

# ショップ・スケジューリング問題のSAT変換による解法

## Solving Shop Scheduling Problems by SAT encoding

○田村 直之<sup>1</sup>  
神戸大学

<sup>1</sup>Naoyuki Tamura  
Kobe University

tamura@kobe-u.ac.jp

多賀 明子<sup>2</sup>  
神戸大学

<sup>2</sup>Akiko Taga  
Kobe University

鍋島 英知<sup>5</sup>  
山梨大学

<sup>5</sup>Hidetomo Nabeshima  
University of Yamanashi

nabesima@iw.media.yamanashi.ac.jp

番原 睦則<sup>3</sup>  
神戸大学

<sup>3</sup>Mutsunori Banbara  
Kobe University

banbara@kobe-u.ac.jp

井上 克巳<sup>6</sup>  
国立情報学研究所

<sup>6</sup>Katsumi Inoue  
National Institute of Informatics

ki@nii.ac.jp

宋 剛秀<sup>4</sup>  
神戸大学

<sup>4</sup>Takehide Soh  
Kobe University

**Abstract** This paper describes a method to solve Shop Scheduling Problems (SSP) by translating them into Boolean Satisfiability Testing Problems (SAT). The encoding method (named order encoding) is basically the same with the one used to encode Job-Shop Scheduling Problems by Crawford and Baker. Comparison  $x \leq a$  is encoded by a different Boolean variable for each integer variable  $x$  and integer value  $a$ . To evaluate the effectiveness of this approach, we applied the method to the Open-Shop Scheduling Problems (OSP). All 192 instances in three OSP benchmark sets are examined, and our program found and proved the optimal results for all instances including three previously undecided problems.

## 1 はじめに

命題論理の充足可能性判定問題 (SAT 問題) は、与えられた命題論理式の充足可能性を判定する問題である。

近年になって、SAT 問題を解くための非常に高速な SAT ソルバーが実現され、プランニング、マイクロプロセッサの検証、ソフトウェア検証、スケジューリング等の問題について、それらを SAT 問題に変換した後、SAT ソルバーに解かせることにより、元の問題に対する求解を実現する SAT 変換の研究が盛んになっている [7, 3, 9]。この点で、SAT 問題は求解が困難な探索問題に対するアセンブリ言語としての位置付けがなされているといえる。

著者らは、整数上の線形の制約条件からなる制約充足問題および制約最適化問題に対して、Order encoding と呼ばれる SAT 変換手法を提案している [12]。この SAT 変換方法は、Crawford および Baker [3] によりジョブショップ・スケジューリング問題に適用された SAT 変換方法をより一般に

拡張したものである。

本論文では、ショップ・スケジューリング問題を対象として、著者らによる論文 [10, 6, 9, 14, 12, 17] の内容をふまえ、Order encoding による SAT 変換方法およびオープンショップ・スケジューリング問題への適用結果について述べる。

## 2 SAT 問題と SAT ソルバー

命題論理の充足可能性判定問題 (SAT 問題) は、与えられた命題論理式が充足可能 (SAT) であるか充足不能 (UNSAT) であるかを判定する問題である。

判定対象の論理式は、通常 CNF 形式 (Conjunctive Normal Form) で与えられる。すなわち、全体の論理式はいくつかの節 (clause) の連言 (AND) であり、各節はいくつかのリテラル (literal, 命題変数またはその否定) の選言 (OR) となっている。

Davis らによって 1962 年に DPLL アルゴリズムが考案されて以来、SAT 問題を解くための SAT ソルバーの研究は近年長足の進歩を遂げており、

最新の SAT ソルバーは数千万リテラルからなる問題を解くことができる。

SAT ソルバーには系統的に解を探索し SAT/UNSAT を判定する系統的 SAT ソルバーと、確率的に解を探索し SAT のみを判定する確率的 SAT ソルバーの二種類が存在する。2002 年以降ほぼ毎年開催されている SAT competition<sup>1</sup> では、DPLL アルゴリズムに基づいた系統的 SAT ソルバーが上位を占めている。

特に MiniSat [4] は、2005 SAT competition での成功以降、現時点で最速と評価されている系統的 SAT ソルバーである<sup>2</sup>。MiniSat は、C++ で 600 行程度 (コメントを除く) のコード中で、リテラル監視 (watched literals) による高速な伝搬 (propagation) と後戻り、conflict 節の学習による探索空間の枝刈り、backjumping (conflict 解析による後戻り)、決定変数選択のヒューリスティックの導入等により、高速な探索を実現している。また、一旦求解が終了した後も、新たな仮定 (リテラルの集合で与えられる) を追加した元で、続けて求解を行う Incremental search の機能を備えている<sup>3</sup>。この Incremental search では、それ以前の探索中に獲得した学習節をそのまま利用可能であり、類似した SAT 問題について繰り返し求解を行う場合に無駄な探索を削減できる。

### 3 制約充足問題と制約最適化問題

本論文で述べる SAT 変換の対象である制約充足問題 (CSP, Constraint Satisfaction Problem) および制約最適化問題 (COP, Constraint Optimization Problem) は、整数有限領域上の線形のものとする [12]。

また、SAT 変換に関する議論を簡単にするために、整数変数だけでなくブール変数を問題記述中に含めて良い形で以下のように定義する。

**定義 1 (制約充足問題 (CSP))** 制約充足問題は以下を満たす組  $(V, \ell, u, B, S)$  である。

- (1)  $V$  は整数変数の有限集合。

- (2)  $\ell$  は  $V$  から  $\mathbf{Z}$  への関数であり、 $\ell(v)$  は整数変数  $v$  の下限を表す。

- (3)  $u$  は  $V$  から  $\mathbf{Z}$  への関数であり、 $u(v)$  は整数変数  $v$  の上限を表す。

- (4)  $B$  はブール変数の有限集合。

- (5)  $S$  は制約条件の有限集合であり、制約条件の連言を表す。また、各制約条件は整数変数、整数定数、ブール変数、ブール定数 ( $\top, \perp$ ) および以下から構成される論理式である。

論理演算:  $\neg$  (否定),  $\wedge$  (連言),  $\vee$  (選言),  $\supset$  (含意)

算術比較:  $=, \neq, <, \leq, >, \geq$

算術演算:  $+, -, \times$  (定数乗算),  $\text{div}$  (定数除算),  $\text{mod}$  (定数剰余),  $\text{abs}$  (絶対値),  $\text{max}, \text{min}$  □

**定義 2 (制約最適化問題 (COP))** 制約最適化問題は以下を満たす組  $(V, \ell, u, B, S, v)$  である。

- (1)  $(V, \ell, u, B, S)$  は制約充足問題。
- (2) 整数変数  $v \in V$  は目的変数であり、制約条件  $C$  のもと  $v$  の値を最小化することが目的である (ここでは、最小化問題のみを対象とする)。 □

## 4 ショップ・スケジューリング問題

ショップ・スケジューリング問題 (SSP) は、ジョブの集合、マシンの集合およびオペレーションの集合からなり、各オペレーション  $O_i$  ( $1 \leq i \leq n$ ) には正整数の処理時間  $p_i$  が与えられている。また、その型 (フローショップ、ジョブショップ、オープンショップ) に応じてオペレーション間に次の制約条件のいくつかが存在する。

**先行制約:**  $O_i \rightarrow O_j$  と表し、 $O_i$  が  $O_j$  よりも前に完了すべきことを意味する。

**資源制約:**  $O_i \leftarrow O_j$  と表し、 $O_i$  と  $O_j$  が同時に処理されないこと、すなわちどちらかが先に完了すべきことを意味する。

ショップ・スケジューリング問題の目的は、これらの制約条件のもとですべてのオペレーションが完了する総所要時間 (makespan) を最小化することである。

例えば、オープンショップ・スケジューリング問題 (OSP) のベンチマーク gp03-01 は、3 ジョブ、

<sup>1</sup><http://www.satcompetition.org>

<sup>2</sup><http://www.cs.chalmers.se/Cs/Research/>

FormalMethods/MiniSat/

<sup>3</sup>ただし、Incremental search を行うためには、以前の求解の結果が充足可能である必要がある。

$$\begin{aligned}
& m \in [1000, 1509] \\
& s_1 \in [0, 1509] \\
& s_2 \in [0, 1509] \\
& \dots \\
& s_1 + 661 \leq m \\
& s_2 + 6 \leq m \\
& \dots \\
& (s_1 + 661 \leq s_2) \vee (s_2 + 6 \leq s_1) \\
& (s_1 + 661 \leq s_3) \vee (s_3 + 333 \leq s_1) \\
& (s_2 + 6 \leq s_3) \vee (s_3 + 333 \leq s_2) \\
& \dots \\
& (s_1 + 661 \leq s_4) \vee (s_4 + 168 \leq s_1) \\
& (s_1 + 661 \leq s_7) \vee (s_7 + 171 \leq s_1) \\
& (s_4 + 168 \leq s_7) \vee (s_7 + 171 \leq s_4) \\
& \dots
\end{aligned}$$

図 1: SSP を CSP として記述した例  $P$

3 マシンおよび以下の処理時間  $p_i$  を持つ 9 つのオペレーションからなる問題である。

$$\begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{pmatrix} = \begin{pmatrix} 661 & 6 & 333 \\ 168 & 489 & 343 \\ 171 & 505 & 324 \end{pmatrix}$$

また、以下のような資源制約が存在する (OSP には先行制約は存在しない)。

$$\begin{aligned}
O_1 &\longleftrightarrow O_2 & O_1 &\longleftrightarrow O_3 & O_2 &\longleftrightarrow O_3 \\
O_4 &\longleftrightarrow O_5 & O_4 &\longleftrightarrow O_6 & O_5 &\longleftrightarrow O_6 \\
O_7 &\longleftrightarrow O_8 & O_7 &\longleftrightarrow O_9 & O_8 &\longleftrightarrow O_9 \\
O_1 &\longleftrightarrow O_4 & O_1 &\longleftrightarrow O_7 & O_4 &\longleftrightarrow O_7 \\
O_2 &\longleftrightarrow O_5 & O_2 &\longleftrightarrow O_8 & O_5 &\longleftrightarrow O_8 \\
O_3 &\longleftrightarrow O_6 & O_3 &\longleftrightarrow O_9 & O_6 &\longleftrightarrow O_9
\end{aligned}$$

OSP を含む SSP は、各オペレーション  $O_i$  の開始時刻を表す整数変数  $s_i$  および makespan を表す目的変数  $m$  を導入し、以下を制約条件とすることで、制約最適化問題として定式化できる。

- 各  $O_i$  に対し、 $s_i + p_i \leq m$
- 各  $O_i \rightarrow O_j$  に対し、 $s_i + p_i \leq s_j$
- 各  $O_i \longleftrightarrow O_j$  に対し、 $(s_i + p_i \leq s_j) \vee (s_j + p_j \leq s_i)$

また、makespan 値  $m$  および各  $s_i$  の下限と上限も与える必要があるが、これは簡単な計算で求めることが可能である。たとえば gp03-01 の場合、 $m$  の下限と上限を 1000 と 1509、各開始時刻  $s_i$  の下限と上限を 0 と 1509 とすれば良い。

gp03-01 について、makespan 値が 1000 から 1509 までに解が存在するかどうかを判定する CSP

として定式化した例  $P$  の一部を図 1 に示す。ここで、 $x \in [\ell, u]$  等の記述は整数変数  $x$  の下限と上限が  $\ell$  と  $u$  であることを表す。また、 $(s_1 + 661 \leq s_2) \vee (s_2 + 6 \leq s_1)$  等の記述は  $O_1 \longleftrightarrow O_2$  なる資源制約を表しており、他の制約条件も同様である。

## 5 SSP の SAT 変換

CSP の SAT 変換手法としては、Direct encoding が広く知られている [13]。この方法では、各整数変数  $x$  と整数定数  $a$  に対して、 $x = a$  を意味するブール変数を用いるが、制約条件を表すのに多数の節が必要となる。

著者らは、各整数変数  $x$  と整数定数  $a$  に対して、 $x \leq a$  を意味するブール変数を用いる **Order encoding** 法を提案している [12]。この方法は、Crawford および Baker によりジョブショップ・スケジューリング問題の SAT 変換に利用された方法 [3] をより一般に拡張したものである。

論文 [12] では、一般の線形な CSP に対する変換方法を述べているが、本論文では SSP に表れる制約に対してのみ説明を行う。

図 1 の例  $P$  に示されているように、SSP を定式化した CSP に表れる整数変数および制約条件は以下のいずれかのタイプになる (ここで  $x$  と  $y$  は整数変数、 $a$  と  $b$  は正の整数定数)。

- $x \in [\ell, u]$
- $x + a \leq y$
- $(x + a \leq y) \vee (y + b \leq x)$

$x \in [\ell, u]$  については、以下のように変換を行う。まず、整数変数  $x$  および整数定数  $a$  ( $\ell(x) \leq a < u(x)$ ) に対してそれぞれブール変数を用意し、それを  $p(x, a)$  で表す。各ブール変数  $p(x, a)$  は  $x \leq a$  を意味している。次に、整数の順序関係を表す公理として、 $\ell(x) < a < u(x)$  なる各  $a$  について  $(x \leq a - 1) \supset (x \leq a)$  を表す節  $\neg p(x, a - 1) \vee p(x, a)$  を用いる。

例えば  $x \in [0, 4]$  については、 $p(x, 0)$ ,  $p(x, 1)$ ,  $p(x, 2)$ ,  $p(x, 3)$  の 4 つのブール変数を用意し、 $\neg p(x, 0) \vee p(x, 1)$ ,  $\neg p(x, 1) \vee p(x, 2)$ ,  $\neg p(x, 2) \vee p(x, 3)$  の 3 つの節に変換する。

次に  $x + a \leq y$  については、 $x + 2 \leq y$  を例として説明する ( $x \in [0, 4]$ ,  $y \in [0, 4]$  とする)。この制約条件は、以下の 4 つの節  $p(x, 2)$ ,  $\neg p(y, 3) \vee p(x, 1)$ ,  $\neg p(y, 2) \vee p(x, 0)$ ,  $\neg p(y, 1)$  に変換される。これら

```

#  $m \in [1000, 1509]$ 
 $\neg p(m, 1000) \vee p(m, 1001)$ 
...
 $\neg p(m, 1507) \vee p(m, 1508)$ 
#  $s_1 \in [0, 1509]$ 
 $\neg p(s_1, 0) \vee p(s_1, 1)$ 
...
#  $s_1 + 661 \leq m$ 
 $p(s_1, 848)$ 
 $\neg p(m, 1508) \vee p(s_1, 847)$ 
...
#  $(s_1 + 661 \leq s_2) \vee (s_2 + 6 \leq s_1)$ 
 $q_{12} \vee q_{21}$ 
 $\neg q_{12} \vee \neg p(s_2, 1508) \vee p(s_1, 847)$ 
...
 $\neg q_{21} \vee \neg p(s_1, 1508) \vee p(s_2, 1502)$ 
...

```

図 2: SSP を SAT 変換した例  $S$

はそれぞれ  $x \leq 2$ ,  $(y \leq 3) \supset (x \leq 1)$ ,  $(y \leq 2) \supset (x \leq 0)$ ,  $\neg(y \leq 1)$  を表している.

$(x + a \leq y) \vee (y + b \leq x)$  については,  $(x + 2 \leq y) \vee (y + 3 \leq x)$  を例として説明する ( $x \in [0, 4]$ ,  $y \in [0, 4]$  とする). 比較のそれぞれを変換した結果を展開して CNF 形式を求めた場合, 数多くの節が生成されてしまう. そこで, 各比較に対応する新しいブール変数  $q_1$  と  $q_2$  を導入し,  $q_1 \supset (x + 2 \leq y)$  と  $q_2 \supset (y + 3 \leq x)$  のそれぞれを変換する. すなわち, 以下の 8 つの節に変換する:  $q_1 \vee q_2$ ,  $\neg q_1 \vee p(x, 2)$ ,  $\neg q_1 \vee \neg p(y, 3) \vee p(x, 1)$ ,  $\neg q_1 \vee \neg p(y, 2) \vee p(x, 0)$ ,  $\neg q_1 \vee \neg p(y, 1)$ ,  $\neg q_2 \vee p(y, 1)$ ,  $\neg q_2 \vee \neg p(x, 3) \vee p(y, 0)$ ,  $\neg q_2 \vee \neg p(x, 2)$ .

図 2 に図 1 の CSP 例  $P$  を SAT 変換した結果  $S$  の一部を示す. # から始まる行は変換前の制約条件を表すコメントである.

この SAT 問題  $S$  の充足可能性と, 元の問題 gp03-01 が makespan 値  $1000 \leq m \leq 1509$  の範囲でスケジュール可能かどうか一致する.

## 6 最適解の探索

前節では, 与えられた SSP に対して与えられた makespan の範囲内でスケジュール可能かどうかを SAT 問題に変換して判定する方法を述べた.

この判定を makespan の範囲を二分しながら繰り返せば, 最適な makespan 値を求めることができる [10, 6]. 単純には, 各範囲毎に SAT 変換を行う必要があると思われるが, 以下のようにすれば

表 1: 最適値の二分探索

下限 $\ell$	上限 $u$	平均 $h$	$S \cup \{p(m, h)\}$ の結果
1000	1509	1254	SAT
1000	1254	1127	UNSAT
1128	1254	1191	SAT
1128	1191	1159	UNSAT
1160	1191	1175	SAT
1160	1175	1167	UNSAT
1168	1175	1171	SAT
1168	1171	1169	SAT
1168	1169	1168	SAT

一度だけの SAT 変換で十分である.

- (1) 最初に与えられた makespan の下限  $\ell$ , 上限  $u$  について SAT 変換を行った結果を  $S$  とする.
- (2)  $\ell$  と  $u$  の算術平均の整数値への切捨てを  $h$  とする.
- (3) makespan 値  $m \leq h$  なる条件を追加した SAT 問題  $S \cup \{p(m, h)\}$  について, 充足可能性を SAT ソルバーで判定する.
- (4) 充足可能ならば  $u := h$  とし, 充足不能ならば  $\ell := h + 1$  とする.
- (5)  $\ell = u$  ならばその値が最適値である.  $\ell < u$  ならば (2) に戻る.

ベンチマーク問題 gp03-01 に対して, 最初の下限  $\ell = 1000$ , 上限  $u = 1509$  としてこの二分探索手順を実行した結果を表 1 に示す. 9 回の判定により最適値 1168 が求められている.

以上のように, 最適値を決定するまでには, 同様の SAT 問題を連続して解く必要がある. 論文 [9, 14] では, SAT ソルバーの Incremental search 機能により SAT ソルバーを連続して動作させ, 学習節の再利用を実現することで最適値の探索速度を向上させる手法を提案している. これに加えて, 探索過程において判明した範囲の条件を節として追加することにより, さらに速度を向上させることも可能である [15].

また, 論文 [10, 6] では, 複数の異なったタイプの SAT ソルバーを並行かつ協調的に動作させることにより, 探索速度を向上させる方法を提案している. 特に, 系統的 SAT ソルバーと確率的 SAT ソルバーを組み合わせた場合, 充足可能な場合には確率的 SAT ソルバーが有効に働き, 充足不能な場合には系統的 SAT ソルバーが有効に働くことが期待できるため, 単一のタイプの SAT ソルバーを利用するよりも高速な求解を実現可能である.

表 2: OSS ベンチマーク問題 17 問に対する結果

Instance	Optim.	CPU (seconds)	SAT	
			Variables	Clauses
j7-per0-0	<b>1048</b>	140967	85887	1051419
j7-per0-1	1055	428	109837	1380492
j7-per0-2	1056	292	113537	1431330
j7-per10-0	1013	332	108687	1368170
j7-per10-1	1000	121	107087	1347411
j7-per10-2	1011	1786	93887	1165467
j7-per20-0	1000	66	95487	1193523
j7-per20-1	1005	132	125087	1595847
j7-per20-2	1003	132	107987	1361349
j8-per0-1	<b>1039</b>	102847	145495	2118473
j8-per0-2	1052	870	177995	2630988
j8-per10-0	1017	2107	168310	2481679
j8-per10-1	1000	8346	140620	2047787
j8-per10-2	<b>1002</b>	7789	136655	1984646
j8-per20-0	1000	148	139255	2030756
j8-per20-1	1000	136	149265	2191364
j8-per20-2	1000	144	145300	2125157

## 7 性能評価

本論文で述べた Order encoding による SAT 変換を実現したシステムとして、CSP2SAT および Sugar を開発・公開している<sup>4</sup>。

本節では、OSP のベンチマーク問題 192 問 [5, 11, 2] に対して、CSP2SAT を Xeon 2.8GHz メモリ 4GB の PC 上で実行した結果について述べる。CSP2SAT は、SAT 変換の部分は Perl で記述されており、SAT ソルバーとしては MiniSat [4] を利用している。

結果として、過去 [1, 8] に最適値が未決定だった 3 問を含めて、192 問のすべてについて最適値を求めることができた。なお、未決定問題に対する新しい結果は、当初、10 台の PC を用いた Grid システム上でソルバーを並列動作させることで求められたものである [16, 12, 17]。

表 2 に、192 問のうち最も困難な 17 問について、1 台の PC で実行した結果を示す (学習節の再利用はしていない)。j7-\* は 7 ジョブ、7 マシン、j8-\* は 8 ジョブ、8 マシンとなっている。表中、欄 Instance はベンチマーク問題名、欄 Optim. は求めた最適値、欄 CPU は実行に要した CPU 秒 (SAT 変換および SAT ソルバーの実行時間を含む)、欄 SAT は変換後の SAT 問題の変数および節の数を表している。なお、太文字はこれまで最適値が未決定だった問題である。

特に j8-per10-2 については、これまで知られ

$$\begin{pmatrix} 247 & 296 & 110 & 618 & 537 & 31 & 500 & 127 \\ 815 & 50 & 328 & 274 & 311 & 672 & 550 & 6 \\ 1 & 583 & 120 & 339 & 876 & 842 & 675 & 58 \\ 293 & 669 & 5 & 72 & 250 & 502 & 403 & 994 \\ 286 & 517 & 870 & 594 & 612 & 347 & 0 & 297 \\ 404 & 252 & 73 & 28 & 83 & 25 & 300 & 734 \\ 707 & 997 & 560 & 12 & 48 & 87 & 842 & 340 \\ 53 & 6 & 703 & 285 & 342 & 872 & 526 & 547 \end{pmatrix}$$

図 3: OSS 問題 j8-per10-2 の最適解

表 3: 学習節の再利用の効果

学習節の再利用	解けた問題数	平均実行時間
無	184 問	35.35 秒
有	185 問	20.25 秒

ていた最良解が 1009 [8]、決定した最適値は 1002 であり、大幅に改善されている。図 3 に、得られた最適解の開始時間を示す。

また、学習節の再利用による効果 [9, 14, 15] を評価するために、同一の OSS 問題 192 問について、タイムアウトを 600 秒として実行した結果を表 3 に示す。学習節の再利用により解けた問題数は 1 問増加しており、平均実行時間は 6 割以下に減少している。なお、表中の平均実行時間には、SAT 変換の時間および SAT 問題の読み込み時間は含まれていない (学習節の再利用を行わない場合、何度も SAT 問題を読み込む必要がある)。含めた場合の値は 60.88 秒および 22.24 秒となる。

## 8 おわりに

本論文では、ショップ・スケジューリング問題を対象として、論文 [10, 6, 9, 14, 12, 17] で述べた内容をふまえ、Order encoding による SAT 変換方法およびオープンショップ・スケジューリング問題への適用結果について報告した。

オープンショップ・スケジューリング問題については、過去 [1, 8] に最適値が未決定だった 3 問を含めて、192 問のすべてについて最適値を求めることができた。なお、未決定問題に対する新しい結果は、当初、10 台の PC を用いた Grid システム上でソルバーを並列動作させることで求められたものである [16, 12, 17]。

<sup>4</sup><http://bach.istc.kobe-u.ac.jp/csp2sat/>

このように、Order encoding は、従来から用いられている Direct encoding と比較して、整数上の制約条件をより自然かつ簡潔に表現でき、SAT ソルバーでより高速に求解することが期待できる。

実際、第 2 回 CSP solver competition<sup>5</sup>の N-ARY-INT 部門 (N 変数からなる整数制約式の部門) において、Order encoding に基づく Sugar ソルバーは第 4 位となっており、他の SAT 変換型ソルバーよりも優れた成績であった。また、特にショップ・スケジューリング問題について、他のすべての CSP ソルバーが制限時間内で解けない問題の求解に成功しており、同様のタイプの問題に対して有効な方法といえる。

## 参考文献

- [1] Christian Blum. Beam-ACO — hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Computers & OR*, Vol. 32, pp. 1565–1591, 2005.
- [2] Peter Brucker, Johann Hurink, Bernd Jurisch, and Birgit Wöstmann. A branch & bound algorithm for the open-shop problem. *Discrete Applied Mathematics*, Vol. 76, No. 1–3, pp. 43–59, 1997.
- [3] James M. Crawford and Andrew B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pp. 1092–1097, 1994.
- [4] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003)*, pp. 502–518, 2003.
- [5] Christelle Guéret and Christian Prins. A new lower bound for the open-shop problem. *Annals of Operations Research*, Vol. 92, pp. 165–183, 1999.
- [6] Katsumi Inoue, Takehide Soh, Seiji Ueda, Yoshito Sasaura, Mutsunori Banbara, and Naoyuki Tamura. A competitive and cooperative approach to propositional satisfiability. *Discrete Applied Mathematics*, Vol. 154, No. 16, pp. 2291–2306, Nov. 2006.
- [7] Henry A. Kautz, David A. McAllester, and Bart Selman. Encoding plans in propositional logic. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pp. 374–384, 1996.
- [8] Philippe Laborie. Complete MCS-based search: Application to resource constrained project scheduling. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pp. 181–186, 2005.
- [9] Hidetomo Nabeshima, Takehide Soh, Katsumi Inoue, and Koji Iwanuma. Lemma reusing for SAT based planning and scheduling. In *Proceedings of the International Conference on Automated Planning and Scheduling 2006 (ICAPS'06)*, pp. 103–112, 2006.
- [10] Takehide Soh, Katsumi Inoue, Mutsunori Banbara, and Naoyuki Tamura. Experimental results for solving job-shop scheduling problems with multiple SAT solvers. In *Proceedings of the 1st International Workshop on Distributed and Speculative Constraint Processing (DSCP'05)*, October 2005.
- [11] Éric D. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, Vol. 64, pp. 278–285, 1993.
- [12] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara. Compiling finite linear CSP into SAT. In *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP 2006)*, pp. 590–603, November 2006.
- [13] Toby Walsh. SAT v CSP. In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP 2000)*, pp. 441–456, 2000.
- [14] 鍋島英知, 宋剛秀, 井上克巳, 岩沼宏治. 効率的な SAT プランニングと SAT スケジューリングのための補題再利用. 電子情報通信学会技術研究報告, Vol. 106, No. 38, pp. 18–24, 2006.
- [15] 志賀彰. 制約最適化問題の SAT 変換による解法. Master's thesis, 神戸大学大学院自然科学研究科, 2006.
- [16] 多賀明子. ショップスケジューリング問題の SAT 変換とグリッド計算環境での実験. 神戸大学工学部情報知能工学科卒業論文, 2005.
- [17] 多賀明子, 田村直之, 北川哲, 番原睦則. グリッド計算環境上でのショップ・スケジューリング問題の SAT 変換による解法. スケジューリング・シンポジウム 2007. (発表予定).

<sup>5</sup><http://www.cril.univ-artois.fr/CPAI06/>