SAT 型制約ソルバーを用いたナンバーリンクの解法

田村直之 [1)]，宋剛秀 [1)]，番原睦則 [1)]，鍋島英知 [2)]

1) 神戸大学　2) 山梨大学

DA シンポジウム 2014
2014 年 8 月 29 日

目次

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 NP-完全であることが最初に証明された問題である [Cook 1971]．
- 具体的な SAT の問題は，連言標準形 (CNF) で与えられる．
  - 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．
- SAT ソルバーの大幅な性能向上を背景として，問題を SAT 問題に符号化し SAT ソルバーを用いて求解する SAT 型システムが様々な分野で成功している．

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．充足可能であればその値割当てを解として出力する．

- 近代的 SAT ソルバーでは，DPLL アルゴリズムに様々な技術が導入され大幅な性能向上が実現されている．
  - 矛盾からの節学習 (CDCL)，非時間順バックトラック法，ランダムリスタート，監視リテラル，変数選択ヒューリスティック (VSIDS)
- 近代的 SAT ソルバーは 10^6 個の命題変数 10^7 個の節を含む SAT 問題も取り扱うことができる．
- 2002 年以降国際 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．

GlueMiniSat

GlueMiniSat

多くの最新 SAT **技術を実装した** SAT **ソルバー**

- 2011, 2013 **年の国際** SAT **競技会の産業応用部門** UNSAT **トラックにて，それぞれ優勝・準優勝** (UNSAT **の求解に強い**)

- **代表的な** SAT **ソルバーである** MiniSat [Eén+ 03] **をベースに開発**
  - **矛盾からの節学習** [Silva+ 99, Bayardo+ 97]
  - **監視リテラルによる高速単位伝搬** [Moskewicz+ 01]
  - **変数選択ヒューリスティクス** VSIDS [Moskewicz+ 01]
- **求解中に** SAT **問題の簡単化を頻繁に行う軽量なアルゴリズムを実装** [Nabeshima+ 13]
- **単位伝搬を促す学習節を保持・獲得するためのリテラルブロック距離と呼ばれる学習節の評価尺度** [Audemard+ 09] **に基づく節削減・リスタート戦略を導入** [鍋島+ 12]

制約充足問題 (CSP)

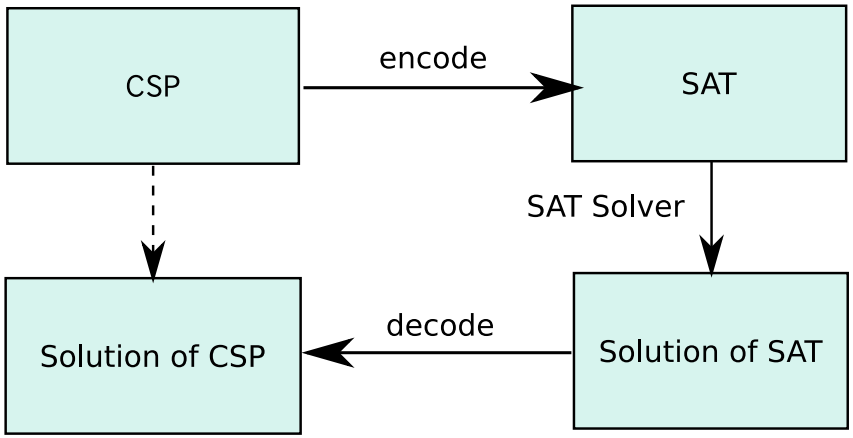
制約充足問題

制約充足問題 (CSP; Constraint Satisfaction Problem) **は以下を満たす組** (X, Dom, C) **で与えられる．**

- X: **変数の有限集合**
- Dom: **各変数の**ドメイン **(値領域) を与える関数**
  - **ここでは，ドメインは整数の有限領域とする**
- C: X **上の制約の有限集合**
  - **ここでは，各制約は算術演算，算術関係，論理演算で表されるとする**

- **与えられた** CSP **の解を探索するプログラムを制約ソルバー (あるいは** CSP **ソルバー) と呼ぶ**
- **多くの制約ソルバーは，与えられた目的関数を最大 (あるいは最小) にする解を探索する問題である制約最適化問題** (COP) **にも対応している．**

Sugar: SAT 型制約ソルバー



SAT 型制約ソルバー Sugar

Sugar は順序符号化を用いた SAT 型制約ソルバー.

- 2008 年, 2009 年 CSP ソルバー競技会のグローバル制約部門で優勝
- 様々な問題で良い性能を示している (ショップスケジューリング，2 次元パッキング，テストケース生成など).

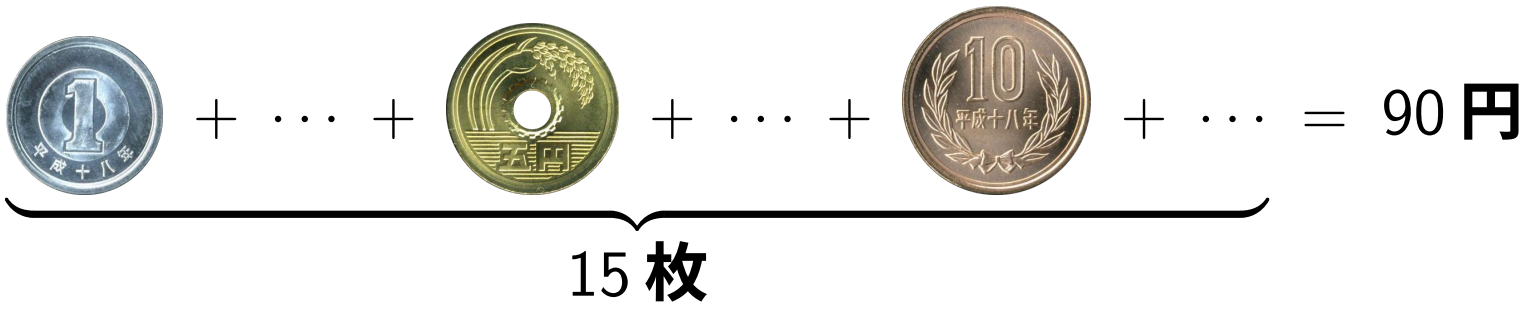
Copris

Copris

Copris は Scala 上に実現された制約プログラミング用 DSL (Domain-Specific Language)

- 制約モデリングが容易
- 複数のバックエンド・ソルバーを切り換えて利用可能
  - SAT ソルバー
  - PB ソルバー
  - SMT ソルバー
  - JSR-331 準拠の制約ソルバー
- SAT ソルバーとして Sat4j を利用すれば，すべてが JVM 上で動作

Coprisのプログラム例




```
import jp.kobe_u.copris._
import jp.kobe_u.copris.dsl._
int('x, 1, 15)
int('y, 1, 15)
int('z, 1, 15)
add('x + 'y + 'z === 15)
add('x + 'y * 5 + 'z * 10 === 90)
if (find)
  print(solution)
```

ナンバーリンクの制約モデル

各マスを頂点とし，隣接するマスとの間に弧がある有向グラフ $D(V, A)$ を考える．D の部分グラフ $D'(V, A')$ がナンバーリンクの解となる条件を制約で表す．

- 基本モデル
- 改良モデル
- 追加制約

基本モデル (1)

基本モデル (1)

- **各弧** $(u, v) \in A$ **について**

$$\textbf{弧}\,(u, v) \in \{0, 1\}$$
$$\textbf{弧}\,(u, v) + \textbf{弧}\,(v, u) \leq 1$$

- **各マス** $u \in V$ **について**

$$\textbf{入次数}\,(u) \in \{0, 1\}$$
$$\textbf{出次数}\,(u) \in \{0, 1\}$$
$$\textbf{入次数}\,(u) = \textbf{弧}\,(u_{\textbf{上}}, u) + \cdots + \textbf{弧}\,(u_{\textbf{右}}, u)$$
$$\textbf{出次数}\,(u) = \textbf{弧}\,(u, u_{\textbf{上}}) + \cdots + \textbf{弧}\,(u, u_{\textbf{右}})$$

- **各マス** $u \in V$ **について**

$$\textbf{数}\,(u) \in \{1, 2, \ldots, \textbf{最大の数}\}$$

基本モデル (2)

基本モデル (2)

- 白マス u について

$$\text{入次数}(u) = \text{出次数}(u)$$

- 数字マスの一方を u, 他方を v とすると

$$\text{入次数}(u) = 0 \qquad \text{出次数}(u) = 1$$
$$\text{入次数}(v) = 1 \qquad \text{出次数}(v) = 0$$

- 各数字マス u について

$$\text{数}(u) = i$$

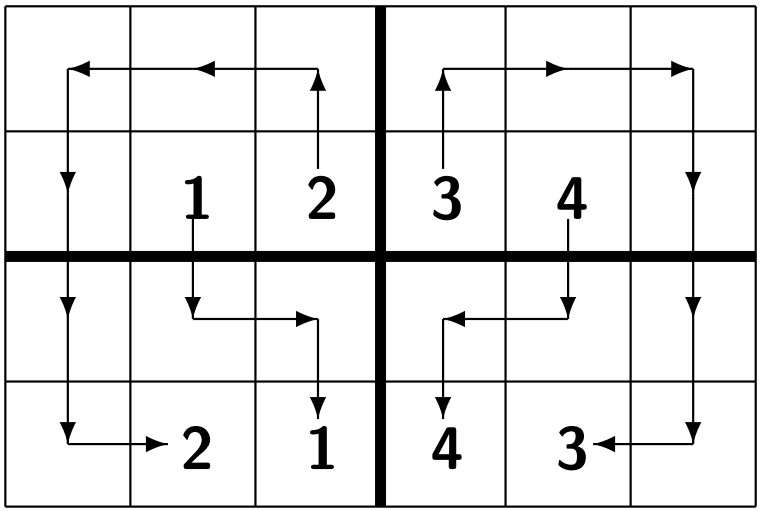
- 各弧 $(u, v) \in A$ について

$$\text{弧}(u, v) > 0 \Rightarrow \text{数}(u) = \text{数}(v)$$

改良モデル

改良モデル

領域を 4 分割し，各領域の出次数に関する制約を追加する．



左上領域の出次数 $= +2$

右上領域の出次数 $= +2$

左下領域の出次数 $= -2$

右下領域の出次数 $= -2$

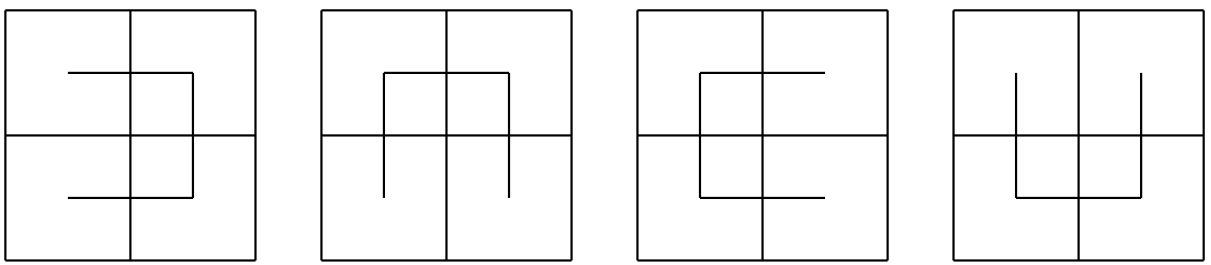
追加制約

追加制約

解を漏らす可能性はあるが，高速化のための追加制約を検討する

- **(追加制約 1) 回り道をしない**
- **(追加制約 2) すべてのマスを通る**

追加制約 1 では，以下の経路を排除する．



追加制約 2 では，各白マス u について以下を追加する．

$$\text{入次数}\,(u) = 1 \qquad \text{出次数}\,(u) = 1$$

性能評価

問題番号	048	110	127	158	160	190
列数	15	42	15	12	12	48
行数	15	25	15	12	12	35
基本	551	703	–	335	759	–
改良	480	49	–	187	283	–
基本+2	36	47	–	26	178	–
改良+2	49	23	–	30	140	–
基本+1	37	175	–	36	167	–
改良+1	70	25	–	24	60	–
基本+1+2	6	16	6	4	4	34
改良+1+2	6	16	7	4	4	30

- www.janko.at サイト中の 280 問から最も難しい 6 問を使用
- 表中の数は，最初の 5 個の解探索に要した時間 (秒)
  - “–” は，1800 秒以内に解探索が終了しなかったことを表す

性能評価

問題番号	048	110	127	158	160	190
列数	15	42	15	12	12	48
行数	15	25	15	12	12	35
基本	551	703	–	335	759	–
改良	480	49	–	187	283	–
基本+2	36	47	–	26	178	–
改良+2	49	23	–	30	140	–
基本+1	37	175	–	36	167	–
改良+1	70	25	–	24	60	–
基本+1+2	6	16	6	4	4	34
改良+1+2	6	16	7	4	4	30

- www.janko.at サイト中の 280 問から最も難しい 6 問を使用
- 表中の数は，最初の 5 個の解探索に要した時間 (秒)
  - “–” は，1800 秒以内に解探索が終了しなかったことを表す
- 改良モデルのほうが良い
- 追加制約 1, 2 どちらでも速度が向上している
- 両方を追加すると大幅に良くなる
- 赤字部分は予稿集からの修正

今回用いた戦略と結果

今回用いた戦略

- 改良モデルを用いた．
- 追加制約 1, 2 の両方を追加すると大幅に性能向上するが，解を漏らす可能性がある．そのため以下の戦略を用いた．
  1. 改良モデル+1+2 で解を求める (数十秒程度で終了する)．
  2. 改良モデル+1 で解を求める (既知の解は除く)．
  3. 改良モデル で解を求める (既知の解は除く)．

今回の結果

2 分以内で全解を求められた問題	8
5 分以内で全解を求められた問題	1
2 分以内で解を求められた問題 (全解かどうかは不明)	7

- 結果的には，全解を求めることができた．
- 希望の方には，プログラム一式をお送りします．

まとめ

- SAT 型制約ソルバーを用いたナンバーリンクの解法について述べた．
  - SAT ソルバー: GlueMiniSat
  - SAT 型制約ソルバー: Sugar
  - 制約プログラミング用 DSL: Copris
- ナンバーリンクの制約モデルを提案した．
  - 基本モデル，改良モデル，追加制約
- SAT 型制約ソルバーは，配線問題だけではなく，配置問題，割当問題，スケジューリング問題など様々に応用可能である．
- ただ，SAT ソルバーの性能を引き出すには適切なモデル，適切な SAT 符号化が必要である．
  - 参考: 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」

- コンテスト・オーガナイザーの方々に感謝いたします．

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 3 4       ; Number of variables and clauses
1 2 3 0         ; p1 ∨ p2 ∨ p3
-1 -2 0         ; ¬p1 ∨ ¬p2
-1 -3 0         ; ¬p1 ∨ ¬p3
-2 -3 0         ; ¬p2 ∨ ¬p3
```

SAT 競技会 (SAT Competitions)

	Gold	Silver	Bronze
Application			
SAT	Lingeling	**ZENN**	satUZK
Certified UNSAT	glucose	**glueminisat**	Riss3g
SAT+UNSAT	Lingeling	Lingeling	**ZENN**
Hard-combinatorial			
SAT	glucose	gluebit_clasp	BreakIDGlucose
Certified UNSAT	Riss3g	glucose	forl
SAT+UNSAT	BreakIDGlucose	gluebit_clasp	glucose
Random			
SAT	probSAT	sattime	Ncca+
Certified UNSAT	**dk-SAT11**	march_br	
SAT+UNSAT	CSHCrandMC	MIPSat	march_vflip

- 2013 **年** SAT **競技会** Core Solvers, Sequential **トラックの結果**
- 2011 **年と比べて，いくらか傾向が変わったようだ**

SAT 型システム (SAT-based Systems)

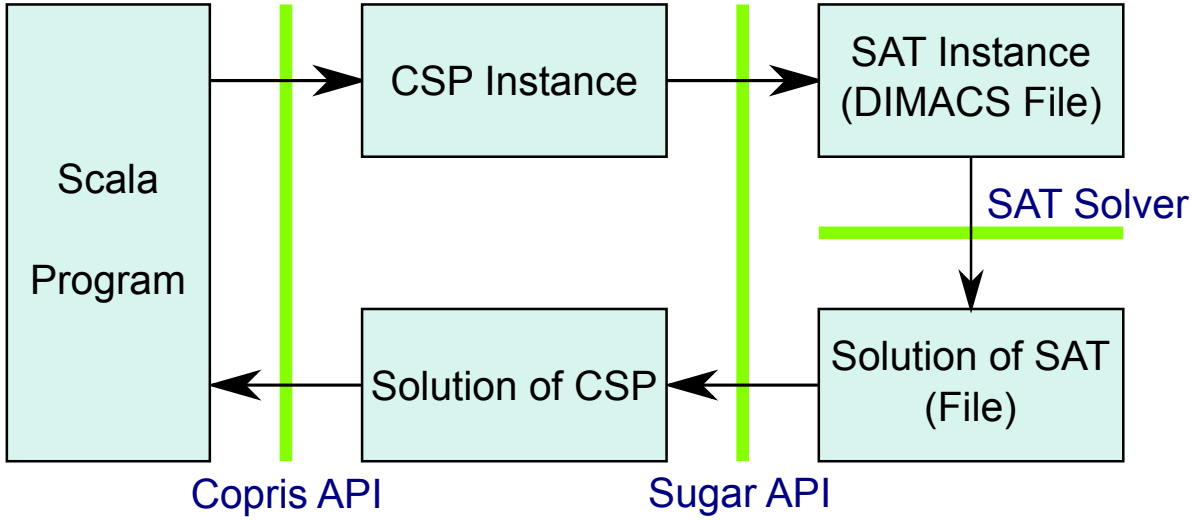
- **プランニング** (SATPLAN, Blackbox) [Kautz & Selman 1992]
- **自動テストパターン生成** [Larrabee 1992]
- **ジョブショップスケジューリング** [Crawford & Baker 1994]
- **ソフトウェア検証** (Alloy)
- **有界モデル検査** [Biere 1999]
- **ソフトウェアパッケージの依存解析** (SATURN)
  - Sat4j **は** Eclipse **で利用されている．**
- **書換えシステム** (AProVE, Jambox)
- **解集合プログラミング** (clasp, Cmodels-2)
- **一階論理定理証明，モデル発見** (iProver, Darwin, Paradox)
- **制約充足問題** (Sugar) [Tamura et al. 2006]
- **プログラミング言語のコンパイラ**
  - Scala **コンパイラで** Sat4j **が利用される予定**

Sugar の CSC 2009 での結果

Series		Sugar+m	Sugar+p	Mistral	Choco	bpsolver
BIBD	(83)	76	77	76	58	35
Costas Array	(11)	8	8	9	9	9
Latin Square	(10)	10	9	5	5	5
Magic Square	(18)	8	8	13	15	11
NengFa	(3)	3	3	3	3	3
Orthogonal Latin Square	(9)	3	3	3	2	3
Perfect Square Packing	(74)	54	53	40	47	36
Pigeons	(19)	19	19	19	19	19
Quasigroup Existence	(35)	30	29	29	28	30
Pseudo-Boolean	(100)	68	75	59	53	70
BQWH	(20)	20	20	20	20	20
Cumulative Job-Shop	(10)	4	4	2	1	0
RCPSP	(78)	78	78	78	77	75
Cabinet	(40)	40	40	40	40	40
Timetabling	(46)	25	42	39	14	1
Total	(556)	446	468	435	391	357

- **グローバル制約 3 部門での解けた問題数**
- Sugar+m : Sugar with MiniSat 2.0 backend
- Sugar+p : Sugar with PicoSAT 535 backend

Copris の構成



1. **ユーザの** Scala **プログラムから** Copris API **を通じて** CSP **が定義される．**
2. Scala **プログラムから** Copris **ソルバーが呼び出される．**
   - **定義された** CSP **が** Sugar **を用いて** SAT **問題ファイルに符号化される．**
   - Copris **から** SAT **ソルバー (デフォールトは** Sat4j**) が呼び出される．**MiniSat, Glucose, GlueMiniSat **等の他の** SAT **ソルバーを利用することも可能．**
   - Sugar **により，**SAT **問題の解が** CSP **の解に変換される．**
3. CSP **の解がユーザのプログラムに戻される．**