SAT 型制約ソルバー Sugar について

田村直之　　丹生智也　　番原睦則

神戸大学

第 81 回 人工知能基本問題研究会
2011 年 1 月 31 日

内容

内容

1. SAT と SAT ソルバー
2. 制約充足問題 (CSP) の SAT 符号化
3. SAT 型制約ソルバー Sugar

内容

1. SAT と SAT ソルバー
2. 制約充足問題 (CSP) の SAT 符号化
3. SAT 型制約ソルバー Sugar
4. 制約充足問題の解法例
   - オープンショップ・スケジューリング (OSS) 問題
   - ラテン方陣問題
5. デモ
   - 巨大な数独
   - ノノグラム

SAT と SAT ソルバー

SAT

SAT

SAT (Boolean satisfiability testing) **は，与えられた命題論理式を真にする値割当てが存在するか否かを判定する問題である．**

- SAT **は，理論上も実際上も計算機科学の中心的課題である．**
- SAT **は** NP-**完全であることが最初に証明された問題である** [Cook 1971]**．**
- MiniSat **等，非常に効率の良い** SAT **ソルバーが実装されている．**
- SAT **ソルバーをエンジンとした** SAT **型システムが多くの分野で利用されるようになってきた．**

SAT 問題 (SAT instances)

具体的な SAT **の問題は，連言標準形** (CNF) **で与えられる．**

CNF 式

- CNF **式は，複数の節の連言である．**
- **節** (clause) **は複数のリテラルの選言である．**
- **リテラル** (literal) **は，命題変数かあるいはその否定である．**

標準的フォーマットとしては DIMACS CNF **が用いられる．**

```
p cnf 9 7       ; Number of variables and clauses
1 2 0           ; a ∨ b
9 3 0           ; c ∨ d
1 8 4 0         ; a ∨ e ∨ f
-2 -4 5 0       ; ¬b ∨ ¬f ∨ g
-4 6 0          ; ¬f ∨ h
-2 -6 7 0       ; ¬b ∨ ¬h ∨ i
-5 -7 0         ; ¬g ∨ ¬i
```

SAT ソルバー (SAT solvers)

SAT ソルバー

SAT ソルバーは，与えられた SAT 問題が充足可能 (SAT) か充足不能 (UNSAT) かを判定するプログラムである．

通常，充足可能であればその値割当てを解として出力する．

- 系統的 SAT ソルバーは，SAT あるいは UNSAT を判定する．
  - ほとんどは DPLL アルゴリズムを用いている．
- 確率的 SAT ソルバーは，SAT のみを判定する (UNSAT は判定できない)．
  - ローカルサーチアルゴリズムが用いられる．

DPLL アルゴリズム (DPLL algorithm)

[Davis & Putnam 1960], [Davis, Logemann & Loveland 1962]

```
(1)  function DPLL(S: a CNF formula, σ: a variable assignment)
(2)    σ := UP(S, σ); /* unit propagation */
(3)    if S is satisfied by σ then return true;
(4)    if S is falsified by σ then return false;
(5)    choose an unassigned variable x from Sσ;
(6)    return DPLL(S, σ ∪ {x ↦ 0}) or DPLL(S, σ ∪ {x ↦ 1});

(1)  function UP(S: a CNF formula, σ: a variable assignment)
(2)    while Sσ contains a unit clause {l} do
(3)      if l is positive then σ := σ ∪ {l ↦ 1};
(4)                       else σ := σ ∪ {l̄ ↦ 0};
(5)    return σ;
```

- **$S\sigma$ は，値割当て σ を S に適用した CNF 式を表す．**

DPLL アルゴリズム (DPLL algorithm)

1. a を選択し $a \mapsto 0$ と決定

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定
   - C_2 より $d \mapsto 1$ が伝播

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定
   - C_2 より $d \mapsto 1$ が伝播
3. e を選択し $e \mapsto 0$ と決定

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定
   - C_2 より $d \mapsto 1$ が伝播
3. e を選択し $e \mapsto 0$ と決定
   - C_3 より $f \mapsto 1$ が伝播

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定
   - C_2 より $d \mapsto 1$ が伝播
3. e を選択し $e \mapsto 0$ と決定
   - C_3 より $f \mapsto 1$ が伝播
   - C_4 より $g \mapsto 1$ が伝播

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定
   - C_2 より $d \mapsto 1$ が伝播
3. e を選択し $e \mapsto 0$ と決定
   - C_3 より $f \mapsto 1$ が伝播
   - C_4 より $g \mapsto 1$ が伝播
   - C_7 より $i \mapsto 0$ が伝播

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定
   - C_2 より $d \mapsto 1$ が伝播
3. e を選択し $e \mapsto 0$ と決定
   - C_3 より $f \mapsto 1$ が伝播
   - C_4 より $g \mapsto 1$ が伝播
   - C_7 より $i \mapsto 0$ が伝播
   - C_5 より $h \mapsto 1$ が伝播

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定
   - C_2 より $d \mapsto 1$ が伝播
3. e を選択し $e \mapsto 0$ と決定
   - C_3 より $f \mapsto 1$ が伝播
   - C_4 より $g \mapsto 1$ が伝播
   - C_7 より $i \mapsto 0$ が伝播
   - C_5 より $h \mapsto 1$ が伝播
   - C_6 で矛盾

DPLL アルゴリズム (DPLL algorithm)

$$C_1 : a \vee b$$
$$C_2 : c \vee d$$
$$C_3 : a \vee e \vee f$$
$$C_4 : \neg b \vee \neg f \vee g$$
$$C_5 : \neg f \vee h$$
$$C_6 : \neg b \vee \neg h \vee i$$
$$C_7 : \neg g \vee \neg i$$

1. a を選択し $a \mapsto 0$ と決定
   - C_1 より $b \mapsto 1$ が伝播
2. c を選択し $c \mapsto 0$ と決定
   - C_2 より $d \mapsto 1$ が伝播
3. e を選択し $e \mapsto 0$ と決定
   - C_3 より $f \mapsto 1$ が伝播
   - C_4 より $g \mapsto 1$ が伝播
   - C_7 より $i \mapsto 0$ が伝播
   - C_5 より $h \mapsto 1$ が伝播
   - C_6 で矛盾
4. バックトラックし $e \mapsto 1$ と決定

近代的 SAT ソルバー (Modern SAT solvers)

- **近代的** SAT **ソルバーでは，**DPLL **に以下の技術が導入され大幅な性能向上が実現されている．**
  - **CDCL** (Conflict Driven Clause Learning) [Silva 1996]
  - **非時間順バックトラック法** [Silva 1996]
  - **ランダムリスタート** [Gomes 1998]
  - **監視リテラル** [Moskewicz & Zhang 2001]
  - **変数選択ヒューリスティック** [Moskewicz & Zhang 2001]
- Chaff **と** zChaff **は十倍から百倍の高速化を実現した．**
- 2002 **年以降** SAT **競技会が開催され，**SAT **ソルバーの実装技術の向上に貢献している．**
- MiniSat **は** C++**で約** 2000 **行というコンパクトなサイズにもかかわらず，**2005 **年の** SAT **競技会で優れた性能を示した．**
- **近代的** SAT **ソルバーは，命題変数が** 10^6 **で節が** 10^7 **以上の** SAT **問題も取り扱うことができる．**

CDCL (Conflict Driven Clause Learning)

- **矛盾が発生すると，矛盾の原因が解析され節として抽出される．抽出された節は学習節** (learnt clause) **として記録される．**
- **学習節はその後の探索の探索空間を大幅に枝刈りする．**
- **学習節は，逆向きに導出を進めることで生成される．**
- **導出は，**First UIP (Unique Implication Point) **まで行う** [Moskewicz & Zhang 2001]**．**

前の例では $\neg b \vee \neg f$ **が学習節として生成される．**

$$\dfrac{\dfrac{\dfrac{C_6 : \neg b \vee \neg h \vee i \quad C_5 : \neg f \vee h}{\neg b \vee \neg f \vee i} \qquad C_7 : \neg g \vee \neg i}{\neg b \vee \neg f \vee \neg g} \qquad C_4 : \neg b \vee \neg f \vee g}{\neg b \vee \neg f}$$

SAT 型システム (SAT-based systems)

SAT **型システムが求解困難な組合せ問題を解くために利用されるようになっている．**

- **プランニング** (SATPLAN, Blackbox) [Kautz & Selman 1992]
- **自動テストパターン生成** [Larrabee 1992]
- **ジョブショップスケジューリング** [Crawford & Baker 1994]
- **ソフトウェア検証** (Alloy) [1998]
- **有界モデル検査** [Biere 1999]
- **ソフトウェアパッケージの依存解析** (SATURN)
  - SAT4J **は** Eclipse 3.4 **に利用されている．**
- **書換えシステム** (AProVE, Jambox)
- **解集合プログラミング** (clasp, Cmodels-2)
- **一階論理定理証明** (iProver, Darwin)
- **一階モデル発見** (Paradox)
- **制約充足問題** (Sugar) [Tamura et al. 2006]

なぜ SAT 型システムか? (個人的意見)

SAT ソルバーは非常に速い.

- 監視リテラル等の優れた実装技術
  - バックトラックに必要な情報の保存を最小にしている.
- キャシュを意識した実装 [Zhang & Malik 2003]
  - たとえば，オープンショップスケジューリング問題の gp10-10 を SAT 符号化したものは，MiniSat で 4 秒以内で求解でき L2 キャッシュへのヒットレートは 99%以上である.

```
$ valgrind --tool=cachegrind minisat gp10-10-1091.cnf
L2 refs:       42,842,531  ( 31,633,380 rd +11,209,151 wr)
L2 misses:     25,674,308  ( 19,729,255 rd + 5,945,053 wr)
L2 miss rate:         0.4% (        0.4%   +       1.0%  )
```

なぜ SAT 型システムか? (個人的意見)

SAT ソルバーをエンジンとして用いるアプローチは，1980 年代に Patterson らが提唱した RISC アプローチに似ている．

- **RISC**: Reduced Instruction Set Computer
- Patterson は “reduced” かつ高速な命令セットの計算機のほうが，“complex” な命令セットの計算機 (**CISC**) よりも高速になると主張した．

SAT ソルバー $\Longleftrightarrow$ RISC
SAT 符号化 $\Longleftrightarrow$ 最適化コンパイラ

- この意味で，SAT ソルバーと SAT 符号化の双方が重要な研究テーマといえる．

制約充足問題の SAT 符号化

Finite linear CSP

Finite linear CSP

- **有限ドメインの整数変数**
  - $\ell(x)$: x **の下限**
  - $u(x)$: x **の上限**
- **命題変数**
- **算術演算**: $+$, $-$, **定数倍**, etc.
- **比較演算**: $=$, $\neq$, $\geq$, $>$, $\leq$, $<$
- **論理演算**: $\neg$, $\wedge$, $\vee$, $\Rightarrow$

- **一般性を失うことなしに比較を** $\sum a_i x_i \leq c$ **の形に制限できる．ここで** x_i **は整数変数で，** a_i **と** c **は整数定数である．**

SAT 符号化 (SAT encodings)

CSP の SAT 符号化については，種々の方法が提案されている．

- **直接符号化** (Direct encoding) **は，最も広く用いられている方法である** [de Kleer 1989]．
- **順序符号化** (Order encoding) **は，多くの問題に対して優れた性能を示す新しい方法である** [Tamura et al. 2006]．
  - **順序符号化はジョブショップ・スケジューリング問題に対して最初に用いられた** [Crawford & Baker 1994]．
  - **オープンショップ・スケジューリング，ジョブショップ・スケジューリング，2 次元ストリップパッキング等について，未解決だった問題の解決に成功している．**
- **他の符号化**:
  - **多値符号化** (Multivalued encoding) [Selman 1992]
  - **支持符号化** (Support encoding) [Kasif 1990]
  - **対数符号化** (Log encoding) [Iwama 1994]
  - **対数支持符号化** (Log-support encoding) [Gavanelli 2007]
  - **コンパクト順序符号化** (Compact order encloding) [Tanjo et al. 2010]

直接符号化 (Direct encoding)

直接符号化では，各整数変数 x とそのドメインの各値 i に対して，$x = i$ を表す命題変数 $p(x = i)$ を用いる [de Kleer 1989]**.**

各整数変数 x に対して用いる命題変数

$$p(x = i) \qquad (\ell(x) \leq i \leq u(x))$$

たとえば整数変数 $x \in \{2, 3, 4, 5, 6\}$ に対しては，以下の 5 つの命題変数を用いる.

$x \in \{2, 3, 4, 5, 6\}$ に対する 5 つの命題変数

$$p(x = 2) \quad p(x = 3) \quad p(x = 4) \quad p(x = 5) \quad p(x = 6)$$

直接符号化 (続き)

命題変数 $p(x = i)$ が，$x = i$ の時かつその時に限って真となるように，以下の at-least-one **節と** at-most-one **節を追加する．**

各整数変数 x に対して追加する節

$$p(x = \ell(x)) \vee \cdots \vee p(x = u(x))$$
$$\neg p(x = i) \vee \neg p(x = j) \qquad (\ell(x) \leq i < j \leq u(x))$$

たとえば $x \in \{2, 3, 4, 5, 6\}$ に対しては，以下の 11 節を追加する．

$x \in \{2, 3, 4, 5, 6\}$ に対して追加する 11 節

$$p(x = 2) \vee p(x = 3) \vee p(x = 4) \vee p(x = 5) \vee p(x = 6)$$
$$\neg p(x = 2) \vee \neg p(x = 3) \qquad \neg p(x = 2) \vee \neg p(x = 4) \qquad \neg p(x = 2) \vee \neg p(x = 5)$$
$$\neg p(x = 2) \vee \neg p(x = 6) \qquad \neg p(x = 3) \vee \neg p(x = 4) \qquad \neg p(x = 3) \vee \neg p(x = 5)$$
$$\neg p(x = 3) \vee \neg p(x = 6) \qquad \neg p(x = 4) \vee \neg p(x = 5)$$
$$\neg p(x = 4) \vee \neg p(x = 6) \qquad \neg p(x = 5) \vee \neg p(x = 6)$$

直接符号化 (続き)

制約は違反点 (conflict points) を列挙することで符号化できる．

制約節

- $x_1 = i_1, \ldots, x_k = i_k$ **が制約に違反する時，以下の節を追加する．**

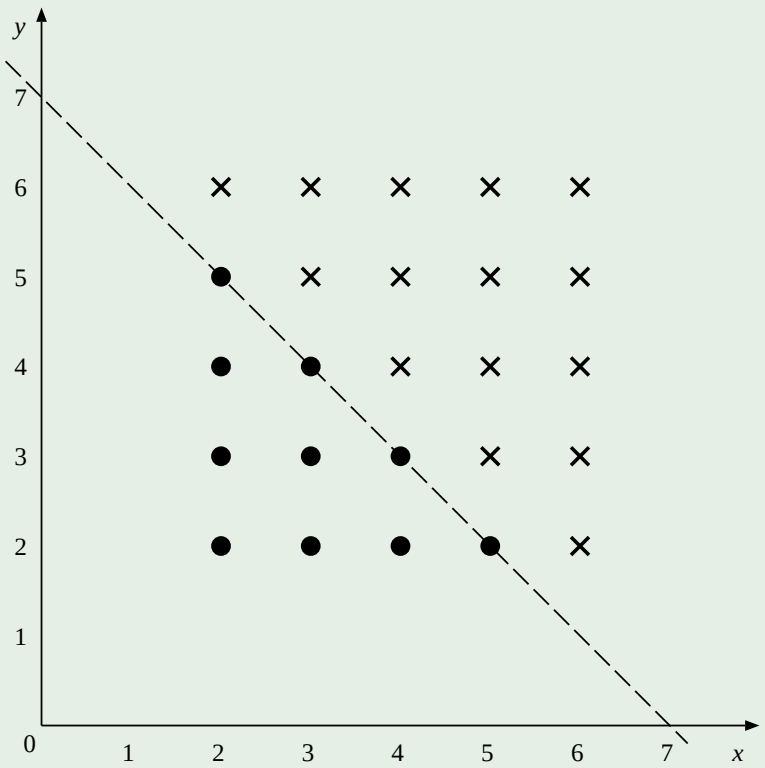
$$\neg p(x_1 = i_1) \vee \cdots \vee \neg p(x_k = i_k)$$

直接符号化 (続き)

制約 $x + y \leq 7$ は，違反点 (図中の $\times$ 点) を列挙することで以下の 15 節に符号化される．

$x + y \leq 7$ の直接符号化

$$
\begin{aligned}
&\neg p(x = 2) \vee \neg p(y = 6)\\
&\neg p(x = 3) \vee \neg p(y = 5)\\
&\neg p(x = 3) \vee \neg p(y = 6)\\
&\neg p(x = 4) \vee \neg p(y = 4)\\
&\neg p(x = 4) \vee \neg p(y = 5)\\
&\neg p(x = 4) \vee \neg p(y = 6)\\
&\neg p(x = 5) \vee \neg p(y = 3)\\
&\neg p(x = 5) \vee \neg p(y = 4)\\
&\neg p(x = 5) \vee \neg p(y = 5)\\
&\neg p(x = 5) \vee \neg p(y = 6)\\
&\neg p(x = 6) \vee \neg p(y = 2)\\
&\neg p(x = 6) \vee \neg p(y = 3)\\
&\neg p(x = 6) \vee \neg p(y = 4)\\
&\neg p(x = 6) \vee \neg p(y = 5)\\
&\neg p(x = 6) \vee \neg p(y = 6)
\end{aligned}
$$



順序符号化 (Order encoding)

順序符号化では，各整数変数 x とそのドメインの各値 i に対して，$x \leq i$ を表す命題変数 $p(x \leq i)$ を用いる [Tamura et al. 2006]**.**

各整数変数 x に対して用いる命題変数

$$p(x \leq i) \qquad (\ell(x) \leq i < u(x))$$

たとえば整数変数 $x \in \{2, 3, 4, 5, 6\}$ に対しては，以下の 4 つの命題変数を用いる.

4 Boolean variables for $x \in \{2, 3, 4, 5, 6\}$

$$p(x \leq 2) \quad p(x \leq 3) \quad p(x \leq 4) \quad p(x \leq 5)$$

$x \leq 6$ が常に真のため命題変数 $p(x \leq 6)$ は不要である.

順序符号化 (続き)

命題変数 $p(x \leq i)$ が，$x \leq i$ の時かつその時に限って真となるように，以下の節を追加する．

各整数変数 x に対して追加する節

$$\neg p(x \leq i-1) \vee p(x \leq i) \quad (\ell(x) < i < u(x))$$

たとえば $x \in \{2,3,4,5,6\}$ に対しては，以下の 3 節を追加する．

$x \in \{2,3,4,5,6\}$ に対して追加する 11 節

$$\neg p(x \leq 2) \vee p(x \leq 3)$$
$$\neg p(x \leq 3) \vee p(x \leq 4)$$
$$\neg p(x \leq 4) \vee p(x \leq 5)$$

順序符号化 (続き)

追加された節を充足する可能な値割当ては以下の通りである.

$$\neg p(x \leq 2) \vee p(x \leq 3)$$
$$\neg p(x \leq 3) \vee p(x \leq 4)$$
$$\neg p(x \leq 4) \vee p(x \leq 5)$$

充足する値割当て

$p(x \leq 2)$	$p(x \leq 3)$	$p(x \leq 4)$	$p(x \leq 5)$	**解釈**
1	1	1	1	$x = 2$
0	1	1	1	$x = 3$
0	0	1	1	$x = 4$
0	0	0	1	$x = 5$
0	0	0	0	$x = 6$

順序符号化 (続き)

充足する部分値割当て

$p(x \leq 2)$	$p(x \leq 3)$	$p(x \leq 4)$	$p(x \leq 5)$	解釈
–	–	–	–	$x = 2..6$
–	–	–	1	$x = 2..5$
–	–	1	1	$x = 2..4$
–	1	1	1	$x = 2..3$
0	–	–	–	$x = 3..6$
0	0	–	–	$x = 4..6$
0	0	0	–	$x = 5..6$
0	–	–	1	$x = 3..5$
0	–	1	1	$x = 3..4$
0	0	–	1	$x = 4..5$

“–” は未定義を表す．

- 命題変数に対する部分値割当ては，整数変数の範囲を表している．

順序符号化 (続き)

制約は，違反点ではなく違反領域 (conflict region) を列挙することで符号化できる．

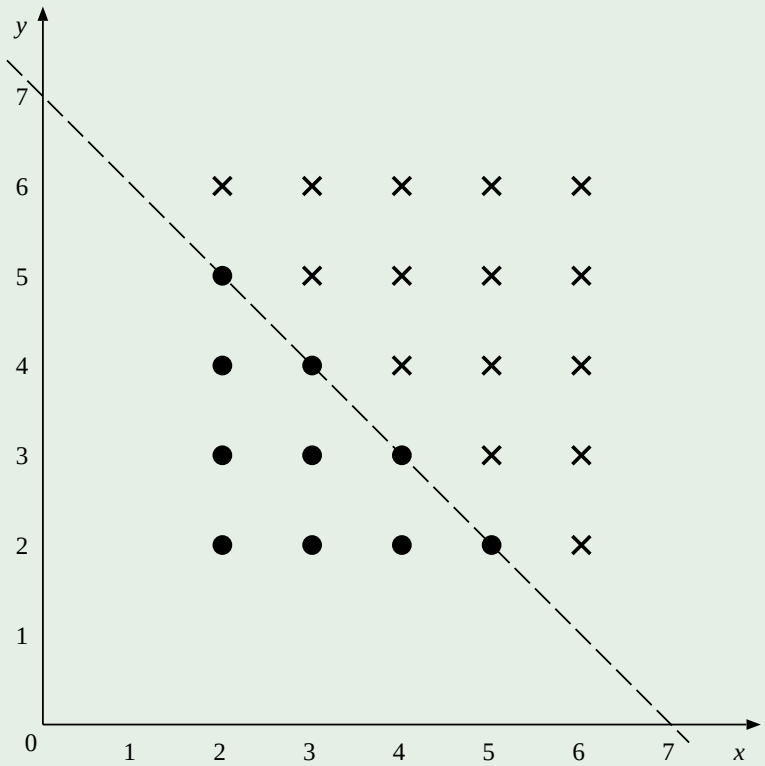
制約節

- **範囲 $i_1 < x_1 \leq j_1,\ \ldots,\ i_k < x_k \leq j_k$ 中のすべての点 $(x_1, \ldots, x_k)$ が制約に違反する時，以下の節を追加する．**

$$p(x_1 \leq i_1) \vee \neg p(x_1 \leq j_1) \vee \cdots \vee p(x_k \leq i_k) \vee \neg p(x_k \leq j_k)$$

順序符号化 (続き)

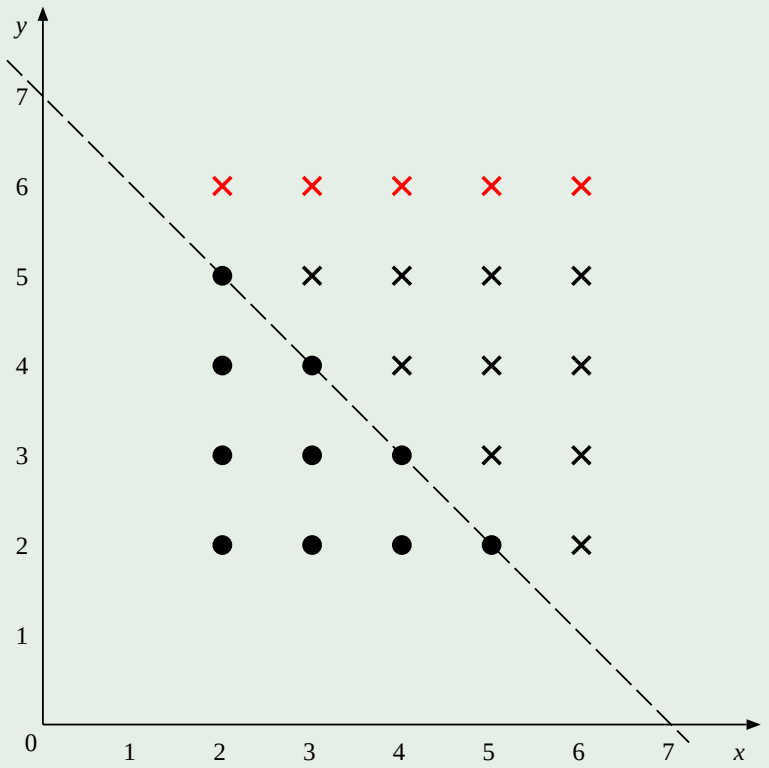
$x + y \leq 7$ の順序符号化



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

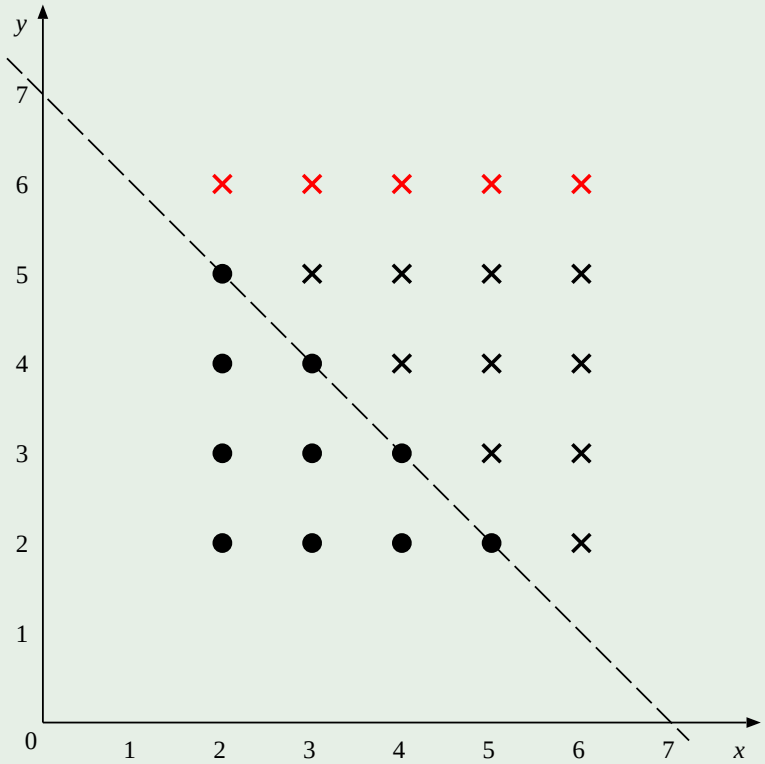
$\neg(y \geq 6)$



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

$p(y \leq 5)$

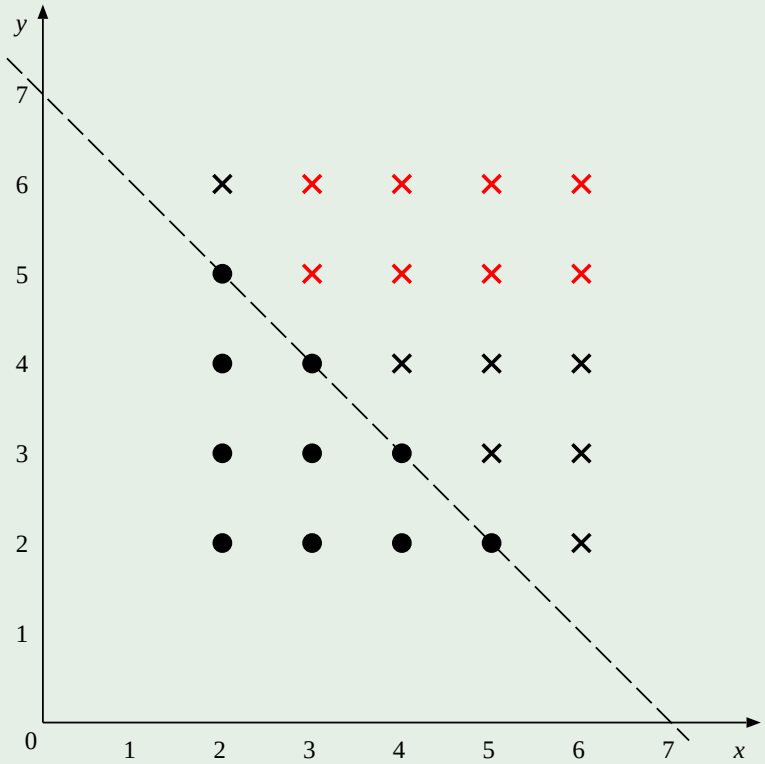


順序符号化 (続き)

$x + y \leq 7$ の順序符号化

$p(y \leq 5)$

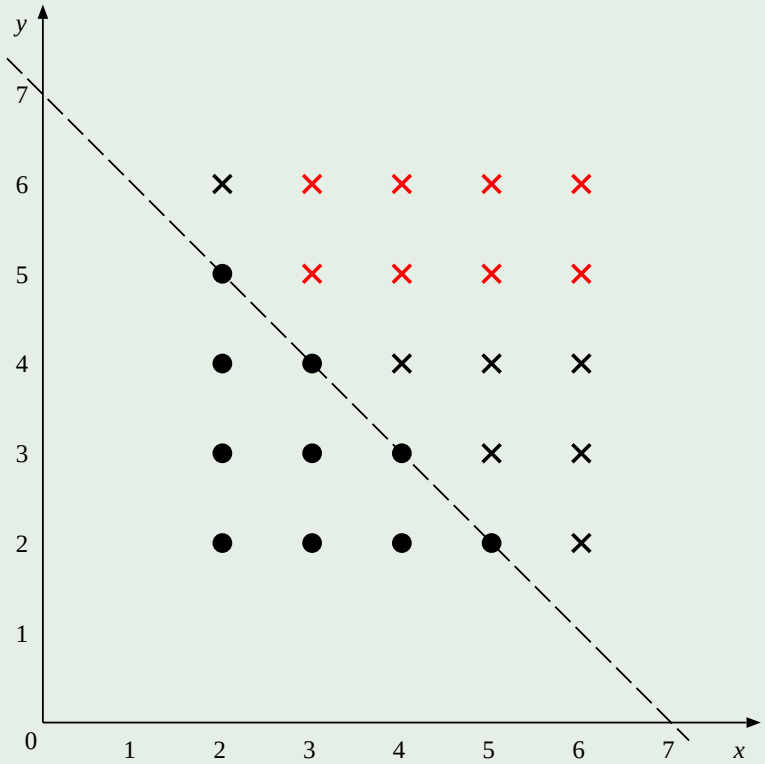
$\neg(x \geq 3 \wedge y \geq 5)$



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

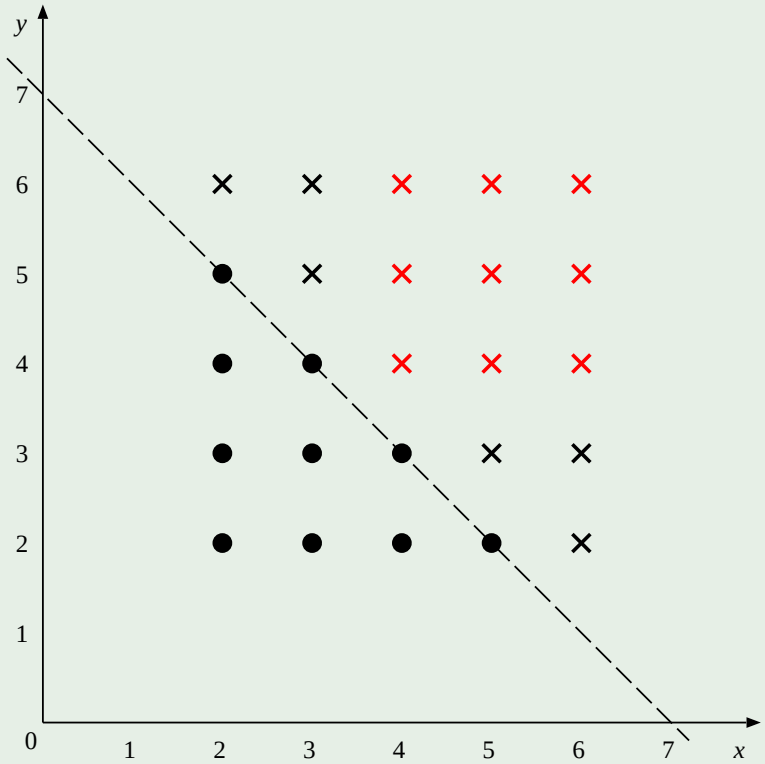
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

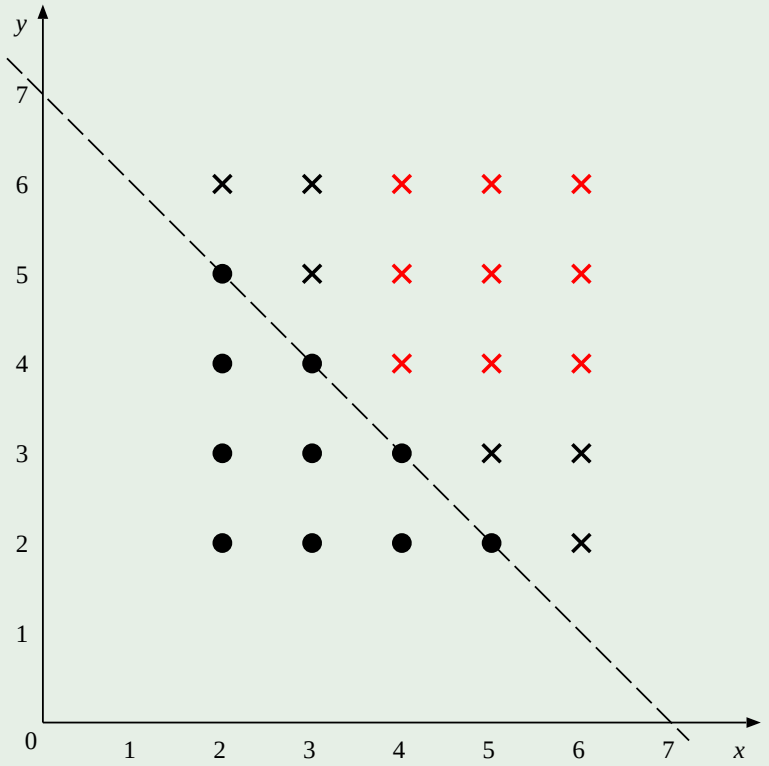
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$
$\neg(x \geq 4 \wedge y \geq 4)$



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

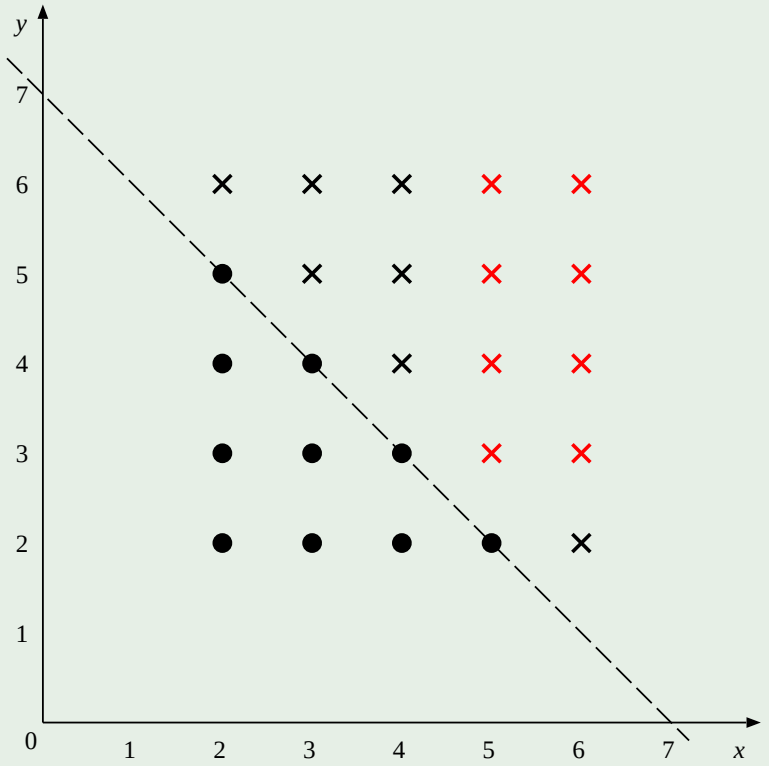
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$
$p(x \leq 3) \vee p(y \leq 3)$



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

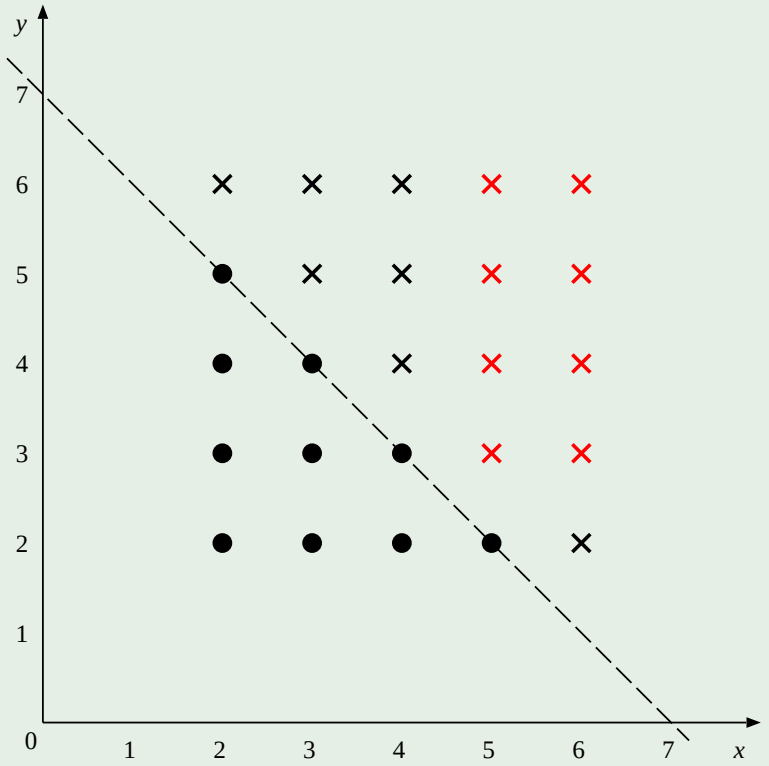
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$
$p(x \leq 3) \vee p(y \leq 3)$
$\neg(x \geq 5 \wedge y \geq 3)$



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

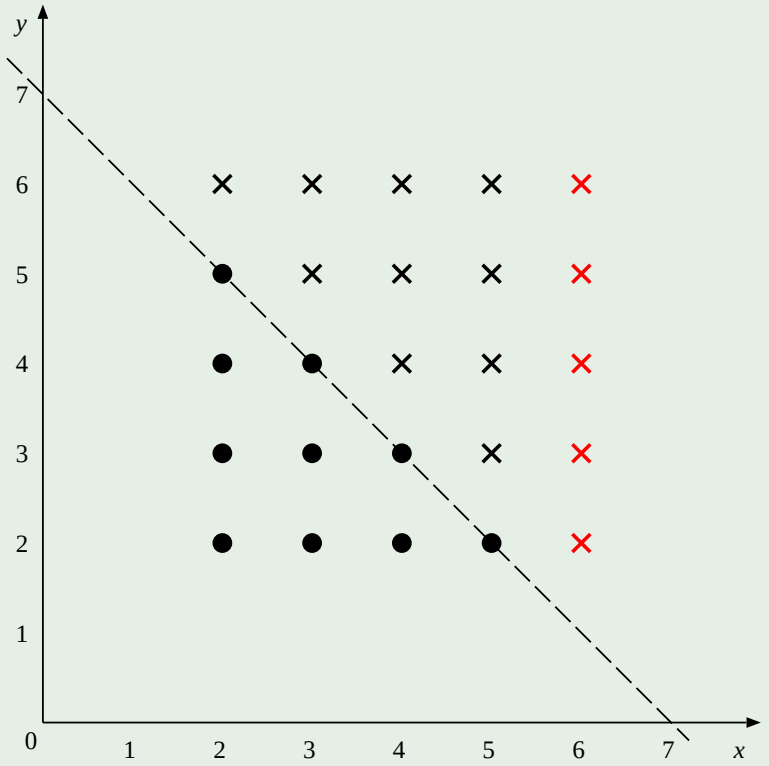
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$
$p(x \leq 3) \vee p(y \leq 3)$
$p(x \leq 4) \vee p(y \leq 2)$



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

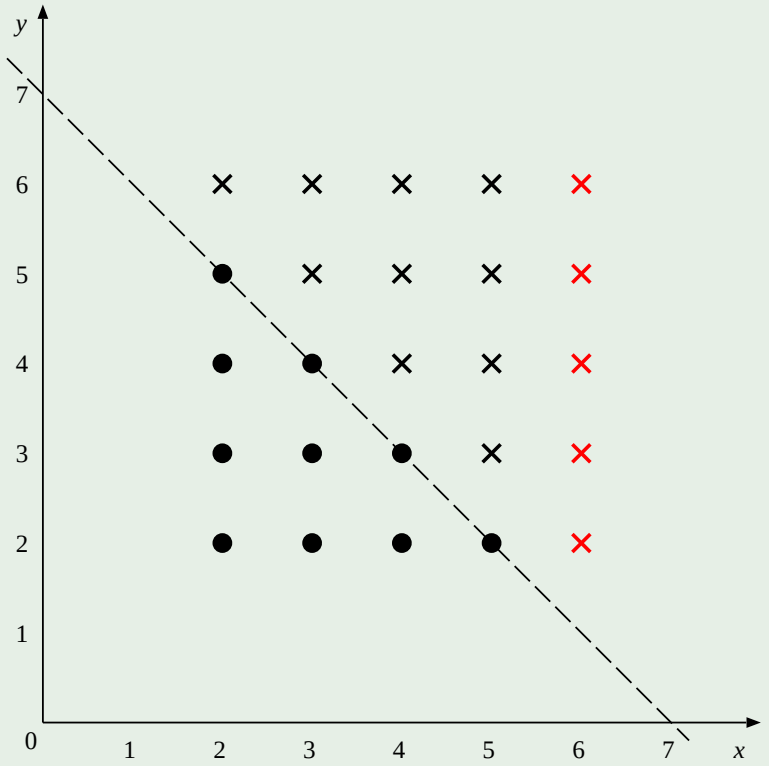
$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$
$p(x \leq 3) \vee p(y \leq 3)$
$p(x \leq 4) \vee p(y \leq 2)$
$\neg(x \geq 6)$



順序符号化 (続き)

$x + y \leq 7$ の順序符号化

$p(y \leq 5)$
$p(x \leq 2) \vee p(y \leq 4)$
$p(x \leq 3) \vee p(y \leq 3)$
$p(x \leq 4) \vee p(y \leq 2)$
$p(x \leq 5)$



順序符号化における範囲伝播

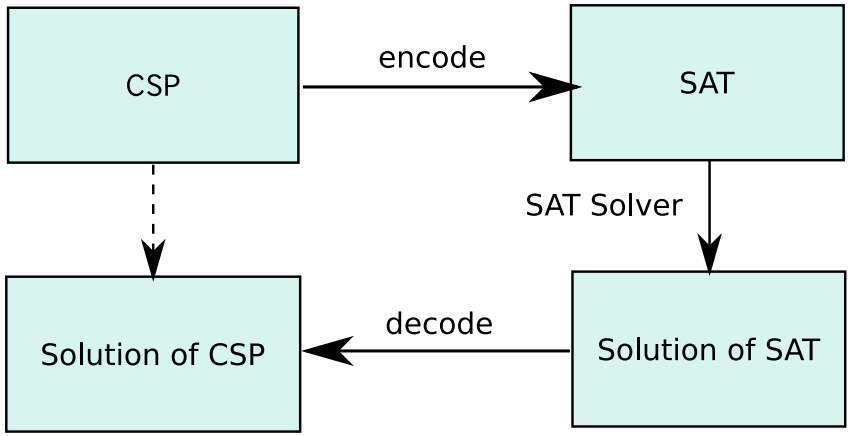
$x + y \leq 7$ の順序符号化

$$
\begin{aligned}
C_1 &: \quad p(y \leq 5) \\
C_2 &: \quad p(x \leq 2) \vee p(y \leq 4) \\
C_3 &: \quad p(x \leq 3) \vee p(y \leq 3) \\
C_4 &: \quad p(x \leq 4) \vee p(y \leq 2) \\
C_5 &: \quad p(x \leq 5)
\end{aligned}
$$

- $p(x \leq 3)$ **が偽 (すなわち** $x \geq 4$**) となれば，**C_3 **に対する単位伝播** (unit propagation) **により** $p(y \leq 3)$ **が真 (すなわち** $y \leq 3$**) となる．**
- **これは制約ソルバーにおける範囲伝播** (bound propagation) **に対応している．**

SAT 型制約ソルバー Sugar

Sugar: SAT 型制約ソルバー



- **Sugar は順序符号化を用いた SAT 型の制約ソルバーである.**
- **2008 年 CSP ソルバー競技会の GLOBAL 部門で優勝.**
- **2008 年 Max-CSP ソルバー競技会の 3 部門で優勝.**
- **2009 年 CSP ソルバー競技会の 3 部門で優勝.**

Sugar の構成

- **Java プログラム**
  - Parser
  - Linearizer
  - Simplifier: **一般アーク整合性** (GAC) **アルゴリズムで変数のドメインを削除**
  - Encoder: **順序符号化に基づく**
  - Decoder
- **SAT ソルバー**
  - MiniSat (default), PicoSAT, **その他**
- **Perl スクリプト**
  - **コマンドライン・スクリプト**

制約の変換

- 線形の制約は順序符号化で変換する．
- 以下のように非線形の制約を変換する．

式	変換
$E = F$	$(E \leq F) \wedge (E \geq F)$
$E \neq F$	$(E < F) \vee (E > F)$
$\max(E, F)$	x
	with $(x \geq E) \wedge (x \geq F) \wedge ((x \leq E) \vee (x \leq F))$
$\min(E, F)$	x
	with $(x \leq E) \wedge (x \leq F) \wedge ((x \geq E) \vee (x \geq F))$
$\mathrm{abs}(E)$	$\max(E, -E)$
E div c	q
	with $(E = c\,q + r) \wedge (0 \leq r) \wedge (r < c)$
E mod c	r
	with $(E = c\,q + r) \wedge (0 \leq r) \wedge (r < c)$

グローバル制約の変換

- alldifferent$(x_1, x_2, \ldots, x_n)$ **制約は以下のように変換される．**

$$\bigwedge_{i<j}(x_i \neq x_j)$$
$$\bigvee_{i}(x_i \geq lb + n - 1)$$
$$\bigvee_{i}(x_i \leq ub - n + 1)$$

最後の 2 **つは鳩の巣原理の節であり，** lb **と** ub **は** $\{x_1, x_2, \ldots, x_n\}$ **の下限と上限である．**

- **他のグローバル制約** (element, weightedsum, cumulative **など) は，定義通りに変換される．**

Sugar で CSP を解く例

- **オープンショップ・スケジューリング** (OSS) **問題**
- **ラテン方陣問題**

オープンショップ・スケジューリング (OSS) 問題

- OSS は n 個のジョブと n 個のマシンから成る．
  - $J_0, J_1, \ldots, J_{n-1}$
  - $M_0, M_1, \ldots, M_{n-1}$
- 各ジョブ J_i は n 個のオペレーションから成る．
  - $O_{i0}, O_{i1}, \ldots, O_{i(n-1)}$
- ジョブ J_i のオペレーション O_{ij} は，マシン M_j で処理され，正の処理時間 p_{ij} を要する．
- 同じジョブに対するオペレーションは同時には処理できない．
- 各マシン M_j は複数のオペレーションを同時には処理できない．

OSS 問題の目的

- すべてのジョブを完了するのに必要な時間 *makespan* を最小にする．

- OSS の可能なスケジューリングの組合せは膨大である．n ジョブ n マシンの OSS は，$(n!)^{2n}$ 個の実行可能解を持つ．

OSS 問題 gp03-01

$$(p_{ij}) = \begin{pmatrix} 661 & 6 & 333 \\ 168 & 489 & 343 \\ 171 & 505 & 324 \end{pmatrix}$$

gp03-01 の最適解 (makespan=1168)

gp03-01 の制約モデル

整数変数の定義

- m: makespan
- s_{ij}: **オペレーション O_{ij} の開始時刻**

制約の定義

- **各 s_{ij} に対して**

$$s_{ij} + p_{ij} \leq m$$

- **同一ジョブ J_i の各オペレーション O_{ij}, O_{il} に対して**

$$(s_{ij} + p_{ij} \leq s_{il}) \vee (s_{il} + p_{il} \leq s_{ij})$$

- **同一マシン M_j の各オペレーション O_{ij}, O_{kj} に対して**

$$(s_{ij} + p_{ij} \leq s_{kj}) \vee (s_{kj} + p_{kj} \leq s_{ij})$$

gp03-01 の制約モデル

gp03-01 の CSP 表現

$$s_{00} + 661 \leq m$$
$$s_{01} + 6 \leq m$$
$$s_{02} + 333 \leq m$$
$$\cdots\cdots\cdots$$
$$s_{22} + 324 \leq m$$
$$(s_{00} + 661 \leq s_{01}) \vee (s_{01} + 6 \leq s_{00})$$
$$(s_{00} + 661 \leq s_{02}) \vee (s_{02} + 333 \leq s_{00})$$
$$(s_{01} + 6 \leq s_{02}) \vee (s_{02} + 333 \leq s_{01})$$
$$\cdots\cdots\cdots$$
$$(s_{02} + 333 \leq s_{12}) \vee (s_{12} + 343 \leq s_{02})$$
$$(s_{02} + 333 \leq s_{22}) \vee (s_{22} + 324 \leq s_{02})$$
$$(s_{12} + 343 \leq s_{22}) \vee (s_{22} + 324 \leq s_{12})$$

gp03-01 を Sugar で解く

充足可能の場合 ($m \leq 1168$)

gp03-01 を Sugar で解く

充足可能の場合 ($m \leq 1168$)

- MiniSat が解を求めるまでの処理
  - 12 decisions
  - 1 conflict (backtrack).

gp03-01 を Sugar で解く

充足可能の場合 ($m \leq 1168$)

- MiniSat が解を求めるまでの処理
  - 12 decisions
  - 1 conflict (backtrack).

充足不能の場合 ($m \leq 1167$)

gp03-01 を Sugar で解く

充足可能の場合 ($m \leq 1168$)

- MiniSat が解を求めるまでの処理
  - 12 decisions
  - 1 conflict (backtrack).

充足不能の場合 ($m \leq 1167$)

- MiniSat が充足不能を示すまでの処理
  - 6 decisions
  - 5 conflicts (backtracks).

- MiniSat の単位伝播により高速な範囲伝播が実現されている.

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)
- **各列について** alldifferent **制約** (5 **列**)

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)
- **各列について** alldifferent **制約** (5 **列**)

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)
- **各列について** alldifferent **制約** (5 **列**)
- **各対角線について** alldifferent **制約** (10)

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)
- **各列について** alldifferent **制約** (5 **列**)
- **各対角線について** alldifferent **制約** (10)

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)
- **各列について** alldifferent **制約** (5 **列**)
- **各対角線について** alldifferent **制約** (10)

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)
- **各列について** alldifferent **制約** (5 **列**)
- **各対角線について** alldifferent **制約** (10)

ラテン方陣問題

サイズ 5 のラテン方陣問題

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

1	2	3	4	5
3	4	5	1	2
5	1	2	3	4
2	3	4	5	1
4	5	1	2	3

- $x_{ij} \in \{1, 2, 3, 4, 5\}$
- **各行について** alldifferent **制約** (5 **行**)
- **各列について** alldifferent **制約** (5 **列**)
- **各対角線について** alldifferent **制約** (10)
- **サイズ** 5 **のラテン方陣問題は充足可能**

ラテン方陣問題の求解時間

Size	SAT/UNSAT	Sugar+m	Abscon	Mistral	bpsolver	Choco
3	UNSAT	0.57	0.66	0.01	0.03	0.41
4	UNSAT	0.59	0.63	0.01	0.03	0.41
5	SAT	0.73	0.68	0.01	0.03	0.58
6	UNSAT	0.79	0.82	0.03	0.17	0.73
7	SAT	0.94	0.78	0.01	0.04	0.77
8	UNSAT	1.01	676.75	-	-	-
9	UNSAT	1.08	-	-	-	-
10	UNSAT	1.16	-	-	-	-
11	SAT	1.35	-	-	-	-
12	UNSAT	1.66	-	-	-	-

- 2009 年 CSP ソルバー競技会おける各ソルバーの CPU 時間 (秒)
- “-” は 1800 秒のタイムアウト以内に解けなかったことを表す

鳩の巣原理の節 (pigeon hole clauses) の効果

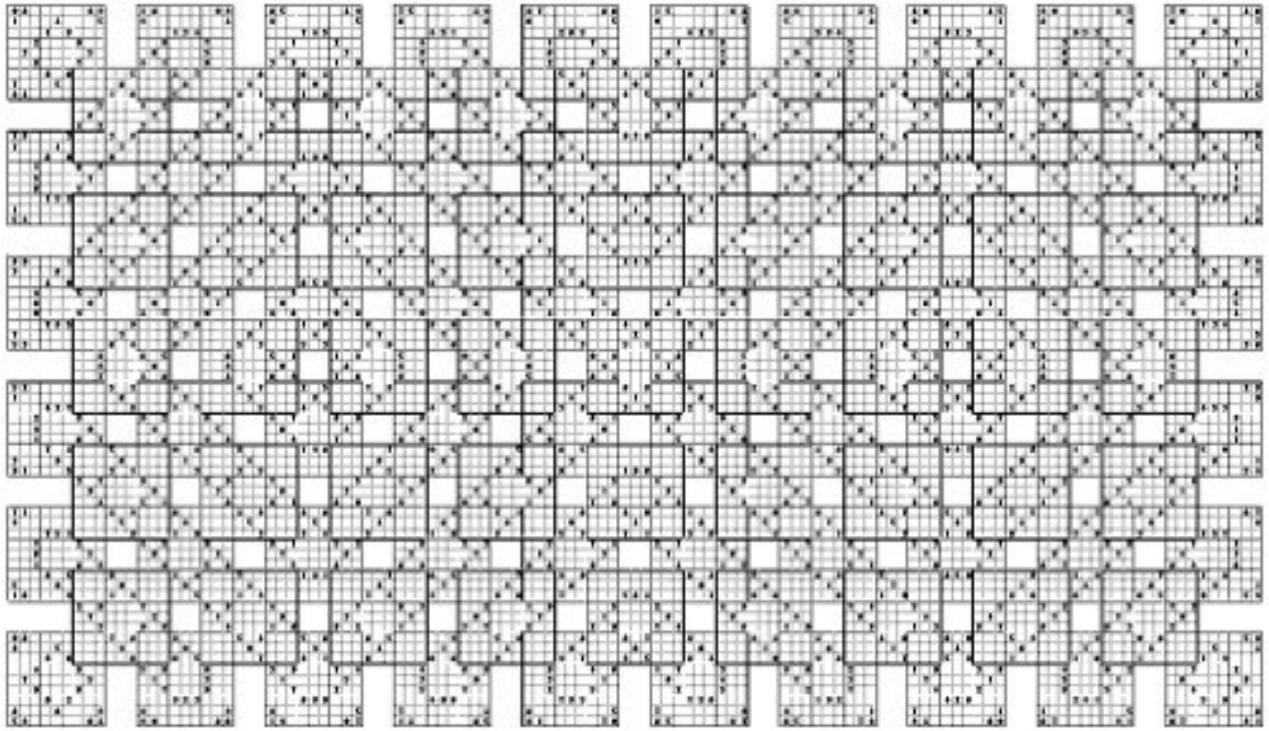
Size	SAT/UNSAT	Sugar+m with p.h. clauses	Sugar+m w/o p.h. clauses
3	UNSAT	0.44	0.35
4	UNSAT	0.47	0.41
5	SAT	0.44	0.43
6	UNSAT	0.52	0.40
7	SAT	0.80	0.69
8	UNSAT	1.08	-
9	UNSAT	0.98	-
10	UNSAT	3.12	-
11	SAT	1.59	-
12	UNSAT	3.23	-

- **鳩の巣原理の節は，**SAT **と** UNSAT **どちらの場合も劇的に性能を改善している．**
- **鳩の巣原理の節は順序符号化に適している．各** alldifferent **制約について** 2 **つの** SAT **節だけが追加される．**

Sugar のデモ

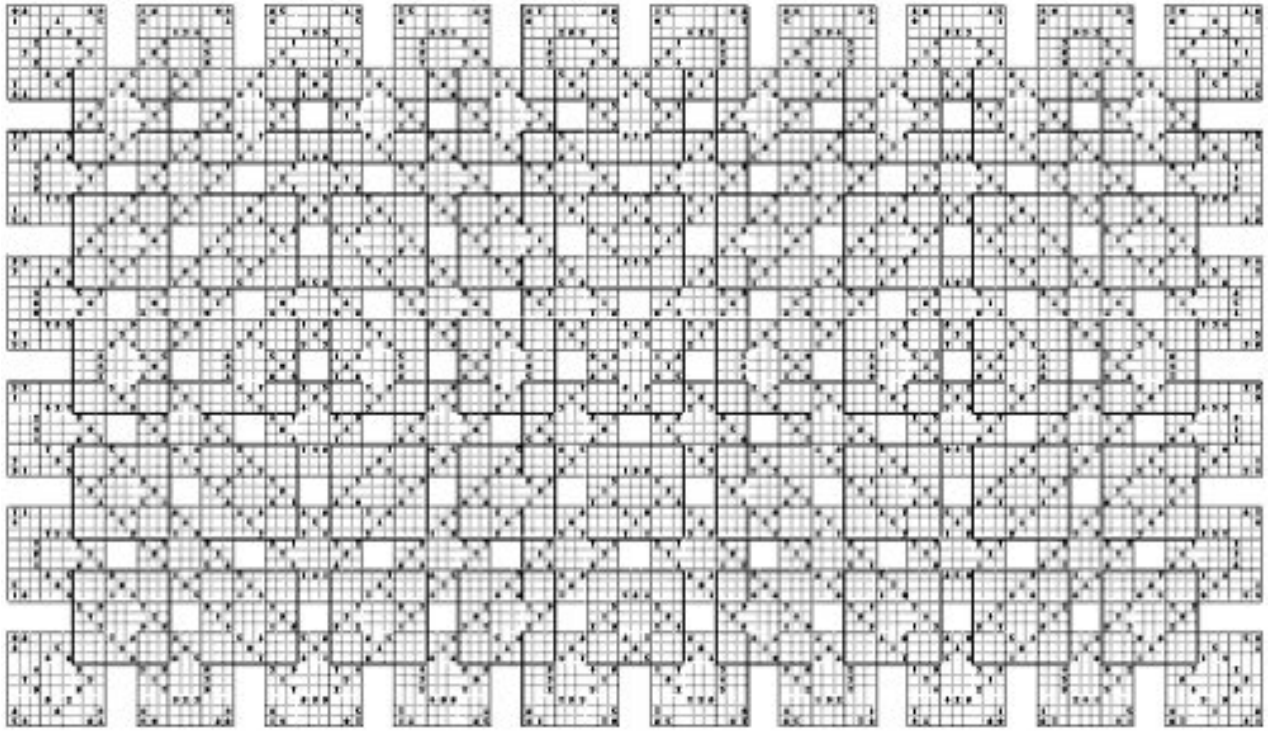
- 巨大な数独
- ノノグラム

巨大な数独



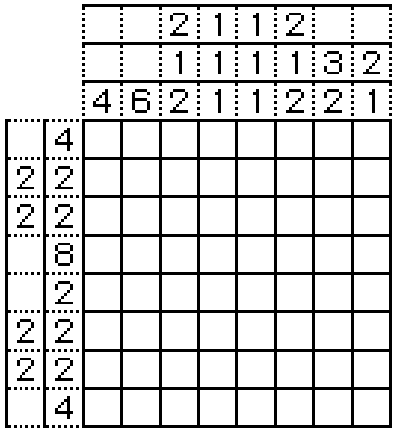
- 105 個の 9×9 数独が重なった問題 (藤原博文さん作).
- 6885 個のマス，1808 個の数字，2655 個の alldifferent 制約から成る.

巨大な数独



- 105 個の 9×9 数独が重なった問題 (藤原博文さん作).
- 6885 個のマス，1808 個の数字，2655 個の alldifferent 制約から成る.
- **Sugar は 30 秒以内で求解する.**

ノノグラムのルール

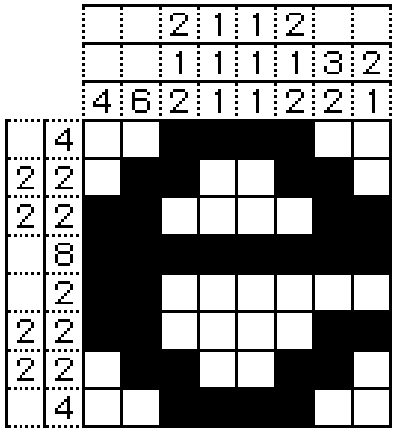


ノノグラムのルール

以下のルールに従って盤面をぬりつぶします．

1. 数字は，その行または列で連続に塗りつぶす黒マスのブロックの長さを表します．
2. 数字が複数ある場合，黒マスのブロックはその順序に並びます．
3. 黒マスのブロックどうしの間は 1 マス以上空いていなければなりません．

ノノグラムのルール

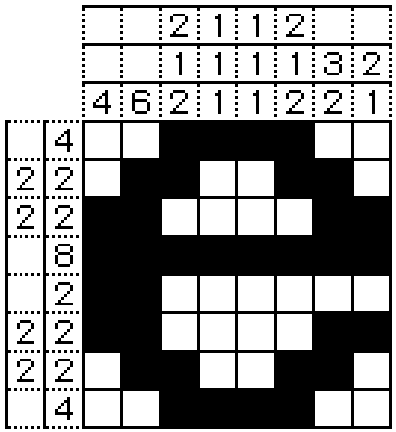


ノノグラムのルール

以下のルールに従って盤面をぬりつぶします．

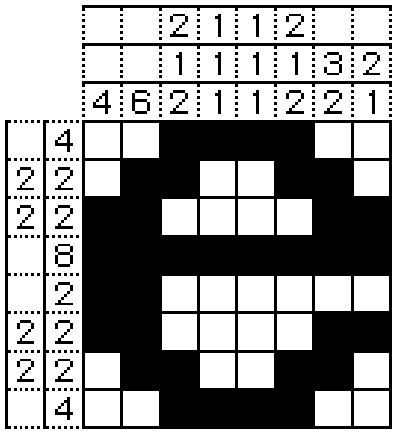
1. 数字は，その行または列で連続に塗りつぶす黒マスのブロックの長さを表します．
2. 数字が複数ある場合，黒マスのブロックはその順序に並びます．
3. 黒マスのブロックどうしの間は 1 マス以上空いていなければなりません．

ノノグラムの制約モデル



- **各マスに整数変数 $x_{ij} \in \{0, 1\}$ を割当てる ($x_{ij} = 1$ は黒に塗ることを意味する).**
  - $x_{00} = 0,\ x_{01} = 0,\ x_{02} = 1,\ x_{03} = 1,\ x_{04} = 1,\ x_{05} = 1,\ \ldots$
- **i 行目の k 番目のブロックの左端の位置を整数変数 h_{ik} で表す.**
  - $h_{00} = 2,\ h_{10} = 1,\ h_{11} = 5,\ h_{20} = 0,\ h_{21} = 6,\ h_{30} = 0,\ \ldots$
- **j 列目の k 番目のブロックの上端の位置を整数変数 v_{jk} で表す.**
  - $v_{00} = 2,\ v_{10} = 1,\ v_{20} = 0,\ v_{21} = 3,\ v_{22} = 6,\ v_{30} = 0,\ \ldots$

ノノグラムの制約モデル



- **数字は，その行または列で連続に塗りつぶす黒マスのブロックの長さを表します．**

$$x_{0j} = 1 \Leftrightarrow (h_{00} \leq j \wedge j < h_{00} + 4)$$
$$x_{1j} = 1 \Leftrightarrow (h_{10} \leq j \wedge j < h_{10} + 2) \vee (h_{11} \leq j \wedge j < h_{11} + 2)$$
$$x_{2j} = 1 \Leftrightarrow (h_{20} \leq j \wedge j < h_{20} + 2) \vee (h_{21} \leq j \wedge j < h_{21} + 2)$$
$$x_{3j} = 1 \Leftrightarrow (h_{30} \leq j \wedge j < h_{30} + 8)$$
$$\ldots$$

ノノグラムの制約モデル

- 数字は，その行または列で連続に塗りつぶす黒マスのブロックの長さを表します．

$$x_{i0} = 1 \Leftrightarrow v_{00} \leq i < v_{00} + 4$$
$$\ldots$$

- 黒マスのブロックどうしの間は 1 マス以上空いていなければなりません．

$$h_{10} + 2 < h_{11}$$
$$\ldots$$

ノノグラムを Sugar で解く

1. Perl スクリプトで 100x100 のノノグラムパズルの CSP を生成する．

```
$ ./nonogram.pl data/nonogram-warship.txt >x.csp
$ wc -l x.csp
31649 x.csp
```

2. CSP は，141092 個の変数と 259085 個の節からなる SAT に符号化され，MiniSat により 4 秒以内で解かれる．

```
$ sugar -vv x.csp | tee x.log
```

3. Perl スクリプトで解が表示される．

```
$ ./nonogram.pl -s x.log data/nonogram-warship.txt
```

まとめ

以下について紹介した．

- 順序符号化
- Sugar 制約ソルバー

なお，Sugar は下記 CSPSAT プロジェクトの一部として研究開発を進めている．

- 科学研究費補助金 基盤研究 (A)

  「制約最適化問題の SAT 変換による解法とその並列分散処理に関する研究」

  http://www.edu.kobe-u.ac.jp/istc-tamlab/cspsat/ ▸ Web

CSPSAT プロジェクト (2008–2011)

研究目的

高性能かつ実際的な SAT 型制約ソルバーの研究開発

研究チーム

- **神戸大学** (3 **名**)
- **国立情報学研究所** (1 **名**)
- **山梨大学** (3 **名**)
- **九州大学** (4 **名**)
- **早稲田大学** (1 **名**)

- SAT **符号化**
  - CSP，**動的** CSP，**時相論理**，**分散** CSP
- **並列** SAT **ソルバー**
  - **マルチコア**，PC **クラスタ**

参考資料

- SAT と SAT ソルバー
  - 人工知能学会誌 第 25 巻 第 1 号 (2010 年 1 月)，特集「最近の SAT 技術の発展」
  - Handbook of Satisfiability, IOS Press, 2009.
  - International Conference on Theory and Applications of Satisfiability Testing (SAT)
  - Journal on Satisfiability, Boolean Modeling and Computation
  - “The Quest for Efficient Boolean Satisfiability Solvers”, CADE 2002 [Zhang & Malik 2002]
  - “An Extensible SAT-Solver”, SAT 2003 [Eén & Sörensson 2003]
- Sugar
  - “Compiling Finite Linear CSP into SAT”, Constraints, Vol.14, No.2, pp.254–272 [Tamura et al. 2009] Open Access