A SAT-based Method for Solving the Two-dimensional Strip Packing problem

Takehide Soh[1] Katsumi Inoue[12] Naoyuki Tamura[3]
Mutsunori Banbara[3] Hidetomo Nabeshima[4]

[1]Department of Informatics
The Graduate University of Advanced Studies

[2]National Institute of Informatics

[3]Kobe University

[4]University of Yamanashi

The 15th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion
Udine, 12-13 December 2008

Background

- Packing problems have many practical applications such as truck loading, LSI layouts and assignments of newspaper articles.
- There has been a great deal of research on these problems: knapsack problems and bin packing problems.
- Enormous progress in performance of *SAT solvers* has been made in the last decade.
- Such progress has enabled it to apply several problems: hardware verification, planning, and scheduling.

We focus the *two-dimensional strip packing problem* (2SPP) and propose a SAT-based exact approach for solving 2SPP.

Background

- Packing problems have many practical applications such as truck loading, LSI layouts and assignments of newspaper articles.
- There has been a great deal of research on these problems: knapsack problems and bin packing problems.
- Enormous progress in performance of *SAT solvers* has been made in the last decade.
- Such progress has enabled it to apply several problems: hardware verification, planning, and scheduling.

We focus the *two-dimensional strip packing problem* (2SPP) and propose a SAT-based exact approach for solving 2SPP.

Background

- Packing problems have many practical applications such as truck loading, LSI layouts and assignments of newspaper articles.
- There has been a great deal of research on these problems: knapsack problems and bin packing problems.
- Enormous progress in performance of *SAT solvers* has been made in the last decade.
- Such progress has enabled it to apply several problems: hardware verification, planning, and scheduling.

We focus the *two-dimensional strip packing problem* (2SPP) and propose a SAT-based exact approach for solving 2SPP.

Contents

Two-dimensional strip packing problem (2SPP)

Definition of 2SPP (Optimization problem)

Input: A set $R = \{r_1, \ldots, r_n\}$ of n rectangles. Each rectangle $r_i \in R$ has a width w_i and a height h_i ($w_i, h_i \in \mathbb{N}$). A *Strip* of width $W \in \mathbb{N}$.

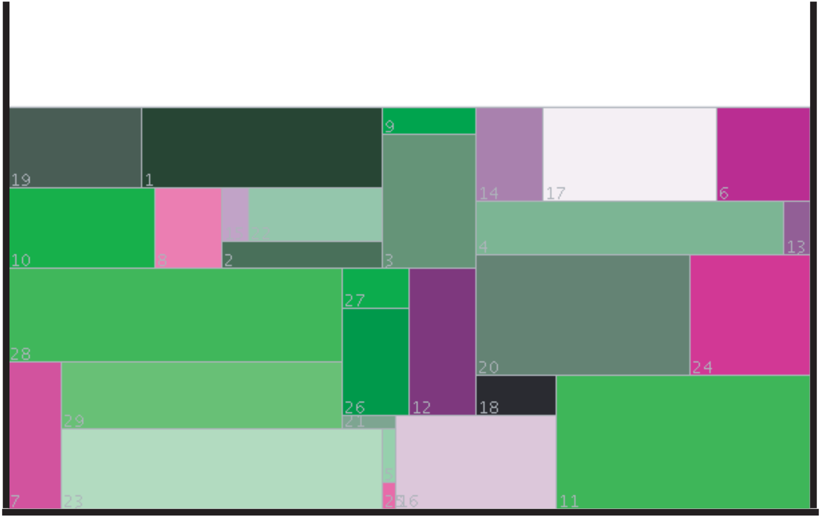
Constraints: Each rectangle cannot overlap with the others and the edges of the strip and must be parallel to the horizontal and the vertical axis.

Question: What is the minimum height such that the set of rectangles can be packed in the given strip?

Definition of two-dimensional orthogonal packing problem (2OPP, Decision problem)

2OPP is a decision problem of the 2SPP with a fixed height of the strip.

2SPP Example: HT08



Previous research

Many algorithms have been developed for solving 2SPP.

- **Exact method:**
  - Branch and bound [Martello et al. 2003]
- **Incomplete methods:**
  - Best Fit Algorithm [Burke et al. 2004]
  - Randomized Best Fit [Neveu and Trombettoni 2008]
  - Reactive GRASP [Alvarez-Valdes et al. 2008]
  - Least wasted first heuristic [Wei et al. 2008]

Some benchmark problems are very hard to solve. In particular, the following 6 problems are still open.

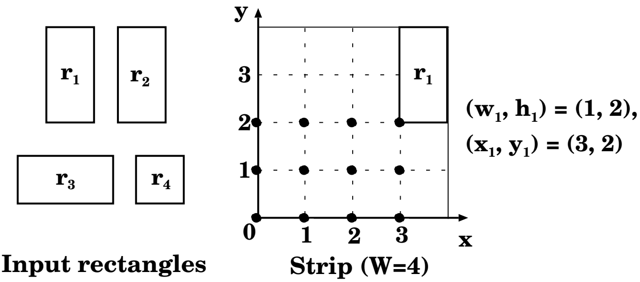
HT08, CGCUT02, CGCUT03, GCUT02, GCUT04, NGCUT09.

Contents

CSP representation of 2OPP

Let x_i and y_i be integer variables such that the pair (x_i, y_i) represents the position of lower left coordinates of the rectangle r_i in the strip. The domains of x_i and y_i are as follows.

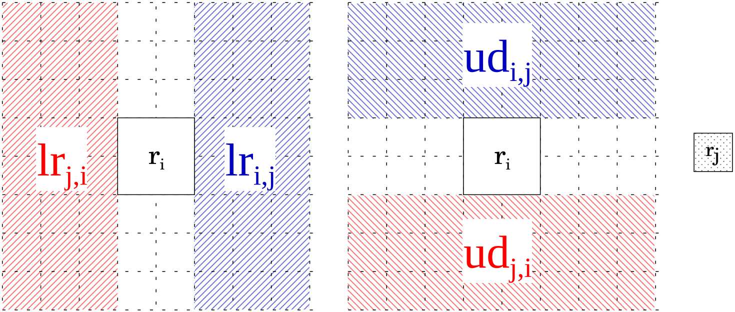
$$\begin{aligned} D(x_i) &= \{a \in \mathbb{N} \mid 0 \le a \le W - w_i\} \\ D(y_i) &= \{a \in \mathbb{N} \mid 0 \le a \le H - h_i\} \end{aligned}$$



CSP representation of 2OPP (Cont.)

Let $r_i, r_j \in R$ $(i \neq j)$ be two rectangles in a 2OPP. We use two kinds of propositional variables: $lr_{i,j}$ and $ud_{i,j}$.

- $lr_{i,j}$ is $true$ if r_i are placed at the left to the r_j.
- $ud_{i,j}$ is $true$ if r_i are placed at the downward to the r_j.

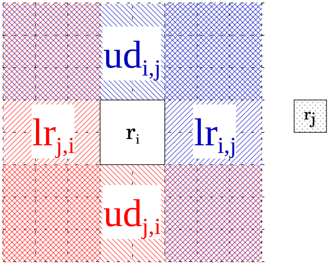


CSP representation of 2OPP (Cont.)

For each rectangles r_i, r_j $(i < j)$, we have the non-overlapping constraints:

$$lr_{i,j} \lor lr_{j,i} \lor ud_{i,j} \lor ud_{j,i}$$

$$\begin{array}{ccc} \neg lr_{i,j} & \lor & x_i + w_i \le x_j \\ \neg lr_{j,i} & \lor & x_j + w_j \le x_i \\ \neg ud_{i,j} & \lor & y_i + h_i \le y_j \\ \neg ud_{j,i} & \lor & y_j + h_j \le y_i \end{array}$$



Encoding 2OPP into SAT

For each rectangles r_i, r_j $(i < j)$, we have the non-overlapping constraints:

$$lr_{i,j} \lor lr_{j,i} \lor ud_{i,j} \lor ud_{j,i}$$

$$\begin{array}{lcl} \neg lr_{i,j} & \lor & \underline{x_i + w_i \le x_j} \\ \neg lr_{j,i} & \lor & \underline{x_j + w_j \le x_i} \\ \neg ud_{i,j} & \lor & \underline{y_i + h_i \le y_j} \\ \neg ud_{j,i} & \lor & \underline{y_j + h_j \le y_i} \end{array}$$

The underlined parts are encoded into SAT by using order encoding.

Order Encoding (Tamura et al. 2006)

- Order encoding is a generalization of the encoding method originally used by Crawford and Baker for Job-Shop Scheduling problems.
- It uses a different Boolean variable $P_{x,a}$ representing $x \le a$ for each integer variable x and integer value a.

$$P_{x,a} \iff x \le a$$

- It naturally represents the order relation of integers.
- It is better for various problems than other encodings, such as direct encoding and support encoding.

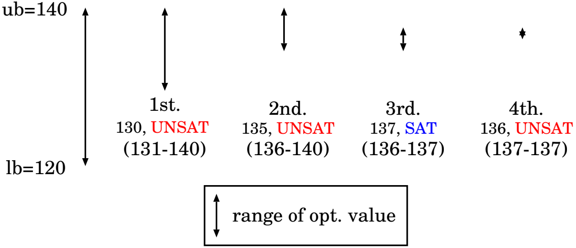
Solving 2SPP with a SAT solver

Bisection Method

while $lb < ub$
  $o := (lb + ub)/2;$
  $result := \Psi \cup \{ph_o\};$
  if $result$ is **SAT**
    then $ub := o;$
    else $lb := o + 1;$
end while

Example:

1st step: 130(UNSAT)
2nd step: 135(UNSAT)
3rd step: 137(SAT)
4th step: 136(UNSAT)



Contents

For solving 2SPP efficiently

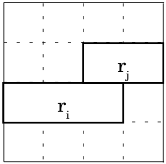
Techniques to reduce the search space by using symmetries and relations of rectangles:

- Large rectangles (LR)
  - Reducing the possibilities for placing large rectangles.
- Same rectangles (SR)
  - Breaking symmetries for same-sized rectangles.
- Largest rectangle (LS)
  - Reducing the domain for the largest rectangle.
- One pair of rectangles (LP)
  - Breaking symmetries for the largest pair of rectangles.

Techniques to reuse the followings during the search:

- Learned clauses (LC)
  - Reusing learned clauses reported by [Marques-Silva and Sakallah 1999].
- Assumptions (AS)
  - Reusing assumptions reported by [Eén and Sörensson].

LR: Reducing the possibilities for placing large rectangles

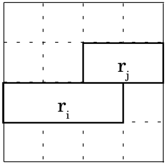


For each rectangle r_i and r_j, if $w_i + w_j > W$ we can not pack these rectangles in the horizontal direction. We therefore modify non-overlapping constraints as follows:

$$
\begin{array}{l}
lr_{i,j} \lor lr_{j,i} \lor ud_{i,j} \lor ud_{j,i} \\
\neg lr_{i,j} \lor x_i + w_i \le x_j \\
\neg lr_{j,i} \lor x_j + w_j \le x_i \\
\neg ud_{i,j} \lor y_i + h_i \le y_j \\
\neg ud_{j,i} \lor y_j + h_j \le y_i
\end{array}
$$

This technique is also applicable for the vertical direction.

LR: Reducing the possibilities for placing large rectangles

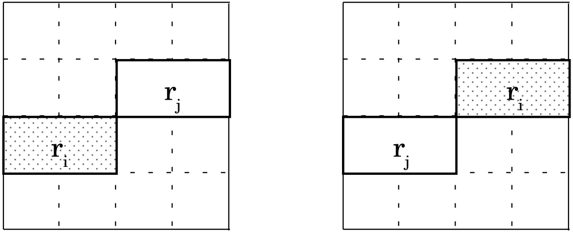


For each rectangle r_i and r_j, if $w_i + w_j > W$ we can not pack these rectangles in the horizontal direction. We therefore modify non-overlapping constraints as follows:

$$\cancel{lr_{i,j} \vee lr_{j,i}} \vee ud_{i,j} \vee ud_{j,i}$$
$$\cancel{\neg lr_{i,j} \vee x_i + w_i \le x_j}$$
$$\cancel{\neg lr_{j,i} \vee x_j + w_j \le x_i}$$
$$\neg ud_{i,j} \vee y_i + h_i \le y_j$$
$$\neg ud_{j,i} \vee y_j + h_j \le y_i$$

This technique is also applicable for the vertical direction.

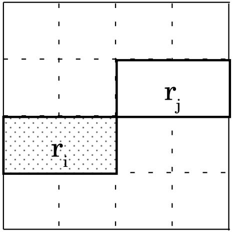
SR: Breaking symmetries for same-sized rectangles

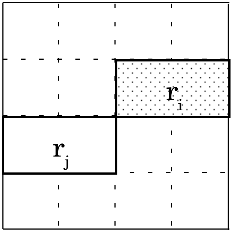


For each rectangle r_i and r_j, if $(w_i, h_i) = (w_j, h_j)$ we can fix the positional relation of these rectangles. We therefore modify non-overlapping constraints as follows:

$$lr_{i,j} \vee lr_{j,i} \vee ud_{i,j} \vee ud_{j,i}$$
$$\neg lr_{i,j} \vee x_i + w_i \le x_j$$
$$\neg lr_{j,i} \vee x_j + w_j \le x_i$$
$$\neg ud_{i,j} \vee y_i + h_i \le y_j$$
$$\neg ud_{j,i} \vee y_j + h_j \le y_i$$
$$\neg ud_{i,j} \vee lr_{j,i}$$

SR: Breaking symmetries for same-sized rectangles

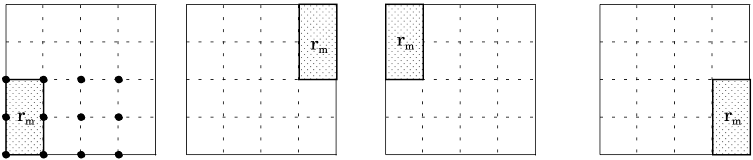




For each rectangle r_i and r_j, if $(w_i, h_i) = (w_j, h_j)$ we can fix the positional relation of these rectangles. We therefore modify non-overlapping constraints as follows:

$$
\begin{array}{l}
lr_{i,j} \cancel{\vee lr_{j,i}} \vee ud_{i,j} \vee ud_{j,i} \\
\neg lr_{i,j} \vee x_i + w_i \le x_j \\
\cancel{\neg lr_{j,i} \vee x_j + w_j \le x_i} \\
\neg ud_{i,j} \vee y_i + h_i \le y_j \\
\neg ud_{j,i} \vee y_j + h_j \le y_i \\
\neg ud_{i,j} \vee lr_{j,i}
\end{array}
$$

LS: Reducing the domain for the largest rectangle

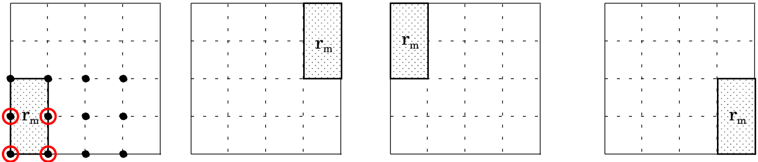


a. Original b. Point Symmetry c. Reflective Symmetry (horizontal) d. Reflective Symmetry (vertical)

With breaking symmetries, we can reduce the domain of largest rectangle.

$$D(x_m) = \{a \in \mathbb{N} \mid 0 \leq a \leq W - w_m\}$$
$$D(y_m) = \{a \in \mathbb{N} \mid 0 \leq a \leq H - h_m\}$$

LS: Reducing the domain for the largest rectangle



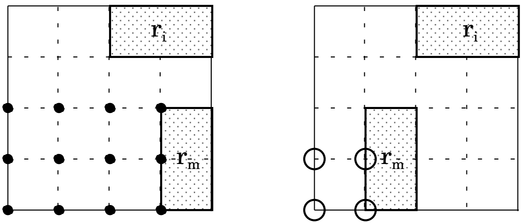
a. Original b. Point Symmetry c. Reflective Symmetry (horizontal) d. Reflective Symmetry (vertical)

With breaking symmetries, we can reduce the domain of largest rectangle.

$$D(x_m) = \{a \in \mathbb{N} \mid 0 \leq a \leq \lfloor \frac{W - w_m}{2} \rfloor\}$$

$$D(y_m) = \{a \in \mathbb{N} \mid 0 \leq a \leq \lfloor \frac{H - h_m}{2} \rfloor\}$$

LS: Reducing the domain for the largest rectangle

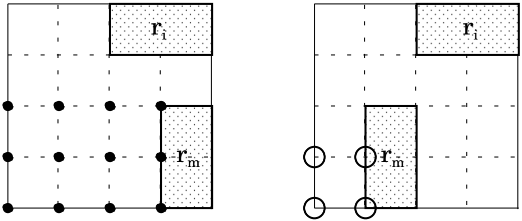


With applying the domain reduction,

if w_i satisfies $w_i > \lfloor \dfrac{W - w_m}{2} \rfloor$, we therefore modify non-overlapping constraints as follows.

$$
\begin{array}{l}
lr_{i,m} \vee lr_{m,i} \vee ud_{i,m} \vee ud_{m,i} \\
\neg lr_{i,m} \vee x_i + w_i \leq x_m \\
\neg lr_{m,i} \vee x_m + w_m \leq x_i \\
\neg ud_{i,m} \vee y_i + h_i \leq y_m \\
\neg ud_{m,i} \vee y_m + h_m \leq y_i
\end{array}
$$

LS: Reducing the domain for the largest rectangle

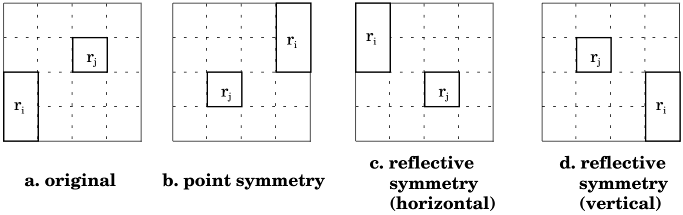


With applying the domain reduction,

if w_i satisfies $w_i > \lfloor \frac{W - w_m}{2} \rfloor$, we therefore modify non-overlapping constraints as follows.

$$\cancel{lr_{i,m}} \vee lr_{m,i} \vee ud_{i,m} \vee ud_{m,i}$$
$$\cancel{\neg lr_{i,m} \vee x_i + w_i \le x_m}$$
$$\neg lr_{m,i} \vee x_m + w_m \le x_i$$
$$\neg ud_{i,m} \vee y_i + h_i \le y_m$$
$$\neg ud_{m,i} \vee y_m + h_m \le y_i$$

LP: Breaking symmetries for the largest pair of rectangles

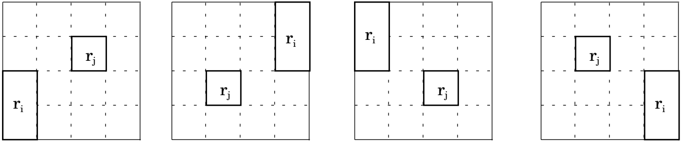


For only one pair of rectangles r_i and r_j, we can restrict their positional relation by using symmetries.

$$lr_{i,j} \vee lr_{j,i} \vee ud_{i,j} \vee ud_{j,i}$$
$$\neg lr_{i,j} \vee x_i + w_i \leq x_j$$
$$\neg lr_{j,i} \vee x_j + w_j \leq x_i$$
$$\neg ud_{i,j} \vee y_i + h_i \leq y_j$$
$$\neg ud_{j,i} \vee y_j + h_j \leq y_i$$

This can not be used with domain reduction (DR) technique.

LP: Breaking symmetries for the largest pair of rectangles



a. original **b. point symmetry** **c. reflective symmetry (horizontal)** **d. reflective symmetry (vertical)**

For only one pair of rectangles r_i and r_j, we can restrict their positional relation by using symmetries.

$$lr_{i,j} \cancel{\vee lr_{j,i}} \vee ud_{i,j} \cancel{\vee ud_{j,i}}$$
$$\neg lr_{i,j} \vee x_i + w_i \leq x_j$$
$$\cancel{\neg lr_{j,i} \vee x_j + w_j \leq x_i}$$
$$\neg ud_{i,j} \vee y_i + h_i \leq y_j$$
$$\cancel{\neg ud_{j,i} \vee y_j + h_j \leq y_i}$$

This can not be used with domain reduction (DR) technique.

Reusing Learned clauses

- Most state-of-the-art solvers implement Clause learning technique [Marques-Silva and Sakallah 1999].
- Minisat efficiently implements this technique.
- Since all SAT-encoded 2OPPs satisfy *reusability condition of learned clauses* [Nabeshima *et al.* 2006][1], learned clauses are available to use for another problems.

Example.

1st step	$H = 130$	$\Psi \cup \{ph_{130}\}$
2nd step	$H = 135$	$\Psi \cup \{ph_{135}\} \cup \{L_{130}\}$
3rd step	$H = 137$	$\Psi \cup \{ph_{137}\} \cup \{L_{130}\} \cup \{L_{135}\}$
4th step	$H = 136$	$\Psi \cup \{ph_{136}\} \cup \{L_{130}\} \cup \{L_{135}\} \cup \{L_{137}\}$

[1] If two SAT problems satisfy following condition, we can reuse learned clauses. Let P and Q be SAT problems such that all non-unit clauses of P are included in Q.

Reusing Assumptions

- Eén and Sörensson proposed a particular set of a unit clauses called *assumptions*.
- An assumption is added before solving the problem, and then removed from the problem.
- By reusing these unit clauses, we can avoid a redundant search space.

Example.

1st.	$H = 130(UNSAT)$	$\Psi \cup \{ph_{130}\}$
2nd.	$H = 135(UNSAT)$	$\Psi \cup \{ph_{135}\} \cup \{\neg ph_{130}\}$
3rd.	$H = 137(SAT)$	$\Psi \cup \{ph_{137}\} \cup \{\neg ph_{130}\} \cup \{\neg ph_{135}\}$
4th.	$H = 136(UNSAT)$	$\Psi \cup \{ph_{136}\} \cup \{\neg ph_{130}\} \cup \{\neg ph_{135}\} \cup ph_{137}\}$

Contents

- Introduction
  - Background
  - Definition of 2SPP and 2OPP
  - Previous Research
- Solving 2SPP with a SAT solver
  - CSP formalization of 2OPP
  - CSP into SAT problems
  - Bisection method
- Techniques to reduce the search space
  - Reducing the search space with breaking symmetries
  - Reusing learned clause and assumption
- Experimental results

Performance Evaluation

Benchmark sets:

CGCUT*	3 instances	[Christofides and Whitlock 1977]
BENG*	10 instances	[Bengtsson1982]
GCUT*	4 instances	[Beasley 1985]
NGCUT*	12 instances	[Beasley 1985]
HT*	9 instances	[Hopper and Turton 2001]
Total	38 instances	

Environment:

- Machine CPU:Xeon 2.66GHz, Mem:2GB
- SAT Solver: Minisat v2.0
- Time Limit: 1 hour (3600 seconds)

Benchmark results

28 problems including two open problems were solved. That is to say, their minimum heights are found and proved to be optimum.

Instance name	Previous result		New results	
	LB	UB	LB	UB
HT08(c3p2)	30	31	30	30
NGCUT09	49	50	50	50

- For HT08, we improve its upper bound and succeed to obtain its optimal solution.
- For NGCUT09, we improve its lower bound and succeed to obtain its optimal solution.

Results of Reducing Techniques

Reducing \ Reusing	*none*	LC+AS
none	**24**	26
LR	24	26
LS	26	26
LS+LR	26	27
SR	27	24
SR+LR	**28**	24
SR+LS	26	27
SR+LS+LR	26	28
LP	25	27
LP+LR	25	**28**
LP+SR	26	25
LP+SR+LR	26	25
total	309	313

- The combinations SR+LR and LP+LR are the best in our reducing techniques.
- The combination LC+AS solves 4 more instances than the method without reusing techniques.

Conclusion

We presented a SAT-based exact method to solve the two-dimensional strip packing problem.

- Our approach solved two open problems HT08 and NGCUT09 in 2SPP.
- Reducing and reusing techniques improve the performance.

Future Works

- Applying our SAT-based approach to other packing problems by taking the rotation of rectangles into considerations.
- Developing a hybrid system which includes incomplete methods as well as exact methods.

Thank you for your attention.

Order Encoding

Order Encoding (Tamura et al. 2006)

- Order encoding is a generalization of the encoding method originally used by Crawford and Baker for Job-Shop Scheduling problems.
- It uses a different Boolean variable $P_{x,a}$ representing $x \leq a$ for each integer variable x and integer value a.

$$P_{x,a} \iff x \leq a$$

- For example, the following four Boolean variables are used to encode an integer variable $x \in \{1, 2, 3, 4, 5\}$.

$$P_{x,1} \qquad P_{x,2} \qquad P_{x,3} \qquad P_{x,4}$$

Please note $P_{x,5}$ (i.e. $x \leq 5$) is not necessary because it is always true.

Order Encoding (Tamura et al. 2006)

- Order encoding is a generalization of the encoding method originally used by Crawford and Baker for Job-Shop Scheduling problems.
- It uses a different Boolean variable $P_{x,a}$ representing $x \leq a$ for each integer variable x and integer value a.

$$P_{x,a} \iff x \leq a$$

- For example, the following four Boolean variables are used to encode an integer variable $x \in \{1, 2, 3, 4, 5\}$.

$$P_{x,1} \qquad P_{x,2} \qquad P_{x,3} \qquad P_{x,4}$$

Please note $P_{x,5}$ (i.e. $x \leq 5$) is not necessary because it is always true.

Order encoding of variables

- Integer variable $x \in \{1, 2, 3, 4, 5\}$ can be encoded into the following *three* SAT clauses while the direct encoding requires 11 clauses (one at-least-one clause and 10 at-most-one clauses).

$$\neg P_{x,1} \lor P_{x,2} \qquad \text{(i.e. } (x \le 1) \supset (x \le 2))$$
$$\neg P_{x,2} \lor P_{x,3} \qquad \text{(i.e. } (x \le 2) \supset (x \le 3))$$
$$\neg P_{x,3} \lor P_{x,4} \qquad \text{(i.e. } (x \le 3) \supset (x \le 4))$$

- The followings are the satisfiable assignments.

$P_{x,1}$	$P_{x,2}$	$P_{x,3}$	$P_{x,4}$	Interpretation
T	T	T	T	$x = 1$
F	T	T	T	$x = 2$
F	F	T	T	$x = 3$
F	F	F	T	$x = 4$
F	F	F	F	$x = 5$

Order encoding of variables

- Integer variable $x \in \{1, 2, 3, 4, 5\}$ can be encoded into the following *three* SAT clauses while the direct encoding requires 11 clauses (one at-least-one clause and 10 at-most-one clauses).

$$
\begin{array}{ll}
\neg P_{x,1} \lor P_{x,2} & \text{(i.e. } (x \le 1) \supset (x \le 2)) \\
\neg P_{x,2} \lor P_{x,3} & \text{(i.e. } (x \le 2) \supset (x \le 3)) \\
\neg P_{x,3} \lor P_{x,4} & \text{(i.e. } (x \le 3) \supset (x \le 4))
\end{array}
$$

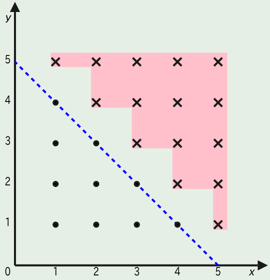
- The followings are the satisfiable assignments.

$P_{x,1}$	$P_{x,2}$	$P_{x,3}$	$P_{x,4}$	Interpretation
T	T	T	T	$x = 1$
F	T	T	T	$x = 2$
F	F	T	T	$x = 3$
F	F	F	T	$x = 4$
F	F	F	F	$x = 5$

Order encoding of linear constraints

- Constraints can be encoded by representing conflict regions instead of conflict points as used in direct encoding.
- For example, $x + y \leq 5$ is encoded into the following *five* SAT clauses while the direct encoding requires 15 clauses.

Clauses	Conflict regions
$P_{y,4}$	$\neg((x \geq 1) \wedge (y \geq 5))$
$P_{x,1} \vee P_{y,3}$	$\neg((x \geq 2) \wedge (y \geq 4))$
$P_{x,2} \vee P_{y,2}$	$\neg((x \geq 3) \wedge (y \geq 3))$
$P_{x,3} \vee P_{y,1}$	$\neg((x \geq 4) \wedge (y \geq 2))$
$P_{x,4}$	$\neg((x \geq 5) \wedge (y \geq 1))$



2OPP into SAT problems

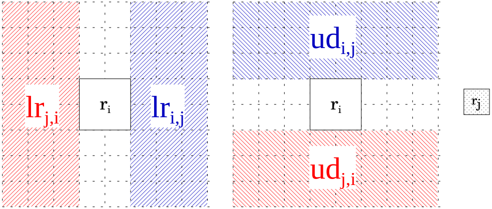
From 2OPP into SAT Problems (1)

Variables

Let $r_i, r_j \in R$ $(i \neq j)$ be two rectangles in a 2OPP. Let e and f be any integer.

Then, the SAT encoding of a 2OPP uses four kinds of atoms, $lr_{i,j}$, $ud_{i,j}$, $px_{i,e}$, and $py_{i,f}$.

- $lr_{i,j}$ is $true$ if r_i are placed at the left to the r_j.
- $ud_{i,j}$ is $true$ if r_i are placed at the downward to the r_j.
- $px_{i,e}$ is $true$ if r_i are placed at less than or equal to e.
- $py_{i,f}$ is $true$ if r_i are placed at less than or equal to f.



From 2OPP into SAT Problems (2)
Axiom Clauses

For each rectangle r_i, and integer e and f such that $0 \leq e \leq W - w_i$ and $0 \leq f \leq H - h_i$, we have the 2-literal axiom clauses due to order encoding,

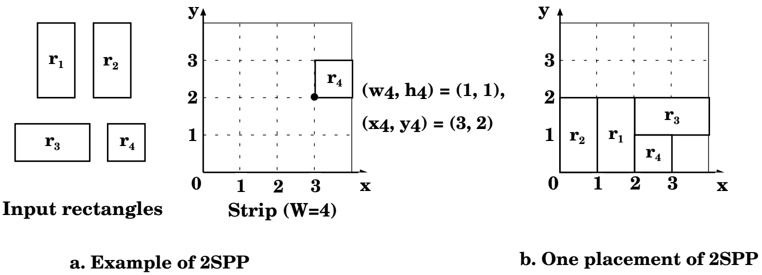
$$\begin{aligned} &\neg px_{i,e} \vee px_{i,e+1} \\ &\neg py_{i,f} \vee py_{i,f+1} \end{aligned} \tag{1}$$

For each rectangles i, j $(i < j)$, we have the non-overlapping constraints as the 4-literal clauses:

$$lr_{i,j} \lor lr_{j,i} \lor ud_{i,j} \lor ud_{j,i} \tag{2}$$

For each rectangles i, j $(i < j)$, and integer e and f such that $0 \le e < W - w_i$ and $0 \le f < H - h_j$, we also have the non-overlapping constraints as the 3-literal clauses:

$$\begin{aligned}
\neg lr_{i,j} &\lor px_{i,e} \lor \neg px_{j,e+w_i} \\
\neg lr_{j,i} &\lor px_{j,e} \lor \neg px_{i,e+w_j} \\
\neg ud_{i,j} &\lor py_{i,f} \lor \neg py_{j,f+h_i} \\
\neg ud_{j,i} &\lor py_{j,f} \lor \neg py_{i,f+h_j}
\end{aligned} \tag{3}$$



r1
r2
r3
r4
Input rectangles
y
3
2
1
0
1
2
3
x
r4
(w4, h4) = (1, 1),
(x4, y4) = (3, 2)
Strip (W=4)
a. Example of 2SPP
y
3
2
1
0
1
2
3
x
r2
r1
r3
r4
b. One placement of 2SPP

Variables

$px_{1,0}, \ldots, px_{1,3} \quad py_{1,0}, \ldots, py_{1,3} \quad px_{2,0}, \ldots, px_{2,2} \quad py_{2,0}, \ldots, py_{2,3}$
$px_{3,0}, \ldots, px_{3,3} \quad py_{3,0}, \ldots, py_{3,2} \quad px_{4,0}, \ldots, px_{4,3} \quad py_{4,0}, py_{4,1}$

Order Constraint (1)

$$\neg px_{1,0} \lor px_{1,1}, \neg px_{1,1} \lor px_{1,2}, \neg px_{1,2} \lor px_{1,3}$$
$$\vdots$$
$$\neg py_{4,0} \lor py_{4,1}, \neg py_{4,1} \lor py_{4,2}, \neg py_{4,2} \lor py_{4,3}$$

Non-overlapping Constraint (2), (3)

$$lr_{1,2} \lor lr_{2,1} \lor ud_{1,2} \lor ud_{2,1}$$
$$\vdots$$
$$lr_{3,4} \lor lr_{4,3} \lor ud_{3,4} \lor ud_{4,3}$$

$$\neg lr_{1,2} \lor \neg px_{2,0} \quad \neg lr_{1,2} \lor px_{1,0} \lor \neg px_{2,1} \quad \neg lr_{1,2} \lor px_{1,1} \lor \neg px_{2,2} \quad \neg lr_{1,2} \lor px_{1,2}$$
$$\vdots$$
$$\neg ud_{3,4} \lor \neg py_{3,0} \quad \neg ud_{3,4} \lor py_{4,0} \lor \neg py_{3,1} \quad \neg ud_{3,4} \lor py_{4,1} \lor \neg py_{3,2} \quad \neg ud_{3,4} \lor py_{4,2}$$

Solving 2SPP with a SAT solver

Solving 2SPP with a SAT solver

- Let ub be *upper bound* of a solution of a 2SPP
- Let lb be *lower bound* of a solution of a 2SPP
- Let o be an integer value such that $lb \le o \le ub - 1$
- Let ph_o be a Boolean variable which is $true$ if all rectangles are packed at the downward to the height o.

For each rectangle i, and a height o such that $lb \le o \le ub - 1$, we have the 2-literal clauses:

$$\neg ph_o \vee py_{i,o-h_i} \tag{4}$$

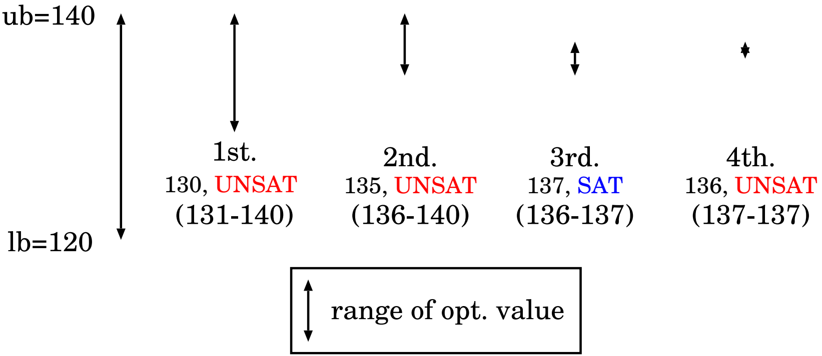
For each o $(lb \le o \le ub - 1)$, we have the 2-literal clauses due to order encoding:

$$\neg ph_o \vee ph_{o+1} \tag{5}$$

Solving 2SPP with a SAT solver

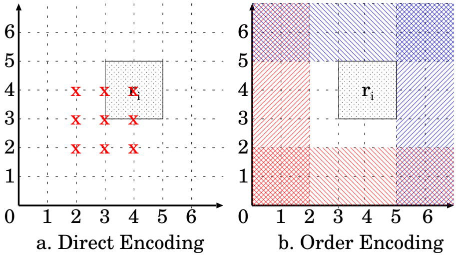
Let Ψ be the set of clauses consisting of all clauses obtained from (1), (2), (3), (4) and (5). Then, we can decide the satisfiability of a 2OPP with the height H by solving the set of clauses:

$$\Psi \cup \{ph_H\} \tag{6}$$



Order Encoding and Direct Encoding

In our method, we use the order encoding as a SAT encoding. However, there have been well studied about SAT encoding. Here, we consider a difference between direct encoding and order encoding.



Let $(w_i, h_i) = (2, 2), (w_j, h_j) = (2, 2)$ and place r_i at $(x_i, y_i) = (3, 3)$. We can represent overlap constraint of CSP between r_i and r_j as follows:

$$(x_j \leq 1) \lor (x_j \geq 5) \lor (y_j \leq 1) \lor (y_j \geq 5)$$

To see difference between order encoding and the others, we compare the SAT clauses encoded with direct encoding [Walsh 2000] and those with order encoding. In the direct encoding, we assign to a SAT variable as $p_{xa} = true$ if and only if the CSP variable x has the domain value a, and constraints are encoded to conflict clauses. The encoded clauses are as follows:

$$
\begin{array}{rl}
CSP: & (x_j \le 1) \lor (x_j \ge 5) \lor (y_j \le 1) \lor (y_j \ge 5) \\
SAT(direct): & \neg p_{x_j 2} \lor \neg p_{y_j 2} \quad \neg p_{x_j 2} \lor \neg p_{y_j 3} \quad \neg p_{x_j 2} \lor \neg p_{y_j 4} \\
 & \neg p_{x_j 3} \lor \neg p_{y_j 2} \quad \neg p_{x_j 3} \lor \neg p_{y_j 3} \quad \neg p_{x_j 3} \lor \neg p_{y_j 4} \\
 & \neg p_{x_j 4} \lor \neg p_{y_j 2} \quad \neg p_{x_j 4} \lor \neg p_{y_j 3} \quad \neg p_{x_j 4} \lor \neg p_{y_j 4} \\
SAT(order): & p_{x_j 1} \lor \neg p_{x_j 4} \lor p_{y_j 1} \lor \neg p_{y_j 4}
\end{array}
$$

In direct encoding, constraints are represented as conflict points. On the other hand, order encoding represents constraints as a conflict region. This indicates SAT-based approach with order encoding is suitable not only 2SPP but also geometric problems such as shop scheduling problem.