ニューラルコンピューティングの遊び方 1

Nクイーン問題から未解決幾何問題への旅

武藤佳恭

はじめに

ニューラルコンピューティングの本格的な遊び方（応用）を，できるだけ多くの例題を説明しながら具体的に紹介していきます．しかしながら私のつまらない説明を読むよりも，INSPEC（有料）や Uncover（無料）などのデータベースへアクセスして，直接英語の論文を取り寄せて読むことを推薦したいと思います．また現在最も強力な OpenText などのインターネットサーチエンジンを使って，インターネット上のデータベースから情報探索することを読者諸氏に切に希望します．本格的な遊び方という意味は，例題の中でニューロンの数にして数百万個のニューラルネットワークを用いることもあるからです．

筆者は 1985 年以前には，ニューラルコンピューティングに関してほとんど知りませんでした（現在もそれほど知りませんが…，インターネットを用いればサーベイなどは容易です）．ニューラルコンピューティングと仲良くなったきっかけは，筆者が米国の南カロライナ大学で働いていた 1985～1986 年ごろに，ニューズウィークやタイム誌に広く報道された科学ニュースに出合ったことです．そのニュースとは，“ナメクジやゴキブリなどの小動物の神経回路網を電子回路（シリコンチップ）で実現し，そのシリコンチップを使うといろいろな問題を解くことができる”というものです．その人工神経回路網をニューラルネットワーク（Neural Network）と呼び，それは人間のように学習機能を備え，また組合せ最適化問題をうまく解くことができるというのです．私はそんな馬鹿な!?…まさか!?…と思いましたが，それらの関連論文を取り寄せてニューラルネットワークの研究内容を少しずつ調べ始めました．

ニューラルネットワークの分野にまったく無知であった私は，その論文を読んでも最初のうちはさっぱり理解できなかったのです．何度も何度も論文を読み，学生達とニューラルシミュレータ（コンピュータ上にニューラルネットワークの数学モデルを構築し，その振舞いをシミュレートできるソフトウェアまたはプログラム）を試行錯誤して作りながら，次第にニューラルネットワークのメカニズムがわかり始めました．多くの科学論文には，隠されたきわめて重要な情報があって，それらの情報はたいてい説明されていないか載っていないかのいずれかです（科学者/エンジニアの飯の種）．今のシリーズではニューラルコンピューティングのいくつかの技（わざ）を公開します．

ニューラルネットワークの応用分野は，通信工学（周波数割当て問題，チャネル routing 問題などを紹介する），電子回路設計（routing 問題，planarization 問題など），グラフ理論（4 色/k 色問題，maximum clique 問題，Ramsey 問題など），分子生物学（アライメント問題，RNA/DNA 2 次構造予測問題など），マネジメントサイエンス（facility layout 問題など），OR（operations research：資源配置問題，小選挙区区割り問題など），放射線医学（MRI 問題など），リモートセンシング（各種の clustering 問題な

ど），ゲーム理論（タイリング問題，騎士巡回問題など），信頼性工学（スペア配置問題など）など，広範にわたっています．皆さんが使っている国際電話のエコーキャンセリングはニューラルネットワークの応用分野の一つです．ニューラルネットワークを使う醍醐味は，難しい問題や未解決問題が解けることです．

われわれの脳神経回路網でいったい何が起こっているのかいまだにすべてが解明されたわけではなく，誰もわからないのですが，わかっているほんの少しの事実をさらに単純化し，数学モデルで表現したものがニューラルネットワークです．その計算作法をニューラルコンピューティングと呼びます．その単純な数学モデルの中には，ニューロンという計算要素とそれらを結合する重み付きシナプス結合要素の二つしかありません．複雑なニューラルネットワークの数学モデルの中には遅延要素，フィルタ要素などの多くの要素が発見されてきています．この連載では計算要素のニューロンと重み付きシナプス結合要素の単純な数学モデルしか扱いません．

ニューロンには入力 U と出力 V があり，ニューロン i の入出力関数 f は通常非線形です：$V_i=f(U_i)$．たとえば，$V_i=U_i$ の場合は線形，$V_i=1$ if $U_i\geq 0$ and $V_i=0$ if $U_i<0$ はマッカロー・ピッツ ニューロンと呼ばれ，非線形関数です．

ニューラルネットワークには大きく分けて，学習と最適化のためのモデルがあります．学習と最適化の両方を兼ね備えたモデルは，セルフオーガニゼーションモデルと呼ばれます．組合せ最適化問題を解く場合は固定型のニューラルネットワークを用います．おなかが減ると赤ちゃんは泣くというのは，母親が赤ちゃんに学習させたものではなく，固定型のニューラルネットワークのおかげです．ニューラルネットワークでいう“学習”とは，たかだかシナプス結合の強さを変えることです．学習にはスーパーバイズド（supervised）学習とアンスーパーバイズド学習（セルフオーガニゼーション）の二つがあります．スーパーバイズド学習では教え込む教師が必ず必要ですが，アンスーパーバイズド学習（セルフオーガニゼーション）では特徴抽出を自動的に行なうので教師は必要ありません．セルフオーガニゼーションはイメージ処理にはたいへん適しています．われわれは MRI（magnetic resonance image）に応用して腫瘍やアルツハイマー診断に用いたり，リモートセンシングで農作物予測/環境アセスメントなどの画像処理に利用しています．

ニューラルネットワークというと，スーパーバイズド学習であるという読者諸氏の固定観念を打ち砕くために，“ニューラルコンピューティングの遊び方”の連載を引き受けた次第です．まず今回は固定型のニューラルネットワークを説明するために N クイーン問題を解くニューラルネットワークを簡単に紹介して，そのプログラムを示します．連載記事を待ち切れない読者諸氏はわれわれのサーバをお訪ねください：[http://www.neuro.sfc.keio.ac.jp/]．

ニューラルネットワークのアーキテクチャにはフィードフォワード型とリカレントフィードバック型があります．フィードフォワード型は，論理回路でいうと組合せ回路で内部にはフィードバックをもちません．リカレントフィードバック型は論理回路でいうと順序回路にあたり，内部にはフィードバックをもつため，たいへん複雑な処理が可能です．フィードフォワード型ニューラルネットワークはリカレントフィードバック型のサブセットにすぎません．フィードフォワード型ニューラルネットワークの学習は比較的容易ですが，リカレントフィードバック型ニューラルネットワークの学習は非常に難しいのです．質問は staff@neuro.sfc.keio.ac.jp にお願いします．

Enjoy neural computing!

Nクイーン問題への旅

8クイーン問題は，8×8のチェスボードに8個のクイーンをお互いがぶつかり合わないように置くというゲームです．ここでクイーンは縦横斜めに効きます（図1）．一般に，固定リカレントフィードバック型のニューラルネットワークのシナプス結合を決めるのは，与えられる問題の制約条件です．この場合制約条

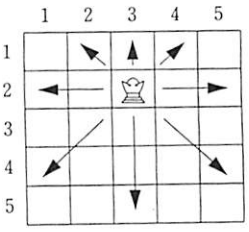


図1　クイーンの効き方

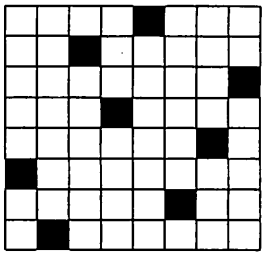


図2　8クイーン問題の解答例

件は，①各行各列に1個のクイーンを置く，②斜めにぶつからないように上手にクイーンを置く，ということです．

8×8のチェスボードには64のます目があるので64個のニューロンを準備します．それぞれのニューロンにはユニークな名前が付けられます：i行j列のニューロンij．ニューロンijの出力が1になるということはi行j列のます目にクイーンを置くことです．一つの答えを図2に示します．

ニューラルネットワークのシナプス結合を決める制約条件を考えると，他のニューロン出力からのニューロンij入力への接続は次の動作方程式で表現されます．

$$\frac{dU_{ij}}{dt}=-\left(\sum_{k=1}^{8}V_{ik}-1\right)-\left(\sum_{k=1}^{8}V_{kj}-1\right)-\sum_{\substack{1\le i-k,j-k\le N\\ k\neq 0}}V_{i-k,j-k}-\sum_{\substack{1\le i-k,j+k\le N\\ k\neq 0}}V_{i-k,j+k}+h\left(\sum_{k=1}^{8}V_{ik}\right)+h\left(\sum_{k=1}^{8}V_{kj}\right)\quad(i,j=1,\cdots,8)$$

慣れていないと，この動作式を見て頭がふらふらするかもしれませんが，じっくり見てください．第1項と第2項は“各行各列に1個のクイーンを置く”という制約条件，第3項と第4項は“斜めにぶつからないように上手にクイーンを置く”という制約条件，第5項と第6項は各行各列に1個もクイーンを置けなかったときの解決策です．第5項と第6項の$h(x)$関数は，$x=0$のとき$h(x)=1$，$x\neq 0$のとき$h(x)=0$となります．$h(x)$関数を私はヒルクライム項と呼んでいます．なぜならばニューラルネットワークの状態がローカルミニマム†に陥ったとき，この項のおかげで脱出できる可能性が出てくるのです．この動作方程式をもう一度じっくり味わってください．64個のニューロンには64本の微分方程式があるので，1次のオイラー法を用いて数値計算すると，簡単に答えが見つかります．1次のオイラー法とは，一番単純な数値計算法で，ニューロンij入力は次の式で毎回変更します．

$$U_{ij}(t+1)=U(t)+\frac{dU_{ij}}{dt}$$

Macintosh上で実行可能なNクイーンのTHINK Cプログラム例をリスト1に示します．このプログラムはDOS上/Windows上あるいはUNIX上でもそのまま動くはずです．読者諸氏はすぐに気がつくと思いますが，プログラムの中ではニューロンの入力値範囲を制限しています．これはニューラルネットワークの解への収束を早くするためです．またヒルクライム項は周期的なインパルス応答の刺激として与えられています．このプログラムの詳しい解説は参考文献1)の*Neural network parallel computing*で述べています．

一般にリカレントフィードバック型ニューラルネットワークの動作方程式は，

$$\frac{dU_i}{dt}=\frac{-dE(V_1,V_2,\cdots,V_n)}{dt}$$

ここで$E(V_1,V_2,\cdots,V_n)$はエネルギー関数と呼ばれますが，i番目のニューロンの時間に対する入力変化量dU_i/dtはあくまでも制約条件によって決定されます．シミュレータをディジタルコンピュータ上で動かす場合は，ニューラルネットワークの状態はn次元のハイパー空間上のある離散点からある離散点へ状態変化します．エネルギー関数の値を時間軸でプロットするとおおまかには減少傾向にありますが，一時的に増加する場合もあり得ます．

未解決幾何問題への旅

“正方形の中にN個の点を置き，それぞれの2点間

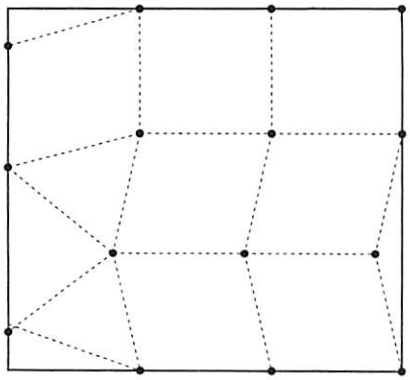
† それぞれのニューロンはニューラルネットワークのシステムの状態がローカルミニマムに陥っているかどうかわからなくてはいけません．この場合制約条件を満たしていないと，システムの状態がローカルミニマムに陥っていると認識し，関係するニューロンを発火させてローカルミニマムから脱出しようと試みます．ヒルクライム項があまり強いと状態が不安定で激しく変わりすぎます．反対にヒルクライム項が弱いとローカルミニマムから脱出できない場合もあります．適度のヒルクライム項はニューラルネットワークではきわめて有効です．非常に深いローカルミニマムからの脱出を考えると，システムの状態初期値を振り直したほうが良い場合があります．

```
#include <stdio.h>
#include <math.h>
short A,B,C,t,hil,i,j,k,sum_column,sum_row,diagonal1,diagonal2,conf,diag,seed,row,length,max;
short U[50][50], V[50][50];

void main ()
{
    A=B=C=1;
    seed=3;
    printf("input the size of the problem\n");
    scanf("%d",&max);

    for(i=1;i<seed;i++)
     {U[1][1]=Random(); }
     for( i=0;i<=max+1;i++) {
      for( j=0;j<=max+1;j++)
      {U[i][j] = -abs(Random()%8);
       V[i][j]=0;
      }}
       /******** Main program ********/
    t=0; diag=1;
    while ((Button()==FALSE) && (diag>0) && (t<100) )
    {
      diag=0;
        for( i=1;i<=max;i++) {
         for( j=1;j<=max;j++)
          {
            sum_column=0;sum_row=0;
            for( k=1;k<=max;k++) {sum_row = sum_row+V[i][k];
            sum_column = sum_column+V[k][j];}
            diagonal1=0; k=1;
            while((j+k<=max) && (i-k>=1))
            {diagonal1=diagonal1+V[i-k][j+k]; k++; }
            k=1;
            while((j-k>=1) && (i+k<=max)) {diagonal1 += +V[i+k][j-k]; k++;}
            diagonal2=0; k=1;
            while((j+k<=max) && (i+k<=max)) {diagonal2 += +V[i+k][j+k]; k++; }
            k=1;
            while((j-k>=1) && (i-k>=1)) {diagonal2 += +V[i-k][j-k]; k++; }
        hil=0;conf=1;
        if (sum_column==0) { hil=1; }
        if (sum_row==0) {hil++;}
        if((sum_column+sum_row==2) && (diagonal1<2) && (diagonal2<2)) {conf=0;}
        U[i][j] += -(sum_row+sum_column-2)-(diagonal1+diagonal2)+C*hil;
        if (U[i][j]<-6) {U[i][j]= -6;}
        if (U[i][j]>0) {V[i][j]=1;}
        if (U[i][j]<-1) {V[i][j]=0;}
        diag=diag+conf;
        }}/******** The end of i and j loop ********/
 for(i=1;i<=max;i++) {
  for(j=1;j<=max;j++)
       {  printf("%d ",V[i][j]);}
        printf("\n");}
       printf("the number of iteration steps=%d\n",t);
       t++; if (t % 10 <4 ) {C=3;} else {C=1; }
  } /******** The end of while loop ********/
  while (Button()==FALSE) {}
}
```

リスト1



d15＝0.338599…	d15＝$(14-3\sqrt{3})/26$
00(0.6613687, 0.0000000)	00($(12+3\sqrt{3})/26$,0)
01(1.0000000, 1.0000000)	01(1,1)
02(0.3227695, 0.0000000)	02($(3\sqrt{3}-1)/13$,0)
03(0.6613772, 0.3386137)	03($(12+3\sqrt{3})/26$,$(14-3\sqrt{3})/26$)
04(0.9272055, 0.6693183)	04($(69-12\sqrt{3})/52$,$(40-3\sqrt{3})/52$)
05(0.3227723, 0.3386062)	05($(3\sqrt{3}-1)/13$,$(14-3\sqrt{3})/26$)
06(0.6614002, 1.0000000)	06($(12+3\sqrt{3})/26$,1)
07(0.3227939, 1.0000000)	07($(3\sqrt{3}-1)/13$,1)
08(0.2500009, 0.6692940)	08(1/4,$(40-3\sqrt{3})/52$)
09(1.0000000, 0.0000000)	09(1,0)
10(1.0000000, 0.3386348)	10(1,$(14-3\sqrt{3})/26$)
11(0.0000000, 0.1023209)	11(0,$(5\sqrt{3}-6)/26$)
12(0.0000000, 0.8977136)	12(0,$(32-5\sqrt{3})/26$)
13(0.0000000, 0.4409218)	13(0,$(8+2\sqrt{3})/26$)
14(0.5886032, 0.6693166)	14($(41-6\sqrt{3})/52$,$(40-3\sqrt{3})/52$)

図3　15点問題の答えと座標

の最小距離を最大にする N 個の座標を求めよ”というのが有名な幾何の未解決問題です．たとえば，小さい問題で9点ぐらいまでは何とか計算できますが，14点，16点，25点を除き10点から27点までの問題ですら未解決問題です．4点問題では四隅に4点を置けば良いわけです．慶應義塾大学環境情報学部生であった藤沢公也君（現在慶應義塾大学政策メディア研究科大学院生）は15個のニューロンを使って15点問題を解き，現在までの世界記録を塗り替えました．これまでの世界記録は $d_{15}=0.337$ でしたが，ニューラルコンピューティング手法は $d_{15}=0.338599$ の新記録を作りました．図3にその答えと15点の座標を示します．なお正方形の一辺の長さは1です．

この問題を解くために，新しいニューロンを提案しています．そのニューロンをわれわれはバルーンネットニューロンと呼んでいます．まず最初に，15個のバルーンネットニューロンを正方形の中心付近に適当に配置します．それぞれのニューロンは時間がたつにつれ，次第次第に膨れながら，お互いにぶつかり合い，押し合いへし合いしながら成長します．これ以上バルーンネットニューロンが膨れられないすし詰め状態まで達すると，任意の2点間の最小距離を最大にする15個の座標を求めたことになります．バルーンネットニューロンは，人工生命とニューラルネットワークの合いの子みたいなものです．詳しくは次号に掲載される予定です．

4色問題への旅

ここでは地図の4色塗問題/k 色塗問題とその応用を紹介します．歴史的背景を調べて見ると，ヨーロッパで地図を製造する職人は地図を塗り分けるために，できるだけ少ない色を用いて隣どうしの領域が同色にならないように工夫していました．ここで言う“二つの領域が隣合せ”とは二つの領域が共通する面をもち，しかも互いに接していることです．1850年ごろ，Francis Guthric はこの問題にたいへん興味をもち，その当時著名な数学者であった Augustus De Morgan（ド・モルガンの定理などで有名）に，この地図の4色塗問題を紹介しました．どのような平面地図も4色あれば塗れるようなのですが，その証明となるとたいへん厄介で，当時から現在に至る著名な数学者がこの問題にチャレンジしても，いまだに証明できていない難しい問題です．

ところが画期的な証明が Kenneth Appel と Wolfgang Haken によって1976年に米国の数学学会で発表されました．数学者が紙と鉛筆で簡単に証明できていないこの問題を，コンピュータを用いてシラミつぶし的に証明してしまったのです．コンピュータによるシラミつぶし的証明の検証も，人間にとってはきわめて難しいことです．いまだにコンピュータによる証明を拒絶する数学者はきわめて多いようです．

Appel と Haken の色塗り手法は $O(n^2)$† の計算時間を要します．ここで n は地図の領域数です．われわ

† $O(n^2)$ とは計算時間量（Order）を表現していて，この場合 n の2乗に比例して計算時間がかかることを意味します．$O(1)$ とは，n に関係なく計算時間が一定であることを意味します．

れは$4n$ 個のニューロンを用いて，実験的にほぼO(1)に抑えることができる4色塗り手法を1991年に提案しました．

図4に9領域からなる地図問題を示します．まず4色のどの色を塗るかを表現するために1000，0100，0010，0001をそれぞれ赤色，黄色，青色，緑色塗りとすると，この問題を解くのに9×4＝36個のニューロンを使えばよいわけです．つまり1領域に対して4個のニューロンを用います．前回説明したように，固定型のニューラルネットワークのシナプス結合を決めるのは問題の制約条件です．この場合制約条件は，

(1)それぞれの領域に必ず4色のうちの1色を割り当てる．

(2)隣合せの領域は同じ色を塗らないようにする．

二つの領域が隣合せかどうか決定してくれるのが図5の隣接行列です．図6は一つのニューラルネットワークの収束状態を示します．図7は図6の状態を4色塗りで示した答えです．

さきほどの二つの制約条件をX領域i色のニューロン動作方程式で示すと次のようになります．

$$\frac{dU_{Xi}}{dt}=-A\left(\sum_{j=1}^{4}V_{Xj}-1\right)-B\sum_{\substack{Y=1\\Y\neq X}}^{n}d_{XY}V_{Yi}$$

最初の項は4色のうちの一つの色を割り当てる，2番目の項は隣合せの領域は同色を割り当てない．

このニューロン動作方程式では不十分で，たとえば図8のケンタッキー州やテネシー州などは，周りの州が4色を使い切ってしまうため塗る色がなくなります．このような状況を軽減するために問題の正規化を必要とします．正規化とは，それぞれの領域が隣接する領域数に関係なく，平等に色の割当てが行なえるよ

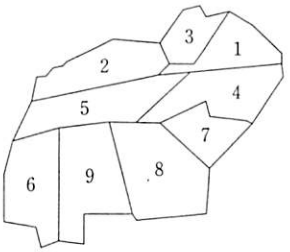


図4　9領域の問題

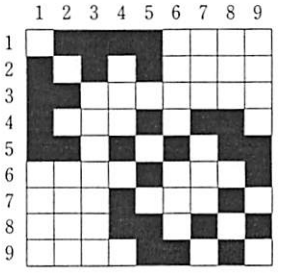


図5　9領域の隣接行列（たとえば領域3は1と2に接する）

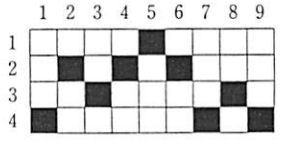


図6　ニューラルネットワークの収束状態

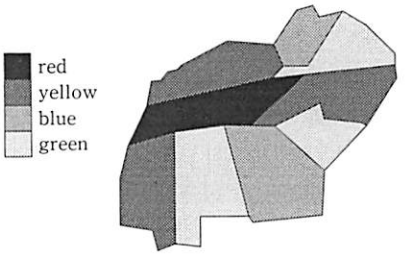


図7　4色塗りの一つの答え

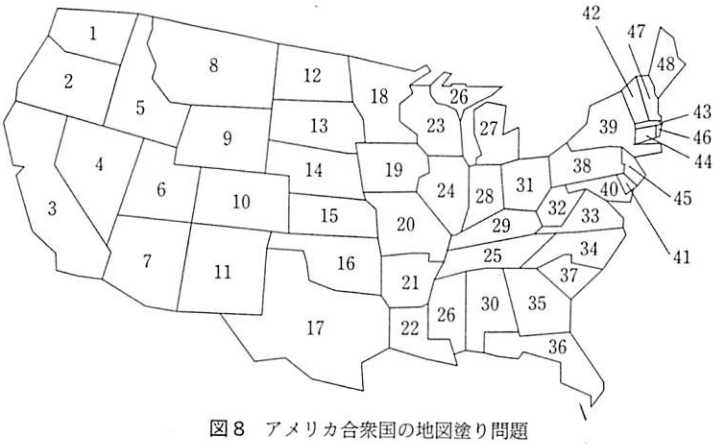


図8　アメリカ合衆国の地図塗り問題

うにする手法です．正規化されたニューロンの動作方程式は次のようになります．

$$\frac{dU_{Xi}}{dt}=-A\left(\sum_{j=1}^{4}V_{Xj}-1\right)-B\sum_{\substack{Y=1\\Y\neq X}}^{n}d_{XY}V_{Yi}\sum_{K=1}^{n}d_{YK}$$

$$+Ch\left(\sum_{j=1}^{4}V_{Xj}\right)\left(C_1\sum_{K=1}^{n}d_{XK}+C_2\frac{\sum_{K=1}^{n}\sum_{Y=1}^{n}d_{XY}d_{YK}}{\sum_{K=1}^{n}d_{XK}}\right)$$

われわれの提案するこの手法は，今のところ世界で一番早い並列アルゴリズムです．

k 色塗り問題への旅

平面地図の3色塗り，非平面グラフの k 色塗りは NP (Nondeterministic Polynomial time) 問題で，一般に NP 問題の答えを見つけるためには多くの時間を要します．もう少し詳しくいうと，NP 問題の計算時間量は指数関数的に爆発します．ニューラルコンピューティングで問題を解く魅力は指数関数的に爆発する NP 問題の計算時間量を非指数関数時間に抑えることができる可能性があることです．

アメリカ合衆国の3色塗りのためのニューロンの動作方程式を下に，その振舞いを図9に示します．

$$\frac{dU_{Xi}}{dt}=-A\left(\sum_{j=1}^{3}V_{Xj}-1\right)-B\sum_{\substack{Y=1\\Y\neq X}}^{n}d_{XY}V_{Yi}\sum_{K=1}^{n}d_{YK}$$

$$+Ch\left(\sum_{j=1}^{3}V_{Xj}\right)\left(C_1\sum_{K=1}^{n}d_{XK}+C_2\frac{\sum_{K=1}^{n}\sum_{Y=1}^{n}d_{XY}d_{YK}}{\sum_{K=1}^{n}d_{XK}}\right)$$

西海岸のカリフォルニア州と東のウエストバージニア州あたりに二つの矛盾が生じ，3色塗りは不可能のようである．

k 色塗りの応用

われわれの提案する k 色塗り手法を実際の問題に応用してみました．最近流行の無線電話の周波数割当てはまさに k 色塗り問題で，周波数が色に相当します．われわれは世界最先端の無線電話の周波数割当てアルゴリズムと比較検討しました．われわれが提案す

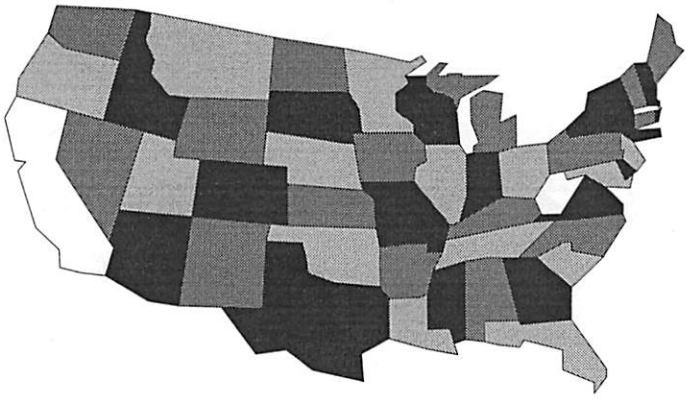


図9　アメリカ合衆国の3色塗り解答

る手法は少ない周波数で正しい答えを見つけることに成功しました．詳しくは“*IEEE Trans. on Vehicular Technology*, Vol. 41, No. 4, pp. 430-437, 1992”を参照してください．別の言葉でいうと，同じ周波数割当て問題の場合，従来手法よりも多くのユーザが同時に電話を利用できるわけです．それでは次号でまた会いましょう．

参考文献

1) Y. Takefuji : *Neural network parallel computing*, Kluwer Academic Pub., 1992.
2) Y. Takefuji, K. C. Lee : Artificial neural networks for four-coloring problems and k-colorability problems, *IEEE Trans. on Circuits and Systems*, 38, 3, pp. 326-333, 1991.
3) Y. Takefuji : Neural network parallelcomputing for combinatorial optimization problems, *Journal of Knowledge Engineering*, 5, 3, pp. 41-61, 1992.
4) N. Funabiki, Y. Takefuji : A neural network parallelalgorithm for channel assignment problems in cellularradio networks, *IEEE Trans. on Vehicular Technology*, 41, 4, pp. 430-437, 1992.
5) [http://www.neuro.sfc.keio.ac.jp/]

（たけふじ よしやす　慶應義塾大学 環境情報学部）
[staff@neuro.sfc.keio.ac.jp]