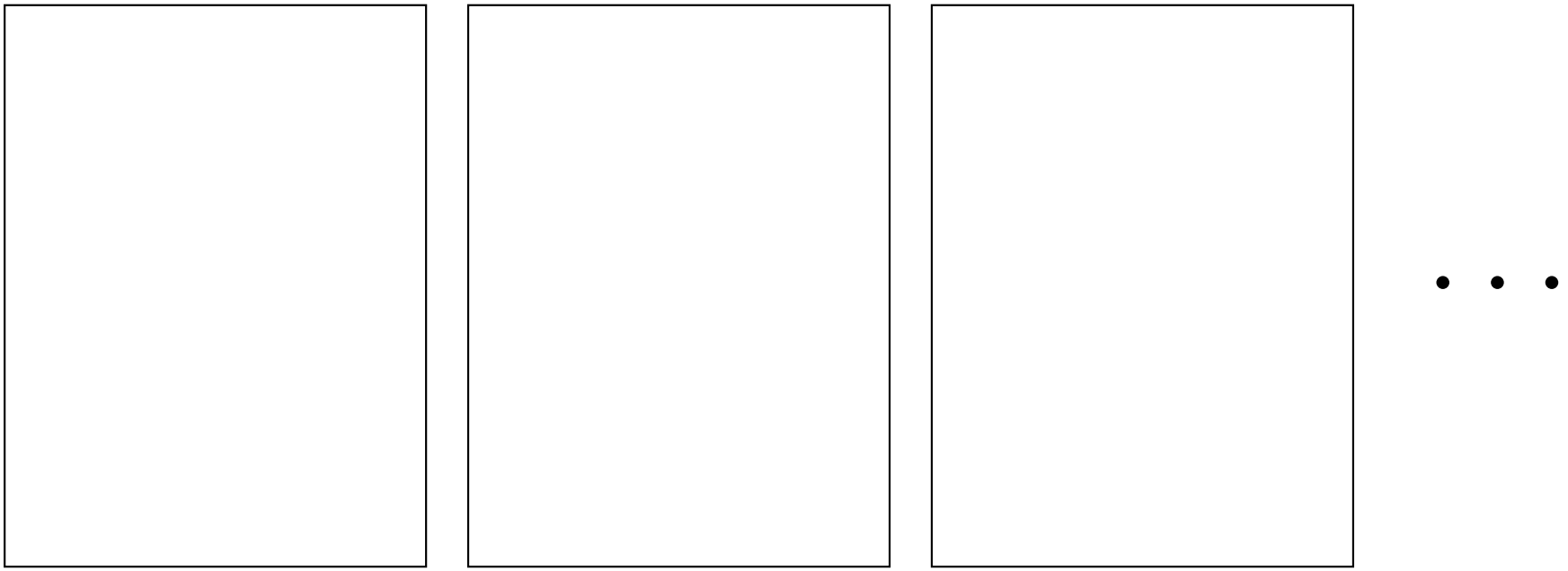


	SMT	MINLP	
Historical / Traditional Community	Computer science	Engineering	Division in mathematical developments stemming from divergent applications
Deductive Reasoning	Strong	Limited	GDP less flexible than SMT
Nonlinear Functions	Limited	Strong	Transcendental functions in MINLP
Optimising Objective	Weak	Strong	Tightly integrated in MINLP
Propositional Satisfiability	Strong	Weak	Logical propositions not typical in MINLP
Warm Starting	Strong	Weak	Would strongly benefit MINLP
Scalability	Limited	Limited	SMT : limited for nonlinear functions MINLP: limited for large problems†
	Limited	Limited	SMT : limited for nonlinear functions MINLP: limited for large problems†
Typical Applications	SMT : Software verification; Scheduling MINLP: Energy systems design; Biomedical engineering		

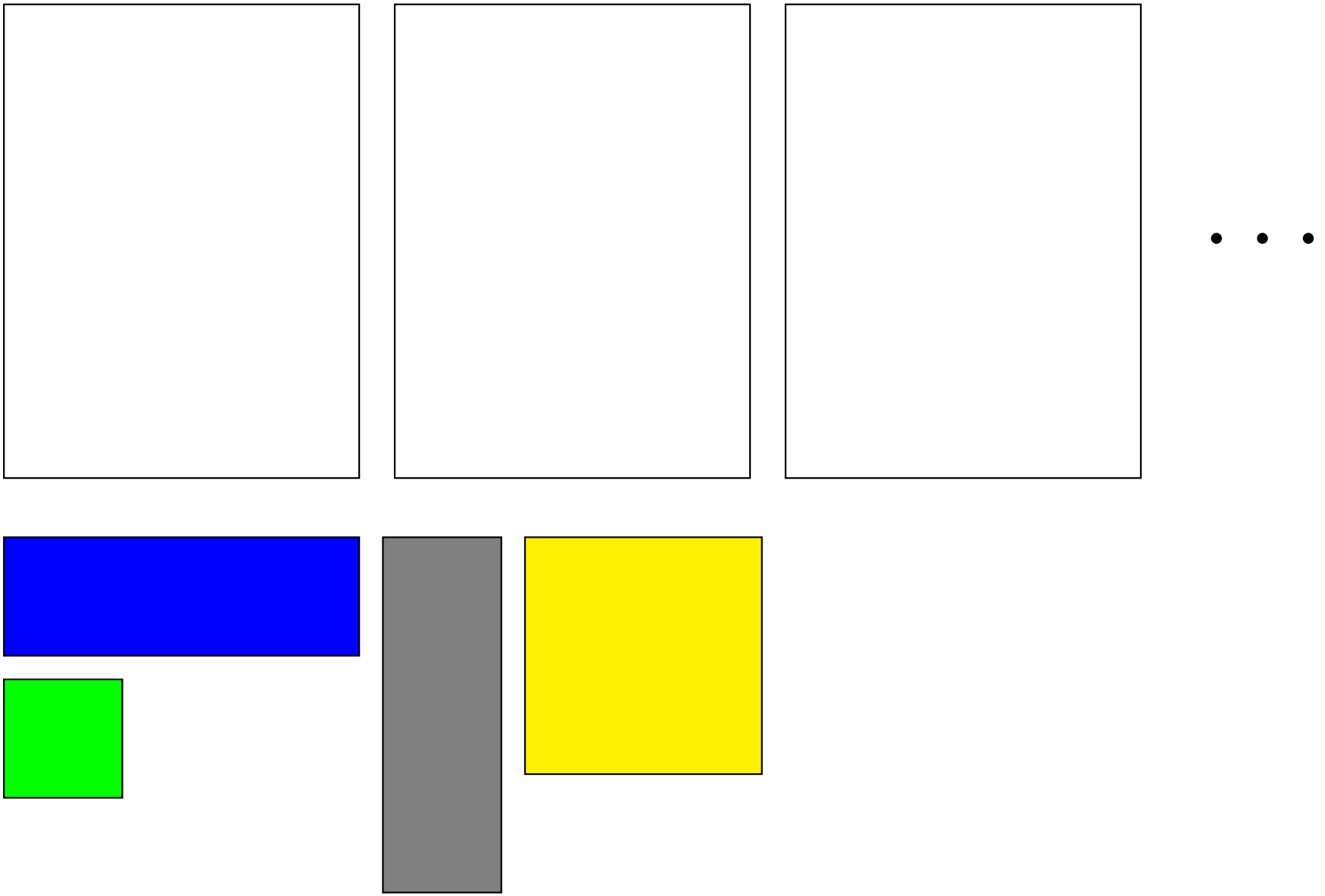
SMT $\equiv$ Satisfiability Modulo Theories; MINLP $\equiv$ Mixed-Integer Nonlinear optimisation

† For some energy systems problems, MINLP can reliably address 10^3 variables/constraints; but 10^2 variables/constraints are more typical for general problem classes

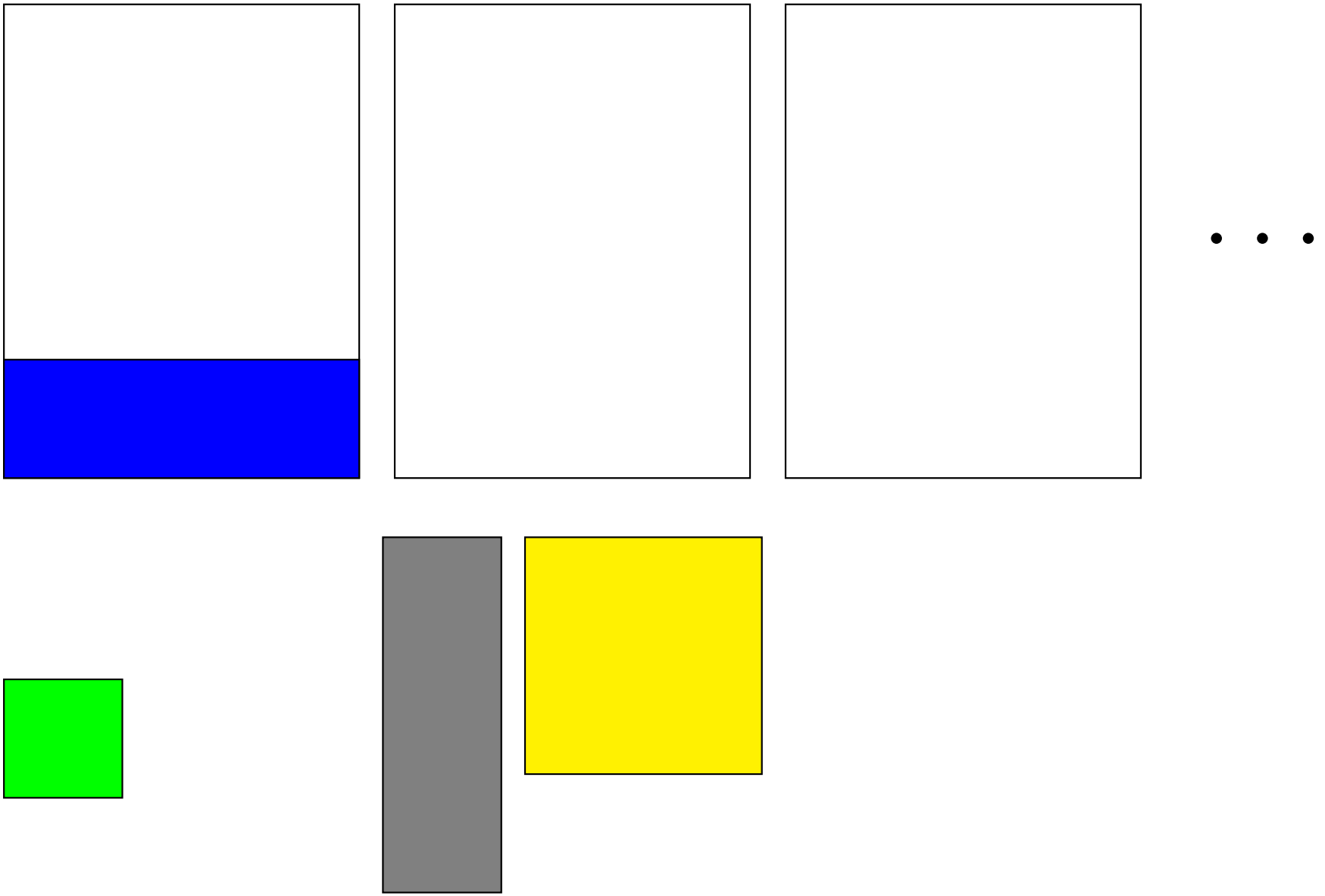
Two Dimensional Bin Packing (2BP)



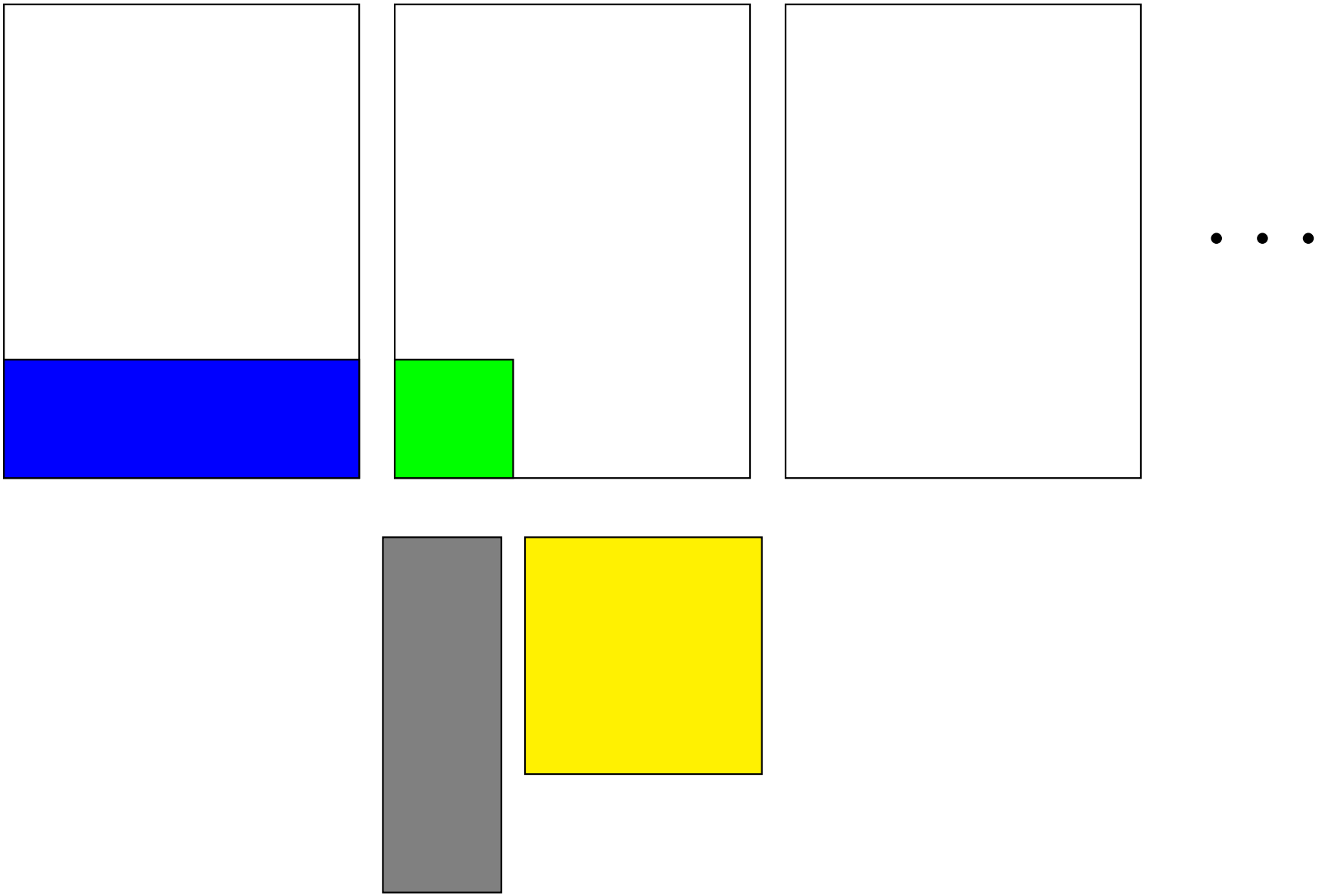
Two Dimensional Bin Packing (2BP)



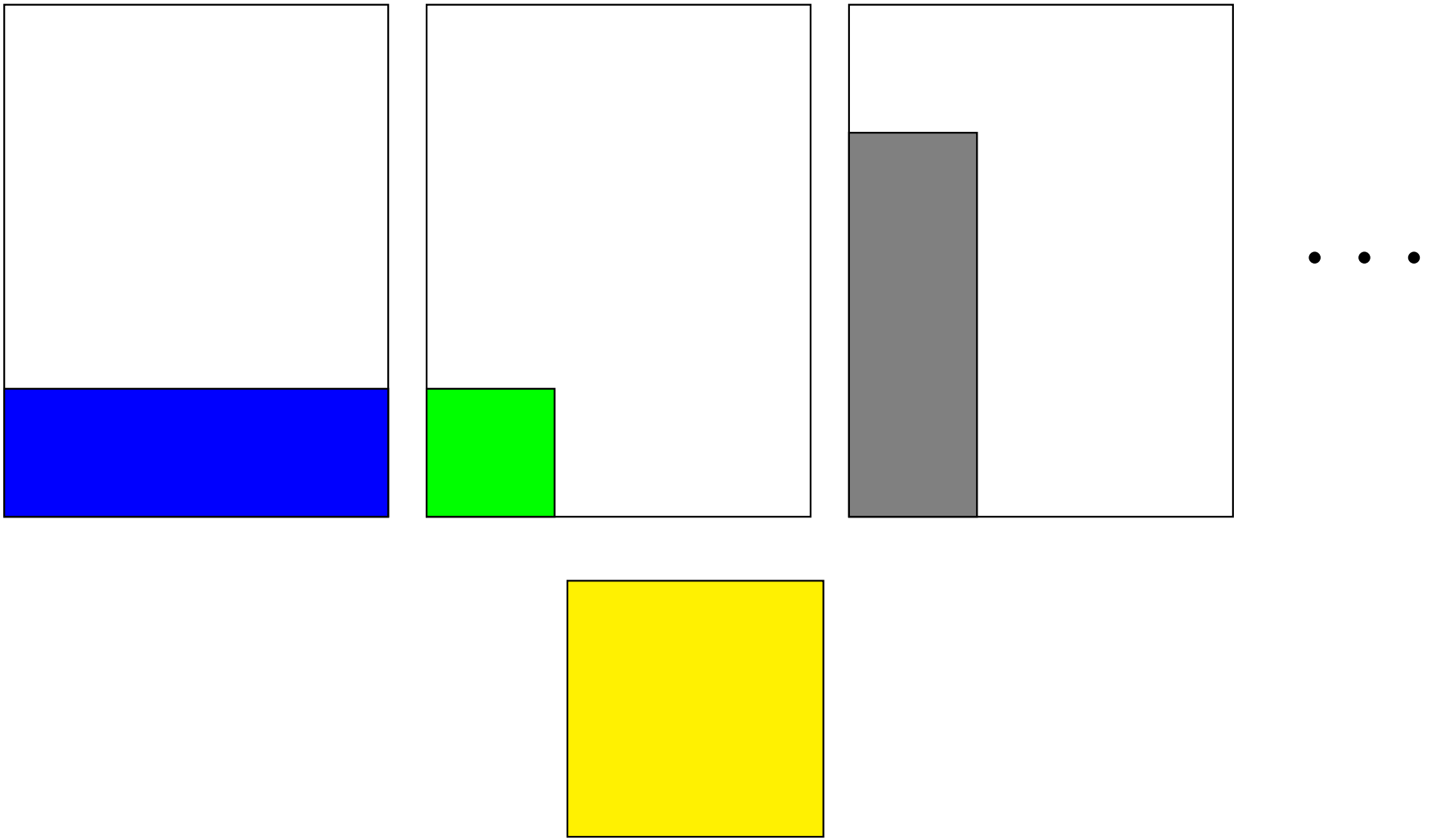
Two Dimensional Bin Packing (2BP)



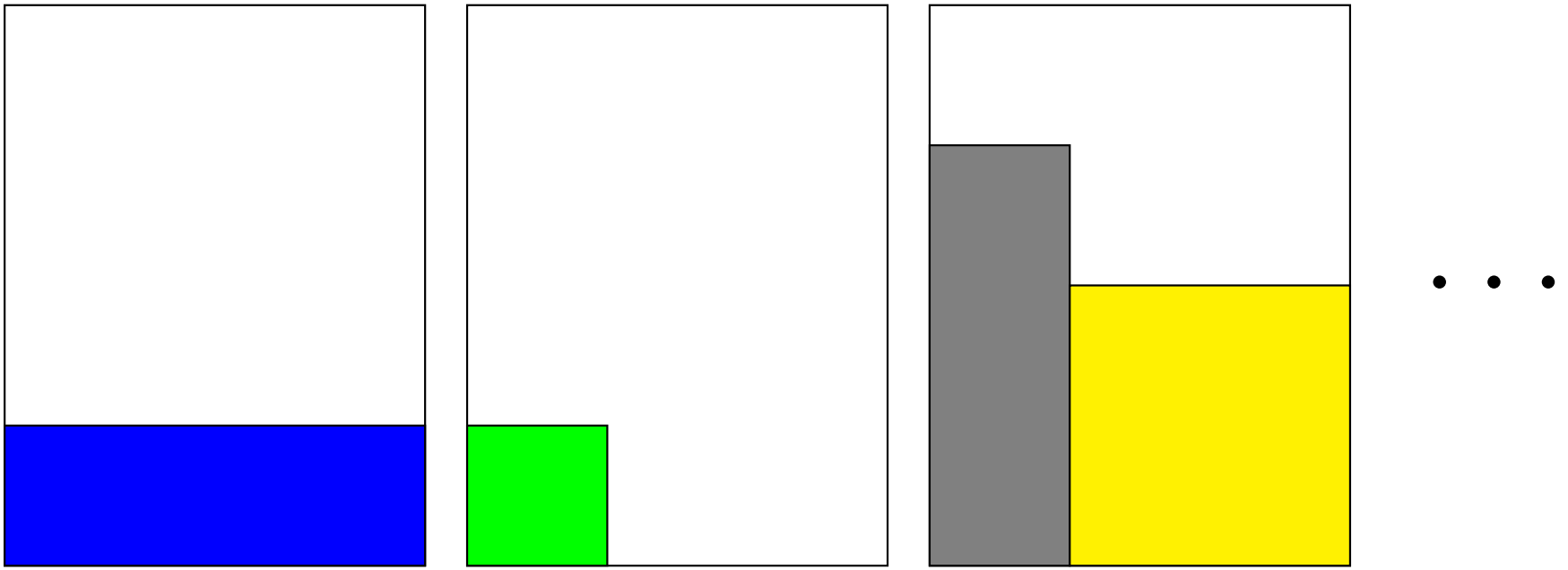
Two Dimensional Bin Packing (2BP)



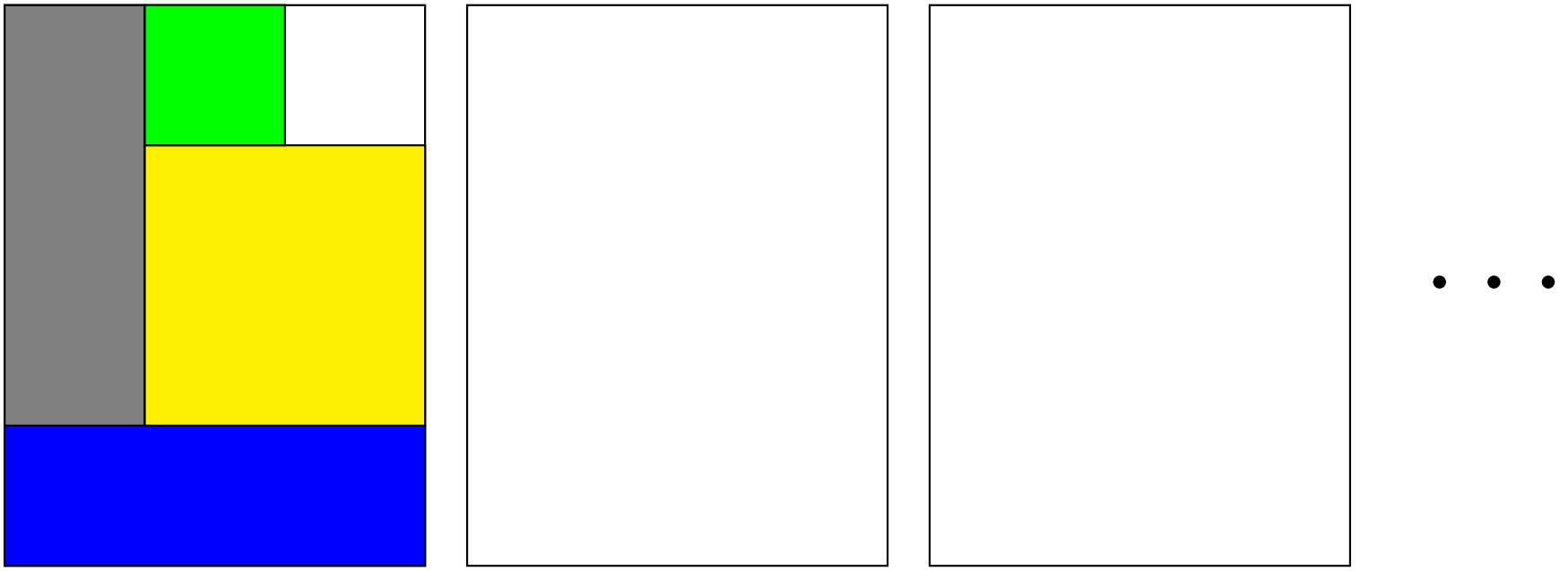
Two Dimensional Bin Packing (2BP)



Two Dimensional Bin Packing (2BP)



Two Dimensional Bin Packing (2BP)



Logical Formulation

- Given items $j \in \{1, \ldots, n\}$ with widths W_j and heights H_j.
- Given n bins with width W and height H.

$$\min \sum_{b=1}^{n} c_b$$

$$\text{s.t.} \bigvee_{b=1}^{n} \left(\zeta_{jb} \wedge \bigwedge_{b' \neq b} \neg \zeta_{jb'} \right)$$

$$(\zeta_{ib} \wedge \zeta_{jb}) \rightarrow \begin{cases} (x_i + W_i \leq x_j) \vee (x_j + W_j \leq x_i) \\ \quad \vee (y_i - H_i \geq y_j) \vee (y_j - H_j \geq y_i) \end{cases}$$

$$\zeta_{jb} \rightarrow \zeta_b$$

$$\zeta_b \rightarrow (c_b = 1), \neg \zeta_b \rightarrow (c_b = 0)$$

$$0 \leq x_j \leq W_j,\ H_j \leq y_j \leq H$$

Mixed Integer Formulation

$$\min \sum_{b=1}^{n} z_b$$

$$\text{s.t.} \sum_{b=1}^{n} z_{jb} = 1$$

$$x_i + W_i \leq x_j + M_{ij}^1(1 - z_{ij}^1),\ x_j + W_j \leq x_i + M_{ij}^2(1 - z_{ij}^2)$$

$$y_i - H_i \geq y_j - M_{ij}^3(1 - z_{ij}^3),\ y_j - H_j \geq y_i - M_{ij}^4(1 - z_{ij}^4)$$

$$\sum_{k=1}^{4} z_{ij}^k = \sum_{b=1}^{n} z_{ib} z_{jb}$$

$$z_b \geq z_{jb}$$

$$0 \leq x_j \leq W_j,\ H_j \leq y_j \leq H$$

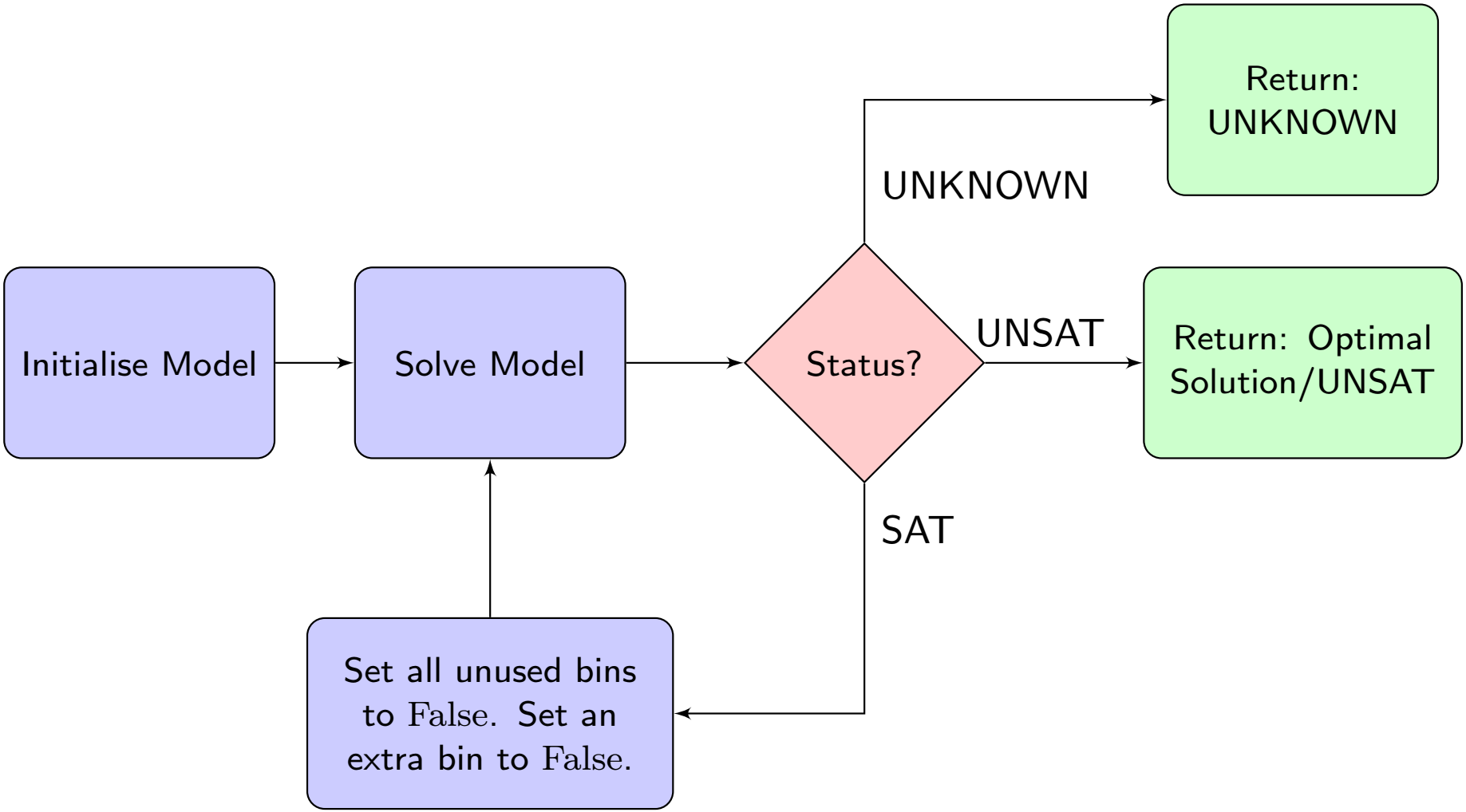
Big-M Formulation, No Symmetry Breaking

Lodi et al. (2002); Trespalacios and Grossmann (2016)

Previous Work

- In the logical model, we do not need variables c_b
- The optimisation part of the algorithm becomes the algorithm
- Instead of minimising a sum, we can activate/deactivate bins directly
- Bin packing has symmetry across bins, activating/deactivating bins in particular order can reduce this

Primal SMT

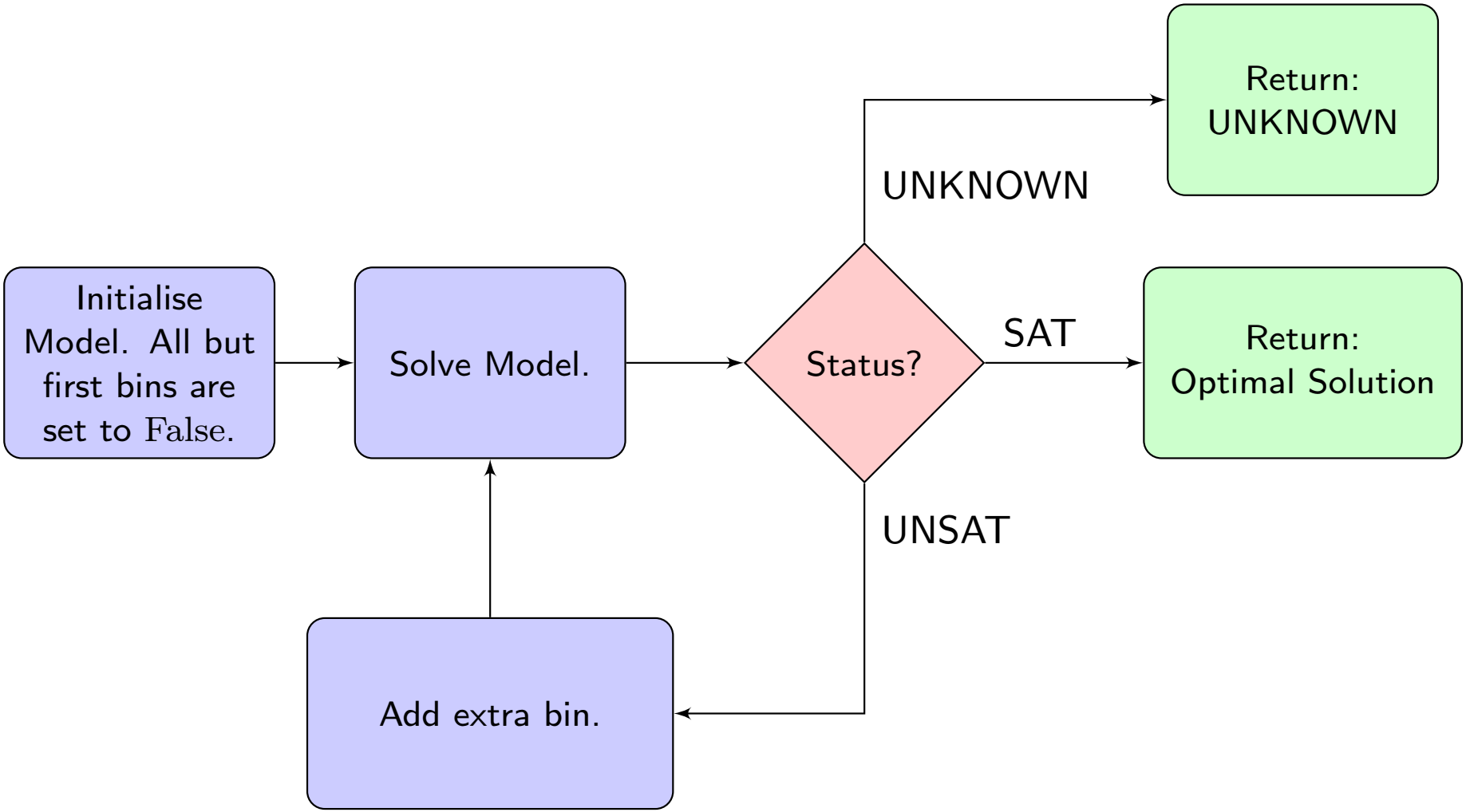

Initialise Model
Solve Model
Status?
UNKNOWN
Return: UNKNOWN
UNSAT
Return: Optimal Solution/UNSAT
SAT
Set all unused bins to False. Set an extra bin to False.

Primal SMT

Algorithm 1 The primal algorithm.

1: $numActiveBins \leftarrow N$
2: **repeat**
3: **for** $numActiveBins + 1 \leq b \leq N$ **do**
4: $z_b = \text{False}$
5: $result \leftarrow$ solve model
6: **if** $result = \text{SAT}$ **then**
7: $numActiveBins \leftarrow$ Number of used bins $- 1$
8: **until** $result \neq \text{SAT}$
9: **return** Last SAT solution

Dual SMT


Initialise Model. All but first bins are set to False.
Solve Model.
Status?
UNKNOWN
Return: UNKNOWN
SAT
Return: Optimal Solution
UNSAT
Add extra bin.

Dual SMT

Algorithm 2 The dual algorithm.

1: $numActiveBins \leftarrow 1$
2: **repeat**
3: **for** $numActiveBins + 1 \leq b \leq N$ **do**
4: $z_b = \text{False}$
5: $result \leftarrow$ solve model
6: **if** $result = \text{UNSAT}$ **then**
7: $numActiveBins \leftarrow numActiveBins + 1$
8: **until** $result \neq \text{UNSAT}$
9: **return** SAT solution or timeout

The Unsatisfiable Core

- An SMT solver results in:
  - Satisfiability - with a witness
  - Unsatisfiability - with an unsatisfiable core
- An unsatisfiable core is subset of the constraints that are mutually responsible for the unsatisfiability
- In this respect, SMT is a theorem prover
- For bin packing, we are interested in unsatifiable constraints associated with overlaps
- With this we can construct a global solution

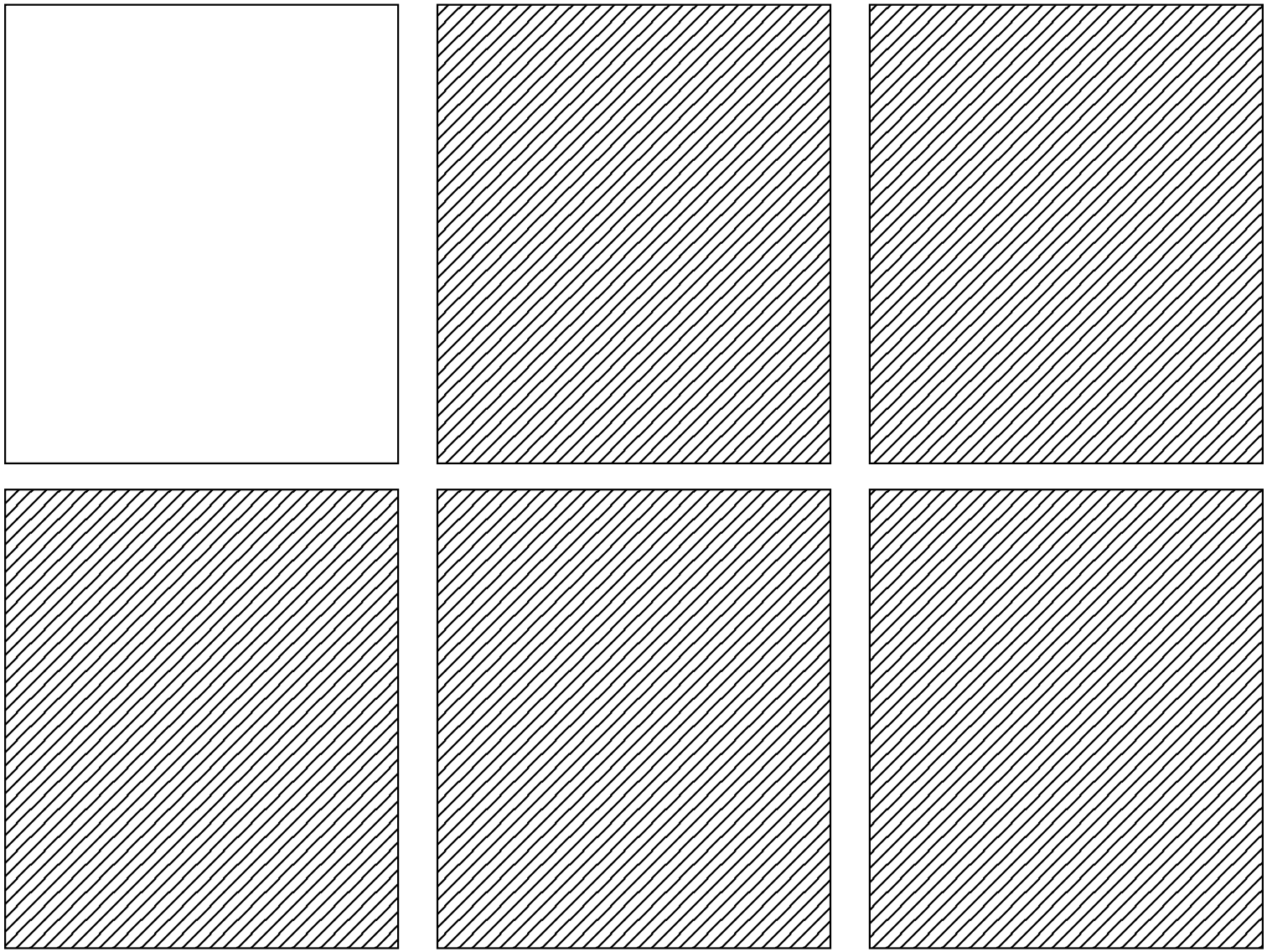
Dual SMT - Using the Unsatisfiable Core

Algorithm 3 How the strong cuts are derived.

```
1:  r ← 2
2:  n ← SIZE(core)
3:  while r ≤ n ∧ (n choose r) ≤ LIMIT do
4:      filteredSubsets ← unchecked and undominated size r subsets of core
5:      if filteredSubsets = ∅ then
6:          break
7:      for subset in filteredSubsets do
8:          if CHECK(subset) = UNSAT then
9:              ADDCUT(subset)
10:     r ← r + 1
```

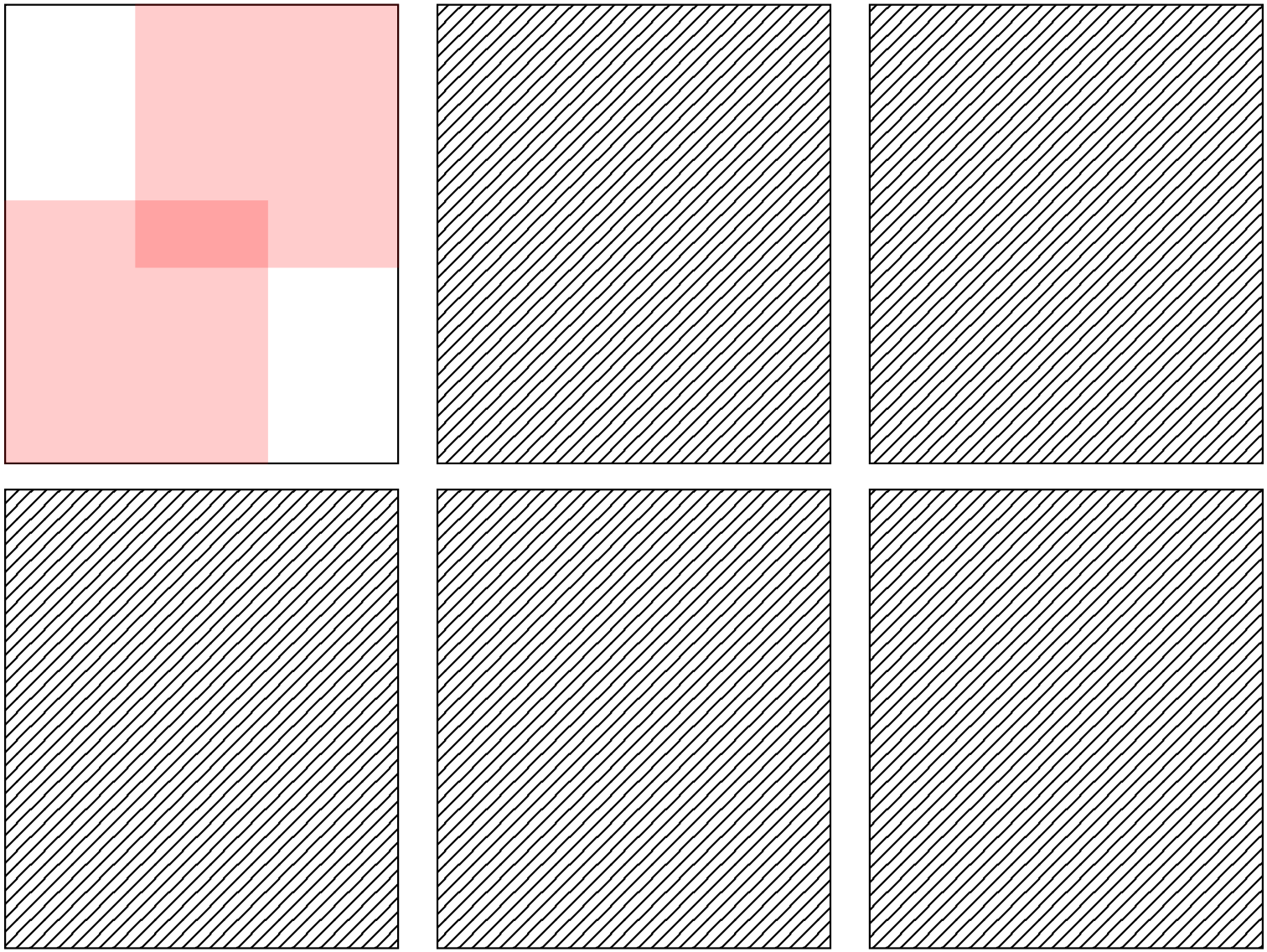
Dual SMT

Fixing items in bins



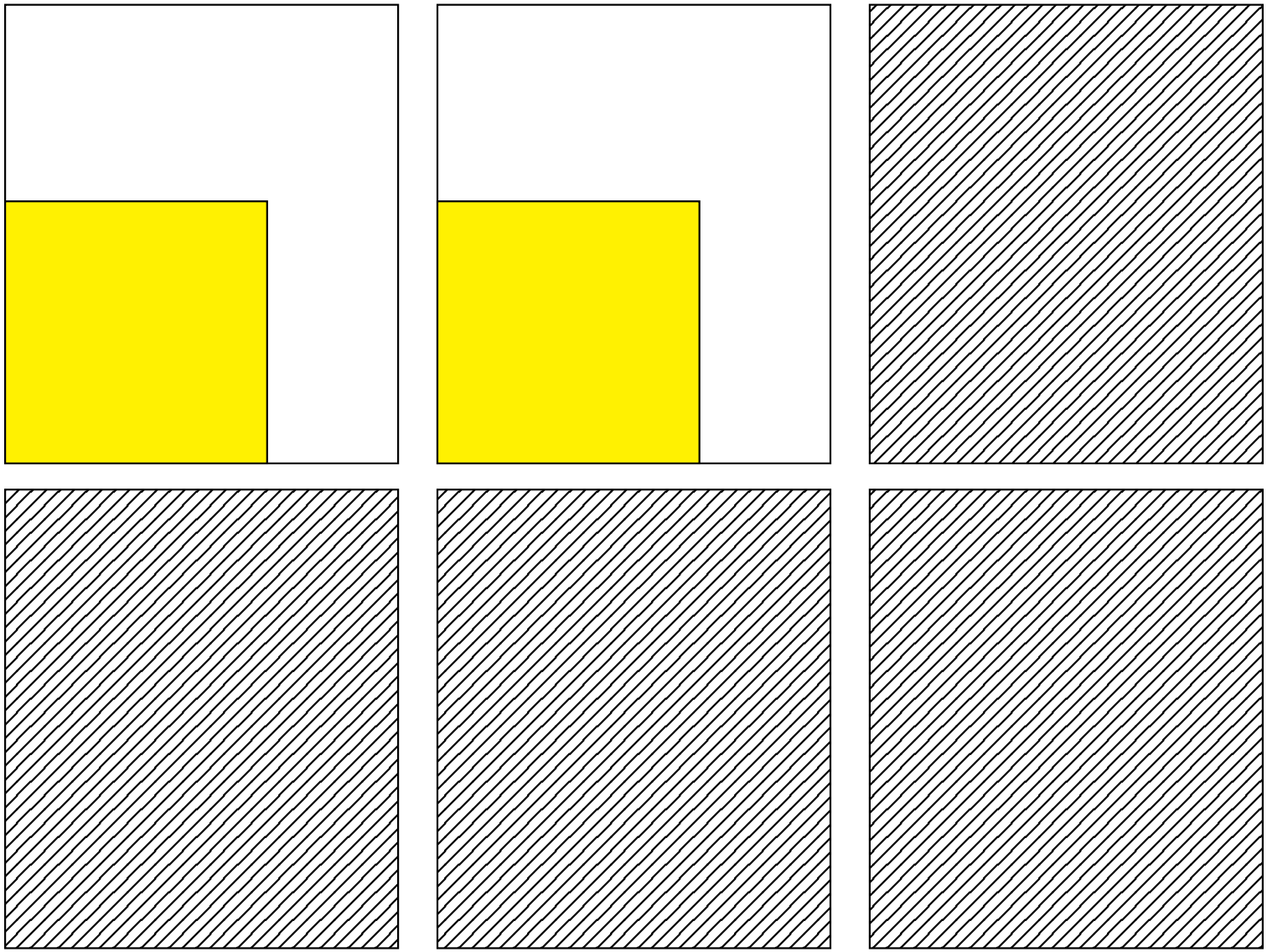
Dual SMT

Fixing items in bins



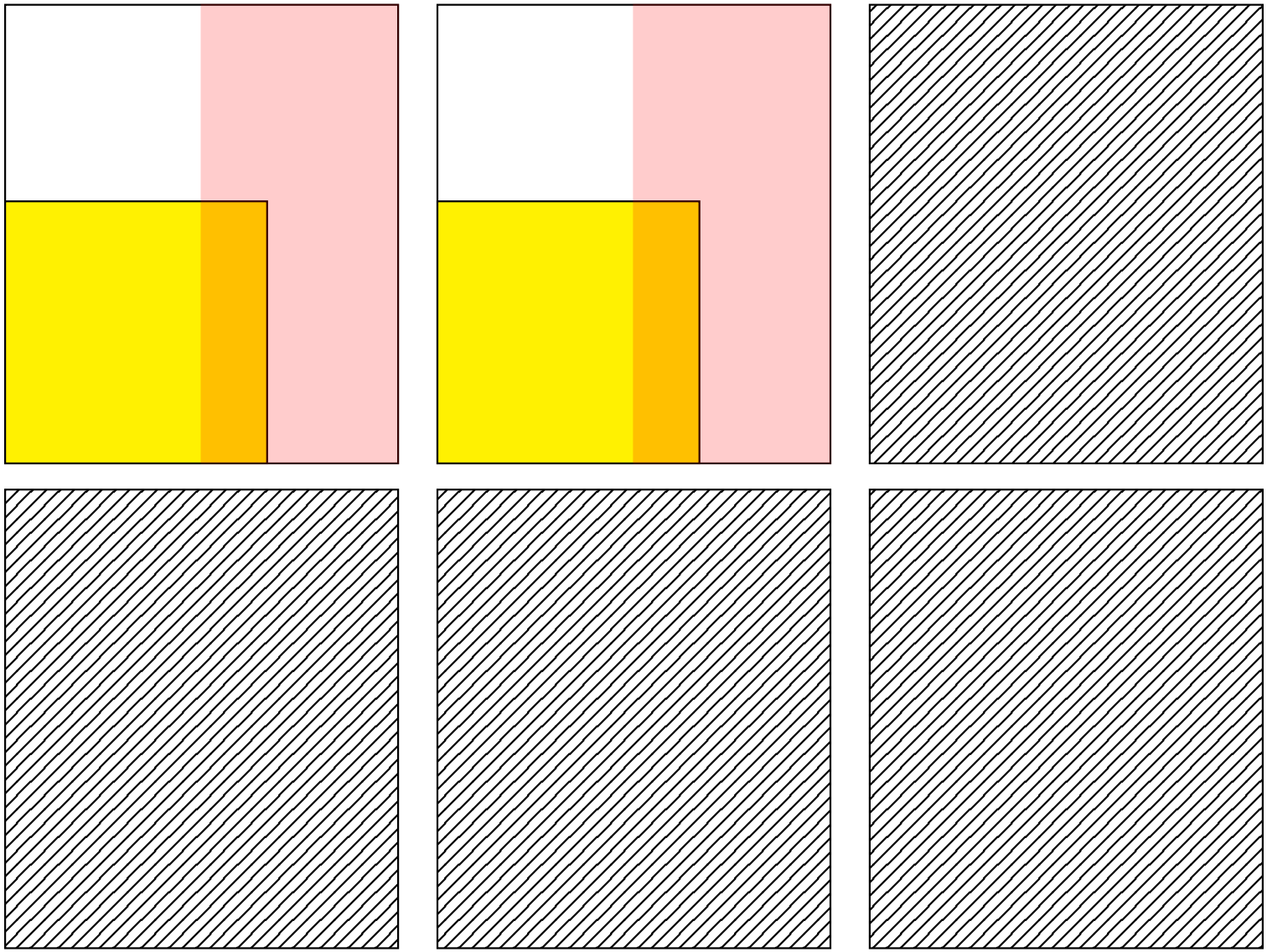
Dual SMT

Fixing items in bins



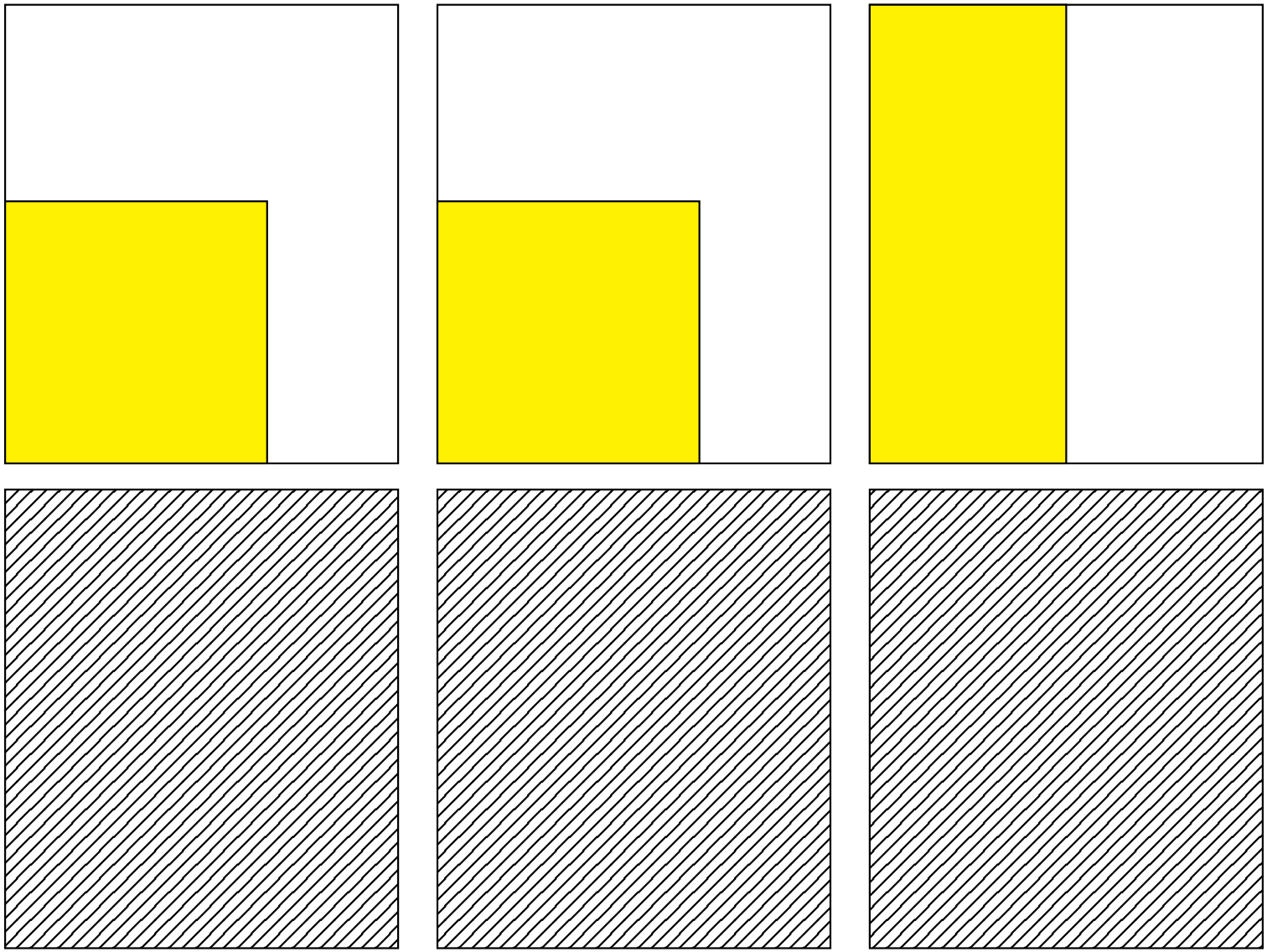
Dual SMT

Fixing items in bins



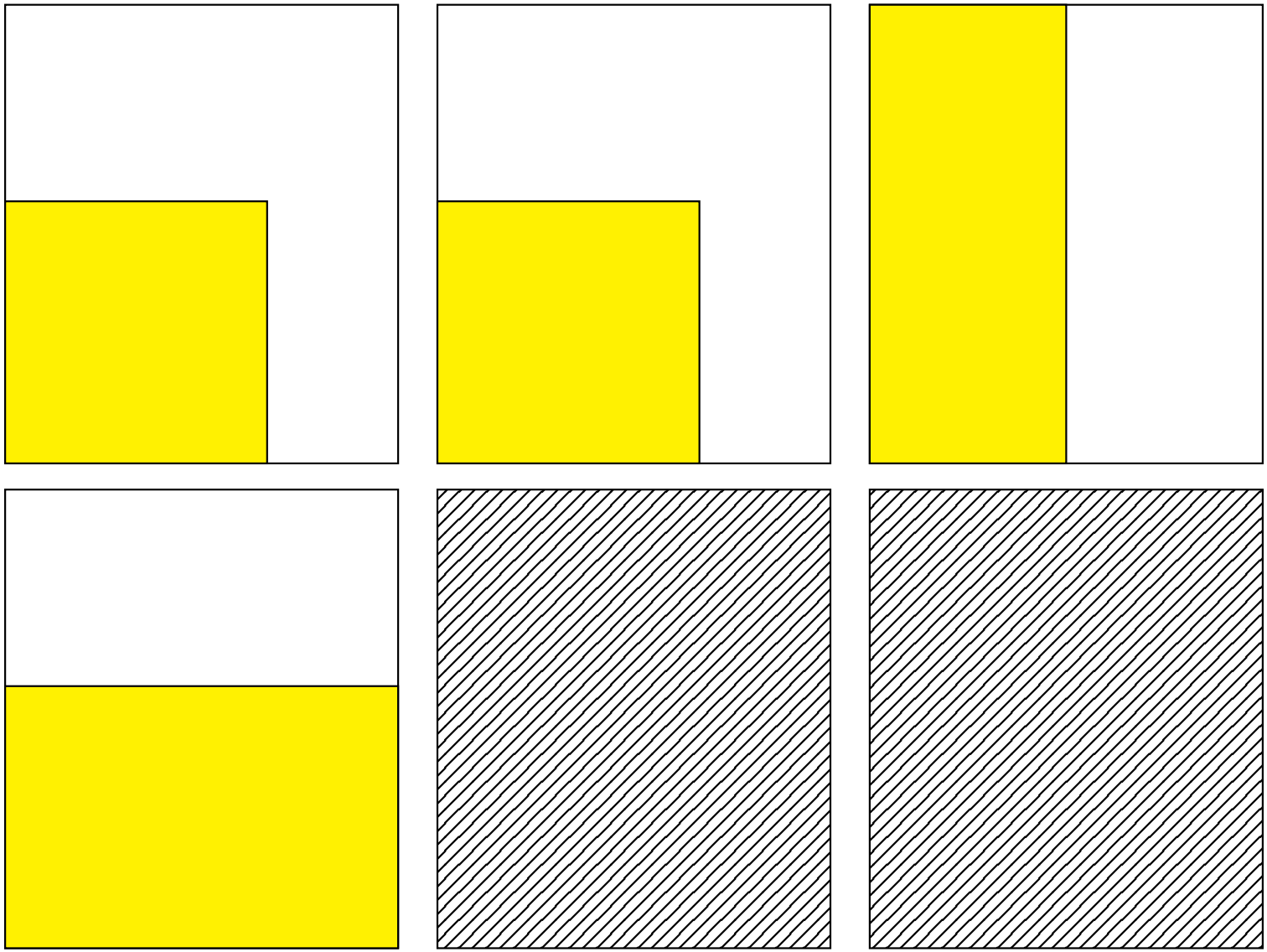
Dual SMT

Fixing items in bins



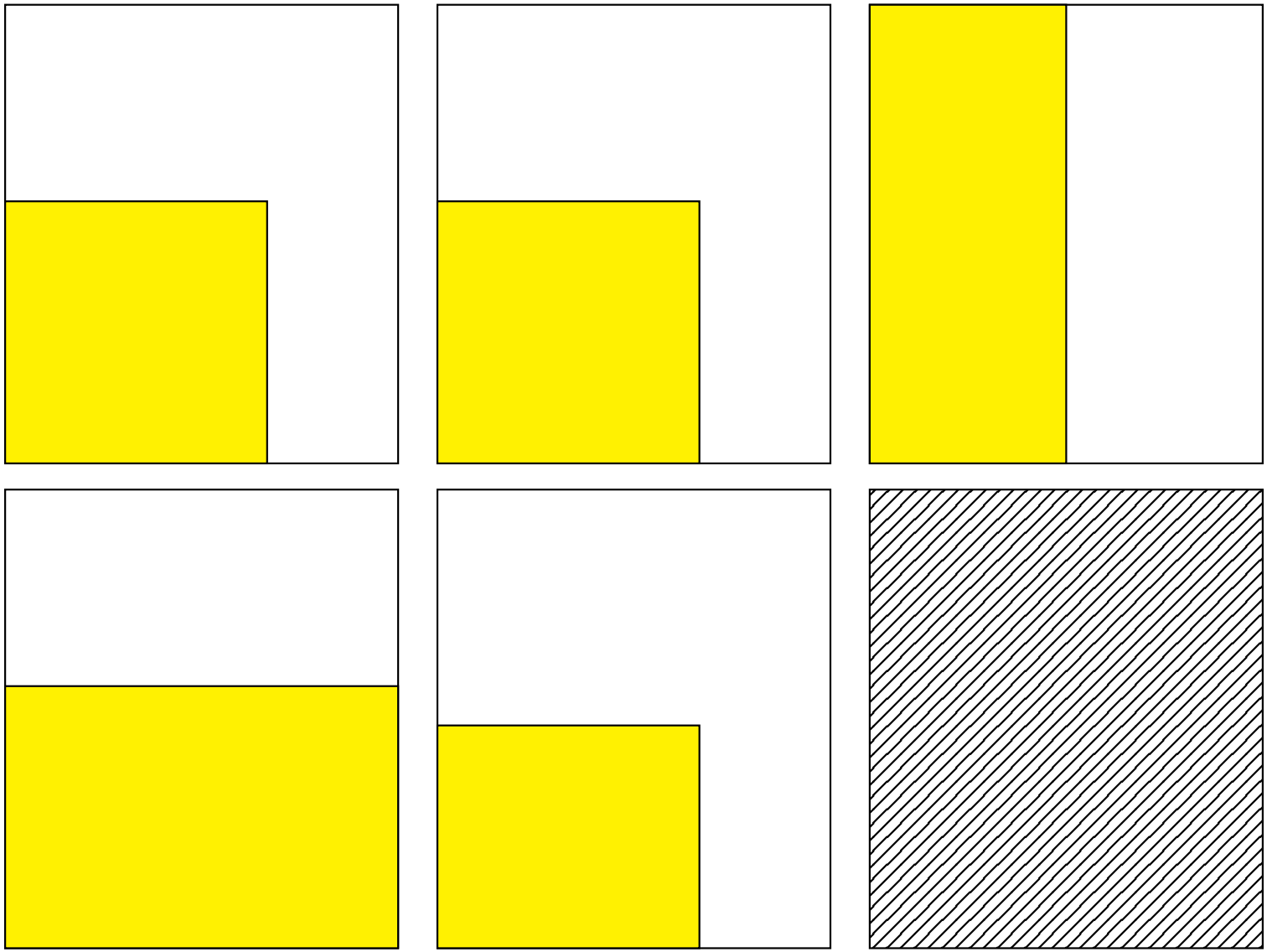
Dual SMT

Fixing items in bins



Dual SMT

Fixing items in bins



Numerical Results

Problem Set

- 500 problems (Lodi et al., 1999)
  - 10 classes
  - each class has 50 problems
  - each problem has 20,40,60,80 or 100 items (10 of each)
- Each problem given a time limit of 3600s
- SMT solver: Z3
- MILP solver: Gurobi

Numerical Results

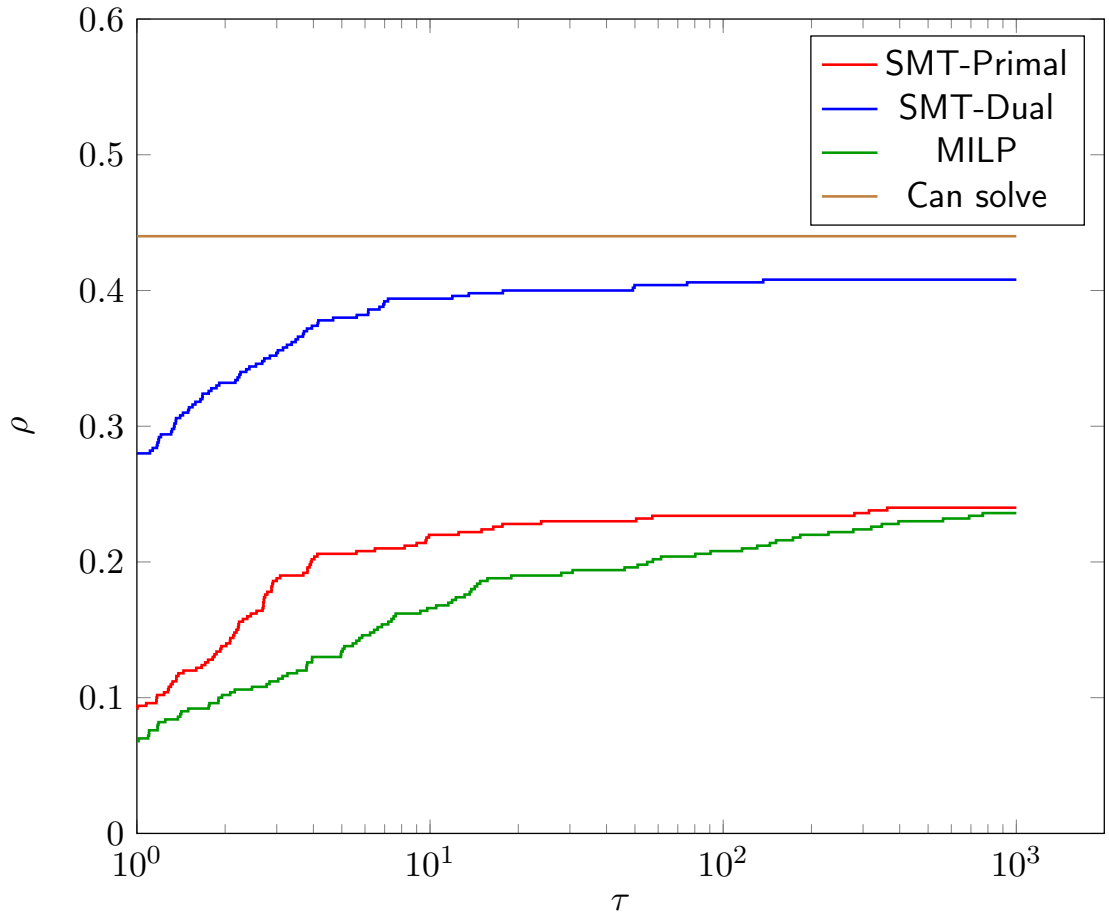
Problems Solved

Class	SMT-Primal	SMT-Dual	MILP
1	11	22	6
2	11	11	10
3	17	32	9
4	12	12	11
5	13	30	12
6	14	14	13
7	10	11	1
8	10	11	2
9	1	37	49
10	21	24	7

Table: Number of problems solved to optimality within 3600s by each of the approaches. Each problem class has 50 problems.

Numerical Results (500 Bin packing problems, 20-100 items)

Performance Profile (Dolan and Moré, 2002)



Best in class: Pisinger and Sigurd (2007)

Constraint satisfaction + Dantzig-Wolfe decomposition

Numerical Results

Pecentage Gaps

- Percentage gap: $\frac{\text{upper bound} - \text{lower bound}}{\text{upper bound}}$

Class	SMT	MILP
1	12.5	24.6
2	53.7	66.2
3	4.9	33.3
4	51.7	65.0
5	9.3	21.0
6	47.3	61.1
7	31.2	60.0
8	31.1	58.4
9	11.8	0.1
10	11.6	53.4

2D Bin Packing

Results with use of the UNSAT Core

- Solvers have a timelimit of 1 hour set

	Upper Bound		Lower Bound	
	SMT	MILP BM	SMT	MILP BM
32 pieces	9^*	9^*	9	8^*
50 pieces	19^*	20^*	19	16^*
75 pieces	21^*	24^*	20^*	17^*

Table: 2D Bin packing bounds achieved in 1 hour, (*not proven to be optimal)

	SMT	MILP BM
32 pieces	0%	11.1%
50 pieces	0% (42.2%)	20.0%
75 pieces	4.8% (47.7%)	29.1%

Table: 2D Bin packing gaps achieved in 1 hour, smaller is better

Logical Cuts Applied to a MIP Solver

- We can add the logical cuts by transforming them into an equivalent inequality (in the context of the mathematical optimisation model)
  - Square brackets are for clarity only

$$\left[\zeta_{pi}\right] \equiv \left[z_{pi} = 1\right]$$

$$\left[\bigvee_{k=1}^{n} \neg\zeta_{p_k i}\right] \equiv \left[\sum_{k=1}^{n} z_{p_k i} \leq n - 1\right]$$

	SMT	MILP BM + Logical Cuts
32 pieces	0%	0% (11.1%)
50 pieces	0%	10.5% (20.0%)
75 pieces	4.8%	9.52% (29.1%)

Table: 2D Bin packing gaps achieved in 1 hour, smaller is better

Generalized Disjunctive Programming

$$
\begin{aligned}
\min \quad & Z = f(\boldsymbol{x}) + \sum_{k \in K} c_k \\
\text{s.t.} \quad & g(\boldsymbol{x}) \leq \mathbf{0} \\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ r_{ik}(\boldsymbol{x}) \leq \mathbf{0} \\ c_k = \gamma_{ik} \end{bmatrix} \qquad k \in K \\
& \Omega(\boldsymbol{Y}) = \mathit{True} \\
& \boldsymbol{x_l} \leq \boldsymbol{x} \leq \boldsymbol{x_u} \\
& \boldsymbol{x} \in \mathbb{R}^n, c_k \in \mathbb{R}, Y_{ik} \in \{\mathit{True}, \mathit{False}\}
\end{aligned}
$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, each disjunction $k \in K$ has conjunctions $i \in D_k$. Conjunctions have: boolean variable Y_{ik}, inequality $r_{ik}(\boldsymbol{x}) \leq \mathbf{0}$, $r_{ik} : \mathbb{R}^n \to \mathbb{R}^p$, and cost variable c_k. If $Y_{ik} = \mathit{True}$, then $r_{ik}(\boldsymbol{x}) \leq \mathbf{0}$ and $c_k = \gamma_{ik}$ are enforced. GDP has propositional formula $\Omega(\boldsymbol{Y}) = \mathit{True}$.

Mixed Logical-Linear Programming

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} \quad & p_j(\boldsymbol{Y}, \boldsymbol{y}) \rightarrow (\boldsymbol{A_j x} \geq \boldsymbol{a_j}),\ j \in \mathcal{J} \ \Big|\ q_i(\boldsymbol{Y}, \boldsymbol{y}),\ i \in \mathcal{I}\ . \end{aligned}$$

where the constraints are split into continuous and logical parts, i.e. the constraints on the left and right of the bar. The logical part $q_i(\boldsymbol{Y}, \boldsymbol{y})$ has $\boldsymbol{Y}$, a vector of propositional variables, and $\boldsymbol{y}$, a vector of binary variables.

The continuous part is a logical implications. If $p_j(\boldsymbol{Y}, \boldsymbol{y})$ is true, then constraint $\boldsymbol{A_j x} \geq \boldsymbol{a_j}$ is imposed.

Scheduling

Problem Definiton

Given a set of facilities each with a **limit on the amount of resources** available at any given time. Also, given a set of tasks where each task has **release** and **due** times and per facility **resource consumption rate** and **processing time**. What is the optimal assignment of tasks to facilities such that we are optimal according to our objective?

- Objectives we study
  - Minimise cost (addition parameter - cost of assignment)
  - Minimise makespan
  - Minimise tardiness
- We study a hybrid strategy
  - We implement the hybrid formulations of Hooker (2007)
- Occurs in the manufacturing and supply chain context

Why Hybrid?

- Planning and scheduling is heavily constrained problem to be optimised
- Constraints associated with assignment, time scheduling and resource consumption
- Using a hybrid approach, optimisation can be delegated to a optimiser and constraint satisfaction can be delegated to a constraint satisfaction solver

Mixed Integer Formulation

- The problem can be formulated for a mixed integer context in different ways
- Continuous time model: based on there being $2n$ events, n start and n finishes
  - Does not perform well
- Discrete time model
  - Performs much better
  - Can potentially have a large number of constraints based on the discretisation

Mixed Integer Formulation

Discrete time model – Minimum cost

$$\min \sum_{ijt} F_{ij} x_{ijt}$$

$$\text{s.t.} \sum_{it} x_{ijt} = 1$$

$$\sum_{j} \sum_{t' \in T_{ijt}} c_{ij} x_{ijt'} \leq C_i$$

$$x_{ijt} = 0, \quad \forall t < r_j \text{ or } t > d_j - p_{ij}$$

where x_{ijt} is binary and $T_{ijt} = \{t' \,|\, t - p_{ij} < t' \leq t\}$.

Logical Formulations

Continuous time model – Minimum cost

$$\min \sum_j f_j$$

$$\text{s.t. } s_j \geq r_j$$

$$\zeta_{ij} \rightarrow (s_j \leq d_j - p_{ij})$$

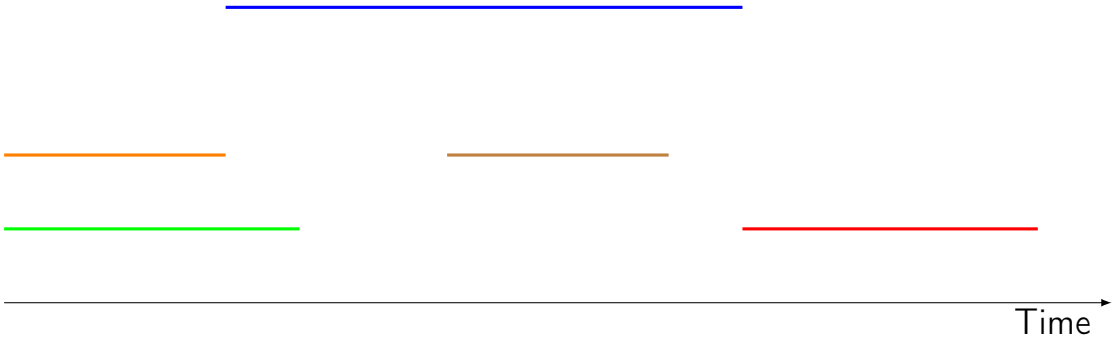
$$\bigvee_i \left(\zeta_{ij} \wedge \bigwedge_{i' \neq i} \neg \zeta_{ij} \right)$$

$$\bigwedge_{j \in \mathcal{J}'} \zeta_{ij} \rightarrow \left(\left(\sum_{j \in \mathcal{J}'} c_{ij} \leq C_i \right) \vee \bigvee_{j' \neq j} (s_j + p_{ij} \leq s'_j) \right)$$

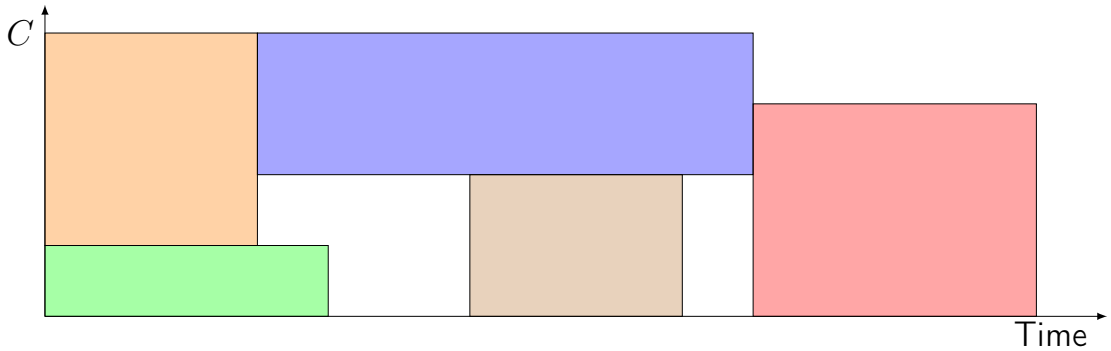
$$\zeta_{ij} \rightarrow (f_j = F_{ij})$$

Logical Formulations

Relation to Bin Packing



$$\sum_j c_{ij} \leq C_i$$



Link between resource constrained scheduling & bin packing

Garey et al. (1976); Castro and Grossmann (2012)

Logical Formulations

Relation to Bin Packing

- Exponential constraint

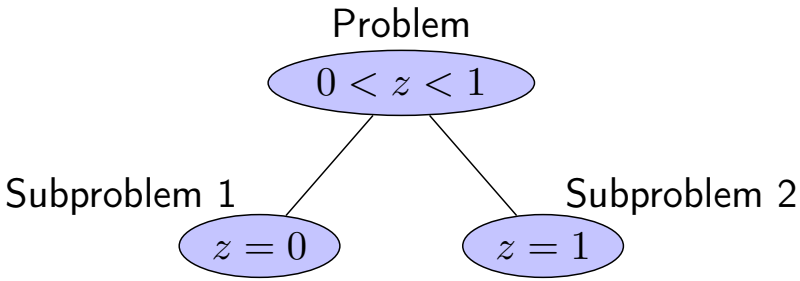
$$\bigwedge_{j \in \mathcal{J}'} \zeta_{ij} \rightarrow \left(\left(\sum_{j \in \mathcal{J}'} c_{ij} \leq C_i \right) \vee \bigvee_{j' \neq j} (s_j + p_{ij} \leq s'_j) \right)$$

- Squared constraint

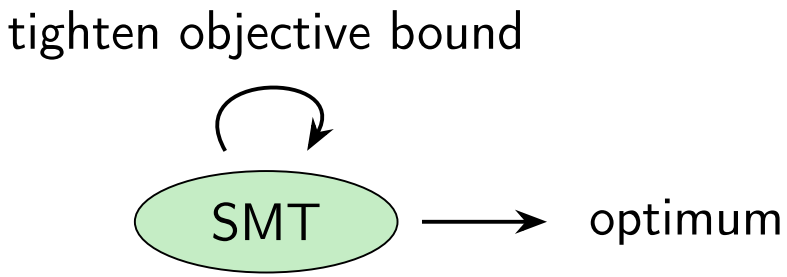
$$(\zeta_{ij} \wedge \zeta_{ij'}) \rightarrow \begin{cases} (s_j + p_{ij} \leq s_{j'}) \vee (s_{j'} + p_{ij'} \leq s_j) \\ \quad \vee (c'_j + c_{ij} \leq c'_{j'}) \vee (c'_{j'} + c_{ij'} \leq c'_j) \end{cases}$$

Optimisation strategies

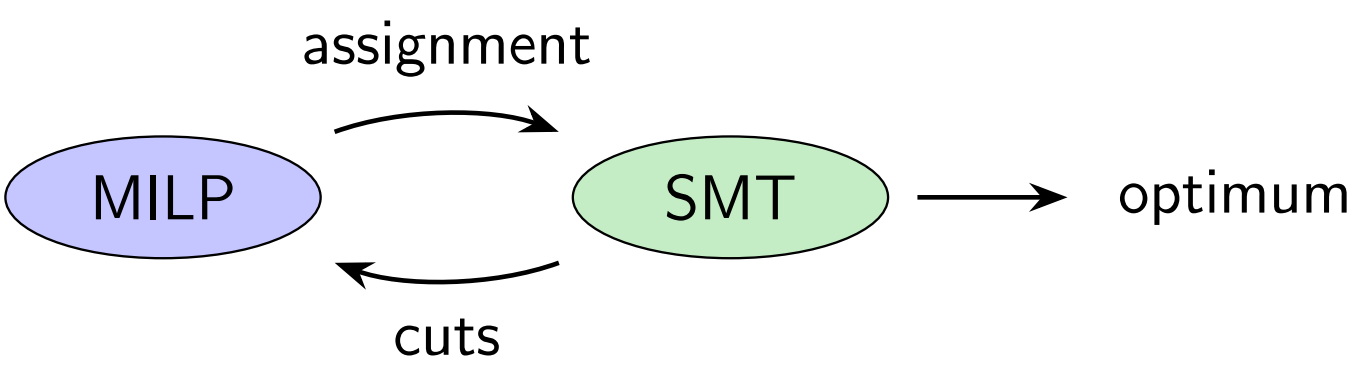
Mixed-Integer Programming



Satisfiability Modulo Theories



MIP + SMT



Benders Decomposition (Benders, 1962)

Method for solving MILPs

Key idea: Fixing variables that make the problem 'hard' and solving the underlying easier problem (subproblem).

Benders decomposition partitions the problem as

$$\begin{aligned} z = \min \quad & \boldsymbol{c}^\top \boldsymbol{x} + f(\boldsymbol{y}) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} + F(\boldsymbol{y}) \leq \boldsymbol{b}, \\ & \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{y} \in S \end{aligned}$$

where $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$, $S \subseteq \mathbb{R}^p$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $f : \mathbb{R}^p \to \mathbb{R}$ and $F : \mathbb{R}^p \to \mathbb{R}^m$. Here we would associate $\boldsymbol{y}$ with the difficult part (they do not have to be integral, e.g. f or F may be nonlinear).

Hybrid Scheduling

- Minimise cost
  - On each iteration, add a cut that removes the current assignment or a subset (subset results in a stronger cut)
- Minimise makespan
  - On each iteration, add a cut based on local makespans
- Minimise tardiness
  - On each iteration, add a cut based on local tardiness

Numerical Results

Experimental Setup

- 335 problems[1]
  - 4 classes of problems: c, de, df, e
  - 5 problems for each (i,j) instance
- Each problem given a time limit of 3600s
- Same set of problems used and generated by Hooker (2007). We compare our findings with his MILP/CP approach.
- SMT solver: Z3
- MILP solver: Gurobi
- Hooker (2007) reports that the hybrid approach is the best choice for each objective

[1] http://web.tepper.cmu.edu/jnh/instances.htm

Numerical Results

Minimum Cost

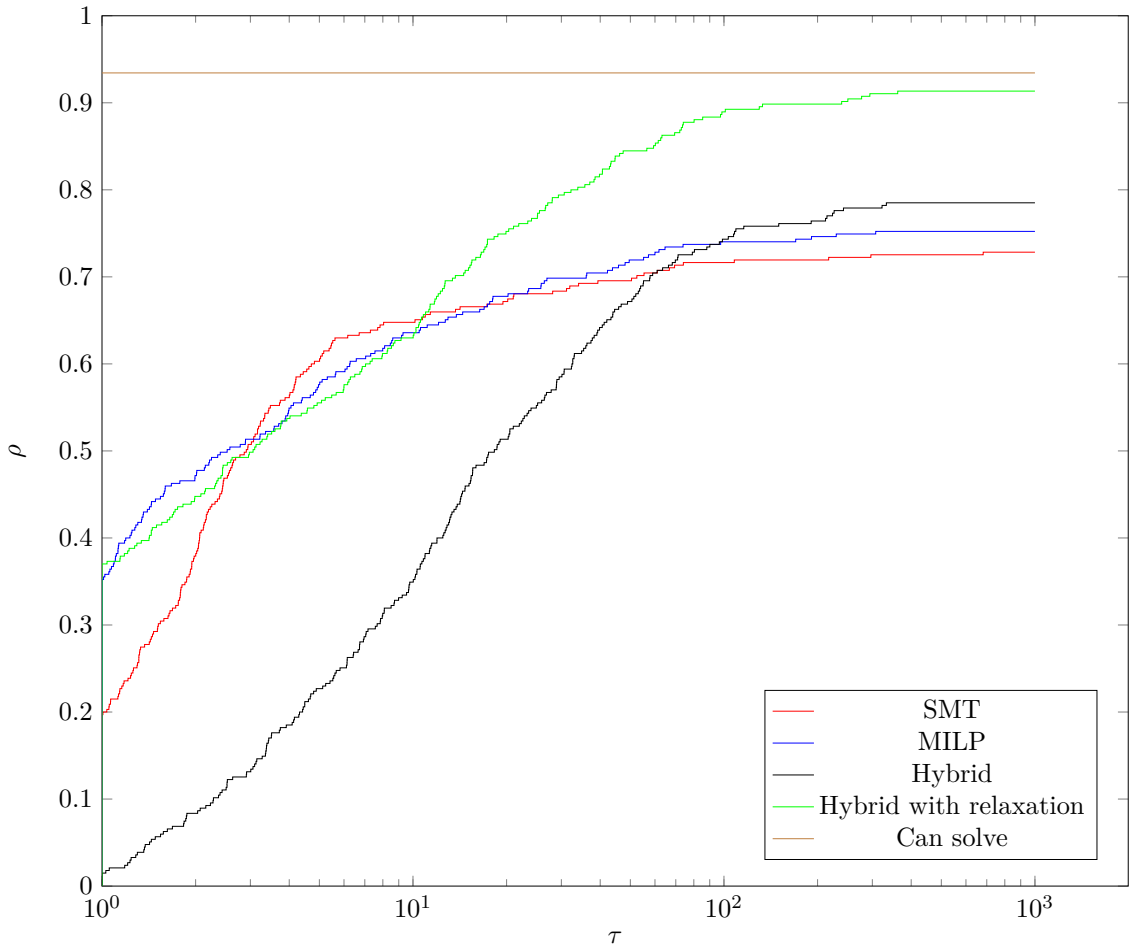


Figure: Solution time performance profile for all minimum cost models.

Numerical Results

Minimum Cost

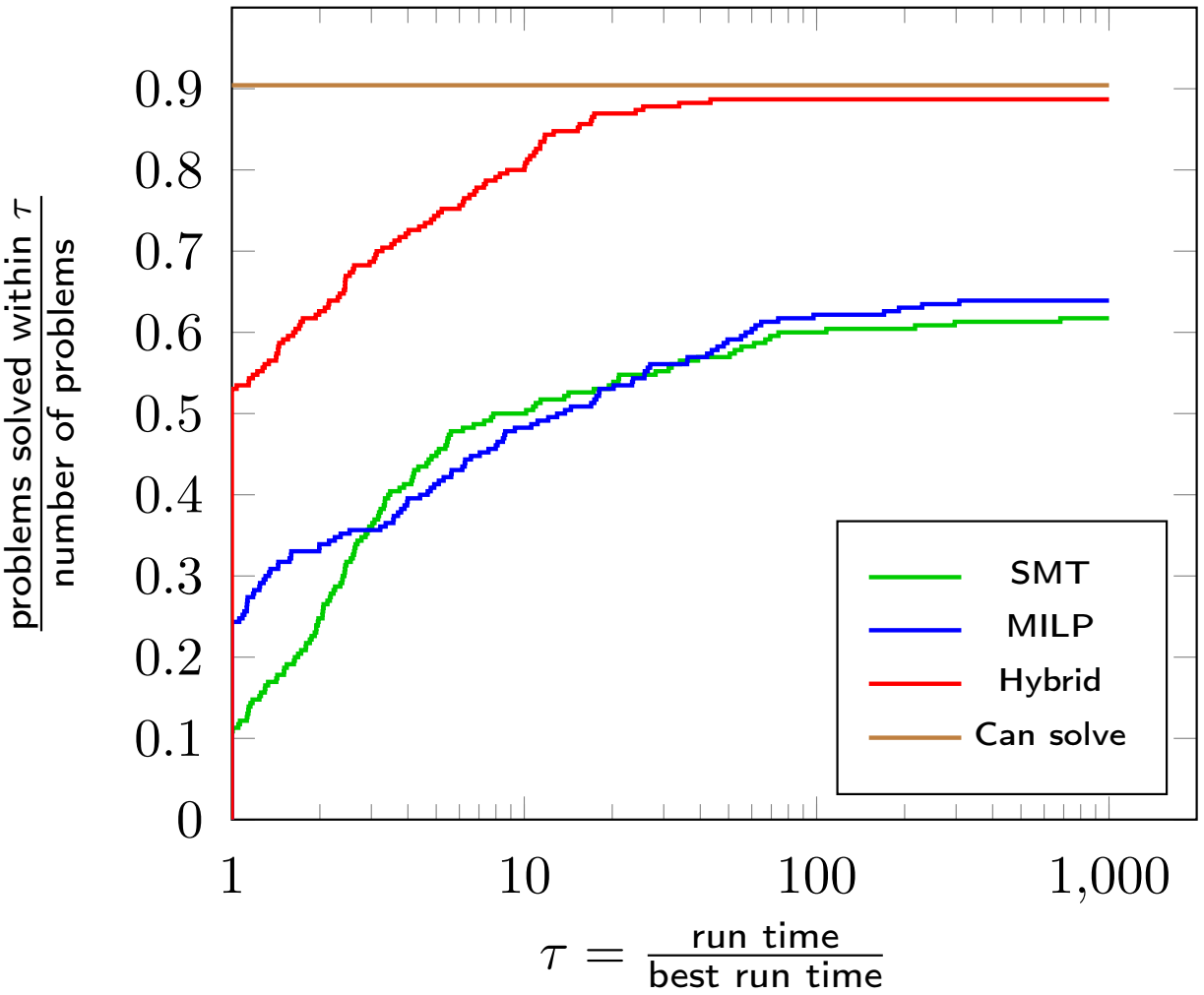


Figure: Solution time performance profile for 'non toy' minimum cost models.

Numerical Results

Minimum Makespan

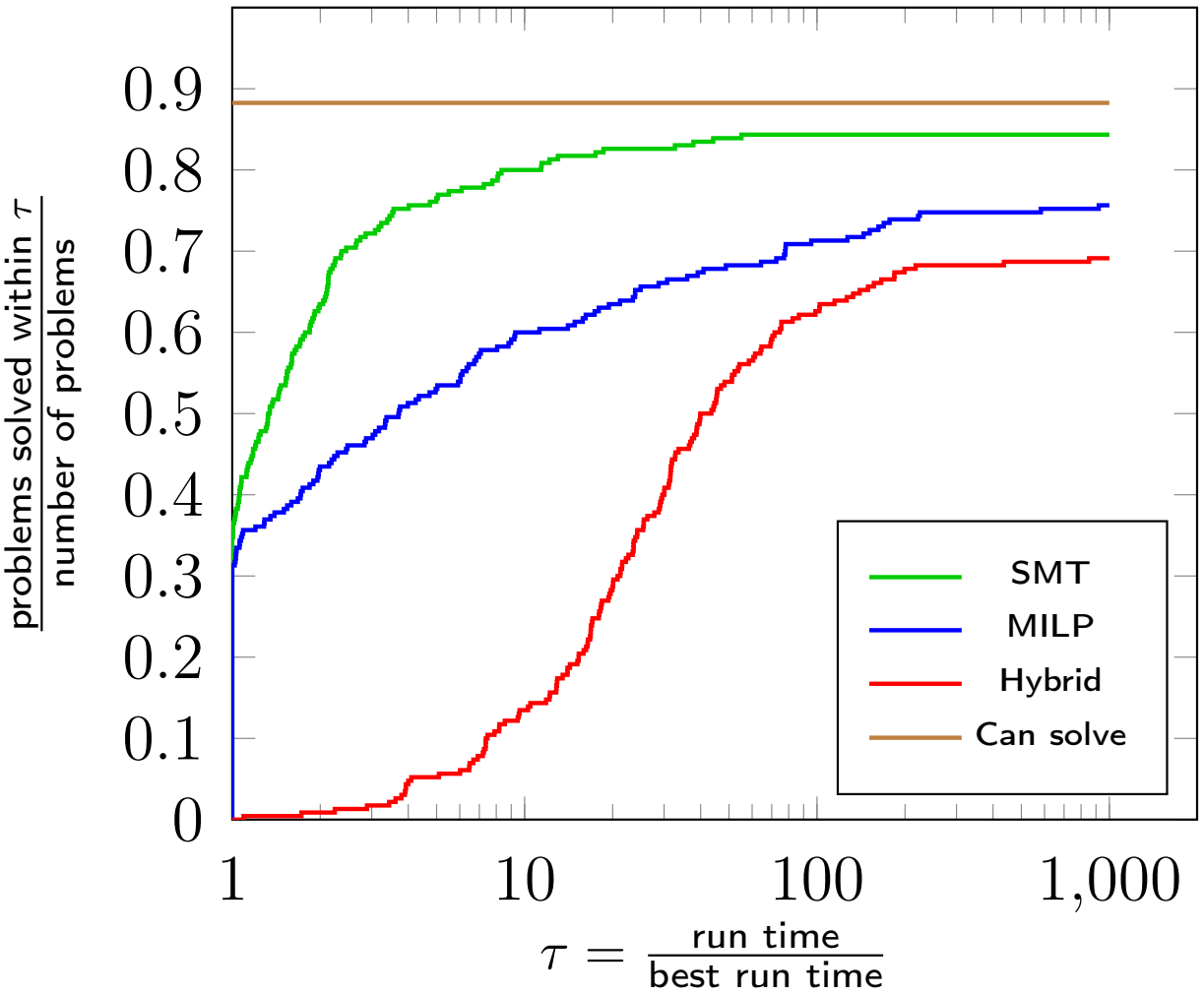


Figure: Solution time performance profile for 'non toy' minimum makespan problems.

Numerical Results

Minimum Tardiness

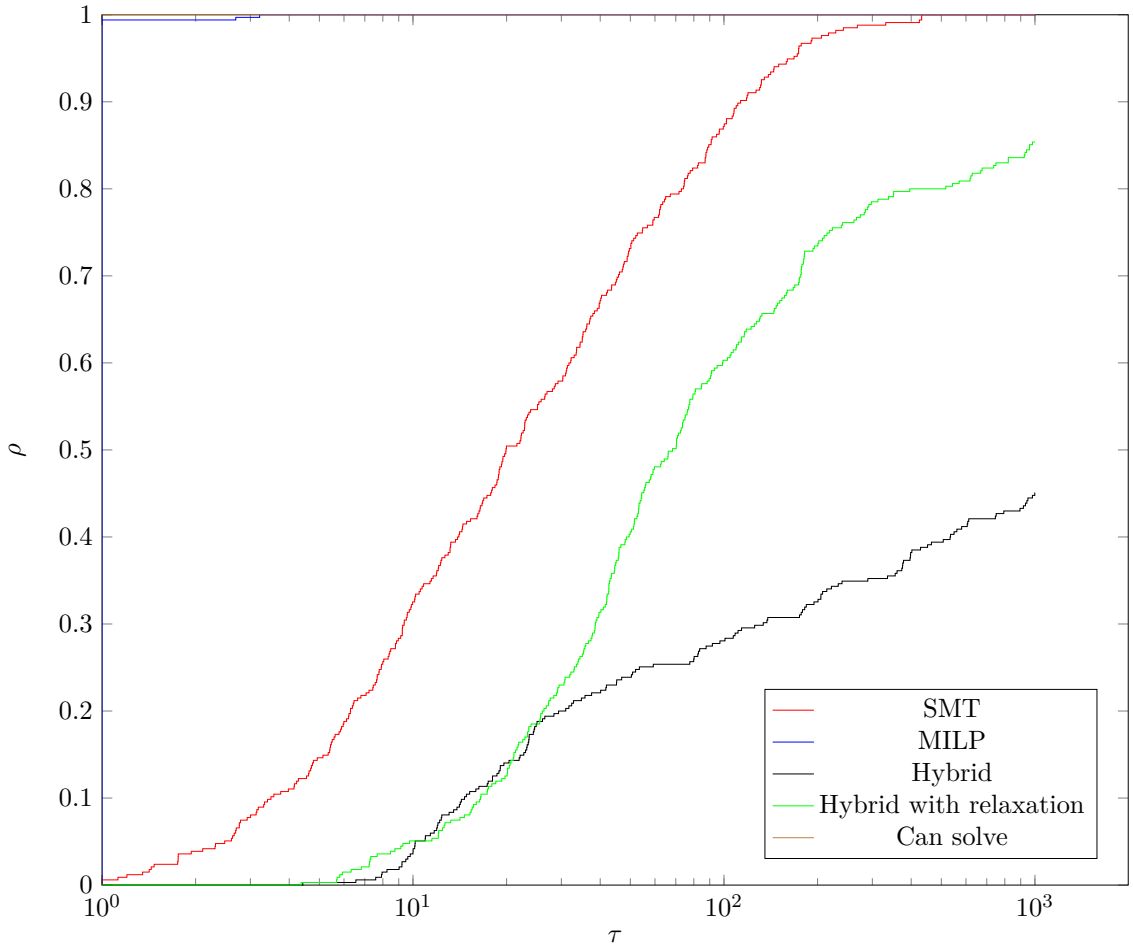


Figure: Solution time performance profile for all minimum tardiness problems.

References I

Bjørner, N. and De Moura, L. (2011). Satisfiability Modulo Theories: Introduction and Applications. *Communications of the ACM*, 54(9):69–77.

Bockmayr, A. and Kasper, T. (1998). Branch and Infer: A Unifying Framework for Integer and Finite Domain Constraint Programming. *INFORMS Journal on Computing*, 10(3):287–300.

Bockmayr, A. and Kasper, T. (2004). *Constraint and Integer Programming: Toward a Unified Methodology*, chapter Branch-and-Infer: A Framework for Combining CP and IP, pages 59–87. Springer US, Boston, MA.

Castro, P. M. and Grossmann, I. E. (2012). From time representation in scheduling to the solution of strip packing problems. *Computers & Chemical Engineering*, 44:45 – 57.

References II

Cook, W., Koch, T., Steffy, D. E., and Wolter, K. (2013). A hybrid branch-and-bound approach for exact rational mixed-integer programming. *Mathematical Programming Computation*, 5(3):305 – 344.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Math Program*, 91(2):201–213.

Garey, M. R., Graham, R. L., Johnson, D. S., and Yao, A. C.-C. (1976). Resource constrained scheduling as generalized bin packing. *J Combin Theory, Ser A*, 21(3):257 – 298.

Hooker, J. N. (2007). Planning and Scheduling by Logic-Based Benders Decomposition. *Operations Research*, 55(3):588–602.

Hooker, J. N. and Ottoson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming*, 96(1):33–60.

References III

Jain, V. and Grossmann, I. E. (2001). Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing*, 13(4):258–276.

Lodi, A., Martello, S., and Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241 – 252.

Lodi, A., Martello, S., and Vigo, D. (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357.

Maravelias, C. T. and Grossmann, I. E. (2004). A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers & Chemical Engineering*, 28(10):1921–1949.

References IV

Maravelias, C. T. and Sung, C. (2009). Integration of production planning and scheduling: Overview, challenges and opportunities. *Computers & Chemical Engineering*, 33(12):1919 – 1930.

Pisinger, D. and Sigurd, M. (2007). Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, 19(1):36–51.

Trespalacios, F. and Grossmann, I. E. (2016). Symmetry breaking for generalized disjunctive programming formulation of the strip packing problem. *Annals of Operations Research*, pages 1–13.