人 工 知 能 89－3
（1993．8．18）

ブロックスワールドを用いた各種プランナの定量的評価

松原繁夫† 石田亨‡
†NTTコミュニケーション科学研究所
‡京都大学工学部情報工学科

この論文ではプランナSTRIPS, NONLIN, TWEAKにブロックスワールドの問題を与え定量的評価を行う．これまでに多くのプランナが提案されている．しかし，各プランナは動作系列生成方法の提案にとどまり，どの程度の問題解決能力を有するか明らかでない．本研究では，まず評価問題としてのブロックスワールドの問題を定義する．この問題を上記のプランナに与え，計画の長さ，計算時間などの面から評価する．実験結果より以下のことが示される．(1) 線形プランナSTRIPSでは，副目標間の干渉の強い問題に対して，計画の長さ，計算時間共に指数関数的に悪化する．(2) 非線形プランナNONLINは計画の長さに関しては優れた性能を示すが，問題が大規模になると，副目標間の干渉除去の負荷が大きくなり，計算時間が指数関数的に増大する．(3) 非線形最小拘束プランナTWEAKは，NONLINにない副目標間の干渉除去法を有するが，どのような場合にどの方法を用いればよいか不明である．そのため，探索制御が難しくなり，十分な問題解決ができない．また，ブロックスワールドの問題はプランナの例題として多く用いられているが，簡易解法による解と最適解との間にほとんど差がなく，プランナの計画の品質の評価用問題として適切とはいえないことが明らかにされる．

Quantitative Evaluation of Blocksworld Planners

Shigeo Matsubara† Toru Ishida‡
†NTT Communication Science Laboratories
2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02, Japan
‡Department of Information Science, Kyoto University

This paper evaluates the available planners such as STRIPS, NONLIN, TWEAK by using so called the blocksworld problem. Many planners have been proposed so far. However, the research on these planners only describes the mechanisms of making plans, and their performances are not well known. In this paper, we first define the blocksworld problem. Using this problem, we evaluate the three planners in terms of plan length and CPU time. Experiments will show the following. (1) The plan length and CPU time of the linear planner, STRIPS, get worse exponentially as the interaction between subgoals increases. (2) The nonlinear planner, NONLIN, has better performance than STRIPS in terms of the plan length. However, the CPU time of NONLIN also gets worse exponentially for a large problem. (3) The nonlinear and least commitment planner, TWEAK, has more interaction-removal methods than NONLIN. However, TWEAK can solve only small problems for lack of the knowledge about how to use the methods. We will also show that blocksworld problems have been used to illustrate many planners, however, they are not appropriate for evaluation because there are little differences between trivial solutions and optimal solutions.

1 まえがき

プランニング研究においては, GPS[Newell and Simon, 1963], STRIPS[Fikes and Nilsson, 1971], [Nilsson, 1980] をはじめ, ABSTRIPS[Sacerdoti, 1973], NOAH[Sacerdoti, 1975], NONLIN[Tate, 1977], TWEAK[Chapman, 1987] など多くのプランナが提案されてきた. ここでいうプランニングとは, いくつかの動作と初期状態, 目標状態が与えられたとき, 目標状態を達成する動作系列を求めるものである.

しかし, 各プランナはおもに動作系列生成方法の提案にとどまり, どの程度の問題解決能力を有するかは, 必ずしも明らかではない. 今後, 基礎研究, 応用研究をさらに進める上でも, 各プランナは何ができ, どこに問題があるかを明確にするとともに, 評価の基準を与えることが重要である.

そこで, 本研究では, 各プランナに共通の問題を与え, 定量的評価を行う. 今後, 新たに提案されるプランナとの性能比較ができるよう, 評価対象プランナとして, Public Domain Software(PDS) として入手可能な STRIPS, NONLIN, TWEAK を用いる. 各プランナの評価には, ブロックスワールドの問題を使用する. ブロックスワールドはこれまで多くのプランナで例題として用いられてきた. 今回は各プランナを正答率, 計画の品質, 問題の処理コストという三点から評価する. 計画の品質の議論では, 最適解との比較を行う.

本研究では,

- 今後提案されるプランナの能力評価の土台となる, 評価問題の設定と比較データの提供
- プランナの問題解決能力, 問題点を明確にすること
- ブロックスワールドの性質を明確にすること

を目指す.

ただし, 本稿では ABSTRIPS などの, 計画の抽象化に関しては議論しない.

2 評価対象プランナ

プランニングの目的は動作記述と世界の初期状態と目標状態の記述を与えたとき, 目標状態を実現する動作系列を求めることである.

従来のプランニング研究では, ある副目標を達成しようとするときに, 既に達成済みの別の副目標を破壊してしまうという副目標間の干渉をいかに解決するかが1つの大きなテーマであった. 以下, プランナ STRIPS, NONLIN, TWEAK について説明する.

2.1 STRIPS

(1) 特徴

STRIPS は線形仮説戦略をとる. これは, 目標状態が複数の副目標の連言で表されるとき, 各副目標は独立で, 干渉がないとし, 任意の順序で一つずつ順に解決すれば全体の目標達成に到るという考え方である.

(2) 問題表現

世界の状態はリテラルの連言で表す. 行為者が意図して状態を別の状態に移す行為を動作という.

STRIPS は動作をオペレータで表現する. オペレータは前提条件式, 削除リスト, 追加リストで表現され, それぞれ, オペレータ適用の条件, 適用後に真でなくなるリテラル, 適用後真となるリテラルを表す.

オペレータは変数を含んだ表現が可能である.

オペレータには真偽値の変化するリテラルのみ記述し, 記述のないリテラルの真偽値は変化しないものとする (STRIPS 仮説).

オペレータ適用後の状態は, まず現在の状態記述から削除リスト中の記述を削除し, その後追加リスト中の記述を加えることにより得られる.

(3) 計画生成機構

以下にアルゴリズムを示す. 初期状態表現には変数は含まれないとする.

初期状態を S, 目標状態を G とし, STRIPS(S,G) を呼ぶ.

[STRIPS(S,G)]

1. 状態 S と状態 G の差 D を計算する. ここで, D は状態 G を表すリテラル中で状態 S で真でないものの集合を表す.
2. 差 D が空ならば nil を返し, 終了.
3. 差 D の解消に最も効果的なオペレータ O を選ぶ.
4. 再帰的に STRIPS(S,P) を呼ぶ. ただし, P はオペレータ O の前提条件式.
5. 4 で得た計画 (P を充足する状態に到るための計画) の最後に O を付加したものを新たな計画とする.
6. 状態 S に 5 で得た計画を適用した後の状態 S' を求める.
7. 再帰的に STRIPS(S',G) を呼ぶ.
8. 5 で得た計画の最後に 7 で得た計画を付加したものを新たな計画とする.
9. 計画を返し, 終了.

(4) インプリメントに依存する部分

上記のSTRIPS(S,G)のステップ3での最も効果的なオペレータは次の基準で順に評価する．これは，手段-目的分析(means-ends analysis)の1つである．

1. 達成される未充足の(差D(副目標)中の)リテラル数が最大のもの
2. 破壊される目標中のリテラル数が最小のもの
3. 前提条件式と現状態との差が最小のもの

2.2 NONLIN

(1) 特徴

STRIPSは動作系列を逐次的に表現する．それに対しNONLINは計画を構成する各動作間に半順序関係を許容する．これは本来各動作間に順序関係がないとし，干渉が生じるときに初めて干渉を除去するよう動作間に順序関係を与えるという考えに基づいている．

NONLINの動作表現は，目標を段階的に詳細な副目標に展開するという階層性を有する．

(2) 問題表現

NONLINは副目標の展開法をスキーマ形式で表現する．これは計画の雛型(部分計画)である．

例えば，次の形に記述する．

$$\text{副目標}\ g \Rightarrow (\text{副目標}\ g' \rightarrow \text{動作}\ a)$$

これは，gを$(g' \rightarrow a)$へ置換することを，$(g' \rightarrow a)$はg'を達成してから，aを実行することを意味する．g'は動作aを実行するための前提条件である．

(3) 計画生成機構

NONLINの探索空間では，1つの状態は作成中の計画を表す．各々の計画はグラフ構造で表される．

以下，混乱を避けるため，探索空間上の節点を節点と呼び，探索空間上の1つの節点内の(計画を表す)グラフ上の節点をノードと呼ぶ．前者は作成中の計画を，後者は計画を構成する副目標(1つのリテラル)，あるいは動作を表す．

NONLINでは目標が与えられると，図1に示す形で初期計画を表現する．各ノードをスキーマを用いて実行可能な動作となるまで展開する．

NONLINは，

- ノード展開時にどのスキーマを適用するか
- 計画内に既存のノードを利用して副目標を達成するとき，どのノードからリンクを付けるか
- 干渉を除去するために，どうリンク付けを行うか

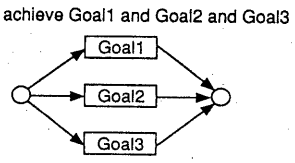

図1: NONLIN Plan Representation

を探索し，それぞれの選択点を後戻り点とする．ノード展開のスキーマがない，またはどのようにしても干渉が除去できないとき，直前の後戻り点に後戻りする．

アルゴリズムを以下に示す．まず，目標をタスクキューに入れる．タスクキューから1ノード(副目標)を取り出し，展開する．ノードを展開したものが新たな後継節点となる．ノード展開により，新ノードが生成されれば，それをタスクキューに入れる．タスクキューが空になるまで繰り返す．

[ノードの展開]

1. ノードnより前か，並行にあるノードの中で，ノードnの副目標を追加するノードの集合をAとする．
2. 副目標を追加するノードが計画内に既存のとき$(A \neq \emptyset)$
   - Aから1つノードを取り出し，そのノードからノードnにリンクを張る(順序関係の導入)．
   - リンク付けの他の候補を後戻り点とする．
3. 計画内に副目標を追加するノードがないとき$(A = \emptyset)$

(a) ノードの展開に有効なスキーマ集合をSとする．

(b) 有効なスキーマが存在するとき$(S \neq \emptyset)$

- Sから1つスキーマを取り出し，そのスキーマを使いノードを部分計画に置き換える．
- 部分計画内の副目標をタスクキューに挿入．
- ノードの展開部分と計画の他部分間との干渉検査を行い，干渉があれば干渉を除去する．
  NONLINはあるリテラルがどのノードで追加され，どのノードで必要となるかという区間を管理する．図2 (a)は干渉が存在する例である．この場合，図2 (b)に示すように順序関係を導入する(2通りの方法がある)．リンク導入によりループが形成される場合はリンクを張らない．
  干渉が除去できないとき後戻りする．
- S中の残りの候補を後戻り点とする．

(c) 有効なスキーマがないとき$(S = \emptyset)$

- 直前の後戻り点に後戻りする．

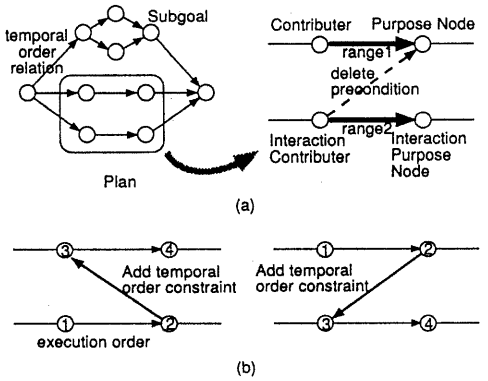

図 2: Interaction Removal

2.3 TWEAK

(1) 特徴

NONLIN は STRIPS とは異なり，計画内の各動作間に半順序関係を許した．TWEAK は最小拘束の観点からさらに変数の具体化に関しても部分的な束縛を許容する．つまり，計画作成途中でオペレータの変数具体化に候補が複数ある場合，制約が十分となって候補が 1 つになるまで，変数のまま置いておく．

時間順序と変数束縛上での部分的関係を許したとき，ある状態において，あるリテラルが真であるか判断する基準として Modal Truth Criteria を提案している．これは簡単には以下のようになる．

> オペレータ O の前提条件 p が必然的に真であるためにはその前に p を追加するオペレータ O_e が必要である．　かつ，p を削除する可能性があるオペレータ O_c(これを clobberer という) が O と O_e の間にあってはならない．

(2) 問題表現

動作の表現法は STRIPS と同一である．TWEAK では初期状態と目標状態を生成する特殊な 2 つのオペレータ I(initial) と G(goal) を用いる．計画は I で始まり，G で終る．

(3) 計画生成機構

TWEAK の探索空間では，NONLIN と同様，1 つの状態は作成中の計画を表す．

TWEAK は計画内の前提条件式が必然的に真でないオペレータを見つけ，それが必然的に真となるようにする．現在の計画の後継状態となる計画の生成法を以下に示す．

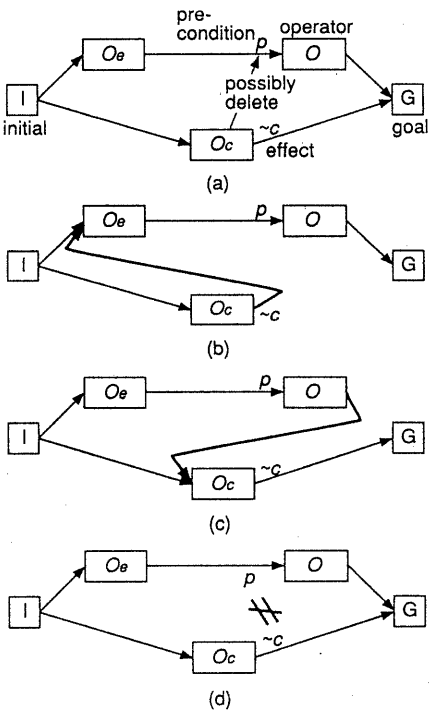

図 3: Declobber Operation

[後継計画生成]

1. 後継計画の集合 $Succ$ を空とする．
2. 現在の計画内で必然的に真でない前提条件式をもつオペレータ O を選び，その必然的に真でないリテラルを p とする．
3. リテラル p を真とする全計画を $Succ$ に挿入．
   リテラル p を真とする方法には 2 種類ある．
   - 計画内に既存のオペレータを利用
   - 新たにオペレータを追加

   各々，必要に応じて，オペレータ実行順序，変数束縛に関して制約を加える．
   リテラル p を真とするオペレータを O_e とする．
4. $Succ$ 中の各後継計画に対し以下の操作を行う．
   リテラル p を削除する可能性があるオペレータ O_c(その削除リスト中のリテラルを c とする) の影響を除去する (図 3 (a) 参照)．
   干渉除去方法には以下の 3 つがある．
   i. $O_c \prec O_e$ という制約を入れる．(図 3 (b))
   ii. $O \prec O_c$ という制約を入れる．(図 3 (c))

iii. $c \not\approx p$ という制約を入れる．(図3 (d))

i. は O_c を実行後に O_e を実行, ii. は O を実行後に O_c を実行するという順序に関する制約である．iii. は変数を具体化するときに c と p が同じにならないようにするという変数束縛に関する制約である．

(4) インプリメントに依存する部分

評価に用いたPDSプログラムでは，探索戦略としてA*用いている．どの探索戦略をとるかは，インプリメントに依存する[1]．

A*で用いる評価関数は次の2項の和である．

- 計画に含まれるオペレータの数(これまでに必要とされたコスト)
- オペレータGの前提条件の中で必然的に真でないリテラルの数(計画完成に必要なコストの推定値)

この推定関数はadmissibleであるので，A*は最適解を与える．

3 ブロックスワールド

3.1 問題の定義

プランナの性能評価にブロックスワールドを取り上げる．ブロックスワールドは多くのプランナの動作表現，アルゴリズムの説明に用いられてきた．しかしながら，これまで性能評価用の問題として厳密には定義されてこなかった．そこで，問題を以下のように定義する．

- 状態空間
  - 世界の状態は述語 BLOCK, ON, CLEAR を用いて表現する．基底リテラルの連言のみを許す．
    BLOCK(X): X は積木である
    CLEAR(X): 積木 X 上に何も載っていない
    ON(X,Y): Y の直接上に積木 X が載っている
    (Y は積木かテーブル)
  - 積木の数は有限とする (状態空間は有限となる)．
  - CLEAR(TABLE) は恒に真であるとする．
- 初期状態

  初期状態は世界の完全な記述を与える．初期状態に曖昧さは存在しないものとする．
- 目標状態

  目標状態は一意に決定できる記述を与える．
  従来，目標状態は1つの塔であることが多く，塔が2つある場合など，その表現は曖昧であり，複数の状態が目標状態とみなせた．一般には，目標状態は一部の積木に関する記述で良い．しかし，後の評価問題としての使用のため，ここでは，目標状態は一意に決定できるものとする．
- プリミティブ動作
  - プリミティブ動作は“1つの積木を持ち上げ，テーブル，または他の積木の上に置く”だけを与える．
  - CLEAR(X) が真である積木 X のみ CLEAR(Y) が真である積木 Y 上 ($X \neq Y$) あるいはテーブル上に動かせる．

問題は上記の下で，初期状態から目標状態を達成するプリミティブ動作の系列を求めることである．

3.2 ブロックスワールドの性質

ブロックスワールドの性質について Gupta and Nau[1991] の議論がある．

まず，最終の位置と deadlock 状態の定義を述べる．

最終の位置： 状態Sで積木bの下にあるすべての積木(テーブルも含める) が目標状態Gにおいても，正確に同じ順序でまた積木bの下にあるならば，状態Sで積木bは最終の位置にあるという．

最終の位置にある積木はそれ以上動かす必要がない．

deadlock 状態： 状態Sにおいて最終の位置にない積木の集合 $\{b_1, b_2, ..., b_p\}$ は以下の2条件を満たす積木の集合 $\{d_1, d_2, ..., d_p\}$ が存在するならば deadlock 状態にあるという．

1. 状態Sで，b_1 が d_1 の上方にあり，かつ b_2 が d_2 の上方にあり,..., かつ b_p が d_p の上方にある．
2. 目標状態で，b_1 が d_2 の上方にあり，かつ b_2 が d_3 の上方にあり,..., かつ b_p が d_1 の上方にある．

Guptaらは最短の計画を見つけることはNP困難であることを示している．主張のポイントは以下の通りである．

1. 目標達成自体は簡単である．必要であればテーブルの上にばらし，下から積み上げるという簡易解法がある．簡易解法では，最悪の場合でも，2n-2回の動作で目標を達成できる．
2. ブロックスワールドの複雑さは，ある動作が他の動作の前提条件を削除するという動作間の干渉によるのではなく，deadlock の存在による．deadlock 状態で deadlock 状態にある積木を1つテーブルの上に置けば，少なくとも1つの deadlock が解消される．場合により，1つの積木を動かすことで複数の deadlock が解消できる．このすべての deadlock を解消する，最小の動作系列を求めることは組合せ問題となる．

[1] Chapman は dependency-directed 横型探索を用いている．

4 評価方法

4.1 評価の方針

- 問題生成法

  初期状態と目標状態をランダムに生成する．まず，最初にn個の積木とm個のテーブル上の位置を用意する．n個の積木中からランダムに1つ取り出し，m個の位置からランダムに選んだ位置に積み上げる．全ての積木を置くまでこれを繰り返す．ただし，計画生成時にはテーブル上の位置は無限とする．

  積木の数nは3個から20個まで変化させる．テーブル上の位置の数mは1から3まで乱数を発生させ決定する．

  各積木の数に対し，10問ずつ用意する．

- 実行方法

  実験はSUN/SPARCstation10上でAllegro Common Lispを用いて行う．計算時間に制限を設け，計算時間が3600[s]を越えると，解なしとした．生成節点数には制限を設けない．

- 評価項目

  正答率，解の品質，処理コストの3項目について評価する．

4.2 最適解の計算

解の品質の評価のため，最適解を求める必要がある．ここではIDA*[Korf, 1985]を用いる．IDA*は探索の一手法であり，プランナではない．

IDA*はA*を改良したものである．IDA*の特徴を以下に示す．

- IDA*は一連の深さ優先探索を繰り返し実行する．深さ限界を，初期状態の推定関数値(目標達成に必要な動作数の推定値)から始め，解が求まるまで，1ずつ増加させる．
- 状態の推定関数がadmissibleであればA*同様最適解が求まる．
- 必要記憶量が，解の長さに対し，A*では指数形となるが，IDA*は線形である．

ここで，IDA*で最適解を求める際に用いる，我々が考案した高精度の推定関数を図4に示す．探索では用いる推定関数の精度により，どの程度の規模の問題が解けるかが決まる．ここでは，最終の位置にない積木数$h1$と要素が1つのdeadlock数$h2$の和を計算する．

図5にこの推定関数を用いた計算例を示す．(a)は与えられた初期状態と目標状態である．(b)には$h1$の計算が，(c)には$h2$の計算が示されている．この例

$h1 \leftarrow$ 最終の位置にない積木の数
$h2 \leftarrow 0$
$B \leftarrow$ 最終の位置にない積木の集合
for each block b in B, **do**
  $D_1 \leftarrow$ 現状態において積木bの下にある積木の集合
  $D_2 \leftarrow$ 目標状態において積木bの下にある積木の集合
  loop
    if D_1 is empty, **then** exit **loop**
      let $d_1 = Pop(D_1)$
      if $d_1 \in D_2$, **then** $h2 \leftarrow h2 + 1$, exit **loop**
      endif
    endif
  endloop
endfor
return $h1 + h2$

図4: Heuristic Evaluation Function

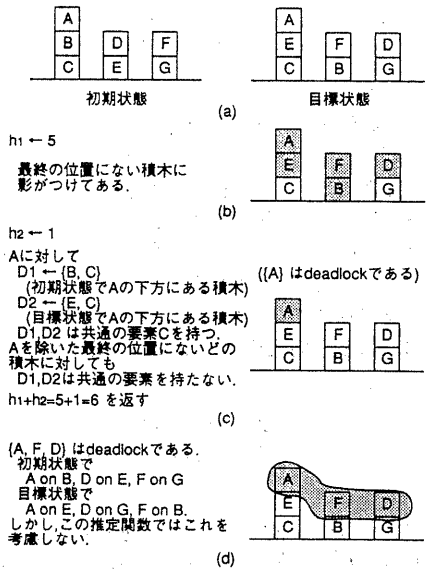

図5: Example of Heuristic Evaluation Function

表 1: Solved Problem

blocks	STRIPS	NONLIN	TWEAK	IDA*
3	10	10	10	10
4	10	9	10	10
5	10	9	6	10
6	10	9	7	10
7	10	9	2	10
8	10	7	0	10
9	10	8	0	10
10	10	10	0	10
11	9	8	0	10
12	9	9	0	10
13	8	9	0	9
14	10	7	0	9
15	9	5	0	8
16	6	9	0	7
17	9	8	0	9
18	4	4	0	8
19	3	3	0	9
20	1	4	0	6

では，推定値は 6 となり，目標達成に必要な実際の最小動作数 6 に一致する．(d) にある要素数が 2 以上の deadlock は計算しない．

deadlock を全て求めること自体，時間を要し，また，その中からどのような順序で deadlock を解消すれば動作系列が短くなるか計算することも時間を要する．よって，簡単に求めることができる，要素が 1 つの deadlock のみを求める．

5 性能測定

5.1 正答率

各積木数の問題に対し，正しい計画を得た数を表 1 に示す (各積木数につき全 10 問)．

- 積木の数が，STRIPS, NONLIN, IDA* については 15 を越えるあたりから，TWEAK については 5 を越えるあたりから正答率が落ちる．

  これらは，NONLIN を除いて，計算時間が制限時間を越えたことによる．

  TWEAK は (1) 干渉除去を計算するため，1 つの節点生成に長い時間を要し，(2) 使用される推定関数の精度が低く，多くの節点を生成するため，他に比べ積木数が少なくても正解が求まらない．

- NONLIN は少ない積木数でも正解が得られない場合がある．

  副目標の達成には，副目標を達成する動作を前に置く必要がある． その方法には，本来 (1) 作成中の計画内に既存の動作を利用 (リンク付け)，(2) 動作の新規挿入 (スキーマによる展開) がある．PDS の NONLIN は前者が可能であるときには，後者を試みない．そのため，正解が得られない場合が生じている．

  前者が成立する場合に後者を後戻り点とするかどうかについては [Tate, 1977] に記載がない．これはインプリメントの仕方による．

- IDA* は積木数が増えても他に比べ正答率が高い．

  推定関数の工夫により，無駄な節点生成が抑制されることによる．

5.2 解の品質

解の品質は計画実行時間を用いて評価する．計画実行時間は，動作系列の長さ，即ち，計画に含まれるプリミティブ動作の数とする．

各積木数につき正解を得た結果の平均をとった動作系列の長さを図 6 に示す．(b) は (a) の一部を拡大してある． 2n-2 の線は簡易解法で解いた場合の最長の動作系列の長さを示す．

- STRIPS は他に比べて極端に動作系列が長い．

  STRIPS は目標の一部しか見ないため，ある副目標を達成しようとして，既に達成済みの副目標を破壊する．また，STRIPS は目標を達成する方法として，オペレータの新規挿入しかない．このため，動作系列は非常に長くなる．

  積木数が 18, 20 で動作系列長さが大きく減少しているが，これは，動作系列の長い問題は解けないため，グラフ上に表れないためである．

- NONLIN は最適解と簡易解法による最悪の場合の解の間にある．

  NONLIN は積木をテーブルの上にばらす傾向があり，簡易解法に近い方法で目標達成を行うためである．

- NONLIN, IDA*, TWEAK は差が小さい．また，簡易解法で得られる最悪の場合の動作系列長さと比べてもほとんど差がない．

  ブロックスワールドの問題では，最適解と簡易解法による解との間の差が小さい． これは，積木は下から積み上げるしかなく，複数の積木を一度に動かすなどの目標達成の近道が存在しないためである．

5.3 処理コスト

処理コストは計画生成時間 (生成節点数と計算時間) を用いて評価する．

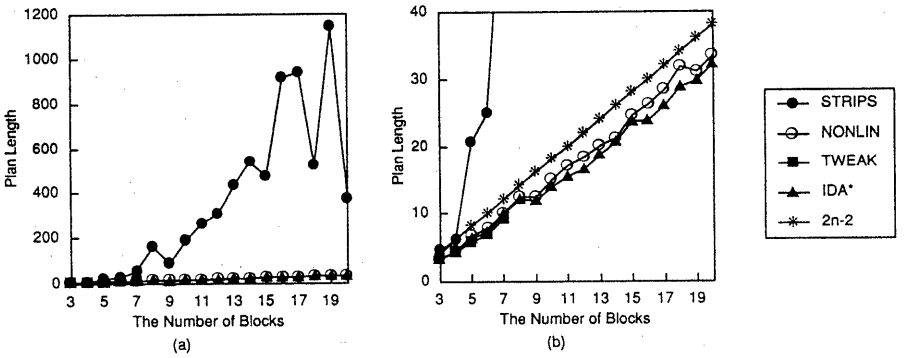

図 6: Plan Length

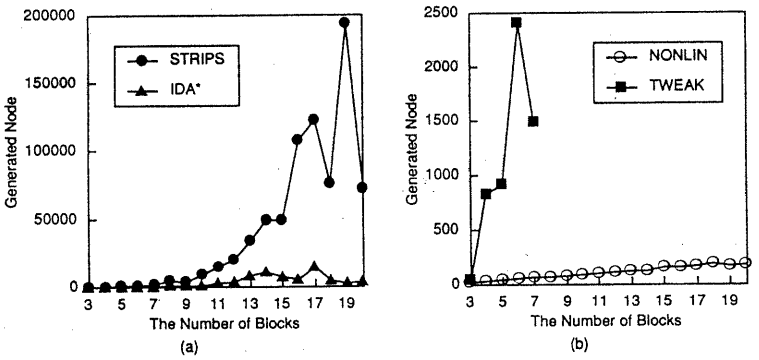

図 7: Generated Node

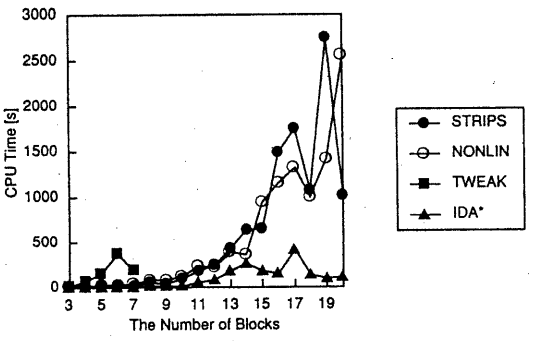

図 8: CPU Time

(1) 生成節点数

各積木数について正解を得た結果の平均生成節点数を図7に示す．図7 (a) では1つの節点は積木の世界の状態を，(b) では1つの節点は作成中の計画を表す．

- STRIPS は積木数が増えるに従い，グラフの傾きが大きくなる．

  積木数が増えれば副目標の数が増える．また，副目標間に干渉が存在するとき，副目標の達成，破壊の繰り返しを含んだ副目標の達成順序の系列は，副目標数に応じて組合せ的に長くなり得る．つまり，積木数が増えるほど，干渉は強くなる．よって，有効なオペレータの選択基準がないとき，目標達成に要する生成節点数は指数関数的に増加する．

- NONLIN の生成節点数は積木数に比例する．TWEAK は NONLIN に対し傾きが非常に大きい．

  探索の制御戦略として，NONLIN は深さ優先探索，TWEAK は A* を用いている．また，NONLIN はテーブルに積木をばらす傾向にあり，簡易解法に近い解法となり，解の長さが積木数に比例する．そのため，生成節点数が，NONLIN では積木数に対し比例するが，TWEAK では指数関数的に増える．

(2) 計算時間

各積木数につき正解を得た結果の平均をとった計算時間を図8に示す．

- NONLIN は積木数が増えるに従い，グラフの傾きが大きくなる．

  生成節点数は線形に増える．即ち，積木数が増えるに従い，1つの節点生成に要する時間が増える．これは，作成中の計画が大きくなり，1つの節点生成時に，リンク付けの影響を調べるのに時間を要するためである．

- 計算時間では，STRIPS と NONLIN にそれほど差はない．

  積木数が少ないときは，双方簡単に問題が解け，差がつかない．積木数が増えると，STRIPS では動作系列の長さが長くなり，NONLIN では1つの節点生成に要する時間が長くなる．この増え方が同様である．

6 評価

6.1 各種プランナの評価

(1)STRIPS

- 線形戦略仮説が成立しない場合，つまり副目標間に強い干渉が存在する場合には生成された計画の品質は非常に悪い

  例えば，[Fikes and Nillson, 1971] に例題としてあるロボットが箱をある部屋から別の部屋へ運ぶ問題など，副目標間の干渉が少ない問題には STRIPS は有効である．しかし，ブロックスワールドの問題は非常に副目標間の干渉が強い．そのため副目標を一旦達成しては破壊することを繰り返し，非常に効率が悪くなる．

- 手段 - 目的分析は十分なオペレータ選択基準となり得ない

  目標状態と初期状態の差を求めたとき，本来差の中にどれを先に解決すべきかという優先順序が存在する．しかし，手段 - 目的分析はこれが評価できない．下から積むという優先順序を基にオペレータ選択を行えば，品質面での性能は向上する．だが，ブロックスワールド以外の問題では一般に副目標の順序付けは難しい．

(2)NONLIN

- 問題が大規模化すれば，干渉除去の負荷が大きくなり，処理コストは指数関数的な増加を示す．

- 領域知識の使い方が implicit である．

  積木を CLEAR 状態にするとき，上に載っている積木を (1) 他の積木の上に動かす方法と (2) テーブル上に動かす方法がある．テーブルの大きさに制限がないときは，後者を選択する方が残された問題は簡単になる．NONLIN は後者を優先するが，これは，プランナ内部，または，実際に生成された計画を調べないと，判断できない．問題記述者はプランナ内部まで知らないと，どのような動作が選択されるか判断できず，NONLIN は扱いにくいシステムとなっている．

(3)TWEAK

- 探索制御が困難である．

  A* では問題規模に対し必要記憶量が指数関数的に増え問題は解けない．副目標達成，干渉除去に関して複数の方法が存在するが，作成中の計画の評価が難しく，探索方向の判断が難しい．

- 各干渉除去法は問題により有効性が変わる．

  NONLIN にはない，干渉除去の方法 (例えば，家のペンキ塗りで，壁を塗る動作がブラシ $b1$ を必要とし，天井を塗る動作もブラシ $b1$ を必要として干渉が生じるとき，壁を塗る動作がブラシ $b2$ を使うことにより，干渉を除去する方法 (図3 (d))) がある．しかし，ブロックスワー

ルドではこの干渉除去の方法が事態を大きく好転させることはない．この操作は探索における分岐を増やすだけである．

- 変数の部分的束縛許容の有効性は判断できない．

  TWEAKは計算途中で変数を，一意に具体化できなければ未束縛のまま置いておく．これは，変数の具体化候補が複数あるとき，ad hocに1つに決めて，後で駄目ならば後戻りすることを避けるためである．これは変数の具体化候補が多いときに効果を発揮する．しかし，探索の制御戦略が悪いため，変数の具体化候補(積木数)が多い問題が解けず，有効性が確認できない．

6.2 ブロックスワールドの評価

- 品質面から見れば，最適解と簡易解法による解にそれほど差がない．

  最適解を求めることはNP困難であり，簡易解法では$O(n)$で解が求まる．一般の問題では，最適解を得るのは困難であり，準最適解をどう求めるかが問題となる．しかし，ブロックスワールドでは簡易解法による解を準最適とみなしてよい．よって，計画の品質面からプランナを評価するとき，ブロックスワールドは評価問題として適切ではない．

7 むすび

本論文ではプランナSTRIPS, NONLIN, TWEAKにブロックスワールドの問題を与え定量的な評価を行なった．まず，評価問題として使うため，目標状態が一意に決まるよう問題を定義した．次に，評価を行い，以下のことを明らかにした．

- 線形プランナSTRIPSでは，副目標間の干渉の強い問題に対して，計画の品質，処理コスト共に指数関数的に悪化する．
- 非線形プランナNONLINは，計画の品質ではSTRIPSより優れた性能を示す．しかし，問題が大規模になると，干渉除去の負荷が大きくなり，処理コストが指数関数的に増大する．
- 非線形最小拘束プランナTWEAKは，NONLINにない副目標間の干渉除去法を有するが，どのような場合にどの方法を用いればよいか不明である．そのため，干渉除去法の豊富さが逆効果となって，探索空間上で分岐を増やす結果となり，十分に問題を解くことができない．
- ブロックスワールド問題は簡易解法による解と最適解との間にほとんど差がなく，プランナの計画の品質の評価用問題として適切とはいえない．

ブロックスワールドはあくまで1つの例題であるが，IDA*は問題に依存した推定関数を用いることにより，プランナよりも優れた性能を示した．問題に特有のヒューリスティックスをどう組み込むことができるかという点で，問題の記述力という面からの評価も必要である．

謝辞

IDM(STRIPS-like planner)の使用を許可して頂いたNick Short, Leonard Dickens (NASA/GFSC), NONLINの使用を許可して頂いたSubrata Ghosh, James Hendler, Subbarao Kambhampati, Brian Kettler(Computer Science Department, University of Maryland), AbTweakの使用を許可して頂いたQiang Yang, Steve Woods (Computer Science Department, University of Waterloo)の各氏に深謝します．

また，本研究を支援して頂いたNTTコミュニケーション科学研究所西川清史前所長，中野良平主幹研究員に感謝します．

参考文献

[Chapman, 1987] D. Chapman, "Planning for Conjunctive Goals," *Artificial Intelligence*, 32, pp.333-377, 1987.

[Fikes and Nilsson, 1971] R. E. Fikes, N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, 2, pp.189-208, 1971.

[Gupta and Nau, 1991] N. Gupta, D. S. Nau, "Complexity Results for Blocks-World Planning," *AAAI-91*, pp.629-633, 1991.

[Korf, 1985] R. E. Korf, "Depth-First Iterative-Deepening: An Optimal Admissible Tree Search," *Artificial Intelligence*, 27, pp.97-109, 1985.

[Newell and Simon, 1963] A. Newell and H. A. Simon, "GPS, A Program that Simulates Human Thought," *Computers and Thought*, ed. E. A. Feigenbaum, J. Feldman, pp.297-293, 1963.

[Nilsson, 1980] N. J. Nilsson, "Principles of Artificial Intelligence," Tioga Publishing Co., 1980. (邦訳: 白井良明, 辻井潤一, 佐藤泰介訳, "人工知能の原理," 日本コンピュータ協会, 1983.)

[Sacerdoti, 1973] E. D. Sacerdoti, "Planning in a Hierarchy of Abstraction Space," *IJCAI-73*, pp.412-422, 1973.

[Sacerdoti, 1975] E. D. Sacerdoti, "The Nonlinear Nature of Plans," *IJCAI-75*, pp.206-214, 1975.

[Tate, 1977] A. Tate, "Generating Project Networks," *IJCAI-77*, pp.888-893, 1977.