制約充足問題への近似解法の適用検討
—N-クイーン問題を中心として—

永 井 保 夫*

制約充足問題は人工知能や画像解析の分野をはじめ、グラフの問題やパズルなどの探索問題、いわゆる組み合わせ問題を対象として研究がおこなわれている。制約充足問題とは、変数の有限集合および各変数に対応する離散値の有限集合のドメインから値を選択し、すべての制約を満足するように各変数に対して値を割り当てる問題である。制約とは適用対象の構成要素およびその属性間で成立する関係を宣言的に記述したものである。

制約充足問題における従来の解法としては、バックトラックを基本とした探索による方法や整合化手法などに代表される前処理を用いる方法が代表的である。この場合には、解法としては探索空間の全域を対象としてすべての可能性を探索するもので、アルゴリズムの完全性を保証している。完全性とは、探索空間に解が存在すれば、必ず見つけられることと、解がなければ必ず存在しないことを保証するものである。

これに対して、局所探索法による解法はこのアルゴリズムの完全性を保証しないかわりに、解を高速で求めることを特徴とした近似解法である。本論文では、制約充足問題の解法である近似解法についてサーベイするとともに、パズルの代表例であるN-クイーン問題を対象として、局所探索法による解法である山登り探索と制約違反最小化ヒューリスティックに基づいた山登り探索の適用結果について説明する。

キーワード：制約充足問題，近似解法，局所探索，アルゴリズム，高速化

Towards Local Search Methods for Constraint Satisfaction Problems, Focusing on N-queen Problem

Yasuo NAGAI

This paper reports on local search methods for constraint satisfaction problems. Especially, it focuses on and specifies a survey of these methods and application of two local search methods to N-queen problem and shows its effectiveness.

Keyword：constraint satisfaction problem, approximate method, local search, algorithm, speed-up

*東京情報大学総合情報学部情報システム学科
Tokyo University of Information Sciences, Faculty of Informatics, Department of Information Systems

2004年11月19日受理

1　はじめに

制約充足問題は人工知能や画像解析の分野をはじめ、グラフの問題やパズルなどの探索問題、いわゆる組み合わせ問題を対象として研究がおこなわれている[1][2][3][4][8][9][10]。制約充足問題とは、変数の有限集合および各変数に対応する離散値の有限集合のドメインから値を選択し、すべての制約を満足するように各変数に対して値を割り当てる問題である。制約とは適用対象の構成要素およびその属性間で成立する関係を宣言的に記述したものである。制約充足問題における従来の解法としては、バックトラックを基本とした探索による方法や整合化手法などに代表される前処理を用いる方法が代表的である［1][2][4][6]。この場合には、解法としては探索空間の全域を対象としてすべての可能性を探索するもので、アルゴリズムの完全性を保証している。完全性とは、探索空間に解が存在すれば、必ず見つけられることと、解がなければ必ず存在しないことを保証するものである。

これに対して、局所探索法による解法はこのアルゴリズムの完全性を保証しないかわりに、解を高速で求めることを特徴とした近似解法である［1][6][12]。本論文では、制約充足問題の解法である近似解法についてサーベイするとともに、パズルの代表例であるN-クイーン問題［11］を対象として、局所探索法による解法である山登り探索と制約違反最小化ヒューリスティックに基づいた山登り探索の適用結果について説明する。

2　制約充足問題

2.1　制約充足問題の定義

制約充足問題は人工知能や画像解析の分野をはじめ、グラフの問題やパズルなどの探索問題、いわゆる組み合わせ問題を対象として研究がおこなわれている。制約充足問題とは、次のような変数の有限集合および各変数に対応する離散値の有限集合のドメインから値を選択し、すべての制約を満足するように各変数に対して値を割り当てる問題である。制約とは適用対象の構成要素およびその属性間で成立する関係を宣言的に記述したものである。

- 変数集合：$V=\{v_1,\ldots,v_n\}$，各 v_i は互いに独立な変数。
- ドメイン：離散値の有限集合$D_i=\{d_{i_1},\ldots,d_{i_m}\}$，$i=1,\ldots,n$，$d_{ij}$ $(j=1,\ldots,m)$ は数値または記号値をあらわす。各変数 v_i には D_i の要素である値 d_{i_m} が割り当てられる。
- 制約集合：$C=\{c_1,\ldots,c_l\}$．ここで、各 c_i は制約で、それは次のような等式
  $(v_1, v_2,\ldots,v_n)=$〈直積$D_1\times D_2\times\ldots\times D_n$の部分集合〉
  の表現形式をとる。

制約充足問題における従来の解法としては、バックトラックを基本とした探索による方法や整合化手法などに代表される前処理を用いる方法が代表的である。ところで、n 個の変数に対する制約、つまり n 項制約は2項制約により表現できるので、今後は、2項制約のみを対象とした制約充足問題を考える。

制約充足問題では、変数を表すノードと制約によりラベル付けされたアーク（エッジ）からなるグラフとして表現される。このようなグラフは制約ネットワークとして静的な問題の記述に適した知識表現とみなれる。ネットワークを用いたアプローチの利点は、知識の構造化が容易であり、知識の無矛盾性を効率的に管理できることである。制約ネットワーク$\mathcal{R}$は、n 個の変数 $v_1, v_2,\ldots,v_n$ を要素とする集合V、各変数に対するドメイン$D_1, D_2,\ldots, D_n$ を要素とする集合 D および $c_1, c_2,\ldots,c_n$ を要素とする制約（n 項関係）集合から構成され、三つ組（V, D, C）により表現される。n 個の変数 $v_1, v_2,\ldots,v_n$ に対する制約 $c(v_1, v_2,\ldots,v_n)$ とは、n 個の集合 $\{D_1, D_2,\ldots, D_n\}$ に対して成り立つ関係（つまり n 項関係）を表し、無矛盾な変数値の直積 $D_1\times D_2\times\ldots\times D_n$ の部分集合である。例えば、2項制約ネットワークはすべての制約が2項関係である（高々2個の変数からなる）ネットワークである。

それから、変数に対してドメインから値が割り当てられるとき、変数が具体化されたという。変数集合の具体化とは、各変数のドメインからの値の割り当てを意味する。つまり、変数集合 $\{v_{i_1},\ldots,v_{i_k}\}$ の具体化とは、順序付けされたペアのタプル（$\langle v_{i_1}, a_{i_1}\rangle,\ldots,\langle v_{i_k}, a_{i_k}\rangle$）を表す。但し、各ペア（$\langle v, a\rangle$）は変数 v への値 a の割り当てを、a は v のドメインを示す。このような変数集合の具体化は（$v_1=a_1,\ldots,v_i=a_i$）と表現する。たとえば、（$\langle v_1, a_1\rangle,\ldots,\langle v_i, a_i\rangle$）は $\bar{a}=(a_1,\ldots,a_i)$ と表す。

制約ネットワーク$\mathcal{R}=(V, D, C)$ の解とは、すべて

の制約を満足するようなすべての変数の具体化を意味する。一般に、制約充足（制約を満足させる）とは、このような制約ネットワークを解くことであり、すべての制約が満足されるようにネットワーク中のすべての変数に対して値を求めることに相当し、単解または全解を求めることを意味する。

2.2 局所探索法を用いる解法（近似解法）

局所探索法（local search）とは、なんらかの方法で得られた近似解（ここでは、すべての変数の割り当て）に対して、その近傍への変更を加えることで得られる他の解を調べて、解への評価が向上するように置き換えていく方法である。局所探索法による解法は、このアルゴリズムの完全性を保証しないかわりに、解を高速で求めることを特徴とした近似解法である。これに対して、バックトラックなどの体系的な探索手法を用いた解法は厳密解法であり、探索空間の全域を対象としてすべての可能性を探索するもので、アルゴリズムの完全性を保証している。ここでいう完全性とは、探索空間に解が存在すれば、必ず見つけられることと、解がなければ必ず存在しないことを保証するものである。

以下では、近似解法の代表的な手法である局所探索法による解法を説明する。

2.2.1 山登り探索（hill-climbing search）

山登り探索アルゴリズムの処理手順は以下のとおりである。

Step 1：ランダムに変数に対して値を割り当て、これを解候補とする。
Step 2：制約違反をチェックする。
Step 3：制約違反がなければ、探索を終了する。
Step 4：制約に違反している場合には、できるだけ制約違反の数が少なくなるように解候補の中から一つの変数を選択し，値の割り当てを変更する（ここでは、ヒューリスティック関数 h の値が最も小さいものを選択する）。
Step 5：Step 2 へ戻る。

なお、山登り探索アルゴリズムの詳細は、図 1 に示される。

山登り探索アルゴリズムは、ヒューリステック関数の値が増加する方向に移動することを単純に繰り返し

```
procedure HILL-CLIMBING
入力：最大実行回数　max_steps
出力：解または失敗 failure
1  begin
2    current ← すべての変数への値の初期割り当て
3    for i = 1 to max_steps do
4      if current がCSP に対する解であるthen return
       current
5      ヒューリステック関数 h の値が最小となる変数
       variable に対する値 value の割り当てを求める
6      current 中の variable を value に置き換える
7    return failure
8  end;
```

図 1　山登り探索アルゴリズム

ている。山登り探索アルゴリズムは、以下の項目に対しては欠点を持っている。

・局所的最大

局所的最大とは、大局的な最大と異なり、状態空間での最大の頂上よりも低い頂上をいう。ひとたび、局所的極大に到達してしまうと、解が満足できるものから程遠くてもアルゴリズムは終了してしまう。

・高原

高原とは評価関数（ここではヒューリスティック関数）が基本的に平坦であるような状態空間である。このような場合には、アルゴリズムはランダムに動き回ることしかできない。

・峰

峰とは脇が急激に傾斜しているため、探索が峰の頂上に容易にはたどり着けるが、峰の頂上はピークに向けて非常にゆったりした傾斜しかない場合が可能性として考えられる。その場合には、峰の頂上に沿って進むオペレータがない限りは、探索は少しの前進しかできないで峰の頂上付近で振動してしまう。

上記のいずれの場合でも、山登り探索アルゴリズムはそれ以上進めなくなったところまで進む。そこで、この状況を打破するために、前進できなくなるところまで進み、新たに前とは異なる出発点から探索を始めるアルゴリズムをランダム再スタート山登り探索アルゴリズムといい、その詳細を図 2 に示す。

```
procedure RESTART-HILL-CLIMBING
入力：最大実行回数　max_steps
出力：解または失敗　failure
1  begin
2    restart : current ← すべての変数への値の初期割り当て
3    for i = 1 to max_steps do
4      if current が CSP に対する解である then return
       current
5      if current が極値（局所的最大（小））である
         then goto restart
6        else ヒューリステック関数 h の値が最小となる変
         数 variable に対する値 value の割り当てを求める
7      current 中の variable を value に置き換える
8    return failure
9  end;
```

図 2　ランダム再スタート山登り探索アルゴリズム

```
procedure MIN-CONFLICTS
入力：最大実行回数　max_steps
出力：解または失敗　failure
1  begin
2    current ← すべての変数への値の初期割り当て
3    for i = 1 to max_steps do
4      if current が CSP に対する解である then return
       current
5      variable ← ランダムに選択された違反制約の変数
6      valve ← 変数に対する新たな値の割り当てに対して、
         他の変数との制約の違反の数を最小化する値
7      current 中の variable を value に置き換える
8    return failure
9  end;
```

図 3　制約違反最小化ヒューリスティックに基づいた山登り探索アルゴリズム

2.2.2　制約違反最小化ヒューリスティックスに基づいた山登り探索（hill-climbing search based on minimum conflicts heuristics）

制約違反最小化ヒューリスティックスは、変数に対して新たな値の割り当てに際しては、他の変数との制約の違反の数を最小化するような値を選択するというヒューリスティックスである。山登り探索に対して、この制約違反最小化ヒューリスティックスを適用することで、大規模なn-クイーン問題（たとえば、$n = 1$ 億）やグラフの彩色問題などを高速に解くことが可能になった。

制約違反最小化ヒューリスティックスに基づいた山登り探索アルゴリズムの処理手順は以下のとおりである。

Step 1：初期値の生成をおこなう。
Step 2：制約違反をチェックする。
Step 3：制約違反がなければ、探索を終了する。
Step 4：制約に違反している場合には、制約に違反している変数を 1 つ選択する。
Step 5：Step 4 で選択した変数に対して、制約違反の数が最少になるような値を選んで代入する。
Step 6：Step 2 へ戻る。

なお、制約違反最小化ヒューリスティックスに基づいた山登り探索アルゴリズムの詳細は、図 3 に示される。

2.2.3　制約違反最小化ランダムウォークに基づいた探索（random-walk search based on minimum conflicts heuristics）

制約違反最小化ランダムウォークに基づいた探索アルゴリズムでは、上記の制約違反最小化ヒューリスティックスに基づいた山登り探索アルゴリズムにおいて、極小値（極大）値を超えることができない場合が発生するので、これに対処するためにノイズ戦略が導入されている。このランダムウォーク戦略は最も一般的な戦略のひとつであり、競合している変数が与えられた場合には、確率 p でランダムに値を取り出し、確率 $1 - p$ で制約違反最小化ヒューリスティックスを適用する。このアルゴリズムでは、ランダムな確率 p により制御され、p がアルゴリズムの性能に大きな影響を与える。経験的には、p は$0.02 \leq p \leq 0.1$ として利用されている。

なお、制約違反最小化ランダムウォークに基づいた探索アルゴリズムの詳細は、図 4 に示される。

3　近似解法の制約充足問題への適用

以下では、n-クイーン問題を取り上げ、近似解法で代表的な手法である山登り探索と制約違反ヒューリスティックスを利用した山登り探索を適用する。

3.1　n-クイーン問題（n=4）

ここでは、制約充足問題の代表的な例題として n-クイーン問題を取り上げる。n-クイーン問題は、n 行 x n 列の盤面に n 個のクイーンを互いの利き筋が重なら

```
procedure MIN-CONFLICTS-RANDOM-WALK
入力：最大実行回数　max_steps、確率 P
出力：解または失敗　failure
1  begin
2    current ← すべての変数への値の初期割り当て
3    for i = 1 to max_steps do
4      if 確率が P である then
5        variable ← ランダムに選択された違反制約の変数
6        valve ← 変数 variable に対してランダムに選択された値の割り当て
7      else
8        variable ← ランダムに選択された違反制約の変数
9        valve ← 変数に対する新たな値の割り当てに対して、他の変数との制約の違反の数を最小化する値
7      current 中の variable を valve に置き換える
8    return current
9  end;
```

図 4　制約違反最小化ランダムウォークに基づいた探索アルゴリズム

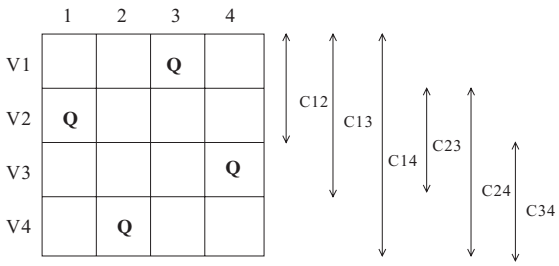

図 5　n-クイーン問題（n＝4）

ないように配置する問題である。クイーンは将棋の飛車と角をあわせた駒とみなせ、縦横斜めの 8 方向へ任意に動くことができる（図 5）。

- 変数集合：$V = \{v_1, \ldots, v_n\}$，各 v_i は互いに独立な変数。盤の $1 \le i \le n$ 行に対応する。
  $V = \{v_1$（＝行 1），v_2（＝行 2），v_3（＝行 3），v_4（＝行 4）$\}$
- ドメイン：離散値の有限集合　$D_i = \{d_{i_1}, \ldots, d_{i_m}\}$，$i = 1, \ldots, n$, d_{i_j} $(j = 1, \ldots, m)$ は数値または記号値をあらわす。盤の列をあらわす。
  $D_1 = \{1, \ldots, 4\}$, $D_2 = \{1, \ldots, 4\}$, $D_3 = \{1, \ldots, 4\}$, $D_4 = \{1, \ldots, 4\}$,
- 制約集合：$C = \{c_1, \ldots, c_t\}$．ここで、各 c_i は制約をあらわす。ここでの制約は任意の 2 つのクイーンが互いに攻撃しあわないという関係 $(v_i \neq v_j) \wedge (|v_i - v_j| \neq j - i)$ をあらわす。具体的に制約で表現すると次のようになる。

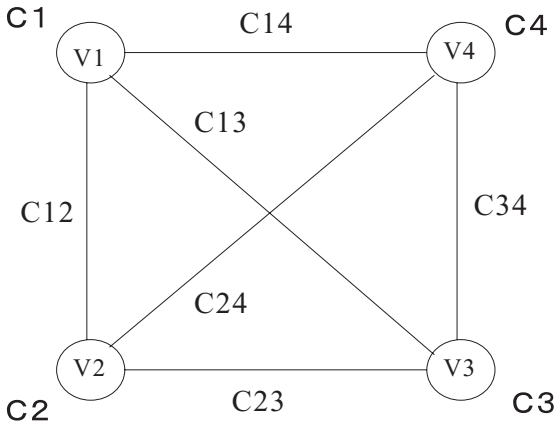

図 6　n-クイーン問題（n＝4）の制約ネットワーク表現

$c_1 = R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$
$c_2 = R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3))\}$
$c_3 = R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4), (4,2), (4,3)\}$
$c_4 = R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$
$c_5 = R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$
$c_6 = R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$

なお、R_{ij} の要素 (a, b) は、2 個のクイーンを置くという関係、すなわち i 行の a 列目にクイーンを置くこと（変数 v_i への値 a の割り当て）と j 行の b 列目にクイーンを置くこと（変数 v_j への値 b の割り当て）という 2 つの関係を表わしている。

一般に、n-クイーン問題は ${}_nC_2 = n(n-1)/2$ 個の 2 項制約を用いたネットワークとして表される。4 クイーン問題は、6 個の 2 項制約として表現され、制約ネットワークは、図 6 のように 4 個のノードからなる完全グラフ K_4 により表現される。各ノードには変数に関する単項制約が与えられ、たとえば、ノード v_1 には変数 v_1 に関する単項制約 C_1 が与えられる。なお、この単項制約は変数 v_1 の値としてドメイン D_1 からどれかひとつの要素 d_{1_k} を必ず割り当てる制約をあらわす。ノード v_1 と v_2 の間のエッジ C_{12} は 2 つの変数 v_1 と v_2 間において成立する 2 項制約 C_{12} を表す。

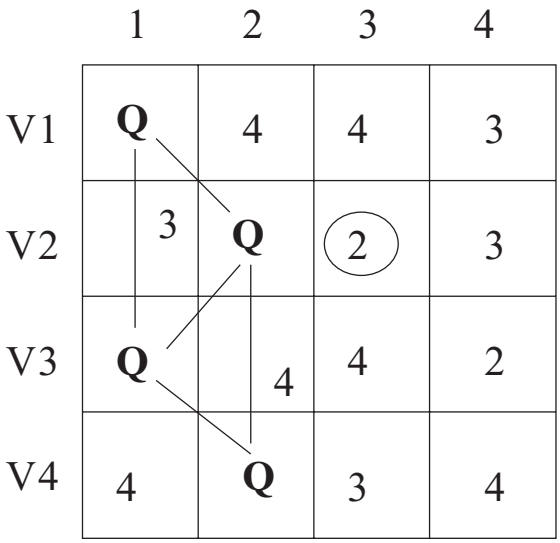

図7　n-クイーン問題（n＝4）：ランダムな初期値の生成

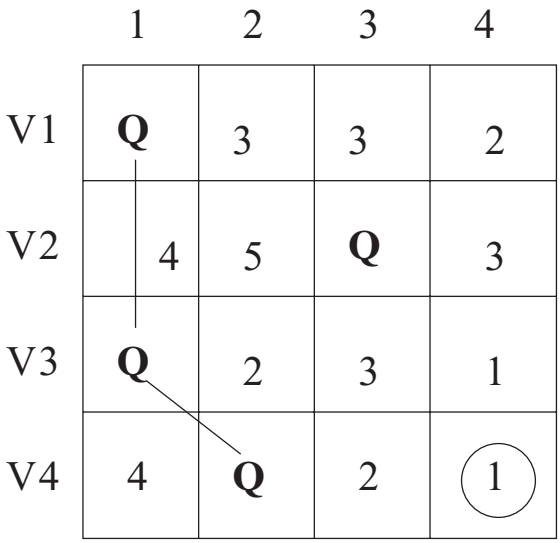

図8　n-クイーン問題（n＝4）：改良（1）

3.2　4-クイーン問題への適用

3.2.1　山登り探索の適用

まず、4-クイーン問題に対する山登り探索の適用を考える。山登り探索で取り扱うヒューリスティック関数 h は、盤面にすべての駒を配置した場合の違反制約の数を求めるものである。この関数の最小値は0（h＝0）であり、その場合には解が得られていることを表す。

実際の4-クイーン問題への山登り探索の適用手順は以下のようになる。

1．図7は、すべての駒がランダムに配置された盤面を示している。この盤面では、h＝5となる。たとえば、盤面上に線で結ばれている一組の駒は制約に違反している状況を表している。また、盤面上で駒が置かれていないマスには、1つの駒をそのマスに移動した場合の違反制約の数が示されている。

2．図7のマスの中で、最も小さい値が表示されているマスを選択する。ここでは、丸で囲まれているマス（h＝2）を選び、v_2 行目の駒を2番目から3番目の列に移動する。その結果の盤面上の駒の配置が、図8に示される。

3．図8は、駒の移動による盤面上の駒の配置は示しており、h＝2となっている。各マスに表示されている h の値で最も小さい数値が1となっている。したがって、丸のついているマス（h＝1）を選択し、v_4 行目の駒を4列目に移動する。その結果の盤面の駒の配置が、図9に示される。ここでは、h の最小値が1であることから、山登り探索をおこなっても、h＝1となる。つまり、この問題では、山登り探索法を今後も適用しても極小値が求まることになり、最小値（h＝0）には到

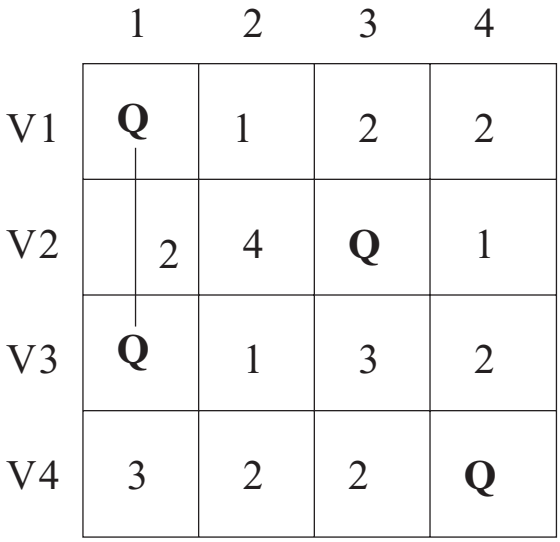

図9　n-クイーン問題（n＝4）：改良（2）

達できない。つまり、解を求めることは出来ない。

3.2.2 制約違反最小化ヒューリスティックスに基づいた山登り探索の適用

4-クイーン問題に対して、この制約違反最小化ヒューリスティックスに基づいた山登り探索を適用した手順を以下に示す。

1．図10は、ランダムにおこなられたすべての変数に対する値の割り当てを示す。この割り当てに対する制約チェックの結果、違反制約に関連する変数は、v_1, v_2, v_3, v_4 となる。ここでは、変数 v_4 を選択する。

2．図11の v_4 行目のそれぞれのマス目には、それぞれのマス目に駒を置いた場合の制約違反の数が

	1	2	3	4
V1	Q			
V2		Q		
V3	Q			
V4		Q		

図10 n-クイーン問題（n= 4）: ランダムな初期値の生成

	1	2	3	4
V1	Q			
V2	3	2	1	(0)
V3	Q			
V4			Q	

図12 n-クイーン問題（n= 4）: 改良（2）

	1	2	3	4
V1	Q			
V2		Q		
V3	Q			
V4	2	2	(0)	2

図11 n-クイーン問題（n= 4）: 改良（1）

	1	2	3	4
V1	1	(0)	3	1
V2				Q
V3	Q			
V4			Q	

図13 n-クイーン問題（n= 4）: 改良（3）

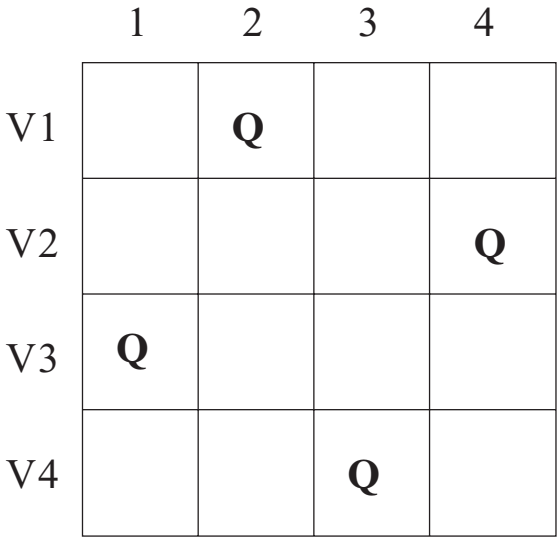

図14　n-クイーン問題（n＝4）：解の生成

示されている。ここでは、制約違反最小化ヒューリスティックスに基づいて、3列目の丸で囲まれた値0を選択する。その結果、得られる盤面上の駒の配置は、図12となる。

3．図12の v_2 行目には、それぞれのマス目に駒を置いた場合の制約違反の数が表示されている。最小の値は0なので、4列目の丸で示されているマス目に駒を移動する。その結果、得られた盤上での駒の配置が図13に示される。

4．図13の v_1 行目には、それぞれのマス目に駒を置いた場合の制約違反の数が表示されている。2列目の丸で示されているマス目の数は最小値0なのでに駒を移動する。その結果、最終的に得られた盤上での駒の配置が図14に示され、制約を満足する解であることがわかる。

4　おわりに

本論文では、制約充足問題の解法である近似解法についてサーベイするとともに、パズルの代表例であるN-クイーン問題を対象として、局所探索法による解法である山登り探索と制約違反最小化ヒューリスティックに基づいた山登り探索の適用結果と有効性について説明した。今後は、大規模な実際的な離散組み合わせ問題（たとえば、スケジューリング問題、プラニング問題など）への適用をおこない、解の精度と処理時間という観点から有効性を評価していく予定である。

参考文献

［1］特集：制約充足問題の基礎と応用、人工知能学会誌、Vol.12, No. 3 （1997）.

［2］西原清一：整合ラベリング問題と応用、情報処理、Vol.31, No.4, pp.500-507（1990）.

［3］西原清一：制約充足問題（CSP）の基礎と動向、1991年度人工知能学会全国大会（第5回）チュートリアル資料B-1（1991）.

［4］Edward Tsang : *Foundations of Constraint Satisfaction*, Computation in Cognitive Science, Academic Press（1993）.

［5］Ian Miguel : *Dynamic Flexible Constraint Satisfaction and its Application to AI Planning*, Distinguished Dissertations, Springer（2003）.

［6］Rina Dechter : *Constraint Processing*, Morgan Kaufmann Publishers（2003）.

［7］Krzysztof R. Apt : *Principles of Constraint Programming*, Cambridge Unversity Press（2003）.

［8］Stuart Russell and Peter Norvig : *Artificial Intelligence A Modern Approach, Second Edition*, Prentice Hall Series in Artificial Intelligence, Pearson Education（2003）.

［9］David Pool, Alan Mackworth, and Randy Goebel : *Computational Intelligence: A Logical Approach*, Oxford Unversity Press（1998）.

［10］Alan Mackworth : Constraint Satisfaction, In *Encyclopedia of Artificial Intelligence ((ed.) Shapiro, S.C.), Vol.1*, pp.205-211, John Wiley & Sons, Inc.（1987）.

［11］Bernard A. Nadel : Representation Selection for Constraint Satisfaction : A Case Study Using n-Queens, In *Proc. of IEEE Expert*, pp.16-23（1990）.

［12］Steven Minton, Mark D. Johnston, Andrew B. Philips. and Philip Laird : Minimizing Conflicts: A Heuristic Method for Constraint Satisfaction and Scheduling Problems, Artificial Intelligence, Vol.58, pp.161-205（1992）.