EFFICIENT SAT AND MAXSAT TECHNIQUES FOR SOLVING THE TWO-DIMENSIONAL STRIP PACKING PROBLEM

Tuyen Van Kieu[1*], Duong Quy Le[2] and Khanh Van To[3*]

*Corresponding author
[1]VNU University of Engineering and Technology, Hanoi, Vietnam / tuyenkv@vnu.edu.vn / https://orcid.org/0009-0007-7800-7172
[2]VNU University of Engineering and Technology, Hanoi, Vietnam / 21020560@vnu.edu.vn / https://orcid.org/0009-0005-9623-6716
[3]VNU University of Engineering and Technology, Hanoi, Vietnam / khanhtv@vnu.edu.vn / https://orcid.org/0009-0008-1907-7848

ABSTRACT. The NP-hard Two-Dimensional Strip Packing Problem (2SPP) demands efficient exact solutions. This paper presents SAT-based Order Encoding models incorporating item rotation and adapted symmetry-breaking (SB). It compares three height-minimization strategies: Non-Incremental SAT Bisection, Incremental SAT Bisection, and Direct MaxSAT Optimization. Benchmarks on established 2SPP instances show our Non-Incremental SAT and Direct MaxSAT strategies significantly outperform earlier incremental techniques. Direct MaxSAT excelled for non-rotational 2SPP, while Non-Incremental SAT was superior for rotational cases among our methods. Google CP-SAT, integrated into our Non-Incremental Bisection search framework, performed best overall. Nevertheless, our specialized SAT/MaxSAT methods were highly competitive, outperforming other CP and MIP solvers. SB was generally beneficial. Item rotation increased encoding complexity; while yielding improved optimal heights for some instances, it did not increase the total number of instances solved optimally by our SAT/MaxSAT methods. This work offers insights into SAT/MaxSAT strategy trade-offs and their performance against general solvers.

Keywords: Two-Dimensional Strip Packing; SAT Encoding; MaxSAT Optimization.

1 Introduction

The Two-Dimensional Strip Packing Problem (2SPP) addresses the optimal orthogonal placement of n rectangular items, $R = \{r_1, \ldots, r_n\}$, each with width w_i and height h_i, onto a strip of fixed width W and effectively unbounded height (Lodi et al. (2002); Dyckhoff

(1990)). The primary objective is to minimize the total height H required to accommodate all items without overlap, ensuring item edges are parallel to the strip's axes. In its standard variant, items maintain a fixed orientation (Lodi et al. (2002)), posing a core challenge in resource allocation under geometric constraints.

The 2SPP is driven by significant economic and logistical drivers across numerous sectors. Direct applications lie in material cutting industries, where minimizing utilized material length translates to reduced waste and cost savings (Gilmore and Gomory (1961); Wäscher et al. (2007)). Logistics and transportation utilize 2SPP models for efficient container or truck loading (Bischoff and Ratcliff (1995); Lodi et al. (1999)), while related concepts appear in VLSI floorplanning demanding optimal component placement (Sait and Youssef (1999); Lodi et al. (2002)). The common goal is optimizing the use of a constrained two-dimensional resource.

Despite its clear definition, obtaining provably optimal 2SPP solutions presents significant computational challenges. The problem is strongly NP-hard (Garey and Johnson (1979); Lodi et al. (2002)), linked to other difficult packing problems like Bin Packing (Martello et al. (2003)). This complexity implies that finding the minimum height H_{opt} likely requires exponential resources relative to the number of items n, stemming from the vast combinatorial search space and intricate geometric constraints. Consequently, traditional exact methods like branch-and-bound (Martello et al. (2003); Boschetti and Montaletti (2010)) or integer programming (Lodi et al. (2002)) often face scalability limitations, struggling with industrially relevant instance sizes (Martello et al. (2003)). While heuristics offer fast approximations (Lodi et al. (2015); Burke et al. (2004)), the difficulty in efficiently guaranteeing optimality motivates exploring alternative exact paradigms, such as the SAT-based methods investigated herein.

Addressing these challenges, this paper makes several key contributions to the exact solution of the 2SPP using SAT and MaxSAT techniques. First, we develop and evaluate SAT-based models using Order Encoding, critically extending these to incorporate item rotation and proposing adapted symmetry-breaking constraints suitable for the rotational variant. Second, we conduct a rigorous comparative study of three distinct height-minimization strategies: (1) Non-Incremental SAT Bisection Search, which involves repeatedly solving SAT instances without reusing learned clauses across different height checks; (2) Incremental SAT Bisection Search, utilizing Glucose 4.2 (Audemard and Simon (2009, 2012)) to leverage learned information between bisection steps; and (3) Direct Weighted Partial MaxSAT Optimization, employing the state-of-the-art solver TT-Open-WBO-Inc (Nadel (2024)) to find the optimal height in a single optimization call. Third, we perform extensive benchmarks of our proposed SAT/MaxSAT configurations against leading commercial and open-source optimization solvers, including CPLEX, Gurobi, and Google OR-Tools. For Constraint Programming (CP) solvers, we integrated them into our Non-Incremental Bisection search framework to solve se-

quences of decision problems, while Mixed-Integer Programming (MIP) models were used for direct optimization.

Our experimental evaluation substantiates these contributions, demonstrating the practical advantages of the proposed Non-Incremental SAT Bisection and Direct MaxSAT optimization strategies over earlier incremental techniques. The findings confirm the utility of the adapted symmetry breaking constraints, particularly in the rotational context. Furthermore, the benchmarking provides a clear perspective on the competitiveness of these specialized SAT/MaxSAT methods against state-of-the-art general-purpose CP (used with Non-Incremental Bisection) and MIP solvers for the 2SPP, highlighting scenarios where logical approaches excel. The impact of item rotation on solution quality versus computational cost is also empirically assessed. Overall, this research offers valuable insights into effective SAT/MaxSAT-based solution design for 2SPP.

The remainder of this paper is organized as follows. Section 2 reviews relevant prior work on 2SPP solution methods and SAT/MaxSAT techniques. Section 3 details our SAT-based methodology for the rotational 2SPP, including the core encoding and symmetry breaking adaptations. Section 4 describes the different height minimization strategies investigated. Section 5 presents the experimental setup, detailed results, and comparative analysis. Finally, Section 6 concludes the paper and suggests directions for future research.

2 Related Work

The 2SPP has attracted extensive research, leading to a variety of heuristic and exact solution methods. Heuristic approaches, aiming for rapid, high-quality solutions without optimality guarantees, include foundational work by Lodi et al. (2015) and Burke et al. (2004), with the latter also advancing guillotine-constrained strip packing. Recent developments feature the Block-Based Heuristic Search algorithm (BBHSA) by Zhang et al. (2024) for guillotine-constrained 2SPP and a Backtracking Heuristic algorithm (BHA) for its rotational, non-guillotine counterpart (Yang et al. (2025)). Deterministic heuristics have also shown promise for rotational 2SPP (Alvarez-Valdés et al. (2008)), and Hopper and Turton (2001) provided a key review of meta-heuristics for 2D strip packing.

Exact methods, while guaranteeing optimality, often incur higher computational costs. Traditional techniques include branch-and-bound, with seminal work by Martello et al. (2003) later refined by Boschetti and Montaletti (2010). Integer Programming (IP) formulations, introduced by Lodi et al. (2002), have been extended to robust models for uncertain 2SPP variants (Liu et al. (2023)). Constraint Programming (CP) has also proven effective, with various implementations exploring diverse constraint models and search

strategies (Berger et al. (2009)).

Boolean Satisfiability (SAT) offers a powerful paradigm for exact combinatorial problem-solving (Biere et al. (2009)), where Order Encoding (Tamura et al. (2006)) is particularly adept at representing geometric packing constraints. Soh et al. (2010) pioneered Order Encoding for fixed-orientation 2SPP, establishing a foundational feasibility model $\Phi(H)$ and introducing basic symmetry breaking. Maximum Satisfiability (MaxSAT), an extension of SAT, seeks to satisfy all hard clauses while maximizing the weight of satisfied soft clauses (Li and Manyà (2009); Ansótegui et al. (2013)), making it suitable for optimization tasks like height minimization in 2SPP. Modern MaxSAT solvers like TT-Open-WBO-Inc (Nadel (2024)) facilitate this, and our work explores its direct application.

Optimization via SAT often employs bisection search on the objective (e.g., height H), iteratively solving SAT decision problems. Incremental SAT techniques (Eén and Sörensson (2003); Audemard and Simon (2009, 2012)) aim to accelerate this by reusing learned clauses, a concept Soh et al. (2010) discussed for 2SPP. This paper offers a contemporary evaluation of incremental SAT against non-incremental and direct MaxSAT methods. Symmetry breaking is vital for pruning the search space (Walsh (2006); Gent and Smith (2000)). While Soh et al. (2010) applied it to fixed-orientation 2SPP, item rotation, a key aspect of many 2SPP variants, necessitates adapted or new techniques, which this work investigates.

The rotational 2SPP, allowing 90-degree item turns, increases problem complexity but can yield better packings. Kenmochi et al. (2009) and Arahori et al. (2012) developed exact algorithms for rotational 2SPP, with the latter using canonical forms. The impact of rotation was further studied by Wei et al. (2011) and Leung et al. (2011) (for guillotine cuts). Heuristics for rotational 2SPP include fast methods by Ha et al. (2010), reactive GRASP by Alvarez-Valdés et al. (2008), and recent backtracking heuristics by Yang et al. (2025). Effective rotation handling involves balancing solution quality and computational cost (Jylänki (2010)). Our research contributes by developing and evaluating SAT/MaxSAT encodings that explicitly manage item rotation.

In essence, while traditional exact methods often struggle with large 2SPP instances, SAT/MaxSAT techniques present viable alternatives. This paper explores advancements in specialized encodings, symmetry breaking, and modern solver utilization to tackle this challenging problem.

3 SAT-based methodology for rotational 2SPP

This section presents the detailed SAT-based methodology constructed and evaluated in this research for tackling the rotational variant of the 2SPP. While the foundational prin-

ciples rely on the well-established Order Encoding paradigm, as applied to the fixed-orientation 2SPP by Soh et al. (2010), significant extensions and adaptations are introduced here to model rectangle rotation and integrate appropriate symmetry-breaking mechanisms. The objective is to systematically develop a robust and efficient SAT formulation, $\Phi_R(H)$, capable of accurately representing the rotational Two-Dimensional Orthogonal Packing Problem (2OPP) feasibility problem.

3.1 Core variables using Order Encoding

The SAT model maps the 2SPP's geometric and decision elements onto Boolean variables. We utilize the established Order Encoding (Tamura et al. (2006)) for positional information and introduce a variable for orientation.

- *Position variables:* For each rectangle r_i and integer thresholds e ($0 \leq e < W$) and f ($0 \leq f < H$), Boolean variables $p_{x_{i,e}}$ (representing $x_i \leq e$) and $p_{y_{i,f}}$ (representing $y_i \leq f$) encode the bottom-left coordinates (x_i, y_i). This encoding facilitates a compact representation of comparison constraints (Tamura et al. (2006)).
- *Relative position variables:* Consistent with (Soh et al. (2010)), auxiliary Boolean variables capture pairwise Non-overlap relations between distinct rectangles r_i and r_j: $l_{i,j}$ (true if r_i is left of r_j) and $u_{i,j}$ (true if r_i is below r_j). The exact geometric meaning is conditioned on rotation, detailed in constraint formulations (Section 3.2).
- *Rotation variable:* To model the rotational variant, we introduce R_i for each rectangle r_i. R_i is true if r_i is rotated 90 degrees from its input dimensions, and false otherwise. This explicit variable is key to formulating rotation-dependent constraints.

These variable sets provide the foundation for constructing the SAT formula representing the rotational 2OPP feasibility problem.

3.2 Core constraints for rotational placement

Building upon the variable definitions established in Section 3.1, particularly the Order Encoding variables ($p_{x_{i,e}}$, $p_{y_{i,f}}$), relative position variables ($l_{i,j}$, $u_{i,j}$), and the crucial rotation variable (R_i), this section details the formulation of the core constraints for the rotational 2OPP feasibility problem, denoted $\Phi_R(H)$. These constraints ensure the logical integrity of the coordinate encoding, enforce containment within the strip Boundaries considering potential rotation, and model the fundamental non-overlap requirement between rectangles.

First, the semantic integrity of the Order Encoding is maintained through Axioms clauses enforcing monotonicity. As established in foundational works (Tamura et al. (2006)), these clauses ensure that if $x_i \leq e$, then $x_i \leq e+1$ (and analogously for y_i). These constraints are standard for Order Encoding (Soh et al. (2010)) and are independent of rectangle dimensions or rotation status. For all r_i, $0 \leq e < W-1$, and $0 \leq f < H-1$:

$$\begin{aligned} (\neg p_{x_{i,e}} \vee p_{x_{i,e+1}}) \\ (\neg p_{y_{i,f}} \vee p_{y_{i,f+1}}) \end{aligned} \tag{1}$$

These clauses provide the essential logical structure guaranteeing that variable assignments correspond coherently to integer coordinates.

A significant adaptation for the rotational problem involves the Boundary constraints. Each rectangle r_i must reside entirely within the strip dimensions ($W \times H$), but its effective width and height depend on the orientation specified by R_i. The horizontal Boundary constraint, $x_i + \text{effective\_width} \leq W$, translates to $x_i \leq W - w_i$ if $\neg R_i$ (original orientation) and $x_i \leq W - h_i$ if R_i (rotated). This conditional logic is captured using implications involving the Order Encoding variables:

$$\begin{aligned} (\neg R_i \implies p_{x_{i,W-w_i}}) \equiv (R_i \vee p_{x_{i,W-w_i}}) \\ (R_i \implies p_{x_{i,W-h_i}}) \equiv (\neg R_i \vee p_{x_{i,W-h_i}}) \end{aligned} \tag{2}$$

Similarly, the vertical Boundary constraint, $y_i + \text{effective\_height} \leq H$, depends on R_i determining whether the effective height is h_i or w_i. This leads to the clauses:

$$\begin{aligned} (\neg R_i \implies p_{y_{i,H-h_i}}) \equiv (R_i \vee p_{y_{i,H-h_i}}) \\ (R_i \implies p_{y_{i,H-w_i}}) \equiv (\neg R_i \vee p_{y_{i,H-w_i}}) \end{aligned} \tag{3}$$

These rotation-aware Boundary constraints are a critical extension compared to fixed-orientation models, ensuring placements are valid regardless of the chosen orientation.

The Non-overlap constraint between any distinct pair of rectangles r_i and r_j ($i < j$) is fundamentally enforced by the disjunction ensuring that one must be to the left of, to the right of, below, or above the other (Soh et al. (2010)):

$$(l_{i,j} \vee l_{j,i} \vee u_{i,j} \vee u_{j,i}) \tag{4}$$

The primary challenge lies in correctly linking the relative position variables ($l_{i,j}, l_{j,i}, u_{i,j}$, $u_{j,i}$) to the Order Encoding position variables ($p_{x_{i,e}}, p_{y_{i,f}}$) while accounting for the effective dimensions (w'_i, h'_i, w'_j, h'_j) which are determined by the rotation variables R_i and R_j. For the 'left-of' relations, $l_{i,j}$ implies $x_i + w'_i \leq x_j$, where w'_i is w_i if $\neg R_i$ and h_i if R_i. This implication is encoded conditionally on R_i:

$$\begin{aligned} (R_i \vee \neg l_{i,j} \vee \neg p_{x_{j,e+w_i}} \vee p_{x_{i,e}}), \quad e \in \{0, \ldots, W-w_i-1\} \\ (\neg R_i \vee \neg l_{i,j} \vee \neg p_{x_{j,e+h_i}} \vee p_{x_{i,e}}), \quad e \in \{0, \ldots, W-h_i-1\} \end{aligned} \tag{5}$$

Symmetrically, $l_{j,i}$ implies $x_j + w'_j \leq x_i$, encoded conditionally on R_j:

$$\begin{aligned} (R_j \vee \neg l_{j,i} \vee \neg p_{x_{i,e+w_j}} \vee p_{x_{j,e}}), \quad e \quad \in \{0,\ldots,W-w_j-1\} \\ (\neg R_j \vee \neg l_{j,i} \vee \neg p_{x_{i,e+h_j}} \vee p_{x_{j,e}}), \quad e \quad \in \{0,\ldots,W-h_j-1\} \end{aligned} \tag{6}$$

Analogous reasoning applies to the 'below' relations. The implication $u_{i,j} \implies y_i + h'_i \leq y_j$, where h'_i depends on R_i, yields:

$$\begin{aligned} (R_i \vee \neg u_{i,j} \vee \neg p_{y_{j,f+h_i}} \vee p_{y_{i,f}}), \quad f \quad \in \{0,\ldots,H-h_i-1\} \\ (\neg R_i \vee \neg u_{i,j} \vee \neg p_{y_{j,f+w_i}} \vee p_{y_{i,f}}), \quad f \quad \in \{0,\ldots,H-w_i-1\} \end{aligned} \tag{7}$$

And $u_{j,i} \implies y_j + h'_j \leq y_i$, conditional on R_j, gives:

$$\begin{aligned} (R_j \vee \neg u_{j,i} \vee \neg p_{y_{i,f+h_j}} \vee p_{y_{j,f}}), \quad f \quad \in \{0,\ldots,H-h_j-1\} \\ (\neg R_j \vee \neg u_{j,i} \vee \neg p_{y_{i,f+w_j}} \vee p_{y_{j,f}}), \quad f \quad \in \{0,\ldots,H-w_j-1\} \end{aligned} \tag{8}$$

These sets of clauses accurately translate the geometric Non-overlap conditions into propositional logic, explicitly incorporating the effect of rotation on rectangle dimensions.

Optionally, to potentially enhance solver performance through stronger propagation, short-circuit clauses can be included. These clauses enforce basic coordinate bounds implied by relative placements, conditioned on rotation, even for small coordinate thresholds. They capture implications like $l_{i,j} \implies x_j \geq w'_i$ and $u_{i,j} \implies y_j \geq h'_i$, and their symmetric counterparts:

$$\begin{aligned} (R_i \vee \neg l_{i,j} \vee \neg p_{x_{j,e}}), \quad e \quad \in \{0,\ldots,w_i-1\} \\ (\neg R_i \vee \neg l_{i,j} \vee \neg p_{x_{j,e}}), \quad e \quad \in \{0,\ldots,h_i-1\} \end{aligned} \tag{9}$$

$$\begin{aligned} (R_i \vee \neg u_{i,j} \vee \neg p_{y_{j,f}}), \quad f \quad \in \{0,\ldots,h_i-1\} \\ (\neg R_i \vee \neg u_{i,j} \vee \neg p_{y_{j,f}}), \quad f \quad \in \{0,\ldots,w_i-1\} \end{aligned} \tag{10}$$

$$\begin{aligned} (R_j \vee \neg l_{j,i} \vee \neg p_{x_{i,e}}), \quad e \quad \in \{0,\ldots,w_j-1\} \\ (\neg R_j \vee \neg l_{j,i} \vee \neg p_{x_{i,e}}), \quad e \quad \in \{0,\ldots,h_j-1\} \end{aligned} \tag{11}$$

$$\begin{aligned} (R_j \vee \neg u_{j,i} \vee \neg p_{y_{i,f}}), \quad f \quad \in \{0,\ldots,h_j-1\} \\ (\neg R_j \vee \neg u_{j,i} \vee \neg p_{y_{i,f}}), \quad f \quad \in \{0,\ldots,w_j-1\} \end{aligned} \tag{12}$$

While logically implied by the main Non-overlap and Axioms clauses, these redundant constraints can sometimes help prune the search space more effectively. Collectively, the constraints presented in this section form the core logical representation of the rotational 2OPP within the SAT framework.

3.3 Adapted symmetry breaking constraints for rotation

Symmetry breaking (SB) is crucial for SAT solver efficiency, particularly as rotation introduces additional solution equivalences. While building upon established SB principles (e.g., Soh et al. (2010) for fixed-orientation 2SPP), the following constraints are specifically adapted to integrate with our rotational Order Encoding model, particularly interacting with the rotation variable R_i (Section 3.1) to prune redundant search paths introduced by item orientation choices.

Square rectangles

A common SB technique, this constraint prevents exploring the rotation of a square item ($w_i = h_i$), as it yields an identical geometric configuration. For our model, this translates to directly fixing its orientation:

$$(w_i = h_i) \implies \neg R_i \tag{13}$$

If $w_i = h_i$ (an input property), the unit clause $\neg R_i$ is added for rectangle i, effectively removing R_i from consideration by the solver and eliminating a redundant branch in the search space.

Out-of-bounds rotation

This pre-processing constraint leverages the explicit rotation variable R_i to fix item orientation if only one orientation is geometrically viable within the strip boundaries (W, H). If the original orientation is invalid (e.g., $w_i > W$) but the rotated one is valid (e.g., $h_i \leq W \wedge w_i \leq H$), rotation becomes mandatory:

$$((w_i > W \vee h_i > H) \wedge (h_i \leq W \wedge w_i \leq H)) \implies R_i \tag{14}$$

Conversely, if the original fits but the rotated does not, rotation is forbidden:

$$((w_i \leq W \wedge h_i \leq H) \wedge (h_i > W \vee w_i > H)) \implies \neg R_i \tag{15}$$

These rules, evaluated based on input parameters, add unit clauses (R_i or $\neg R_i$) that simplify the problem by pre-determining orientations, directly reducing the solver's search task. Figure 1 illustrates a case forcing rotation.

Large rectangles (LR) adaptation

This constraint, adapted from (Soh et al. (2010)), prevents adjacent placements of r_i, r_j if their combined effective dimensions (dependent on R_i, R_j) exceed strip boundaries:

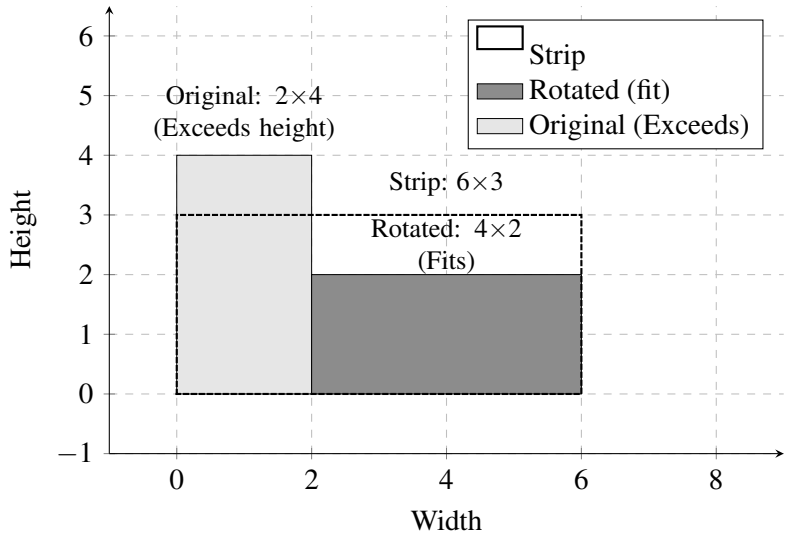

Figure 1: Illustration of out-of-bounds rotation: Original 2x4 exceeds $H = 3$; rotated 4x2 fits in 4x2 strip, forcing R_i.

- *Horizontal:* If effective_width(i) + effective_width$(j) > W$, then $\neg l_{i,j} \wedge \neg l_{j,i}$.
- *Vertical:* If effective_height(i) + effective_height$(j) > H$, then $\neg u_{i,j} \wedge \neg u_{j,i}$.

Our LR adaptation avoids complex conditional SB clauses by directly modifying the generation of the core Non-overlap disjunction (Equation 4) and its linking clauses (Section 3.2). If rotation states for r_i, r_j cause their combined dimensions to exceed strip limits (e.g., $w_i + w_j > W$), impossible relative placement literals (e.g., $l_{i,j}, l_{j,i}$) are omitted from the disjunction under that specific rotational context. This prunes unsatisfiable branches, streamlining the SAT formula by reducing complex clauses, potentially improving solver efficiency.

While extending known principles, these SB constraints are specifically adapted for our Order Encoding with explicit rotation variables (R_i). The *Out-of-bounds rotation* and the *LR adaptation*—modifying clause generation instead of adding explicit constraints—are notable implementation choices leveraging the encoding's structure to simplify problem representation. SB for Order Encoded rotational 2SPP is less explored; our adaptations offer practical rules for rotation-induced symmetries within this paradigm, vital for managing the expanded search space and enhancing efficiency (Section 5).

3.4 The complete SAT model $\Phi_R(H)$ for rotational 2OPP

The complete SAT instance $\Phi_R(H)$, encoding the feasibility of the rotational 2OPP for a given height H, is generated by systematically combining the variables and constraints detailed above.

The conjunction of all these clause sets—Axioms, Rotational Boundaries, Non-Overlap disjunctions, Conditional Non-Overlap Links, and Adapted Symmetry breaking constitutes the complete SAT instance $\Phi_R(H)$. A satisfying assignment for $\Phi_R(H)$ directly corresponds to a valid rotational 2OPP solution for height H. This formula is therefore the precise logical query submitted to the SAT or MaxSAT solver within the height minimization framework discussed in the subsequent section.

4 Height minimization strategies for 2SPP

Having established the SAT methodology for checking the feasibility of the rotational 2OPP for a given height H ($\Phi_R(H)$) in Section 3, we now explore distinct high-level strategies for utilizing this feasibility check to solve the 2SPP optimization problem – finding the minimum required height H_{opt}.

4.1 Bisection search framework with SAT feasibility checks

To find the minimum height H_{opt}, a primary strategy employs the SAT-based feasibility check $\Phi_R(H)$ (Section 3) within a Bisection Search. This iterative approach solves the 2OPP decision problem for different candidate heights.

4.1.1 Bisection search framework

Bisection search (Algorithm 1) finds the minimum height H_{opt} in a range $[H_{lb}, H_{ub}]$ where $\Phi_R(H)$ is satisfiable. Leveraging the monotonicity of packing feasibility (feasibility at H implies feasibility at $H' > H$), it iteratively tests H_{mid} using a CheckFeasibility() function, which solves the SAT instance $\Phi_R(H_{mid})$. The search interval $[low, high]$ is adjusted based on the SAT/UNSAT result until $low > high$. Effective initial bounds H_{lb}, H_{ub} are vital for efficiency.

4.1.2 Non-incremental SAT implementation

In the non-incremental SAT implementation of the CheckFeasibility() function (Algorithm 2), each call for a height H_{check} generates $\Phi_R(H_{check})$ anew and uses a fresh SAT

Algorithm 1 Bisection search framework for minimum height

Require: Strip width W, set of items R, initial lower bound H_{lb}, initial upper bound H_{ub}
Ensure: Optimal height H_{opt} or indication of failure

$low \leftarrow H_{lb}$
$high \leftarrow H_{ub}$
$H_{opt} \leftarrow high + 1$
while $low \leq high$ **do**
  $H_{mid} \leftarrow \lfloor (low + high)/2 \rfloor$
  $result \leftarrow CheckFeasibility(W, H_{mid}, R)$ {Core SAT/MaxSAT call}
  if $result =$ SAT **then**
    $H_{opt} \leftarrow H_{mid}$
    $high \leftarrow H_{mid} - 1$
  else
    $low \leftarrow H_{mid} + 1$
  end if
end while
if $H_{opt} > H_{ub}$ **then**
  return Failure
else
  return H_{opt}
end if

solver instance. This treats checks independently, without reusing learned information. The height H_{check} directly defines the y-coordinate scope and boundaries within $\Phi_R(H_{check})$; no internal height indicator variables (e.g., p_h) are used within this formula for bisection. Height is managed externally by the loop. This serves as a fundamental baseline.

4.1.3 Incremental SAT solving

Incremental SAT solving (adapting Soh et al. (2010)) uses a persistent Glucose 4.2 solver to reuse learned clauses. The solver is initialized with a formula for the full height range $[H_{lb}, H_{ub}]$, including core packing constraints (Section 3 extended to H_{ub}) and height indicator variables p_h for $h \in [H_{lb}, H_{ub}]$. Clauses for p_h include:

- *Height linking clauses:* For each rectangle r_i and each $h \in [H_{lb}, H_{ub}]$ (where h is large enough to accommodate r_i):
  - $(\neg p_h \vee R_i \vee p_{y_{i,h-h_i}})$: If p_h is true and r_i is not rotated, $y_i \leq h - h_i$.
  - $(\neg p_h \vee \neg R_i \vee p_{y_{i,h-w_i}})$: If p_h is true and r_i is rotated, $y_i \leq h - w_i$.

Algorithm 2 CheckFeasibility() using Non-Incremental SAT solving

Require: Strip width W, height to check H_{check}, set of items R
Ensure: SAT or UNSAT result
1: $\Phi_R(H_{check}) \leftarrow GenerateSATFormula(W, H_{check}, R)$
2: $solver \leftarrow InitializeSATSolver()$
3: $AddClausesToSolver(solver, \Phi_R(H_{check}))$
4: $is\_sat \leftarrow solver.solve()$
5: $DeleteSolverInstance(solver)$
6: **if** *is_sat* **then**
7: **return** SAT
8: **else**
9: **return** UNSAT
10: **end if**

- *Height monotonicity:* For each $h \in [H_{lb}, H_{ub} - 1]$: $(\neg p_h \vee p_{h+1})$.

The CheckFeasibility() function (Algorithm 3) then assumes $p_{H_{check}}$ to test height H_{check}. However, a loose initial upper bound H_{ub} significantly enlarges the initial formula. This overhead can counteract incrementality benefits, potentially leading to performance inferior to the non-incremental method, as discussed in Section 5 and Section 4.3. This trade-off is a critical consideration.

Algorithm 3 CheckFeasibility() using Incremental SAT solving

Require: Global persistent SAT solver instance *solver* (initialized for range up to H_{ub})
Require: Mapping from height h to assumption variable p_h
Require: Height to check H_{check}
Ensure: SAT or UNSAT result
1: $assumption\_literal \leftarrow GetVariableForHeight(H_{check})$ {e.g., $p_{H_{check}}$}
2: $is\_sat \leftarrow solver.solve(assumptions = [assumption\_literal])$
3: **if** *is_sat* **then**
4: **return** SAT
5: **else**
6: **return** UNSAT
7: **end if**

While incremental solving aims to leverage learned clauses from previous (often UNSAT) calls, its effectiveness is sensitive to the initial upper bound H_{ub}. If H_{ub} is significantly larger than the true optimal height H_{opt}, the initial formula loaded into the persistent solver becomes substantially larger, increasing the number of variables and constraints related to height. This overhead, stemming from a potentially loose H_{ub}, can

counteract the benefits of incrementality, potentially leading to performance inferior to the non-incremental method in practice, as observed in preliminary experiments. The trade-off between the potential speedup from clause reuse and the overhead from a large initial formula is a critical consideration for this approach.

4.2 Direct optimization via weighted partial MaxSAT

Alternatively, the 2SPP height minimization can be directly addressed by encoding it as a single Weighted Partial MaxSAT (WCNF) instance (Li and Manyà (2009); Ansótegui et al. (2013)). This method uses hard clauses for mandatory packing constraints and soft clauses to penalize height, enabling a MaxSAT solver (Morgado et al. (2013); Nadel (2024)) to find the minimum feasible height H_{opt} in one call.

4.2.1 WCNF formulation

A single Weighted Conjunctive Normal Form (WCNF) formula is constructed, comprising hard and soft clauses, defined over variables encompassing the packing configuration and potential heights up to an initial upper bound H_{ub}.

Hard clauses (∞): These enforce the fundamental requirements for a valid packing solution up to H_{ub}. They include:

- Core rotational 2OPP constraints (Section 3) ensuring geometric validity.
- Height feasibility linking clauses, connecting the packing state to height indicator variables p_h:

$$\begin{aligned} (\neg p_h \vee R_i \vee p_{y_i,h-h_i}), \quad h \quad &\in \{h_i,\ldots,H_{ub}\} \\ (\neg p_h \vee \neg R_i \vee p_{y_i,h-w_i}), \quad h \quad &\in \{w_i,\ldots,H_{ub}\} \end{aligned} \tag{16}$$

- Height variable ordering clauses, ensuring monotonicity ($p_h \implies p_{h+1}$):

$$(\neg p_h \vee p_{h+1}), \quad h \in \{H_{lb},\ldots,H_{ub}-1\} \tag{17}$$

- Basic feasibility clause, ensuring at least one height within $[H_{lb},H_{ub}]$ is achievable:

$$(p_{H_{lb}} \vee \ldots \vee p_{H_{ub}}) \tag{18}$$

Soft clauses (Weight 1): These encode the optimization objective. For each potential height $h \in [H_{lb},H_{ub}]$, a unit soft clause p_h is added with weight 1:

$$(1,[p_h]), \quad h \in \{H_{lb},\ldots,H_{ub}\} \tag{19}$$

The MaxSAT solver aims to satisfy all hard clauses while minimizing the sum of weights of satisfied soft p_h clauses. Combined with the height ordering clauses, this directly yields the minimum H_{opt} for which $p_{H_{opt}}$ is true.

4.2.2 Solving with TT-Open-WBO-Inc

The resulting WCNF formula is solved once using a Weighted Partial MaxSAT solver, such as TT-Open-WBO-Inc (Nadel (2024)), as outlined in Algorithm 4.

Algorithm 4 Direct optimization using MaxSAT

Require: Strip width W, Set of rectangles R, Lower bound H_{lb}, Upper bound H_{ub}
Ensure: Optimal height H_{opt} and optional model M_{opt}, or indication of failure

$\mathscr{H} \leftarrow \{h \mid H_{lb} \leq h \leq H_{ub}\}$
$\mathscr{V} \leftarrow DefineAllVariables(W, R, \mathscr{H})$ {Includes all $p_{x..}, p_{y..}, R_i, l_{ij}, u_{ij}, p_h$}
$HardClauses \leftarrow EncodePackingConstraints(\mathscr{V}, W, H_{ub}, R)$ {Core constraints up to H_{ub}}
$HardClauses \leftarrow HardClauses \cup EncodeHeightLogic(\mathscr{V}, \mathscr{H})$ {Height linking and ordering}
$SoftClauses \leftarrow \{(1, [Var(p_h)]) \mid h \in \mathscr{H}\}$ {Penalize achievable heights}
$WCNF \leftarrow Combine(HardClauses, SoftClauses)$
$status, M_{opt} \leftarrow SolveMaxSAT(\text{"tt-open-wbo-inc"}, WCNF)$ {Single solver call}
if $status =$ OPTIMUM FOUND **then**
    $H_{opt} \leftarrow \min\{h \in \mathscr{H} \mid IsVarTrue(Var(p_h), M_{opt})\}$ {Extract min height}
    return H_{opt}, M_{opt}
else
    return Failure, None
end if

Unlike iterative bisection, this direct MaxSAT approach delegates the entire optimization to the solver. However, its performance, akin to incremental SAT, is sensitive to the initial upper bound H_{ub} which dictates the WCNF instance size (Section 5).

4.3 Impact of upper bound estimation

Both the incremental SAT approach within Bisection Search (Section 4.1.3) and the direct MaxSAT method (Section 4.2) require an initial upper bound H_{ub} to define the scope of the SAT/MaxSAT formula. The quality of this bound significantly impacts performance, as it dictates the size of the formula handled by the solver.

A loose upper bound (i.e., $H_{ub} \gg H_{opt}$) inflates the initial SAT/MaxSAT formula. This occurs by expanding the range of y-coordinate Order Encoding variables ($p_{y_{i,f}}$) and consequently the number of related Axioms, Boundary, and Non-overlap clauses. Furthermore, it increases the count of height indicator variables (p_h) spanning $[H_{lb}, H_{ub}]$ and the associated height linking and ordering clauses. Such formula growth leads to higher memory use and slower solver speeds, potentially counteracting the advantages of incre-

mental or direct optimization techniques.

To mitigate this, we employ a heuristic upper bound calculation (Algorithm 5) based on the First Fit Decreasing Height (FFDH) strategy (Coffman et al. (1980)), combined with the trivial sum-of-heights bound.

Algorithm 5 Heuristic upper bound calculation (H_{ub})

Require: Strip width W, Set of rectangles $R = \{(w_1, h_1), \ldots, (w_n, h_n)\}$
Ensure: Heuristic upper bound H_{ub}
1: $R_{sorted} \leftarrow SortRectanglesByHeightDescending(R)$
2: $H_{FFDH} \leftarrow CalculateFFDHHeight(W, R_{sorted})$ {Standard FFDH level packing}
3: $H_{sum} \leftarrow \sum_{i=1}^{n} h_i$
4: $H_{ub} \leftarrow \min(H_{FFDH}, H_{sum})$
5: **return** H_{ub}

Despite using this heuristic, empirical observations (detailed in Section 5) indicate that the calculated H_{ub} can still exhibit a substantial gap compared to the optimal height H_{opt}. This persistent gap contributes to the observed phenomenon where the overhead introduced by the large formula size in incremental SAT and MaxSAT methods can sometimes make the simpler, non-incremental Bisection Search approach (Section 4.1.2) more competitive in terms of overall runtime. Finding tighter, efficiently computable upper bounds remains an important factor for maximizing the performance of these advanced SAT/MaxSAT-based optimization techniques. The quantitative extent of this gap and its impact on the performance of incremental SAT and direct MaxSAT approaches are further detailed with specific data in Section 5.

5 Experimental evaluation and Results

This section presents a comprehensive empirical evaluation of the methodologies developed and discussed in Sections 3 and 4. We aim to assess their practical performance, compare different optimization strategies, evaluate the impact of item rotation and symmetry breaking, and benchmark these specialized approaches against established general-purpose optimization solvers. The evaluation is performed against optimal heights established in the literature, specifically considering that non-rotational and rotational problem variants may have different optimal values for the same base instance (Kenmochi et al. (2009); Arahori et al. (2012)). To facilitate transparency and reproducibility, the benchmark instances and implementations are made available at `https://github.com/strippacking/SPP`.

5.1 Experimental setup

A consistent experimental setup is crucial for rigorous evaluation. This section details the benchmark instances, computational environment, solvers, and performance metrics.

5.1.1 Benchmark instances

We evaluated our approaches on a standard set of 41 2SPP benchmark instances, aggregating well-known datasets from various sources, including *HT* instances from Hopper and Turton (2001), *BENG* instances from Bengtsson (1982), *CGCUT* instances from Christofides and Whitlock (1977), *GCUT* instances from Beasley (1985a), and *NGCUT* instances from Beasley (1985b), obtained via the University of Bologna's OR Group library. These instances vary widely in size (from 7 to 200 rectangles) and item dimensions, providing a robust testbed. Known optimal heights, primarily from Kenmochi et al. (2009); Ha et al. (2010); Arahori et al. (2012), served as targets for solution quality, distinguishing between non-rotational and rotational optima where applicable.

5.1.2 Computational environment and solvers

Experiments ran on Google Cloud Platform (GCP) using e2-highcpu-16 instances (16 vCPUs, 16GB RAM, 20GB disk) with Ubuntu 20.04 LTS. A 1800-second (30-minute) time limit per instance/solver execution included model generation and solving. For SAT-based bisection, Glucose 4.2 by Audemard and Simon (2009) was employed, and for Direct MaxSAT optimization, TT-Open-WBO-Inc by Nadel (2024) was used. These were benchmarked against several established solvers: CPLEX 22.1.1 by IBM Corporation (2023) (providing both CP Optimizer and MIP solver capabilities), MIP solver by Gurobi Optimization, LLC (2023) (9.5.0), and OR-Tools 9.1 by Google Inc. (2023) (offering both CP-SAT and MIP solvers). Solvers used default parameters unless stated. Implementations were generated by Python 3 scripts in the same environment.

5.1.3 Performance metrics

Performance was assessed by several key indicators. Primarily, we measured the number of instances solved to confirmed optimality (# Opt) by comparing achieved heights against the best-known values reported in Kenmochi et al. (2009); Alvarez-Valdes et al. (2009); Neveu and Trombettoni (2008); Arahori et al. (2012), all within a strict time limit of 1800 seconds per instance; this comparison acknowledged that optimal heights can differ for rotational and non-rotational problem variants. Secondly, we recorded the total wall-clock runtime in seconds ($\sum \tau$) for those instances solved optimally, ensuring

this included time for both model generation and solving. Finally, for the SAT/MaxSAT methods specifically, we tracked the complexity of the generated formulas by noting the total number of variables (#Vars) and clauses (#Cls). For Non-Incremental SAT Bisection approaches, the reported #Vars and #Cls correspond to the SAT instance generated for the final, optimal height H_{opt} found. In contrast, for Incremental SAT Bisection and Direct MaxSAT methods, these counts represent the single, initial formula that encodes the entire problem up to the heuristic upper bound H_{ub}. These metrics allow a thorough evaluation of effectiveness, efficiency, and scalability. Detailed instance-wise results, including comparisons to specific Lower Bounds (LB) and Optimal Heights (Opt H), are provided in the Appendix (see Tables 5 and 6).

5.2 Results and Discussion

This section analyzes the performance of our methodologies, focusing on the comparison between different strategies and their efficacy against general optimization solvers, all contextualized by established optimal solution values. For detailed instance-by-instance performance of all configurations, please refer to Tables 5 and 6 in the Appendix.

5.2.1 Performance of non-rotational SAT/MaxSAT strategies

For the non-rotational experiments, two symmetry breaking configurations were evaluated: SB C1, which combines Domain Reduction (DR), Large Rectangles (LR), and Same Rectangles (SR); and SB C2, which combines Large Rectangles (LR) and One Pair of Rectangles (OPR) from Soh et al. (2010). We first evaluate the performance of different SAT/MaxSAT strategies for the 2SPP without item rotation. Table 1 shows these findings, with a comprehensive breakdown of instances provided in Table 6 (under the "Without Rotation" columns).

A key observation is the superior performance of our proposed Direct MaxSAT and Non-Incremental SAT Bisection strategies compared to the Incremental SAT Bisection approaches (*SPP_INC_SB_C1*, *SPP_INC_SB_C2*). These incremental methods, representative of earlier SAT-based techniques such as those in Soh et al. (2010), solve fewer instances (23). This difference in performance can be attributed to several factors. Firstly, modern SAT solvers (like Glucose 4.2 used here) and especially MaxSAT solvers (like TT-Open-WBO-Inc) have significantly advanced since the solvers available at the time of earlier works like Soh et al. (2010), offering more powerful search heuristics and learning mechanisms. Secondly, while incremental SAT aims to reuse learned clauses, the overhead of managing a potentially very large initial formula (if H_{ub} is loose, as discussed in Section 4.3 and quantified by the H_{ub} Gap in Table 1) and the cost of evolving this formula can outweigh the benefits for this problem class. The Non-Incremental

Table 1: Performance of non-rotational SAT/MaxSAT strategies (41 Instances). Avg. H_{ub} Gap (%) is the average $(H_{ub} - H_{opt})/H_{opt}$ over solved instances.

Configuration	**# Opt**	$\sum \tau$ **(s)**	**#Vars**	**#Clauses**	**Avg.** H_{ub} **Gap (%)**
Non-Incremental SAT Bisection					
SPP (No SB)	24	1075.43	717,304	31,336,000	1.48
SPP_SB_C1	25	1190.67	714,810	31,081,735	1.43
SPP_SB_C2	24	388.95	714,285	30,936,737	1.47
Incremental SAT Bisection (cf. Soh et al. Soh et al. (2010))					
SPP_INC_SB_C1	23	328.90	885,991	44,681,429	2.89
SPP_INC_SB_C2	23	704.02	885,991	44,463,534	1.68
Direct MaxSAT					
SPP_MS (No SB)	**26**	1722.19	885,991	44,910,928	28.68
SPP_MS_SB_C1	25	1557.03	885,991	44,683,379	31.46
SPP_MS_SB_C2	**26**	1906.08	885,991	44,467,953	28.68

SAT Bisection, by generating fresh, smaller, and more focused SAT instances for each H_{check}, avoids this large initial overhead. Similarly, the Direct MaxSAT approach, while also starting with a large formula up to H_{ub}, leverages the specialized optimization algorithms of TT-Open-WBO-Inc designed to handle such structures effectively. In contrast, the Direct MaxSAT configurations, *SPP_MS* (no SB) and *SPP_MS_SB_C2*, achieve the highest number of optimal solutions (26 instances each), demonstrating the power of leveraging modern MaxSAT solvers like TT-Open-WBO-Inc for direct optimization.

The Non-Incremental SAT Bisection approaches are equally competitive. *SPP_SB_C1* solves 25 instances, and while *SPP_SB_C2* solves 24, it does so with exceptional speed ($\sum \tau = 388.95$s). These methods benefit from relatively small average H_{ub} gaps (around 1.4-1.5%), which define a tighter range for the bisection search. More importantly, each SAT instance $\Phi_R(H_{check})$ is generated for a specific H_{check}, avoiding the large initial formula problem faced by incremental and direct MaxSAT methods. This efficiency suggests that repeatedly solving potentially smaller, focused SAT problems can be more effective than managing the larger, evolving formulas of incremental SAT or the single large formula of MaxSAT for many instances. The overhead of learned clause management and formula evolution in the incremental approach, exacerbated by its initial H_{ub} gap, appears detrimental for this problem class compared to fresh solves. Similarly, while Direct MaxSAT is powerful, its efficiency is tempered by the very large formulas resulting from substantial H_{ub} gaps.

SB constraints show varied effects in the non-rotational context. For Non-Incremental SAT, SB C1 (*SPP_SB_C1*) slightly increases #Opt compared to no SB (*SPP*), while SB

C2 (*SPP_SB_C2*) significantly improves runtime for the same #Opt as *SPP*, highlighting that different SB strategies can offer a trade-off between solution power and speed. For Direct MaxSAT, SB C2 (*SPP_MS_SB_C2*) maintains the same high #Opt as no SB (*SPP_MS*) but with a slightly higher total time. Formula sizes (#Vars, #Cls) are generally larger for Direct MaxSAT and Incremental SAT compared to Non-Incremental SAT.

5.2.2 Performance of rotational SAT/MaxSAT strategies

We now consider the 2SPP with 90-degree item rotations. Optimal heights for rotational variants can differ from their non-rotational counterparts (Kenmochi et al. (2009)). Table 2 presents these results. For a detailed view of how each rotational SAT/MaxSAT strategy performed on individual instances, see the "Rotation" columns in Table 6.

Table 2: Performance of rotational SAT/MaxSAT strategies (41 Instances). Avg. H_{ub} Gap (%) is the average $(H_{ub} - H_{opt})/H_{opt}$ over solved instances.

Configuration	**# Opt**	**$\sum \tau$ (s)**	**#Vars**	**#Clauses**	**Avg. H_{ub} Gap (%)**
Non-Incremental SAT Bisection					
SPP_R (No SB)	25	2932.17	757,356	66,975,824	1.22
SPP_R_SB	**26**	3197.32	757,356	67,246,956	1.13
Incremental SAT Bisection					
SPP_INC_R_SB	21	3897.94	929,557	94,222,243	6.17
Direct MaxSAT					
SPP_MS_R (No SB)	20	1532.58	929,909	93,657,427	44.31
SPP_MS_R_SB	21	2113.95	929,545	93,283,709	40.13

When rotation is allowed, the Non-Incremental SAT Bisection with symmetry breaking (*SPP_R_SB*) solves the most instances (26), matching the best performance in the non-rotational setting and emerging as the top SAT/MaxSAT performer for rotational 2SPP. Its counterpart without SB (*SPP_R*) is close, solving 25 instances. These methods maintain small average H_{ub} gaps (around 1.1-1.2%), contributing to their robustness. This again underscores the effectiveness of the Non-Incremental SAT bisection approach. Symmetry breaking (*SPP_R_SB* vs *SPP_R*) provides a gain of one optimal solution, indicating its utility in navigating the larger search space induced by rotation.

Conversely, the Incremental SAT approach (*SPP_INC_R_SB*) solves fewer instances (21) in the rotational setting. This is likely compounded by a larger average H_{ub} gap of approximately 6.17% (Table 2), which, when combined with the inherent complexity of rotation, further burdens the incremental solver. The Direct MaxSAT (*SPP_MS_R*, *SPP_MS_R_SB*) approaches also solve fewer instances (20-21) in the rotational setting.

These methods face even larger average H_{ub} gaps of around 40.13-44.31% (Table 2). The combination of this substantial initial overestimation of height with the increased complexity of rotational encoding (evident in the higher #Vars and especially #Cls) poses a significant challenge for these strategies, making them less effective with rotation compared to Non-Incremental SAT Bisection. The Direct MaxSAT approach, very strong non-rotationally, appears particularly affected by this increased complexity when rotation is introduced.

Comparing the best results, our SAT/MaxSAT methods solve up to 26 instances optimally, both with and without rotation (*SPP_MS*, *SPP_MS_SB_C2* for non-rotational; *SPP_R_SB* for rotational). For the standard benchmarks used, allowing 90-degree rotation does not enable our SAT/MaxSAT methods to solve more unique instances to their known optima beyond what the best non-rotational configurations achieve. Furthermore, as established in literature such as Kenmochi et al. (2009) and Arahori et al. (2012) for many of these standard instances, rotation does not necessarily lead to packings with smaller optimal heights. The primary impact of incorporating rotation into our SAT/MaxSAT framework is a notable increase in encoding complexity and, consequently, solver runtime. For instance, comparing the baseline non-rotational model (*SPP*) with its rotational counterpart (*SPP_R*), allowing rotation increases the average number of variables by approximately 5.6% (from approx. 717k to 757k) and, more significantly, the average number of clauses by about 113.7% (from approx. 31.3M to 67.0M) for the Non-Incremental Bisection approach. This substantial growth in formula size, particularly in clauses, contributes to the increased computational effort required for rotational problems. Symmetry breaking generally proved beneficial across both rotational and non-rotational settings, either by increasing the number of optima found or by significantly improving solution times, with its application being key for *SPP_R_SB* to reach 26 optima in the rotational context.

5.2.3 Comparison with general Optimization solvers

We now benchmark our most promising SAT/MaxSAT configurations against leading general-purpose CP and MIP solvers. Table 3 provides this comparison. The #Opt here is evaluated against the specific known optimal heights for non-rotational or rotational variants as appropriate. Detailed results for each general optimization solver on every instance are available in Table 5 in the Appendix.

Google OR-Tools' CP-SAT solver demonstrates the strongest performance among the general solvers, with *OR-TOOLS_CP_C2* (non-rotational, SB C2) solving 28 instances optimally (detailed in Table 5). This is the highest number of optimal solutions achieved by any method in this study. Its rotational counterpart, *OR-TOOLS_CP_R_SB*, solves 24 instances. These results highlight the power of modern CP-SAT technology.

Table 3: Performance comparison of selected SAT/MaxSAT configurations and general optimization solvers (41 Instances).

Solver type	**Configuration**	**# Opt**	$\sum \tau$ **(s)**
SAT/MaxSAT (Selected Best)			
Non-Inc. SAT (No Rot, SB C2)	SPP_SB_C2	24	388.95
Direct MaxSAT (No Rot, No SB)	SPP_MS	26	1722.19
Direct MaxSAT (No Rot, SB C2)	SPP_MS_SB_C2	26	1906.08
Non-Inc. SAT (Rot, SB)	SPP_R_SB	26	3197.32
CP Solvers			
CPLEX CP (No Rot, SB C2)	CPLEX_CP_C2	22	1577.02
CPLEX CP (Rot, SB)	CPLEX_CP_R_SB	21	989.12
OR-Tools CP-SAT (No Rot, SB C2)	OR-TOOLS_CP_C2	**28**	3056.64
OR-Tools CP-SAT (Rot, SB)	OR-TOOLS_CP_R_SB	24	1149.13
MIP Solvers			
CPLEX MIP (No Rot, SB C2)	CPLEX_MIP_C2	6	5112.50
CPLEX MIP (Rot, SB)	CPLEX_MIP_R_SB	15	7929.91
Gurobi MIP (No Rot, SB C2)	GUROBI_MIP_C2	13	5488.42
Gurobi MIP (Rot, SB)	GUROBI_MIP_R_SB	15	6708.18
OR-Tools MIP (No Rot, SB C2)	OR-TOOLS_MIP_C2	8	931.71
OR-Tools MIP (Rot, SB)	OR-TOOLS_MIP_R_SB	10	1758.48

Our best SAT/MaxSAT configurations (*SPP_MS*, *SPP_MS_SB_C2*, *SPP_R_SB*), each solving 26 instances, are highly competitive, outperforming other general CP solvers (CPLEX CP) and all tested MIP solvers in terms of #Opt (comparative details in Table 5). The Non-Incremental SAT approach *SPP_SB_C2* (24 Opt) again distinguishes itself with excellent runtime efficiency. While CP-SAT demonstrates the best overall performance, our SAT/MaxSAT approaches offer strong alternative paradigms. They provide a different lens through which to explore the problem's combinatorial structure and can be particularly adept at integrating complex logical side constraints that might be more cumbersome to express in traditional CP or MIP frameworks.

MIP solvers generally struggled with 2SPP instances, finding fewer optimal solutions as shown in Table 3. The combinatorial structure of 2SPP, with its numerous disjunctive constraints (e.g., Non-overlap requirements), is known to challenge typical MIP branch-and-cut algorithms. Such disjunctions often lead to weak linear programming (LP) relaxations in standard MIP formulations and consequently, extensive branching, hindering the solver's ability to effectively prune the search space (Vielma (2015)). Conversely, methods solving a sequence of decision problems, like our Non-Incremental SAT Bisec-

tion and the CP-based approaches using the same Non-Incremental Bisection framework (Algorithm 1), proved more effective. This strategy allows CP and SAT solvers to leverage their strengths in handling logical constraints and applying specialized propagation techniques to focused decision instances (Achterberg (2009)). The strong performance of the OR-Tools CP-SAT solver, in particular, underscores the effectiveness of modern constraint programming technology for tackling the combinatorial nature of 2SPP. This fundamental difference in approach—iterative decision-making versus monolithic MIP optimization—likely explains the superior performance observed for our SAT/MaxSAT methods and the Non-Incremental Bisection-based CP solvers, highlighting the suitability of logical and constraint-based reasoning for 2SPP.

These results collectively underscore the effectiveness of carefully designed SAT and MaxSAT encodings, particularly Non-Incremental Bisection and Direct MaxSAT optimization with modern solvers. They also highlight the strong performance of contemporary CP-SAT technology.

5.2.4 Comparison with prior research results

Table 4 benchmarks our best-performing configurations ("Prop.", derived from Tables 5 and 6) against prior incomplete (Kenmochi et al. (2009); Alvarez-Valdes et al. (2009); Neveu and Trombettoni (2008); Wei et al. (2009)) and exact literature results. Our "Prop." results are highly competitive, matching a significant number of known optima (28 for rotational, 29 for non-rotational as per "Exact" in Table 4). For instance, our best configurations often equal or surpass prior incomplete methods.

Table 4: Comparison of proposed best results (Prop.) with prior incomplete and exact literature results. (38 Instances)

	Rotation				Without Rotation			
Instance	Opt	Prop.	Incomp.	Exact	Opt	Prop.	Incomp.	Exact
BENG01	30	30	30	30	30	30	30	30
BENG02	57	57	57	57	57	57	57	57
BENG03	84	84	84	84	84	84	84	84
BENG04	107	108	107	107	107	108	107	107
BENG05	134	136	134	134	134	134	134	134
BENG06	36	36	36	36	36	36	36	36
BENG07	67	69	67	67	67	69	67	67
BENG08	101	105	101	101	101	102	101	101
BENG09	126	131	126	126	126	128	126	126

Continued on next page

Table 4 – Continued

	Rotation				Without Rotation			
Instance	Opt	Prop.	Incomp.	Exact	Opt	Prop.	Incomp.	Exact
BENG10	156	163	156	156	156	162	156	156
CGCUT01	23	23	23	23	23	23	23	23
CGCUT02	63	63	63	63	64	66	65	64
CGCUT03	-	708	-	-	-	677	658	-
GCUT01	696	696	-	696	1016	1016	1016	1016
GCUT02	-	1149	-	-	1187	1213	1187	1187
GCUT03	-	1696	-	-	1803	1803	1803	1803
GCUT04	-	3034	-	-	-	3125	3002	-
HT01(c1p1)	20	20	20	20	20	20	20	20
HT02(c1p2)	20	20	20	20	20	20	20	20
HT03(c1p3)	20	20	20	20	20	20	20	20
HT04(c2p1)	15	15	15	15	15	15	15	15
HT05(c2p2)	15	15	15	15	15	15	15	15
HT06(c2p3)	15	15	15	15	15	15	15	15
HT07(c3p1)	30	30	30	30	30	30	30	30
HT08(c3p2)	30	30	30	30	30	30	31	30
HT09(c3p3)	30	30	30	30	30	30	30	30
NGCUT01	20	20	20	20	23	23	23	23
NGCUT02	28	28	28	28	30	30	30	30
NGCUT03	28	28	28	28	28	28	28	28
NGCUT04	18	18	18	18	20	20	20	20
NGCUT05	36	36	36	36	36	36	36	36
NGCUT06	29	29	29	29	31	31	31	31
NGCUT07	10	10	10	10	20	20	20	20
NGCUT08	33	33	33	33	33	33	33	33
NGCUT09	49	49	49	49	50	50	50	50
NGCUT10	59	59	59	59	80	80	80	80
NGCUT11	51	51	-	51	52	52	52	52
NGCUT12	77	77	77	77	87	87	87	87
Total #Opt		28	32	**34**		29	32	**36**

While our methods demonstrate strong performance, some prior exact studies report more optimal solutions overall (e.g., 34-36 optima matched in Table 4 under "Exact" compared to our 28-29). A contributing factor might be the differing experimental conditions, particularly the timeout limits. Many prior studies employed longer timeouts (e.g., 3600 seconds), whereas our experiments were conducted with a stricter 1800-

second limit to assess efficiency under tighter constraints. Exploring the impact of extended timeouts on our best configurations remains a direction for future investigation and could potentially bridge these minor gaps. Nevertheless, the current results affirm the state-of-the-art capability of the top configurations identified in our study.

These results collectively underscore the effectiveness of carefully designed SAT and MaxSAT encodings, particularly Non-Incremental SAT Bisection and Direct MaxSAT optimization with modern solvers. They also highlight the strong performance of contemporary CP-SAT technology. Detailed instance-by-instance results, including achieved heights versus known Lower Bounds and Optimal Heights for all evaluated methods, are available in the Appendix (Tables 5 and 6).

6 Conclusion

This paper investigated SAT and MaxSAT methodologies for the 2SPP, including item rotation. We developed robust models, adapted symmetry breaking constraints, compared Non-Incremental SAT Bisection, Incremental SAT Bisection, Direct MaxSAT optimization, and benchmarked them against general-purpose solvers.

Experiments showed that our Non-Incremental SAT Bisection and Direct MaxSAT strategies significantly outperformed earlier incremental SAT techniques. Symmetry breaking generally proved beneficial. While item rotation yielded better heights for some instances (consistent with literature, e.g., Kenmochi et al. (2009); Arahori et al. (2012)), it did not increase the overall count of optimally solved benchmarks by our SAT/MaxSAT methods, mainly adding to encoding complexity and runtime. Against general solvers, OR-Tools' CP-SAT performed best when employed within our Non-Incremental Bisection search framework. However, our leading SAT/MaxSAT configurations were highly competitive, surpassing the CPLEX CP Optimizer and MIP solvers in optimally solved instances, underscoring the suitability of logical approaches for 2SPP. The efficiency of OR-Tools CP-SAT, when applied using this bisection strategy, may stem from its advanced internal heuristics and multi-threading, features not in our current SAT/MaxSAT implementations. The success of our Non-Incremental Bisection methods reinforces the viability of solving a sequence of decision problems for this type of optimization task.

Even with OR-Tools CP-SAT's strong showing under this framework, the developed SAT/MaxSAT methods hold significant value. They offer robust alternative solution pathways, facilitate the exploration of problem structure from a distinct logical viewpoint, can more naturally incorporate intricate logical side-constraints, and provide a solid basis for future hybrid methodologies that could combine the strengths of different paradigms. This study's importance is underscored by its demonstration of improved effectiveness in using Non-Incremental SAT Bisection and Direct MaxSAT optimization

for 2SPP, thorough assessment of symmetry breaking adaptations in rotational environments, and offering of explicit performance comparisons. These findings offer valuable insights for selecting appropriate solution strategies for complex packing problems.

Future work could explore several avenues. Hybrid approaches, combining the strengths of SAT/MaxSAT with other paradigms like CP or local search, might yield further improvements. Investigating more sophisticated symmetry breaking techniques tailored for specific instance structures could also be beneficial. Furthermore, re-evaluating the top-performing configurations identified in this study with extended computational timeouts (e.g., 3600 seconds, comparable to some prior literature) could provide a more direct comparison regarding the absolute number of optimally solved instances and potentially enhance their competitive standing further.

DATA AVAILABILITY. To facilitate transparency and reproducibility, the benchmark instances are made available at `https://github.com/strippacking/SPP`. Additional data supporting this study are available from the corresponding author upon reasonable request.

FUNDING. This work has been supported by VNU University of Engineering and Technology under project number CN24.10.

AUTHORS' CONTRIBUTION. All authors contributed significantly to the preparation of this work. The contributions are as follows: [Tuyen Kieu Van: Conceptualization; Data Curation; Investigation; Methodology; Software; Validation; Visualization; Writing - Original Draft Preparation; Writing - Review & Editing; Duong Quy Le: Software; Validation; Investigation; Visualization; Khanh Van To: Conceptualization; Methodology; Project Administration; Supervision; Writing - Review & Editing].

CONFLICTS OF INTEREST. The authors declare that they have no conflicts of interest related to this work. No competing financial interests or personal relationships that could have appeared to influence the work reported in this paper exist.

References

Achterberg, T. (2009). *Constraint integer programming*. Springer Science & Business Media.

Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M. (2008). Reactive grasp for the strip-packing problem. *Computers & Operations Research*, 35(4):1065–1083.

Alvarez-Valdes, R., Parreño, F., and Tamarit, J. M. (2009). A grasp algorithm for the container loading problem with multi-drop constraints. *Computers & Industrial Engineering*, 56(1):44–56.

Ansótegui, C., Bonet, M. L., and Levy, J. (2013). Solving (weighted) partial MaxSAT through SAT. In *Principles and Practice of Constraint Programming*, pages 115–130. Springer.

Arahori, Y., Imamichi, T., and Nagamochi, H. (2012). An exact strip packing algorithm based on canonical forms. *Computers & Operations Research*, 39(12):2991–3011.

Audemard, G. and Simon, L. (2009). Glucose: a solver that predicts learnt clauses quality. *SAT Competition*, pages 7–8.

Audemard, G. and Simon, L. (2012). Refining restarts strategies for sat and unsat. In *Principles and Practice of Constraint Programming: 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 118–126. Springer.

Beasley, J. (1985a). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36(4):297–306.

Beasley, J. E. (1985b). An exact two-dimensional non-guillotine cutting tree search procedure. *Operations research*, 33(1):49–64.

Bengtsson, B.-E. (1982). Packing rectangular pieces—a heuristic approach. *The computer journal*, 25(3):353–357.

Berger, M., Schröder, M., and Küfer, K.-H. (2009). A constraint-based approach for the two-dimensional rectangular packing problem with orthogonal orientations. In *Operations Research Proceedings 2008: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR) University of Augsburg, September 3-5, 2008*, pages 427–432. Springer.

Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors (2009). *Handbook of Satisfiability*. IOS Press.

Bischoff, E. E. and Ratcliff, M. S. W. (1995). Issues in the development of approaches to container loading. *Omega*, 23(4):377–390.

Boschetti, M. A. and Montaletti, L. (2010). An exact algorithm for the two-dimensional strip-packing problem. *Operations Research*, 58(6):1774–1791.

Burke, E. K., Kendall, G., and Whitwell, G. (2004). A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655–671.

Christofides, N. and Whitlock, C. (1977). An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44.

Coffman, Jr, E. G., Garey, M. R., Johnson, D. S., and Tarjan, R. E. (1980). Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826. Not cited in PDF.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159.

Eén, N. and Sörensson, N. (2003). An extensible sat-solver. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 502–518. Springer.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Gent, I. P. and Smith, B. (2000). Symmetry breaking in constraint programming. In *Proceedings of ECAI 2000 (European Conference on Artificial Intelligence)*, pages 599–603. IOS Press.

Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859.

Google Inc. (2023). OR-Tools: Operations Research Tools. `https://developers.google.com/optimization`. Software library.

Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC. Version 10.0.

Ha, M., Clautiaux, F., Hanafi, S., and Wilbaut, C. (2010). New fast heuristics for the 2d strip packing problem with guillotine constraint. In *Experimental Algorithms: 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings 9*, pages 302–313. Springer.

Hopper, E. and Turton, B. C. (2001). An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):34–57.

IBM Corporation (2023). *IBM ILOG CPLEX Optimization Studio: CPLEX User's Manual*. IBM Corporation. Version 22.1.

Jylänki, J. (2010). A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing. *Retrieved from http://clb.demon.fi/files/RectangleBinPack.pdf*, 1:1–50.

Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M., and Nagamochi, H. (2009). Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research*, 198(1):73–83.

Leung, S., Zhang, D., Zhou, C., and Wu, T. (2011). Two-dimensional packing with rotations and guillotine constraints. *Computers & Operations Research*, 38(1):298–308.

Li, C. M. and Manyà, F. (2009). MaxSAT, hard and soft constraints. *Handbook of Satisfiability*, pages 613–631.

Liu, K., Zhang, H., Wang, C., Li, H., Chen, Y., and Chen, Q. (2023). Robust optimization for the two-dimensional strip-packing problem with variable-sized bins. *Mathematics*, 11(23):4781.

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.

Lodi, A., Martello, S., and Vigo, D. (2015). Heuristic and metaheuristic approaches for packing problems: A survey. In *Proceedings of the International Conference on Combinatorial Optimization*, pages 102–120. Springer.

Martello, S., Monaci, M., and Vigo, D. (2003). An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15(3):310–319.

Morgado, A., Heras, F., Liffiton, M., Planes, J., and Marques-Silva, J. (2013). Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534.

Nadel, A. (2024). TT-Open-WBO-Inc: an efficient anytime MaxSAT solver. *Journal of Satisfiability, Boolean Modeling and Computation*, 15(1):1–7.

Neveu, B. and Trombettoni, G. (2008). Strip packing based on local search and a randomized best-fit. In *Fifth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: First Workshop on Bin Packing and Placement Constraints (CPAIOR: BPPC), Paris, France*.

Sait, S. M. and Youssef, H. (1999). *VLSI Physical Design Automation: Theory and Practice*. McGraw-Hill.

Soh, T., Inoue, K., Tamura, N., Banbara, M., and Nabeshima, H. (2010). A sat-based method for solving the two-dimensional strip packing problem. *Fundamenta Informaticae*, 102(3-4):467–487. Also relevant for defining the standard non-rotational context based on prior SAT work.

Tamura, N., Taga, A., Kitagawa, S., and Banbara, M. (2006). Compiling finite linear CSP into SAT. In *Principles and Practice of Constraint Programming–CP 2006*, pages 590–603. Springer.

Vielma, J. P. (2015). Mixed integer linear programming formulation techniques. *Siam Review*, 57(1):3–57.

Walsh, T. (2006). General symmetry breaking constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 650–664. Springer.

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130.

Wei, L., Qian, F., Du, R., Tian, R., and Chu, X. (2011). Two dimensional packing: The power of rotation. *Theoretical Computer Science*, 412(29):3743–3753.

Wei, L., Zhang, D., and Chen, Q. (2009). A least wasted first heuristic algorithm for the rectangular packing problem. *Computers & Operations Research*, 36(5):1608–1614.

Yang, Y., Zhu, W., Jiang, Y., Zheng, F., and Jiang, Z. (2025). A backtracking heuristic algorithm for two-dimensional strip packing problem with rotations and without guillotine cutting. *Journal of the Operational Research Society*, 76(1):183–197.

Zhang, H., Yao, S., Zhang, S., Leng, J., Wei, L., and Liu, Q. (2024). A block-based heuristic search algorithm for the two-dimensional guillotine strip packing problem. *Engineering Applications of Artificial Intelligence*, 134:108624.

A Detailed experimental results tables

Table 5: Experimental results for Optimization solvers.

Instance	n	W	LB	Rotation						Without Rotation					
				Opt H	CPLEX CP SB	CPLEX MIP SB	Gurobi MIP SB	OR-Tools MIP SB	OR-Tools CP SB	Opt H	CPLEX CP C2	CPLEX MIP C2	Gurobi MIP C2	OR-Tools CP C2	OR-Tools MIP C2
BENG01	20	25	30	30	**30**	46	46	46	**30**	30	**30**	**30**	57	**30**	57
BENG02	40	25	57	57	59	60	83	83	**57**	57	59	94	94	**57**	94
BENG03	60	25	84	84	87	116	116	116	**84**	84	85	138	138	**84**	138
BENG04	80	25	107	107	117	150	150	150	108	107	109	172	172	108	172
BENG05	100	25	134	134	184	184	184	184	136	134	142	217	217	**134**	217
BENG06	40	40	36	36	**36**	48	48	48	**36**	36	**36**	60	60	**36**	60
BENG07	80	40	67	67	73	95	95	95	69	67	69	117	117	69	117
BENG08	120	40	101	101	141	141	141	141	105	101	146	163	163	102	163
BENG09	160	40	126	126	176	176	176	176	131	126	599	220	220	128	220
BENG10	200	40	156	156	223	223	223	223	163	156	741	267	267	162	267
CGCUT01	16	10	23	23	**23**	**23**	40	**23**	**23**	23	**23**	43	43	**23**	**23**
CGCUT02	23	70	63	63	64	64	136	136	**63**	64	66	66	140	66	140
CGCUT03	62	70	636	-	781	781	781	781	708	-	690	805	805	677	805
GCUT01	10	250	655	696	**696**	**696**	**696**	**696**	756	1016	**1016**	1020	**1016**	**1016**	**1016**
GCUT02	20	250	1099	-	1149	1506	1506	1506	1149	1187	1256	1557	1557	1213	1557
GCUT03	30	250	1631	-	1762	2159	2159	2159	1696	1803	**1803**	2291	2291	**1803**	2291
GCUT04	50	250	2926	-	3143	3796	3796	3796	3034	-	3125	3904	3904	3170	3904
HT01(c1p1)	16	20	20	20	**20**	**20**	**20**	21	**20**	20	**20**	43	**20**	**20**	**20**
HT02(c1p2)	17	20	20	20	**20**	**20**	**20**	38	**20**	20	**20**	46	46	**20**	46
HT03(c1p3)	16	20	20	20	**20**	**20**	**20**	**20**	**20**	20	**20**	**20**	**20**	**20**	39
HT04(c2p1)	25	40	15	15	**15**	16	**15**	16	**15**	15	**15**	23	23	**15**	23
HT05(c2p2)	25	40	15	15	**15**	16	**15**	38	**15**	15	**15**	26	26	**15**	26
HT06(c2p3)	25	40	15	15	**15**	**15**	32	16	**15**	15	**15**	29	29	**15**	29
HT07(c3p1)	28	60	30	30	31	63	31	33	31	30	31	65	65	**30**	65
HT08(c3p2)	29	60	30	30	31	72	31	72	**30**	30	31	51	51	**30**	51
HT09(c3p3)	28	60	30	30	31	66	31	32	**30**	30	**30**	62	62	**30**	62
HT10(c4p1)	49	60	60	60	62	117	117	117	62	60	62	143	143	61	143
HT11(c4p2)	49	60	60	60	61	134	134	134	61	60	62	145	145	61	71
HT12(c4p3)	49	60	60	60	61	104	104	104	61	60	61	150	150	61	71
NGCUT01	10	10	19	20	**20**	**20**	**20**	**20**	**20**	23	**23**	41	**23**	**23**	41
NGCUT02	17	10	28	28	**28**	39	39	**28**	**28**	30	**30**	64	**30**	**30**	64
NGCUT03	21	10	28	28	**28**	43	43	43	**28**	28	**28**	59	59	**28**	59
NGCUT04	7	10	17	18	**18**	**18**	**18**	**18**	19	20	**20**	**20**	**20**	**20**	**20**
NGCUT05	14	10	36	36	**36**	**36**	**36**	**36**	**36**	36	**36**	48	**36**	**36**	**36**
NGCUT06	15	10	29	29	**29**	**29**	**29**	41	**29**	31	**31**	69	**31**	**31**	**31**
NGCUT07	8	20	9	10	**10**	**10**	**10**	**10**	**10**	20	**20**	**20**	**20**	**20**	**20**
NGCUT08	13	20	32	33	**33**	**33**	**33**	**33**	**33**	33	**33**	**33**	**33**	**33**	**33**
NGCUT09	18	20	49	49	**49**	**49**	71	71	**49**	50	**50**	107	107	51	107
NGCUT10	13	30	58	59	**59**	**59**	**59**	**59**	**59**	80	**80**	171	**80**	**80**	171
NGCUT11	15	30	50	51	52	**51**	**51**	75	**51**	52	**52**	**52**	**52**	**52**	130
NGCUT12	22	30	77	77	**77**	79	**77**	118	78	87	92	173	**87**	**87**	173
Total # of Opt.					21	15	15	10	**24**		22	6	13	**28**	8

Table 6: Experimental results for SAT/MaxSAT based solvers.

Instance	n	W	LB	Rotation						Without Rotation								
				Opt H	INC R SB	MS R	MS R SB	R	R SB	Opt H	SPP	INC SB C1	INC SB C2	MS	MS SB C1	MS SB C2	SB C1	SB C2
BENG01	20	25	30	30	**30**	**30**	**30**	**30**	**30**	30	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
BENG02	40	25	57	57	**57**	**57**	**57**	**57**	**57**	57	58	58	58	**57**	**57**	**57**	**57**	58
BENG03	60	25	84	84	86	**84**	**84**	86	86	84	86	86	86	**84**	**84**	**84**	86	86
BENG04	80	25	107	107	110	172	172	110	110	107	110	110	110	172	172	172	110	110
BENG05	100	25	134	134	138	217	217	138	138	134	138	138	138	**134**	217	**134**	138	138
BENG06	40	40	36	36	**36**	**36**	**36**	**36**	**36**	36	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
BENG07	80	40	67	67	69	117	117	69	69	67	69	69	69	117	117	117	69	69
BENG08	120	40	101	101	108	163	163	108	108	101	108	108	108	163	163	163	108	108
BENG09	160	40	126	126	137	220	220	131	131	126	131	137	137	220	220	220	131	131
BENG10	200	40	156	156	169	267	267	169	169	156	169	169	169	267	267	267	169	169
CGCUT01	16	10	23	23	**23**	**23**	**23**	**23**	**23**	23	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
CGCUT02	23	70	63	63	64	140	140	64	64	64	66	66	66	140	140	140	66	66
CGCUT03	62	70	636	-	720	805	805	720	720	-	720	720	720	805	805	805	720	720
GCUT01	10	250	655	696	**696**	**696**	**696**	**696**	**696**	1016	**1016**	**1016**	**1016**	**1016**	**1016**	**1016**	**1016**	**1016**
GCUT02	20	250	1099	-	1213	1557	1557	1155	1155	1187	1213	1557	1213	1557	1557	1557	1213	1213
GCUT03	30	250	1631	-	1795	2291	2291	1795	1795	1803	1961	2291	1961	2291	2291	2291	1961	1961
GCUT04	50	250	2926	-	3170	3904	3904	3170	3170	-	3170	3904	3170	3904	3904	3904	3170	3170
HT01(c1p1)	16	20	20	20	**20**	**20**	**20**	**20**	**20**	20	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
HT02(c1p2)	17	20	20	20	**20**	**20**	**20**	**20**	**20**	20	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
HT03(c1p3)	16	20	20	20	**20**	**20**	**20**	**20**	**20**	20	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
HT04(c2p1)	25	40	15	15	**15**	**15**	**15**	**15**	**15**	15	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
HT05(c2p2)	25	40	15	15	26	26	26	**15**	**15**	15	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
HT06(c2p3)	25	40	15	15	16	16	16	**15**	**15**	15	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
HT07(c3p1)	28	60	30	30	31	65	65	31	**30**	30	**30**	**30**	**30**	**30**	65	**30**	**30**	**30**
HT08(c3p2)	29	60	30	30	31	**30**	**30**	**30**	**30**	30	31	31	31	51	**30**	51	**30**	**30**
HT09(c3p3)	28	60	30	30	**30**	62	62	**30**	**30**	30	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
HT10(c4p1)	49	60	60	60	61	143	143	64	64	60	64	64	64	143	143	143	64	64
HT11(c4p2)	49	60	60	60	61	145	145	61	61	60	61	61	61	145	145	145	61	61
HT12(c4p3)	49	60	60	60	61	150	150	61	61	60	61	61	61	150	150	150	61	61
NGCUT01	10	10	19	20	**20**	**20**	**20**	**20**	**20**	23	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
NGCUT02	17	10	28	28	**28**	**28**	**28**	**28**	**28**	30	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
NGCUT03	21	10	28	28	**28**	**28**	**28**	**28**	**28**	28	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
NGCUT04	7	10	17	18	**18**	**18**	**18**	**18**	**18**	20	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
NGCUT05	14	10	36	36	**36**	**36**	**36**	**36**	**36**	36	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
NGCUT06	15	10	29	29	**29**	**29**	**29**	**29**	**29**	31	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
NGCUT07	8	20	9	10	20	20	20	**10**	**10**	20	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
NGCUT08	13	20	32	33	**33**	**33**	**33**	**33**	**33**	33	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
NGCUT09	18	20	49	49	**49**	**49**	**49**	**49**	**49**	50	100	**50**	**50**	**50**	**50**	**50**	100	**50**
NGCUT10	13	30	58	59	**59**	**59**	**59**	**59**	**59**	80	160	**80**	**80**	**80**	**80**	**80**	**80**	**80**
NGCUT11	15	30	50	51	**51**	130	**51**	**51**	**51**	52	**52**	**52**	**52**	**52**	**52**	**52**	**52**	**52**
NGCUT12	22	30	77	77	**77**	173	173	**77**	**77**	87	88	88	88	173	173	173	88	88
Total # of Opt.					21	20	21	25	**26**		24	23	23	**26**	25	**26**	24	25