特　集

SATによるプランニングとスケジューリング

SAT Planning and SAT Scheduling

鍋島 英知
Hidetomo Nabeshima
山梨大学大学院医学工学総合研究部
Department of Research Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi
nabesima@yamanashi.ac.jp

keywords: planning, scheduling, incremental SAT solving

1. はじめに

命題論理の充足可能性判定 (SAT) 問題を解くソルバーの飛躍的な性能向上は，SAT 符号化に基づく問題解決手法の研究を大きく活性化させている．その発端となったのがプランニング問題を SAT ソルバーを利用して解く SAT プランニング (SAT planning) である．

Kautz と Selman は，STRIPS 形式の古典的プランニング問題が SAT 問題に符号化できることを示し，それを汎用の高速な SAT ソルバーで解くことで効率良くプランを求めることができることを示した [Kautz 92, Kautz 96]．この手法はその後，プランの探索空間を荒く絞り込むプランニンググラフを利用した探索手法 [Blum 97] と組み合わされ [Kautz 98]，古典的プランニング問題に対する有効な解法として広まった．例えば，プランナーの性能を競う 2004 年と 2006 年の国際プランニング競技会 (international planning competition; IPC) [*1] では，古典的プランニング問題から最適プランを求める部門において，この手法に基づく SAT ベースのプランナーが優勝している．

一方，プランの最適性にこだわらない場合（すなわち準最適なプランを求める場合）はヒューリスティックスに基づくプランナーが優れた性能を示している [Bonet 01]．しかし近年，SAT ベースの手法においても，緩和したプランの定義とプラン長に関する投機的な探索戦略を用いることで，ヒューリスティックスベースの手法に匹敵する性能を持つことが示されている [Rintanen 06]．また古典的プランニング問題を拡張し，資源や時間に関する数値制約を付加した問題に対しては，SMT (SAT modulo theories) を利用した手法が用いられてきたが [Wolfman 99, Shin 05]，最近では数値変数のドメインを絞り込んだ後で SAT へ符号化する優れた手法も報告されている [Hoffmann 07]．2008 年の IPC からは，単純な古典的プランニング問題を解く部門はなくなり，アクションにコストを付加し，最小コストのプランを求める問題へと拡張されている．この部門においても SAT ベースのプランナーが参戦しているが [Chen 08, Robinson 08]，残念ながら現在のところヒューリスティックスベースのプランナーが優勢である．

最小コストのプランを求める問題は，スケジューリング問題に非常に近い．スケジューリング問題は，与えられた目的関数を最適化するように，有限の資源を適切なタイミングでタスクに割り当てる問題である．これまで，様々な制約下での充足解や最適解を探索するために，分枝限定法に代表される探索アルゴリズムを中心に研究が進められてきた．その一方で，Crawford と Baker は，ショップ・スケジューリング問題に対する SAT 符号化法を示し，その中で整数変数に対する範囲制約の効率的な表現方法を与えている [Crawford 94]． この符号化法に基づき，田村らと越村らは，それぞれそれまで未知であったいくつかのオープンショップ・スケジューリング問題とジョブショップ・スケジューリング問題の最適値決定に成功している [田村 07, 多賀 07, 越村 09]．

本稿では，まず SAT プランニングにおける最適・準最適プランを求める手法の概要と最近の進展を示し，その後，SAT スケジューリング手法としてショップ・スケジューリング問題に対する符号化法の概要を紹介する．これらの手法においては，解を得るまでに複数の SAT 問題を解く必要があるが，そのような場合に有効な SAT ソルバーを用いたインクリメンタル探索についても紹介する．

2. SAT プランニング

2·1 プランニング問題とプラン

STRIPS 形式の古典的プランニング問題を示すために諸定義を行う．命題 f に対し，f またはその否定 $\neg f$ を**リテラル**といい，前者を**正リテラル**，後者を**負リテラル**と呼ぶ．$\neg\neg f = f$ である．S を状態を表す命題の集合とする．**状態** s は，すべての命題 $f \in S$ について，正または負のリテラルのいずれかを含む集合と定義する．すなわち $s = P \cup \{\neg f \mid f \in S \setminus P\}$，ここで $P \subseteq S$ である．**アクション**を $\langle p, e \rangle$ で定義する．p, e は，それぞれアクションの**前提条件**と**効果**を表すリテラルの集合であり，S 中

[*1] http://ipc.icaps-conference.org/

の命題から構成される．しばしばリテラルの集合をリテラルの連言と同一視する．アクション $a = \langle p, e \rangle$ が状態 s において**実行可能** (executable) であるとは $p \subseteq s$ かつ e が無矛盾な場合（互いに補なリテラルを含まない場合）をいう．このとき，状態 s でアクションを実行した後の状態 $\mathit{exec}_a(s)$ を以下で定義する：

$$\mathit{exec}_a(s) = e \cup \{f \in s \mid f \notin e \wedge \neg f \notin e\}$$

もし a が s において実行可能でない場合，$\mathit{exec}_a(s)$ は定義されない．アクションの列 $a_1; a_2; \ldots; a_n$ に対し，$\mathit{exec}_{a_1;a_2;\ldots;a_n}(s) = \mathit{exec}_{a_n}(\cdots \mathit{exec}_{a_2}(\mathit{exec}_{a_1}(s))\cdots)$ とする．

プランニング問題 π を $\langle S, I, A, G \rangle$ で定義する．S は状態を表す命題の集合，I は初期状態，A はアクションの集合，G は目標条件を表すリテラルの集合である．

プランニング問題 $\pi = \langle S, I, A, G \rangle$ に対し，アクション列 $a_1; a_2; \ldots; a_n$ が π の**逐次プラン** (sequential plan) であるとは，$G \subseteq \mathit{exec}_{a_1;a_2;\ldots;a_n}(I)$ であるときをいう．逐次プランでは任意の 2 つのアクションの間に順序関係が成り立つが，その一方で，いくつかのアクションについて実行順序を定める必要のない場合がある（つまりどのアクションから実行しても良い，または同時に実行しても良い）．そのようなプランを**並列プラン** (parallel plan) と呼び，ここでは Rintanen らの定義 [Rintanen 06] に従い，2 種類の並列プランを示す．

1 つ目の並列プランは，もともと**半順序プラン** (partially ordered plan) と呼ばれていたものである．本稿では **∀-ステッププラン** (∀-step plan) と呼ぶ．プランニング問題 $\pi = \langle S, I, A, G \rangle$ に対し，アクション集合の列 $\langle A_0, \ldots, A_{l-1} \rangle \in (2^A)^l$ $(l \geq 0)$ が **∀-ステッププラン**であるとは，以下の条件を満たす状態の列 $s_0, \ldots, s_l$ が存在するときをいう[*2]：

(1) $s_0 = I$.

(2) 各 $i \in \{0, \ldots, l-1\}$ について，A_i に含まれるすべてのアクションの任意の並び $a_1; \ldots; a_n$ に対し，$\mathit{exec}_{a_1;\ldots;a_n}(s_i) = s_{i+1}$.

(3) $G \subseteq s_l$.

すなわち ∀-ステッププランとは，各 A_i に含まれるアクションを任意の順序で実行して良いプランである[*3]．

2 つ目の並列プランは，∀-ステッププランの条件 (2) を緩和したものである．アクション集合の列 $\langle A_0, \ldots, A_{l-1} \rangle \in (2^A)^l$ $(l \geq 0)$ が **∃-ステッププラン** (∃-step plan) であるとは，以下の条件を満たす状態の列 $s_0, \ldots, s_l$ が存在するときをいう（条件 (1)(3) は ∀-ステッププランと同じ）：

(2') 各 $i \in \{0, \ldots, l-1\}$ について，A_i に含まれるすべてのアクションの少なくとも 1 つの並び $a_1; \ldots; a_n$ に対し，$\mathit{exec}_{a_1;\ldots;a_n}(s_i) = s_{i+1}$.

並列プランにおける各 A_i を**ステップ**と呼ぶ．

〔例 1〕 [Rintanen 06] ロシアの入れ子人形（マトリョーシカ人形，大きな人形の中に小さな人形が入っている）を考える．ここでは 4 重の入れ子を考える．

$$a_1 = \langle \mathrm{out}_1 \wedge \mathrm{out}_2 \wedge \mathrm{empty}_2, 1\mathrm{in}2 \wedge \neg\mathrm{out}_1 \wedge \neg\mathrm{empty}_2 \rangle$$
$$a_2 = \langle \mathrm{out}_2 \wedge \mathrm{out}_3 \wedge \mathrm{empty}_3, 2\mathrm{in}3 \wedge \neg\mathrm{out}_2 \wedge \neg\mathrm{empty}_3 \rangle$$
$$a_3 = \langle \mathrm{out}_3 \wedge \mathrm{out}_4 \wedge \mathrm{empty}_4, 3\mathrm{in}4 \wedge \neg\mathrm{out}_3 \wedge \neg\mathrm{empty}_4 \rangle$$

例えば a_1 は，最も小さな人形を 2 番目に小さな人形の中にしまうアクションである．4 体の人形を入れ子にするための ∀-ステッププランは $\langle \{a_1\}, \{a_2\}, \{a_3\} \rangle$ となる．一方，∃-ステッププランは $\langle \{a_1, a_2, a_3\} \rangle$ となる．これはアクションの並びの 1 つである $a_1; a_2; a_3$ によって目標を達成できるためである．しかし他の並びでは達成できないため，∀-ステッププランは結果として 3 つのステップからなる．

STRIPS 形式の古典的なプランニング問題の場合，プラン長が最も短いプランを最適なプランとするのが一般的である．逐次プランの場合は，最適プランはアクション数が最も少ないプランとなる．一方，並列プランの場合は，1 つのステップに複数のアクションが含まれるため，ステップ数を最小化する考え方と，プランに含まれる総アクション数を最小化する考え方がある．2006 年までの IPC では，古典的プランニング問題における最適プランの定義として前者が用いられてきた．2008 年からは，アクションにコストを付加し，プランのコストを最小化することを目的としている．従って，すべてのアクションのコストが等しいと考えれば，後者のアクション数を最小とする定義に相当する．

以下では，まず 2·2 節で最短ステップの ∀-ステッププランを求める手法を示し，次に 2·3 節で準最適な ∃-ステッププランを生成する手法を紹介する．2·4 節と 2·5 節では，それぞれ最小コストのプランを求める手法と数値変数を含むプランニング問題に対する手法を紹介する．

2·2 最短ステップの ∀-ステッププランの探索

最短ステップの ∀-ステッププランを求める SAT プランナーの代表は Blackbox [Kautz 98] である．2004, 2006 年の IPC で優勝したプランナーも Blackbox の手法をベースにしている．Blackbox の詳細については，文献 [鍋島 01] に解説があるのでそちらを参照されたい．ここでは概略のみを示す．基本的な手続きは以下の通りである．プランニング問題を $\pi = \langle S, I, A, G \rangle$ とする．

(i) プランのステップ数を $l = 1$ と仮定する．

(ii) ステップ数 l までに到達できる可能性のある状態および実行できる可能性のあるアクションを列挙する．簡単には次のようになる．各 $i \in \{0, \ldots, l-1\}$ に対して以下を求める．ここで A_i はステップ i で実行できる可能性のあるアクションの集合，S_i はス

---

*2 本稿ではアクションの並びを ; で区切って表し，アクション集合の並びを , で区切って表す．

*3 本稿のアクションの定義では条件付き効果 (conditional effects) を考えないため，$\mathit{exec}_{a_1;\ldots;a_n}(s_i)$ はすべてのアクションを同時に実行した結果 $\bigcup_{a \in A_i} \mathit{exec}_a(s_i)$ に等しい．アクションが条件付き効果を含む場合は，任意の順序で実行できても同時に実行できるとは限らない [Rintanen 06]．ここでは紙面の都合上，アクションに条件付き効果を持たせていない．

テップ i で真にできる可能性のあるリテラルの集合（状態ではない）である．$S_0 = I$ とする．

$$A_i = \{a \in A \mid a = \langle p, e\rangle, p \subseteq S_i\}$$
$$S_{i+1} = S_i \cup \bigcup_{\langle p,e\rangle \in A_i} e$$

実際にはプランニンググラフ [Blum 97] を利用し，特定のステップ数では明らかに同時に成立しない命題や並列に実行きないアクションをチェックしておく．

(iii) $G \not\subseteq S_l$ ならば l に 1 を加え (ii) に戻る（すなわち目標条件を達成できる可能性が得られるまでステップを進める）．

(iv) $S_0, \ldots, S_l$, $A_1, \ldots, A_{l-1}$ を用いて，プランニング問題を SAT 問題 P_l に変換する．詳細は省くが，プランニンググラフを SAT に直接符号化する手法が示されている [Kautz 96].

(v) 汎用の SAT ソルバーを用いて P_l を解く．もし P_l が充足可能であれば，その真偽値割り当てからステップ数 l のプランを抽出し終了する．充足不能であれば l に 1 を加え，(ii) に戻る．

ステップ (ii) の命題またはアクションの到達可能性のチェックでは，同一ステップに含まれる 2 つの命題（またはアクション）が**相互排他関係** (mutual exclusion relation; mutex) にあるかどうかを調べ，それを利用した絞り込みが行われてきた [Blum 97]．Chen らは，離れたステップに含まれる命題（アクション）間における相互排他関係を抽出する手法を提案し，実行時間を大きく改善できることを示している [Chen 09].

上に示した手続きでは徐々にステップ数を増加させていく．この場合，充足可能な SAT 問題に出会うまで充足不能な問題を解き続けることになる．充足不能を示すためには全解探索を必要とするため，充足可能な問題を解くよりも，一般により多くの時間を必要とする．2006 年の IPC で優勝した MaxPlan [Xing 06] では，最初にヒューリスティックスベースのプランナーにより準最適なプラン（ステップ数が最小ではないプラン）を求めておき，充足不能な問題に出会うまでステップ数を 1 ずつ減らしていく戦略を採っている．

以降では Rintanen らによるプランニング問題から SAT への符号化法を示す [Rintanen 06]．この手法ではプランニンググラフを利用しないが，同様の探索空間の絞り込みを行うことで，Blackbox に匹敵する性能を持つことが報告されている[*4]．

プランニング問題を $\pi = \langle S, I, A, G\rangle$ とし，プランのステップ数を l と仮定する．初期状態から数えて t 番目の状態を示すために t を時刻と呼ぶ $(0 \le t \le l)$．各アクション $a \in A$ に対し，時刻 t において a が実行されたかどうかを表す命題変数 a^t を用意する．同様に，状態を表す各命題 $f \in S$ に対し，時刻 t における真偽値を表す命題変数 f^t を用意する．リテラルの集合 q に対し，q^t は q 中のリテラルに添え字 t を付加したものとする．2 つのアクション $a_1, a_2 \in A$ について，a_1 と a_2 が**干渉する** (interfere) とは，片方の前提条件にリテラル f が含まれ，かつ，他方の効果に $\neg f$ が含まれる場合をいう．

SAT 問題 P_l を以下の変換規則に従って生成する．

(a) 初期状態は時刻 0 において真であり，目標条件は時刻 l において真である．

$$I^0 \wedge G^l$$

(b) 各時刻 $t \in \{0, \ldots, l-1\}$ と各アクション $a \in A$ に対し，時刻 t においてアクション $a = \langle p, e\rangle$ が実行されたならば，その前提条件 p は時刻 t で真であり，効果 e は時刻 $t+1$ で真である．

$$a^t \rightarrow p^t \wedge e^{t+1}$$

(c) 各時刻 $t \in \{0, \ldots, l-1\}$ と各命題 $f \in S$ に対し，時刻 t から $t+1$ にかけて f の真偽値が変化したならば，それを効果に含むアクションが少なくとも 1 つ実行されることを意味する．ここで f または $\neg f$ を効果に含むアクションの集合を，それぞれ $\{a_1, \ldots, a_n\}$ と $\{b_1, \ldots, b_m\}$ とし，以下の式を追加する．

$$(\neg f^t \wedge f^{t+1}) \rightarrow (a_1^t \vee \ldots \vee a_n^t)$$
$$(f^t \wedge \neg f^{t+1}) \rightarrow (b_1^t \vee \ldots \vee b_m^t)$$

(d) 各時刻 $t \in \{0, \ldots, l-1\}$ と互いに干渉する 2 つのアクション a_1, a_2 のすべてのペアに対し，時刻 t において両方とも実行されることはないことを意味する以下の式を追加する．

$$\neg a_1^t \vee \neg a_2^t$$

上記変換規則より得られる SAT 問題 P_l がもし充足可能ならば，かつそのときに限りステップ数 l の ∀-ステッププランが存在する．上記変換規則 (a)〜(c) より生成される式のサイズは，ステップ数 l とアクション数と状態を表す命題の数に対して線形に比例するが，規則 (d) についてはアクション数の 2 乗に比例する．Rintanen らはアクション数に対して線形に比例するよりコンパクトな符号化法を提案している [Rintanen 06].

2·3 ∃-ステッププランの探索

∀-ステッププランでは，各ステップに含まれるアクションを任意の順序で実行しても目標条件を達成できる．一方，もし任意の順序で実行可能でない場合は複数のステップに分割されることになるため，ステップ数の増大を招く．結果として，SAT 問題のサイズが大きくなり求解までに多くの時間を要することがある．

∀-ステッププランを緩和した概念は Dimopoulos らによって提案され [Dimopoulos 97]，Rintanen らによって詳細な定式化が行われた [Rintanen 06]．まずアクション

*4 ここで示す符号化は，[Ernst 97] における regular explanatory 符号化に等しい．[Rintanen 06] では，条件付き効果 (conditional effects) を含むより一般化した形式を扱っている．

間の干渉関係を緩和した概念を定義する．2 つのアクション $a_1, a_2 \in A$ について，a_1 が a_2 に**影響する** (affect) とは，a_1 の効果にリテラル f が含まれ，かつ，a_2 の前提条件に $\neg f$ が含まれる場合をいう．

∃-ステッププランを求めるため，すべてのアクションの上に適当な全順序 $\prec$ を事前に仮定する．実行時には，各ステップ A_i 中のアクションをこの順序に従って実行するものとする．そして符号化において，アクション $a \in A_i$ は，後続する任意のアクション $a' \in A_i$ $(a \prec a')$ に影響を与えてはならないとする．すなわち a' が a に影響を与えるとしても，実行順序が a の後であるならば，a' を同一ステップに含めても良いとする．ここが ∀-ステッププランとの違いである．

SAT 問題への変換方法は，規則 (a)〜(c) までは ∀-ステッププランと同じであり，(d) のみを以下と置き換える．

(d') 各時刻 $t \in \{0, \ldots, l-1\}$ と任意の 2 つのアクション a_1, a_2 $(a_1 \prec a_2)$ に対し，a_1 が a_2 に影響を与えるならば，a_1 と a_2 は時刻 t において両方とも実行されることはないことを意味する以下の式を追加する．

$$\neg a_1^t \vee \neg a_2^t$$

変換規則 (a)〜(c)(d') より得られる SAT 問題 P_l がもし充足可能であるならば，長さ l の ∃-ステッププランが存在する．アクションの実行順序を事前に仮定しているため，逆は成立しない．つまり最短ステップの ∃-ステッププランが求まるとは限らない．例 1 では，実行順序を $a_1 \prec a_2 \prec a_3$ とすれば，規則 (d') は何も式を生成しないため，結果としてステップ数 1 のプランが求まる．しかし，もし $a_3 \prec a_2 \prec a_1$ とした場合は，∀-ステッププランの規則 (d) と等しい式が生成されることになる．

∃-ステッププランでは，各ステップ中のアクションをある固定された順序において影響なく実行できればよい．∀-ステッププランのように任意の順序において干渉なく実行できることを保証せずに済むため，プランのステップ数が大幅に小さくなり，SAT 問題のサイズも小さくなる．結果としてプランを非常に高速に求めることができる [Rintanen 06]．この手法は，実行可能なプランを高速に求めたい場合に有望な手法であるといえる．

さらに最短ステップのプランにこだわらないのであれば，効率良く充足可能な SAT 問題を発見するためのステップ数の探索戦略が提案されている [Rintanen 06]．SAT プランニングでは，充足可能な問題の直前の充足不能な問題の証明にしばしば多大な時間を要する．図 1 はその一例を示している．横軸は時刻（ステップ数）を表し，縦軸が実行時間である．この例では時刻 71 までが充足不能，以降が充足可能である．各時刻 i における SAT 問題 P_i は独立に生成可能なので，適当な上限までの SAT 問題 $P_1, P_2, \ldots, P_n$ を生成し，それらを同時に並列に解くことを考える（複数のプロセッサを利用しても良いし，時分割して実行しても良い）．最短ステップ数から離れるほど一般に問題は簡単になるので，並列に解くことにより，P_{71} のような難しい充足不能な問題を証明することなく，充足可能な問題を素早く発見することが可能になる．さらに各 P_i を解くソルバーに割り当てる実行時間を，i が小さいほど多くの時間を割り当てるようにすれば（図 1 の曲線），最短ステップ数に比較的近いところで充足可能な問題を発見することが可能になる．

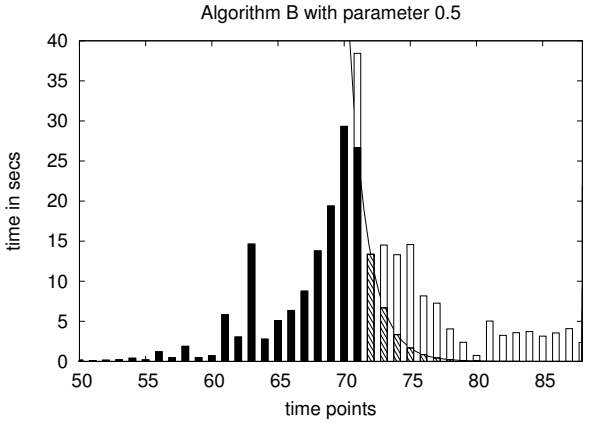


図 1 SAT ソルバーの実行時間分布の例 [Rintanen 06]

2·4 最小コストプランの探索

アクションにコストを付加したプランニング問題から，最小コストのプランを求める SAT ベースの手法を簡単に紹介する．ステップ数の最小化では，ステップ長を 1 ずつ延ばしていき，充足可能な SAT 問題を発見した時点で終了することができる．しかしコストの最小化では，例えあるステップ長 l における最小コストのプランを求めたとしても，より長いステップ数においてよりコストの小さなプランが存在するかも知れない．

Plan-A [Chen 08] は，あるステップ数以下の最小コストのプランを求める SAT ベースの準最適プランナーである．Plan-A では拡張した SAT 問題を扱う．各命題変数に非負整数のコストを付加し，SAT 問題のモデルの中で，真を割り当てられた命題変数のコストの総和が最小になるモデルを求める．このために CDCL (conflict-driven clause learning) ベースの SAT ソルバー*5に対して，以下の改良を行う．ここで $c_{\min}$ を最小のコストを表す変数とする．最初は $c_{\min} = \infty$ である．

- 通常 SAT ソルバーはモデルを 1 つ発見するとそれを返して終了するが，最小コストのモデルを求めるために全解探索を行うようする．具体的には，モデルが発見されるたびに，そのモデルの否定を節として SAT 問題に追加し，再度解き直す．最終的に充足不能になれば終了である．またモデルが発見されるたびに，そのコストが $c_{\min}$ 以下ならば $c_{\min}$ を更新する．
- コストは非負整数で与えられるため，探索過程で得

*5 CDCL ソルバーについては本特集 [鍋島 10] を参照されたい．

られる部分的なモデルのコストが $c_{\min}$ を超えるならば以降の探索は無駄である．そこで (1) と同様に部分モデルの否定を節として追加する．

以上の改良により，最小コストのモデルを求めることができる．Plan-A の基本的な手続きは Blackbox と同様である．ただし SAT 符号化において，アクションを表す命題変数に，そのアクションのコストを付加する点が異なる．また，もし充足可能な SAT 問題を発見したとしても，指定されたステップ長まで最小コストのプランを求めて探索を継続する．

CO-Plan [Robinson 08] は，最小コストのプランを求める最適プランナーである．その特徴は，前処理として SAT プランニングを利用する点にある．まず Plan-A と同様の手法によって，最小ステップ数かつ最小コストのプランを求めておく．このときのコストを上限として，前向き状態空間探索アルゴリズムをあらためて実行し，最小コストのプランを求める．つまり，SAT プランニングを探索空間の絞り込みのために利用する．多くのプランニング問題では，経験的に最小コストのプランは最短ステップのプランとなるため [Chen 08]，有効な探索空間の枝刈り手法であるといえる．

2·5 数値制約を含むプランニング問題

資源を消費するアクションなどの表現のために，アクションの前提条件や効果に数値変数と制約を含めたプランニング問題を考える．例えば場所 p から q に移動するときに燃料を消費するアクションを次のように記述する．

$$\begin{aligned}\mathrm{move}(p,q) = \langle &\mathrm{at}(p) \wedge \mathrm{fuel} \geq 2,\\ &\neg\mathrm{at}(p) \wedge \mathrm{at}(q) \wedge \mathrm{fuel} := \mathrm{fuel} - 2\rangle\end{aligned}$$

このような数値制約を含めたプランニングに対し，これまで SMT を利用した手法が提案されているが [Wolfman 99, Shin 05]，最近では前処理として数値変数のドメインを絞り込んだ後，SAT へ符号化する優れた手法も報告されている [Hoffmann 07].

Hoffmann らの手法では，整数変数 x とその値 c に対して，$x = c$ を意味する命題変数 $p(x = c)$ を用意する．すなわち**直接符号化**に等しい*6．ただし変数の取りうる値の集合（ドメイン）は明示されないため，それを絞り込む必要がある．そこで 2·2 節で示した命題またはアクションの到達可能性のチェックと同様にして（2·2 節のステップ (ii)），数値変数を扱えるように拡張したプランニンググラフを用いて，変数のドメインをステップ毎に荒く絞り込む．これによりプランの探索空間を絞り込み，SAT 符号化後の問題のサイズを抑えている．

それでも SMT に符号化した場合と比較して，SAT 問題のサイズはかなり大きくなる．しかしメモリに収まる範囲内では SMT を上回る性能を持つことを示している．さらに制約が強いプランニング問題においては，最短ステップのプランを求めているにもかかわらず，ヒューリスティックスベースの準最適プランナーよりも良い性能を示す傾向にあると述べている．

*6 直接符号化については本特集 [田村 10] を参照されたい．

3. SAT スケジューリング

プランニングとスケジューリングは密接に関係した分野である．両者ともに，解決すべき問題は，達成すべき目標条件とアクション，アクション実行のために必要となる資源，実行されるアクション系列に対する評価関数が与えられたときに，どのアクションをいつ，どのように実施するのが最適な計画なのかを決定する問題である [宮下 01]．近年では国際プランニング競技会においても，資源や時間，コストに関する制約を考慮した部門が設けられ，両者の関係は非常に近づきつつある．

ここでは，ショップ・スケジューリング問題に対する SAT 符号化として Crawford と Baker により提案された手法を紹介する [Crawford 94]．この符号化法は，その後，田村らによって制約充足問題および制約最適化問題に適用可能な順序符号化法へと拡張され，SAT 上で数値制約を効率的に扱う有望な方法として大きな成果を収めている [Tamura 06, Tamura 09].

ショップ・スケジューリング問題 (shop scheduling problem; SSP) は，ジョブの集合とマシンの集合からなる．各ジョブは複数のオペレーションから構成される．オペレーション全体の集合を $O_1,\ldots,O_n$ とする．各オペレーション O_i $(1 \leq i \leq n)$ には，それを処理するためのマシンと正整数の処理時間 p_i が割り当てられている．各マシンは一度に 1 つのオペレーションしか処理できない．また SSP の種類（フローショップ，ジョブショップ，オープンショップ・スケジューリング問題（それぞれ FSP, JSP, OSP と表す））に応じて，いくつかのオペレーション間に処理順序を規定する以下の制約が存在する．

- もし O_i を O_j より先に処理する必要があるならば，**先行制約** $O_i \Longrightarrow O_j$ を課す．これは O_i が O_j よりも前に完了すべきことを意味する．
- もし O_i と O_j が同時に処理できないならば，**資源制約** $O_i \Longleftrightarrow O_j$ を課す．これは O_i と O_j のどちらかが先に完了すべきことを意味する．

FSP では，各ジョブにおいて使用するマシンの順序は同じだが，JSP では異なる．一方 OSP では使用するマシンの順序に関する指定はない．例えば，OSP のベンチマーク問題 `gp03-01` は，3 ジョブ，3 マシン及び以下の処理時間 p_i を持つ 9 つのオペレーションからなる問題である [Guéret 99]．行がジョブを表し，同じ列中のオペレーションは同じマシンを使用する．

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

この問題では以下のような資源制約が存在する（OSP には先行制約が存在しない）.

$$
\begin{array}{lll}
O_1 \Longleftrightarrow O_2 & O_1 \Longleftrightarrow O_3 & O_2 \Longleftrightarrow O_3 \\
O_4 \Longleftrightarrow O_5 & O_4 \Longleftrightarrow O_6 & O_5 \Longleftrightarrow O_6 \\
O_7 \Longleftrightarrow O_8 & O_7 \Longleftrightarrow O_9 & O_7 \Longleftrightarrow O_9 \\
O_1 \Longleftrightarrow O_4 & O_1 \Longleftrightarrow O_7 & O_4 \Longleftrightarrow O_7 \\
O_2 \Longleftrightarrow O_5 & O_2 \Longleftrightarrow O_8 & O_5 \Longleftrightarrow O_8 \\
O_3 \Longleftrightarrow O_6 & O_3 \Longleftrightarrow O_9 & O_6 \Longleftrightarrow O_9
\end{array}
$$

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下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.

SAT プランニングと同様に，まず総所要時間をたかだか m と仮定し，その仮定の下で SSP を SAT 問題 P_m に変換する．もし P_m が充足可能ならば，総所要時間がたかだか m のスケジュールが存在する．充足不能の場合は，総所要時間が m 未満のスケジュールは存在しない．従って，P_k が充足可能で P_{k-1} が充足不能を満たす正整数 k を見つければ，最小の総所要時間であると結論できる．この k を境に，$1 \leq i < k$ であれば P_i は充足不能であり，$k \leq i$ であれば P_i は充足可能となる.

ショップ・スケジューリング問題を SAT 問題に変換するために，3 種類の命題変数を導入する.

- $pr_{i,j}$: オペレーション O_i がオペレーション O_j に先行する (precede) ことを意味する.
- $sa_{i.t}$: オペレーション O_i が時刻 t 以降に開始する (start at or later) ことを意味する.
- $eb_{i.t}$: オペレーション O_i が時刻 t までに終了する (end by) ことを意味する.

SAT 問題 P_m を以下の変換規則に従って生成する.

(a) 任意の 2 つのオペレーション O_i, O_j に対し，もし $O_i \Longrightarrow O_j$ ならば以下の単位節を追加する.

$$pr_{i.j}$$

もし $O_i \Longleftrightarrow O_j$ ならば，以下の節を追加する.

$$pr_{i.j} \vee pr_{j,i}$$

(b) オペレーション O_i は時刻 0 以降に開始し，時刻 m までに終了する.

$$sa_{i,0} \wedge eb_{i,m} \quad (1 \leq i \leq n)$$

(c) オペレーション O_i が時刻 t 以降に開始するなら，時刻 $t-1$ 以降でも開始している（$sa_{i,t}$ の定義に相当する）.

$$\neg sa_{i,t} \vee sa_{i,t-1} \quad (1 \leq i \leq n,\ 0 < t \leq m)$$

(d) オペレーション O_i が時刻 t までに終了するなら，時刻 $t+1$ までにも終了している（$eb_{i,t}$ の定義に相当する）.

$$\neg eb_{i,t} \vee eb_{i,t+1} \quad (1 \leq i \leq n,\ 0 \leq t < m)$$

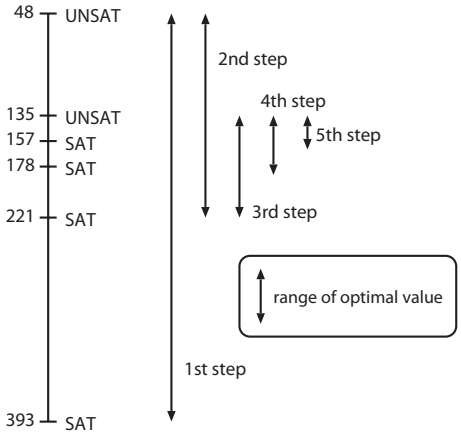


図 2 最適値の二分探索

(e) オペレーション O_i が時刻 t 以降に開始するなら，それは時刻 $t+p_i-1$ までには終了しない（O_i の処理時間を保証する）.

$$\neg sa_{i,t} \vee \neg eb_{i,t+p_i-1} \quad (1 \leq i \leq n,\ 0 \leq t \leq m-p_i)$$

(f) オペレーション O_i, O_j について，O_i が時刻 t 以降に開始し，かつ O_i が O_j に先行するなら，O_j は O_i の終了以降に開始される（$pr_{i,j}$ の定義に相当する）．各 $pr_{i,j}$ に対し，以下の節を追加する.

$$\neg sa_{i,t} \vee \neg pr_{i,j} \vee sa_{j,t+p_i} \quad (0 \leq t \leq m-p_i)$$

FSP, JSP, OSP における違いは規則 (a) にある．FSP, JSP では，各ジョブのオペレーションは指定された順序で実行されるため，ジョブ内の連続する 2 つのオペレーション間に先行制約がある．OSP ではジョブ内のオペレーション順序を入れ換えて良いため先行制約はないが，オペレーション同士が重なることを避けるため，ジョブ内の任意の 2 つのオペレーション間に資源制約がある．FSP, JSP, OSP ともに同じマシンを使用するオペレーション間には資源制約がある．規則 (b) では，各オペレーションの開始時刻と終了時刻をぞれぞれ 0 と m としているが，FSP や JSP では先行制約から時刻を絞り込むことが可能である（例えば $O_1 \Longrightarrow O_2$ ならば，O_2 の開始時刻は早くても p_1 以降となる）.

この変換規則より得られる SAT 問題 P_m が充足可能であれば総所要時間が m 以下のスケジュールが存在する．総所要時間の最小化のために二分探索を用いることができる [Inoue 06]．まず総所要時間の下界 l と上界 u を何らかの手法で求めておく．例えば下界が 49, 上界が 393 と分かっているならば，その中央値は 221 となるので P_{221} を生成する（図 2）．もし充足可能ならば上界を 222 に更新し，充足不能ならば下界を 223 に更新する．この操作を再帰的に繰り返せば最適値を発見できる.

Crawford と Baker による符号化法の特徴は，数値変数の取りうる範囲を $sa_{i.t}$ と $eb_{i.t}$ によって表現しているこ

とにある（それぞれ時刻 t 以降と時刻 t までの範囲を表している）．SSP のベンチマーク問題における総所要時間はしばしば 1000 を超える値を取る．数値変数のドメイン・サイズを d とすると，もし直接符号化法を利用した場合は $O(d^2)$ 個の節を必要とするが，この符号化法では $O(d)$ 個の節で済むため [田村 10]，非常にコンパクトに表現することが可能である．この符号化に基づき，田村らと越村らは，それぞれそれまで未知であったいくつかの OSP と JSP の最適値決定に成功している [田村 07, 多賀 07, 越村 09] [*7]．

Crawford と Baker の符号化はさらに簡単化することが可能である．まず規則 (b) の $\mathit{eb}_{i,m}$ は $\neg \mathit{sa}_{i,m-p_i+1}$ に置き換えることができる．これは $\neg \mathit{sa}_{i,m-p_i+1}$ が，オペレーション O_i が時刻 m までに終了することを意味するからである．このことは命題変数 $\mathit{eb}_{i,t}$ が不要であり，削除できることを意味する．これにより $\mathit{eb}_{i.t}$ に関する規則 (d)(e) を取り除くことができる．規則 (e) は O_i の処理時間を保証する規則だが，O_i に後続するオペレーション O_j が存在するならば，先行制約または資源制約から生成される $\mathit{pr}_{i.j}$ と規則 (f) によって O_i の処理時間を保証できるため，規則 (e) は不要になる．命題変数 $\mathit{eb}_{i,t}$ の削除によって，変数の数を約半減できる（節数は規則 (f) がその大半を占めるため微減するだけである）．簡単化した符号化を用いて，OR-library [Beasley 90] に含まれる JSP 30 問について評価したところ[*8]，1.1 倍程度の速度向上が得られることを確認した．

4. SAT ソルバーのインクリメンタル探索

SAT プランニングや SAT スケジューリングでは，最短のプランやスケジュールを求めるために複数の SAT 問題を解く必要がある．SAT は判定問題であるため，SAT を利用した手法において何らかの最適化を行う場合には，複数の SAT 問題を解くことが本質的に必要となる．各 SAT 問題は通常数多くの共通する節を含む．よってそれらを個別に生成し解くことは，生成の手間の重複や，各問題に共通する探索空間を何度も解く必要が生じ，効率が悪い．これを改善する手法として，SAT ソルバーの**インクリメンタル探索** (incremental search) 手法が提案されている [Eén 03]．

最新の高速 SAT ソルバーにおける特徴の 1 つは，探索過程で矛盾が発生した際にその原因を解析し，以後同様の矛盾に陥ることを防ぐための**学習節** (learned clause) [Marques-Silva 99] を獲得することにある．SAT 問題の系列を $\langle P_1, P_2, \ldots, P_n \rangle$ とする．もし $P_i \subseteq P_j$ $(i < j)$ ならば，P_i から得られる学習節は P_i の論理的帰結であるので，明らかに P_j を解くためにも利用でき，探索しても無駄であることが分かっている空間を再度探索し直すことを避けることが可能になる．

ただし，SAT プランニングや SAT スケジューリングにおいて節が単調に増加することはない．SAT プランニング（スケジューリング）におけるステップ数（総所要時間）が k のときの SAT 問題を P_k とし，2 つの SAT 問題 P_s, P_t $(s < t)$ を考える．このときプランニングでは，目標条件 G がステップ数 s で真となることを表す単位節 G^s が P_t に含まれない．スケジューリングでは，各オペレーション O_i が時刻 s までに終了することを表す単位節 $\mathit{eb}_{i.s}$ が P_t に含まれない．ただし P_s に含まれるその他の節はすべて P_t に含まれる．すなわち，単位節を除けば $P_s \subseteq P_t$ が成立する．

このような関係にある 2 つの SAT 問題 P_s, P_t に対しては，学習節の再利用が可能であることが示されている [Nabeshima 06]．すなわち，任意の 2 つの SAT 問題 P, Q に対し，単位節を除き $P \subseteq Q$ ならば，P から生成された学習節を Q を解くために利用することが可能である[*9]．よって，P を解く過程で生成された学習節の集合を L_P とすると，Q を解くときに L_P を追加して解くことにより，探索しても無駄であることが分かっている空間を再度探索し直すことを避けることが可能になる．

インクリメンタル探索の具体的な手順は次のようになる．まず P を SAT ソルバーに読み込ませて解く．通常 SAT ソルバーは充足可能性判定後も，節集合 P と，P を解く過程で生成した学習節集合 L_P を保持している．そこで次に差分の節集合 $Q \setminus P$ のみを生成し，SAT ソルバーに登録する．また逆に P に含まれる単位節のうち，Q に含まれないものは SAT ソルバーから取り除く．その後，SAT ソルバーに充足可能性を判定させれば，L_P を利用しながら Q を解くことができる．

ところで SAT スケジューリングのように最適値発見のために二分探索を用いる場合，SAT 問題のサイズが減少することがある．例えば 図 2 では，P_{221} を解いた後に P_{135} を解くため，単位節を除いても $P_{221} \subseteq P_{135}$ とならない．このような場合には，まず上界の 1 つ手前の時刻 392 における SAT 問題 P_{392} を作成しておき，P_{392} 中の各オペレーション O_i の終了時刻を表す単位節 $\mathit{eb}_{i,392}$ を必要に応じて他の時刻のものと入れ換えればよい．

学習節はしばしば多量に生成されるため，再利用にあたっては，例えば短い学習節のみを利用するなどの選別を行うと効果的である．SAT プランニングと SAT スケジューリングでは，長さ 10 以下の学習節を再利用することで，再利用なしと比較して，それぞれ約 2 倍，約 1.2 倍速度が向上することが示されている [Nabeshima 06]．

∗7 文献 [越村 09] では abz9 の最適値決定のみ報告されているが，発表時には yn1 についても最適値決定に成功したとの報告があった．

∗8 対象とした問題および実験環境は文献 [Nabeshima 06] にある実験と同じである．

∗9 この性質は，学習節の導出時に単位節を用いた融合が行われないことを利用している．最新の SAT ソルバーでは，単位節と融合することで学習節をより簡単にすることがあるが，その場合，指定された単位節集合に対し，これを抑制する機能も組み込まれている．

5. お わ り に

本稿では SAT 利用したプランニングについて，最短ステップ，最小コストのプランを求める基本的な手法と準最適なプランを求めるための探索戦略を紹介した．SAT スケジューリングにおいて紹介した Crawford と Baker による符号化は，整数変数に対する範囲制約の効率的な表現方法を与えている．これを，資源や時間などの制約及びコストを考慮したプラン生成のために適用することは，今後の興味深い課題の 1 つである．

SAT プランニングの成功の要因の 1 つは，性能向上の著しい SAT ソルバーを単純に利用するだけでなく，その前に探索空間をできるだけ絞り込んでおく前処理にあるといえる．SAT スケジューリングの高速化や，数値制約・コストなどを含む高度なプランニングの効率化のためには，この前処理が重要な要素になると考えられる．

◇ 参 考 文 献 ◇

[Beasley 90] Beasley, J. E.: OR-Library: Distributing Test Problems by Electronic Mail, *Operational Research Society*, Vol. 41, pp. 1069–1072 (1990)

[Blum 97] Blum, A. L. and Furst, M. L.: Fast Planning through Planning Graph Analysis, *Artificial Intelligence*, Vol. 90, No. 1–2, pp. 279–298 (1997)

[Bonet 01] Bonet, B. and Geffner, H.: Planning as Heuristic Search, *Artificial Intelligence*, Vol. 129, pp. 5–33 (2001)

[Chen 08] Chen, Y. and Lv, Q.: Plan-A: A Cost Optimal Planner Based on SAT-Constrained Optimization, in *Proceedings of the sixth International Planning Competition (IPC-6)* (2008)

[Chen 09] Chen, Y., Huang, R., Xing, Z., and Zhang, W.: Long-distance Mutual Exclusion for Planning, *Artificial Intelligence*, Vol. 173, No. 2, pp. 365–391 (2009)

[Crawford 94] Crawford, J. M. and Baker, A. B.: Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems, in *Proceedings of AAAI-94*, pp. 1092–1097 (1994)

[Dimopoulos 97] Dimopoulos, Y., Nebel, B., and Koehler, J.: Encoding Planning Problems in Nonmonotonic Logic Programs, in *Proceedings of the 4th European Conference on Planning (ECP-97)*, pp. 169–181 (1997)

[Eén 03] Eén, N. and Sörensson, N.: Temporal Induction by Incremental SAT Solving, *Electronic Notes in Theoretical Computer Science*, Vol. 89, No. 4, pp. 543–560 (2003)

[Ernst 97] Ernst, M. D., Millstein, T. D., and Weld, D. S.: Automatic SAT-Compilation of Planning Problems, in *Proceedings of IJCAI-97*, pp. 1169–1177 (1997)

[Guéret 99] Guéret, C. and Prins, C.: A New Lower Bound for the Open-shop Problem, *Annals of Operations Research*, Vol. 92, pp. 165–183 (1999)

[Hoffmann 07] Hoffmann, J., Gomes, C. P., Selman, B., and Kautz, H. A.: SAT Encodings of State-Space Reachability Problems in Numeric Domains, in *Proceedings of IJCAI-07*, pp. 1918–1923 (2007)

[Inoue 06] Inoue, K., Soh, T., Ueda, S., Sasaura, Y., Banbara, M., and Tamura, N.: A Competitive and Cooperative Approach to Propositional Satisfiability, *Discrete Applied Mathematics*, Vol. 154, No. 16, pp. 2291–2306 (2006)

[Kautz 92] Kautz, H. and Selman, B.: Planning as Satisfiability, in *Proceedings of the Tenth European Conference on Artificial Intelligence*, pp. 359–379 (1992)

[Kautz 96] Kautz, H. and Selman, B.: Pushing the Envelope: Planning, Propositional Logic and Stochastic Search, in *Proceedings of AAAI-96*, pp. 1194–1201 (1996)

[Kautz 98] Kautz, H. and Selman, B.: BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving, in *AIPS98 Workshop on Planning as Combinatorial Search*, pp. 58–60 (1998)

[越村 09] 越村 三幸, 鍋島 英知, 藤田 博, 長谷川 隆三：SAT 変換による未解決ジョブショップスケジューリング問題への挑戦, スケジューリング・シンポジウム 2009 講演論文集 (2009)

[Marques-Silva 99] Marques-Silva, J. P. and Sakallah, K. A.: GRASP – A Search Algorithm for Propositional Satisfiability, *IEEE Transactions on Computers*, Vol. 48, pp. 506–521 (1999)

[宮下 01] 宮下 和雄：プランニングとスケジューリング, 人工知能学会誌, Vol. 16, No. 5, pp. 611–616 (2001)

[鍋島 01] 鍋島 英知, 井上 克巳：プランニンググラフと SAT プランニング, 人工知能学会誌, Vol. 16, No. 5, pp. 605–610 (2001)

[Nabeshima 06] Nabeshima, H., Soh, T., Inoue, K., and Iwanuma, K.: 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)

[鍋島 10] 鍋島 英知, 宋 剛秀：高速 SAT ソルバーの原理, 人工知能学会誌 (2010)

[Rintanen 06] Rintanen, J., Heljanko, K., and Niemelä, I.: Planning as Satisfiability: Parallel Plans and Algorithms for Plan Search, *Artificial Intelligence*, Vol. 170, No. 12-13, pp. 1031–1080 (2006)

[Robinson 08] Robinson, N., Gretton, C., and Pham, D.-N.: CO-PLAN: Combining SAT-Based Planning with Forward-Search, in *Proceedings of the sixth International Planning Competition (IPC-6)* (2008)

[Shin 05] Shin, J.-A. and Davis, E.: Processes and Continuous Change in a SAT-based planner, *Artificial Intelligence*, Vol. 166, No. 1-2, pp. 194–253 (2005)

[多賀 07] 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 講演論文集, pp. 109–114 (2007)

[Tamura 06] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M.: 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 (2006)

[田村 07] 田村 直之, 多賀 明子, 番原 睦則, 宋 剛秀, 鍋島 英知, 井上 克巳：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 講演論文集, pp. 97–102 (2007)

[Tamura 09] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M.: Compiling Finite Linear CSP into SAT, *Constraints*, Vol. 14, No. 2, pp. 254–272 (2009)

[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 (2010)

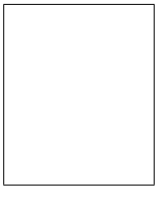
[Wolfman 99] Wolfman, S. A. and Weld, D. S.: The LPSAT Engine & Its Application to Resource Planning, in *Proceedings of IJCAI-99*, pp. 310–317 (1999)

[Xing 06] Xing, Z., Chen, Y., , and Zhang, W.: MaxPlan: Optimal Planning by Decomposed Satisfiability and Backward Reduction, in *Proceedings of the fifth International Planning Competition (IPC-5)*, pp. 53–56 (2006)

〔担当委員：××○○〕

19YY 年 MM 月 DD 日　受理

著 者 紹 介



鍋島　英知（正会員）

2001 年神戸大学大学院・自然科学研究科情報メディア科学専攻博士課程修了．同年山梨大学工学部コンピュータ・メディア工学科助手．現在，同大学大学院・医学工学総合研究部准教授．博士 (工学)．アクション理論，プランニング，定理自動証明プログラム，WEB 知的処理などの研究に従事．2004 年 FIT 優秀論文賞受賞．