1M1-OS-02a-4

制約充足問題のASP符号化に関する一考察

A Study of ASP Encoding of Constraint Satisfaction Problems

坡山直樹 *1 Naoki Hayama　番原睦則 *2 Mutsunori Banbara　宋剛秀 *2 Takehide Soh　田村直之 *2 Naoyuki Tamura

*1神戸大学大学院システム情報学研究科 Graduate School of System Informatics, Kobe University

*2神戸大学情報基盤センター Information Science and Technology Center, Kobe University

Answer Set Programming (ASP) is an approach to declarative problem solving, combining a rich yet simple modeling language with high performance solving capacities. We develop first-order ASP encodings for solving finite domain Constraint Satisfaction Problems (CSPs). We present an ASP-conscious encoding utilizing ASP's weighted cardinality constraints, as well as ones based on existing SAT encodings for CSP solving. Given CSP instances of fact format are combined with a first-order ASP encoding for CSP solving, which can subsequently be solved by any off-the-shelf ASP systems. We empirically evaluate the effectiveness of our encodings by using the two-dimensional strip packing problem.

1. はじめに

解集合プログラミング (Answer Set Programming; ASP [Gelfond 88, 井上 08]) は，論理プログラミングから派生した宣言的プログラミングパラダイムである．ASP 言語は一階論理に基づく知識表現言語の一種である．ASP システムは安定モデル意味論に基づく解集合を計算するシステムである．近年，SAT ソルバー技術を応用した高速 ASP システムが実現され，ロボット工学，システム生物学，システム検証，プランニング，スケジューリングなど様々な分野への実用的応用が急速に拡大している．

ASP は記号上の制約を簡潔に表現できるが，**制約充足問題** (Constraint Satisfaction Problems; CSPs [Rossi 06]) のような各種ドメイン上の算術制約を自然に表現できない．そのため，算術制約を記号上の制約に変換 (**ASP 符号化**) する必要がある．例えば，整数有限ドメイン上の算術制約の場合，各整数変数 x と各整数定数 $a \in Dom(x)$ に対して，$x = a$ を意味するアトム `d(`x`,`a`)` を導入し，制約に違反する点を符号化する方法が一般的である．しかし，これが最良であるとは限らない．

本論文では，ASP で算術制約を効率的に取り扱うために，線形の算術制約を ASP に符号化する方法について述べる．本論文は整数有限ドメイン上の 2 変数 CSP を対象とする．与えられた CSP インスタンスは，開発した CSP を解く符号化プログラムと結合され，汎用の ASP システムによって解かれる．得られた解集合は元の CSP の解に相当する．

まず最初に，CSP から SAT への符号化法である直接符号化法，多値符号化法，支持符号化法，対数符号化法，順序符号化法を参考にした 5 種類の ASP 符号化プログラムについて述べる．次に，ASP の特長を生かした ASP 対数符号化法を提案する．この符号化法の特長は，線形の算術制約を**重み付き個数制約**と呼ばれる単一の ASP 規則として表現する点である．最後に，二次元矩形パッキング問題を用いた比較実験を行い，各 ASP 符号化法の有効性を評価する．

次節で ASP について簡単に述べるが，より詳しい説明については文献 [Gebser 12, 井上 08] を参照していただきたい．また，SAT 符号化の詳しい解説については，文献 [田村 10] がある．

連絡先: 坡山直樹．神戸大学大学院システム情報学研究科．神戸市灘区六甲台町 1-1．078-803-5365．hayama@stu.kobe-u.ac.jp

2. 解集合プログラミング

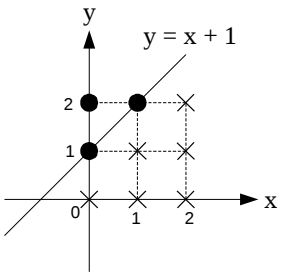
ASP 言語は一般拡張選言プログラムをベースとしている．説明の簡略化のため，そのサブクラスである**標準論理プログラム** (Normal Logic Program; NLP) について説明する．NLP は以下の形式の規則の集合である．

$$L_1 \texttt{ :- } L_2\texttt{, ...,}L_m\texttt{, not } L_{m+1}\texttt{,..., not } L_n\texttt{.}$$

この規則の直観的な意味は，「$L_2, \ldots, L_m$ がすべて成り立ち，$L_{m+1}, \ldots, L_n$ のそれぞれが成り立たないならば，L_1 が成り立つ」である．ここで $n \geq m \geq 0$ であり，L_i は正リテラル (アトム) で，`not` は**デフォルトの否定**，“`,`” は連言を表す．“`:-`” の左側を**ヘッド**，右側を**ボディ**と呼ぶ．ボディが空の規則を**ファクト**と呼び，ファクトは “`:-`” を省略してよい．ヘッドが空の規則を**一貫性制約**と呼ぶ．例えば，“`:- not` L`.`” は L が成り立たなければならないことを意味し，“`:-` L`.`” は，L が成り立たないことを意味する．

さらに，最新の ASP 言語では，組合せ問題を簡潔に記述するために，**アグリゲート** (aggregate) と呼ばれる表記法がいくつか用意されている．例えば，**選択子** “`{`L_1`;...;`L_k`}`.” は，集合 $\{L_1, \ldots, L_k\}$ の任意の部分集合を解集合に含めることを意味する．**重み付き個数制約** “`#sum {`w_1`:`L_1`;...;`w_k`:`L_k` } =` c`.`” は，$L_1, \ldots, L_k$ のうち真となるリテラルの重み和が整数定数 c に等しくなることを意味する．重み w_i は整数定数であり，演算子としては “`=`” 以外にも “`<`”,“`>`” などを使用できる．

ASP システムは，与えられた一階の ASP プログラムに対して，安定モデル意味論 [Gelfond 88, 井上 08] に基づく解集合を計算するシステムである．最新の高速 ASP システムは，一階 ASP プログラムを命題 ASP プログラムに変換 (**基礎化**) したのち，ASP ソルバーを用いて解集合を計算する．ASP ソルバーは，重み付き個数制約を符号化することなくソルバー内部で処理するため，SAT 符号化で問題となる節数の増大を回避し，より大きなドメインをもつ CSP を解くことが期待できる．本論文で使用する ASP システム *clingo* は，基礎化のためのグラウンダー *gringo* と ASP ソルバー *clasp* をシームレスに結合したシステムである．



$x, y \in \{0, 1, 2\}$
$x + 1 \leq y$

図 1: CSP の例

```
int(x,range(0,2)).   int(y,range(0,2)).   le(x,1,y).
```

Listing 1: CSP の ASP ファクト表現

3. 線形制約の ASP 符号化

まず最初に，CSP から SAT への符号化法である直接符号化法，多値符号化法，支持符号化法，対数符号化法，順序符号化法を参考に開発した 5 種類の ASP 符号化プログラムと，ASP の特長を生かすよう考案した ASP 対数符号化法の概要について述べる．

直接符号化法： 各整数変数 x と各整数定数 $a \in Dom(x)$ [*1] に対して，$x = a$ を意味するアトム `d(`x`,`a`)` を導入する．制約は違反点を排除する規則に符号化される．

多値符号化法： 直接符号化法と同じだが，各整数変数が (1 つではなく) 複数のドメイン値を取ることを許す点が異なる．

支持符号化法： 整数変数の符号化は直接符号化法と同じ．制約は支持点を強制する規則に符号化される．

順序符号化法： 各整数変数 x と各整数定数 $a \in Dom(x)$ に対して，$x \leq a$ を意味するアトム `p(`x`,`a`)` を導入する．制約は違反範囲を排除する規則に符号化される．

対数符号化法： 各整数変数 x の 2 進表現に着目し，x の i 桁目が 1 であることを意味するアトム `b(`x`,`i`)` を導入する．制約は違反点を排除する規則に符号化される．

ASP 対数符号化法： 整数変数の符号化は対数符号化法と同じ．各制約は単一の重み付き個数制約で表現される．

本節では紙面の都合上，後述の実験でよい性能を示した順序符号化法と ASP 対数符号化法について詳細に述べる．

ASP ファクト形式. 与えられた CSP インスタンスを ASP のファクトとして表現するために，述語 `int/2` と `le/3` を導入する．アトム `int(`x`,range(`lb`,`ub`))` は整数変数 $x \in \{lb, \ldots, ub\}$ を表す．アトム `le(`x`,`c`,`y`)` は $x + c \leq y$ を表す．ただし，x と y は整数変数，c は整数定数である．簡単な 2 変数 CSP の例を図 1 に示す．この CSP は 2 つの整数変数と 1 つの線形制約から成り，解は 3 つ存在する．図中の黒丸が支持点，バツ印が違反点を表している．この例を ASP ファクトとして表現したものを Listing 1 に示す．

補助アトム. CSP を解く ASP 符号化を簡潔に記述するために，与えられたファクト形式の CSP インスタンスから，いくつかの補助アトムを生成する．アトム `var(`x`)` は変数名を表す．アトム `dom(`x`,`a`)` は，変数 x が値 a をもつことを意味する．アトム `dom_min(`x`,`lb`)` は変数 x の下限値が lb，`dom_max(`x`,`ub`)` は変数 x の上限値が ub であることを表す．アトム `msb(`x`,`msb`)` は，変数 x を 2 進数表現した場合の最上位ビットが msb であることを意味する．Listing 2 に，Listing 1 の整数変数 `x` に対して生成される補助アトムを示す．

*1 関数 Dom は，各変数の取り得る値集合 (ドメイン) を定める．

```
var(x).  dom(x,0).  dom(x,1).  dom(x,2).
dom_min(x,0).  dom_max(x,2).  msb(x,2).
```

Listing 2: 補助アトム

```
1 % 整数変数
2 { p(V,A) : dom(V,A) } :- var(V).
3 :- p(V,A), not p(V,A+1), dom(V,A+1).
4 :- not p(V,Max), dom_max(V,Max).
5 % 線形制約
6 :- not p(X,A-C), p(Y,A), dom(X,A-C), le(X,C,Y).
7 :- p(Y,A), A-C<Min, dom_min(X,Min), le(X,C,Y).
```

Listing 3: 順序符号化法

一階 ASP 符号化. 順序符号化法は，各整数変数 x と各整数定数 $a \in Dom(x)$ に対して，$x \leq a$ を意味するアトム `p(`x`,`a`)` を導入する．制約は違反範囲を排除する規則に符号化される．

Listing 3 に，順序符号化法を用いて CSP を解く ASP 符号化プログラムを示す．2 行目の選択子を含む規則は，各変数 `V` と，`dom(V,A)` となる整数定数 `A` に対して，`p(V,A)` を生成する．選択子中のコロン (`:`) は変数に対する条件づけを行う略記法である．3 行目の規則は，各変数 `V`，値 `A`，`A+1` に対して，`V`≤`A` ならば `V`≤`A+1` であるという公理を表す．4 行目の規則は，各変数 `V` とその上限値 `Max` に対して，常に `V`≤`Max` が成り立つことを表している．これら二つの規則により，各変数 `V` がただ一つの値を取ることが保証される．

アトム `le(X,C,Y)` で表される線形制約 `X`+`C`≤`Y` は，6–7 行目の規則で符号化される．6 行目の一貫性制約は，`le(X,C,Y)` である変数 `X`，`Y` と整数定数 `C`，`Y` の各ドメイン値 `A` に対して，`Y`≤`A` かつ `A-C` が `X` のドメイン値ならば，`X`≤`A-C` となることを強制する．7 行目の制約は，`A-C` が変数 `X` のドメインの下限値 `Min` より小さいならば，`Y`≤`A` となることを禁止している．

先ほどの例 (Listing 1) を順序符号化法に基づく符号化プログラム (Listing 3) と結合し，それをグラウンダー *gringo* で基礎化した結果得られる命題 ASP プログラムを Listing 4 に示す．例えば，8 行目の最初の制約は，`y`≤`1` かつ `x`≥`1` の範囲に含まれる 4 つの違反点を排除している．

ASP 対数符号化法は，各整数変数 x の 2 進表現に着目し，x の i 桁目 (最下位ビットを 1 桁目とする) が 1 であることを意味するアトム `b(`x`,`i`)` を導入する．既存の対数符号化法との大きな違いは，対数符号化法が各制約を違反点を排除する複数の規則に符号化するのに対して，ASP 対数符号化法は，各制約を単一の重み付き個数制約で表現する点である．

Listing 5 に，ASP 対数符号化法を用いて CSP を解く ASP 符号化プログラムを示す．2 行目の選択子を含む規則は，各変数 `V` に対して，`b(V,1)`, . . . ,`b(V,M)` を生成する．ただし，`M` は変数 `V` の最上位ビットを表す．変数のドメインの下限は 0 以上とする．3 行目の規則は，ドメインに含まれない値 `S` を，重み付き個数制約を使って排除している．

アトム `le(X,C,Y)` で表される線形制約 `X`+`C`≤`Y` は，5–6 行目の規則で符号化される．この規則は，`le(X,C,Y)` である変数 `X`，`Y` と整数定数 `C` に対して，ASP の重み付き個数制約を使って，`X` と `Y` の値を 10 進数変換した上で，`-X`+`Y`<`C` を禁止 (すなわち，`X`+`C`≤`Y` を強制) している．

先ほどの例 (Listing 1) をこの ASP 対数符号化プログラム (Listing 5) と結合し，それをグラウンダー *gringo* で基礎化し

```
% 整数変数
{p(x,0); p(x,1); p(x,2)}.    {p(y,0); p(y,1); p(y,2)}.
:- p(x,0), not p(x,1). :- p(x,1), not p(x,2).
:- not p(x,2).
:- p(y,0), not p(y,1). :- p(y,1), not p(y,2).
:- not p(y,2).
% 線形制約
:- p(y,1), not p(x,0). :- p(y,2), not p(x,1). :- p(y,0).
```

Listing 4: 順序符号化法：命題 ASP プログラム

```
% 整数変数
{ b(V,1..M) } :- msb(V,M).
:- #sum{ 2**(A-1) : b(V,A) } = S, not dom(V,S), var(V).
% 線形制約
:- #sum{ -(2**(A-1)) : b(X,A); 2**(B-1) : b(Y,B) } < C,
   le(X,C,Y).
```

Listing 5: ASP 対数符号化法

た結果得られる命題 ASP プログラムを Listing 6 に示す．例えば，3–4 行目の規則は，各々，変数 x と y が値 3 を取ることを排除している．また，6 行目の規則は，-x+y<1 を禁止 (すなわち，x+1≤y を強制) している．

4. 実験

各符号化法の性能を評価・比較するために，二次元矩形パッキング問題 (Two-dimensional Strip Packing Problem; 2SPP [Soh 10]) を用いて実験を行った．

2SPP は，長方形の集合とそれらを配置する大きな長方形 (ストリップと呼ぶ) が与えられたとき，各長方形がお互いに重ならないように配置できるストリップの最小の高さを求める組合せ最適化問題である．本実験では，ストリップの高さを与えることにより，2SPP を判定問題として解いた．2SPP 判定問題は CSP として定式化でき，各長方形の非重複に関する制約は 2 変数の線形制約の選言で表される．

本実験では，論文等で広く用いられている 2SPP ベンチマーク問題 [*2] のうち，最適値が既知の 45 問を対象とした．各問題を定数倍 ($c = 1, 10^1, 10^2, 10^3$) した 4 通りの問題を作成し，最適値 (SAT) と最適値 -1(UNSAT) の 2 通り，計 360 問で実験を行った．実験の概要を以下に示す．

実験 1： オリジナルのサイズ ($c = 1$) の問題 90 問を用いて，6 つの符号化法で解けた問題数と実行時間を比較した．

実験 2： 整数変数のドメインサイズによる性能の違いを比較するために，全問題 ($c = 1, 10^1, 10^2, 10^3$，360 問) を用いて，実験 1 で良い性能を示した ASP 対数符号化法と順序符号化法で解けた問題数と実行時間を比較した．

表 1: 実験 1：解けた問題数

符号化法	最適値 (SAT)	最適値-1(UNSAT)
直接符号化法	20	12
多値符号化法	20	11
支持符号化法	24	12
順序符号化法	**30**	**18**
対数符号化法	20	13
ASP 対数符号化法	27	14

*2 http://or.dei.unibo.it/sites/or.dei.unibo.it/files/instances/2sp.zip

```
% 整数変数
{b(x,1); b(x,2)}.        {b(y,1); b(y,2)}.
:- #sum {1:b(x,1); 2:b(x,2)} = 3.
:- #sum {1:b(y,1); 2:b(y,2)} = 3.
% 線形制約
:- #sum {-1:b(x,1); -2:b(x,2); 1:b(y,1); 2:b(y,2)} < 1.
```

Listing 6: ASP 対数符号化法：命題 ASP プログラム

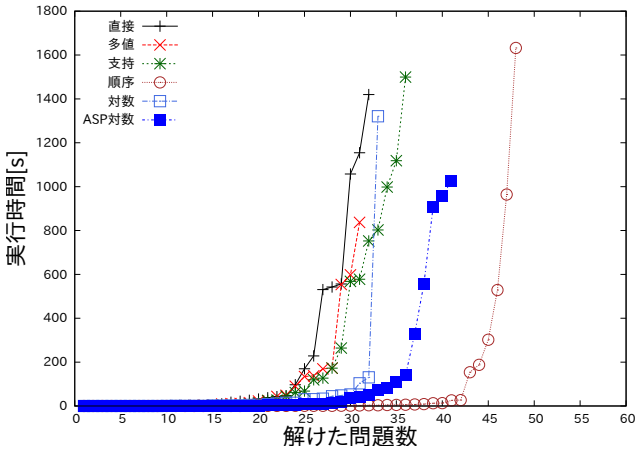


図 2: 実験 1：解けた問題数と実行時間の分布

ASP システムには，*clingo*-5.2.0 を使用し，1 問当たりの時間制限は 1,800 秒とした．実験環境は，Mac OS, Xeon 3.7GHz, 64GB メモリである．

実験 1 について，各符号化法で解けた問題数を表 1 に示す．順序符号化法が最も多い 48 問を解き，SAT と UNSAT の両方で良い性能を示した．また，ASP 対数符号化法は，それに次ぐ性能を示した．図 2 は，各符号化法で解けた問題数と実行時間の分布を示したものである．図の縦軸は実行時間，横軸は解けた問題数である．この図は，線が右にあればあるほど多くの問題を解き，下にあればあるほど高速に求解したことを表している．図より，順序符号化法が最も多くの問題を解き，かつ，高速に求解していることがわかる．なお，縦軸の実行時間は，ASP ソルバー *clasp* が要した求解時間である．

実験 2 では，整数変数のドメインサイズによる性能の違いを比較する．表 2 に，全問題 ($c = 1, 10^1, 10^2, 10^3$，360 問) に対して，実験 1 で良い性能を示した ASP 対数符号化法と順序符号化法が解けた問題数を示す．この表より，定数倍 c が大きくなるにつれて，順序符号化法と ASP 対数符号化法の差が小さくなり，10^3 倍では，ASP 対数符号化法が順序符号化法を大きく上回っていることがわかる．さらに，*clingo* の実行ログを見たところ，ASP 対数符号化法のみ，整数変数のドメインサイズが 10 万を超える問題を解けていた．その一方で，順序

表 2: 実験 2：解けた問題数

定数倍 c	整数変数のサイズ	ASP 対数符号化法 解けた問題数	順序符号化法 解けた問題数
1	$10 \sim 10^4$	41	**48**
10^1	$10^2 \sim 10^5$	37	**43**
10^2	$10^3 \sim 10^6$	32	**35**
10^3	$10^4 \sim 10^7$	**18**	6

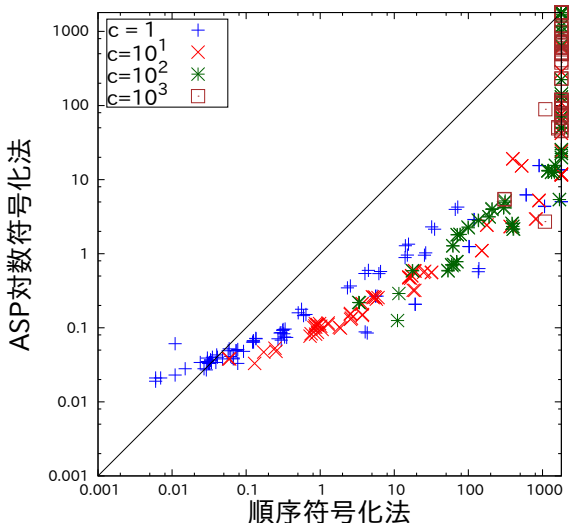


図 3: 実験 2：基礎化時間

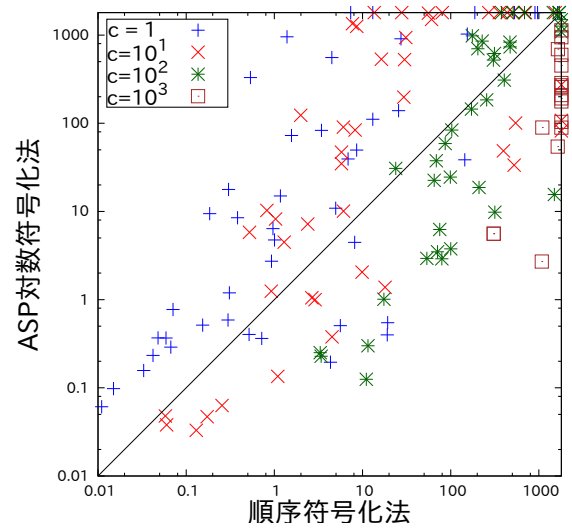


図 4: 実験 2：基礎化時間+求解時間

符号化法は，ドメインサイズが 3 万を超える問題では制限時間内に基礎化が終わっていないことがわかった．以降，この点についてより詳細に考察する．

図 3 に，グラウンダー *gringo* が基礎化に要した時間の分布を示す．縦軸は ASP 対数符号化法，横軸は順序符号化法の基礎化時間を示しており，両軸とも対数スケールである．また，各点は定数倍 c ごとに，4 色に色分けされている．ほとんどの点が対角線の下側にあることから，ASP 対数符号化法は，順序符号化法よりも基礎化時間が大幅に短いことがわかる．また，右端に集中している点群から，順序符号化法はこれらの問題群を制限時間内に基礎化できなかったことがわかる．この原因は，順序符号化法はドメインサイズ d の 2 変数線形制約に対して，$O(d)$ 個の規則を必要とするためと考えられる．

図 4 は，ASP システム *clingo* が要した実行時間の分布である．この実行時間は，グラウンダー *gringo* の基礎化時間と ASP ソルバー *clasp* の求解時間の両方を含んでいる．縦軸，横軸，スケール，色分けは，先に示した図 3 と同様である．図 4 より，$c = 10^2, 10^3$ を示す点が対角線よりも下側に集中していることから，ドメインサイズが大きい問題では，ASP 対数符号化法が順序符号化法よりも優れている．一方，$c = 1, 10^1$ を示す点が上側に集中していることから，ドメインサイズが小さい問題では，順序符号化法が優れている．

5. おわりに

本論文では，ASP で算術制約を効率的に取り扱うために，線形の算術制約を ASP に符号化する方法について述べた．既存の SAT 符号化法を参考に，5 種類の ASP 符号化プログラムを開発した．これにより，直接符号化法，多値符号化法，支持符号化法，対数符号化法，順序符号化法を簡潔に記述できることを確認した．さらに，ASP の特長を生かした ASP 対数符号化法を提案した．この符号化法の特長は，ASP の重み付き個数制約を用いて，線形の算術制約を単一の規則で表現できる点である．二次元矩形パッキング問題を用いた比較実験を行った結果，順序符号化法と ASP 対数符号化法が良いことがわかった．特に，ASP 対数符号化法は，ドメインサイズが大きい (10^5 以上の) 問題に対して，最も優れた性能を示した．さらに，順序符号化法を用いて，二次元矩形パッキングの未解決問題 `GCUT13` について，既知の上限を大きく更新する新しい上限 4907 を求めることに成功した．

関連研究としては，順序符号化法を応用した ASP 型 CSP ソルバーの研究 [*3]．直接，支持，区間，順序符号化法を応用した制約解集合ソルバーの研究 [Drescher 10] がある．今後の課題は，ASP 対数符号化法と順序符号化法を融合したハイブリッド ASP 符号化法を設計・実装することである．

参考文献

[Drescher 10] Drescher, C. and Walsh, T.: A translational approach to constraint answer set solving, *TPLP*, Vol. 10, No. 4-6, pp. 465–480 (2010)

[Gebser 12] Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T.: *Answer Set Solving in Practice*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers (2012)

[Gelfond 88] Gelfond, M. and Lifschitz, V.: The Stable Model Semantics for Logic Programming, in *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pp. 1070–1080, MIT Press (1988)

[井上 08] 井上 克巳, 坂間 千秋：論理プログラミングから解集合プログラミングへ, コンピュータソフトウェア, Vol. 25, No. 3, pp. 20–32 (2008)

[Rossi 06] Rossi, F., Beek, P. v., and Walsh, T.: *Handbook of Constraint Programming*, Elsevier Science Inc., New York, NY, USA (2006)

[Soh 10] Soh, T., Inoue, K., Tamura, N., Banbara, M., and Nabeshima, H.: A SAT-based Method for Solving the Two-dimensional Strip Packing Problem, *Fundamenta Informaticae*, Vol. 102, No. 3-4, pp. 467–487 (2010)

[田村 10] 田村 直之, 丹生 智也, 番原 睦則：制約最適化問題と SAT 符号化, 人工知能学会誌, Vol. 25, No. 1, pp. 77–85 (2010)

*3 `http://www.cs.uni-potsdam.de/aspartame/`