高速化の知識を取り入れた制約充足問題の一般解法

6P－8　山田弘展　内野寛治　狩野均　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（以下CSP）とは、複数の構成要素間に関わる局所的制約が与えられたときに、対象物全体の矛盾のない解釈を探索によって求める問題である。著者らは、このパラダイムにおける、制約の表現方法および、探索の高速化のための知識の構成方法を検討している。

本稿では、従来の列挙型の表現のほかに、手続き型で表現されたCSPを高速に解くアルゴリズムを提案する。また、本手法を通常の木探索と比較検討した結果も示す。

2. 本手法の概要

2.1 制約充足問題

CSPは４つ組(U,L,T,R)で定義される。U={1,..,M}はユニット(変数)の集合、Lはラベル(値)の集合を表す。制約$T=\{t_1,...,t_N\}$は、その成分であるユニット同士が局所的な制約条件の下にあることを示している。そして、実際にどのような局所解が許されるかは、$R=\{R_1,...,R_N\}$で具体的に与えられる。CSPを解くとは全ユニットに対し、全ての制約条件を満たす解を求める操作である。

文献[1]では、上記(U,L,T,R)の形式で表されたCSPを木探索法で解く方法が提案されている。さらに高速化の技法として、先読みオペレータを用いて、ユニットの探索順序を決定することにより、探索空間の縮小化を図っている。

2.2 基本方針

文献[1]の木探索法では、局所解を列挙型の記述で表記している。このため、局所解の数が多くなるような制約では、全ての局所解を表記しきれない場合がある。

またCSPに、ラベルが２種類のみである（二分木）場合、解に偏りがある場合には、上記オペレータを用いた探索では、高速化はあまり期待できない。

そこで以下の２点を改良した。

(1) 制約を手続き型の記述で表現できるようにした。例えば、制約と局所解が列挙型で

(1 2 3 4 5){(真 真 偽 偽 偽)(真 偽 真 偽 偽)
...(真 偽 真 真 真)(偽 真 真 真 真)}

と表現されている場合、手続き型では

(1 2 3 4 5)('真 even2)

すなわち「'真'の数が偶数個である」と簡潔に表現できる[2]。

手続き型の記述で表現された制約は、その制約が参照されるときにのみ、既にラベルが割り当てられたユニットを参照して列挙型の表現に展開されるため、不必要な局所解を作らず、候補解集合を縮小できるという特徴がある。

(2) ユニットの探索順序を、各制約に含まれるユニットの数と局所解の数により、探索前に決定することとした。決定した探索順序は、ユニットのリスト（探索順序リスト）として探索部に入力される。

これにより、文献[1]の方法では高速化が期待できなかった二分木探索においても高速化が期待できる。

3. 提案する手法

3.1 処理手順

提案する手法の処理手順を図１に示す。図１中の「探索順序決定」と「制約の参照と展開」について次節で説明する。

3.2 制約の参照と展開

手続き型の制約を列挙型に展開する手順は、参照する制約に探索済みユニットが含まれる場合は、探索済みユニットの値を考慮し手続きを展開し、そうでない場合は、通常の手続きとして展開するものである。

例えば、2.2節で挙げた制約

(1 2 3 4 5)(labels '真 even2)

を展開すると次のようになる。

Efficient Solving Method for Constraint Satisfaction Problem
Hironobu Yamada, Kanji Uchino, Hitoshi Kanoh, Seiichi Nishihara
Institute of Information Sciences and Electronics, University of Tsukuba

ユニット1,2,3は探索済みとし、それぞれにラベル'真'が割り当てられているとすると、残りのユニット4,5は、'真'の合計は偶数個であることから、'真 偽'または'偽 真'となり、得られる局所解は以下の２つである。

(1 2 3 4 5){(真 真 真 真 偽)(真 真 真 偽 真)}

3.3 探索順序の決定

探索順序は、各制約に含まれるユニットの数と、手続きを展開してできる局所解の数をパラメータとする評価関数により決定する。また、いくつかのユニットに対し、一意にラベルを決定してしまうような制約に含まれるユニットは、優先的に探索するようにした。

step1：一つの制約でラベルが決定するユニットを探索順序リストに加える。

step2：探索順序リストにあるユニットを探索すると一意にラベルが決定するユニットを探索順序リストに加える。

step3：探索順序リストに無いユニットのうちで評価関数 h (i)

$h(i) = a\times(1000 - |R_i|) + b\times(100 - |t_i|)$

の値の大きい制約に含まれるユニットから探索順序リストに加える。

ただしiは制約番号、a,bは定数、$|R_i|$は局所解の数、$|t_i|$は制約中のユニットの数とする。

4. 実験と考察

実験結果を表１に示す。

表１から、問題のサイズ｜U｜、ならびに問題の複雑さ｜T｜,｜R｜が大きくなるに従い、本手法が有効に機能していることがわかる。

またメモリのサイズも、従来の手法に比べて少なくなっている。

5. おわりに

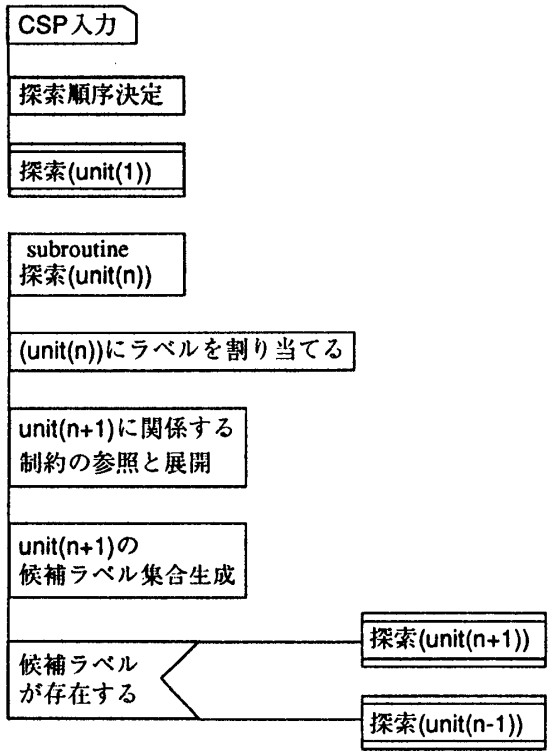
本稿では、制約を手続き型で表記し高速に解くアルゴリズムを提案した。また従来の木探索法と比較し、その有効性を示した。

今後は、三面図理解問題や時間割編成問題などの、CSPのアプリケーションに本手法を適用し、その有効性を考察する予定である。

参考文献

[1] R.M,Haralick,Shapiro,L.G :The Constraint Labeling Problem : Part I , IEEE Tr.PAMI,PAMI-1,2(1979), 173-184.

[2] 若林,内野,狩野,西原：制約充足に基づく三面図理解システム,第49回情処大会5C-2(1995).



unit(n)：「探索順序決定」で作られた探索順序を示すリストのn番目のユニット

図１ 処理手順

表１ 実行結果

｜U｜	｜T｜	｜R｜	従来の手法 時間	従来の手法 メモリ	本手法 時間	本手法 メモリ	解 (個)
20	62	338	50	404	33	238	1
24	54	179	17	365	33	208	1
33	69	284	1267	879	633	272	35
38	131	545	250	998	100	269	1
55	143	466	1570	1457	117	262	1
60	126	708	2066	3825	183	280	1
70	227	1908	—	5570+ α	517	329	1

時間：全解探索にかかった時間（単位：$\times 10^{-3}$秒）

メモリ：変数に使用したメモリのサイズ（単位：kbyte）

'—'はメモリオーバーのため探索が中断したケース