Vol. 35 No. 8 情報処理学会論文誌 Aug. 1994

弱コミットメント戦略を用いた制約充足問題の解法

横尾 真†

制約充足問題の解を探索するバックトラッキングアルゴリズムにおいては，探索途中で構成された部分解は，解の一部分とならないことが判明しない限り修正されることはない．このため，探索中に解になりえないような部分解を構成した場合，その部分解を含むすべての解に関する網羅的な探索が必要となり，探索の効率は著しく低下する．そのような部分解を構成しないための，変数の値の選択のヒューリスティックとして制約違反最少化ヒューリスティックが提案されているが，解になりえない部分解を構成することを完全に避けることは一般に不可能である．本論文では，部分解に弱くコミットする，制約違反最少化ヒューリスティックを用いた新しい探索アルゴリズムを提案する．この方法によれば，部分解が成長している間は，その部分解を変更しないが，ある変数に関して，部分解と制約を満たす値が存在しない場合には，それまで構成された部分解を放棄し，改めて部分解の構成を最初からやり直すことにより，解になりえない部分解に関する網羅的な探索を回避する．確率的モデルおよび実験結果を用いて，バックトラッキングアルゴリズムに対する本アルゴリズムの優越性を示す．さらに，実験結果を用いて，近年，バックトラッキングアルゴリズムより効率的であることが示された，状態空間探索型のアルゴリズムに対する本アルゴリズムの優越性を示す．

Solving Constraint Satisfaction Problem with Weak-commitment Strategy

Makoto Yokoo†

Backtracking algorithms are widely used for solving constraint satisfaction problems. In these algorithms, a partial solution constructed during search will never be changed unless it is proved that no solution can be derived from that partial solution. Therefore, by constructing a partial solution that can not be a part of any final solution, the efficiency of the algorithms will be significantly reduced. Although some heuristics, such as *min-conflict* heuristic, for avoiding bad partial solutions are proposed, bad partial solutions can not be avoided completely. In this paper, we develop a new search algorithm that commits to partial solutions *weakly*, i.e., this algorithm commits to a partial solution as long as the partial solution can be extended, but it abandons the partial solution if no consistent value exists for the partial solution, and re-start constructing a new partial solution. By using a probabilistic model and experimental results, we show that this new algorithm is more efficient than backtracking algorithms. Furthermore, by using experimental results, we show that this algorithm is more efficient than state-space search algorithms, which are recently shown to be more efficient than backtracking algorithms.

1. はじめに

制約充足問題 (Constraint Satisfaction Problem, CSP)[3] は，人工知能におけるさまざまな問題を定式化できる一般的な問題であり，制約充足問題を解く一般的な解法として，縦型の探索アルゴリズムであるバックトラッキングアルゴリズムが広く用いられている．バックトラッキングアルゴリズムでは，探索途中で構成された部分解は，解の一部分とならないことが判明しない限り修正されない．このため，探索中に解になりえないような部分解を構成した場合，その部分解を含むすべての解に関する網羅的な探索が必要となる．目的が全解探索ではなく，任意の一つの解を得れば良い場合，このような網羅的な探索を行うことは望ましくない．

近年文献 4) によって，制約違反最少化ヒューリスティック (*min-conflict* heuristics) と呼ばれる，解になりえないような部分解を構成することを避けるための，値の順序付けのヒューリスティックが提案された．文献 4) では，このヒューリスティックをバックトラッキングアルゴリズムに導入することにより，n-queens 等の例題において，悪い部分解を構成せず，バックトラックを行うことなく効率的に解が得られることが示されている．しかしながら，このヒューリス

† NTT コミュニケーション科学研究所
NTT Communication Science Laboratories

ティックが，完全に悪い部分解を避けられる程度に有効であるためには，変数に与えられた初期値が最終的な解に十分近く，また，変数間の制約の個数が多いことが必要とされ，すべての場合に解になりえないような部分解を構成することを避けることは一般には不可能である．

本論文では，部分解に弱くコミットする，制約違反最少化ヒューリスティックを用いた新しい探索アルゴリズムを提案する．この方法によれば，部分解が成長している間は，その部分解を変更しないが，ある変数に関して，部分解と制約を満たす値が存在しない場合には，それまで構成された部分解を放棄し，部分解の構成を最初からやり直す．このため，解になりえない部分解を構成しても網羅的な探索は行われない．本アルゴリズムでは，放棄された部分解を記録し，そのような部分解を再び構成しないようにすることにより，アルゴリズムの完全性が保証される．

本論文では以下，制約充足問題の定義（第2章）を示し，制約違反最少化ヒューリスティックを用いたバックトラッキングアルゴリズムの紹介を行い，いくつかの例題に関する実験結果を示す（第3章）．さらに，弱コミットメント戦略を用いた探索アルゴリズムを示し，確率モデルによる解析と実験結果を用いてその有効性を示す（第4章）．さらに，近年，バックトラッキングアルゴリズムより効率的であることが示された，状態空間探索型のアルゴリズム[5]と比較しても，本アルゴリズムが効率的であることを実験結果を用いて示す（第5章）．

```
procedure min-conflict-backtracking (left, partial-solution, nogood-list)
  when すべての変数が制約を満足している do
      アルゴリズムを終了, 現在の値の割当が解; end do
  (x_i, d) ← left 中の, 制約を満足していない変数と値の組;
  values ← partial-solution 中の変数の値と整合する x_i の値のリスト;†
  if values が空リスト then
    if partial-solution が空リスト
      then 解が存在しないためアルゴリズムを終了;
      else partial-solution を nogood-list に追加;
        partial-solution の最後の要素を partial-solution から取り除き left に加える;
        min-conflict-backtracking(left, partial-solution, nogood-list) を呼ぶ; end if;
    else value ← values 中で, left 中の変数との制約条件違反の個数が最少である値;
      (x_i, d) を left から取り除く;
      (x_i, value) を partial-solution の最後に加える;
      min-conflict-backtracking(left, partial-solution, nogood-list) を呼ぶ; end if;
```

† x_i の値 d_i が *partial-solution* 中の変数の値と整合するとは，*partial-solution* 中の変数の値との制約をすべて満たし，かつ (x_i, d_i) を *partial-solution* に加えた時，*partial-solution* が *nogood-list* リスト中の要素を包含しないことをいう

図1 制約違反最少化バックトラッキングアルゴリズム
Fig. 1 Min-conflict backtracking algorithm.

2. 制約充足問題

制約充足問題 (CSP) は次のように定義される．n 個の変数 $x_1, x_2, \cdots, x_n$ と，変数のそれぞれが値をとる有限で離散的な領域 $D_1, D_2, \cdots, D_n$，および制約の集合が存在する．本論文では制約は述語によって内包的に定義されるとする．以下の議論では，簡単のためすべての制約は2変数間に定義されるもの (binary) であることを仮定する．この仮定を緩和し一般化することは容易である．変数 x_i の値が d_i であることを，(x_i, d_i) のように二項組で表すことにする．述語 $P_{ij}((x_i, d_i), (x_j, d_j))$ が真となる場合に，これらの変数の値 d_i, d_j はこの制約を満足するという．制約充足問題の解を求めることは，すべての制約を満足する変数の値の組を求めることである．

3. 制約違反最少化バックトラッキング

3.1 アルゴリズム

制約違反最少化ヒューリスティックを用いたバックトラッキングアルゴリズム（制約違反最少化バックトラッキング）を**図1**に示す．このアルゴリズムは，min-conflict-backtracking（すべての変数と初期値のリスト，空リスト，空リスト）を呼ぶことによって実行される．すべての変数には暫定的な初期値が与えられている．最初，すべての変数は *left* に含まれており，アルゴリズムは *left* 中の変数の値を一つ一つ確定して，*partial_solution* に加えていく．変数の値を確定し *partial_solution* に加える際には，その値が *partial_solution* に含まれている変数の値すべてと制約を満たす必要がある．値を確定しようとしている変数に関して，そのような値が存在しない場合にはバッ

クトラックが生じる*．そのような値が複数存在する場合，*left* に含まれている変数（暫定的な初期値は与えられているが，まだ確定されていない変数）との間の制約条件違反の個数が少ないものを選択する．

3.2 実験結果

制約違反最少化バックトラッキングアルゴリズムの性質を調べるため，2種類の例題に関する実験結果を示す（**表1**, **表2**）．実験は，制約充足問題を解くアルゴリズムの評価において広く用いられている n-queens 問題およびグラフの色塗り問題[4]を用いて行う．これらの実験では，前述の手続き（min-conflict-bakctracking）の一回の呼び出しを一ステップとした場合のステップ数（この回数は変数の値の変更の回数に等しい）の合計を測定する．

表1 制約違反最少化バックトラッキングのステップ数（n-queens）
Table 1 Required steps of min-conflict backtracking (n-queens problem).

n	all trials	trials without BT		trials with BT	
	# of steps	# of trials	# of steps	# of trials	# of steps
10	220.7	19	6.0	81 (0)	271.0
50	264.2	83	16.5	17 (3)	1473.4
100	76.4	98	25.7	2 (1)	2563.5

表2 制約違反最少化バックトラッキングのステップ数（グラフの色塗り問題）
Table 2 Required steps of min-conflict backtracking (graph-coloring problem).

n	all trials	trials without BT		trials with BT	
	# of steps	# of trials	# of steps	# of trials	# of steps
60	1852.0	16	15.9	84 (26)	2201.7
90	3065.9	22	17.9	78 (22)	3925.5
120	2494.6	18	18.0	82 (41)	3038.3

変数の初期値は，それぞれの問題に関して，以下のような方法より，比較的良い（最終的な解に近い）初期値が設定されている．すなわち，n-queens では，変数の初期値を一つ一つ決定していくが，その際に，すでに決定された変数の初期値との制約条件違反の個数を最少化する値を選択する．また，グラフの色塗り問題では，文献 4) に示されている，変数の初期値を一つ一つ決定していく際に，すでに決定された変数の初期値と制約を満たす値の個数が少ない変数から優先して値を割り当てる方法を用いる**．

n-queens では，$n=10$, 50, 100 に関して，100 個の異なる初期値に関する試行を行う（$n>100$ の場合に関しては，ほとんどの場合バックトラックなしで解が得られることが示されているため本論文では実験を行わない）．グラフの色塗り問題では，各ノードの取りうる色は3色，全体として $n\times 2$ 本のリンクが存在する問題に関して，$n=60$, 90, 120 に関する評価を行う***．このパラメータの設定は，文献 4) で用いられた，変数間の制約の密度が疎である問題に対応し，制約違反最少化バックトラッキングで効率的に解が得られないことが示されている．グラフの色塗り問題では10個の異なる問題に関して，それぞれ10通りの初期値に関して試行を行う（合計100回の試行を行う）．

実験を妥当な時間内で終了させるため，各試行に関し，5,000 ステップを上限とし，これを越えた場合は処理を中断する．中断された試行に関しては，5,000 回としてカウントする．各項目について，100 回の試行全体としてのステップ数の平均を示す．また，100 回の試行中バックトラックを全く生じることなく解が得られた試行の回数とステップ数の平均，バックトラックを生じた試行のみに関するステップ数の回数の平均も同時に示す．バックトラックを生じた試行の回数の欄の括弧内の数値は，制限時間を越えた試行の回数を示している．

これらの結果により次のことがわかる．

- バックトラックを生じた試行に要するステップ数はバックトラックを生じない試行と比較してはるかに大きく，全体の試行の平均に関して支配的な要因となっている．

4. 弱コミットメント戦略

4.1 基本的なアイデア

前章で示したように，制約違反最少化ヒューリスティックを用いても，バックトラックを完全に避けるこ

* バックトラックアルゴリズムはスタックによりバックトラックが管理されるのが通例であるが，本論文では弱コミットメントアルゴリズムとの比較を容易にするため，解になりえない部分解（nogood）を記録し，そのような部分解を繰り返さないことによりバックトラックを管理するアルゴリズムを示す．

** n-queens ではすべての変数間に制約があるが，グラフの色塗り問題は制約の個数が少なく，変数への初期値の割り当て順序を考慮しないと良い初期値が得られない．

*** 文献 4) で示されている方法に従い，解が存在し，グラフが連結されている問題のみを生成している．

とは一般には不可能である．本論文では次のような新しい探索戦略を提案する．

弱コミットメント戦略：部分解が成長している間は，その部分解を変更しないが，ある変数に関して，部分解と制約を満たす値が存在しない場合（バックトラックが生じる場合）には，それまで構成された部分解を放棄し，現在の変数への値の割当を初期値として，部分解の構成を最初からやりなおす．

この戦略は，従来のバックトラッキングアルゴリズムにおける，一度構成された部分解は，解になりえないことが示されない限り変更されないという強いコミットメント戦略と対照的である．

この戦略を用いる理由は次のとおりである．前章で示したように，バックトラックが生じた時点で，そのまま通常のバックトラックに基づく探索を続けた場合，非常に多くの探索の労力が必要となることはほぼ確実である．一方，バックトラックを生じた時点で，これまで構成した部分解を放棄して部分解の構成を最初からやり直せば，今度はバックトラックなしで非常に早く解が得られる可能性があるのだから，そのままバックトラックに基づく探索を行うよりも，より早く解に到達することが期待されるためである．

4.2 弱コミットメント探索アルゴリズム

図2に弱コミットメント戦略に基づいた探索アルゴリズムを示す．このアルゴリズムでは，値の選択のヒューリスティックとして制約違反最少化ヒューリスティックが用いられている．

このアルゴリズムは，weak-commitment（すべての変数と初期値のペアのリスト，空リスト，空リスト）を呼ぶことによって起動される．

このアルゴリズムと制約違反最少化バックトラッキングアルゴリズムとの差は図2(a)の部分のみであり，通常のバックトラッキングアルゴリズムであればバックトラックが生じる時点で，それまで構成された部分解を放棄して，部分解の構築を改めてやり直す．

このアルゴリズムは，*nogood-list* に放棄された部分解を記録し，そのような部分解を再び構成しないため，アルゴリズムの完全性が保証される．

注意すべきことは，放棄された部分解に含まれる変数は，そのすべてに関して値が変更されるのではない点である．他の変数すべてと制約を満たしている変数は，値の変更の対象とならないため，部分解に含まれている変数のうち，他の変数と制約条件違反が生じている変数のみが変更の対象となる．このように，部分解を放棄する際に，現在の値の割当を新しい暫定的な初期値として探索をやり直すことにより，段階的に変数の値が改善され，最終的な解に近付いていくことが期待される．

4.3 評　　価

4.3.1 確率モデルによる評価

本アルゴリズムとバックトラックに基づく探索アルゴリズムを比較するため，次のような簡単なモデルを考える．すなわち，任意の初期値からスタートしてバックトラックに基づく探索を行った場合，バックトラックなしで解が得られる確率が，初期値によらず一定の値 p で与えられるとする．また，バックトラックなしで解が得られる場合の探索に要するステップ数を n_s とする（$n_s \leq n$）．一方，バックトラックを生じる場合，バックトラックを生じるまでに要する探索のステップ数を n_b（$n_b \leq n$）．バックトラックを生じてからの探索に要するステップ数を B とすると，通常のバックトラックに基づく探索のステップ数の期待値は，

$$n_s p + (B + n_b) q$$

で与えられる（$q = 1 - p$）．

```
procedure weak-commitment (left, partial-solution, nogood-list)
  when すべての変数が制約を満足している do
      アルゴリズムを終了, 現在の値の割当が解; end do
  (x_i, d) ← left 中の, 制約を満足していない変数と値の組;
  values ← partial-solution 中の変数の値と整合する x_i の値のリスト;
  if values が空リスト;
    if partial-solution が空リスト
      then 解が存在しないためアルゴリズムを終了;
      else partial-solution を nogood-list に追加;
        partial-solution の要素をすべてを partial-solution から取り除き
          left に加える; — (a)
        weak-commitment(left, partial-solution, nogood-list) を呼ぶ; end-if;
    else value ← values 中で, left 中の変数との制約条件違反の個数が最少である値;
    (x_i, d) を left から取り除く;
    (x_i, value) を partial-solution に加える;
    weak-commitment(left, partial-solution, nogood-list) を呼ぶ; end if;
```

図2 弱コミットメント探索アルゴリズム
Fig. 2 Weak-commitment search algorithm.

これに対して，弱コミットメント戦略に基づき，改めて部分解の構成を行う場合の探索に要するステップ数の期待値は，やり直しなしで解が得られる確率 p でその場合のステップ数は n_s，1回のやり直しで解が得られる確率は pq，ステップ数は n_s+n_b，以下同様に，k 回のやり直しで解が得られる確率は pq^k，ステップ数は n_s+kn_b で与えられる．やり直しの回数は，良く知られた確率 p の幾何分布となり，その期待値は q/p で与えられる．よって，ステップ数の期待値は

$$n_s+n_bq/p$$

で与えられる．

弱コミットメント探索アルゴリズムがバックトラック型のアルゴリズムより効率的である条件は，

$$n_sp+(B+n_b)q>n_s+qn_b/p$$

であり，これより $p>n_b/(B+n_b-n_s)$ が得られる．$B+n_b\gg n_s$ と考えて良く，また，$n_b\leq n$ を用いて，弱コミットメント探索アルゴリズムがバックトラックアルゴリズムより効果的であるための十分条件として，

$$p>n/(B+n_b)$$

が得られる．すなわち，バックトラックなしで解が得られる確率が，変数の個数 n とバックトラックを生じる場合のステップ数の比より大きければよい．

図 1, 2 の実験結果から得られる，n とバックトラックを生じた場合のステップ数との比は，明らかに 1/20 より小さく，よって，$p>0.05$ なら，この条件が成立する．実験結果から明らかなように，100 回の試行中，バックトラックなしで解が得られた回数は5回よりも大きく，弱コミットメント探索のほうが効率的であることが予想される．

実際には，バックトラックなしで解が得られる確率は初期値に依存して変化する．しかしながら，最初に変数に与えられた初期値が，比較的解に近い良い値である場合には，最初の時点でバックトラックなしで解が得られる確率と，やり直しを行う時点でバックトラックなしで解が得られる確率には大きな差はないと考えられる．一方，変数の初期値が解から離れている場合には，やり直しを繰り返すことにより，次第に変数の値は最終的な解に近付いていき，バックトラックなしで解が得られる確率は増加していくことが予想される．このような場合には，前述の条件が成立しない場合，すなわち，任意の初期値からスタートした際のバックトラックなしで解が得られる確率が非常に低い場合でも，やり直しを繰り返すことによって初期値が改善されていくため，弱コミットメント探索アルゴリズムがバックトラック型のアルゴリズムより効率的となることが予想される．

4.3.2 実験結果

表 3, **表 4** に，表 1，表 2 で用いた各問題に関する，弱コミットメント探索アルゴリズムと制約違反最少化バックトラッキングアルゴリズムの，制限時間内に処理が終了しなかった試行の回数，ステップ数，制約条件のチェックの回数の比較を示す．制約チェックの回数の測定に当たっては，前回のステップでの制約チェックの結果を保存して，差分のみを計算することによる最適化を行っている．また，これらのアルゴリズムでは，最初に与えられた制約条件のチェック以外に nogood との整合性チェックが必要となるため，一つの nogood との整合性チェックを一回の制約チェックとしてカウントする．

また，確率モデルの妥当性を示すため，また，表1の実験結果から得られた p の実測値を前述のモデルにあてはめた，弱コミットメント探索アルゴリズムの

表 3 弱コミットメント探索と制約違反最少化バックトラッキングの比較 (n-queens)
Table 3 Comparison between weak-commitment search and min-conflict backtracking (n-queens).

n	weak commitment			min-conflict BT		
	# of failure	# of steps	# of checks	# of failure	# of steps	# of checks
10	0	29.7	2266.6	0	220.7	13167.0
50	0	23.6	48593.5	3	264.2	300174.8
100	0	27.1	236821.7	1	76.4	912465.1

表 4 弱コミットメント探索と制約違反最少化バックトラッキングの比較（グラフの色塗り問題）
Table 4 Comparison between weak-commitment search and min-conflict backtracking (graph-coloring).

n	weak commitment			min-conflict BT		
	# of failure	# of steps	# of checks	# of failure	# of steps	# of checks
60	0	107.7	18875.1	26	1852.0	413750.6
90	0	133.1	38331.4	22	3065.9	1198969.6
120	0	206.5	71852.0	41	2494.6	895012.2

やり直しの回数の予想値 (q/p)，および実測値を**表 5** に示す．

これらの評価結果から次のような知見が得られる．

- 弱コミットメント探索アルゴリズムは，すべての項目に関して制約違反最少化バックトラッキングアルゴリズムより効率的である．
- 確率モデルを用いたやり直しの回数の予想値は実測値と比較的良い一致が得られており，確率モデルの妥当性を示している．

また，両方の例題に関して，ランダムに与えた初期値に関して実験を行ったところ，制約違反最少化バックトラッキングアルゴリズムは，バックトラックを生じる場合が増加して，効率が著しく低下するのに対し，弱コミットメント探索アルゴリズムはほとんど変化が見られなかった．この理由は，やり直しを行うことによって，ランダムな初期値からスタートしても，すぐに比較的良い値が変数に与えられるためだと考えられる．

5. 状態空間探索アルゴリズムとの比較

近年，文献 5) により，状態空間探索に基づく制約充足問題の解法が提案され，制約違反最少化バックトラッキングで効率的に解が得られない，疎に結合されたグラフの色塗り問題に関して，効率的に解が得られることが示されている．本章では文献 5) に示されている二種類のアルゴリズムと，弱コミットメント探索アルゴリズムとの比較を行う．

5.1 状態空間探索アルゴリズム

状態空間探索アルゴリズムでは，バックトラック型のアルゴリズム，もしくは弱コミットメントアルゴリズムのように部分解を構成することなく，全体として制約条件違反の個数が少なくなるように変数の値がくり返し変更される．状態空間探索アルゴリズムでは，局所最適（どの一つの変数の値を変更しても，制約条件違反の個数が減少しない状態）に陥る可能性があり，文献 5) では局所最適への二種類の対応方法が示されている．

fill アルゴリズム：局所最適となった値の割当の評価値（制約条件違反の個数）を 1 増加させたものを新しい評価値とし，局所最適の近傍の状態の評価値を相対的に良くすることにより，局所最適から脱出する．

breakout アルゴリズム：局所最適となった状態で制約条件違反が生じている制約に関して重み付けを増加させたものを，新しい評価基準とし，局所最適の近傍の状態の評価値を相対的に良くすることにより，局所最適から脱出する．

fill アルゴリズムは LRTA* アルゴリズム[2]の変形であると考えられる．これらのうち，fill アルゴリズムは，解が存在すれば必ず解が得られることが保証されるが，breakout アルゴリズムは無限ループに陥る可能性がある．また，どちらのアルゴリズムも解が存在しない場合には終了せず，解が存在しないことを発見できない☆．

表 5 n-queens 問題におけるやり直しの回数
Table 5 No. of restarting in n-queens.

n	model	experiment
10	4.0	3.6
50	0.2	0.19
100	0.02	0.02

5.2 比較結果

これらのアルゴリズムと**表 6**，**表 7** に，前章の評価と同じ例題に関する制限時間内に処理が終了しなかった試行の回数，ステップ数，制約チェックの回数の比較結果を示す．実験を妥当な時間で終了させるため，各試行に関して，ステップ数 5,000 回を上限とし，上限を越えた場合は処理を中断し，ステップ数 5,000 としてカウントする．制約チェックの回数の測定に当たっては，すべてのアルゴリズムに関して，前回のステップでの制約チェックの結果を保存して，差分のみを計算することによる最適化を行っている．また，弱コミットメント探索アルゴリズムでは nogood との整合性チェックの回数を加算したものを示す．

これらの結果から次のような傾向が示されている．

- 弱コミットメントアルゴリズムは，fill アルゴリズムと比較して，すべての項目に関してステップ数，制約チェックの回数が少なくなっている．一方，breakout アルゴリズムと比較すると，ステップ数は多くなる場合が存在するが，すべての項目に関して制約チェックの回数は少なくなっている．
- 状態空間探索型のアルゴリズムは，弱コミットメント探索アルゴリズムと比較して各ステップ（一回の変数の値の変更）に要する処理量が多い．値を変更する変数を選択する際に，弱コミットメン

☆ fill アルゴリズムに関しては，admissible な評価関数を用いることにより解が存在しないことを発見するように改良できる可能性があるが，文献 5) では，そのような評価関数は用いられていない．

トアルゴリズムでは，制約条件違反を生じている変数から任意のものを選べば良い．一方，状態空間探索アルゴリズムでは，制約条件違反の個数を減少させるものを選ばなくてはならず，最悪の場合（現在の値の割当が局所最適である場合）には，制約条件違反を生じているすべての変数に関して，制約条件違反の個数を減少させるものがあるかどうかのチェックが必要になる．

- グラフの色塗り問題に関しては fill アルゴリズムは他のアルゴリズムと比較して非効率的である．この理由は推定凹部 (heuristic depression) と呼ばれる問題[1]に起因すると考えられる．推定凹部とは，多くの局所最適解が隣接しているものであり，いったん推定凹部に迷いこむと，そこから脱出するのに多くの労力が必要とされる．実際，グラフの色塗り問題では，評価値が1である（制約条件違反の個数が一つだけである）隣接した状態が多く存在し，これらの状態に陥ると，そこから脱出するのに多くの探索ステップが必要とされる．breakout アルゴリズムでは，アルゴリズムの完全性を犠牲にして，推定凹部に陥ったときに，推定凹部内のいくつかの状態の評価値を同時に増加させることにより，推定凹部からの脱出を早くしていると考えられる．

一方，弱コミットメント探索アルゴリズムでは，制約条件違反の個数は絶対的な評価基準ではなく，現在注目している変数の値を決める際に，部分解と制約を満たす値が複数存在する場合の補助的な選択の基準としてのみ用いられる．このため，制約条件違反の個数は，値の変更後増加する可能性もあり，fill アルゴリズムと比較して推定凹部に陥ることの影響が小さい．

6. 考　　察

6.1 アルゴリズムの性質

本論文で評価を行った4種類のアルゴリズムの性質を**表8**にまとめて示す．最初の性質は，変更した変数の値に対して強くコミットするかどうかである．強いコミットメントを行う場合は，悪い値を決定した場合に網羅的な探索が必要となり，アルゴリズムの効率は著しく低下する．状態空間探索型のアルゴリズムではこのような強いコミットメントは行われない．

二番目の性質は，アルゴリズムの実行過程で制約を満たす部分解が構築されるかどうかである．制約を満

表 6 弱コミットメント探索と状態空間探索アルゴリズムの比較 (n-queens)
Table 6 Comparison between weak-commitment search and state-space search (n-queens).

n	weak commitment			fill			breakout		
	# of failure	# of steps	# of checks	# of failure	# of steps	# of checks	# of failure	# of steps	# of checks
10	0	29.7	2266.6	0	39.8	6535.8	0	41.7	7065.0
50	0	23.6	48593.5	0	45.4	232970.5	0	37.9	180393.5
100	0	27.1	236821.7	0	51.9	1114047.0	0	38.9	777051.1

表 7 弱コミットメント探索と状態空間探索アルゴリズムの比較（グラフの色塗り問題）
Table 7 Comparison between weak-commitment search and state-space search (graph-coloring).

n	weak commitment			fill			breakout		
	# of failure	# of steps	# of checks	# of failure	# of steps	# of checks	# of failure	# of steps	# of checks
60	0	107.7	18875.1	1	284.0	165716.3	0	82.0	43174.4
90	0	113.1	38331.4	1	693.7	547243.2	0	106.1	92224.5
120	0	206.5	71852.0	4	904.6	1005676.2	0	157.1	201937.1

表 8 アルゴリズムの性質の比較
Table 8 Comparison of the characteristics of algorithms.

	min-conflict BT	weak commitment	breakout	fill
avoid strong commitment	no	yes	yes	yes
construct partial solution	yes	yes	no	no
always find a solution if exists	yes	yes	no	yes
stop if no solution exists	yes	yes	no	no

たす部分解を，構築，拡張することによって解が得られるというのは制約充足問題に特有の性質である．制約違反最少化ヒューリスティックが十分な情報を与えられない場合（推定凹部に陥った場合），fill アルゴリズムのような一般的な状態空間探索型のアルゴリズムにより効率的に解を得ることは期待できない．

3番目，4番目の性質はアルゴリズムの完全性に関するものであり，解があればそれを必ず見つける，解がなければ，解がないことを発見することにより終了することを保証するものである．

6.2 アルゴリズムの計算量

制約充足問題はNP完全な問題であり，弱コミットメント探索アルゴリズムの最悪ケースの計算量は変数の個数 n に対して指数的である．

必要とされるメモリ量は，*nogood-list* の大きさによって決まり，これも最悪ケースには変数の個数 n に対して指数的となる．一方，スタックを用いてバックトラック型のアルゴリズムをインプリメントした場合のメモリ量は n に対してリニアであり，弱コミットメント探索アルゴリズムは，探索の順序を柔軟に変えながらも完全性を保証しているため，必要とされるメモリ量が指数オーダになってしまっている．これは fill アルゴリズムに関しても同様で，メモリ量は最悪ケースには変数の個数 n に対して指数的となる．

しかしながら，*nogood-list* の大きさ，すなわち放棄された部分解の個数は，解が求まるまでに必要とされるステップ数を越えることはなく，実験結果で用いた例題の範囲では，必要とされるステップ数は n に対して，ほぼリニアの範囲であり，現実的には *nogood-list* の大きさが問題になることはないと考えられる．また，他の *nogood-list* の要素を包含するような要素は冗長であり，このような要素を *nogood-list* から取り除くことにより，*nogood-list* の要素数を減少させることが可能である．

また，*nogood-list* の大きさを制限し，最も新しく得られた定数個の nogood のみを記録することも可能である．この場合は，いくつかの nogood の間を巡回する無限ループに陥る可能性があり，理論的なアルゴリズムの完全性は保証されないが，現実的には非常に多くの nogood を巡回するループが生じることは稀であると考えられる．実際，*nogood-list* に保存される nogood の個数を10個に制限した場合，すべての例題で無限ループに陥ることなく解が得られている．

6.3 他のヒューリスティックスとの組合せ

本論文では，値の順序付けヒューリスティックとして制約違反最少化ヒューリスティックを用いた弱コミットメント探索アルゴリズムを示したが，弱コミットメント探索アルゴリズムを他のヒューリスティックと組み合わせて用いることも可能である．例えば，値を変更する変数を選択する際に，最も強く制約されている変数を選択するなどの，変数の順序付けヒューリスティックスを用いることが考えられる．

一方，値／変数の順序付けヒューリスティックスを全く用いない場合には，弱コミットメント探索アルゴリズムによる高速化は期待できない．この理由は，ヒューリスティックスなしではバックトラックなしで解が得られる確率はほとんど0であり，やり直しを繰り返しても解が得られることが期待できないためである．

7. おわりに

本論文では制約充足問題を解く際に，部分解に弱くコミットする，完全性が保証される探索アルゴリズムを提案した．確率的なモデルおよび実験結果を用いて，バックトラッキングアルゴリズムに対する本アルゴリズムの優越性を示した．さらに，実験結果を用いて，近年，バックトラッキングアルゴリズムより効率的であることが示された，状態空間探索型のアルゴリズムと比較しても，本アルゴリズムが効率的であることを示した．

今後の課題として，より多くの例題，および現実的な応用問題（通信ネットワークにおける回線割当問題等[6]）でこの探索アルゴリズムの効果を確認すること，さらに，複数の自律的なエージェントの関連する制約充足問題を解く非同期バックトラッキングアルゴリズム[7),8)]に，弱コミットメント戦略を適用することが挙げられる．

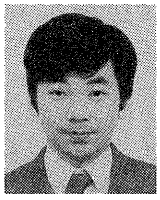
謝辞 本研究の機会を与えて下さったNTTコミュニケーション科学研究所の河岡所長，中野グループリーダに感謝致します．また，草稿の段階で有益なコメントを頂いた同研究所の桑原主任研究員，西部社員に感謝致します．

参考文献

1) Ishida, T.: Moving Target Search with Intelligence, *AAAI-92*, pp. 525-532 (1992).
2) Korf, R. E.: Real-Time Heuristic Search, *Artif. Intell.*, Vol. 42, No. 2-3, pp. 189-211

(1990).
3) Mackworth, A.K.: Constraint Satisfaction Shapiro, S.C. (ed.), *Encyclopedia of Artificial Intelligence*, pp. 205-211, Wiley-Interscience Publication, New York (1987).
4) Minton, S., Johnston, M.D., Philips, A.B. and Laird, P.: Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems, *Artif. Intell.*, Vol. 58, No. 1-3, pp. 161-205 (1992).
5) Morris, P.: The Breakout Method for Escaping From Local Minima, *AAAI-93*, pp. 40-45 (1993).
6) 西部喜康, 桑原和宏, 石田 亨, 横尾 真: 分散制約充足の高速化と通信網回線設定への適用, 電子情報通信学会論文誌, Vol. J76-DII, No. 10, pp. 2204-2213 (1993).
7) Yokoo, M., Durfee, E., Ishida, T. and Kuwabara, K.: Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving, *Proc. of 12th IEEE International Conference on Distributed Computing Systems*, pp. 614-621 (1992).
8) 横尾 真, エドモンド H. ダーフィ, 石田 亨, 桑原和宏: 分散制約充足による分散協調問題解決の定式化とその解法, 電子情報通信学会論文誌, Vol. J-75D-I, No. 8, pp. 704-713 (1992).

(平成5年8月23日受付)
(平成6年4月21日採録)



横尾 真 (正会員)

1962年生. 1984年東京大学工学部電子工学科卒業. 1986年同大学院修士課程修了. 同年NTTに入社. 1990年～1991年ミシガン大学客員研究員. 現在NTTコミュニケーション科学研究所に勤務. 分散AI, 制約充足問題に関する研究に従事. 分散探索, 分散制約充足問題等に興味を持つ. 1992年人工知能学会論文賞. 人工知能学会, ソフトウェア科学会, AAAI各会員.