7A−3　バックトラック無しアルゴリズムの実験評価

李江洪　山下正吾　内野寛治　窪田信一郎　西原清一
筑波大学　電子・情報工学系

1.はじめに

制約充足問題(CSP)[1]は問題の対象の構成要素の局所的な構造に基づいて可能な局所解の候補を求め，それらの中から対象全体について矛盾のない局所解の組合せを求める問題である．CSPを解くとは，制約を充足するような局所解すなわち変数への値の割り付けの組合せを探索していく処理である．この処理はバックトラックを基本とした縦型探索が主流である．これは，矛盾が起こらない限り中間解を拡張していく方法であり，矛盾が生じると探索木を遡り,別の局所解の候補を調べる．したがって，もし探索の過程が常に最終解に至る探索路に乗っていることが保証されるならば,バックトラックは発生しないはずである．このような探索を，'バックトラック無し'という[2]．我々は先に，バックトラックによって探索木を遡る範囲がある一定サイズk(「幅」という,後述)以内に限られるようなCSPの部分クラスに注目して，バックトラック無しアルゴリズムを提案した[3]．本稿はバックトラック無しアルゴリズムの計算時間量について実験で評価したものである．

2.バックトラック無しアルゴリズム

CSPは四つ組(U,L,T,R)で定義される．(具体的な定義は文献[4]に譲る)．CSPは$t_i \in T$の次元によって二項制約と多項制約に分類される．任意の多項制約は等価な二項制約に変換できる．

二項制約において，'順序付きCSP'とは次のような条件を満たすものを言う：

1.変数集合Uに全順序'$\prec$'が定義されていること．

2.任意の変数組$t(\in T)=(u,v)$において，$u \prec v$.

3.変数制約関係Tの要素は，$\prec$を用いて辞書式順序に並んでいること.すなわち，任意の$t_i=(u_i,v_i)$,$t_j=(u_j,v_j)$において$i<j$であれば，$u_i \prec u_j$,または,$u_i=u_j$が常に成り立つこと．

CSPを解くとき，・一つの解を求める，・全ての解を求める，という2つの場合があり，計算複雑さはいずれの設問であるかによって異なってくる．

我々が提出したバックトラック無しアルゴリズムは二項制約をもつ順序付きCSPの全解を求める設問に基づくものとする．

現実のCSPにおいて変数が関与している制約には局所性が存在する．すなわち，一つの変数はCSPの変数の一部としか制約関係を持たないということである．探索の過程で，ある変数に関する制約が出尽くしてしまうと，その変数のレベルまでバックトラックが遡ることはない．我々の方法は，この性質を利用して,バックトラックを避けつつ与CSPの制約を順次チェックし，制約グラフ(制約木という,後述)を生成するものである．

手続きを図1に示す．この手続きCSは,順序付きCSPを扱う．制約木(constraint tree)とは，根以外のレベルに変数を順序にしたがって対応させ，処理途中における変数への合法的値割り付けの状況を表現するものである.図1に示すように，手続きCSはTの要素t_i(および対応するR_i)を順に処理してゆくことによって進行する．重要な操作は，update(u)，merge(u)，check/generate(u,v)の三つである．与CSPは順序付きであるので，Tの要素を順次処理していくと$t_i=(u,v)$の成分uがいつかは出尽くすときがくる．これを判定するのが,4行目のif文である.変数uが出尽くすと，uに割り当てられた値は，update(u)によって中間解として制約木の葉ノードの中間解木に保存(更新)される．そして，merge(u)によって，uを根とするサブ制約木は整理統合(併合)され，整理統合されたサブ制約木が制約木から削除される,そして，制約木からuのノードが削除され，制約木の

```
INPUT :CSP
OUTPUT:CSPの全ての解を含む解木
Algorithm CS
 begin
 1  u' = 1;
 2  for i=1 to |T| do
 3  {   (u,v) <- t_i;
 4      if (u'≠u)
 5      {  for w=u' to u do
 6         {  update(w);    /*中間解の更新*/
 7            merge(w);     /*制約木の併合*/
 8         }
 9         u'=u;
10      }
11      check/generate(u,v);
12  };
13  for w=u' to |U| do   /*制約木の縮約*/
14  {  update(w);    /*中間解の更新*/
15     merge(w);     /*制約木の併合*/
16  }
 end.
```

図.1: バックトラック無しアルゴリズム CS

Experimental Evaluation On Backtrack − free Algorithm
Jiang − Hong LI, Shogo YAMASHITA, Kanji UCHINO
Shinichirou KUBOTA, Seiichi NISHIHARA
Inst. Inf. Sci. & Electr., University of TSUKUBA

高さが1減る．check/generate(u, v)は，変数uとvを要素として含む新たな変数組t_iの処理を行う．すなわち，tiに対する値制約関係R_iと照合して，現在の制約木の適否をチェックする．もし制約を満たさなかったらその満たさない枝とその枝に付着している中間解木が削除される．ただし，uまたはvが初めて登場した変数である場合は，制約木に新たにレベルuまたはvを追加し木を増大させる．変数が制約木の葉に追加される場合,中間解木が新しい葉ノードに伝播される．Tの最初の要素t_1の第1成分は常に変数1であるから，初期化操作はu' = 1によって処理が始まる．このようにして，制約を順次処理していく過程で，常に不必要な中間解を削除することによって，調べるべき値組の個数すなわち探索空間を抑制する．（注：アルゴリズムCSの実行例は[3]を参照のこと）．

3.実験と評価

このアルゴリズムの計算複雑さの実験を説明するために,「幅」と言う概念を定義する[3]．

[定義1]‘変数uの幅（width)’とは，手続きCSにおいてcheck／generate(u, v)によって生成したuを根とする部分制約木においてu以外の変数の数をいう．‘変数順序列の幅’とは，与CSPに対して手続きCSは変数順序列の順序で変数の制約を充足していく過程で,変数の幅の最大値をいう．‘CSPの幅’とは,与CSPの全ての可能な変数順序列の幅のうち最小のものをいう．

アルゴリズムCSの計算複雑さは次の定理で証明される[3]．

[定理1] 手続きCSによって幅がkであるCSPの全解を求めるのに要する計算時間量は$O(nd^{w+1})$である．ここで,nは変数の個数で，dは変数の可能な値の個数の最大値である．

定理1により，CSPの幅がCSPの制約充足時間を左右する.しかし，この定理はただ幅がkであるCSPの全解を求める計算時間量の上限を定義しただけである．したがって，現実の問題に対して，このアルゴリズムの効果を調査する必要がある.ここで，我々は以下の四種類の変数順序についてその計算時間量を実験で評価した．

a.最適順： 与CSPの全ての可能な順序列に対して最小幅を計算し，その最小幅の順序で変数の制約を充足する．

b.幅昇べき順： 与CSPの最小幅を計算し，最初に幅が一番小さい変数から制約を充足し，そして，残りの変数から幅が一番小さい変数に関する制約を充足する．このように全ての変数に関する制約を充足していく．いつも順序列の幅を極小にする．

c.度数昇べき順： 与CSPの最小幅を計算するのではなく，最初に度数（変数uの度数とはuと制約関係を持つ変数の個数のことを言う）が一番小さい変数から制約を充足し，残りの変数から度数が一番小さい変数に関する制約を充足する．このように全ての変数に関する制約を充足していく．常に順序列の度数を極小にする．

d.ランダム順： ランダムで変数を選び，その変数に関する制約を充足し，全ての変数が選ばれたら，終了する．

この実験はCSPを200個ランダムで発生させ，その計算時間を評価したものである．また，本稿の実験の目的は，主にCSPの制約グラフ構造が制約充足時間にどのように影響するかを調べることであるので，全ての値間制約の個数を同じにした．

実験結果を表1に示す．表中の数値の上段はそれぞれの対比的計算時間に属したCSPの個数で，下段は全体に占めるそれらのCSPの割合である．また評価の基準は制約木をアクセスするコスト（制約木を一回アクセスするコストは制約木にあるノードの個数と等しい）である．実験結果により，幅昇べき順と度数昇べき順はランダム順より効率がよいが，最適順より効率が悪いことがわかった.また，幅昇べき順と度数昇べき順の効率は平均的にほぼ同じで，ランダム順の効率が一番悪かった．

対比的計算時間 / 変数順序 ＜比較の対比となる変数順序＞	速い	同じ	遅い
a＜b,c,d＞	135／67.7	16／8.2	49／24.1
b＜c＞	69／33.8	63／31.3	63／32.3
d＜a,b,c＞	13／2.7	55／27.7	132／65.6

表1： バックトラック無しアルゴリズムの効率評価

4.おわりに

本稿はバックトラック無しアルゴリズムの計算時間量について考察した．実験結果により，最適順は一番効率がよいが，最小幅を持つ変数順序列の計算時間は平均的に$O(n!)$である．従って，最小幅を持つ変数順序列の近似計算法の開発が今後の課題として考えられる．また，制約充足に適用する変数順序についてCSPを分類することも今後の課題である．

5.参考文献

[1]Haralick, R.M. etc: The Consistent Labeling Problem, Part I, IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-1, No. 2 (1979), 173-184.

[2]Freuder, E.C.: A Sufficient condition for back track-free search, Comm. ACM 21(1982)958-966.

[3]李，西原： 多項式時間で計算可能な制約充足問題について．第5回人工知能大会(1991)(12-1)．

[4]窪田，内野，李，山下，西原：併合法による制約充足の並列化効果について，本論文集7A－1（1993）．