卒業研究
プランニング問題のドメイン非依存なモデリング手法: 近年のソルバ高速化手法を考慮して

氏名：堀江 慧
指導教官：福永 Alex
学生証番号：08-132021
所属：教養学部 学際科学科 総合情報学コース

2015 年 1 月 25 日

目次

図目次

表目次

1 序論

1.1 研究の概要

本研究で扱うプランニング問題は，現実の意思決定をモデル化した問題であり，人工知能分野の主要な問題のうちの一つとされ，かつてから広く研究の対象として扱われているものの一つである。

直接的な応用としてはロボットの自律的な行動決定が想像されるが，行為者は必ずしも物理的な実態を持つものに限らない。例えばゲームのプレイヤーは現状に対して行為の意思決定を求められるが物理的な実態は必ずしも必要としない。このように物理的な実態を持たない行為者のことをアバターといい，ロボットとアバターとを合わせた抽象的な行為者をエージェントという。

人工知能の備える’知能’の性質にふさわしいものとして汎用性が挙げられる。与えられた問題に対して，ドメイン特化の知識を人間が与えれば，よりよいパフォーマンスを発揮することができることができるであろう。(Kautz&Selman, 98 [10]) 一方で，そのような人間の’知能’を介助を得ること無くその問題解決を達成するような人工知能は，より望ましいものであるといえる。そして，そのような知能の実現は当然より難しいものとなる。

本研究では汎用モデルとそのソルバーを利用し，プランニング問題を解くドメイン非依存な手法について取り扱う。また，そのようなモデルとして充足可能性問題, 整数計画問題,Constraint Problem(CP) を扱う。

ここで言うドメイン非依存性とは，あるモデルで表現できる問題であればどのような問題インスタンスに対しても同様の処理で解を得ることが出来る性質をいう。プランニング問題自身の柔軟性，実用性からこのドメイン非依存性は有用な性質である。ドメイン特化の知識を利用して解を得る手法の場合，プランニング問題のドメインごとにその処理を変更する必要がある。一方，ドメイン非依存なプランニング手法は，そのモデルで表現される問題であればドメインによらずに適応可能な手法であり，より一般性の高い手法となる。

ドメイン非依存なプランニング問題へのアプローチとして，充足可能性問題，整数計画問題，CP などの他のモデルにプランニング問題を変換し，各モデルのソルバによってその解を計算する手法が研究されている。上のモデルのうち，本研究では CP のモデルにおけるプランニング問題へのアプローチについて研究を行う。特に，CP モデルにおける global constraint に着目したプランニング問題の変換手法を扱う。

global constraint とは制約プログラミングの分野で近年特に研究の進んでいる特定の制約群である。

かつての研究では，二項の間の制約を中心に研究が進んできた。このような素朴な制約を global constraint に対して local constraint と呼ぶことがある。このアプローチは任意の多項制約は二項制約の組み合わせで表現出来るという事実によって正当化されてきた。それに関連し，二項制約における効果的な制約伝播の方法として arc consistency などの解の計算手法が開発されてきた。

一方で，実用を考えた場合には理論的な計算可能性だけでなくその実行時間も考慮する必要がある。

現在主流となっている多くの CP のソルバでは，制約伝播による枝刈りを利用したバックトラック探索によって解を計算する。global constraint を利用した CP の定式化では，それを利用しない素朴な定式化と比較してより効率的な制約伝播による探索空間の縮小が可能となり (regin, 04 [15])，結果として効率的な探索が可能になることがあることが知られている。二項制約に対して制約伝播を行い探索空間を削減する有名なアルゴリズムとして arc consistency を達成するものがあるが，global constraint には専用のより大きく探索空間の削減を実現する制約伝播方法が開発されているものもある。

ここで注意するのは，制約伝播の回数あたりにより多くの探索空間を削減できることが，必ずしも解の計算

時間を短くするとは限らないことである。これは，制約伝播自体の計算コストの大きさが実行する制約伝播アルゴリズムの複雑さに影響される事による。

global constraint をうまく取り込んだプランニング問題の定式化を行えば，素朴な制約のみから作成したモデルよりも効率的に解を得ることができる可能性があると考えられる。

また，表現力豊かな制約を利用することで，モデリング自体の困難さを緩和し，等価なモデルをよりシンプルに表現することができる。

例えば all-different 制約という global constraint を考えてみる。ここでは簡単のため変数を 0-1 変数とするが，多値変数の場合でも同様の議論が可能である。

制約 all-different$\{x_i \mid i = 0 \cdots n\}$ は，ある変数の集合 $\{x_i \mid i = 0 \cdots n\}$ の全ての要素が互いに異なる値を取ることを要求する単一の制約である。

これを素朴な論理式で表現すると，

$$\bigwedge_{i,j \in \{0 \cdots n\}, i \neq j} (\neg x_i \vee \neg x_j) \wedge (x_i \vee x_j) = \bigwedge_{i,j \in \{0 \cdots n\}, i \neq j} (\neg (x_i \wedge x_j) \wedge (x_i \vee x_j)$$

となり，$O(n^2)$ 個の節が必要である。

線形不等式で表現すると

$$x_i + x_j <= 1 \quad (\forall i, j \in \{0 \cdots n\} (i \neq j))$$

となり，$O(n^2)$ 個の不等式制約が必要である。

このように，all-different 制約を利用すれば，式の大きさが n に依存してしまう困難を回避しシンプルな記述が可能となっている事がわかる。

さらに，all-different 制約には専用の制約伝播のためのアルゴリズムがいくつか開発されている。(Hoeve, 2001 [19]) 例えば，all-different 制約は制約の変数集合と，全ての変数の定義域の和集合をそれぞれ集合とする二部グラフ上の最大マッチング問題によって行う事ができる。すなわち最大マッチングの個数が変数の個数より小さかった場合にはこれらの変数について完全な値の割り当てが達成される見込みはなく，即座に矛盾を検出することが出来る。

この計算は，二部グラフの頂点数を m，辺の数を n とした場合，$O(mn)$ で行えることが知られており，これはアーク無矛盾性を達成する汎用アルゴリズムの AC-4 [13] の $O(m^2n^2)$ と比べて高速である。

ここで注意しておくのは，all-different の専用制約伝播アルゴリズムを利用した場合でも，等価な二項制約の集合に対して arc consistency を達成する制約伝播アルゴリズムを計算する場合と比べてステップ数は小さくなるものの，実行時間は必ずしも小さくならない。(Regin,2004 [15])

まとめると，本研究ではプランニング問題を他のモデルに変換し，その解を汎用ソルバによって計算する手法についての研究を行う。

また，制約プログラミングのモデルで古典的なプランニング問題を定式化する際に，global constraint を利用した有効な定式化な手法についての提案を行う。

提案手法の有効性の検証として，充足可能性モデルの SATPLAN2006 [9]，整数計画モデルの optiplan [18] といったプランナに対して，global constraint を利用した本研究の手法による定式化を追加したプランナとの比較を行う。

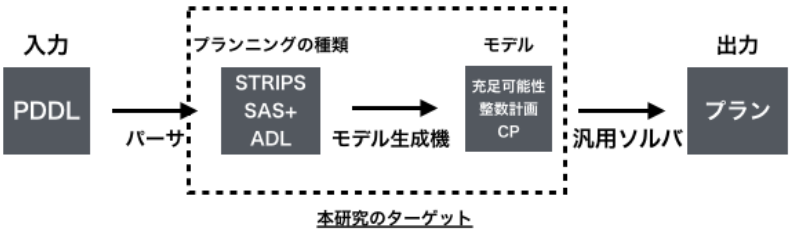

図 1　モデルを利用したプランニング問題

1.2　背景

正確な定義は後に述べるが，プランニング問題とは例えば以下のような問いに応える問題である。エアコンの電源を入れるにはどうすればよいか，部屋を綺麗にするにはどうすればよいか，洞窟の中から脱出するにはどうしたらよいか...。また，スライディングタイルやハノイの塔といったパズルの類もこの枠組で表現することができる。

このように，この問題は現実の行動計画，意思決定の直接的な定式化である一方で，また，多くの現実的な問いをこの枠組みで表現することができる。

迷路の問題を元に考えてみる。迷路内で行える行為として，右に向き換えること，左に向きを換えること，向いている方向に直進すること，の３つの行為が可能であるものとする。この時に，迷路内のある地点にあるゴールにたどり着くようなプランを考える。ゴールに辿り着けるプランが存在するのであれば，実行可能な行為を全て列挙してしまえばゴールに辿り着くようなプランが得られるであろう。このように，初期状態から行為を実行していき，各行為の効果を書く時点での状態に反映させていくことを繰り返せば，探索によってプランを得ることができる。一方で，行為の選択の度にその際の選択肢の数だけ可能な状態が分岐することから，その探索空間の大きさはプランの深さに対して指数的に増加していくこととなる。

このように，この問題は一般には解の長さに対して指数的に探索空間の大きさが増えていき，最悪の場合でも有限多項式時間で解を求める方法はまだ見つかっていない。一方で，充足可能性問題, 整数計画問題, 制約プログラミングなどの問題も同様に，汎用な枠組みでありかつ計算時間上の困難に直面する問題であることが示されていながらも，近年では比較的大規模な実用的問題への応用が可能になってきている。このことを背景に，上に挙げた汎用モデルでプランニング問題を表現し，各モデルのソルバでプランを得るアプローチが研究されている。(図 1.2 参照)

先行研究としては，充足可能性問題を利用したものとして SATPLAN, 整数計画のものとして optiplan, 制約プログラミングのものとして GP-CSP などが開発されており，現在も引き続きより高速な手法が考案されて続けている。

モデルによるプランニング問題のアプローチの近年の成果として，Graphplan で初めて実装されたプランニンググラフを利用する方法がある。

この手法では，与えられたプランニング問題からまずプランニンググラフを生成する。

プランニンググラフは完全性を損なわない範囲で比較的大きく解空間を縮めることのできる問題の表現であり，このプランニンググラフ上の縮小された探索空間から解を発見する。また，構成されたプランニンググラフからは解の長さの下限を得ることもできる。プランニンググラフの構成と探索空間の縮小自体は，実際の探索と比べて小さいオーダーの実行時間で行うことが出来るのがポイントである。

この手法は充足可能性モデルで実装された後，その有効性が認められ，整数計画モデル，CP モデルにも持ち込まれている。

1.3 研究の目的

本研究はプランニング問題についてモデル表現とソルバを利用した方法によるアプローチについての研究である。プランニング問題は PSPACE 完全であることが知られている。(Bylander, 91 [4]) この問題の解を得るため，プランニング問題を汎用モデルに変換するアプローチが研究されている。(図 1.2 参照) このアプローチではソルバの性能向上がプランニングシステムの性能に直結し，各モデルのソルバは日々進展している。特に整数計画ソルバは Operation Research の分野で提案された多くの手法によって，近年目覚ましい高速化を達成している。([2], [12])

しかしながら，近年の研究はモデルの中でも充足可能性モデルに関するものが多くを占めている。

この状況に対し，本研究では CP モデル，整数計画モデルでの近年のソルバの性能向上がそれぞれのプランニングシステムの性能向上にどのくらい影響を及ぼしているのかを再調査する必要があると考えた。

また，特に CP モデルにおける近年の性能向上に大きな影響を与えている global constraint について，それを効果的に織り込んだプランニングの定式化を示すことを目指した。

1.4 本論文の構成

2 章では，まず，プランニング問題とはどのようなものかを正確に表現するための用語の定義を行う。続いてプランニング問題を形式的に表現し，これを書き表すための形式的な言語 PDDL の説明を例を交えながら行う。その後，各モデルでプランニング問題の表現がどのようなものとなるかを考察し，最も素朴な充足可能性モデルでの表現を具体例を交えながら追う。

3 章では，充足可能性問題，整数計画問題，CP がどのようなものであるかをそれぞれ解説する。

4 章では，プランニング問題を先に述べた 3 つのモデルでそれぞれ表現することができることを示す。また，それぞれのモデルの表現力の包含関係についても説明を行う。続いて，プランニング問題の素朴な表現での無駄を省略するためのプランニンググラフ (Blum et al., 97 [3]) を利用した探索空間の削減手法について説明する。その後，本研究で扱った optiplan で用いられているプランニング問題の整数計画による表現のためのモデリングである state change model について説明する。

5 章では，SATPLAN2006 プランナと optiplan プランナの IPC ベンチマーク問題集による比較実験について述べる。まずはそれぞれのプランナの特徴について説明した後，実験結果を報告する。

6 章では，CP モデルでのプランナの実装について述べる。本研究の実装で利用した minizinc について説明した後，本研究で扱ったプランニング問題に global constraint を導入する場合の定式化について述べる。

7 章では，6 章で述べた global constraint を利用して実装した CP モデルのプランナと optiplan のモデルを元にした整数計画モデルのプランナとの比較実験について述べる。

2 プランニング問題

プランニング問題の設定としては，マルチエージェント環境やリアルタイム性を考慮したものなど多くのものが挙げられているが，本研究では最も広く扱われている古典的プランニング問題を扱う。

2.1 古典的プランニング問題の定義

以下，プランニング問題の古典的な表現である古典的プランニング問題の定式化についての説明を行う。古典的プランニング問題とは’十分に観察可能で，決定論的で，有限で，静的（エージェント自身が行為を実行した場合にのみ変化する）で，かつ（時間，行為，対象，効果が）離散的な環境のみを扱う’ [17] 問題である。

十分に観察可能であるとは，プランニング問題の中で扱う命題に関してはその真偽が定まることである。決定的とはある状態に置いて行為を行うと至る状態が一意に定まることである。有限であるとは，命題，行為などのプランニングに関連するオブジェクトか高々有限個しか存在しないことを述べる。静的であるとはエージェントの行為の効果のみが環境を構成する命題の値を変化させることを述べる。離散的であるとは，扱うオブジェクトが非連続的であることを述べる。

また，ここで述べるもの以外にも，プランニングの表現としては多くのものが提案されていることを述べておく。

まず，世界のうちの個別の事象の状態を表現するような種々の命題というものが存在するとする。命題 p に対して，その否定を $\neg p$ と表現する。命題とその否定を合わせてリテラルと呼ぶ。また，前者を正のリテラル，後者を負のリテラルと呼ぶこともある。

p と $\neg p$ はある命題の成否と対応するリテラルで，その真偽値は同時には真にならない。

また，$p \vee \neg p$ であるものとする。(閉世界仮説)

ここで，現実の世界のうちどの事象を考慮の対象とすることで問題を十分に表現しうるか，という問題はフレーム問題として知られており，人工知能分野の古くから知られている難問のうちの一つで有ることを注意しておく。しかしながら，今，問題を定義するための命題の集合

を定めたとすると，この命題集合の上でプランニング問題を定義することができる。

プランニング問題は以下の４つ組で定義される。

$$\{P, I, G, A\}$$

- $P = \{p_i \mid i = 0, \cdots, n\}$ は命題集合
- $I(\subseteq P)$ は初期状態
- $G(\subseteq P)$ は目的状態
- $A(\subseteq P \times P \times P)$ は行為集合

ここで，世界の状態 S とは，

$$S = \{(p_i, b) \mid \forall p_i \in P, b \in \{\text{true}, \text{false}\}\}$$

であり，命題のすべての要素への値の割り当てを表現しているものとする。

S の要素のうち (p, b) を (p, b') に置き換える操作を $S[p] \setminus b'$ と表すこととする。また，S の要素 (p, b) が存在するとき，$S[p] = b$ と定める。

行為は以下のように定義される。

$$a = (Pre, Add, Del)(a \in A, Pre \in 2^P, Add \in 2^P, Del \in 2^P)$$

Pre は前提条件であり，この行為が実行される際にその真偽値が真となっていることを必要とする命題の集合である。一方，Add と Del は行為の効果であり，行為の実行後に真となっているべき命題，偽となっているべき命題の集合である。Add を正の効果，Del を負の効果という。

行為 $a = (Pre, Add, Del)$ に対して，その前提条件を $Pre(a)$，正の効果を $Add(a)$，負の効果を $Del(a)$ と書くことにする。

S に対して実行可能な行為 α とは，α の前提条件集合 Pre に対して $Pre \in S$ が，成立するようなもののことである。

この行為を実行すると，世界の新たな状態 S' が得られる。

$$S' = (S \cup Add) \setminus Del$$

すなわち S' においては，Add に入っている命題はその真偽値が真であり，Del に入っている命題はその真偽値が偽であり，いずれにも入っていない命題は S の真偽値を保存する。

ここで，Add と Del には同じリテラルは含まれないことを仮定していることに注意すると，S の要素の書き換えはどの順序で行っても結果が変わらないことに注意する。また，この仮定は Add と Del の意味を考えると適当なものである。

これを状態 S に行為 α を適応するという意味で $S' = \alpha(S)$ と書くこととする。

また，初期状態 I は全ての命題への真偽値の割り当てを表現する。ここでは閉世界仮説を採用していることから，初期状態で正の値を割り当てられる命題の集合を S とすれば偽が割り当てられる命題は明らかである。

$$I \subseteq S$$

すなわち，開始時点の状態 S_0 では I の要素となっている命題には真が，入っていない要素には偽が割り当てられていることを要求する。

$$\underset{(p,b)\in S_0}{\forall} b = \begin{cases} \text{true} & p \in I \text{ のとき} \\ \text{false} & \text{それ以外のとき} \end{cases}$$

続いて，目的状態 G はプランニング問題のゴール，最終的に要求される状態である。

$$G \subseteq S$$

ここで，初期状態では集合 I に入っていない要素には偽が割り当てられることを要求していたが，目的状態 S_T では集合 G に入っていない状態に関しては何も要求をしないことに注意する。すなわち，初期状態は状態集合の値の割り当てを一意に特定するが，目的状態では興味のある命題にのみ値の割り当てを要求する。

$$\underset{(p,b)\in S_T \wedge p\in G}{\forall} b = true$$

このことから，初期状態は I から一意に定まるが，目的状態に関しては複数の状態がそれを満たしうることがある。

プランニング問題の解，すなわちプランとは，初期状態 S_0 に順次適応することで目的状態 S_T へと至るような行為の列である。ここで，目的状態に至ったかどうかの判定はその時点での状態集合が目的状態を全て含んでいるかどうかで行われることに注意する。

すなわち，

$$\begin{cases} S_T = \alpha_n(\alpha_{n-1}(\cdots\alpha_1(\alpha_0(S_0)))) \\ S_T \supseteq G \end{cases}$$

の時に

$$(\alpha_0, \alpha_1, \cdots, \alpha_{n-1}, \alpha_n)$$

がプランである。

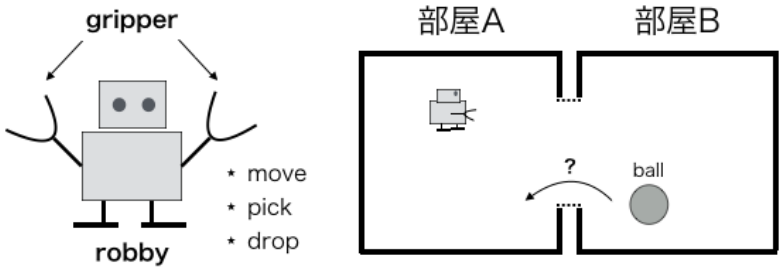

図 2　ドメイン: gripper

2.2　PDDL

プランニング問題を記述するための標準的な形式として Planning Domain Definition Language(PDDL) という形式が広く利用されている。

PDDL を構成するファイルには問題ドメインについての共通の情報を扱うドメインファイルと個別の問題ファイルがある。

ドメインファイルにはドメイン名，述語の定義，行為の定義などが含まれる。行為の定義では各行為の前提条件と効果が述べられる。

問題ファイルにはオブジェクトの一覧と，初期状態，終了状態が含まれている。

また，プランニング問題の解も PDDL 形式で出力されることがある。これは出力されたプランを並べたものである。

2.2.1　プランニング問題の例

プランニング問題の例として以下の素朴な問題が考えられる。

２つの部屋 A,B が存在する。現在，持ち手を２つ備えたロボットが部屋 A にいるものとする。ロボットに可能な行為は以下の３つである。

- 現在の部屋から他の部屋に移動すること
- 空いている持ち手でロボットのいる部屋にあるボールを拾うこと
- 持ち手にあるボールをロボットのいる部屋にボールを落とすこと

ここで，ボールが部屋 A にあり，ロボットが部屋 B にいるものとするとき，ボールを部屋 B に移動させるようなプランを求める。

2.2.2　PDDL の例

プランニングの例で述べた gripper ドメインの問題を表現する PDDL の例を挙げる。以下の PDDL ファイルは fastdownward [7] 付属の benchmarks の gripper ドメインの domain.pddl, prob01.pddl を元に著者がボールの個数を減らし，コメントを付与したものである。

まず，ドメインファイルは以下のようになる。(なお; 以降はコメントである。)

```
;gripper-strips のドメインファイルを定義
(define (domain gripper-strips)
    ; 述語の定義
    (:predicates
        (room ?r)          ;r は部屋であるか
        (ball ?b)          ;b はボールであるか
        (gripper ?g)       ;g は持ち手であるか
        (at-robby ?r)      ;r にロボットがいるか
        (at ?b ?r)         ;b が r にあるか
        (free ?g)          ;g が空いているか
        (carry ?o ?g))     ;g に o を持っているか

   ; 行為 move の定義
   (:action move           ;from から to に移動する
       :parameters  (?from ?to)
       :precondition (and  (room ?from) (room ?to) (at-robby ?from))
                           ; 前提条件は
                           ; from,to が部屋であり,from にロボットがいること
       :effect (and  (at-robby ?to)
             (not (at-robby ?from))))
                           ; 効果は
                           ; ロボットが to にいるようになること（正の効果）
                           ; ロボットが from にいないようになること（負の効果）

   ; 行為 pick の定義
   (:action pick           ;obj を room で gripper が拾う
       :parameters (?obj ?room ?gripper)
       :precondition  (and  (ball ?obj) (room ?room) (gripper ?gripper)
                            (at ?obj ?room) (at-robby ?room) (free ?gripper))
                           ; 前提条件は
                           ; obj がボールであり,room が部屋であり,
                           ; gripper が持ち手であり,obj が room にあり,
                           ; ロボットが room にいて,gripper が空いていること
       :effect (and (carry ?obj ?gripper)
                    (not (at ?obj ?room))
                    (not (free ?gripper))))
                           ; 効果は
                           ; gripper に obj を持っているようになること（正の効果）
                           ; obj が room にあるようになること（負の効果）
                           ; gripper が空いているようになること（負の効果）
```

```
    ; 行為 drop の定義
    (:action drop           ;obj を room で gripper から落とす
        :parameters  (?obj  ?room ?gripper)
        :precondition  (and  (ball ?obj) (room ?room) (gripper ?gripper)
                             (carry ?obj ?gripper) (at-robby ?room))
                           ; 前提条件は
                           ; obj がボールであること,room が部屋であること
                           ; gripper が持ち手であること,gripper に obj を持っていること
                           ; room にロボットがいること
        :effect (and (at ?obj ?room)
                     (free ?gripper)
                     (not (carry ?obj ?gripper))))
                           ; 効果は
                           ; obj が room にあるようになること（正の効果）
                           ; gripper が空いているようになること（正の効果）
                           ; gripper に obj を持っていないようになること（負の効果）
)
```

問題ファイルは以下のようになる。

```
;strips-gripper の問題ファイルを定義
(define (problem strips-gripper-x-1)
    (:domain gripper-strips)
    (:objects rooma roomb ball left right)
                           ; オブジェクトとして rooma,roomb,ball,left,right が存在する
    (:init (room rooma)
        (room roomb)
        (ball ball)
        (at-robby roomb)
        (free left)
        (free right)
        (at ball rooma)
        (gripper left)
        (gripper right))
                           ; 初期状態は
                           ; rooma,roomb は部屋である,ball はボールである
                           ; ロボットは rooma にいる
                           ; left,right は空いている
                           ; ball は rooma にある
                           ; left,right は持ち手である
```

```
    (:goal (and (at ball roomb)))
                            ; 目的状態は
                            ; ball が roomb にあるようになることである
)
```

2.3 生成されるプラン

上の PDDL ファイルを入力としてプランナが出力するのは例えば以下のようなプランである。

```
(move rooma roomb)
(pick ball rooma right)
(move roomb rooma)
(drop ball roomb left)
```

ここで，ボールを右手に持つか左手に持つかなどの自由度がプランにはあることを注意しておく。

また，明らかに冗長性を含むが，以下の行為の列もプランである。

```
(move rooma roomb)          ; 無駄に roomb に行く
(move roomb rooma)          ; 戻ってくる
(move rooma roomb)          ; 以下は前のものと同じ
(pick ball rooma right)
(move roomb rooma)
(drop ball roomb left)
```

プランニング問題ではプランの中でも長さが最小のもの，あるいは行為にコストを与えそのコストが最小のものを求めることを要求することもある。

3 モデル

以下，上述のように定義されたプランニング問題を他のモデルに変換することを考えていく。そのため，まずは本研究で扱った他のモデルである充足可能性モデル，整数計画モデル，Constraint Problem(CP, 制約問題) モデルの 3 つを定義する。

3.1 充足可能性問題

充足可能性とは，命題論理式が与えられた時に，その論理式を充足する命題変数への完全な値の割り当てが存在するかどうかを判定する問題である。

完全な割り当てとは，全ての変数にその定義域内の値を割り当てることである。

論理式は命題のリテラルと論理和，論理積の論理演算子から作られる。また，リテラルの論理和を節という。

定義には $\Rightarrow$ 論理演算子を含まなかったが，$p \Rightarrow q = \neg p \vee q$ でありモデルの表現力にはこの演算子を含んでも変化がないことに注意する。

3.2 整数計画問題

整数計画問題は目的関数，変数集合，不等式制約集合から構成される。

目的関数は変数集合内の変数の任意の関数である。整数計画問題においては，変数の定義域は離散的である。不等式制約集合は変数集合内の任意の変数の一次結合を左辺，右辺を定数とする不等式の集合である。一般の整数計画問題では不等式制約として任意の不等式を許すこともあるが，高速に解を求める事ができる線形不等式制約が広く使われる。この研究でも，不等式制約としては線形不等式のみを認めた。

整数計画問題の全ての変数に対して，全ての不等式制約を満たすような値の割り当てを整数計画問題の実行可能解という。実行可能解の集合を実行可能領域という。整数計画問題を解くとは，実行可能領域内に含まれる変数の割り当てのうち，目的関数を最大化するような変数の割り当てを得ることをいう。

ここで，目的関数 $f(x)$ の最大化は $-f(x)$ の最小化と同値であること，任意の等式制約を不等式制約に変換できること，最小値，最大値，絶対値といったものを含む不等式の一部を線形不等式に変形できることなどが知られていることを注意しておく。整数計画問題の表現能力とその具体的な変形方法については森&松井, 04 [21]，茨木&木村, 11 [22] などにまとまった記述がある。

この変数への割り当てを整数計画問題の最適解といい，この時得られる目的関数の値を最適値という。また，整数計画問題において単に解，といった場合，実行可能解ではなく最適解を表すことが多い。

線数計画問題と充足可能性問題の応用上の大きな違いとして，目的関数の存在がある。前者にはモデルに自然に解の評価値が与えられており，変数の値の任意の関数を目的関数として利用できることから，解の選好 (preference) を自然に扱うことが出来る。

生成されるプランの長さを最小化することは充足可能性問題においても行為に対応する命題の個数を最小化する，という形でモデル内で表現可能だが，実際の応用では資源の最小化，エネルギーの最小化，体積の最小化など様々な目的関数を用いて解の良さを定量化し，これをモデルの中で最適化出来ることは整数計画モデルの長所である。

また，モデルの変換の章でも述べるように，充足可能性問題は一般に整数計画問題に変換が可能である。

3.3 CP

Constraint Programming(以下 CP と表記する。) は，変数の集合とその定義域 $\{x_i\}(i = 0 \cdots n)$, $\{d_i\}(i = 0 \cdots n)$，制約の集合 $C = \{c_i\}(i = 0 \cdots m)$ から定義される。全ての変数には有限の定義域が与えられているものとする。

また，制約は全ての変数の関係として与えられる。

$$D = d_0 \times d_1 \times \cdots \times d_n$$

に対して

$$c_i \subseteq D \ (i = 0 \cdots m)$$

である。

ここで，制約は明示的な集合として与えられるとは限らないことに注意する。例えば，変数の集合 $X = (x_0, x_1)$ に対して $x_0 = x_1$ という等式の形で，これを満たす X の部分集合，すなわち制約が提示されることもある。

	表現方法	preference	論理制約	線形不等式制約	global constraint
SAT	論理式		○		
IP	線型不等式制約	○		○	
CP	変数ドメインと制約（論理，線形制約，その他）	○	○	○	○

図 3　モデルの比較

ここで全ての変数への値の割り当て a を考える

$$a = (v_0, v_1, \cdots, v_n) \in D$$

CP の充足解とは，全ての制約を満たすような，変数への完全な値の割り当てである。

すなわち，充足解 a は

$$a \in \bigwedge_{i \in 0 \cdots m} c_i$$

で与えられる。

このような解を見つける問題を Constraint Safisfaction Probrem(CSP，制約充足問題) という。

また，CP には目的関数を必ずしも含まないが，目的関数を利用することもできる。目的関数を利用し，解のうち目的関数を最大化するものを求める問題も含む，という意味合いで本論文では CP という。

すなわち，充足解の集合 A は以下のようになり，

$$A = \bigwedge_{i \in 0 \cdots m} c_i$$

目的関数 Obj を

$$Obj : X \to value \ (value \text{ は全順序な集合})$$

とした時に

$$\underset{a \in A}{Max}\{Obj(a)\}$$

を計算し，この時の a を求める問題である。

以上のように，CP では制約として，不等式制約，論理制約を共に扱え，また，目的関数の最大化もそのモデルに含む。すなわち，充足可能性問題，整数計画問題は一般に CP に変換が可能である。

4　プランニング問題のモデリング

プランニング問題を先に述べた 3 つのモデルに変換することができる。

CP モデルは，その制約として論理制約，不等式制約を利用することができるので，他の 2 つのモデルで表現することのできる問題を表現することはできる。

また，任意の論理式を論理式として同値な連言標準系に変換できる。(例えば Handbook of satisfiability [1] を参照) ここで，その連言標準系の各節内の変数の真偽値を 1,0 値として持つ変数 x_l を考え，全ての節 L_i に対して

$$\sum_{l \in L_i} x_l \geq 1$$

という制約を加えることで任意の充足可能性モデルの問題は整数計画モデルの中で表現可能である。

以上に述べたことから，３つのモデルのうち，充足可能性モデルでプランニング問題を表現することが出来たならば，整数計画モデル，CP モデルでもプランニング問題を表現できることがわかる。

以下ではまず，Kautz と Selman [11] で示されたプランニング問題の制約充足モデルの表現方法について解説する。続いて，素朴な問題エンコーディングに加えて，ソルバがより高速に解を求めるための工夫として大きな効果を発揮するプランニンググラフ [3] について説明する。

さらに，今回の実験で使用した optiplan で採用されているプランニング問題のエンコーディング方法 state change model [20] について説明する。

4.1 制約充足問題によるプランニング問題の表現

プランニング問題を制約充足問題で表現するにあたって，論理命題に状態を対応させる必要がある。論理命題と状態を単純に一対一対応させると時間の表現を含まなくなってしまうため，あるステップにおいてある状態に対応する命題が真であるということを SAT 命題を一つ使って表現する。すなわち，各ステップにおいて命題の真偽値を保持する SAT 命題を定義する。

ここまずで問題となるのは，事前にはプランの存在するステップの深さがわかっていないことである。論理式を構成するには関連する命題論理式を列挙出来る事が必要である。

そのため，まずは最大ステップ数を１として論理式を構成し，その論理式からプランが得られるならばそこで終了し，そうでないならば最大ステップ数を増やして論理式を作成し，プランが得られるかを検証，この手順をプランが得られるまで繰り返すこととする。

ステップ t での命題の真偽値と対応する命題の集合を P_t と表すこととする。

また，同様に，あるステップで行為が行われるかどうかに対応する命題も最大ステップ数以下の全てのステップについて定義する。すなわち，ステップ t での行為の実行の有無と対応する命題の集合を A_t と表すこととする。

プランニング問題を SAT に変換するには以下の条件を論理式で表現すれば必要かつ十分である。

- 初期状態制約
- 終了状態制約
- 次状態公理
- 前提条件公理
- 排他行為公理

初期状態は $t = 0$ での世界の状態を定める。(ここで p_s_t はプランニング問題の命題 s に対応するステップ t での命題である。すなわち $p\_s\_t \in P_t$ である。)

$$\bigwedge_{s \in I} p\_s\_0 \wedge \bigwedge_{s \notin I} \neg p\_s\_0$$

終了状態制約は終了状態 t=T での命題変数が真となっていることを要求する。

$$\bigwedge_{s \in G} p\_s\_T_{max}$$

次状態公理は，ある命題に対して，その命題が真となっているならば，直前のステップでその命題を真とする行為が実行されること，あるいは直前のステップでその命題は真でありかつその命題を偽とするような行為が実行されないことを要求する。(ここで p_a_t はプランニング問題の行為 a に対応するステップ t での命題である。すなわち $p\_a\_t \in P_t$ である。また，ADD_s，DEL_s はそれぞれ正の効果，負の効果に s を含むような行為の集合である。)

$$\forall_{t=0 \cdots T_{max}-1} \quad \bigwedge_{s \in S} p\_s\_t{+}1 \Rightarrow (\bigvee_{a \in ADD_s} p\_a\_t) \vee (p\_s\_t \wedge \bigvee_{a \in DEL_s} \neg p\_a\_t)$$

前提条件公理は，ある行為を実行するならば，その直前で行為が前提条件とする命題の割り当てが満たされていることを要求する。(ここで PRE_a は a の前提条件集合である。)

$$\forall_{t=0 \cdots T_{max}-1} \quad \bigwedge_{a \in A} p\_a\_t{+}1 \Rightarrow \bigwedge_{s \in PRE_a} p\_s\_t$$

排他行為公理は，あるステップではひとつの行為までしか実行されないことを要求する。

$$\forall_{t=0 \cdots T_{max}} \quad \bigwedge_{a1,a2 \in A} (a1 \neq a2 \Rightarrow \neg a1 \vee \neg a2)$$

以上の公理群を表現する命題を節とし，これらの論理積からなる論理式の真偽値はあるステップ数までの長さの直列プランが存在することと一対一対応を持つ。

4.2 プランニンググラフ

上に述べた定式化の問題点として，不要な命題が多く存在していることがあげられる。例えば，初期状態において偽である命題を前提条件として含む行為は t=0 で決して実行されることは無いため不要である。(先の gripper の例でいうと，初期状態ではロボットは部屋 B にいるので，部屋 A にあるボールを拾うことはできない。)

探索空間が命題数に対して指数的に大きくなることから，不要な命題をソルバの実行前に除くことで探索空間を大きく削減できる。

問題の構造から効率的にこれらのような不要な命題を除くことを可能にする工夫として，プランニンググラフを利用することが挙げられる。

プランニンググラフの構成自体の実行時間は元のプランニング問題の行為と命題の個数に対して多項式オーダーで行うことができ，これを利用して定式化の完全性を損なわない範囲で多くの不要な命題を除くことができるといった有用なデータ構造である。この手法は充足可能性モデルにとどまらず，整数計画モデル (optiplan) や CP モデル (GP-CSP) といった他のモデルにも応用され，広く使われている。

以下，プランニンググラフの構成の方法について説明する。

プランニンググラフは命題，行為をノードとし，その間の依存関係，相互排他関係をエッジとするグラフである。

まず，初期状態に含まれる命題ノードをグラフに加える。これらのノードをグラフ上の深さ 0 に存在する命題ノードと呼ぶことにする。

この後追加するノードについては，以下の相互排他関係を全ての同じ深さのノードの組について記録する。この相互排他関係はその深さのステップにおいて，共には実行されることのない行為，命題の組を表現している。

深さ t の命題ノードについては以下の条件を満たすノードの組に相互排他関係を記録する。

- 片方が他方の否定となっている。
- それらのノードを追加した t-1 の行為ノードが互いに相互排他関係にある。

深さ t の行為ノードについては以下の条件を満たすノードの組に相互排他関係を記録する。

- 一方の前提条件に含まれる命題が他方の負の効果に含まれる。
- 一方の正の効果に含まれる命題が他方の負の効果に含まれる。
- 一方の前提条件に含まれるノードと他方の前提条件に含まれるノードに相互排他関係がある。

続いて行為ノード，命題ノードを深さを増やしながら交互に追加していく。

深さ t の行為ノードとしてグラフに追加されるのは以下の条件を満たすものである。

- 深さ t の命題ノードに前提条件を全て含む。
- それらの前提条件集合の中に相互排他関係がない。

また，プランニング問題に含まれる行為の他に，深さ t の命題ノードを前提条件とし，正の効果としてその命題を含むような行為ノードを追加する。この行為ノードは，実行された行為の効果に含まれていない命題ノードに対応するノードには影響がないことを保証する。

深さ t の命題ノードとしてグラフに追加されるのは以下の条件を満たすものである。

- 深さ t-1 の行為ノード集合に含まれるある行為の正の効果に含まれている。

プランニンググラフは命題から余分なものを除くかつ必要なものを除かないことから，プランニンググラフの中で目的命題が全て活性化されるようなステップ T_{min} はプランの存在するステップ数の下限となる。この性質を利用すると，プランニングの定式化は $T = T_{min}$ から論理式の構成を開始することができる。

また，プランニンググラフに記録された相互排他な命題，行為の情報を使ってモデルを構築することもできる。

ここで，プランニンググラフによって活性化済みの命題の数は単調に増加すること，また，元のプランニング問題の命題の数は有限であることから，プランニンググラフの展開は有限回でレベルオフすることがわかる。一方，レベルオフして，かつゴール命題の全てが活性化されている場合にはその時点から有限回の展開によって解が得られることが示されている。

4.3 直列プランニングと並列プランニング

先に述べた定式化では，一つのステップに１つの行為のみを許す直列プランニングとしてプランニングをモデリングした。一方で，一つのステップに相互排他関係のない範囲で複数の行為を許す並列プランニングというモデリングもプランニング問題のモデルとして扱われている。

先の充足可能性モデルによるプランニングの定式化では，排他行為公理は全ての行為の間に相互排他を要求したが，この排他を適切に緩和すると，並列プランが得られる。

直列プランニングは並列プランニングの特殊な場合であると考えることができる。すなわち，並列プランニングの解が得られた場合には，同ステップの行為は任意の順序で実行可能であり，直列プランニングの解が得られていると言える。

このため，IPC の並列プランニング部門では行為を均一コストとして最短のプランを最適としている一方で直列プランニング部門では行為にそれぞれコストを指定してコスト最小のプランを最適プランとしている。

4.4 state change model

本研究で扱ったプランナ optiplan では state change model という定式化を利用してプランニング問題を整数計画にエンコードしている。

state change model によって整数計画モデルでは SAT の定式化を直接移植したものと比べて優れたパフォーマンスを示すこと，整数計画のソルバで利用される LP-relaxation から得られる下限が大きくなる傾向があることなどが Vossen99 [20] に報告されている。

state change model は Vossen99 で提案された定式化で，先のモデルでは命題の真偽値を各ステップに対して考えていたのに対して，命題の真偽値の遷移をブール変数として持つようにした定式化である。

すなわち，各ステップでの命題の真偽値を，以下の 4 種類の命題を使用して間接的に表現する。

- 真または偽から真に遷移する
- 真から真に遷移する
- 真から真または偽から偽に遷移する
- 真から偽に遷移する

state change model のうえで初期状態，終了状態，次状態公理，前提条件経理，無矛盾制約を満たす制約を記述すれば先に述べた充足可能性問題によるプランニングのモデリングと同値な問題を記述することが出来る。

また，この定式化を改良したものが Briel05 [18] に記されており，本研究では整数計画モデルのプランナの実装としてこの定式化を利用した。

4.5 optiplan

ここでは，(Briel,05 cite) に記された optiplan において利用されているプランニング問題の整数計画問題への変換方法を説明する。

充足可能性モデルと同様に，事前に解の長さがわかっていないことが問題となるため，この定式化でも最大ステップ数 T を $T = 1, 2, 3, \cdots$ というように，解が見つかるまで T を順次大きくしながらモデルを拡大再構築することを繰り返す。

まず，このモデルでは変数として，行為変数と state change variable を利用する。

ステップ t で行為 a が実行されたかどうかに対応する行為変数を $y_{a,t}$ とする。また，プランニング問題の行為の集合 A に対して以下の 3 つの部分集合を定義する。

- $Pre_p = \{a : p \in Pre(a)\}$
- $Add_p = \{a : p \in Add(a)\}$
- $Del_p = \{a : p \in Del(a)\}$

すなわち，Pre_p は命題 p を前提条件として持つような行為の集合，Add_p は命題 p を正の効果として持つような行為の集合，Del_p は命題 p を負の効果として持つような命題の集合である。

一方，ステップ t における命題 p の真偽値を表現するため，optiplan の state change model では以下の 5 つの state change variable を利用する。

state change variable の定義

1. $x_{p\_t}^{maintain}$: ステップ t で命題 p の真偽値が真のまま変化しないならば 0，そうでないならば 1
2. $x_{p\_t}^{add}$: ステップ t で行為 a($a \in Add_p \wedge a \notin Pre_p$) が実行されたならば 0，そうでないならば 1
3. $x_{p\_t}^{preadd}$: ステップ t で行為 a($a \in Pre_p \wedge a \notin Del_p$) が実行されたならば 0，そうでないならば 1
4. $x_{p\_t}^{del}$: ステップ t で行為 a($a \in Del_p \wedge a \notin Pre_p$) が実行されたならば 0，そうでないならば 1
5. $x_{p\_t}^{predel}$: ステップ t で行為 a($a \in Pre_p \wedge a \in Del_p$) が実行されたならば 0，そうでないならば 1

ステップ t において命題 p に対応する全ての state change variable の値を定めると，命題 p の値の遷移が一意に定まることに注意する。すなわち，これらの変数の値の組によって間接的に命題の真偽値を表現している。

optiplan では実行された行為の個数を目的関数としている。

$$Obj: \sum_{t \in 0 \cdots T} \sum_{p \in P} y_{a,t}$$

目的関数を定めなくとも実行可能解を得る事はできるが，目的関数を適切に定義することで整数計画ソルバの探索における枝刈りの助けとなり，解を得るのにかかる時間が小さくなる傾向があることが知られている。

続いて，不等式制約として以下のものを導入する。

- 初期状態制約
- 終了状態制約
- 論理関係 (logical relation) 制約
- state change variable の相互排他制約
- 後方連鎖 (backward chaining) 制約

初期状態制約は，プランニング問題において初期状態に入っている命題ははじめのステップで真となり，そうでない命題は偽となることを要求する。

$$x_{p\_0}^{add} = 1 (\forall p \in I)$$

$$x_{p\_0}^{preadd}, x_{p\_0}^{del}, x_{p\_0}^{predel} = 0 (\forall p \notin I)$$

終了状態制約は，プランニング問題において終了状態に入っている命題は最後のステップで真となっていることを要求する。

$$x_{p\_T}^{add} + x_{p\_T}^{preadd} + x_{p\_T}^{maintain} = 1 (\forall p \in I)$$

論理関係制約は行為変数と state change variable の定義が矛盾しないことを要求する。

以下，全ての命題 $p \in P$ と $t \in 1 \cdots T$ に対して，

1. state change variable の定義 2 で述べた x の定義が満たされることを要求している。

$$\sum_{a \in Add_p - Pre_p} y_{a,t} \geq x_{p\_t}^{add}$$

これは，$x_{p\_t}^{add} = 1$ ならば，p を正の効果として持ち，p を前提条件として持たないような行為がステップ t において少なくともひとつ実行されることを要求している。

$$y_{a,t} \leq x_{p\_t}^{add} (\forall a \in Add_p - Pre_p)$$

一方これは，p を正の効果として持ち，p を前提条件として持たないような行為がステップ t において実行されたならば，$x_{p\_t}^{add} = 1$ となっていることを要求している。

2. state change variable の定義 3 述べた x で述べた x の定義が満たされることを要求している。

$$\sum_{a \in Pre_p - Del_p} y_{a,t} \geq x_{p\_t}^{preadd}$$

これは，$x_{p\_t}^{preadd} = 1$ ならば，p を前提条件として持ち，p を負の効果として持たないような行為がステップ t において少なくともひとつ実行されることを要求している。

$$y_{a,t} \leq x_{p\_t}^{preadd} (\forall a \in Pre_p - Del_p)$$

一方これは，p を前提条件として持ち，p を負の効果として持たないような行為がステップ t において実行されたならば，$x_{p\_t}^{preadd} = 1$ となっていることを要求している。

3. state change variable の定義 4 で述べた x の定義が満たされることを要求している。

$$\sum_{a \in Del_p - Pre_p} y_{a,t} \geq x_{p\_t}^{del}$$

これは，$x_{p\_t}^{del} = 1$ ならば，p を負の効果として持ち，p を前提条件として持たないような行為がステップ t において少なくともひとつ実行されることを要求している。

$$y_{a,t} \leq x_{p\_t}^{del} (\forall a \in Del_p - Pre_p)$$

一方これは，p を負の効果として持ち，p を前提条件として持たないような行為がステップ t において実行されたならば，$x_{p\_t}^{del} = 1$ となっていることを要求している。

4. state change variable の定義 5 で述べた x の定義が満たされることを要求している。

$$\sum_{a \in Del_p \wedge Pre_p} y_{a,t} = x_{p\_t}^{predel}$$

これは，$x_{p\_t}^{predel} = 1$ ならば，p を負の効果として持ち，p を前提条件としても持つような行為がステップ t においてちょうどひとつ実行されることと，その逆も成り立つこと要求している。ここで，predel に関して，ある命題を負の効果，前提条件の両方に含むような行為は同じステップに実行することは出来ないのでこの式は等号になっていることに注意する。この等号から，$x_{p\_t}^{predel}$ の値は行為命題の値から決定されることがわかる。これは $x_{p\_t}^{predel}$ を定式化から除くことが可能であることを意味する。

state change variable の相互排他制約は，state change variable の定義から，同時に 1 となることの出来ない state change variable を定めている。

$$x_{p\_t}^{add} + x_{p\_t}^{del} + x_{p\_t}^{predel} + x_{p\_t}^{maintain} \leq 1 (\forall t \in \{0 \cdots T\})$$

$$x_{p\_t}^{preadd} + x_{p\_t}^{del} + x_{p\_t}^{predel} + x_{p\_t}^{maintain} \leq 1 (\forall t \in \{0 \cdots T\})$$

すなわち，$x^{add}_{p\_t}$ と $x^{add}_{p\_t}$ のみが同一のステップで 1 となることが出来る。

後方連鎖制約はある命題を前提条件に含んでいる行為があるステップで実行されるためには，その直前のステップにおいて前提条件に含まれている命題が真となっていることを要求する。

$$x^{preadd}_{p\_t} + x^{maintain}_{p\_t} + x^{predel}_{p\_t} \leq x^{add}_{p\_t-1} + x^{preadd}_{p\_t-1} + x^{maintain}_{p\_t-1} (\forall t \in \{1 \cdots T\})$$

以上に挙げた制約を全て満たすような変数の割り当ては並列プランニングの最適解と対応している。

初期条件制約，終了条件制約が初期状態と終了条件を満たすことを，相互排他制約が並列プランとしての健全性を，後方連鎖制約が前提条件を満たしている行為のみが実行されることを，保証しており，この定式化は健全であるといえる。

また，$T = 1, 2, 3 \cdots$ と最大ステップ数を大きくしながら順次モデルを拡大していくことから，初めて解を見つけた場合そのプランは並列プランとしての最適性を持つ，すなわち最短のプランである。

5 実験: optiplan のモデルの再現実装と SATPLAN2006 との性能比較

国際会議 ICAPS と共に行われるプランニングシステムのコンテスト International Planning Competition(以下 IPC とする) の過去のベンチマーク問題集 (fastdownward システムに付属しているもの) のうち，本研究で実装したプランナの対象とする問題である全てのドメイン (表 1 に一覧した) に対して本研究で実装したプランナと比較対象のプランナ群 (表 2) を実行した。

表 1　実験に使用したドメイン一覧

大会	ドメイン名
IPC98	gripper, logistics98, blocks, grid, mprime, mystery, movie
IPC00	miconic
IPC02	depot, driverslog, zenotravel, satellite, rovers, freecell
IPC04	psr-small, pipesworld-tankage
IPC06	tpp, storage, pathways, trucks-strips
IPC08	elevators-sat08-strips, elevators-opt08-strips, parcprinter-08-strips

これらの問題集合はプランニング問題の標準的な形式である Planning Problem Definition Langage(PDDL) にしたがって記述されたドメインファイルと問題ファイルの組で与えられる。

プランナはそれらのファイルを入力とし，実行時間の制限 5 分以内に実行可能なプランを出力出来たものをその問題を解けた，とみなし，その実行時間を記録した。

表 2　実験に使用したプランナ一覧

プランナ	詳細
SATPLAN2006	http://www.cs.rochester.edu/~kautz/satplan/のもの
optiplan	Gurobi Optimizer での再実装，1 コア
optiplan	Gurobi Optimizer での再実装，8 コア

実行時間の計測は入力ファイルからモデルを生成する時間と，生成したモデルを元にソルバが求解を行う時間を分けて計測した。また，出力されたプランの正しさの検証には VAL [8] というツールを利用した。

また，実行時に生成される各ソルバの入力サイズを bzip2 で圧縮したもののファイルサイズを比較した。この比較から，問題を各モデルに変換したソルバ入力ファイルの情報量の大きさを近似的に見積もることかできるであろう。

Gurobi Optimizer のバージョン 6.0.0, SATPLAN2006 の linux binary を利用し，実験は ubuntu14.04 上で行った。

5.1 optiplan の実装

optiplan プランナは Briel, Kambhampati, Subbarao [18] によって実現されたプランナであり，整数計画モデル，ソルバによって実装されているプランナにして初めて International Planning Competition に参加したプランナである。

本研究では，近年の整数計画ソルバの進歩による効果実証を行うため，optiplan 論文にしたがってそのモデルを再実装した。その際に，optiplan 現論文では商用ソルバ CPLEX [5] を利用した実装となっていたが，本研究の実装では近年のベンチマーク [12] で好成績を上げている商用ソルバ Gurobi Optimizer [14] を利用して実装を行った。また，実装の言語には C++ を利用した。実装はプランニンググラフの生成，モデル生成が 440 行，パーサが 230 行，その他ツールが 100 行程度となった。

さらに，当時の環境と比べ，ソルバの並列実行に対する対応が進んでいることが考えられるため，単一コア，複数コアでの計測を行うこととした。

optiplan の定式化の特徴としてはプランニンググラフ利用していることが挙げられる。また，state change model として Vossen の研究で述べられた定式化を工夫し，IPC のドメインで繰り返し現れる状態遷移を効率良く表現できるよう改良されたモデルを利用して実装していることが挙げられる。

モデルは optiplan の論文の定式化を実装し，論文には数式として提示されていなかったが実装の工夫として述べられていた predel 遷移命題の削除を行った。本実装ではプランニンググラフから得られる並列プランの長さの下限を利用し，その深さから探索を開始した。

5.2 SATPLAN2006

SATPLAN は SATPLAN2006 として第 5 回 IPC(2006 年) の Optimal Planning(Propositional Domains) 部門，SATPLAN2004 として第 4 回 IPC(2004 年) の Optimal 部門での優勝ソルバであり，Kautz らのホームページでバイナリファイルが公開されている。

特徴としては，プランニンググラフを利用した制約充足モデルのプランナである。また，内部で局所的な探索を行うソルバ，大域的な探索を行うソルバなどを使い分けている。前者のソルバでは解の最適性を保証することはできないが，確率的かつ貪欲な探索を行うことで高速に解を発見できることがある。後者のソルバでは完全性を保証した探索が可能であり，解が存在するならば必ずその解を発見するが，問題によっては非常に多くの時間を消費する。以上の長所，短所を組み合わせ，局所的な探索と大域的な探索を切り替えて利用することでより高性能な探索が可能になっていることが特徴である。

5.3 結果

ドメインごとの正しいプランが出力された問題数をまとめた。表 3 がその結果である。表において特に注記していないプランナの実行環境は 1 コア，実行時間 5 分である。SATPLAN2006 は IPC 参加当時のプログラムが提供されており，IPC5 以外のドメインでは不安定であり，fail となっている欄はエラーを出力し，うまく動作しなかった。また，IPC4 optiplan の欄は http://ipc.icaps-conference.org/からの結果の引用であり，実行時間が 30 分であることに注意する。

表 3　optiplan とその再実装の比較（セル内の数字は解けた問題数である。ドメインごとの問題数をドメイン名の後ろの括弧内に記した。また，SATPLAN2006 の実行時エラーの会ったドメインは fail, optiplan の IPC4 の結果に含まれていないドメインはハイフンを記した。）

プランナ	SATPLAN2006	optiplan 再実装 1 コア	optiplan 再実装 8 コア	IPC4 optiplan
gripper(2)	3	3	3	-
logistics98(35)	fail	6	8	-
blocks(35)	34	17	17	-
grid(5)	1	1	1	-
mprime(35)	fail	25	25	-
mystery(30)	17	15	15	-
depot(22)	16	7	10	-
driverlog(20)	15	10	9	-
zenotravel(20)	15	11	12	-
psr-small(50)	fail	31	35	29
pipesworld-tankage(50)	fail	7	7	9
tpp(30)	24	8	8	-
storage(30)	15	9	9	-
elevators-sat08-strips(30)	fail	1	2	-
elevators-opt08-strips(30)	fail	12	16	-
movie(30)	fail	29	30	-
miconic(150)	23	41	45	-
satellite(36)	16	5	5	8
rovers(40)	15	12	13	-
freecell(80)	21	8	19	-
pathways(30)	fail	7	7	-
trucks-strips(30)	fail	1	1	-
parcprinter-08-strips(30)	fail	30	30	-
合計 (868)	214	296	327	-

5.4 考察

SATPLAN2006 が optiplan の再実装と比べて全体的によりよい結果となったのは，SATPLAN2006 ではモデルの工夫だけでなく，実行時の工夫として性質の異なる複数のソルバを使い平均的に良い結果を得られる用になっているのに対して，今回 optiplan の実装として実装したのはモデルとプランニンググラフによる枝刈りのみであったことが理由としてあげられる。

同じモデルを利用している optiplan のモデルに関しては，IPC4 の optiplan の結果は実行時間 30 分，今回の実験は実行時間 5 分であったにも関わらず，3 ドメイン中 2 ドメインで今回の再実装がより多くの問題を解いた。これは整数計画ソルバの近年の大きな高速化のためであると考えられる。

表 4　speedup 値 (= optiplan モデル (1 コア) の実行時間 / optiplan モデル (8 コア) の実行時間)

ドメイン	speedup	共通で解けた問題数
gripper	2.1958	3
logistics98	1.8029	6
blocks	2.0107	17
grid	0.9835	1
mprime	1.9686	25
mystery	2.0275	15
depot	2.4363	7
driverlog	2.4852	10
zenotravel	2.3426	11
psr-small	2.7802	31
pipesworld-tankage	1.7814	7
tpp	1.8077	8
storage	2.2128	9
elevators-sat08-strips	2.0376	1
elevators-opt08-strips	3.9142	12
movie	2.0071	29
miconic	1.9633	40
satellite	2.9370	5
rovers	2.7389	12
freecell	1.8653	18
pathways	1.8714	7
trucks-strips	1.4661	1
parcprinter-08-strips	2.0654	30
合計	2.3804	305

optiplan の再実装 1 コアと 8 コアの比較は表 4 に掲載した。これを見ると，スピードアップ値はドメインに

よるばらつきはあるものの，2 から 3 の間くらいで並列計算の一定の効果は確認出来る一方で，8 コアに対しての効率は必ずしもよいとは言えない結果である。

optiplan の発表された 2000 年代半ばと比べ，現在ではより一層並列計算を行える環境が一般に普及している。Gurobi Optimizer などの近代的な整数計画ソルバでは並列コアを活かした探索を行うので，並列化が有効に働くような定式化についても考えてみる価値があろう。

6 CP モデル

一定の成果を挙げた CP モデルでのプランナの実装として GP-CSP [6] がある。これは GRAPHPLAN の成果を取り込んだプランニンググラフ解析を行う CP モデルプランナの実装である。

一方で, 充足可能性モデル, 整数計画モデルの成果と比べて, CP モデルを利用したプランナは International Planning Competition での実績がない。

各モデルの優劣に対して決定的な理論付けがないままに CP モデルの研究が進展していない近年の状況に対して問題意識を持ち，本研究では充足可能性モデル，整数計画モデルで得られた知見を利用した CP モデルプランナの実装を行い，その性能を測定，比較した。

6.1 global constraint

global constraint と呼ばれている制約が存在する。これらの制約は，素朴な制約 (local constraint) の積集合として表現可能である。すなわち，一つの制約で複数の制約を表現する。

local constraint としては二項間の相互排他や，変数集合 X の要素に v が少なくとも n 個割り当てられることを要求する制約 at-least(X,v,n) 制約，変数集合 X の要素に v が最大で n 個まで割り当てられることを要求する制約 at-most(X,v,n) などがある。

これに対して，多項間の相互排他関係 all-different や，変数集合に対して複数の値の出現回数の上限，下限を要求する cardinality constraint が global constraint となっている。

かつては素朴な制約を利用して基礎的な理論の構築が行われたが，近年実際の実行時処理の有効性から global constraint に再度注目が集まり，各制約の性質が盛んに研究されている。

実問題への理論の適用を考えた場合，制約を利用して解を計算する必要がある。この過程において，構造を含んだ global constraint を利用して表現された問題の方が，構造を含まない local constraint のみから構成された問題よりも，少ない制約伝播で解空間の縮小を行え，その結果として高速に解を計算することが出来ることがあるからである。

CP ソルバでは制約伝播法によって探索空間の縮小を行いながらバックトラック法によって探索的に解を計算する。その過程で，global constraint を利用した制約伝播では一度の制約伝播によってより大きく探索空間を削減することが出来る。この例は次節で紹介する。

また，素朴な制約の集合から等価な global constraint を得ることは可能であるか，その計算を行うことは必ずしも簡単ではない。

例えば，2 変数間の相互排他制約の集合が与えられたとき，そこから要素数最大の変数の集合に対する all-different 制約を構築することは最大クリークの計算と等価であり，この計算は NP 困難である。一方で，all-different を二項制約の積に分解することは容易である。このように，global constraint と local constraint の間には相互変換が必ずしも容易では無いものがある。

すなわち，最初から global constraint をモデルの中で扱い問題を定式化することで，素朴な定式化と比べてより変数の構造に対する情報を利用して解を計算することが出来る可能性があるのである。

6.2 global constraint : among

本研究では，プランニング問題に optiplan のプランニング問題のモデリングに対して，among という global constraint が適用可能であることに着目した。以下，among がどのような global constraint であるかを説明する。

$among\{list, int, set\}$ は list 内に，set の要素がちょうど int 回現れることを要求する global constraint である。

幾つか例を挙げると，$among\{[1,2,3,1],2,\{2,3\}\}$ は成り立つ。なぜなら、$[1,2,3,1,2,3,1]$ の中に 2 が一回、3 が一回、すなわち $\{2,3\}$ の要素が合計ちょうど 2 回現れているからである。一方，$among\{[1,2,3,1,2,3,1],3,\{2,3\}\}$ は成り立たない。なぜなら、$[1,2,3,1,2,3,1]$ の中に 2 は二回、3 は二回、すなわち $\{2,3\}$ の要素が合計 $4(<3)$ 回現れているからである。同様に，$among\{[1,2,3,1,2,3,1],2,\{2,3\}\}$ は成り立たない。なぜなら、$[1,2,3,1,2,3,1]$ の中に 2 は二回、3 は二回、すなわち $\{2,3\}$ の要素が合計 $2(>3)$ 回現れているからである。

ここで，fastdownward システムの PDDL の translator は各命題変数の相互排他関係検出し，それらの命題を一つの多値変数で表現した形式でモデルを出力することが出来る。この相互排他関係を本研究で実装したプランナでは among 制約で表現し，optiplan のモデルに追加した。

optiplan のモデルでは全ての変数は 0-1 変数である。また，各命題 p に対応する state change 変数は $x_{p\_t}^{add}, x_{p\_t}^{del} x_{p\_t}^{preadd}, x_{p\_t}^{predel}, x_{p\_t}^{maintain}$ というように複数ある。

補助変数 $x_{p\_t}^{sum}$ を以下のように定義する。

$$x_{p\_t}^{sum} = x_{p\_t}^{add} + x_{p\_t}^{preadd} + x_{p\_t}^{maintain}$$

ここで，ある命題 p がステップ t で真になる際には

$$x_{p\_t}^{sum} \geq 1$$

が成立している。

すなわち，相互排他集合 $mutex$ が与えられたとき，$\forall p, q \in mutex$ に対して

$$\bigwedge_{t \in 0..T} x_{p\_t}^{sum} = 0 \vee x_{q\_t}^{sum} = 0$$

である。また，translator の出力は多値変数なので，多値変数の定義域に対応する命題のうち一つは真であることも必要である。

among 制約を使えば上の条件を表現することが出来る。すなわち，

$$among\{[x_{p\_t}^{sum} \mid p \in mutex], \#(mutex) - 1, \{0\}\}\{\forall t \in \{0..T\}\} \quad (\#(mutex) \text{ は } mutex \text{ の要素の個数})$$

は，mutex 内の sum 変数がただひとつだけ 0 でないことを要求する。

ここで，先の local constraint $x_{p\_t}^{sum} = 0 \vee x_{q\_t}^{sum} = 0$ の積を among が単一の制約として表現していることから，among は global constraint であることも確認出来る。

この制約を optiplan のモデルに添加すると，プランナは命題間の相互排他の情報を事前に与えられてプランの計算を行う事ができる。

図 4　CP ベースプランナの概要

7　実験: global constraint を利用した CP モデルのプランナ実装の検証

本研究では global constraint を利用したプランニング問題の CP への変換方法の有効性を検証する。ここで global constraint として among 制約を利用してプランニング問題を定式化する。

7.1　minizinc

minizinc という CP モデルを記述するための標準化された高級言語がある。また，minizinc 言語はより低レベルな flatzinc という CP モデルを記述するより低級な言語への変換が可能である用に設計されている。minizinc 言語は mzn2fzn というツールによって flatzinc 言語にコンパイル可能である。

CP の多くのソルバは flatzinc を入力として受け入れるインターフェースを持っており，こうしてモデルの実装と記述を切り離すことができている。

本研究での CP モデルのソルバとしては近年の制約プログラミングソルバのコンテスト MiniZinc Conpetition で優秀な実績を収めている gecode と，g12-lazy を利用した。

また，実装は C++ 言語で PDDL で記述されたプランニング問題を minizinc 言語に変換し，これを CP のシステムの入力とし，解を計算した。

この際，変換後の minizinc 言語をテキストファイルとして出力し，MiniZinc システムにこれを入力として与えた。実際のパフォーマンスを考慮すると，同一プログラム内でデータの受け渡しを行うのに比べてテキストファイルに書き出してモデルをソルバに渡すのは不利であるが，MiniZinc システムのプログラミング言語のためのインターフェース libmzn はまだ開発途中であるため利用できない。

そのため本研究ではテキストデータを介してモデルジェネレータとソルバの間の minizinc ファイルの受け渡しを行った。

しかしながら，CP モデルは表現力豊かであり，充足可能性モデル，整数計画モデルと比べて書きだされる minizinc ファイルの大きさは比較的小さい。一方モデルから解を得る計算は問題のサイズに対して最悪の場合指数的に大きくなると予想される。そのため，minizinc ファイルの書き出し，読み取りにかかる時間は無視できる程度に小さいと考えられ，また，実際にどの問題でもその時間は数秒以内に収まった。これは実行時間５分に対して十分小さいと考えられる。

まとめると，本研究の CP プランナの実装では PDDL 形式で与えられたプランニング問題を，minizinc 形式に変換するモデルジェネレータを実装し，これを flatzinc 言語に mzn2fzn で変換し，CP ソルバでその解を計算する CP モデルベースのプランナを実装した。(図 7.1)

7.2 CP プランナの実装

本研究で実装した CP ベースのプランナは，PDDL 形式で記述されたファイルを入力とする。入力ファイルを fastdownward システムの translator によって SAS フォーマットに変換する。この SAS ファイルから，C++ で実装したプランニンググラフ抽出ツールを利用してプランニンググラフを構築する。また，それを利用して minizinc 言語のテキストファイルとしてを出力する。minizinc システムに出力したテキストファイルを入力し，この解を計算することで，もとのプランニング問題の解が得られる。

ここで，minizinc システムへの入力としてテキストファイルへの書き出しを行っているが，この点はプログラミング言語の API を利用して問題データの受け渡しが出来た方が実行速度の面で望ましい。この点を考慮し，実験はプランナの各モジュールに対してその実行時間を計測した。

ベンチマークとして，fastdownward に付属の IPC の過去問題集のうち，本研究で実装したプランナの扱える全てのドメイン (表 5 に一覧した) に対して本研究で実装したプランナと比較対象のプランナ群 (表 6) を実行した。

表 5　実験に使用したドメイン一覧

大会	ドメイン名
IPC1 (1998)	gripper, logistics98, blocks, grid, mprime, mystery, movie
IPC2 (2000)	miconic
IPC3 (2002)	depot, driverslog, zenotravel, satellite, rovers, freecell
IPC4 (2004)	psr-small, pipesworld-tankage
IPC5 (2006)	tpp, storage, pathways, trucks-strips
IPC6 (2008)	elevators-sat08-strips, elevators-opt08-strips, parcprinter-08-strips

これらの問題集合はプランニング問題の標準的な形式である Planning Problem Definition Langage(PDDL) にしたがって記述されたドメインファイルと問題ファイルの組で与えられる。

プランナはそれらのファイルを入力とし，実行時間の制限 5 分以内に実行可能なプランを出力出来たものをその問題を解けた，とみなし，その実行時間を記録した。

表 6　実験に使用したプランナ一覧

プランナ	詳細
optiplan	Gurobi Optimizer での再実装
CP ベース	optiplan のモデルを minizinc に変換，ソルバは mzn-g12lazy
CP ベース	optiplan のモデルを minizinc に変換，global constraint 添加，ソルバは mzn-g12lazy

実行時間の計測は入力ファイルからモデルを生成する時間と，生成したモデルを元にソルバが求解を行う時間を分けて計測した。また，出力されたプランの正しさの検証には VAL [8] というツールを利用した。

また，実行時に生成される各ソルバの入力サイズを bzip2 で圧縮したもののファイルサイズを比較した。この比較から，問題を各モデルに変換したソルバ入力ファイルの情報量の大きさを近似的に見積もることかでき

るであろう。

minizinc のバージョン 2.0 とそれに含まれるソルバ g12-lazy を利用し，実験は ubuntu14.04 上で行った。

7.3 結果

ドメインごとの正しいプランが出力された問題数を表 7 にまとめた。先にも述べたように，SATPLAN2006 は IPC 参加当時のプログラムが提供されており，IPC5 以外のドメインでは不安定であり，fail となっている欄はエラーを出力し，うまく動作しなかった。また，IPC4 optiplan の欄は http://ipc.icaps-conference.org/ からの結果の引用であり，実行時間が 30 分であることに注意する。

表 7　CP-optiplan のベンチマーク（セル内の数字は解けた問題数である。ドメインごとの問題数をドメイン名の後ろの括弧内に記した。また，SATPLAN2006 の実行時エラーの会ったドメインは fail, optiplan の IPC4 の結果に含まれていないドメインはハイフンを記した。）

プランナ	CP-optiplan	CP-optiplan with among	optiplan 再実装	SATPLAN2006	IPC4 optiplan
gripper(2)	2	3	3	3	-
logistics98(35)	6	12	6	fail	-
blocks(35)	11	11	17	34	-
grid(5)	0	0	1	1	-
mprime(35)	20	20	25	fail	-
mystery(30)	13	13	15	17	-
depot(22)	7	8	7	16	-
driverlog(20)	11	14	10	15	-
zenotravel(20)	13	14	11	15	-
psr-small(50)	43	42	31	fail	29
pipesworld-tankage(50)	6	7	7	fail	9
tpp(30)	12	15	8	24	-
storage(30)	9	9	9	15	-
elevators-sat08-strips(30)	3	6	1	fail	-
elevators-opt08-strips(30)	16	29	12	fail	-
movie(30)	30	30	29	fail	-
miconic(150)	16	38	41	23	-
satellite(36)	3	13	5	16	8
rovers(40)	22	19	12	15	-
freecell(80)	6	8	8	21	-
pathways(30)	7	7	7	fail	-
trucks-strips(30)	1	2	1	fail	-
parcprinter-08-strips(30)	27	25	30	fail	-
合計 (868)	284	345	296	214	-

7.4 考察

実行結果は表 7 のようになった。この結果を見ると，グローバル制約 among を optiplan のモデルに添加したプランナが解けた問題の数の合計で最も良い結果を出していることがわかる。

整数計画の商用ソルバ Gurobi Optimizer を利用して実装した optiplan のマルチコアの場合の結果 (表 3) も上回っており，among 制約を添加していることを除くと論理的等価なモデルであることを考慮すると，グローバル制約 among を添加した事による高速化が強く働いている事がわかる。

個別の問題ごとの実行時間を比較してみると global constraint を添加した CP プランナがそうでない CP プランナよりも必ずしも早いわけではない傾向があったが，これは global constraint を添加したため余剰な制約を計算する必要が生じるためのオーバーヘッドであると言える。余剰な制約は定式化の工夫によって除ける可能性があるため，これは今後の課題である。

表 8　全てのプランナがモデルを生成できた問題集合 (563 問) について，モデルファイルを bzip2 で圧縮した場合のファイルサイズの平均値 (単位は byte である。また，ドメイン名の後ろに各ドメインの問題数を記した。)

プランナ	CP-optiplan	CP-optiplan(among 制約添加)	optiplan 再実装	SATPLAN2006
gripper(20)	6258	92767	52777	239583
ogistics98(20)	63342	127863	621781	1265246
blocks(35)	7367	31479	62032	1770222
grid(4)	141292	46413	871223	18057729
mystery(17)	97511	67909	633971	1599221
depot(21)	76334	77290	799698	7590199
driverlog(19)	27780	28296	281964	1925665
zenotravel(17)	41626	42221	238809	3500288
psr-small(50)	14277	14743	111321	106954
pipesworld-tankage(26)	112681	108968	1060168	17718661
tpp(25)	72705	74230	591925	1263048
storage(30)	173122	174791	1233072	5261584
movie(30)	1783	2024	2856	783
miconic(150)	12690	13283	1455044	4026849
satellite(21)	40514	42267	3362502	2272437
rovers(31)	76157	78489	545779	9260737
freecell(31)	56353	138168	1376763	14421452
trucks-strips(16)	128625	108931	1033287	4050482
合計	46784	56488	899009	4534890

全てのプランナがモデルを生成できた問題について最後に生成されたモデルファイルを圧縮したものの平均

ファイルサイズを表 8 に掲載した。ただし，この表では解けていないドメインも含んでいるため，全てのプランナが同じレベルまでモデルを生成してはいないことに注意する。例えば，一部のドメインにおいて，among 制約を添加した CP-optiplan のモデルの大きさが among 制約を添加していない CP-optiplan のモデルの大きさより小さくなっているのはこのためであると考えられる。

この表を見ると，モデルファイルの大きさは CP，整数計画，充足可能性の順に一桁程度のオーダーで大きくなっていることがわかる。充足可能性モデルのプランナは素朴な制約しか利用できないことから，モデルファイルが大きくなり過ぎてメモリの制約から問題が解けなくなることも多く，この性質は有用なものである。

今回の実験は実行時間が 5 分と短く，メモリの限界に達してしまう問題が少なかったと考えられるが，より大きなサイズの問題を解く際に，CP モデルを採用するメリットであると言えるだろう。

8 おわりに

本研究では古典的プランニング問題を充足可能性，整数計画，CP のモデルに変換して各モデルの汎用ソルバを実行し解を求めるプランニングアプローチについての実装，比較実験を行った。特に整数計画のソルバに関しては近年高速化が大きく進んでおり，マルチコア化の影響も計測した。

さらに，整数計画モデルのプランナ optiplan のモデルを CP モデルに移植し，また global constraint をその定式化に取り込む方法を試みた。

この実験では，global constraint を添加した定式化で素朴な定式化と比べ大きな高速化となることが確認できた。

今後の課題として挙げられるものとしては例えば以下のものがある。本研究では整数計画モデルで成果を挙げた optiplan の定式化に制約を追加する形で補助的に global constraint を加えたが，そもそもの定式化を global constraint を利用した CP モデルで行う前提で考えたならばよりよい定式化の方法があるかもしれない。例えば，CP モデルと充足可能性モデルとの違いとして多値変数が扱える点があり，本研究でも多値変数の各値の排他制約を利用したように，多くの global constraint は多値を変数を扱えるようになっているので多値変数を扱えるプランニング問題の定式化をもとに CP のモデルを構築することが考えられる。

本研究では既存のツールから抽出出来る情報で定式化に添加可能ななものとして among 制約を利用したが，global constraint には他にも多くの種類があり，他の制約を利用した定式化も十分に考えられる。

他には，現在モデルによるアプローチで最高の成果をあげているのプランナである Madagascar Planner(Rintanen, 11 [16]) の手法を CP モデル，整数計画モデルに輸入することも考えられる。Mp は多くの高速化の工夫を取り込んでいるが，本研究で扱ったプランナ群との最も大きな違いとしては，Mp の充足可能性問題ソルバは汎用ソルバでなく，Mp のために実装されたプランニング問題に特化して実装された充足可能性問題ソルバである点である。

CP についてもそのようなソルバを実装すれば高速化を図れるであろう。一方で，CP ではあるプランニング問題の定式化に利用できる global constraint に対してソルバが高速化を行えば，プランニング問題に特化のソルバを実装せずとも同様の高速化が得られる可能性も考えられる。

しかしながら，PDDL 言語は応用上の新たなニーズの現れや，人工知能分野の野心を反映して数年毎に新たな標準が登場することをここで考えておく必要がある。ソルバにプランニング特化の工夫を施すことはこの変化に対応するためにはその度にソルバの最適化をやりなおす必要が生じるが，ソルバの汎用性を保ったままモデル上の工夫で同様の高速化が行えれば，モデルに新たな要素を加えるだけで PDDL の変化に対応することが出来る。この点は CP と global constraint によるモデリングアプローチの優位性のうちの一つであると

言えより進んだ成果が期待される。

9 謝辞

右も左もわからない私に日々教育的視点からためになるアドバイス，課題を示してくださった福永先生には大変感謝しています。

本研究の審査を務めていただきました金子先生，田中先生，福永先生，山口先生，お忙しい中担当いただきましてありがとうございます。

また，本論文に関して有益なご意見を頂いた福永研のみなさまにも感謝しています。

最後に，学際科学科１期生の諸君，特に学生控室で日々議論に付き合ってくれた陣内くん，僕にプログラミングの手引を与えてくれた田中くんを始め，みなさまと共に，またみなさま自身からも多くのことを学びました。ありがとうございました。

索引

参考文献

[1] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, Vol. 185. IOS Press, 2009.

[2] Robert E Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, Vol. 50, No. 1, pp. 3–15, 2002.

[3] Avrim L Blum and Merrick L Furst. Fast planning through planning graph analysis. *Artificial intelligence*, Vol. 90, No. 1, pp. 281–300, 1997.

[4] Tom Bylander. Complexity results for planning. In *IJCAI*, Vol. 10, pp. 274–279, 1991.

[5] IBM ILOG CPLEX. 12.2 user's manual, 2010.

[6] Minh Binh Do and Subbarao Kambhampati. Planning as constraint satisfaction: Solving the planning graph by compiling it into csp. *Artificial Intelligence*, Vol. 132, No. 2, pp. 151–182, 2001.

[7] Malte Helmert. The fast downward planning system. *J. Artif. Intell. Res.(JAIR)*, Vol. 26, pp. 191–246, 2006.

[8] Richard Howey, Derek Long, and Maria Fox. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pp. 294–301. IEEE, 2004.

[9] H Kautz, B Selman, and J Hoffmann. Satplan: Planning as satisfiability. *Booklet of the 2006 International Planning Competition*, 2006.

[10] Henry Kautz. The role of domain-specific knowledge in the planning as satisfiabuity framework. 1998.

[11] Henry A Kautz, Bart Selman, et al. Planning as satisfiability. In *ECAI*, Vol. 92, pp. 359–363, 1992.

[12] Bernhard Meindl and Matthias Templ. Analysis of commercial and free and open source solvers for linear optimization problems. *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*, 2012.

[13] Roger Mohr and Thomas C Henderson. Arc and path consistency revisited. *Artificial intelligence*, Vol. 28, No. 2, pp. 225–233, 1986.

[14] Gurobi Optimization. Gurobi optimizer, 2013.

[15] Jean-Charles Régin. Global constraints and filtering algorithms. In *Constraint and Integer Programming*, pp. 89–135. Springer US, 2004.

[16] Jussi Rintanen. Madagascar: Efficient planning with sat. *The 2011 International Planning Competition*, p. 61, 2011.

[17] Stuart Jonathan Russell, Peter Norvig, 康一, 古川. エージェントアプローチ人工知能. 共立出版, 2008.

[18] Menkes van den Briel and Subbarao Kambhampati. Optiplan: Unifying ip-based and graph-based planning. *J. Artif. Intell. Res.(JAIR)*, Vol. 24, pp. 919–931, 2005.

[19] Willem-Jan van Hoeve. The alldifferent constraint: A survey. *arXiv preprint cs/0105015*, 2001.

[20] Thomas Vossen, Michael O Ball, Amnon Lotem, and Dana Nau. On the use of integer programming models in ai planning. 1999.

[21] 雅夫, 森, 知己, 松井, 隆夫, 円川. オペレーションズ・リサーチ. 朝倉書店, 2004.

[22] 木村俊房, 茨木俊秀. 最適化の数学. 共立出版, 2011.