TR-I-0223

伝送誤りに強い VQ 符号帳構成法に関する検討
A Study on Noise Robust VQ Codebook Generation

杉山 雅英　嵯峨山 茂樹
Masahide SUGIYAMA　Shigeki SAGAYAMA

ABSTRACT

This report describes a preliminary study on noise robust VQ codebook generation. VQ technique is widely used in various speech fields. When VQ codes are encoded and transmitted with bit errors, the decoded VQ codes produce transmission error. This error produces the distortion of speech. The aim of this study is to generate robust VQ codebook. In this report, the problem is formulated and the several preliminary experiments are described. The results show that the generated robust VQ codebook performs 2 or 3dB reduction. This results is very encouraging. In the last part, Neural Networks applied to the optimal solution search algorithm will be explained.

1 はじめに

ベクトル量子化 (VQ) は音声符号化・音声認識において用いられている強力な情報圧縮手法である。伝送誤りの伴う音声伝送系においては VQ 符号 のビット誤りは復号時に歪みを発生する。この歪みをできるだけ小さくするように VQ 符号帳を再設計することを検討する。即ち、ビット表現された VQ 符号に対して確率的に発生するビット誤りに伴う歪みを最小化するように VQ 符号のビット表現を決定する。これは確率と歪みとを同時に考慮した確率歪み量を目的関数とする組合せ問題である。組合せが有限であるので有限回の探索で最適な解が求められるが、符号帳が現実に用いられている程度の大きさの場合その可能な組合せが膨大になるため最適解の探索は容易ではない。ここではその最適解の探索に Hopfield Model などの Neural Network Model を適用することとし、その基礎的な検討を行なう。文献 [1], [2] において同様の動機から検討がなされている。

2 問題の定式化

K からなる ベクトルの集合 $V = \{v_i\}$ は $k = \log_2 K$ ビットで符号化される。この符号の付け方に関して最適化を図ることがこの検討の目的である。

$$C = \{(b_1, b_2, \cdots, b_k) \epsilon R^k | b_i : 0, 1\}$$

を符号の集合とする。以下に示すように V から C への写像を σ とする。

$$\sigma : V \longrightarrow C$$

さらに C におけるビット誤りを表す写像を β とする。

$$\beta : C \longrightarrow C$$

これを復号する写像を τ とする。これは σ の逆写像である。

$$\tau : C \longrightarrow V$$

$$\tau\beta\sigma : V \longrightarrow C \longrightarrow C \longrightarrow V$$

図 1に示すように $v_i \epsilon V$ は $v_j = \tau\beta\sigma(v_i)$ に写像されることになるので、$d(v_i, v_j)$ だけ誤差 (歪み) が生じることになる。図 2に示すように、この 1 ビットの誤りが生ずる確率を p で表すことにする。ここではビット誤りについて検討するので誤りは C の上でおこるものとする。さらに簡単化して各位のビットは等確率で起こるものとする。各位のビットが異なる確率で起こる場合については後で述べることにする。そこで符号の 1 ビット誤りに伴う誤差 (歪み) を以下のように定義する。

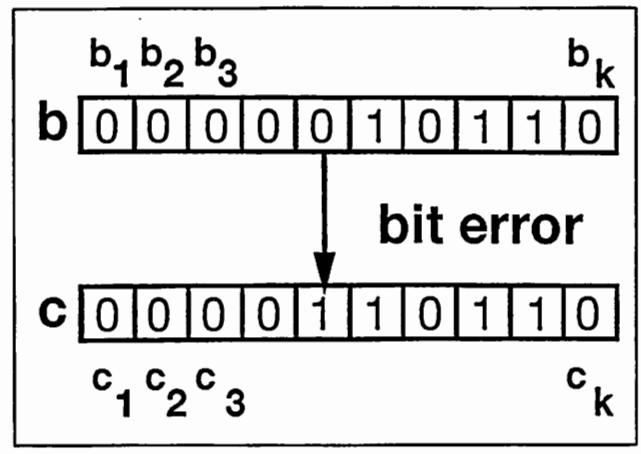

図 2: bit error

$$D_{p,d}(\sigma) = \frac{p}{K} \sum_{i=1}^{K} d(v_i, v_j) \tag{1}$$

これを以下のように書く直すことができる。

$$\frac{p}{K} \sum_{v} d(v, u)$$

ただし、$u = \tau\beta\sigma(v)$ であり、$\sigma(u) = \beta\sigma(v)$ であるので、h をビット表現間の hamming 距離とすると

$$h(\sigma(v), \sigma(u)) = 1$$

となる。

$$u = \tau\beta\sigma(v) \iff h(\sigma(u), \sigma(v)) = 1$$

即ち、以下のようになる。

$$D_{p,d}(\sigma) = \frac{p}{K} \sum_{h(\sigma(v),\sigma(u))=1} d(v, u)$$

さらに一般化して β が n ビット誤る写像である場合は

$$h(\sigma(v), \sigma(u)) = n$$

であるので、下のように定義する。

$$D_{p,d}(\sigma) = \sum_{n=1}^{k} \frac{p^n}{N_n} \sum_{h(\sigma(v),\sigma(u))=n} d(v, u) \tag{2}$$

N_n は $h(\sigma(v), \sigma(u)) = n$ を満たすベクトルの個数である。$N_1 = K, N_k = 1$ である[1]。集合 V は K 個の要素から成り立っているので、$d(v_i, v_j)$ は距離行列としてあらかじめ計算しておけば良い。従って、この時最適化問題を以下のように定式化できる。

[1] 一般に N_n を決定する問題があるが、まだその解を得ていない。

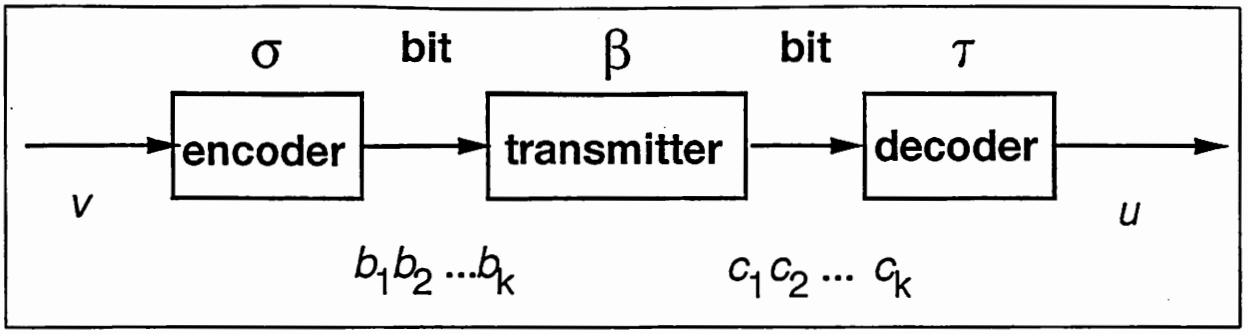

図 1: 符号・復号系と伝送系について

K からなるベクトルの集合 $V = \{v_i\}$ とその距離行列 $d(v_i, v_j)$ と確率 p が与えられた時、V を k ビットで表現する写像 σ に関して $D_{p,d}(\sigma)$ を最小化する σ を決定すること。

$$\sigma_{MIN} = arg \min_{\sigma} D_{p,d}(\sigma) \tag{3}$$

さらに上の最小値・最大値、その比を求めること。

$$r = 10 \log_{10} \frac{\max_{\sigma} D_{p,d}(\sigma)}{\min_{\sigma} D_{p,d}(\sigma)} \tag{4}$$

距離行列をあらかじめ与えておくので一般化して d^{α} に関しての最適化を扱うことも同様である。以下では $\alpha = 2$ (即ち、自乗距離) について検討している。式(1) から分かるように 1 ビット誤りの場合は $D_{p,d}$ の最小化は p には依存しないので、d のみで決定されることになる。図3にこの概念図を示す。VQ 符号帳が作成された後、その番号を入れ変えることにより評価関数を最適化し、それを用いて伝送することになる。

評価関数 $D_{p,d}$ をさらに一般化して、ビット表現中の列のどのビットで誤るかに応じた確率を指定する、もしくは重み付けするなどの方法も考えられる。後者の方法の場合はそのビットへの重み関数を w として、以下のように評価関数を定式化できる。

$$D_{p,d,w}(\sigma) = \sum_{n=1}^{k} \frac{p^n}{N_n} \sum_{h(\sigma(v),\sigma(u))=n} w(\sigma(u),\sigma(v)) d(v,u) \tag{5}$$

3 組合せ最適化問題の探索方法について

3.1 全数探索の方法

要素の数を K とすると、σ は $K!$ 通りある。この最適化問題は有限個の組合せの中から選ぶ組合せ問題となり、全数探索を行なえば最適な σ を決定できる。しかしながら、符号帳の大きさ K が大きくなるに連れ、組合せの数 $K!$ が膨大になり、その探索問題を解くことは現在の計算機の処理能力を用いても容易ではない。以下に符号帳の大きさと組合せの個数との関係を示す。

表 1: 符号帳の大きさと組合せの個数との関係

ビット (k)	K	組合せ
1	2	2.0000e+00
2	4	2.4000e+01
3	8	4.0320e+04
4	16	2.0923e+13
5	32	2.6313e+35
6	64	1.2689e+89
7	128	3.8562e+215
8	256	infinity (e506-509)

$$\sigma_1 = (1\ 2\ 3 \ldots K-1\ K)$$

$$\sigma_2 = (1\ 2\ 3 \ldots K\ K-1)$$

のように下位ビットから置き換えて生成していく。従って、例えば 128! 回以下の探索では上位の 128 個の要素は変化しないことになる。128! ですら十分大きな探索量であるので、結果的には下位の要素のみ変化させることしかできない。

3.2 複数の初期写像を用いた探索方法

下位のビットから順に探索する全数探索を行なう場合は、例えば 256 個からなる符号帳における探索においては 128! 回探索を行なっても上位の 128 ビットは変わらず、下位の 128 ビットのみが変化するだけである。

初期の写像を恒等写像で与えているが、探索範囲が大きくなるとその初期写像から生成できる写像は写像全体の中で限定された範囲になる。そこで、複数の初期写像を用いることにする。

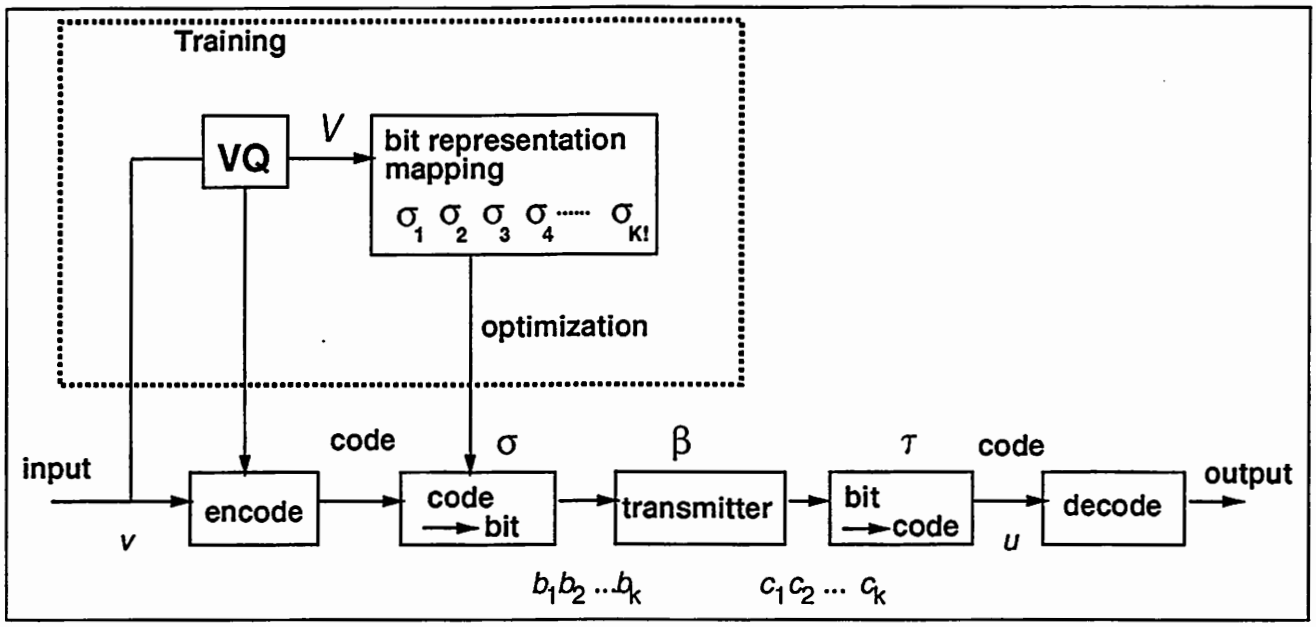

図 3: RVQ の概念図

3.2.1 原始根を用いる方法

簡単に初期写像を生成する方法として素数の原始根を用いることが考えられる。原始根 a とは素数 q を法として

$$1, a^1, a^2, \ldots, a^{q-2}$$

が $1, 2, \ldots, q-1$ を生成できる数のことであり、q が素数の場合はその存在が保証されている。逆に存在するのは素数に限られる。従って、$K = 2^k$ の場合は K より大きい素数を用いて容易に写像が生成できる。以下に要素の数 K とそれより大きな最小の素数、及びその原始根を示す。$K = 256$ の場合は $q = 257$ [2]となり、原始根は $a = 3$ で与えられる。従って、

$$a_n = a^n$$

として、写像は以下のようになる。ただし、n は $q-1$ と互いに素であることが必要十分である。$q = 257$ の場合 $q - 1 = 256 = 2^8$ であるので、n は 2 を因数として持たなければよい。従って、n は奇数であれば良いことになる。これより、初期の写像は 128 種類となり、これらを容易に作成することが可能である。

$$\sigma_n = (a_n, a_n^2, a_n^3, \ldots, a_n^{q-1})$$

ここで $a_n^i = 256$ となる場合、$a_n^i = 0$ と置き換えることにする。これから、より効率的に準最適な解を探索でき、この例においては約 2.87 db 程度の改善が得られる。

[2] $K = 257$ は Fermat 数 (2^{2^t}) である。

表 2: 要素数とそれに近い素数およびその原始根

要素数	素数	原始根
4	5	2,3
8	11	2,6,7,8
16	17	3,5,6,10,11,12,14
32	37	2,5,13,15,17,18,19,20,22,24,32,35
64	67	2,7,11,12,13,18,... , 63
128	131	2,6,8,10,14,17,..., 128
256	257	3,5,6,7,10,12,14,..., 254

3.2.2 $GF[2^k]$ を用いる方法

F を標数 q の体とし、$Z_q = Z/qZ$ を素数 q を法とする整数の商体とする。このとき、F は Z_q を係数とする線形空間と同一視できる。線形空間としての次数 $n = [F : Z_q]$ を用いて、F の要素の数が q^n となることが知られている。この F は Z_q の既約多項式の根 ξ を用いて

$$Z_q[\xi] = \{a_0 + a_1\xi + a_2\xi^2 + \ldots + a_{n-1}\xi^{n-1} | a_i \in Z_q\}$$

と表される。即ち、特に $q = 2$ とすると任意の k に対して、$F = GF[2^k]$ が存在することになる。この集合とその要素を用いて、$\sigma : F \to F$ なる写像が $K = 2^k$ 個、作成できる。

$$\sigma_i : a \to ia$$

この σ_i は 1 対 1 かつ上への写像であり、すべて異なっている。F と C とを同一視することにより、C の上での写

像が構成できることになる。これから、より効率的に準最適な解を探索できる。

3.3 ランダム探索の algorithm

現在以下の方法を検討中である。

1. 1 から K までの中から任意の 2 数 a, b をランダムに選ぶ。
2. 並びのうちで a 番目及び b 番目を入れ変える。
3. これを写像として評価関数（歪み）を算出する

3.4 $K!$ の大きさについて

$$s(n) = \sum_{k=1}^{n} \log k = \log n!$$

と $\log x$ の積分との比較を行なうことで以下の不等式を得られる。

$$t(n) = n(\log n - 1) + 1 = \int_1^n \log x dx$$

$$t(n) < s(n) = \log n! < t(n+1)$$

$$t(n) < \log n! = \log_{10} n! / \log_{10} e < t(n+1)$$

表1の $K = 256$ の場合は $t(256), t(257)$ を用いて算出した。

4 VQ 符号帳を用いた実験

256 のベクトルからなる VQ 符号帳による距離行列を用いて、生成した σ に対するビット誤りにより生ずる確率歪みを算出する。実験条件を以下の表 3に示す。VQ 符号帳は 1 名の男性話者の発声した音素バランス 216 単語（約 10000 サンプル）から 256 個の符号を作成した。さらにこの符号相互の距離を WLR 尺度で算出した。WLR 尺度は対称であるので、距離行列は対称になっている。また通常認識においては WLR 尺度の自乗で歪みの累積などを行なっているので、ここでも自乗距離を用いている。VQ 符号帳の距離行列の一部分及び写像番号とその際の確率歪みの値を以下に示す。この写像番号は恒等写像から出発して下位のビットから変形するものである。

表 3: 実験条件

no of bits	8
VQ size	256
no of errors	1
probability	0.10
LPC order	12
cepstrum order	16
codebook file	VQ/mau_216_256_wlr.book

VQ codebook の距離行列 (256 codes, WLR)

```
0.00 0.14 0.19 0.25 0.31 0.29 0.11 0.47 0.58 0.78
0.14 0.00 0.24 0.26 0.22 0.26 0.12 0.28 0.35 0.58
0.19 0.24 0.00 0.16 0.13 0.25 0.16 0.39 0.62 0.71
0.25 0.26 0.16 0.00 0.27 0.30 0.30 0.47 0.43 0.39
0.31 0.22 0.13 0.27 0.00 0.08 0.11 0.10 0.35 0.49
0.29 0.26 0.25 0.30 0.08 0.00 0.09 0.07 0.32 0.43
0.11 0.12 0.16 0.30 0.11 0.09 0.00 0.15 0.41 0.61
0.47 0.28 0.39 0.47 0.10 0.07 0.15 0.00 0.32 0.48
0.58 0.35 0.62 0.43 0.35 0.32 0.41 0.32 0.00 0.11
0.78 0.58 0.71 0.39 0.49 0.43 0.61 0.48 0.11 0.00
```

写像番号とその際の確率歪みの値

```
nc = 0,  dist=0.657446
nc = 1,  dist=0.657608
nc = 2,  dist=0.657590
nc = 3,  dist=0.658091
nc = 4,  dist=0.657903
nc = 5,  dist=0.658241
nc = 6,  dist=0.657622
nc = 7,  dist=0.657552
nc = 8,  dist=0.658352
nc = 9,  dist=0.658240
nc = 10, dist=0.657977
```

4.1 符号帳の大きさと確率歪み減少量との関係

符号帳の大きさと確率歪み減少量との関係について述べる。符号帳が小さい場合は組合せの数が小さいので探索に必要な処理時間は小さくなる。また、最大・最小歪みの差は大きい。それに対して符号帳が大きくなるにつれて探索量が増大し、最大・最小歪みの差は減少している。改善効果も減少の傾向である。またこの結果は符号帳が大きくなるにつれて探索範囲が大きくなり、改善効果が小さくなっている。

表 4: 符号帳の大きさと確率歪み減少量との関係 (WLR 尺度)

符号帳の大きさ	初期歪み	最大歪み	最小歪み	改善率 (dB)	探索量
8	0.348774	0.564783	0.328717	2.63	40000*
16	0.424642	0.785846	0.406479	2.86	17144000
32	0.471432	0.692299	0.468178	1.70	1655500
64	0.538096	0.703748	0.538096	1.17	4625500
128	0.579360	0.688416	0.579360	0.75	1348500
256	0.641119	0.708837	0.641119	0.44	202000
256	0.657446	1.260437	0.655997	2.84	622500

*: 全数探索が終了

ここで改善率は以下で定義した歪みの比（最大歪みと最小歪みの比）を dB 単位で計算している。

$$10(\log_{10} D_{max} - \log_{10} D_{min})$$

ここで定義した改善率は符号誤りが 1 つの場合には誤りの発生確率には依存していない。符号誤りの発生確率に依存せずに高い改善率が得られることが示されている。

符号帳の大きさが 8 の場合のみ、全数探索が終了している。8 以上の場合は探索範囲が大きく、終了していない。また、表の最下の段の 256 の項目は複数の初期写像から一定の回数探索を行なった場合について示している。一つの初期写像から大きな回数探索を行なうよりも、複数の初期写像から探索を開始した方が容易に準最適な解を探索できることが分かる。

ランダム探索を用いた場合の確率歪み減少量を以下の表に示す。この場合探索回数はすべての大きさの符号帳に対して 20000 回とした。この結果から全数探索に基づく部分探索や複数の初期写像からの部分探索に比べて少ない探索量で良好な結果が得られることが分かる。

表 5: ランダム探索を用いた確率歪み減少量 (WLR 尺度)

符号帳の大きさ	改善率 (dB)	探索量
8	2.350591	200000
16	2.745822	200000
32	2.290091	200000
64	2.616857	200000
128	2.803688	200000
256	2.978840	200000

5 確率歪みの値に関する考察

1 ビット誤りが生じる時、符号が k ビットで表されているならその各位のビット誤りによって異なる v_j に復号化されるので各 v_i に対して、k 個の v_j が発生するすることになる。また、ビット 誤りによってはもとのビットパターンには一致しないので、v_i に行き先は $K-1$ 個の中から k 個選ぶことになる。即ち、${}_{K-1}C_k$ となる。さらに、v_i は K 通りあり得るので、それらのとり得る組合せの数は $K\,{}_{K-1}C_k$ となる。従って、$k=8$ の場合は、

$$K\,{}_{K-1}C_k = \frac{256\ 255\ 254\ 253\ 252\ 251\ 250\ 249\ 248}{8!}$$

6 NN を用いた最適解探索問題の解法について

6.1 ニューラルネットによる探索法の歴史概観

1. Hopfield & Tank の研究
2. Wilson の反例
3. Gaussian Machine, Boltzmann Machine, Elastic Machine
4. Carstern による benchmark test

6.2 ホップフィールドモデルによる探索法

ある種の最適解の探索問題に対して、ホップフィールドモデルやボルツマンマシンなどのニューラルネットワークモデルの応用が提案されている。従って、探索問題の 1 つの解法としてニューラルネットワークよる最適解の効率的な方法を用いることも考えられる。以下ではホップフィールドモデルに限定してその解法について述べる。

図 4に示すようなホップフィールドモデル

$$H = (v_i(t), w_{ij}, \theta_i)$$

は以下のように定式化される。N 個のユニットから構成されるネットワークにおいて時刻 t における i 番目のユニッ

トの出力を $v_i(t)$ で表す。これは 1 または 0 の値を取るものとする。この時確率的に（非同期的に）選ばれたユニット i が他の $N-1$ 個のユニットから受ける信号の総和 $u_i(t)$ を以下のように定義する。

$$u_i(t) = \sum_{j=1, j\neq i}^{N} w_{ij} v_j(t) + \theta_i$$

ここで、w_{ij} はユニット i, j の間の相互結合係数であり、θ_i はユニット i における敷居値である。これらは事前に決定されているものとする。この時、時刻 $t+1$ におけるユニット i の出力値 $v_i(t+1)$ を次のように定義する。

$$v_i(t+1) = \begin{cases} 1 & u_i(t) \geq 0 \\ 0 & u_i(t) < 0 \end{cases}$$

ここで i 以外のユニットの出力は変化しないものとする。

$$v_j(t+1) = v_j(t) \quad (j \neq i)$$

このように非同期的にネットワークを動作させる時、以下の 2 次形式で定義されるネットワークのエネルギー関数が減少することが証明される。

$$E(t) = -\frac{1}{2} \sum_{i \neq j} w_{ij} v_i(t) v_j(t) - \sum_{i=1}^{N} \theta_i v_i(t)$$

このホップフィールドモデルを適用するために以下のように符号とそのビット表現との間の行列表現を用いる。これらの考え方は巡回セールスマン問題にモデルを当てはめる場合を参考にしている。

行列において行に符号の番号、列にビット表現の番号を取る。即ち、符号 x が i 番目のビット表現で表される時、その x, i 行列要素を 1, 対応のない要素の時、0 とする。符号は必ずどれかのビット表現で表されなければならないので、各行、各列ともに 1 がただ 1 回だけ出現することになる。また、このような条件を満たす行列が与えられる時、1 つのビット表現の写像が決まるので、その時の歪み $D_{p,\sigma}$ が決定されることになる。そこで、これらの関係を表す K^2 個のユニットを持つホップフィールドモデル

$$H = (v_{xi}, w_{xi,yj}, \theta_{xi}) \quad (1 \leq x, i, y, j \leq K)$$

を導入する。それぞれの記号の意味は前に定義したものと同様である。さらに 2 次形式で定義されるエネルギー関数を定義する必要がある。

$$E_1 = \frac{1}{2} D \sum_x \sum_{y \neq x} \sum_i$$

$$E_2 = \frac{A}{2} \sum_x \sum_i \sum_{i \neq j} v_{xi} v_{xj} + \frac{B}{2} \sum_i \sum_x \sum_{y \neq x} v_{xi} v_{yi}$$

$$+ \frac{C}{2} \left(\sum_x \sum_i v_{xi} - K \right)^2$$

第 1 の式は巡回セールスマンの問題とは異なり、確率歪みに対応するするように式を作ることができない。

第 2 の式で第 1 項は 1 つの行 x で 2 つの列 i, j のユニットの出力値が同時に 1 の時、エネルギーを A だけ大きくする項である。第 2 項は行列の 1 つの列 i で 2 つの行 x, y の出力値が同時に 1 の時、エネルギーを B だけ大きくする項である。さらに、第 3 項は行列のすべての要素の和が K からずれるとき、エネルギーを C に比例して大きくする項である。この E_1, E_2 の和をネットワークのエネルギーとすれば、ネットワークのエネルギーが最小の時、歪み $D_{p,\sigma}$ が最小となり、最適なビット表現写像が得られたことになる。従って、第 1 の式が定式化されれば、ネットワークの相互結合係数および敷居値を与え、確率的にネットワークを更新することにより、最適な写像 σ を作成できることになる。

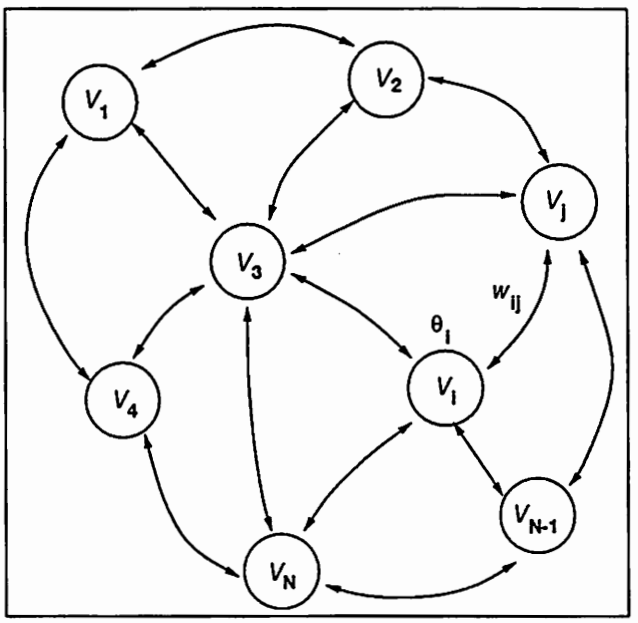

図 4: Hopfield Model の概念図

7 むすび

Universal 符号帳（共通符号帳）や segment based 符号帳を扱う場合は符号帳の大きさは 1024 − 2048 とさらに大きくなり探索の困難さが増大する。今後はより効率的な探索方法の開発が必要である。また、探索に用いられる探索問題の性質（解析的な性質）をより明らかにし、不必要な探索を減らす工夫が必要である。

今後の課題を以下にまとめる。

1. N_n の決定
2. 確率歪みの分布関数の決定
3. NN を用いた探索の解法
4. Travelling Salesman Problem (TSP) との関係

参考文献

[1] 熊沢, 笠原, 滑川, 通信路誤りを考慮したベクトル量子化器の構成, 電子通信学会論文誌, J67-B, pp.1-8 (1984).

[2] Nariman Farvardin, A Studay of Vector Quantization for Noisy Channels, UMIACS-TR-88-17 (Feb. 1988).

[3] J.Hopfield, D.Tank, Neural Computations of Decisions in Optimization Problems, Biol. Cybern, 52, pp.141-152, 1985.

[4] G.Wilson, G.Pawley, On the Stability of the Travelling Salesman Problem algorithm of Hopfield and Tank, Biol. Cybern. 58, pp.63-70, 1988.

[5] 中野, ニューロコンピュータ, 技術評論社, 1990.

[6] Carsten Peterson, Parallel Distributed Approarches to Combinatorial Optimization: Benchmark studies on Traveling Salesman Problem, Neural Computation, 2, 261-269 (1990).

[7] 平山, SIMD 方式並列計算機による神経回路モデルの実現, 電子情報通信学会, CPSY90-59 (1990-07).

[8] 秋山, ガウシアンマシンとその応用, 第9回神経情報科学特選講座, ニューロモデルの新しい流れ, (1990-09).

[9] 高木, 初等整数論講義, 共立出版, 1972.

A 簡単な例

VQ コードが 4 つでその距離行列が対称・非対称の場合について考察する。

例 1.

```
0.00 0.10 0.10 0.30
0.10 0.00 0.01 0.40
0.10 0.01 0.00 0.40
0.30 0.40 0.40 0.00
```

以下の条件で実験を行なった。

```
no_of_bits      = 2
max_number      = 4
no_of_errors    = 1
probability     = 0.20
```

結果は以下のようになった。

(0 1 2 3) --> (0 1 2 3) の場合

```
( 0 0 ) 0-->[0-->1]-->1(0.10)  0-->[0-->2]-->2(0.10)
( 1 0 ) 1-->[1-->0]-->0(0.10)  1-->[1-->3]-->3(0.40)
( 0 1 ) 2-->[2-->3]-->3(0.40)  2-->[2-->0]-->0(0.10)
( 1 1 ) 3-->[3-->2]-->2(0.40)  3-->[3-->1]-->1(0.40)
  distance = 0.400000
```

(0 1 2 3) --> (0 1 3 2) の場合

```
( 0 0 ) 0-->[0-->1]-->1(0.10)  0-->[0-->2]-->3(0.30)
( 1 0 ) 1-->[1-->0]-->0(0.10)  1-->[1-->3]-->2(0.01)
( 1 1 ) 2-->[3-->2]-->3(0.40)  2-->[3-->1]-->1(0.01)
( 0 1 ) 3-->[2-->3]-->2(0.40)  3-->[2-->0]-->0(0.30)
  distance = 0.324000
```

上の表は次のようにみる。

```
        符号  誤り 復号  歪み
( 0 0 ) 0-->[0-->1]-->1(0.10)  0-->[0-->2]-->2(0.10)
```

符号 0 は 0 (00) に符号化され、bit error のため 1 (10) に誤り、これが 1 に復号化されたので、歪み 0.10 が発生した。

符号の置換えを行なって確率歪みが小さくなる場合があるということである。

この組合せの場合は 4 ! 個数だけ組合せがあり、確率歪み (PD) の分布は以下の

PD	histogram
0.48	16
0.44	8

ようになっている。

```
PD histogram
0.4  8
0.324 16
```

例 2. (非対称の場合)

```
 0.00 0.10 0.20 0.30
 0.10 0.00 0.20 0.30
 0.40 0.20 0.00 0.50
 0.40 0.20 0.60 0.00
```

結果は以下のようになった。

(0 1 2 3) --> (0 1 2 3) の場合

```
( 0 0 ) 0-->[0-->1]-->1(0.10)  0-->[0-->2]-->2(0.20)
( 1 0 ) 1-->[1-->0]-->0(0.10)  1-->[1-->3]-->3(0.30)
( 0 1 ) 2-->[2-->3]-->3(0.50)  2-->[2-->0]-->0(0.40)
( 1 1 ) 3-->[3-->2]-->2(0.60)  3-->[3-->1]-->1(0.20)
  distance = 0.480000
```

(0 1 2 3) --> (0 3 1 2) の場合

```
( 0 0 ) 0-->[0-->1]-->2(0.20)  0-->[0-->2]-->3(0.30)
( 1 1 ) 1-->[3-->2]-->3(0.30)  1-->[3-->1]-->2(0.20)
( 1 0 ) 2-->[1-->0]-->0(0.40)  2-->[1-->3]-->1(0.20)
( 0 1 ) 3-->[2-->3]-->1(0.20)  3-->[2-->0]-->0(0.40)
  distance = 0.440000
```

確率歪み (PD) の分布は以下のようになっている。
これらの例から次のような問題が提出される。

問題
(1) 与えられた距離行列に従ってすべての順列を生成した場合の取り得る値の種類 (分布) とその頻度に関して性質を述べよ。(2) 距離行列が対象・非対称形の場合について特にその性質が異なるか?

B 実験例

例 1 の結果

```
 no_of_bits       = 2
 max_number       = 4
 no_of_errors     = 1
 probability      = 0.20
 distance matrix = dm-symetric


 0.00 0.10 0.10 0.30
 0.10 0.00 0.01 0.40
 0.10 0.01 0.00 0.40
 0.30 0.40 0.40 0.00
------ nc=0
( 0 0 ) 0-->[0-->1]-->1(0.10)  0-->[0-->2]-->2(0.10)
( 1 0 ) 1-->[1-->0]-->0(0.10)  1-->[1-->3]-->3(0.40)
( 0 1 ) 2-->[2-->3]-->3(0.40)  2-->[2-->0]-->0(0.10)
( 1 1 ) 3-->[3-->2]-->2(0.40)  3-->[3-->1]-->1(0.40)
  distance = 0.400000

------ nc=1
( 0 0 ) 0-->[0-->1]-->1(0.10)  0-->[0-->2]-->3(0.30)
( 1 0 ) 1-->[1-->0]-->0(0.10)  1-->[1-->3]-->2(0.01)
( 1 1 ) 2-->[3-->2]-->3(0.40)  2-->[3-->1]-->1(0.01)
( 0 1 ) 3-->[2-->3]-->2(0.40)  3-->[2-->0]-->0(0.30)
  distance = 0.324000

------ nc=2
( 0 0 ) 0-->[0-->1]-->2(0.10)  0-->[0-->2]-->1(0.10)
( 0 1 ) 1-->[2-->3]-->3(0.40)  1-->[2-->0]-->0(0.10)
( 1 0 ) 2-->[1-->0]-->0(0.10)  2-->[1-->3]-->3(0.40)
( 1 1 ) 3-->[3-->2]-->1(0.40)  3-->[3-->1]-->2(0.40)
  distance = 0.400000

------ nc=3
( 0 0 ) 0-->[0-->1]-->3(0.30)  0-->[0-->2]-->1(0.10)
( 0 1 ) 1-->[2-->3]-->2(0.01)  1-->[2-->0]-->0(0.10)
( 1 1 ) 2-->[3-->2]-->1(0.01)  2-->[3-->1]-->3(0.40)
( 1 0 ) 3-->[1-->0]-->0(0.30)  3-->[1-->3]-->2(0.40)
  distance = 0.324000

------ nc=4
( 0 0 ) 0-->[0-->1]-->2(0.10)  0-->[0-->2]-->3(0.30)
( 1 1 ) 1-->[3-->2]-->3(0.40)  1-->[3-->1]-->2(0.01)
( 1 0 ) 2-->[1-->0]-->0(0.10)  2-->[1-->3]-->1(0.01)
( 0 1 ) 3-->[2-->3]-->1(0.40)  3-->[2-->0]-->0(0.30)
  distance = 0.324000
```

```
------ nc=5
( 0 0 ) 0-->[0-->1]-->3(0.30)  0-->[0-->2]-->2(0.10)
( 1 1 ) 1-->[3-->2]-->2(0.01)  1-->[3-->1]-->3(0.40)
( 0 1 ) 2-->[2-->3]-->1(0.01)  2-->[2-->0]-->0(0.10)
( 1 0 ) 3-->[1-->0]-->0(0.30)  3-->[1-->3]-->1(0.40)
  distance = 0.324000

------ nc=6
( 1 0 ) 0-->[1-->0]-->1(0.10)  0-->[1-->3]-->3(0.30)
( 0 0 ) 1-->[0-->1]-->0(0.10)  1-->[0-->2]-->2(0.01)
( 0 1 ) 2-->[2-->3]-->3(0.40)  2-->[2-->0]-->1(0.01)
( 1 1 ) 3-->[3-->2]-->2(0.40)  3-->[3-->1]-->0(0.30)
  distance = 0.324000

------ nc=7
( 1 0 ) 0-->[1-->0]-->1(0.10)  0-->[1-->3]-->2(0.10)
( 0 0 ) 1-->[0-->1]-->0(0.10)  1-->[0-->2]-->3(0.40)
( 1 1 ) 2-->[3-->2]-->3(0.40)  2-->[3-->1]-->0(0.10)
( 0 1 ) 3-->[2-->3]-->2(0.40)  3-->[2-->0]-->1(0.40)
  distance = 0.400000

------ nc=8
( 1 0 ) 0-->[1-->0]-->2(0.10)  0-->[1-->3]-->3(0.30)
( 0 1 ) 1-->[2-->3]-->3(0.40)  1-->[2-->0]-->2(0.01)
( 0 0 ) 2-->[0-->1]-->0(0.10)  2-->[0-->2]-->1(0.01)
( 1 1 ) 3-->[3-->2]-->1(0.40)  3-->[3-->1]-->0(0.30)
  distance = 0.324000

------ nc=9
( 1 0 ) 0-->[1-->0]-->3(0.30)  0-->[1-->3]-->2(0.10)
( 0 1 ) 1-->[2-->3]-->2(0.01)  1-->[2-->0]-->3(0.40)
( 1 1 ) 2-->[3-->2]-->1(0.01)  2-->[3-->1]-->0(0.10)
( 0 0 ) 3-->[0-->1]-->0(0.30)  3-->[0-->2]-->1(0.40)
  distance = 0.324000

------ nc=10
( 1 0 ) 0-->[1-->0]-->2(0.10)  0-->[1-->3]-->1(0.10)
( 1 1 ) 1-->[3-->2]-->3(0.40)  1-->[3-->1]-->0(0.10)
( 0 0 ) 2-->[0-->1]-->0(0.10)  2-->[0-->2]-->3(0.40)
( 0 1 ) 3-->[2-->3]-->1(0.40)  3-->[2-->0]-->2(0.40)
  distance = 0.400000

------ nc=11
( 1 0 ) 0-->[1-->0]-->3(0.30)  0-->[1-->3]-->1(0.10)
( 1 1 ) 1-->[3-->2]-->2(0.01)  1-->[3-->1]-->0(0.10)
( 0 1 ) 2-->[2-->3]-->1(0.01)  2-->[2-->0]-->3(0.40)
( 0 0 ) 3-->[0-->1]-->0(0.30)  3-->[0-->2]-->2(0.40)
  distance = 0.324000
```

```
------ nc=12
( 0 1 ) 0-->[2-->3]-->3(0.30)  0-->[2-->0]-->1(0.10)
( 0 0 ) 1-->[0-->1]-->2(0.01)  1-->[0-->2]-->0(0.10)
( 1 0 ) 2-->[1-->0]-->1(0.01)  2-->[1-->3]-->3(0.40)
( 1 1 ) 3-->[3-->2]-->0(0.30)  3-->[3-->1]-->2(0.40)
  distance = 0.324000

------ nc=13
( 0 1 ) 0-->[2-->3]-->2(0.10)  0-->[2-->0]-->1(0.10)
( 0 0 ) 1-->[0-->1]-->3(0.40)  1-->[0-->2]-->0(0.10)
( 1 1 ) 2-->[3-->2]-->0(0.10)  2-->[3-->1]-->3(0.40)
( 1 0 ) 3-->[1-->0]-->1(0.40)  3-->[1-->3]-->2(0.40)
  distance = 0.400000

------ nc=14
( 0 1 ) 0-->[2-->3]-->3(0.30)  0-->[2-->0]-->2(0.10)
( 1 0 ) 1-->[1-->0]-->2(0.01)  1-->[1-->3]-->3(0.40)
( 0 0 ) 2-->[0-->1]-->1(0.01)  2-->[0-->2]-->0(0.10)
( 1 1 ) 3-->[3-->2]-->0(0.30)  3-->[3-->1]-->1(0.40)
  distance = 0.324000

------ nc=15
( 0 1 ) 0-->[2-->3]-->2(0.10)  0-->[2-->0]-->3(0.30)
( 1 0 ) 1-->[1-->0]-->3(0.40)  1-->[1-->3]-->2(0.01)
( 1 1 ) 2-->[3-->2]-->0(0.10)  2-->[3-->1]-->1(0.01)
( 0 0 ) 3-->[0-->1]-->1(0.40)  3-->[0-->2]-->0(0.30)
  distance = 0.324000

------ nc=16
( 0 1 ) 0-->[2-->3]-->1(0.10)  0-->[2-->0]-->2(0.10)
( 1 1 ) 1-->[3-->2]-->0(0.10)  1-->[3-->1]-->3(0.40)
( 0 0 ) 2-->[0-->1]-->3(0.40)  2-->[0-->2]-->0(0.10)
( 1 0 ) 3-->[1-->0]-->2(0.40)  3-->[1-->3]-->1(0.40)
  distance = 0.400000

------ nc=17
( 0 1 ) 0-->[2-->3]-->1(0.10)  0-->[2-->0]-->3(0.30)
( 1 1 ) 1-->[3-->2]-->0(0.10)  1-->[3-->1]-->2(0.01)
( 1 0 ) 2-->[1-->0]-->3(0.40)  2-->[1-->3]-->1(0.01)
( 0 0 ) 3-->[0-->1]-->2(0.40)  3-->[0-->2]-->0(0.30)
  distance = 0.324000

------ nc=18
( 1 1 ) 0-->[3-->2]-->3(0.30)  0-->[3-->1]-->2(0.10)
( 0 0 ) 1-->[0-->1]-->2(0.01)  1-->[0-->2]-->3(0.40)
( 1 0 ) 2-->[1-->0]-->1(0.01)  2-->[1-->3]-->0(0.10)
( 0 1 ) 3-->[2-->3]-->0(0.30)  3-->[2-->0]-->1(0.40)
  distance = 0.324000
```

```
------ nc=19
( 1 1 ) 0-->[3-->2]-->2(0.10)  0-->[3-->1]-->3(0.30)
( 0 0 ) 1-->[0-->1]-->3(0.40)  1-->[0-->2]-->2(0.01)
( 0 1 ) 2-->[2-->3]-->0(0.10)  2-->[2-->0]-->1(0.01)
( 1 0 ) 3-->[1-->0]-->1(0.40)  3-->[1-->3]-->0(0.30)
  distance = 0.324000

------ nc=20
( 1 1 ) 0-->[3-->2]-->3(0.30)  0-->[3-->1]-->1(0.10)
( 1 0 ) 1-->[1-->0]-->2(0.01)  1-->[1-->3]-->0(0.10)
( 0 0 ) 2-->[0-->1]-->1(0.01)  2-->[0-->2]-->3(0.40)
( 0 1 ) 3-->[2-->3]-->0(0.30)  3-->[2-->0]-->2(0.40)
  distance = 0.324000

------ nc=21
( 1 1 ) 0-->[3-->2]-->2(0.10)  0-->[3-->1]-->1(0.10)
( 1 0 ) 1-->[1-->0]-->3(0.40)  1-->[1-->3]-->0(0.10)
( 0 1 ) 2-->[2-->3]-->0(0.10)  2-->[2-->0]-->3(0.40)
( 0 0 ) 3-->[0-->1]-->1(0.40)  3-->[0-->2]-->2(0.40)
  distance = 0.400000

------ nc=22
( 1 1 ) 0-->[3-->2]-->1(0.10)  0-->[3-->1]-->3(0.30)
( 0 1 ) 1-->[2-->3]-->0(0.10)  1-->[2-->0]-->2(0.01)
( 0 0 ) 2-->[0-->1]-->3(0.40)  2-->[0-->2]-->1(0.01)
( 1 0 ) 3-->[1-->0]-->2(0.40)  3-->[1-->3]-->0(0.30)
  distance = 0.324000

------ nc=23
( 1 1 ) 0-->[3-->2]-->1(0.10)  0-->[3-->1]-->2(0.10)
( 0 1 ) 1-->[2-->3]-->0(0.10)  1-->[2-->0]-->3(0.40)
( 1 0 ) 2-->[1-->0]-->3(0.40)  2-->[1-->3]-->0(0.10)
( 0 0 ) 3-->[0-->1]-->2(0.40)  3-->[0-->2]-->1(0.40)
  distance = 0.400000

 Initial distance = 0.400000

 Minimal distance = 0.324000 (3)-th  0  2  3  1
```

```
---------------------- 例 2 の結果 ----------------------------
 no_of_bits      = 2
 max_number      = 4
 no_of_errors    = 1
 probability     = 0.20
 distance matrix = distance_matrix


 0.00 0.10 0.20 0.30
 0.10 0.00 0.20 0.30
 0.40 0.20 0.00 0.50
 0.40 0.20 0.60 0.00
------ nc=0
( 0 0 ) 0-->[0-->1]-->1(0.10)  0-->[0-->2]-->2(0.20)
( 1 0 ) 1-->[1-->0]-->0(0.10)  1-->[1-->3]-->3(0.30)
( 0 1 ) 2-->[2-->3]-->3(0.50)  2-->[2-->0]-->0(0.40)
( 1 1 ) 3-->[3-->2]-->2(0.60)  3-->[3-->1]-->1(0.20)
  distance = 0.480000

------ nc=1
( 0 0 ) 0-->[0-->1]-->1(0.10)  0-->[0-->2]-->3(0.30)
( 1 0 ) 1-->[1-->0]-->0(0.10)  1-->[1-->3]-->2(0.20)
( 1 1 ) 2-->[3-->2]-->3(0.50)  2-->[3-->1]-->1(0.20)
( 0 1 ) 3-->[2-->3]-->2(0.60)  3-->[2-->0]-->0(0.40)
  distance = 0.480000

------ nc=2
( 0 0 ) 0-->[0-->1]-->2(0.20)  0-->[0-->2]-->1(0.10)
( 0 1 ) 1-->[2-->3]-->3(0.30)  1-->[2-->0]-->0(0.10)
( 1 0 ) 2-->[1-->0]-->0(0.40)  2-->[1-->3]-->3(0.50)
( 1 1 ) 3-->[3-->2]-->1(0.20)  3-->[3-->1]-->2(0.60)
  distance = 0.480000

------ nc=3
( 0 0 ) 0-->[0-->1]-->3(0.30)  0-->[0-->2]-->1(0.10)
( 0 1 ) 1-->[2-->3]-->2(0.20)  1-->[2-->0]-->0(0.10)
( 1 1 ) 2-->[3-->2]-->1(0.20)  2-->[3-->1]-->3(0.50)
( 1 0 ) 3-->[1-->0]-->0(0.40)  3-->[1-->3]-->2(0.60)
  distance = 0.480000

------ nc=4
( 0 0 ) 0-->[0-->1]-->2(0.20)  0-->[0-->2]-->3(0.30)
( 1 1 ) 1-->[3-->2]-->3(0.30)  1-->[3-->1]-->2(0.20)
( 1 0 ) 2-->[1-->0]-->0(0.40)  2-->[1-->3]-->1(0.20)
( 0 1 ) 3-->[2-->3]-->1(0.20)  3-->[2-->0]-->0(0.40)
  distance = 0.440000

------ nc=5
( 0 0 ) 0-->[0-->1]-->3(0.30)  0-->[0-->2]-->2(0.20)
```

```
( 1 1 ) 1-->[3-->2]-->2(0.20)  1-->[3-->1]-->3(0.30)
( 0 1 ) 2-->[2-->3]-->1(0.20)  2-->[2-->0]-->0(0.40)
( 1 0 ) 3-->[1-->0]-->0(0.40)  3-->[1-->3]-->1(0.20)
  distance = 0.440000

------ nc=6
( 1 0 ) 0-->[1-->0]-->1(0.10)  0-->[1-->3]-->3(0.30)
( 0 0 ) 1-->[0-->1]-->0(0.10)  1-->[0-->2]-->2(0.20)
( 0 1 ) 2-->[2-->3]-->3(0.50)  2-->[2-->0]-->1(0.20)
( 1 1 ) 3-->[3-->2]-->2(0.60)  3-->[3-->1]-->0(0.40)
  distance = 0.480000

------ nc=7
( 1 0 ) 0-->[1-->0]-->1(0.10)  0-->[1-->3]-->2(0.20)
( 0 0 ) 1-->[0-->1]-->0(0.10)  1-->[0-->2]-->3(0.30)
( 1 1 ) 2-->[3-->2]-->3(0.50)  2-->[3-->1]-->0(0.40)
( 0 1 ) 3-->[2-->3]-->2(0.60)  3-->[2-->0]-->1(0.20)
  distance = 0.480000

------ nc=8
( 1 0 ) 0-->[1-->0]-->2(0.20)  0-->[1-->3]-->3(0.30)
( 0 1 ) 1-->[2-->3]-->3(0.30)  1-->[2-->0]-->2(0.20)
( 0 0 ) 2-->[0-->1]-->0(0.40)  2-->[0-->2]-->1(0.20)
( 1 1 ) 3-->[3-->2]-->1(0.20)  3-->[3-->1]-->0(0.40)
  distance = 0.440000

------ nc=9
( 1 0 ) 0-->[1-->0]-->3(0.30)  0-->[1-->3]-->2(0.20)
( 0 1 ) 1-->[2-->3]-->2(0.20)  1-->[2-->0]-->3(0.30)
( 1 1 ) 2-->[3-->2]-->1(0.20)  2-->[3-->1]-->0(0.40)
( 0 0 ) 3-->[0-->1]-->0(0.40)  3-->[0-->2]-->1(0.20)
  distance = 0.440000

------ nc=10
( 1 0 ) 0-->[1-->0]-->2(0.20)  0-->[1-->3]-->1(0.10)
( 1 1 ) 1-->[3-->2]-->3(0.30)  1-->[3-->1]-->0(0.10)
( 0 0 ) 2-->[0-->1]-->0(0.40)  2-->[0-->2]-->3(0.50)
( 0 1 ) 3-->[2-->3]-->1(0.20)  3-->[2-->0]-->2(0.60)
  distance = 0.480000

------ nc=11
( 1 0 ) 0-->[1-->0]-->3(0.30)  0-->[1-->3]-->1(0.10)
( 1 1 ) 1-->[3-->2]-->2(0.20)  1-->[3-->1]-->0(0.10)
( 0 1 ) 2-->[2-->3]-->1(0.20)  2-->[2-->0]-->3(0.50)
( 0 0 ) 3-->[0-->1]-->0(0.40)  3-->[0-->2]-->2(0.60)
  distance = 0.480000

------ nc=12
( 0 1 ) 0-->[2-->3]-->3(0.30)  0-->[2-->0]-->1(0.10)
```

```
( 0 0 ) 1-->[0-->1]-->2(0.20)  1-->[0-->2]-->0(0.10)
( 1 0 ) 2-->[1-->0]-->1(0.20)  2-->[1-->3]-->3(0.50)
( 1 1 ) 3-->[3-->2]-->0(0.40)  3-->[3-->1]-->2(0.60)
  distance = 0.480000

------ nc=13
( 0 1 ) 0-->[2-->3]-->2(0.20)  0-->[2-->0]-->1(0.10)
( 0 0 ) 1-->[0-->1]-->3(0.30)  1-->[0-->2]-->0(0.10)
( 1 1 ) 2-->[3-->2]-->0(0.40)  2-->[3-->1]-->3(0.50)
( 1 0 ) 3-->[1-->0]-->1(0.20)  3-->[1-->3]-->2(0.60)
  distance = 0.480000

------ nc=14
( 0 1 ) 0-->[2-->3]-->3(0.30)  0-->[2-->0]-->2(0.20)
( 1 0 ) 1-->[1-->0]-->2(0.20)  1-->[1-->3]-->3(0.30)
( 0 0 ) 2-->[0-->1]-->1(0.20)  2-->[0-->2]-->0(0.40)
( 1 1 ) 3-->[3-->2]-->0(0.40)  3-->[3-->1]-->1(0.20)
  distance = 0.440000

------ nc=15
( 0 1 ) 0-->[2-->3]-->2(0.20)  0-->[2-->0]-->3(0.30)
( 1 0 ) 1-->[1-->0]-->3(0.30)  1-->[1-->3]-->2(0.20)
( 1 1 ) 2-->[3-->2]-->0(0.40)  2-->[3-->1]-->1(0.20)
( 0 0 ) 3-->[0-->1]-->1(0.20)  3-->[0-->2]-->0(0.40)
  distance = 0.440000

------ nc=16
( 0 1 ) 0-->[2-->3]-->1(0.10)  0-->[2-->0]-->2(0.20)
( 1 1 ) 1-->[3-->2]-->0(0.10)  1-->[3-->1]-->3(0.30)
( 0 0 ) 2-->[0-->1]-->3(0.50)  2-->[0-->2]-->0(0.40)
( 1 0 ) 3-->[1-->0]-->2(0.60)  3-->[1-->3]-->1(0.20)
  distance = 0.480000

------ nc=17
( 0 1 ) 0-->[2-->3]-->1(0.10)  0-->[2-->0]-->3(0.30)
( 1 1 ) 1-->[3-->2]-->0(0.10)  1-->[3-->1]-->2(0.20)
( 1 0 ) 2-->[1-->0]-->3(0.50)  2-->[1-->3]-->1(0.20)
( 0 0 ) 3-->[0-->1]-->2(0.60)  3-->[0-->2]-->0(0.40)
  distance = 0.480000

------ nc=18
( 1 1 ) 0-->[3-->2]-->3(0.30)  0-->[3-->1]-->2(0.20)
( 0 0 ) 1-->[0-->1]-->2(0.20)  1-->[0-->2]-->3(0.30)
( 1 0 ) 2-->[1-->0]-->1(0.20)  2-->[1-->3]-->0(0.40)
( 0 1 ) 3-->[2-->3]-->0(0.40)  3-->[2-->0]-->1(0.20)
  distance = 0.440000

------ nc=19
( 1 1 ) 0-->[3-->2]-->2(0.20)  0-->[3-->1]-->3(0.30)
```

```
( 0 0 ) 1-->[0-->1]-->3(0.30)  1-->[0-->2]-->2(0.20)
( 0 1 ) 2-->[2-->3]-->0(0.40)  2-->[2-->0]-->1(0.20)
( 1 0 ) 3-->[1-->0]-->1(0.20)  3-->[1-->3]-->0(0.40)
  distance = 0.440000

------ nc=20
( 1 1 ) 0-->[3-->2]-->3(0.30)  0-->[3-->1]-->1(0.10)
( 1 0 ) 1-->[1-->0]-->2(0.20)  1-->[1-->3]-->0(0.10)
( 0 0 ) 2-->[0-->1]-->1(0.20)  2-->[0-->2]-->3(0.50)
( 0 1 ) 3-->[2-->3]-->0(0.40)  3-->[2-->0]-->2(0.60)
  distance = 0.480000

------ nc=21
( 1 1 ) 0-->[3-->2]-->2(0.20)  0-->[3-->1]-->1(0.10)
( 1 0 ) 1-->[1-->0]-->3(0.30)  1-->[1-->3]-->0(0.10)
( 0 1 ) 2-->[2-->3]-->0(0.40)  2-->[2-->0]-->3(0.50)
( 0 0 ) 3-->[0-->1]-->1(0.20)  3-->[0-->2]-->2(0.60)
  distance = 0.480000

------ nc=22
( 1 1 ) 0-->[3-->2]-->1(0.10)  0-->[3-->1]-->3(0.30)
( 0 1 ) 1-->[2-->3]-->0(0.10)  1-->[2-->0]-->2(0.20)
( 0 0 ) 2-->[0-->1]-->3(0.50)  2-->[0-->2]-->1(0.20)
( 1 0 ) 3-->[1-->0]-->2(0.60)  3-->[1-->3]-->0(0.40)
  distance = 0.480000

------ nc=23
( 1 1 ) 0-->[3-->2]-->1(0.10)  0-->[3-->1]-->2(0.20)
( 0 1 ) 1-->[2-->3]-->0(0.10)  1-->[2-->0]-->3(0.30)
( 1 0 ) 2-->[1-->0]-->3(0.50)  2-->[1-->3]-->0(0.40)
( 0 0 ) 3-->[0-->1]-->2(0.60)  3-->[0-->2]-->1(0.20)
  distance = 0.480000

 Initial distance = 0.480000

 Minimal distance = 0.440000 (4)-th  0  3  1  2
```