NAIST-IS-DT9661022

博士論文

相互結合型ニューラルネットワークによる 2次割当て問題の解法

新妻 弘崇

1999年1月8日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

新妻 弘崇

審査委員：　伊藤 実 教授
西谷 紘一 教授
高橋 豊 教授
石井 信 助教授

相互結合型ニューラルネットワークによる 2次割当て問題の解法*

新妻 弘崇

内容梗概

組合せ最適化問題のニューラルネットワークによる解法としては, ボルツマンマシンによる解法がある. この一種の確率的探索手法は計算スピードが非常に遅いため, 平均場近似と呼ばれる近似手法が良く用いられる. しかし, この近似により決定論的ダイナミクス, いわゆる力学系となるため確率的探索と違い, 局所最適解からの脱出が困難となる. この局所最適解からの脱出を行う為に, カオスニューラルネットワークを使う手法が提案されている. カオスニューラルネットワークとは, 非線型な演算ユニットである各ニューロンに自分自身への負の再帰的入力を加えることで, カオスを生じさせる手法である. このカオスが長い時間相関を持つような乱数の役割を果たし, 局所最適解からの脱出が可能になる.

本研究では, 組合せ最適化問題でも, 特に2次割当て問題に注目する. 2次割当て問題, 巡回セールスマン問題, N-Queen 問題等を解くニューラルネットワークでは, 解の表現に共通した特徴をもっている. 具体的には, 解の表現においていくつかの変数の和が一定になるような制約条件が存在する. 前述の平均場近似を使ったアプローチでは, これらの制約条件は制約条件を表現するペナルティ項を目的関数に加えることにより, 制約条件が満たされやすくなる様にする手法がしばしば用いられてきた. しかし, この方法は制約条件を満たすとは限らないために, 本来の組合せ最適化問題の解の候補にならないような非解をしばしば導いてしまう. そこで, 全ての制約条件を変数に明に入れることにより, ほぼ制約条件が満たさ

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DT9661022, 1999年1月8日.

れるようにした手法である 2 重制約ネットワーク (doubly constrained network, DCN) が提案され, 良い解を得ている.

本研究では、新しいニューラルネットワークに基づく組合せ最適化問題の解法を 2 つ提案する.

新たに提案する手法の 1 つは, カオスニューラルネットワークによる解法で, カオスによる非平衡探索がより良く機能するための手法である. 目的関数がより平坦になるような, その結果, 目的関数の曲率行列の固有値が平均化されシステムはどの方向にも動きやすくなるような, 座標変換を行うことにより, カオスによる非平衡ダイナミクスがより多くの局所最適解を探索できるようにする. 本研究では, この座標変換の手法を DCN のカオスバージョンに対して適用した. 本手法の有効性を確認するために, 中規模の 2 次割当て問題に応用した結果, より多くの局所最適解を探索でき, より良い解を得ることが出来ることが分かった.

提案する 2 つ目の手法は,λ-opt ヒューリスティクスのアナログ版に基づく手法である. λ-opt ヒューリスティクスとは, 解を表現する順列ベクトルのλ個の要素に対する置換操作を行うことで, 解の改善を行う手法である. このλ-opt ヒューリスティクスをニューラルネットワークを用いてアナログ的に表現し, 従来のλ-opt ヒューリスティクスよりも, 遠距離への探索を少ない計算時間で実現できるようにした. この手法は,DCN に対して新たな制約条件を加えた手法とみなすこともできる. この手法 (λ-DCN と呼ぶ) を 2 次割当て問題の標準的ベンチマークの集合である QAPLIB のベンチマークに適用して計算機実験を行った結果, いくつかのベンチマークに対しては今までチャンピオンであったアルゴリズムよりも良い解を得られることがわかった. また、その他の比較的大規模な ($N = 80 \sim 150$)2 次割当て問題に対しても, λ-DCN は今までのチャンピオンアルゴリズムとほぼ同程度の近似解を計算できることが分かった. さらにλ-DCN に内点法の考えを部分的に導入したλ-iterior DCN も提案した. λ-iterior DCN は, λ個以下の要素の置換操作を考える手法である. すなわち, 設定されたλの値よりも少ない要素の入れ替えを行われた順列が最適解である時に有効な手法である. 計算機実験の結果, λ-iterior DCN はλ-DCN よりも, さらに良い近似解が求められる傾向があることが分かった.

本研究で主に扱うニューラルネットによる組合せ最適化問題の解法は, 行列演算を多用するアルゴリズムである. これらの行列とベクトルで表現された問題を, いくつかの部分空間上の問題に分割することができるならば, アルゴリズムの高速化を行うことができる. 行列で表された問題を, 部分空間上の問題に分割する際に有用なものとして, ジョルダン標準形やスミス標準形などがある. ジョルダン標準形はその特性行列のスミス標準形から導くことができる。一方, 安定化理論は, 数式処理アルゴリズムを浮動小数を使って近似し, 高速化する手法であるが, 浮動小数近似を安定に行う時に必要な浮動小数桁数を理論的には与えない. 本研究では, 安定化理論を使いスミス標準形を近似的に求める時に必要となる浮動小数桁数の実験的な見積もりを行った. 結果として, 近似に必要となる浮動小数桁数は行列の大きさと共に指数的に増加していくことが分かった.

以上の研究により, ニューラルネットワークによる組合せ最適化問題の解法で、より良い近似解を計算できるようになった. 本研究で提案した手法は, 原理的にDCN 以外のニューラルネットワーク手法にも適用可能であり, 多くのニューラルネットワーク手法を改善できる可能性がある. これらの手法は, 比較的大規模な組合せ最適化問題を確率的手法を取り入れることにより, 実用的な時間で扱える手法でもある. また, ニューラルネットワークと数式処理の融合に必要と考えられる基礎的研究を行った.

キーワード

非線型最適化 , 2 重制約ネットワーク , 2 次割り当て問題, λ-opt , カオスニューラルネットワーク, 安定化理論

Recurrent neural networks for quadratic assignment problems*

Hirotaka Niitsuma

Abstract

Boltzmann Machine is a neural network model for solving combinatorial optimization problems. This method requires a lot of computational time because it is statistical. A mean field approximation improves this method by making the system deterministic. The deterministic method, on the other hand, can not escape from local minima by itself. Then, a chaos neural network is proposed in order to overcome this problem.

This thesis studies recurrent neural network models that solve combinatorial optimization problems. In particular, we deal with quadratic assignment problems (QAPs) as optimization problems. QAPs are very difficult combinatorial problems. A solution of a neural network solving QAPs , traveling sales parson problems and N-Queen problems have a common feature, namely, there are constraints that the sum of variables should be a constant. If these constraints are treated as a penalty term which is added to the objective function of the combinatorial problem, the system becomes to find poor solutions especially when the problem scale is large. In a model called doubly constrained network (DCN), these constraints are automatically satisfied, and consequently obtained solutions are fairly good.

*Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9661022, January 8, 1999.

We propose novel neural network models that solve QAPs. These new models are derived from DCN. We call these methods MDCN, λ-DCN and λ-interior DCN.

MDCN is a nonlinear optimization method based on a combination of chaotic dynamics and a coordinate transformation. In this method, the coordinate transformation makes the eigen values of the curvature matrix for the objective function, defined for each optimization problem, small. Then, the objective function becomes flat so that the chaotic dynamics is able to search for feasible solutions over a wide domain region. MDCN is applied to QAP benchmark problems. As a result, our new method achieves solution improvements.

λ-DCN is a neural network approach based on λ-opt heuristics. We applied λ-DCN to relatively large-scale problems in QAPLIB. λ-DCN is comparable to some algorithms which can find the best feasible solution for problems in QAPLIB. Moreover, λ-DCN can obtain better solutions than these algorithms for two benchmark problems. We also invented a method called λ-interior DCN, which partly combines an interior point method with λ-DCN. λ-interior DCN is apt to find better solution than λ-DCN.

The above-mentioned neural network methods are expressed by matrices and vectors. If these problems which are described by matrices and vectors decompose into sub space, the problems are decomposed to several small sub problems and the algorithm can be much faster. In particular, we focus on Smith normal form. The stabilization theory is often used to compute Smith normal form. However, the theory does not estimate an upper bound of the computational error for stable computation. Then, we experimentally estimated the upper bound in special case.

In conclusion (1) we propose novel neural methods that solve combinatorial optimization problems ; (2) some proposed methods can obtain better solutions of QAPs than previous champion algorithms ; (3) in order to decompose a problem expressed by matrices and vectors, the upper bound of the error realized by the stabilization theory is experimentally estimated.

Keywords:

nonlinear optimization , doubly constrained network , quadratic assignment problem , λ-opt, chaotic neural network

目 次

図目次

表目次

業績リスト

- 博士論文に関連する業績
  - 論文
    * 新妻 弘崇, 石井 信, 伊藤 実: 座標変換を用いたカオス最適化手法, 電子情報通信学会誌 A, 条件付き採録 (3 章の内容)
    * 新妻 弘崇, 石井 信, 伊藤 実: ニューラルネットによるλ-opt アルゴリズムを使った 2 次割り当て問題の解法, 電子情報通信学会誌 D, 条件付き採録 (4 章の内容)
    * H.Niitsuma and S.Ishii: λ-Opt neural approaches to quadratic assignment problem, Neural Computation, submitted (4 章の内容)
    * 白柳 潔, 新妻 弘崇: 実多項式行列スミス標準形の安定な浮動小数点計算法, 情報処理学会, 印刷中 (5 章の内容)
  - 査読付き国際会議
    * H.Niitsuma, S.Ishii and M.Ito: Chaotic optimization and linear transformation, Proceedings of the 1997 International Symposium on Nonlinear Theory and Its Applications, 601-604 (1997).
  - 国内会議等
    * 新妻 弘崇, 石井 信, 伊藤 実: 信学技報,NLP98-72 (1998-11),17-23 (1998).
    * 新妻 弘崇, 白柳 潔: 浮動小数ジョルダン標準形の安定化, 数理解析研究所講究録 986,195-205 (1997).
- その他の業績
  - S.Nakashima, K.Kisoda, H.Niizuma and H.Harima: Raman scattering of disordered SiC, Physica B, 219&220 , 371-373 (1996).

1. 序論

1.1 最適化問題

本研究では最適化問題の解法を扱う. 多くの最適化問題は次の式で記述できる.

$$\text{minimize}_x f(x) \ \text{ subject to } \ g_i(x) \leq 0 \ \ (x \in R^n, i = 1, 2, 3...) \tag{1.1}$$

式 (1.1) は不等式制約条件の下で記述されているが, 等式制約条件は複数の不等式制約条件で記述できるため省略してある[1]. $f(x)$ および $g(x)$ が線形の場合, この問題は容易に解くことができる. 等式制約条件のみしか現れない場合には, ラグランジュの未定係数法が多く用いられる. しかし, ラグランジュの未定係数法は, 非線形な最適化問題を非線形な代数方程式へと変換する手法であるため, 特に大規模な問題に対しては, 常に有効な方法とは限らない.

不等式制約条件を含む問題の一般的な解法としては, 制約条件付きの問題を制約のない問題の列に変換して逐次的にこれを解く変換法と呼ばれる方法がある. 変換法の例としては, ペナルティ関数法がある. ペナルティ関数法では, 制約条件を満たさない点 $x \notin F$ (以下, 制約条件で表される領域を F と表す) に対しては非常に大きな値をとるような項 (これをペナルティ項と呼ぶ) を最小化すべき目的関数 $f(x)$ に加えて, 拡張された目的関数

$$\hat{f}(x;t) = f(x) + tG(x) \tag{1.2}$$

を作り, この $\hat{f}$ を最小にするような制約のない問題を解く. ここに, $G(x)$ はペナルティ項であり, t はスカラーのパラメーターである. この t を変化させながら $\hat{f}$ の最小化を繰り返し行ない, 原問題の最適解 x^0 へ収束させる. 具体的には次の 2 つの方法がある.

- 外点法

  図 1のように, $G(x)$ を $x \in F$ ならば 0 で, $x \notin F$ ならば正の (大きな) 値をとる x の連続関数とする. このとき, $k \to \infty$ とともに無限大に発散する単調増

[1] $h(x) = 0$ という制約条件は $h(x) \leq 0, h(x) \geq 0$ という 2 つの不等式制約条件で記述できる

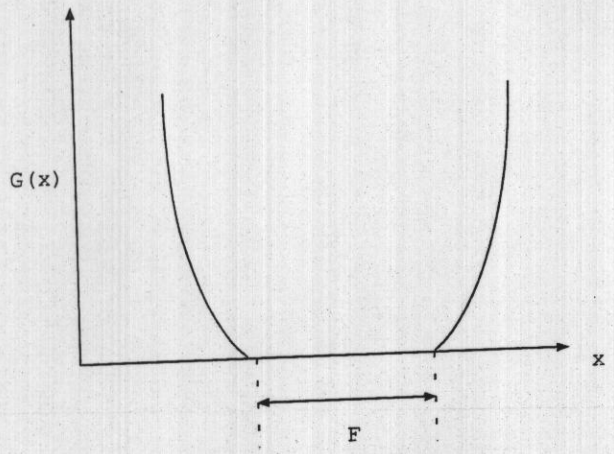


図 1 外点法のペナルティ項

加数列 $\{t^{(k)}\}$ に対応して式 (1.2) の関数の最小化を行なえば, その解 $\{x^{(k)}\}$ は適当な条件のもとで x^0に収束する.

- 内点法

  図 2のように,$G(x)$ を x が Fの内点ならば非負の値をとり,x が Fの境界上の点ならば無限大となる x の連続関数とする. このとき,$k \to \infty$ とともに 0 に収束する単調減少数列 $\{t^{(k)}\}$ 対応して式 (1.2) の関数の最小化を行なえば, その解 $\{x^{(k)}\}$ は適当な条件のもとで x^0に収束する.

外点法は可能領域の外部から, また内点法は可能領域の内部から x^0に収束させる.

本研究では主としてニューラルネットワークによる組合せ最適化問題の解法を扱う. この手法は, 離散的な問題である組合せ最適化問題を式 (1.1) の様な連続な最適化問題に変換し, この連続な問題を解くことで組合せ最適化問題を解く手法である. したがって, その連続な問題を扱うためにここで説明したような, 内点法, ラグランジュ未定係数法などの考え方が使われる.

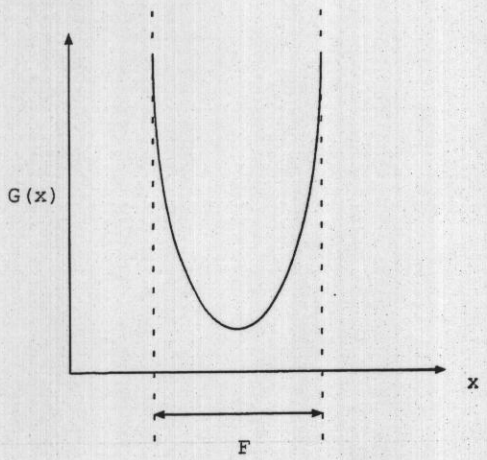


図 2 内点法のペナルティ項

1.2 ニューラルネットワークによる組合せ最適化問題の解法

神経細胞のモデルとしてMcCulloch と Pitts [25] は, 次の様なモデルを提案した.

$$S_i(t+1) = \Theta(\sum_j w_{i,j} S_j(t) - \mu_i) \tag{1.3}$$

$S_i(t)$ は1,0の値をとる2値変数であり,i 番目のニューロンが時刻 t で発火する状態を $S_i(t) = 1$ で表現し, 発火しない状態を $S_i(t) = 0$ として表現している. 時間 t は, 1つの演算プロセスが終わるまでを1ステップとする離散的な時間 $t = 1, 2, 3, 4...$ とする. Θは次のようなステップ関数である.

$$\Theta(x) = \begin{array}{ll} 1 & (x \geq 0) \\ 0 & (\text{それ以外}) \end{array} \tag{1.4}$$

$w_{i,j}$は i 番目のニューロンと j番目のニューロンをつなぐシナプスの強さを表しており, 正と負, 両方の値を取ることができる. 正の値をとるシナプスは興奮性, 負の値をとるシナプスは抑制性のシナプスに対応している. $w_{i,j}$が 0 の場合は,i 番目のニューロンと j番目のニューロンをつなぐシナプスが存在しないことを示している. 式 (1.3) は, 「i 番目のニューロンが発火するかどうかは, 多くのニューロ

ンからシナプスを伝って入力される信号の総和$\sum_j w_{i,j}S_j(t)$ によって決まる」というモデルを表している. μ_iは,i 番目のニューロンの発火のしきい値である.

式 (1.3) のモデルを次に行うの議論のために, ニューロンの状態を $S_i = 1, 0$ ではなく,$\sigma_i = 1, -1$ の 2 値で記述するモデルに書き換えてみる.

$$\sigma_i(t+1) = \mathrm{sgn}(\sum_j w_{i,j}\sigma_j(t) - \theta_i) \tag{1.5}$$

ここで

$$\sigma_i = 2S_i - 1$$

という変数変換を行った. $S_i = 1$ が$\sigma_i = 1$ に,$S_i = 0$ が$\sigma_i = -1$ に対応している. $\theta = 2\mu_i - \sum_j w_{i,j}$であるが, 簡単の為に$\theta_i = 0$ とする. つまり,

$$\sigma_i(t+1) = \mathrm{sgn}(\sum_j w_{i,j}\sigma_j(t)) \tag{1.6}$$

の繰り返し計算を考える. すると, 式 (1.6) のモデルは, 次の関数 E_{obj}の値を常に小さくする方向にσ_iの値を更新するモデルとなる.

$$E_{obj} = -\sum_{i,j} w_{i,j}\sigma_i\sigma_j \tag{1.7}$$

式 (1.6) の演算の結果σ_iの値が更新されず, $\sigma_i(t+1) = \sigma_i(t)$ となるならば, 関数 E_{obj}の値も変化しない. したがって$\sigma_i(t+1) \neq \sigma_i(t)$ つまり$\sigma_i(t+1) = -\sigma_i(t)$ の場合を考える. ここで, 一回の演算プロセスで変化するニューロンはσ_iのみであり, σ_i以外のニューロンσ_j $(j \neq i)$ の値は, 演算の前後で変化しない点に注意する[2]. この時の E_{obj}の変化は

$$\begin{aligned} E_{obj}(\sigma(t+1)) - E_{obj}(\sigma(t)) &= -\sum_{j\neq i} w_{i,j}\sigma_i(t+1)\sigma_j(t) + \sum_{j\neq i} w_{i,j}\sigma_i(t)\sigma_j(t) \\ &= 2\sigma_i(t)\sum_{j\neq i} w_{i,j}\sigma_j(t) \\ &= 2\sigma_i(t)\sum_j w_{i,j}\sigma_j(t) - 2w_{i,i} \end{aligned} \tag{1.8}$$

[2]このように, 一回の演算プロセスで 1 つのニューロンのみの値を更新する手法を, 非同期型ニューラルネットワークと言う. 一回の演算プロセスで全てのニューロンの値を更新する手法を, 同期型ニューラルネットワークと言う.

と計算できる. 式(1.8)の最後の行の第1項は負の値となる. なぜならば,$\sigma_i(t+1) = -\sigma_i(t)$ となるためには, 式(1.6)において$\sigma_i(t)$ と$\sum_j w_{i,j}\sigma_j(t)$ の符合が異なっていなければならないため, 2つの符合が異なるものの積$\sigma_i(t)\sum_j w_{i,j}\sigma_j(t)$ は負となるからである. 式(1.8)の最後の行の第2項 $w_{i,i}$はi番目のニューロンが自分自身に与える影響を示す項である. この項は,$w_{i,i} \geq 0$ とすると[3], 常に負になる. したがって, 式(1.6)の繰り返し計算は, 常に式(1.7)の関数 E_{obj}を小さくするか, 変数σ_iの値の更新をしないかのどちらかであることがわかる. つまり, 式(1.6)の繰り返しが収束すると, 関数 E_{obj}の極小点を求めることができる.

組合せ最適化問題を2値変数$\sigma_i = -1, 1$ の関数である式(1.7)の E_{obj}の最小点を求める問題として記述できるならば, 式(1.6)の繰り返しを行うことで, その組合せ最適化問題の局所最適解の1つを求めることができる. これがニューラルネットワークによる組合せ最適化問題の近似解法の基本である. このような関数 E_{obj}を以下で目的関数と呼ぶ.

しかし, 式(1.6)の繰り返し計算は, 最適化問題のあまり良くない局所最適解に収束してしまう場合がある. これを避けるために, システムの挙動を確率的にした次のモデルを考える.

$$p(\sigma_i \to \sigma_i') = f_\beta(-\triangle E_{obj}) = \frac{1}{1+\exp(\beta \triangle E_{obj})} \tag{1.9}$$

$$\triangle E_{obj} = E_{obj}(\sigma') - E_{obj}(\sigma) \tag{1.10}$$

ここで,$p(\sigma_i \to \sigma_i')$ は,σ_iの値が変化する確率を示す. つまり,$\sigma_i = 1$ ならば$\sigma_i = -1$ へ変化する確率, $\sigma_i = -1$ ならば$\sigma_i = 1$ へ変化する確率をまとめて表記している. 以下, 本論文では, 状態「$\cdot$」が生じる確率を $p(\cdot)$ で示す. $\triangle E_{obj}$は,σ_iが変化した場合の目的関数 E_{obj}の変動量を示す. βはパラメーターである. 図3に関数 $f_\beta(-x)$ の形状を示す. 関数 $f_\beta(-x)$ は, パラメーターβが大きくなるほど式(1.4)のステップ関数$\Theta(x)$ に近づいていく. $f_\beta(0) = 0.5$ であり,$x > 0$ ならば $f_\beta(-x)$ は1に近い値をとるためσ_iが変化する確率が高くなる. 逆に $x < 0$ ならば $f_\beta(x)$ は0に近い値をとるためσ_iが変化する確率が低くなる. $x = -\triangle E_{obj}$であることを考えると, 式(1.9)のシステムは, 目的関数 E_{obj}の値が小さくなるような状態へ変化する

[3] p 個の極小点を持つように関数 E_{obj}を構成すると $w_{i,i} = p/N$となる. ここで Nはニューロンの個数である

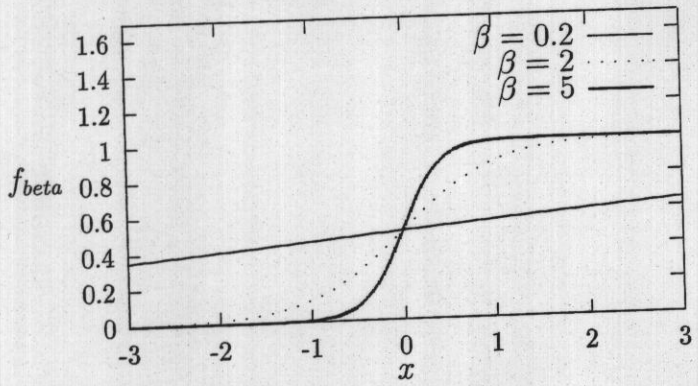


図 3 シグモイド関数

確率が高いシステムであると言える. βの値として非常に大きな値をとると f_βはステップ関数となるため, 式 (1.9) のシステムは, 式 (1.6) のシステムと等価なシステムになる. つまり, 変数σ_iを変化させると目的関数値が下がるならばσ_iを常に変化させ, 変数σ_iを変化させると目的関数値が上がる場合には常にσ_iを変化させないような決定的システムとなる. したがって, 式 (1.9) のシステムは式 (1.6) のシステムをさらに一般化したものであると言える. さらに, 式 (1.9) のシステムは, 目的関数 E_{obj}として式 (1.7) の形以外の一般的な関数 E_{obj}を考えた場合にも同様の議論から, 関数 E_{obj}の極小点を発見できるシステムであると言える.

1.2.1 ニューラルネットワークとボルツマン分布

以下では, ニューロンの状態を記述する変数として,$\sigma_i = -1, 1$ ではなく $S_i = 0, 1$ を再び使うことにする. 式 (1.9) のシステムは, ニューラルネットワーク全体が, 状態$\boldsymbol{S} = (S_1, S_2, S_3....)$ にある確率 $p(\boldsymbol{S})$ を, 次の統計力学でボルツマン分布と呼ばれる確率分布で与えたシステムと見ることもできる.

$$p(\boldsymbol{S}) = \exp(-\beta E_{obj}(\boldsymbol{S}))/Z \tag{1.11}$$

$p(\boldsymbol{S})$ は各ニューロン $S_1, S_2, S_3....$が$\boldsymbol{S} = (S_1, S_2, S_3....)$ の状態にある確率である. Zはボルツマン分布関数の規格化定数で

$$Z = \sum_{\boldsymbol{S} \in \boldsymbol{C}} \exp(-\beta E_{obj}(\boldsymbol{S})) \tag{1.12}$$

である. $\boldsymbol{C}$は, 可能な全てのニューロンの状態の集合$\boldsymbol{C} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \boldsymbol{S}_3, ...\}$ を表す. この Zは分配関数と呼ばれる. 各ニューロンの状態を表す変数 S_i $(i = 1, 2....)$ は統計力学とのアナロジーから, スピンとも呼ばれる. 同様に統計力学とのアナロジーから, パラメータβのかわりに$T = 1/\beta$というパラメータを使った$p(\boldsymbol{S}) = \exp(-E_{obj}(\boldsymbol{S})/T)/Z$ という表記法も使われる. このパラメーターTは, 統計力学とのアナロジーから温度と呼ばれる. ボルツマン分布 (1.11) と, 式 (1.9) のシステムとの関係を示すために, 次の式を考える.

$$\begin{aligned} p(S_j = \hat{S}_{j \neq i}, S_i = 1) &= p(S_i = 1 | S_j = \hat{S}_{j \neq i}) \;\; p(S_j = \hat{S}_{j \neq i}) \\ &= p(S_i = 1 | S_j = \hat{S}_{j \neq i}) \;\; (p(S_j = \hat{S}_{j \neq i}, S_i = 1) \\ &\qquad + p(S_j = \hat{S}_{j \neq i}, S_i = 0)) \end{aligned} \tag{1.13}$$

$p(S_j = \hat{S}_{j \neq i}, S_i = 1)$ は,i 番目のスピン S_iが $S_i = 1$ であり, i 番目以外のスピン S_j $(j \neq i)$ は全て $S_j = \hat{S}_j$である確率を表している. $p(S_j = \hat{S}_{j \neq i}, S_i = 0))$ も同様である. $p(S_j = \hat{S}_{j \neq i})$ はスピン S_iの値を指定しない (つまり,$S_i = 1, 0$ 両方の場合を考えた) 同様の確率である. $p(S_i = 1 | S_j = \hat{S}_{j \neq i})$ は, $S_j = \hat{S}_{j \neq i}$ということが決まっている状態で, $S_i = 1$ となる確率である. 式 (1.13) を次の様に書き換える.

$$\begin{aligned} p(S_i = 1 | S_j = \hat{S}_{j \neq i}) &= \frac{p(S_j = \hat{S}_{j \neq i}, S_i = 1)}{p(S_j = \hat{S}_{j \neq i}, S_i = 1) + p(S_j = \hat{S}_{j \neq i}, S_i = 0)} \\ &= \frac{1}{1 + \frac{p(S_j = \hat{S}_{j \neq i}, S_i = 0)}{p(S_j = \hat{S}_{j \neq i}, S_i = 1)}} \end{aligned} \tag{1.14}$$

式 (1.14) にボルツマン分布 (1.11) を代入すると

$$\begin{aligned} p(S_i = 1 | S_j = \hat{S}_{j \neq i}) &= \frac{1}{1 + \frac{\exp(-E_{obj}(S_j = \hat{S}_{j \neq i}, S_i = 0)/T)}{\exp(-E_{obj}(S_j = \hat{S}_{j \neq i}, S_i = 1)/T)}} \\ &= \frac{1}{1 + \exp((E_{obj}(S_j = \hat{S}_{j \neq i}, S_i = 1) - E_{obj}(S_j = \hat{S}_{j \neq i}, S_i = 0))/T)} \\ &= \frac{1}{1 + \exp(\triangle E_{obj}(S_j = \hat{S}_{j \neq i}, S_i = 0 \to 1)/T)} \end{aligned} \tag{1.15}$$

となる．ここで,$E_{obj}(S_j = \hat{S}_{j\neq i}, S_i = 0), E_{obj}(S_j = \hat{S}_{j\neq i}, S_i = 1)$ は, 確率 $p(S_j = \hat{S}_{j\neq i}, S_i = 0), p(S_j = \hat{S}_{j\neq i}, S_i = 1)$ が表現しているのと同様の状態に対する目的関数 E_{obj}の値である．

$$\triangle E_{obj}(S_j = \hat{S}_{j\neq i}, S_i = 0 \rightarrow 1) = E_{obj}(S_j = \hat{S}_{j\neq i}, S_i = 1) - E_{obj}(S_j = \hat{S}_{j\neq i}, S_i = 0)$$

であり, スピン S_iが 0 の時と 1 の時の E_{obj}の差を表している．この式 (1.15) は, $S_j = \hat{S}_j\ (j \neq i)$, $S_i = 0$ の状態から S_iが変化して $S_i = 1$ になる確率を表現しているとも解釈できる．スピン S_iが 1 から 0 へ変化する場合についても, 同様に考えることができる．そして, ニューロンの状態を S_iではなくσ_iで表現し,$\beta = 1/T$の関係式を使うと式 (1.9) を導くことができる．

ボルツマン分布 (1.11) にしたがうようにスピンを確率的に変化させて, 目的関数 E_{obj}の最小点を探す手法は, ボルツマンマシンと呼ばれる．ボルツマン分布は,E_{obj}が小さな値となるスピンほど生じやすくなるような確率分布である．しかし, 温度 Tが大きいと全てのスピンの確率は同程度となり, 一方で温度 Tが小さいと小さな E_{obj}の値を持つスピン値が生じる確率が相対的に大きくなる．また,$T \rightarrow +0$ においては, 目的関数を最小化するスピンの確率が 1 になる．したがって, 温度 0 の極限のボルツマン分布を実現できれば良いが, 実際には無限の時間がかかるため, 有限の温度がしばしば用いられる．

ボルツマンマシンは, 式 (1.9) のシステムと同様に, 任意の形の目的関数 E_{obj}を扱うことができる．本研究で特に興味がある, 多くの NP 完全 (NP 困難) な組合せ最適化問題は, 以下の 2 次の目的関数 E_{obj}の最小化問題として定式化することができる．

$$E_{obj}(\boldsymbol{S}) = \frac{1}{2}\boldsymbol{S}^t\boldsymbol{W}\boldsymbol{S} + \boldsymbol{J}\boldsymbol{S} = \frac{1}{2}\sum_{i,j=1}^{N} W_{i,j}S_iS_j + \sum_{i=1}^{N} J_iS_i \qquad (1.16)$$

$N \times N$次行列$\boldsymbol{W}$と N次元ベクトル$\boldsymbol{J}$は, 記述したい組合せ最適化問題によって決まる定数である．$W_{i,j}$は i 番目と j番目のニューロンをつなぐシナプス結合の強さ, J_iは i 番目のニューロンの発火しやすさを記述するパラメータと見ることもできる．なぜならば, 式 (1.9) のシステムをσ_iではなく, スピン S_iを使った表記に

書き換えて, 目的関数 (1.16) を代入してみると

$$\begin{aligned} p(S_i = 1) &= f_\beta([\sum_j W_{i,j} S_j + J_i]) \\ p(S_i = 0) &= 1 - f_\beta([\sum_j W_{i,j} S_j + J_i]) = f_\beta(-[\sum_j W_{i,j} S_j + J_i]) \end{aligned} \quad (1.17)$$

という式を導くことができる. 導出は付録 Aに示す. この式 (1.17) と式 (1.3) を比較してみると, 式 (1.17) は式 (1.3) に確率的モデルを導入して拡張したものになっていることがわかる. また $W_{i,j}$は i 番目と j番目のニューロンをつなぐシナプス結合の強さ $w_{i,j}$に, J_iは i 番目のニューロンの発火しやすさを記述するパラメータ μ_iに対応していることもわかる.

1.2.2 ポッツ平均場近似

確率的探索手法であるボルツマンマシンではボルツマン分布を実現する必要がある. これにはメトロポリス法などの手法があるが, 一般に多くの計算時間を必要とする. そこでスピンの期待値 $V_i =< S_i >$ $(0 \leq V_i \leq 1)$ を平均場近似 (Mean field theory, MFT) [5, 28] を使って計算する手法が良く用いられる. 次に, 平均場近似について説明する.

ボルツマン分布 (1.11) は, 次の自由エネルギーϕを最小にするような分布関数である.

$$\phi =< E_{obj} > -TH \quad (1.18)$$

ここで,$< E_{obj} >$は E_{obj}の期待値であり,

$$< E_{obj} >= \sum_{\boldsymbol{S} \in \boldsymbol{C}} p(\boldsymbol{S}) E_{obj}(\boldsymbol{S}) \quad (1.19)$$

である. Tは温度,Hはエントロピーと呼ばれる関数であり, 次で定義される.

$$H = -\sum_{\boldsymbol{S} \in \boldsymbol{C}} p(\boldsymbol{S}) \log p(\boldsymbol{S}) \quad (1.20)$$

自由エネルギーϕからボルツマン分布 (1.11) は, 次のように導ける. $p(\boldsymbol{S})$ は確率なので,

$$\sum_{\boldsymbol{S} \in \boldsymbol{C}} p(\boldsymbol{S}) = 1 \quad (1.21)$$

でなければいけない．この制約条件の基でϕを最小化する$p(\boldsymbol{S})$を求めるために，次のラグランジュ関数を考える．

$$\phi + \lambda(\sum_{\boldsymbol{S}\in\boldsymbol{C}} p(\boldsymbol{S}) - 1)$$

ここでλは制約条件 (1.21) を表すラグランジュの未定係数である．このラグランジュ関数を$p(\boldsymbol{S})$で微分すると，次の停留条件を導ける．

$$E_{obj}(\boldsymbol{S}) + T(\log p(\boldsymbol{S}) + 1) - \lambda = 0 \tag{1.22}$$

$$\Sigma_{\boldsymbol{S}\in\boldsymbol{C}}\, p(\boldsymbol{S}) = 1 \tag{1.23}$$

この停留条件を計算すると，ボルツマン分布 (1.11) となる．[4]

スピンとして，より拡張された次のようなベクトルのスピンを扱うことを考える．より具体的には，M次元のスピンをN個並べたものである，$N \times M$-次元の行列のスピンを考える．つまり

$$\boldsymbol{S} = [\vec{S}_1^t ... \vec{S}_N^t]^t \tag{1.24}$$

である．ここで$\vec{S}_1 ... \vec{S}_N$は，それぞれM次元ベクトルであり，各M次元ベクトルの要素で1の値を取るのは1つだけで，残り$M-1$個の要素は0となる場合を考える．つまり

$$\sum_{n=1}^{M} S_{a,n} = 1 \tag{1.25}$$

である．このようなベクトルを以下，スピンベクトルと呼ぶ．行列$\boldsymbol{S}$を以下，スピン行列と呼ぶ．このスピンベクトル$\vec{S}_i = (S_{i,1}, ..., S_{i,M})$は，組合せ最適化問題で，1番から$M$番の要素の中から1つだけの要素を選択したい，という状況を記述するために使われる．このような，制約条件 (1.25) を満足するようなベクトルのスピンを扱うシステムは，ポッツスピンと呼ばれる．

制約条件 (1.25) がある場合に，節 1.2.1で行なったのと同様に，ボルツマン分布から式 (1.9) のようなシステムを導出することを考えてみよう．制約条件 (1.25)

[4]付録で説明する自由エネルギーの最小原理によると，最も生じる確率の高い状態は，自由エネルギーの最小状態である．

の結果, スピンベクトルが「1番からM番の要素の中から1つだけの要素を選択したい」という状態を表現できるようになった. したがって,i番目のスピンのみについての確率$p(S_i=1|S_j=\hat{S}_{j\neq i})$ではなくスピンベクトルに対する確率

$$V_{a,n}=p(\vec{S}_a=\boldsymbol{e}_n|\vec{S}_j=\hat{\vec{S}}_{j\neq a}) \tag{1.26}$$

を考えることにする. ここで$\boldsymbol{e}_n$は, 第n要素のみが1で他の成分が全て0の単位ベクトルである. 以下,

$$\vec{V}_a = (V_{a,1},...,V_{a,M}) \tag{1.27}$$

$$\boldsymbol{V} = [\vec{V}_1^t...\vec{V}_N^t]^t \tag{1.28}$$

と表記する. $V_{a,n}$はスピン行列の期待値にもなっている.

$$\begin{aligned} <S_{a,n}> &= 1\times p(\vec{S}_a=\boldsymbol{e}_n|\vec{S}_j=\hat{\vec{S}}_{j\neq a})+0\times\sum_{m\neq n}p(\vec{S}_a=\boldsymbol{e}_m|\vec{S}_j=\hat{\vec{S}}_{j\neq a}) \\ &= p(\vec{S}_a=\boldsymbol{e}_n|\vec{S}_j=\hat{\vec{S}}_{j\neq a})=V_{a,n} \end{aligned} \tag{1.29}$$

$V_{a,n}$はスピンベクトル$\vec{S}_a$の状態に関する確率分布を表現しているため, 次の制約条件を満足しないといけない.

$$\sum_{n=1}^{M}V_{a,n}=1 \tag{1.30}$$

$<S_{a,n}>=V_{a,n}$であるため, 式(1.30)は制約条件(1.25)に対応しており, 式(1.26)の形の確率分布を扱う場合, 制約条件(1.25)に対応した制約条件(1.30)は常に成り立つ.

式(1.26)を

$$V_{a,n}=p(\vec{S}_a=\boldsymbol{e}_n|\vec{S}_j=\hat{\vec{S}}_{j\neq a})\approx p(\vec{S}_a=\boldsymbol{e}_n) \tag{1.31}$$

と近似する手法は, 平均場近似(Mean field thory, MFT)と呼ばれる. この近似の意味は, 次のようなものである. $\vec{S}_a=\boldsymbol{e}_n$となる確率$V_{a,n}$は, スピン$S_{a,1},...S_{a,N}$が, a番目のスピンベクトル以外のスピンベクトルのニューロンとも, シナプスで結合しているため, $S_{a,1},...S_{a,N}$の値のみでは決められない. つまり, $V_{a,n}=p(\vec{S}_a=\boldsymbol{e}_n|\vec{S}_j=\hat{\vec{S}}_{j\neq a})$となるが, この$a$番目の以外のスピンベクトル$\vec{S}_1,...,\vec{S}_{a-1},\vec{S}_{a+1},...,\vec{S}_1$からの

影響を, 平均化してしまい, $\vec{S}_1, ..., \vec{S}_{a-1}, \vec{S}_{a+1}, ..., \vec{S}_1$の値に依存しない確率 $p(\vec{S}_a = \boldsymbol{e}_n)$ で近似しているのである. この平均場近似の結果, この N個のスピンベクトルに対する確率分布が, それぞれ独立であると近似できるようになる. つまり,

$$p(\boldsymbol{S}) \approx \Pi_{i=1}^{N} p(\vec{S}_i) \tag{1.32}$$

と近似できる.

式 (1.9) のシステムは, ボルツマン分布から導くことができた. ボルツマン分布は, 自由エネルギーを最小とするような確率分布であった. 同様に, 自由エネルギーを最小とするような式 (1.9) に対応した確率分布 $V_{a,n}$を, 次に求める.

行列で表されたスピン$\boldsymbol{S}$を使い,E_{obj}をより一般的な次の様な形で定義することにする.

$$E_{obj}(\boldsymbol{S}) = \frac{1}{2}\boldsymbol{S}^t \boldsymbol{W} \boldsymbol{S} + \boldsymbol{J}\boldsymbol{S} = \frac{1}{2}\sum_{a,b=1}^{N} \vec{S}_a^T \boldsymbol{W}_{a;b} \vec{S}_b + \sum_{a=1}^{N} \boldsymbol{J}_a \vec{S}_a \tag{1.33}$$

平均場近似 (1.32) を使うと, 式 (1.33) で定義された E_{obj}の期待値$< E_{obj} >$を, 次の様に計算できる.

$$\begin{aligned}
< E_{obj} > &= \sum_{\boldsymbol{S} \in \boldsymbol{C}} p(\boldsymbol{S}) E_{obj}(\boldsymbol{S}) \\
&\approx \sum_{\boldsymbol{S}_1} \cdots \sum_{\boldsymbol{S}_N} (\frac{1}{2} \sum_{a,b=1}^{N} \boldsymbol{S}_a^T \boldsymbol{W}_{a;b} \boldsymbol{S}_b + \sum_{a=1}^{N} \boldsymbol{J}_a \boldsymbol{S}_a) \Pi_{a=1}^{N} p(\boldsymbol{S}_a) \\
&\approx \sum_{a \neq b} \{ \sum_{\{\boldsymbol{S}_c | c \neq a,b\}} \Pi_{c \neq a,b} p(\boldsymbol{S}_c) \} \{ \sum_{\boldsymbol{S}_a} \sum_{\boldsymbol{S}_b} (\frac{1}{2} \boldsymbol{S}_a^T \boldsymbol{W}_{a;b} \boldsymbol{S}_b) p(\boldsymbol{S}_a) p(\boldsymbol{S}_b) \} \\
&+ \sum_{a} \{ \sum_{\{\boldsymbol{S}_c | c \neq a\}} \Pi_{c \neq a} p(\boldsymbol{S}_c) \} \{ \sum_{\boldsymbol{S}_a} (\frac{1}{2} \boldsymbol{S}_a^T \boldsymbol{W}_{a;a} \boldsymbol{S}_a + \boldsymbol{J}_a \boldsymbol{S}_a) p(\boldsymbol{S}_a) \} \\
&\approx \sum_{a,b} (\frac{1}{2} \boldsymbol{V}_a^T \boldsymbol{W}_{a;b} \boldsymbol{V}_b + \boldsymbol{J}_a \boldsymbol{V}_a) \\
&\approx \frac{1}{2} \boldsymbol{V}^T \boldsymbol{W} \boldsymbol{V} + \boldsymbol{J}\boldsymbol{V} \equiv E(\boldsymbol{V})
\end{aligned} \tag{1.34}$$

ここで$\boldsymbol{W}_{a;a} = 0$ を仮定した. 以下, 上式の最後の行の式を $E(\boldsymbol{V})$ と表すことにする.

同様に, エントロピー Hも計算することができる.

$$H \approx -\sum_{a=1}^{N}\sum_{\boldsymbol{S}_a} p(\boldsymbol{S}_a)\log p(\boldsymbol{S}_a) \approx -\sum_{a=1}^{N}\sum_{n=1}^{N} V_{a,n}\log V_{a,n} \tag{1.35}$$

以上の計算から, 自由エネルギーϕは次の式で計算できる.

$$\phi \approx E(\boldsymbol{V}) + T\sum_{a=1}^{N}\sum_{n=1}^{N} V_{a,n}\log V_{a,n} \tag{1.36}$$

この近似された自由エネルギーを最小化するようなスピンの期待値$\boldsymbol{V}$を求めることで, 目的の式 (1.9) に対応した式を求められる. 行列$\boldsymbol{V}$は式 (1.30) の制約条件を満たさないといけない. その様な$\boldsymbol{V}$の値は, 次のラグランジュ関数についてのラグランジュの未定係数法で求めることができる.

$$L = E(\boldsymbol{V}) + T\sum_{a=1}^{N}\sum_{n=1}^{M} V_{a,n}\log V_{a,n} + \sum_{a=1}^{N}\tilde{\mu}_a(\sum_{n=1}^{M} V_{a,n} - 1) \tag{1.37}$$

ここで$\tilde{\mu}_a$ $(a = 1, ..., N)$ はラグランジュの未定係数である. このラグランジュ関数の停留条件は, 次のポッツ MFT 方程式 [28] と呼ばれる方程式となる.

$$\boldsymbol{U}_a = -\nabla_{\boldsymbol{V}_a} E_{obj}(\boldsymbol{V}) \tag{1.38}$$

$$\boldsymbol{V}_a = \frac{\exp(\boldsymbol{U}_a/T)}{\sum_{m=1}^{M}\exp(U_{a,m}/T)} \tag{1.39}$$

ここで $\nabla_{\boldsymbol{V}_a} E_{obj}(\boldsymbol{V})$ は $E_{obj}(\boldsymbol{V})$ のベクトル$\boldsymbol{V}_a$の各要素に対する $E_{obj}(\boldsymbol{V})$ の偏微分を表している. これで, 目的の式 (1.9) に対応した式を求められた. この方程式の解は, 自由エネルギーを最小化するようなスピンの期待値であるが, 温度が小さい場合, 自由エネルギーは E_{obj}とほぼ同程度の値となる. したがって, 温度が小さい場合に, この方程式を解ければ E_{obj}を最小とするようなスピンの期待値が求まる. このようなポッツスピンを扱うシステムを, ポッツスピンモデルと呼ぶことにする.

式 (1.38),(1.39) の解は, 次の連続時間のポッツスピンモデルを使って求めることができる.

$$\tau\dot{U}_{a,n} = -U_{a,n} + \sum_{b,m} W_{a,n;b,m}V_{b,m} + J_{a,n} \tag{1.40}$$

$$V_{a,n} = I_n(\boldsymbol{U}_a) \equiv \frac{\exp(U_{a,n}/T)}{\sum_m \exp(U_{a,m}/T)} \tag{1.41}$$

ここで$\dot{U}_{a,n} \equiv dU_{a,n}/dt$である. tは時間を表す. τは時間についてのスケーリングパラメーターである. この時間発展システムは$\dot{\phi} \leq 0$つまり常に自由エネルギーを減らすシステムであることが証明できる. したがって, 式 (1.40),(1.41) の時間発展を追っていけば, 式 (1.37) で定義されるラグランジュ関数の停留点を求めることができる. なお$M = 2$の場合に (1.40),(1.41) は, 特にアナログ Hopfield モデル [11] と呼ばれる.

1.2.3 カオスポッツスピン

式 (1.40),(1.41) で記述される系は$\dot{\phi} \leq 0$であるため, 一度, 固定点アトラクタである局所最適解に収束してしまうと, そこから脱出してより良い解を探索することができない. そこで, 局所最適解からの脱出の為にカオスを用いる手法が提案された. 次に, そのカオスを使った手法について説明する.

式 (1.38),(1.39) の時間発展を, オイラー法による差分方程式から求めていく場合を考える. その場合の各変数の時間発展は, 次の様に記述できる.

$$U_{a,n}(t) = kU_{a,n}(t-1) + (1-k)(\sum_{b,m} W_{a,n;b,m}V_{b,m}(t-1) + J_{a,n}) \tag{1.42}$$

$$V_{a,n}(t) = I_n(\boldsymbol{U}_a) \tag{1.43}$$

ここで$k = 1 - \delta t/\tau$である. tは離散時間であり, δtはオイラー積分における連続時間の時間間隔である. この力学系は$W_{a,n;a,n} > 0$とすると, カオス的振る舞いをすることがある. このカオス的振る舞いを使い, 局所最適解からの脱出を行うのがカオスポッツスピン [13] である. カオスポッツスピンは, カオス的特性を持ちつつもポッツスピン系の自由エネルギー最小化の特性により, 自由エネルギーの小さな状態になりやすい傾向をもっている. 従って, 局所最適化としての特性とカオスによるランダムサーチの特性を併せ持つ手法であると言える.

1.3 2 次割り当て問題

本研究では、特に組合せ最適化問題の中でも 2 次割り当て問題に注目する. 本節では 2 次割り当て問題について説明する.

2 次割当て問題 (Quadratic assignment problem QAP) [6] は, 非常に難しい組合せ最適化問題として知られている. QAP は NP-困難な問題のクラスに属する. QAP をうまく表す例の 1 つとして, 工場配置問題がある. 工場配置問題とは, コストを最小とする様な工場の配置の仕方を探す問題である. コストは, それぞれの工場の間でやりとりされる部品の流量と, 工場が配置された位置間の距離の積を, 全ての工場と位置の組合せに対して和をとることで与えられる. 巡回セールスマン問題 (TSP) や, 最大クリーク問題などの多くの組合せ最適化問題は, この 2 次割り当て問題の特殊で実質的に簡単な場合である.

$\boldsymbol{D}, \boldsymbol{C}$を, それぞれ位置間の距離, 工場間の流量を表す $(N \times N)$-次元行列とする. 大きさ N（工場の数に相当する）の 2 次割り当て問題は次で定義される.

$$\min_{\pi \in \Pi} \sum_{a=1}^{N} \sum_{b=1}^{N} D_{a,b} C_{\pi(a),\pi(b)} \tag{1.44}$$

Πは集合 $\{1, 2, ..., N\}$ 上の順列の集合, $\pi(a)$ は順列$\pi \in \Pi$の a 番目の位置に配置された要素を表す. 工場配置問題の場合は, 位置 a に工場$\pi(a)$ が配置されることに対応する.

QAP の厳密解を求める手法としては $N \leq 15$ 程度の小規模問題については, 分枝限定法による手法が有効であるが, $N \geq 15$ の中から大規模の QAP に対して厳密解を求めるのは困難である. したがって, 近似解を求めることが実用上重要である.

良い近似解を求められる手法としては

- シミュレーティッド アニーリング
- 遺伝的アルゴリズム
- タブーサーチ法

などの, いわゆる「メタヒューリスティクス」と呼ばれる手法がある.

本研究では, ニューラルネットワークに基づく QAP の近似解法を提案する. 本研究で提案したアルゴリズムの実用性の検討のために計算機実験を行うが, 実験で使用したベンチマークは, QAP に対する標準的なベンチマークの集合である QAPLIB [6] のベンチマークを使う.

1.4 2 重制約ネットワーク

TSP, QAP, N-Queen 問題等を解くニューラルネットワークは共通した特徴をもっている. 具体的には, いくつかの変数の和が一定になるような制約条件が存在する. これらの制約条件は変数に明には入らず, 弱い制約として扱われることが多い. 弱い制約とは, 制約条件を表現するペナルティ項を目的関数に加えることにより, 制約条件が満たされやすくなる様にする方法である. しかし, この弱い制約を用いたシステムは, 本来の組合せ最適化問題の解の候補にならないような非解をしばしば導いてしまう. 全ての制約条件を強い制約として扱う, すなわち, 変数に制約を明に入れることにより, ほぼ制約条件が満たされるようにした手法である 2 重制約ネットワーク (doubly constrained network, DCN) [12] を, これまでに提案し, 良い解を得ている.

ここでは DCN の説明を行うがその前に, ポッツスピンモデルでの QAP の解法について説明する. ポッツスピンモデルでは, 式 (1.44) の問題を式 (1.24) で定義されたスピン行列を使って次の様に表現する.

$$S_{a,n} = \begin{array}{ll} 1 & (\text{順列}\pi\text{の}\ a\ \text{番目が}\ n) \\ 0 & (\text{それ以外}) \end{array}$$

この $N \times N$-次元行列は, 式 (1.24) で定義されたスピン行列の特別な場合である. この行列を, 以下で配置行列と呼ぶことにする.

配置行列は, 順列をスピンで表現したものであるため

$$\sum_{a=1}^{N} S_{a,n} = 1 \ (n = 1, ..., N) \tag{1.45}$$

$$\sum_{n=1}^{N} S_{a,n} = 1 \ (a = 1, ..., N) \tag{1.46}$$

という制約条件を満たさないといけない. 式 (1.45) は, 要素 n が順列中に 1 回しか現れないことを示す制約条件であり, 式 (1.46) は, 順列の同じ場所に複数の要素が入らないことを示す制約条件であり, 式 (1.25) で $M = N$とする場合に相当する. すなわち, この制約条件はポッツスピンについての制約条件 (1.25) に新たな制約条件 (1.45) を加えたものである.

配置行列を使うと, 式 (1.44) は次の様な目的関数の最小化問題と定義される.

$$E_{obj}(\boldsymbol{S}) = \frac{1}{2} \sum_{a,n,b,m=1}^{N} W_{a,n;b,m} S_{a,n} S_{b,m} \tag{1.47}$$

ここで $W_{a,n;b,m} = D_{a,b} C_{n,m}$である. これはポッツスピンを定義する時に用いた, 式 (1.33) の特別な場合である. [5]

したがって, 配置行列$\boldsymbol{S}$を使うと QAP は, 次の様に表せるようになる.

$$\begin{aligned}
\text{minimize} \quad & E_{obj}(\boldsymbol{S}) = \tfrac{1}{2} \textstyle\sum_{a,n,b,m=1}^{N} W_{a,n;b,m} S_{a,n} S_{b,m} && (1.48)\\
\text{subject to} \quad & \textstyle\sum_{a=1}^{N} S_{a,n} = 1 \ \ (n = 1, ..., N) && (1.49)\\
& \textstyle\sum_{n=1}^{N} S_{a,n} = 1 \ \ (a = 1, ..., N) && (1.50)
\end{aligned}$$

QAP を, この様にスピンで表現することで, スピンモデルに基づいたニューラルネットワークモデルで扱うことができるようになる.

スピンモデルに基づいたアナログニューラルネットワークモデルは, 最も最適な配置行列の期待値$\boldsymbol{V} = < \boldsymbol{S} >$を求めるシステムであり, 次の問題を解くシステムと見ることができる.

$$\begin{aligned}
\text{minimize} \quad & \phi(\boldsymbol{V}) = E(\boldsymbol{V}) - TH(\boldsymbol{V}) && (1.51)\\
\text{subject to} \quad & \textstyle\sum_{a=1}^{N} V_{a,n} = 1 \ \ (n = 1, ..., N) && (1.52)\\
& \textstyle\sum_{n=1}^{N} V_{a,n} = 1 \ \ (a = 1, ..., N) && (1.53)
\end{aligned}$$

しかし, ポッツスピンモデルは, 式 (1.25) つまり QAP の場合での式 (1.46) の制約条件のみを満たすようなスピンの期待値を求める手法であった. つまり, ポッツ

[5] 一方, $C_{n,m} = (\delta_{n,m+1} + \delta_{n,m-1})/2$ とすると QAP は TSP となることに注意する. ここで$\delta_{i,j}$はクロネッカーのデルタである. すなわち TSP は QAP の特別な場合である.

スピンモデルは, 式 (1.45) の制約条件 (スピンの期待値を用いた場合では制約条件式 (1.52)) を自動的に満足することができない. したがって QAP を, ポッツスピンモデルで扱おうとする場合には, 式 (1.52) の制約条件は, 弱い制約条件, つまり目的関数に式 (1.52) の制約条件を表すペナルティ項を加えることで, 表現される. しかし, 式 (1.52) の制約条件を弱い制約条件として扱うと, 特に問題の大きさが大きくなる時に, 妥当でない, つまり式 (1.45) の制約条件を満たさないスピン値を多く計算してしまう. 一方, それを防ぐために, ペナルティ項の効果を大きくしすぎると, 逆に制約条件は満たすが目的関数値の大きなスピンが求められる. その為, ポッツスピンモデルは, あまり良い近似解を見つけることが出来なかった. その問題を解決する為に, 式 (1.52) の制約条件をも自動的に満足するようにした, つまり, 式 (1.52) を強い制約条件として扱うことが出来るようにしたシステムが DCN である.

次に DCN ついて説明する. DCN は, 制約条件 (1.52) を満たす為に, ポッツスピンの様に目的関数に制約条件 (1.52) を表すペナルティ項を加えるのではなく, 式 (1.37) のラグランジュ関数に, 制約条件 (1.52) に対応する項,

$$\sum_{n=1}^{N} \tilde{\lambda}_n (\sum_{a=1}^{N} V_{a,n} - 1) \tag{1.54}$$

を加えるシステムである. ここで$\tilde{\lambda}_n$ $(n = 1, ..., N)$ はラグランジュの未定係数である. こうすることで, 制約条件 (1.52) を強い制約条件として扱うことが出来るようになる. すなわち,DCN では次のラグランジュ関数を扱う.

$$L = < E_{obj} > (\boldsymbol{V}) + T \sum_{a=1}^{N} \sum_{n=1}^{N} V_{a,n} \log V_{a,n} + \sum_{a=1}^{N} \tilde{\mu}_a (\sum_{n=1}^{N} V_{a,n} - 1) + \sum_{n=1}^{N} \tilde{\lambda}_n (\sum_{a=1}^{N} V_{a,n} - 1) \tag{1.55}$$

このラグランジュ関数の停留点, すなわち, $\frac{\partial L}{\partial V_{an}} = 0$ となる点は次の連立方程式の解として与えられる.

$$V_{a,n} = \frac{\exp(U_{a,n})/\lambda_n}{\sum_m (\exp(U_{a,m})/\lambda_m)} \tag{1.56}$$

$$U_{a,n} \equiv \exp(-\frac{1}{T} \frac{\partial E(\boldsymbol{V})}{\partial V_{a,n}}) \tag{1.57}$$

$$\lambda_n = \sum_a \frac{\exp(U_{a,n})}{\sum_m(\exp(U_{a,m})/\lambda_m)} \tag{1.58}$$

ここでλ_n $(n=1,...,N)$ はラグランジュの未定係数に, 変数変換をしたものである. この方程式はDCN方程式と呼ばれる. 付録にDCN方程式の導出を示す.

DCN方程式は, 制約条件(1.45),(1.46)がある場合の式(1.9)に対応した式である. DCN方程式(1.56),(1.57),(1.58)の解を得るために, 式(1.9)のシステムと同様な, 次のアルゴリズムを実行するのがDCNである.

1. $t=0$ とする.$\boldsymbol{V}(0)$ に次の値を代入する.

$$V_{a,n}(0) = (1/N)(1+\epsilon_{a,n})$$

ここで,$\epsilon_{a,b}$は一様乱数である. また

$$\lambda_n(0) = \phi_n \ \ (n=1,...,N)$$

とする. ここでϕ_aは一様乱数である.

2. すべての a,n について以下を計算する.

$$U_{a,n}(t+1) = \exp(-\frac{1}{T}\frac{\partial E(\boldsymbol{V}(t))}{\partial V_{a,n}(t)}) \tag{1.59}$$

3. λ_n^{old}に$\lambda_n(t)$ を代入する.

4. 次の計算を繰り返しする.

   (a) 全ての n について次の計算をする

$$\lambda_n^{new} = \sum_{a=1}^{N} \frac{U_{a,n}(t+1)}{\sum_{m=1}^{N}\frac{U_{a,m}(t+1)}{\lambda_m^{old}}}$$

   λ_nの値が収束したならば, その値を$\lambda_n(t+1)$ に代入する. その後で$\boldsymbol{\lambda}(t+1)$ が $\sum_{n=1}^{N}\lambda_n = 1$ となるように$\boldsymbol{\lambda}(t+1)$ をスケールする.

5. 全ての a,n について以下を計算する.

$$V_{a,n}(t+1) = \frac{U_{a,n}(t+1)/\lambda_n(t+1)}{\sum_{m=1}^{N}\frac{U_{a,m}(t+1)}{\lambda_m(t+1)}} \tag{1.60}$$

6. tに1を加えてステップ (2) に戻る.$\boldsymbol{V}$が収束すれば終了する

以上のアルゴリズムを, 以下でDCNの基本アルゴリズムと呼ぶ.

DCNの基本アルゴリズムは, 収束することが保証されたアルゴリズムではない. しかし, 経験的には, 式 (1.47) で定義された目的関数に問題の本質的意味を変えないような微少な修正を行うことで, 収束するようにできることが分かっている. 具体的には, 次の修正を行う. 式 (1.47) で定義されたE_{obj}は, ポッツスピンを定義する時に用いた, 式 (1.33) を$W_{a,n;b,m}=0, J_{a,n}=0$としたものであった. しかし,$W_{a,n;b,m}, J_{a,n}$に0でない微少な数値を代入するとDCNの基本アルゴリズムを収束できるようにできる. ただし, この微少な数値を求める理論的手法はない.

DCNの基本アルゴリズムによって, DCN方程式 (1.56),(1.57),(1.58) の解を計算できるが, 高い温度に対する解をはじめに計算し, 温度を少しづつ小さくしながらDCNの基本アルゴリズムを実行する手法も提案されている. この手法はDCNアニーリング (DCA) [12] と呼ばれる.

1.4.1 カオス的DCN

DCN方程式の解は, ポッツスピンモデルと同様に, 次の時間的に連続なシステムの時間発展を求めることでも求まると期待できる.

$$\tau \dot{U}_{a,n} = -U_{a,n} + \sum_{b,m} W_{a,n;b,m} V_{b,m} + J_{a,n} \tag{1.61}$$

$$V_{a,n}(t) = \frac{\exp(U_{a,n}(t))/\lambda_n(t)}{\sum_m (\exp(U_{a,m}(t))/\lambda_m(t))} \tag{1.62}$$

ここで$\dot{U}_{a,n} \equiv dU_{a,n}/dt$である. tは時間を表す. τは時間についてのスケーリングパラメーターである. $\lambda_n(t)$は, 次のλ_nについての計算を収束するまで繰り返した時の収束値である.

$$\lambda_n = \sum_a \frac{\exp(U_{a,n}(t))}{\sum_m (\exp(U_{a,m}(t))/\lambda_m)} \tag{1.63}$$

式 (1.63) の収束については証明されているが, 式 (1.61),(1.62) が, ポッツスピンモデルと同様に$\dot{\phi} \leq 0$を与えることは証明できない. 実際にポッツスピンの場合と同様に式 (1.61),(1.62) の時間発展をオイラー法で差分方程式を扱いながら求めて

procedure k-exchange$(k, \mathrm{N}_k(\pi), \pi)$

1. do πが $\mathrm{N}_k(\pi)$ 内の最適解ではない→

2. $\mathrm{N}_k(\pi)$ 内からπよりも良い解 qをみつける;

3. $\pi = q$とする;

4. od;

5. return(π)

図 4 k-exchange

いく場合, $W_{a,n;a,n} > 0$ とすると, カオス的振る舞いをすることがある. このシステムはカオス的 DCN (chaotic doubly constrained network, CDCN) [14] と呼ばれる.

カオスポッツスピンと同様に CDCN を定式化すると次の様になる.

$$U_{a,n}(t) = kU_{a,n}(t-1) + (1-k)(\sum_{b,m} W_{a,n;b,m} V_{b,m}(t-1) + J_{a,n}) \quad (1.64)$$

$$V_{a,n}(t) = \frac{\exp(U_{a,n}(t))/\lambda_n(t)}{\sum_m (\exp(U_{a,m}(t))/\lambda_m(t))} \quad (1.65)$$

ここで $k = 1 - \delta t/\tau$であり,δt はオイラー法での時間を区切る間隔である. CDCN は, カオスポッツスピンモデルが弱い制約条件として扱っていた制約条件を強い制約条件として扱うことができるため, より良い解を発見できるシステムとなっている.

1.5 2 次割り当て問題の置換操作に基づく近似解法

式 (1.44) から分かるように,QAP は最適な順列を発見する問題である. 本研究では, 順列への置換操作に基づく QAP の近似解法に注目する. 置換操作を利用した QAP の近似解法で, 特に良い近似解を発見できる手法としては Greedy Randomized Adaptive Search Procedure(GRASP) [22] などがある. GRASP は QAPLIB のいくつかのベンチマークに対して, 現時点での最良解を計算している.

表 1 各近傍の性質

	N_k	N_λ	N^*
直径	$\min\{2k, N\}$	N	N
サイズ	C_k^N	$N^3/24$	$N^4/18$

このことから, 置換操作に基づく近似解法は,QAP に対しても有効な手法であると考えられる.

置換操作を利用する局所ヒューリスティクスとしては, k-exchange,λ-exchange,N^* 近傍に基づく手法などがある. これらの手法は, ある順列πの近傍集合から, 最も良い順列を選びだす手法である. この近傍として, 次で定義される k-exchange 近傍 $\mathrm{N}_k(\pi)$ を使うのが k-exchange である.

$$\mathrm{N}_k(\pi) = \{q | d(q, \pi) \leq k\} \tag{1.66}$$

ここで,$d(q, \pi)$ は,2 つの順列 q, πの距離を定義する関数で

$$d(q, \pi) = |\delta(q, \pi)| \tag{1.67}$$

で与えられる. $\delta(q, \pi)$ は,

$$\delta(q, \pi) = \{i | q(i) \neq \pi(i)\} \tag{1.68}$$

であり, 順列 q, πの異なる要素の集合を表している. $d(q, \pi)$ は, 集合$\delta(q, \pi)$ の大きさ, つまり順列 q, πの異なる要素の数である. 図 4に k-exchange アルゴリズムを示す. QAPLIB のいくつかのベンチマークの最良解を計算している手法は, この 2-exchange にタブー効果や遺伝的アルゴリズムを組み合わせた手法である. [6]

λ-exchange はλ-exchange 近傍 $\mathrm{N}_\lambda(\pi)$ を, N^*近傍に基づく手法は N^*近傍を用いる手法である. 付録 Bにλ-exchange と N^*近傍に基づく手法の詳細を示す.

[6]GRASP はタブー効果を含まない手法であるため, タブーサーチよりも置換操作の有効性を示す手法であると言える.

これらの局所ヒューリスティクスの性質を比較したのが表 1である. 表 1の直径とは, ヒューリスティクスが利用する近傍 $\mathbf{N}(\pi)$ の大きさ示すもので

$$\text{直径} = \max_{q,r \in \mathbf{N}} d(q,r)$$

で与えられる. 直径は, 大きいほど近傍に含まれる順列が多様であることを示している. したがって, より大きい直径を与えられる近傍を使うほど, 遠距離探索を行っていると見なせる. 表 1のサイズは近傍に含まれる順列の数を示している. このサイズが小さいほど, 局所ヒューリスティクスの計算時間は少ない. $\mathbf{N}_k(\pi)$ のサイズは,$k = 4$ の時に $\mathbf{N}_\lambda$,$\mathbf{N}^*$と同程度となる. しかし $k = 4$ の場合,$\mathbf{N}_k(\pi)$ の直径は $\mathbf{N}_\lambda$,$\mathbf{N}^*$よりも小さい. $\mathbf{N}_k(\pi)$ のサイズは $k > 4$ とすると非常に大きくなる. $\mathbf{N}^*$のサイズは $\mathbf{N}_\lambda$よりもわずかに大きいが, 計算機実験ではよりよい性能を発揮する [21].

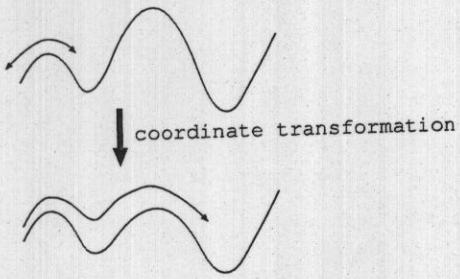


図 5 本手法の概念図

2. 座標変換を用いたカオス最適化手法

2.1 はじめに

本節では, 特にカオスニューラルネットワークによる最適化手法に注目する. 野沢 [26] は, アナログ Hopfield モデルのカオスバージョンを提案した. これは, 合原ら [1] のカオスニューラルネットワークと等価である. 物理モデルで言えば, 合原らのモデルはイジングスピンモデルに基づく. 一方, 節 1.2.3ではポッツスピンに基づいたカオスニューラルネットワークの概説を行った. また, 節 1.4.1でも DCN のカオスバージョン (CDCN) の説明を簡単に行った. 長谷川ら [10] は, 局所的ヒューリスティックとカオスニューラルネットとを組合せた最適化アルゴリズムを提案した.

本節では, 本研究で提案した座標変換を用いたカオス最適化手法について説明する. ここで座標変換とは, 目的関数の形状を定義域にわたってより平坦にするものである. このことによって, 図 5に示すように, 複数の局所最適解の間にある目的関数の山を緩やかにできる. 解の探索手法としてカオスによる非平衡ダイナミクスを用いると, カオスダイナミクスが局所最適解からの脱出により効果的に働き, より多くの解候補を探索できるようになる. よって, システムの初期状態に関わらず多くの良い局所最適解を探索でき, 得られる解も向上すると考えられる.

我々の座標変換手法そのものは, 非平衡ダイナミクスを持つ各種探索アルゴリズムに適用可能であるが, 本論文ではより具体的に CDCN に本手法を適用して

QAP を解くことによって, 手法の有効性を示す. DCN は最急降下法と似た計算を行うアルゴリズムであり, CDCN はカオスによる非平衡ダイナミクスを併用した最急降下法に類似する. 従って我々の座標変換手法が有効に働くと考えられる.

最急降下法において, 線形な座標変換はしばしば収束速度を速めるための局所的スケーリングに用いられる [24]. 一方, 本手法では, 目的関数の山を大域的に緩やかにするために使われる. これはスケーリングあるいは内点法 [17] などとは異なる手法である.

2.2 手法

$\boldsymbol{V} \equiv (V_1, V_2, ...V_N)^t$ を N次元変数ベクトルとする. ただし, t は転置である. $\boldsymbol{V}$ を変数とした以下の最適化問題を考える.

$$\text{minimize} \quad P(\boldsymbol{V}) \tag{2.1}$$

$$\text{subject to} \quad A\boldsymbol{V} = \boldsymbol{r} \tag{2.2}$$

$$V_i \geq 0 \tag{2.3}$$

A は $(M \times N)$ 次行列, $\boldsymbol{r}$は M次元ベクトルである. 式 (2.2) は線形の制約条件を表す. 以下, Pを目的関数と呼ぶ. 式 (2.1), (2.2), (2.3) で定義される最適化問題は, 式 (1.1) の特別な場合であるが, 一般に多くの局所最適解が含まれる. すなわち, 目的関数は多くの極小点をもち, それらは目的関数の壁によって隔てられている. そうした場合, 最急降下法等の方法によって大域的最適解を求めるのは難しく, ランダム性やカオスによる非平衡ダイナミクスによって目的関数の壁を乗り越え, 局所最適解から脱出し, 異なる解を探索する方法がしばしば用いられる [10, 1, 14, 13, 27].

本論文では, 座標変換を用いてカオスによる解の探索が効率的に動作する手法を提案する. 目的関数のラプラシアン $\triangle_V P = \sum_{i=1}^{N} \frac{\partial^2}{\partial V_i^2} P$ を下げると, $\boldsymbol{V}$の周りの局所的な関数の変化が小さくなる. 定義域全体において, $\triangle_V P$を小さくすることができれば, 目的関数をより平坦な関数にできる. 結果として局所最適解からの脱出を阻む目的関数の壁は小さくなり, カオスによる非平衡な解探索により目的関数の壁を乗り越えやすくなる. これが本手法の概要である.

具体的には, 以下で定義するエネルギー関数を小さくする様な座標変換を選ぶ.

$$E_{laplacian} = \int |\triangle_X P|^2 dV \tag{2.4}$$

ここで, $\boldsymbol{V}$と$\boldsymbol{X}$はそれぞれ座標変換前の座標系と座標変換後の座標系を表わす. $\triangle_X = \sum_{i=1}^{N} \frac{\partial^2}{\partial X_i^2}$ は, 変換後の座標系でのラプラシアンである. 積分は, 目的関数の定義域 (すなわち$\boldsymbol{V}$の定義域全体) にわたって行う.

本論文では, 簡単のため, 座標変換として以下の式で定義する線形変換のみを考える.

$$\boldsymbol{V} = F\boldsymbol{X} \tag{2.5}$$

ここで, Fは $(N \times N)$ 次行列である. 線形な座標変換を行う場合の $E_{laplacian}$の意味については, 2.5で詳しく説明する.

$E_{laplacian}$を小さくする座標変換を単純に求めようとすると, 単に座標を引き延ばすだけになりやすい. その様な当たり前の座標変換を求めることを避ける為に, 以下の弱い制約を導入する.

$$E_{intra} = \sum_{i=1}^{N} (1 - |\boldsymbol{f}_i|^2)^2 \tag{2.6}$$

$$E_{inter} = -\sum_{i,j=1}^{N} \log(1 - \cos^2(\boldsymbol{f}_i, \boldsymbol{f}_j)) \tag{2.7}$$

ここで, $\boldsymbol{f}_i$は Fの i 列目の転置ベクトルである. $\boldsymbol{f}_i$は, 新しい座標系の基底ベクトルを変換前の座標系に射影したベクトルである. 以下, $\boldsymbol{f}_i$を変換基底と呼ぶ. $\cos(\boldsymbol{f}_i, \boldsymbol{f}_j) = (\boldsymbol{f}_i \cdot \boldsymbol{f}_j)/(|\boldsymbol{f}_i||\boldsymbol{f}_j|)$ は i,j番目の変換基底間の方向余弦である. 式 (2.6) において, E_{intra}が小さくなるほど変換基底の長さは 1 に近づく. E_{intra}を弱い制約としてエネルギー関数に含めることで, 極端な座標系の伸縮に対応する座標変換行列 F が導かれるのを避けることができる.

本論文で提案する座標変換による手法では, ある方向には引き延ばされ, 別な方向には縮められるという座標変換が行われる. 大きな目的関数の壁がある場合, その大きな壁を平坦にするためにその壁と垂直方向に引き延ばす様な座標変換が導かれる. しかし, 極端な場合, 引き延ばしたい方向を無限に引き延ばすような座標変換が得られる事がある. これは$\boldsymbol{f}_i$の間で平行なものが生じるときに起きる.

その結果, 座標変換は一次独立ではなくなる. 式 (2.7) は, そのような座標変換を導かないための弱い制約である. 式 (2.7) の, E_{inter}が小さいほど各変換基底は互いに直交する方向に向くようになるため, 変換基底どうしの直交性をある程度維持できる. 変換基底は線形変換行列 Fを構成する列ベクトルなので, 各変換基底が一次独立であれば,Fは縮退しない.

以上のことから, 以下のエネルギー E_Fを最小にするような線形変換行列 Fを求めることが都合が良い.

$$E_F = C_E E_{laplacian} + C_{intra} E_{intra} + C_{inter} E_{inter} \tag{2.8}$$

ここで, C_E, C_{intra} および C_{inter}は定数パラメーターである. 具体的には線形変換行列 Fの初期値として単位行列を与え, その初期値から最急降下法によって, E_Fの値がより小さくなるような行列の値へと変化させていく. こうして求まった行列 Fを, 線形変換行列 Fとする. この計算を行うことにより, 単位行列に近く小さな E_Fの値を持った行列を計算できる. エネルギー E_Fには一般に多くの局所最適解が存在するが, その中で出来るだけ元の座標系に近い座標系への変換行列を求める. これによって, 線形変換がシステムの挙動を大きく変化させないですむと考えられる.

こうして得られた線形変換を, 従来提案されているカオスによる最適化手法 [13, 14, 26, 27] に組合せる.

以上が本論文で提案する手法である. この線形変換行列 Fを求める手続きは, 最適化問題の解を求める手続きに先立ち前処理として行われるため, 一度しか実行する必要はない.

2.3 2 次元関数への応用

本論文で提案する手法は, 変数の次元が高く局所最適解が非常に多い状況を想定している. しかし多次元において手法の効果を説明するのは困難であるので, ここでは単純な 2 次元空間における大域的最適解の探索において, 本手法がどう働くかについて説明する. 例として, 定義域 $0 \leq V_1, V_2 \leq 1$ 内で以下の 4 次の目的

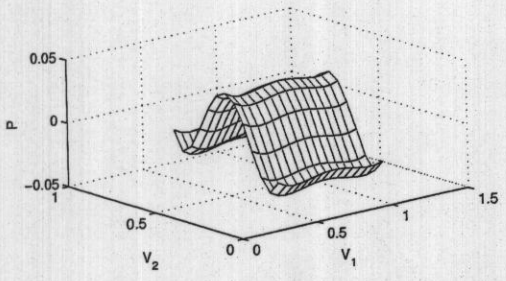


図 6 目的関数 Pの形状

関数を最小化する問題を考える. [7]

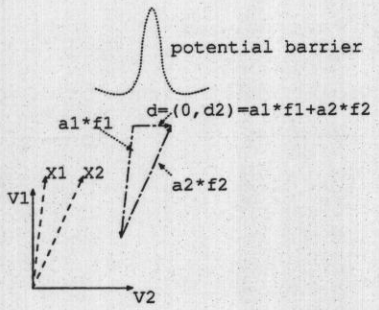
$$\begin{aligned} P(V_1, V_2) = \quad & (V_1 - 0.48)(V_1 - 0.91) \times (V_1 - 0.35)(V_1 - 0.93) \\ + \quad & 16(V_2 - 0.65)(V_2 - 0.02) \times (V_2 - 0.51)(V_2 - 0.20) \end{aligned} \tag{2.9}$$

図 6 は, この目的関数の形状を示す. この関数は 4 つの局所最適解をもつ. 大域的最適解は $(0.4, 0.1)$ である. カオスやランダム性を取入れた最急降下法によってシステムが局所最適解から脱出できるようにする場合, V_1軸と平行に存在する大きな目的関数の壁を越えるのは難しい. 例えば, 3 番目に小さい局所最適解 $(0.4, 0.6)$ の近傍から脱出して, 大域的最適解 $(0.4, 0.1)$ に到達しようとする場合, この大きな目的関数の壁を越えなければならず, たとえカオスによる非平衡ダイナミクスを用いたとしても, システムが大域的最適解を発見するのは難しい.

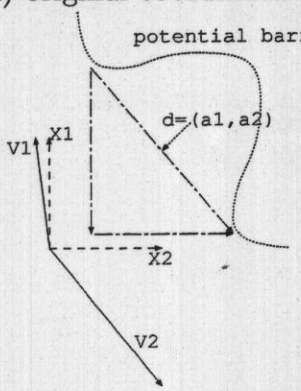
システムがこの大きな目的関数の壁を越えやすくするために式 (2.5) の線形座標変換を行う. 但し式 (2.5) において $N = 2$ であり, $\boldsymbol{V} = (V_1, V_2)^t$, $\boldsymbol{X} = (X_1, X_2)^t$ である. (2×2) 次行列 $F = [f_{ij}]$ は式 (2.8) のエネルギー E_Fが小さくなるように選ばれる. $E_{laplacian}$の積分は定義域である $0 \leq V_1, V_2 \leq 1$ の領域に対して行う.

図 7は, この E_Fが小さくなる座標変換が, どのようなものかを説明する図である. 図 7(a) では, 変換前の座標系での$\boldsymbol{f}_1$と$\boldsymbol{f}_2$ の方向を破線で示している. 今, 何らかのダイナミクスにより, システムがベクトル$\boldsymbol{d}$だけの状態変化をする状況を考える. そのときの座標変換前と変換後のそれぞれの系でのダイナミクスの違い

[7] この関数は 2 変数で 4 つの局所最適解を持つ例として作成したものであり, それ以上の特別な意味はない.



(a) original coordinate system



(b) new coordinate system

図 7 座標変換後, 変換前での基底ベクトル

について考察する．ここで，$\boldsymbol{d} = (0, d_2)$ は変換前の座標系で大きな目的関数の壁を垂直に横切る長さの短いベクトルであり，V_2軸と平行であるとする．$\boldsymbol{f}_1$と$\boldsymbol{f}_2$の方向は，大きな目的関数の壁とほぼ平行であり，$\boldsymbol{d}$とほぼ直交方向である．図 7(b) では，変換後の座標系においての$\boldsymbol{f}_1$と$\boldsymbol{f}_2$の方向を破線で表し，変換前の座標系での基底ベクトルの方向を実線で表わしている．

図 7(a) には，変換前の座標系において一次独立なベクトル$\boldsymbol{f}_1, \boldsymbol{f}_2$ の重合せで $\boldsymbol{d} = (0, d_2)$ を表現した図が示されている．$\boldsymbol{d} = a_1\boldsymbol{f}_1 + a_2\boldsymbol{f}_2$とする．ここで，$a_1, a_2$ は，ベクトル$\boldsymbol{d}$がもつ$\boldsymbol{f}_1, \boldsymbol{f}_2$の方向の長さである．$V_2$方向にほとんど成分を持たないベクトル$\boldsymbol{f}_1, \boldsymbol{f}_2$ の重合わせで V_2方向ベクトルを表そうとするため，a_1, a_2は d_2 と比較して大きな値となる．図 7(b) には，変換後の座標系での同様の図が示されている．a_1, a_2は大きな値であるため，変換後の座標系において，ベクトル$\boldsymbol{d}$の長さは変換前の座標系での値よりも伸ばされている．従って，変換後の座標系では，大きな目的関数の壁を横切る方向（すなわち，V_2軸と平行な方向）は引き延ばされている．その結果，この目的関数の壁は，変換後の座標系では緩やかな山となり，目的関数はより平坦になっている．

この座標変換を用いることにより探索手続きがいかに変わるかについて，探索法としてランダム性が含まれる最急降下法を用いた場合を例として述べる．変換後の座標系での最急降下法は，以下の式で表される．

$$\boldsymbol{X}(s+1) = \eta \nabla_X P + \boldsymbol{X}(s) \tag{2.10}$$

ここで，ηは小さな負の定数，s は離散時間である．$\nabla_X P = (\frac{\partial P}{\partial X_1}, \frac{\partial P}{\partial X_2})$ は，変換後の座標系での目的関数の勾配である．$\boldsymbol{X}(s)$ は離散時刻 s における$\boldsymbol{X}$の値である．式 (2.10) は次のようにも表記できる．

$$\boldsymbol{V}(s+1) = \eta \nabla_V^X P + \boldsymbol{V}(s) \tag{2.11}$$

ここで，$\boldsymbol{V}(s)$ は時刻 s における$\boldsymbol{V}$の値である．また，

$$\nabla_V^X P \equiv F \nabla_X P = F F^t \nabla_V P \tag{2.12}$$

$$\nabla_V P = (\frac{\partial P}{\partial V_1}, \frac{\partial P}{\partial V_2}) \tag{2.13}$$

である．$\nabla_V P$ は変換前の座標系での目的関数の勾配である．以下，$\nabla_V^X P$を変換勾配と呼ぶ．変換勾配を用いた変換前の座標系での最急降下法は，変換後の座標系での最急降下法と等価である．

一方，変換前の座標系での目的関数の定義域から変換後の座標系での定義域を求めることは，行列 Fの逆行列の計算が必要なので，特に多次元の系においては困難である．この困難を避けるため，実際の計算では，定義域の変換を行わなくても済むように，座標変換前の座標系で座標変換を行った場合と等価な繰返し計算を行う．具体的には，座標変換前の座標系での最急降下法の繰返し式

$$\boldsymbol{V}(s+1) = \eta \nabla_V P + \boldsymbol{V}(s)$$

において，勾配 $\nabla_V P$を変換勾配 $\nabla_V^X P$ に置換えた繰返し計算を行う．本章での 2 次元の例においても，同様に変換勾配を用いた計算を用いて説明する．

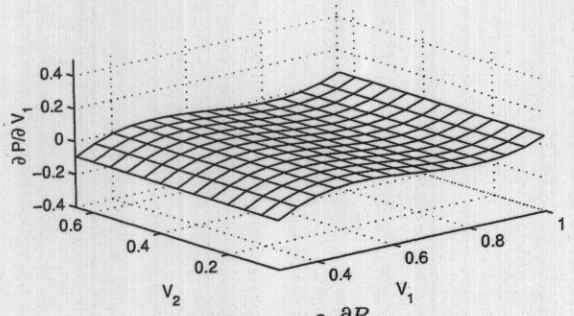
図 8 は，変換前の座標系での勾配と変換勾配を比較した図である．図 8(a), (b), (c) および (d) は，それぞれ$\frac{\partial P}{\partial V_1}$, $\frac{\partial P}{\partial V_2}$, $\nabla_V^X P$ の V_1 方向成分，および，$\nabla_V^X P$ の V_2 方向成分を表している．V_2方向の勾配が，変換勾配では緩和されているのが分かる．

この座標変換による効果を調べるため，システムの初期状態と得られる解との関係が最急降下法にランダム性が含まれることによってどう変化するかを調べた．実験は以下の様にして行った．

定義域 $0 \leq V_1, V_2 \leq 1$ の空間を 20×20 のメッシュに区切り，そのメッシュの各点を初期値として最急降下法を行い，収束する位置が 4 つのうちどの局所最適解の近くであるか調べる．その際，勾配に 10%のランダム性が含まれるようにする．そして，通常のランダム性を含まない最急降下法によって得られる局所最適解と比較を行う．

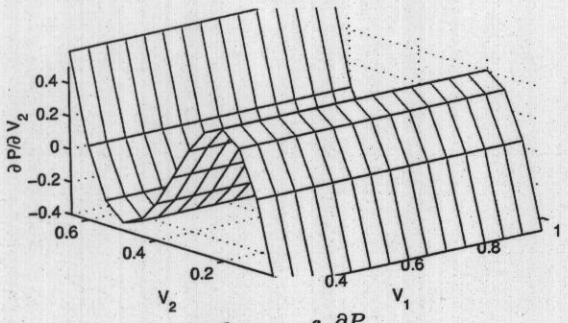
異なる局所最適解に収束したということは，勾配に加えられた 10%のランダム性が目的関数の山を越えることに有効に働いたことを示している．メッシュの各点から 1 度ずつ探索するという実験の結果，[8] 収束する局所最適解が，勾配にランダム性が含まれる場合と含まれない場合とで異なるメッシュ上の点の数は，座標変換後の系の方が 5 倍多いという結果を得た．この結果は，座標変換によって局所最適解からの脱出が起きやすくなっている事を示唆している．

[8]実験回数を増やしても結果にあまり変化はなかった．



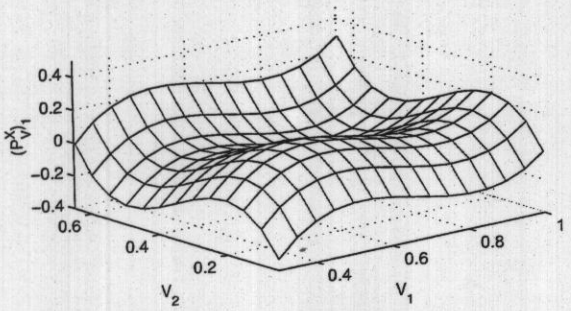
shape of $\frac{\partial P}{\partial V_1}$

(a)



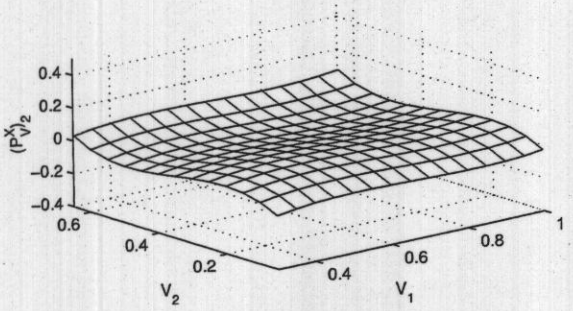
shape of $\frac{\partial P}{\partial V_2}$

(b)



shape of V_1 element of $\nabla_V^X P$

(c)



shape of V_2 element of $\nabla_V^X P$

(d)

図 8 勾配の比較

2.4 2次割当て問題への応用

本節では, 我々の提案した手法を問題として2次割当て問題, 探索手法としてカオスによる非平衡ダイナミクスを併用した2重制約ネットワーク (CDCN) [14] を用いた場合に応用する.

DCNの基本アルゴリズムは次の様にも記述できる. 導出は, 付録Eで行う.

$$\log V_{an}(s+1) = -(\frac{\partial \phi}{\partial V_{an}}(V_{an}(s)) + \tilde{\lambda}_n(s) + \tilde{\mu}_a(s))/T + \log V_{an}(s) \qquad (2.14)$$

$V_{an}(s)$, $\tilde{\lambda}_n(s)$ および$\tilde{\mu}_a(s)$ は離散時刻 s における V_{an}, $\tilde{\lambda}_n$および$\tilde{\mu}_a$の値である. 式 (2.14) は, ベクトル $\log \boldsymbol{V} = (\log V_{11}, \log V_{12} ... \log V_{NN})$ を自由エネルギーϕの傾き方向に変化させる形式になっている点で最急降下法の式と似ている. 以下, このベクトルを度合いベクトルと呼ぶ.

式 (2.14) で定義される繰返し計算を, 度合いベクトルに対する最急降下法とみなすと, 2.3と同様, 式 (2.5) で定義した線形変換によって, より局所最適解から脱出しやすいシステムにすることが可能である. $(N^2 \times N^2)$ 次の線形変換行列 F は, 式 (2.8) が小さくなるように決定される. この行列Fによって変換された式 (2.14) の繰返し計算式は以下のようになる.

$$\log V_{an}(s+1) = -(U^F_{an} + \tilde{\lambda}_n(s) + \tilde{\mu}_a(s))/T + \frac{\partial H}{\partial V_{an}} + \log V_{an}(s) \qquad (2.15)$$

$$U^F_{an} = \sum_{b,m,c,k=1}^{N} F_{anbm} F_{ckbm} U_{ck}(s) \qquad (2.16)$$

$$U_{an} \equiv \frac{\partial P}{\partial V_{an}} = \sum_{b,m=1}^{N} W_{anbm} V_{bm}(s) + I_{an} \qquad (2.17)$$

さらに, U^F_{an}を次で定義される U^F_{Can}に置換えることによってカオスダイナミクスをもつモデルが実現できる.

$$U^F_{Can}(s+1) = (1-k) U^F_{Can}(s) + k(\sum_{b,m,c,k=1}^{N} F_{anbm} F_{ckbm} (U_{ck} + \gamma V_{ck}(s))) \qquad (2.18)$$

ここで, kとγは定数パラメーターである. この線形変換された勾配法に基づくカオスニューラルネットワークを修正2重制約ネットワーク (modified chaotic doubly constrained network, MDCN) と呼び, 今回提案するモデルである.

表 2 解の改善

Name	feas.sol	CDCN	MDCN	N CDCN	N MDCN	ratio
tai50a	4941410	4%	3%	3	38	0.75
tai80a	13557864	∞	3%	0	16	0.43
tai100a	21125314	3.3%	2.7%	1	35	0.79

MDCN と座標変換の効果を入れていないカオスニューラルネットワーク (CDCN) [14] との比較結果を表 2に示す. 比較実験には, QAPLIB [6] の問題を用いた. 表 2において "Name" はベンチマーク問題の名称, "feas. sol" は現在知られている最良解の値, "CDCN" は CDCN の計算結果を知られている最良解からのずれの百分率で示したもの, "MDCN" は MDCN の計算結果を知られている最良解からのずれの百分率で示したもの, "N CDCN" は CDCN が 1000 回繰返し計算で発見できた解の数, "N MDCN" は MDCN が 1000 回繰返し計算で発見できた解の数, "ratio" は変換後の式 (2.4) の値と, 変換前の式 (2.4) の値との比を, それぞれ示す.

CDCN と MDCN において, 得られた解は 2opt 法 [6] によって改善した. "tai80a" において CDCN は 100000 回以上の繰返し計算によっても解を発見できなかったため "∞" としている. なお, 問題の名称に含まれる数字は問題の大きさ (N) を表している. 例えば, $N = 100$ である "tai100a" などは, 非常に難しいベンチマークとなっている.

表 2から, 本手法によって式 (2.4) の値が減少していることが分かる. そのため, システムはより多くの解を発見でき, 結果として得られる解が向上していることが分かる.

2.5 考察

ここでは式 (2.4) の意味について考察する.

式 (2.4) では, 変換後の座標系でのラプラシアンが, 変換前の座標系において積分されている. この意味について述べる. 本論文では, 実際の繰返し計算を変換

勾配を用いることによって変換前の座標系で行う. 従って, 式 (2.4) において変換勾配に対応したラプラシアンを変換前の座標系において積分するのが妥当であろう. 変換勾配に対応したラプラシアン $\triangle_V^X P$としては, 以下の式が考えられる.

$$\triangle_V^X P = \mathrm{trace}(C_V^X) \tag{2.19}$$

$$(C_V^X)_{ij} = \frac{\partial}{\partial V_i}(\nabla_V^X P)_j \tag{2.20}$$

ここで, $(\nabla_V^X P)_j$は変換勾配 $\nabla_V^X P$の第 j成分, $(C_V^X)_{ij}$は行列 C_V^Xの (i,j) 成分である. trace は行列のトレースを表す. 線形な座標変換のみを考える場合には以下のように計算できる.

$$\begin{aligned}
\triangle_V^X P &= \mathrm{trace}(C_V^X) \\
&= \sum_{i,j,k=1}^{N} \frac{\partial}{\partial V_i} F_{ij} F_{kj} \frac{\partial}{\partial V_k} P \\
&= \sum_{j=1}^{N} (\sum_{i=1}^{N} F_{ij} \frac{\partial}{\partial V_i})(\sum_{k=1}^{N} F_{kj} \frac{\partial}{\partial V_k}) P \\
&= \sum_{j=1}^{N} (\sum_{i=1}^{N} \frac{\partial V_i}{\partial X_j} \frac{\partial}{\partial V_i})(\sum_{k=1}^{N} \frac{\partial V_k}{\partial X_j} \frac{\partial}{\partial V_k}) P \\
&= \sum_{j=1}^{N} \frac{\partial^2}{\partial X_j^2} P = \triangle_X P
\end{aligned}$$

従って, 変換後の座標系でのラプラシアンの二乗を変換前の座標系において積分する式 (2.4) は, $\triangle_V^X P$の二乗を変換前の座標系で積分する式と等しい. これから, 式 (2.4) の値を小さくする座標変換は目的関数をより平坦にする座標変換であると考えられる.

次に, 線形な座標変換がシステムを改善すると考えられる理由について述べる. 本手法により求められる座標変換は, 式 (2.4) のエネルギーが小さくなり, かつ, 式 (2.6),(2.7) を小さくするような意味のあるようなものであった. 目的関数の曲率行列が$\boldsymbol{V}$に関わらず定数であるとすると (例えば, QAP の場合がそうである), 式 (2.4) は以下のように書ける.

$$C(\sum_{i=1}^{N} \lambda_i)^2 \tag{2.21}$$

ここで, λ_iは目的関数の曲率行列の固有値, Cは$\int dV$に相当する定数である. 弱い制約である式 (2.6) と (2.7) を用いることにより, 変換前の座標系と変換後の座標系における体積の比をほぼ一定に保ちやすい. 従って, 固有値λ_iの積は, 変換前と変換後とであまり変わらない. すなわち, 2.2で定義した操作は, 曲率行列の固有値の積をほぼ一定に保ちながら, 固有値の和を最小化する操作に近いことが分かる. この操作により, 各固有値の値は互いに近い値になる.

目的関数の曲率行列が定数であるとき, 目的関数は曲率行列の固有ベクトルの方向に対応する固有値の大きさの山をもっている. 曲率行列の固有値の中に他の固有値と比べて非常に大きな値が含まれる場合, 目的関数は, 対応する固有ベクトルの方向に非常に大きな目的関数の壁をもつ. この大きな目的関数の壁をシステムは乗り越えにくいので, 全ての空間を探索するのが困難となる. 座標変換の結果, 各固有値の値を比較的近い値にできるため, 大きな目的関数の壁は全方向に平均化され, 状態はどの方向に移動するのも比較的容易になり, システムはより広い空間を探索しやすくなる. 従って, より多くの局所最適解を探索することができ, 結果として良い解が発見できる.

座標変換によって目的関数の勾配が平均的に小さくなるため, 1 つの局所最適解近傍にとどまる時間は小さくなる. また, より広い空間を探索することができるようになるため, 同じ計算時間内において, 同じ局所最適解を通る確率が少なくなる. これらによって, より多くの解を発見できる一方で, 計算時間が長くなることはない.

2.6 結論と課題

組合せ最適化問題への応用において, カオスによる非平衡ダイナミクスが, より多くの解を発見できるように座標変換をする手法を提案した. この手法により目的関数の滑らかさを表すラプラシアンの空間的平均値が下げられた.

この手法の具体的な適用アルゴリズムとしてさらにカオス的 2 重制約ネットワークモデルに適用したアルゴリズム提案した. 新しいモデルによると座標変換の効果により以前より多くの局所最適解を探索でき, 結果としてより良い 2 次割当て問題の解を発見することができた. なお本論文では座標変換として線形変換

に限ったが, 非線型変換への拡張については今後の課題である.

3. ニューラルネットによるλ-opt アルゴリズムを使った 2 次割当て問題の解法

3.1 はじめに

本節では, 我々が提案したニューラルネットワークによる新しい組合せ最適化問題の解法について説明する. 特に QAP [6] を扱う. しかし, 我々の提案する手法は, 解が順列で表せる様々な問題にも適用可能であり, 例えば TSP などにも適用可能である.

本研究で提案する新しい手法では, 順列への置換操作に注目する. 順列の入れ替え操作により解を改善する手法は局所ヒューリスティクスとして研究されてきた. 順列の 2 つの要素の入れ替え操作しか考えない手法は,2-opt 探索 [23] や 2-exchange と呼ばれる. $\lambda > 2$ 以上の要素の入れ替えを行う同様の手法はλ-opt 探索と呼ばれる. 2-opt 探索は, 非常に簡単な局所ヒューリスティクスであるが, すぐに局所最適解に収束してしまう. 2-opt に基づいたニューラルネットワークによる手法は, 長谷川ら [9] によってすでに提案されている. 2-opt 探索が局所最適解にトラップされるのを防ぐための手法としては, タブーリストを使う手法 [34] やカオス [9] を使う手法が提案されている. 一方,λ-opt ($\lambda > 2$) ヒューリスティクスは,2-opt 探索では脱出できない局所最適解からの脱出を可能とするが,λの値が大きくなると非常に計算時間のかかる手法となってしまう.

我々の提案する手法は, このλ-opt ヒューリスティクスのアナログ版である. 本手法は DCN 法に基づいたλ-opt のアナログニューラル手法であるため, λの値として 15 ~ 60 の値を取ることが可能となる. 従って,λの値に応じた中程度に遠い空間を探索して, その中で良い解を発見できるようになる. また, ある程度大きなλの値を使うことによって, 浅い局所最適解からの脱出も可能となる.

評価のために QAPLIB [6] の比較的大きな問題 ($N = 80 \sim 150$) に対して計算機実験を行った. 我々の新しい手法は, 現在のチャンピオンのアルゴリズムであるタブーサーチに基づく手法 [33, 4] , 遺伝アルゴリズムに基づく手法 [8] , やシュミレーッテドジャンピングと呼ばれる手法 [3] とほぼ同程度の良い解を計算できる. さらに 2 つのベンチーマーク問題に対しては, 我々の手法は現在のチャンピオン

よりも良い解を計算できる. 結果として, 我々の手法はQAPに対する最も良いアルゴリズムの1つであると考えられる.

3.2 λ-DCN

3.2.1 λ-DCN

QAPを扱うために,$(N \times N)$-次元の配置行列$\boldsymbol{S}$を扱うのがDCNであった. 本研究では, 配置行列$\boldsymbol{S}$で表現された順列への置換操作を同様の行列で表現する方法に基づくアルゴリズムを提案する. そのような行列を以下で定義し, 置換行列と呼ぶことにする.

$$Y_{a,b} = \begin{array}{ll} 1 & (\text{要素 } p(b) \text{ が順列の } a \text{ 番目に移動する時}) \\ 0 & (\text{それ以外}) \end{array}$$

配置行列$\boldsymbol{S}$は,$p(n) = n$ となるような順列に対する置換行列であるとみなすこともできる. しかし, 説明の都合上, 以下では配置行列と置換行列を異なるものとみなす. 置換行列に対しても配置行列と同様の制約条件が存在する.

$$\sum_{a=1}^{N} Y_{a,b} = 1 \ \ (b = 1, ..., N) \tag{3.1}$$

$$\sum_{b=1}^{N} Y_{a,b} = 1 \ \ (a = 1, ..., N) \tag{3.2}$$

さらに, 次の制約条件も考えることにする.

$$\sum_{a=1}^{N} Y_{a,a} = N - \lambda \tag{3.3}$$

ここで,λは $0 \leq \lambda \leq N$となる様な整数の定数である. 式 (3.3) は, 置換$\boldsymbol{Y}$によって入れ換えられる要素の数をλ個となる様に制限する.

以下で, 式 (3.1),(3.2),(3.3) を満足する様な置換行列をλ-置換行列と呼ぶ. 置換操作$\boldsymbol{Y}$をある順列$\boldsymbol{S}$に行なった結果, できる新しい順列を表す配置行列$\boldsymbol{S}'$は, 次の様に行列の積を使って書ける.

$$\boldsymbol{S}' = \boldsymbol{Y}\boldsymbol{S} \tag{3.4}$$

次にλ-置換行列の集合$\Lambda = \{(Y^k, P^k)|k = 1, ...\}$を考える．ここで$P^k$は，置換$\boldsymbol{Y}^k$が生じる確率である．この集合からのλ-置換行列によって置換された配置行列のアンサンブル平均は次で表せる．

$$< \boldsymbol{S}' >_\Lambda = < \boldsymbol{Y} >_\Lambda \boldsymbol{S} \tag{3.5}$$

ここで$< \cdot >_\Lambda$はλ-置換行列の集合Λに対する平均を表す．すなわち，$< f(k) >_\Lambda = \sum_k f(k)P^k$である．以下で$\boldsymbol{V} \equiv < \boldsymbol{S}' >_\Lambda, \boldsymbol{X} \equiv < \boldsymbol{Y} >_\Lambda$とする．

この新しい表記法を使って，QAP を目的関数E_{obj}の最小化問題として定義することを考える．そのためには，式 (3.4) の$\boldsymbol{S}'$に対して定義される次の 2 次関数を，E_{obj}として考えればよい．

$$E_{obj}(\boldsymbol{S}') = \frac{1}{2} \sum_{a,n,b,m=1}^{N} W_{a,n;b,m} S'_{a,n} S'_{b,m} + \sum_{a,n=1}^{N} J_{a,n} S'_{a,n} \tag{3.6}$$

ここで，重み行列$\boldsymbol{W}$は対称と仮定する．すなわち，$W_{a,n;b,m} = W_{b,m;a,n}$である．

DCN と同様に，自由エネルギーϕとエントロピーHを用いて，この目的関数の最小化問題を記述しなおしてみる．ただしDCN と異なり，配置行列の期待値$\boldsymbol{V}$を求めるのではなく，目的関数を最小化するような置換行列の平均値$\boldsymbol{X}$を求めるアルゴリズムを考える．

この場合，次の近似を導くことができる．

$$\begin{aligned}
< E_{obj}(\boldsymbol{S}') >_\Lambda &\approx \frac{1}{2} \sum_{a,n,b,m=1}^{N} W_{a,n;b,m} V_{a,n} V_{b,m} + \sum_{a,n=1}^{N} J_{a,n} V_{a,n} \\
&= \frac{1}{2} \sum_{a,b,c,d=1}^{N} \Big(\sum_{n,m=1}^{N} W_{a,n,b,m} S_{c,n} S_{d,m} \Big) X_{a,c} X_{b,d} + \sum_{a,b=1}^{N} \Big(\sum_{n=1}^{N} J_{a,n} S_{b,n} \Big) X_{a,b} \\
&= \frac{1}{2} \sum_{a,b,c,d=1}^{N} W^*_{a,b,c,d} X_{a,b} X_{c,d} + \sum_{a,b=1}^{N} J^*_{a,b} X_{a,b} \\
&\equiv E^*(\boldsymbol{X}; \boldsymbol{S})
\end{aligned} \tag{3.7}$$

ここで，

$$W^*_{a,b,c,d}(\boldsymbol{S}) \equiv \sum_{n,m=1}^{N} W_{a,n,c,m} S_{b,n} S_{d,m} \tag{3.8}$$

$$J^*_{a,b} \equiv \sum_{n=1}^{N} J_{a,n} S_{b,n} \tag{3.9}$$

とした. 新しい重み行列も対称となる点に注意する. すなわち $W^*_{a,b,c,d} = W^*_{c,d,a,b}$ である. 式 (3.7) の第 1 行目において, 平均場近似 [5, 28] と同様の近似を行った.

一方, 平均場近似と同様の近似により計算される, 式 (1.35) のエントロピー H には, 次の計算を行うことができる.

$$\begin{aligned} H &\approx -\sum_{a,n=1}^{N} V_{a,n} log V_{a,n} \\ &= -\sum_{a,n=1}^{N} X_{a,\pi^{-1}(n)} log X_{a,\pi^{-1}(n)} \\ &= -\sum_{a,b=1}^{N} X_{a,b} log X_{a,b} \end{aligned} \tag{3.10}$$

ここで,πは$\boldsymbol{S}$で表される順列を表す.

置換行列への制約条件 (3.1),(3.2),(3.3) と同様の条件が$\boldsymbol{X}$についても現れる. この点に注意して, 新しい変数$\boldsymbol{X}$を使って問題を書き直すと, 次のようになる.

$$\text{minimize} \quad \phi^*(\boldsymbol{X};\boldsymbol{S}) = E^*(\boldsymbol{X};\boldsymbol{S}) - TH(\boldsymbol{X}) \tag{3.11}$$

$$\text{subject to} \quad \sum_{a=1}^{N} X_{a,b} = 1 \ (b = 1, ..., N) \tag{3.12}$$

$$\sum_{b=1}^{N} X_{a,b} = 1 \ (a = 1, ..., N) \tag{3.13}$$

$$\sum_{a=1}^{N} X_{a,a} = N - \lambda \tag{3.14}$$

この新しい問題を解くことによって目的関数を最小化するような置換行列の平均値を探すことができる. この問題は 2 次割り当て問題の最適解を大域的に探す問題ではなく, 直前に考えていた順列$\boldsymbol{S}$に依存した領域内 (この領域を以下, 順列$\boldsymbol{S}$のλ-opt 近傍と呼ぶ) での局所最適解を局所的に探す問題となっている点にも注意する. 問題 (3.11),(3.12),(3.13),(3.14) において, 式 (3.10) のエントロピーと, 式 (3.12),(3.13) の制約条件を考えると, 置換の平均値$\boldsymbol{X}$は, 定義された領域 $[0, 1]^{N^2}$ から出ていかない. 従って, エントロピーはバリヤー関数 [24] として働いていると

見なすこともできる. ここで, この連続な問題に対する定数λは, 整数である必要はなくなっている点に注意する.

DCN [12] と同様にラグランジュの未定係数法を使うと, 問題 (3.11),(3.12),(3.13),(3.14) の解は次の連立方程式の解として与えられるようになる. 導出は付録に示す.

$$X_{a,b} = \frac{U_{a,b}}{\alpha_a \beta_b \gamma^{\delta_{a,b}}} \tag{3.15}$$

$$U_{a,b} \equiv \exp(-\frac{1}{T}\frac{\partial E^*(\boldsymbol{X};\boldsymbol{S})}{\partial X_{a,b}}) \tag{3.16}$$

$$\alpha_a = \sum_b \frac{U_{a,b}}{\beta_b \gamma^{\delta_{a,b}}} \tag{3.17}$$

$$\beta_b = \sum_a \frac{U_{a,b}}{\alpha_a \gamma^{\delta_{a,b}}} \tag{3.18}$$

$$\gamma = \frac{1}{N-\lambda}\sum_{a=1}^{N}\frac{U_{a,a}}{\alpha_a \beta_a} \tag{3.19}$$

ここで$\alpha_a \ \ (a = 1, ..., N), \beta_b \ \ (b = 1, ..., N), \gamma$はラグランジュの未定係数に, 簡単な変数変換をしたものである. 式 (3.7) で定義されるエネルギーを考えるならば, $\frac{\partial E^*}{\partial X_{a,b}} = \sum_{c,d=1}^{N} W^*_{a,b,c,d} X_{c,d} + J^*_{a,b}$である. 式 (3.15),(3.16),(3.17),(3.18),(3.19) を以下λ-DCN 方程式と呼ぶ.

3.2.2 基本アルゴリズム

λ-DCN 方程式の解を得るために次の基本アルゴリズムを提案する.

1. $t = 0$ とする.$\boldsymbol{X}(0)$ に次の値を代入する.

$$X_{a,b}(0) = \begin{cases} (1-\lambda/N)(1+\epsilon_{a,b}) & \text{if } a = b \\ \frac{\lambda}{N(N-1)}(1+\epsilon_{a,b}) & \text{if } a \neq b \end{cases}$$

ここで,$\epsilon_{a,b}$は, 区間 $[-0.1, 0.1]$ の一様乱数である. また

$$\alpha_a(0) = \phi_a \ \ (a = 1, ..., N) \tag{3.20}$$

$$\gamma(0) = 1 \tag{3.21}$$

とする. ここでϕ_aは区間 $[0, 1]$ の一様乱数である.

2. すべての a, b について以下を計算する.

$$U_{a,b}(t+1) = \exp(-\frac{1}{T}\frac{\partial E^*(\boldsymbol{X}(t);\boldsymbol{S})}{\partial X_{a,b}(t)}) \tag{3.22}$$

3. α_a^{old}に$\alpha_a(t)$ を代入する.
   β_b^{old}に$\sum_{a=1}^{N} \frac{U_{a,b}(t+1)}{\alpha_a(t)\gamma(t)^{\delta_{a,b}}}$を代入する.
   γ^{old}に$\gamma(t)$ を代入する.

4. 次の計算を繰り返しする.

   (a) 全ての a について次の計算をする

$$\alpha_b^{new} = \sum_{b=1}^{N} \frac{U_{a,b}(t+1)}{{\beta_b}^{old}{\gamma^{old}}^{\delta_{a,b}}}$$

   (b) 全ての b について次の計算をする

$$\beta_b^{new} = \sum_{a=1}^{N} \frac{U_{a,b}(t+1)}{{\alpha_a}^{new}{\gamma^{old}}^{\delta_{a,b}}}$$

   (c) 次の計算をする.

$$\gamma^{new} = \frac{1}{N-\lambda}\sum_{a=1}^{N} \frac{U_{a,a}(t+1)}{\alpha_a^{new}{\beta_a}^{new}} \tag{3.23}$$

   $\alpha_a, \beta_b, \gamma$の値が収束したならば, その値を$\alpha_a(t+1), \beta_b(t+1), \gamma(t+1)$ に代入する. その後で$\boldsymbol{\alpha}(t+1)$ が $\sum_{a=1}^{N} \alpha_a = 1$ となるように$\boldsymbol{\alpha}(t+1)$ をスケールする.

5. 全ての a, b について以下を計算する.

$$X_{a,b}(t+1) = \frac{U_{a,b}/(\alpha_a(t+1)\gamma(t+1)^{\delta_{a,b}})}{\sum_c U_{c,b}/(\alpha_c(t+1)\gamma(t+1)^{\delta_{c,b}})}$$

6. t に 1 を加えてステップ (2) に戻る.$\boldsymbol{X}$が収束すれば終了する

このアルゴリズムは DCN [12] の様には収束する必要はない. しかし, 式 (3.11) で定義される自由エネルギーは, この基本アルゴリズムの 1 ステップごとに減っていく傾向にある.

3.2.3 λ-DCN アルゴリズム

温度 Tが小さい時には, 基本アルゴリズムによって得られる平均置換行列$\boldsymbol{X}$の値は,$\boldsymbol{X}$の定義域である単位超立方体の格子点に近い値となる. 従って, この行列は1つのλ-置換行列を表しているとみなせる. さらに, 基本アルゴリズムの繰り返し計算の間に, 多くのλ-置換行列を得ることができる. これは基本アルゴリズムが離散ダイナミクスを持つためである. こうして得られた多くのλ-置換行列の中で, その結果得られる配置行列$\boldsymbol{S}'$について目的関数 (3.6) を最も下げることができる置換行列を選ぶ. この方法により, 以前の配置行列をよりよく改善する置換行列を求められることが期待できる.

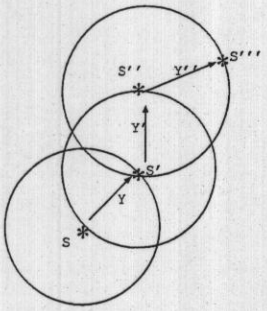
我々の提案する手法である,λ-DCN アルゴリズムは, 上述したプロセスを繰り返す手法である. 図 9は, この手法の概念図を示したものである. 基本アルゴリズムが終わると, 以前の配置行列$\boldsymbol{S}$から新しい配置行列$\boldsymbol{S}'$へのλ-置換行列$\boldsymbol{Y}$が求まる. 新しい配置行列$\boldsymbol{S}'$は, 以前の配置行列$\boldsymbol{S}$と比べてより小さな目的関数の値を持つように改善された行列であると期待できる. そして次に, 配置行列$\boldsymbol{S}'$からさらに改善された配置行列$\boldsymbol{S}''$へのλ-置換行列$\boldsymbol{Y}'$が求まる.

$\lambda = 2$ とするならばλ-DCN アルゴリズムは,2-opt 探索のアナログ版と見なせる. 2-opt は, 順列の 2 要素を入れ替えることによって得られる最近接な近傍しか探索しない手法である. 一方,λの値として比較的大きな値を選ぶと,λ-DCN アルゴリズムは中距離探索を実現できるようになる. この中距離探索は, 近傍にある浅い局所最適解を無視する手法であるとも考えられる. また,λ-DCN アルゴリズムはλ-置換行列の繰り返し適用であり, 一般に収束しない. すなわち非平衡ダイナミクスであると言える. このことは言い換えれば, システムが局所最適解から脱出できることを示している.

3.3 λ-interior DCN アルゴリズム

λ-DCN アルゴリズムは解を改善できるλ-置換行列を探す手法であるため,λ-置換行列で表せない, より近傍にある配置行列が良い局所最適解である場合には, その局所最適解を無視してしまう.

この問題を解決する為,(3.3) の等式制約条件を次の不等式制約条件に置き換え



この図はλ-DCN を概念的に示した図である. ある順列 (配置行列$\boldsymbol{S}$) のλ-opt 近傍を$\boldsymbol{S}$を囲む円で示している. まず順列$\boldsymbol{S}$のλ-opt 近傍内で, 平均的に最も目的関数E_{obj}を下げられるような順列への置換$\boldsymbol{Y}$を求める. 次に置換$\boldsymbol{Y}$によって置換された新たな順列$\boldsymbol{S'}$のλ-opt 近傍を同様に探索し, 新たな順列$\boldsymbol{S''}$を求める. この操作を繰り返していく.

図 9 λ-opt 近傍の探索

たアルゴリズムを提案する.

$$\sum_{a=1}^{N} Y_{a,a} \geq N - \lambda \tag{3.24}$$

この制約条件により,λよりも小さな数の要素しか入れ替えない様な置換行列$\boldsymbol{Y}$を表すことができる[9]. この制約条件を扱うために次のようなバリヤー関数を導入する.

$$K^*(\boldsymbol{X}) = T(\sum_{a=1}^{N} X_{a,a} - M)log(\sum_{a=1}^{N} X_{a,a} - M) + T(\theta - 1)(\sum_{a=1}^{N} X_{a,a} - M) \tag{3.25}$$

ここで,$M \equiv N - \lambda$であり,θは 0 ~ 1 の定数パラメータである. このバリヤ関数を導入するかわりに, 式 (3.14) の制約条件を取り除く. 従って,λ-DCN 方程式の式 (3.16),(3.19) を次の式に入れ替える.

$$U_{a,b} = \exp(-\frac{\partial E^*(\boldsymbol{X};\boldsymbol{S})}{T\partial X_{a,b}} - \theta\delta_{a,b}) \tag{3.26}$$

[9]この制約条件は, 中心となる順列の k-exchange 近傍を表現している. ここで $k = \lambda$である. したがってλ-interior DCN は,k-exchange DCN と呼ぶこともできるだろう.

$$\gamma = \frac{\sqrt{M^2 + 4\sum_a \frac{U_{a,a}}{\alpha_a \beta_a}} - M}{2} \tag{3.27}$$

この式の導出は付録で行う. λ-DCN 方程式が変えられた結果, 基本アルゴリズムの式 (3.22) は, 次の式で入れ替えられる.

$$U_{a,b}(t+1) = \exp(-\frac{\partial E^*(\boldsymbol{X}(t);\boldsymbol{S})}{T\partial X_{a,b}(t)} - \theta\delta_{a,b}) \tag{3.28}$$

同様に, 基本アルゴリズムの式 (3.22) は次の式で置き換えられる.

$$\gamma^{new} = \frac{\sqrt{M^2 + 4\sum_a \frac{U_{a,a}(t+1)}{\alpha_a^{new} \beta_a{}^{new}}} - M}{2} \tag{3.29}$$

アルゴリズムの他の部分はλ-DCN と同じである.

この修正された基本アルゴリズムは, 配置行列を改良できるより良い l-置換行列 $(0 \leq l \leq \lambda)$ を探す. 従って, この手法は局所的な探索と中距離探索を同時に行っているアルゴリズムであると見なせる. この手法をλ-interior DCN と呼ぶことにする.

3.4 計算機実験

本節では, 本研究で提案したアルゴリズムによる計算実験結果を示す. 本論文で提案したアルゴリズムは, 解が順列で表現できるどんな 2 次問題にも適用可能である. 具体的に QAP の場合は, $W_{a,n;b,m} = D_{a,b}C_{n,m}$ $(a,n,b,m = 1,...,N)$ $I_{a,n} = \eta$ $(a,n = 1,...,N)$ $W_{a,n;a,n} = -\eta$ とする. ここでηは基本アルゴリズムの収束を速める為のパラメーターであり, 小さな正の値を用いた. λの値は問題の大きさによって変えるが, 本研究では問題に応じて 15 ~ 60 の値を使った.

計算実験には QAPLIB [6] の "Tai80a"$(N = 80)$"Tai100a"$(N = 100)$ "Wil100"$(N = 100)$ "Tho150"$(N = 150)$ というベンチマークを使った. 都市の間の距離を表す行列, 工場の間の流量を表す行列は, すべて対称で密な行列を扱っている. "Tho150" は,QAPLIB の中で密な行列要素をもつ最も大きな問題である. "Tai80a","Tai100a" に対する現在のチャンピオンアルゴリズム [33, 4] は, タブサーチに基づく手法である. "Wil100" に対する現在のチャンピオンアルゴリズム

表 3 最良解

Name	N	feas.sol	algorithm
tai80a	80	13557864	robust tabu search
tai100a	100	21125314	reactive tabu search
wil100	100	273038	genetic hybrids
tho150	150	8133484	SIMJ

Name	DCA	2 opt	λ-DCN	λ-interior DCN
tai80a	1.2%	3.1%	-0.060% (2568)	-0.040% (2481)
tai100a	5.7%	3.2%	-0.015% (688)	-0.082% (5656)
wil100	0.27%	0.73%	0.46% (2109)	0.070% (2563)
tho150	0.33%	1.5%	0.24%(7139)	0.28% (6956)

[8] は, 遺伝的アルゴリズムに基づく手法である. "Tho150" に対する現在のチャンピオンアルゴリズム [3] は, シュミレーテッドジャンピングと呼ばれる手法である.

表 3に計算実験の結果を示す. 表のそれぞれの値はチャンピオンデータからのずれを 100 分率で示したものである. 比較の為に, 我々が以前に提案した手法 [12] である DCN アニーリング (DCA) による実験値と,2-opt ヒューリスティックを何度も実行した結果得られた中で最良な解も同様に示した. この表より,DCA,λ-DCN,λ-interior DCN によって得られる解は 2-opt ヒューリスティックよりも良いことが分かる. λ-DCN,λ-interior DCN に対しては最良解と, その最良解を得るまでにかかった基本アルゴリズムの繰り返し回数も示す.

表 3より, 我々の提案する手法は非常に良い結果を得られることが分かる. "Wil100", "Tho150" に対しては現在のチャンピオンのデータとほぼ同じ程度の値が得られている. "Tai80a", "Tai100a" に対しては我々の提案したλ-DCN,λ-interior DCN の両方の手法ともに以前のチャンピオンによる解よりも良い解を得ている.

3.5 考察

制約条件式 (3.3) を考えないでλ-DCN と同様のシステムを構成することも可能であるが, 計算機実験では制約条件式 (3.3) のないシステムで余り良い結果を得ることができなかった. 本節では, 制約条件式 (3.3) の意味について考察する.

制約条件式 (3.3) を考えないシステムは, オリジナルの DCN [12] とほぼ等価なシステムであるとみなすことができる. DCN では, 順列 p の a 番目が n である確率を表す確率変数 $V_{a,n}$ を使って,QAP の解を表現する. すなわち変数 $\boldsymbol{V}$ は配置行列 $\boldsymbol{S}$ のアンサンブル平均である.

DCN は, 式 (1.55) で定義されるラグランジュ関数の停留点を, 式 (2.14) の最急降下法と似ている繰り返し計算によって計算する手法であったことを思い出そう.

式 (1.51) で与えられる自由エネルギー関数の停留点は, 温度 T が十分に小さくないと, エントロピーの影響によって, 目的関数 E_{obj} の極小点から離れた定義域の内点になってしまう. 一方で温度が小さくなると, 自由エネルギー関数は多くの極小点をつようになるが, 低温での繰り返し計算式 (2.14) では, 特に $W_{a,n,a,n} = 0$ の場合に,$V_{a,n}$ は, ほとんど 0 か 1 近くの値をとるようになる. この際, 図 10に示す様な問題が生じる. 図 10において横軸は 2 つの変数 $(V_{a,n}, V_{b,m})$ についての 2 次元の状態空間を模式的に表し, 縦軸は目的関数 E_{obj} の値, 曲線が目的関数 E_{obj}, 点線は start 点でみた E_{obj} の局所勾配を表す. この図の場合の様に $(V_{a,n}, V_{b,m}) = (0.5, 0.5)$ 付近に停留点がある場合は, 低温での $V_{a,n}, V_{b,m}$ が 0 か 1 の近くしかとらないために, 勾配を用いた探索によっては, 停留点を通り越してしまうことが起こる. 具体的には, 図 10の start という点で目的関数 E_{obj} の勾配をみると,start 点から jump 点方向に変化した方が目的関数の値を下げられると判定してしまい, 式 (2.14) のダイナミクスはそれを実行するが, 実際に start 点から jump 点に移動させてみると, 勾配から予想した jump2 という点には移らず jump という点に移動してしまうことになり, 目的関数 E_{obj} の値は勾配から予想したように下がるのではなく, 逆に上がってしまう. [1]

[1] この問題を, 温度を下げながら極小点を発見していくアニーリングによって解決しようとしているのが,DCA [12] である. 温度を下げていくと自由エネルギー関数 (1.51) の極小点は分岐していくが,DCA はその分岐の時に極小点の選択を行う方法であり, 低温で多く存在する極小点を, 全て探索しなくてもよい点が優れている.

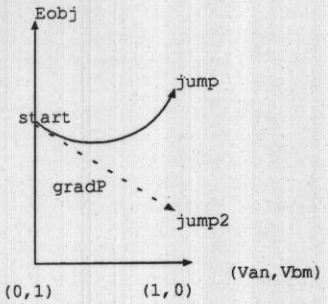


図 10　目的関数の勾配だけで状態を変化させた時の目的関数値が予想できない場合

制約条件式 (3.3) だけを除外してλ-DCN と同様のシステムを作ったとしても, 同様の現象が生じる. 上の説明は, 便宜上 2-opt 近傍を用いたが, 実際にはλ-opt 近傍についてλが大きくなる程, この問題は深刻となる. このことをさらに説明する.

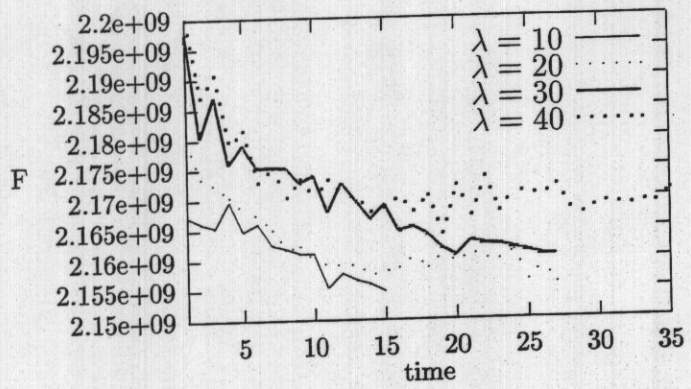
位置 a に要素 n が配置される確率 $V_{a,n}$が 0 から 1 へと変化する時の目的関数 E_{obj}の値の変化$\Delta E_{obj}(V_{a,n}=0\rightarrow 1)$ および,$V_{a,n}$が 1 から 0 へ変化する時の目的関数 E_{obj}の値の変化$\Delta E_{obj}(V_{a,n}=1\rightarrow 0)$ は, 以下で計算できる.

$$\begin{aligned}\Delta E_{obj}(V_{a,n}=0\rightarrow 1) &= \frac{\partial E_{obj}}{\partial V_{a,n}}\\ \Delta E_{obj}(V_{a,n}=1\rightarrow 0) &= -\frac{\partial E_{obj}}{\partial V_{a,n}}\end{aligned}$$

つまり$\Delta E_{obj}(V_{a,n}=0\rightarrow 1)$,$\Delta E_{obj}(V_{a,n}=1\rightarrow 0)$ は$\frac{\partial E_{obj}}{\partial V_{a,n}}$から直接計算できる. 従って,λ-opt 近傍へ順列が変化した時の目的関数 E_{obj}の変化量も $\nabla_V E_{obj}\cdot\delta V$で計算できると予想できる. ここで $\nabla_V E_{obj}$は E_{obj}の勾配ベクトル,δVは状態変数 V の変化量ベクトルを表す.

では,2-opt 近傍のみを考えた場合の目的関数の変化量を実際に計算してみる. ここでは$\boldsymbol{V}$が順列 p を表現しているとする. $s=p(e)$ と $t=p(f)$ が入れ替えられる場合の目的関数の変化量は

$$\begin{aligned}&E_{obj}(V_{e,s}=1\rightarrow 0, V_{f,t}=1\rightarrow 0, V_{e,t}=0\rightarrow 1, V_{f,s}=0\rightarrow 1)\\ &=\frac{\partial E_{obj}}{\partial V_{e,t}}+\frac{\partial E_{obj}}{\partial V_{f,s}}-W_{e,t;f,s}-W_{f,s;e,t}-(\frac{\partial E_{obj}}{\partial V_{e,s}}+\frac{\partial E_{obj}}{\partial V_{f,t}}-W_{e,s;f,t}-W_{f,t;e,s})\end{aligned}$$



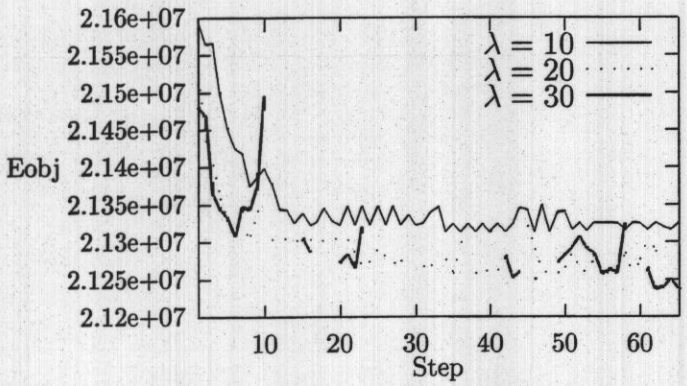
一回の基本アルゴリズムを実行した時の, 自由エネルギーの時間変化をプロットしたグラフである. グラフは$\lambda = 10, 20, 30, 40$ の 4 つの場合について示した. $\lambda = 40$ 以外の場合は,35 ステップ以内で基本アルゴリズムが収束しているのでグラフが途中で終わっている.

図 11 基本アルゴリズム内での自由エネルギーの時間発展

となり,$-(W_{e,t;f,s} + W_{f,s;e,t}) + (W_{e,s;f,t} + W_{f,t;e,s})$ だけ, 勾配から予想できる値 $\nabla_V E_{obj} \cdot \delta V$からずれている. 同様にλ-opt 近傍内での目的関数の変化量の計算式を作っていくと, λの値を増やしていくほど, このようなずれを表す項の数は増加し, 上で述べた予想からのずれが大きくなる. すなわちλが大きくなるにしたがって, 勾配法 (2.14) が目的関数を下げる確率は小さくなると考えられる. このことを確かめるために計算機実験を行った結果が図 11である. 一回の基本アルゴリズムにおける自由エネルギーの時間変化は全体的に下がる傾向にはあるが, 時として増加する. 図 11から, この自由エネルギーが増加してしまう現象は, λの値が大きい時ほど, 頻繁に起こることが確認できる.

制約条件 (3.3) がないシステムでは, できるだけ多くの要素を入れ替えて, すなわち, 非常に大きなλの値を取る様にして, より大きく目的関数の値を下げることを試みるであろう. しかし, これは上記の理由により実際はあまり成功しない.

以上の議論からλの値が小さい程, 自由エネルギーの極小化という点からは良いことになる. しかし,λの値が小さい場合, 一回の基本アルゴリズムで探索できる



基本アルゴリズムを繰り返し実行した時の E_{obj} の値を, 基本アルゴリズムの繰り返しステップ数のグラフで表した図である. グラフは $\lambda = 10, 20, 30$ の 3 つの場合について示してある. 基本アルゴリズムは, いつも新しい解を発見できるわけではない. 基本アルゴリズムが, 解を発見できないステップにはグラフがプロットされていないため, グラフは途切れている.

図 12 λ-DCN アルゴリズムの時間発展

λ-opt 近傍が狭いため, 基本アルゴリズムを繰り返し実行すると, すぐに局所最適解に収束してしまう. この様子を示したのが図 12である. 図 12を見ると分かるように, λの値が小さい場合ほど, すぐに E_{obj}の値が, ある一定値付近におちついてしまう傾向がある. λの値が小さいため, 局所最適解からの脱出が難しいからである.

このようにλ-DCN は, 自由エネルギーの緩和と非平衡ダイナミクスによる局所最適解からの脱出の両方の性質を持っている. λが小さいと前者が, 逆にλが大きいと後者が優勢なシステムになる. そこで,λの値として問題に応じて中程度の値に制限する制約条件 (3.3) が重要になると考えられる.

3.6 結論と課題

大きなλの値に対しては, 全てのλ-置換行列の中から最良なλ-置換行列を求めるのは計算量的に困難である. 本論文では, この困難を解決する為に, 置換行列の空間で定義される, 自由エネルギーに基づくニューラルネットワークによる手法を提案した. この手法,λ-DCN,λ-interior DCN は, 我々が以前に提案した手法 DCN アニーリング (DCA) を改良した手法でもある. DCA よりも改良された点は, 次の２つの点である : (1) 非平衡なダイナミクスを導入することによって, 局所最適解からの脱出を可能とした. 従って, より多くの局所最適解を発見できるようになった. このことにより, 繰り返し回数を増やすほど, より良い解を発見できると期待できる. (2) 順列への置換操作を考えているので, 次々と行われる計算の途中での解は, すべて解の候補となっている. これは, 我々のアルゴリズムがうまく働く理由の１つと考えられる.

計算機実験において, 我々は比較的大きな ($N = 80 \sim 150$) 2 次割り当て問題を扱った. これらの問題に対しては, 我々のアルゴリズムは現在のチャンピオンのアルゴリズムと同程度の解を計算できることを示した. また, ２つのベンチマークに対しては我々のアルゴリズムは, 以前のチャンピオンのアルゴリズムよりも良い解を計算できる. 従って, 本論文で提案するアルゴリズムは QAP に対する, 最も強力なアルゴリズムの１つであると結論できる.

λ-interior DCN が扱う近傍は, 節 1.5で定義した k-exchange 近傍 $\mathrm{N}_k(\pi)$ と同じ

である．表 1は,k-exchange 近傍よりもλ-exchange 近傍,N*近傍を扱う手法が, より効率的であることを示している．したがって,λ-exchange 近傍,N*近傍をアナログで表現する手法を構成できるならば, λ-interior DCN よりも高い性能のアルゴリズムを実現できる可能性がある．そのようなアルゴリズムの構成は, 今後の課題である．

4. 実多項式行列スミス標準形の安定な浮動小数点計算法

4.1 はじめに

節2,3では, ニューラルネットによる組み合わせ最適化問題の解法について議論してきた. これらのニューラルネットによる手法は, 行列演算を多用するアルゴリズムである. 従って, 仮に行列とベクトルで表現された問題を, いくつかの部分空間上の問題に分割することができるならば, アルゴリズムの高速化を行うことができる. 本節では, 行列で表された問題を部分空間上の問題に分割する時に有用となる行列のジョルダン標準形, スミス標準形に注目する. 特に, スミス標準形を主題とする. ジョルダン標準形も重要であるが, それはその特性行列のスミス標準形から導くことができる.

スミス標準形は, 行, 列の基本的な変形操作とユークリッドの互除法の組合せによって計算できる. しかし, 一般に行列のサイズが大きいときなどに, 厳密計算でその計算を行うと, 膨大な時間と記憶容量を必要とする. そこで, 浮動小数点計算によって, その負荷を軽減できればよい. ところが, 上述の計算法 (アルゴリズム) は不安定である. つまり, 浮動小数点計算を用いるとき, 入力値の精度桁がどれほど大きくなっても, 答が正確なスミス標準形に収束するとは限らない. 文献 [32] は, このようなアルゴリズムを自動的に変形して, 計算の精度桁をある値以上に大きくすると, 正確な出力に近い答えを返すようなアルゴリズムを生成するテクニックについて述べている.

本節では, 正確な入力に収束する近似列を一つ一つ入力していったとき, それに応じて, その出力の列が正確なスミス標準形に収束することを保証するアルゴリズムを実現する. さらに, 実際にそれを計算機で実行して, その効果を報告する. 提案する方法は, 単純に従来のアルゴリズムに浮動小数点計算を導入したものではなく, 文献 [32] の安定化手法のテクニックを導入する. 文献 [36] はそのテクニックを日本語に要約したもの, 文献 [37] は一般読者向けに平易に解説したものである.

まず4.2節では, 安定化手法の概略を述べる. 4.3節では, 安定化手法を適用する対象となるアルゴリズムについて述べる. スミス標準形を効率的に求めるアルゴ

リズムは数多く存在するが, 本研究では, 効率性よりも安定性に重点を置き, 不安定性の原因を明確に説明するなどの理由で, 最も素朴と思われるアルゴリズムを選んだ. 4.3.2節では, そのアルゴリズムの不安定性の原因について詳しく述べる. 4.4.1節では, 元となるアルゴリズムに文献 [32] のテクニックを適用する. 4.4.2節で, 安定化の効果を見るための例を示し, 4.4.3節では, より大きな行列の大量な例について計算機実験を報告し, 安定化理論の有効性を示す.

4.2 アルゴリズムの安定化手法

文献 [32] に基づき, 次のクラスのアルゴリズムについて安定化手法を説明する.

- 入力, 中間ステップ, 出力の全てのデータは, $R[x_1, \ldots, x_m]$[10]に属する. Rは実数体の部分体である.
- アルゴリズムで行われる操作は,$R[x_1, \ldots, x_m]$ における加算, 減算, 乗算, 剰余計算のみである.
- 述語の不連続点は, あるとすれば 0 のみである.

ここで, 剰余計算とは, 多項式を多項式で割ったときの剰余を計算することである. 変数が 1 つしかない場合は, 普通の多項式の除算のアルゴリズムを使う. 剰余計算において, 割られる多項式の各係数を, 割る多項式の先頭の項の係数で割るときに, 体 Rでの除算が行われる. 2 変数以上の場合は, 主項を定義するための項順序が必要となるが, スミス標準形を計算する場合は 1 変数の多項式しか取り扱わないので, ここでは, 項順序についての詳細に立ち入らない.

次に述語の不連続点について説明する. 述語は多項式の集合から

$$\{\text{“TRUE”}, \text{“FALSE”}\}$$

への写像である. 述語 p は, $f_i \to f$となる $R[x_1, \ldots, x_m]$ の要素の列 $\{f_i\}_i$が存在して, $p(f_i) \not\to p(f)$ (すなわち $p(f_i) \neq p(f)$) となるとき,$f \in R[x_1, \ldots, x_m]$ で不連続と言われる. ここで,$f_i \to f$ とは f_iが係数ごとに fに収束することを示す. 係数

[10]係数が Rの要素で, 変数が $x_1, \ldots, x_m$である多項式の全体の集合.

の収束とは普通の実数集合における収束を示す. 0 が述語 p の不連続点であるとは, 従って, 0 多項式 (全ての係数が 0 である多項式) において述語 p が不連続であることである.

上述の 3 つの条件を満たすアルゴリズムを不連続点 0 の代数的アルゴリズムと呼ぶことにする. 多項式を扱う大抵の数式処理のアルゴリズムは, 不連続点 0 の代数的アルゴリズムであるか, またはそれにに変換できるものである. 例えば,$X = 3$? という述語は,$Y := X - 3$ という変数置き換えと $Y = 0$?という不連続点 0 の述語に変換できる.

簡単のために, アルゴリズムに現れる操作は, 多項式演算と剰余計算しかないとしているが, 文献 [32] では平方根や微分などの他の多くの関数についても議論している.

$\mathcal{A}$ を不連続点 0 の代数的アルゴリズムとする. $\mathcal{A}$ は近似入力に対して, 不安定となることがある. 例えば次のようなアルゴリズム $\mathcal{B}$を考える:

初期設定:	(X)
1.	$Y = 3X - 1$
2.	goto 4. if $Y \geq 0$
3.	stop (0)
4.	stop (1)

すなわち入力 Xに対して, アルゴリズム $\mathcal{B}$は,$3X - 1 \geq 0$ ならば 1 を返し, そうでなければ 0 を返す. 入力が $X = 1/3$ の場合を考える.$\mathcal{B}(1/3)$ は 1 の値を返す. しかし, 任意の精度μに対し, 浮動小数近似 $(1/3)_\mu$は $0.\overbrace{333\cdots3}^{\mu \textit{ digits}}$となり, $\mathcal{B}((1/3)_\mu)$ はいつも 0 を返してしまう. これが不安定性である. 換言すれば, $(1/3)_\mu \to (1/3)$ であるが, $\mathcal{B}((1/3)_\mu) \not\to \mathcal{B}(1/3)$ である. $\mathcal{B}$は, 不連続点として $\{0\}$ をもつ述語 $Y \geq 0$ を持っているので, 不連続点 0 の代数的アルゴリズムである. この不連続性がアルゴリズムの不安定性の原因である.

アルゴリズムの安定化は, アルゴリズムの構造を変えずにデータ集合を変えることで行われる. データ集合は, 元の係数から区間係数に変えられる. 区間係数

は,2 つの浮動小数点数のペア $[A, \alpha]$ で表され, A は中心,αは半径を意味する.$[A, \alpha]$ は, $|x - A| \leq \alpha$なる実数 x の全体を表現する. 区間の詳細については, 文献 [2] を参照されたい. 元のアルゴリズムにおいて係数の間で演算が行われる部分は, 区間係数の間で区間演算が行われる. キーポイントは, 述語を評価する直前に,「0 書換え」を行うことである. 具体的には, 区間係数が 0 を含むとき, その区間係数を 0 区間 (中心 0 半径 0 の区間) に書き換える. それ以外のときは, 何も変えず, そのままとする. 0 書換えを行った後, その区間係数の第一要素, つまり区間の中心において述語を評価する. 0 書換えの重要な特徴は, 区間係数の列の収束性を保存すること, そして, 区間係数の列が 0 に収束するならば, 0 書換えされた区間係数の列は有限回で 0 に「到達する」という点である. 従って, 述語が不連続点の 0 で評価できるようになる. もし 0 書換えを行わなかったとしたら, 述語は 0 の値で評価されない可能性があり, もとより 0 はその述語の不連続点であったため, アルゴリズムは元のアルゴリズムと同じ実行過程を辿らなくなる.

アルゴリズムの安定化手法についてまとめる. $\mathcal{A}$ を不連続点 0 の代数的アルゴリズム, $Int(\mathcal{A})$ を文献 [32] のテクニックによって安定化されたアルゴリズムとする. アルゴリズム $Int(\mathcal{A})$ は次の特徴をもつ.

区間領域 データ領域は区間を係数に持つ多項式の集合である. 区間係数は $[A, \alpha]$ の形をとる. ここで $A \in R$であり,αは非負の実数である. $[A, \alpha]$ は, 集合 $\{x \in R \mid |x - A| \leq \alpha\}$ を表現する.

区間演算 区間係数の加減乗除に対しては, 区間演算を用いる [2]. 区間係数間の二項演算子$* \in \{+, -, \times, \div\}$ に対し,

$$[A, \alpha] * [B, \beta] = [A * B, \gamma_*],$$

である. ここに, γ_*は,

$$|x - A| \leq \alpha, |y - B| \leq \beta \Rightarrow |x * y - A * B| \leq \gamma_*$$

を満足するように定義される.

0 書換え 不連続点 0 をもつ述語が評価される直前に, 0 書換えを行う. すなわち, 各区間係数 $[C,\gamma]$ に対して, $|C| \leq \gamma$ ならば $[C,\gamma]$ を [0,0] に書き換える. そうでなければ,$[C,\gamma]$ のままとする.

$Int(\mathcal{A})$ は区間係数多項式を入力として受け入れ, 区間係数多項式を出力として掃き出す. $\mathcal{A}$ に対する元の入力 $f \in R[x_1,\ldots,x_m]$ は, 区間係数多項式の列 $\{Int(f)_j\}_j$ に近似される. 具体的には, f を $\sum_{i_1,\ldots,i_m} a_{i_1\ldots i_m} x_1^{i_1}\cdots x_m^{i_m}$ とすると,$Int(f)_j$ は次のように定義される.

$$\sum_{i_1,\ldots,i_m} [(a_{i_1\ldots i_m})_j, (\alpha_{i_1\ldots i_m})_j]\, x_1^{i_1}\cdots x_m^{i_m},$$

ここで, それぞれの j に対して

$$|a_{i_1\ldots i_m} - (a_{i_1\ldots i_m})_j| \leq (\alpha_{i_1\ldots i_m})_j$$

0 であり,$j \to \infty$ のとき, 全ての $i_1\ldots i_m$ に対して $(\alpha_{i_1\ldots i_m})_j \to 0$ である. このとき, 省略して $Int(f)_j \to f$ と書く. 各 j に対し,$Int(\mathcal{A})$ を入力 $Int(f)_j$ で実行したとき, その出力を $Int(\mathcal{A})(Int(f)_j)$ と表記する.

次の定理は $Int(\mathcal{A})$ の特徴を述べている.

定理 1 (係数収束 [32]) 入力 $f \in R[x_1,\ldots,x_m]$ に対してアルゴリズム $\mathcal{A}$ は正常終了するとする. $Int(f)_j \to f$ となるような区間係数多項式の列 $\{Int(f)_j\}_j$ を考える. このとき, ある n が存在して, $j \geq n$ ならば, 入力 $Int(f)_j$ に対して $Int(\mathcal{A})$ は正常終了し, かつ,

$$Int(\mathcal{A})(Int(f)_j) \to \mathcal{A}(f).$$

しかし, この係数ごとの収束だけでは, 正確なスミス標準形を計算するという目的にはまだ不十分である. 最終的にもう一度 0 書換えを行う必要がある. すなわち, 出力 $Int(\mathcal{A})(Int(f)_j)$ の各区間係数に 0 書換えを行う. そして, 各区間係数の第一要素のみを取り出して, 出力を普通の実係数多項式に戻す. このように, 基本的には $Int(\mathcal{A})$ と同じであるが, 最後にその出力を 0 書換えし, 実係数多項式への変換を行うアルゴリズムを以下 $Int(\mathcal{A})_R$ と表す.

最後の 0 書換えの効果は,「台収束」と呼ばれる, 係数収束よりも強い収束を実現することである. 台収束について説明するために多項式の台を定義する. 多項式

$$f = \sum_{i_1,\ldots,i_m} a_{i_1\ldots i_m} x_1^{i_1} \cdots x_m^{i_m}$$

の台とは,0 でない係数をもつ変数の巾積 (係数 1 の単項式) の集合

$$\{x_1^{i_1} \cdots x_m^{i_m} \mid a_{i_1\ldots i_m} \neq 0\}$$

であり,$Supp(f)$ と書く. 同様に区間係数多項式 Fに対しても $Supp(F)$ は, 区間係数が $[0,0]$ でない巾積の集合である. 直観的に言えば, 台は「多項式の形」である. $Int(\mathcal{A})_R(Int(f)_j)$ の形は有限回のステップで (ある有限の jで), $\mathcal{A}(f)$ の形と同じになる. 換言すれば,$Int(\mathcal{A})_R(Int(f)_j)$ の係数で, 0 に収束するものは, 有限ステップで 0 に到達する. これを台収束と呼ぶ. これが, 正確なスミス標準形の計算のために必要となる.

次の定理が,$Int(\mathcal{A})_R$の安定性を述べるものである.

定理 2 (台収束 [32]) 定理 *1*と同じ仮定を置く. このとき, ある n が存在して, $j \geq n$ のとき, 入力 $Int(f)_j$に対して $Int(\mathcal{A})_R$は正常終了し, かつ

1. $Int(\mathcal{A})_R(Int(f)_j) \to \mathcal{A}(f)$, そして
2. ある Nがあって,$j \geq N$ならば,

$$Supp(Int(\mathcal{A})_R(Int(f)_j)) = Supp(\mathcal{A}(f)).$$

係数収束や台収束の実用例として, ブッフバーガーのアルゴリズム [30, 31] やスツルムのアルゴリズム [39] などがある. その他の例は, 文献 [37] を参照されたい.

4.3 スミス標準形

4.3.1 アルゴリズム

ここでは, 成分が $R[x]$ の要素である正方行列に対して, スミス標準形と単因子に関する理論を復習する. この理論は単項イデアル整域上の矩形行列に対しても

成立する [20, 19]. 成分が Rの要素である $n \times n$ の正方行列の集合を $M_n(R)$ として表す. $M_n(R[x])$ は,R係数の多項式を成分にもつ行列の集合である. $A \in M_n(R)$ とする. この研究の目的は,A の特性行列 $A - xE \in M_n(R[x])$ のスミス標準形を計算することである. ここで Eは $M_n(R)$ の単位行列を表す.

スミス標準形は $M_n(R[x])$ での行と列の基本変形によって計算できる. 基本変形とは次の 3 種類の操作を言う.

- 行または列の入れ換え.
- 行または列ベクトルのスカラー倍.
- 行 (列) ベクトルを多項式倍したものを別な行 (列) ベクトルに加えること.

$A(x), B(x) \in M_n(R[x])$ とするとき,$A(x)$ が有限回の基本変形で $B(x)$ になるとき $A(x)$ と $B(x)$ は対等であるという.

定理 3 (スミス標準形) 任意の $A(x)$ は, 次の対角行列と対等である.

$$\begin{bmatrix} e_1(x) & & & & & & \\ & e_2(x) & & & & & \\ & & \ddots & & & & \\ & & & e_r(x) & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix} \tag{4.1}$$

ここに,$e_i(x)$ $(i = 1, \ldots, r)$ は以下を満足する.

1. $e_{i-1}(x)$ は $e_i(x)$ を割り切る $(i = 2, \ldots, r)$.
2. $e_i(x)$ $(i = 1, \ldots, r)$ の主係数は 1 である.

定理に現れる多項式 $e_1(x), \ldots, e_r(x)$ を $A(x)$ の単因子と呼ぶ. 単因子は,$A(x)$ から一意に決まる.(4.1) の行列を $A(x)$ のスミス標準形と呼ぶ.

定理 3はよく知られた定理であるが, ここではその詳しい証明を載せる. その理由は, その証明が構成的で, そのままでスミス標準形を計算するアルゴリズムを表

現していること, そのアルゴリズムの不安定性の原因を説明するのに簡便であること, 実際にそのアルゴリズムを実装してそれを安定化し, 計算機実験を行ったことの 3 つである. なお, 証明には文献 [38] を参考にした.

証明:

証明は, いくつかのステップからなる. ここで $A(x) = [a_{ij}]_{1 \leq i,j \leq n}$ とする.

I. 「$A(x) = 0$ ならば $A(x)$ はスミス標準形である.」

以下では $A(x) \neq 0$ の場合を考える.

II. 「任意の (i,j) に対して,$A(x)$ と対等な $B(x)$ が存在して,$b_{11} = a_{ij}$ である. ここに,b_{11}は $B(x)$ の $(1,1)$ 成分である.」

これは明らかである.$i \neq 1$, $j \neq 1$ のときは $A(x)$ の i 行と 1 行を交換し, 次に j列と 1 列を交換すればよい.

III. 「a_{11}が $A(x)$ の第 1 行 (第 1 列) の成分を全て割り切るならば, $A(x)$ と対等な $B(x)$ が存在して,$b_{11} = a_{11}$であり, $(1,1)$ 成分以外の第 1 行 (第 1 列) の成分が 0 である.」

もし,$a_{1j} = q_j a_{11} (j \neq 1)$ ならば,j列から 1 列の q_j倍を引けば, 第 1 列の $(1,1)$ 成分以外を 0 にできる.

IV. 「第 1 行 (第 1 列) に a_{11}で割り切れない成分があるとき,$A(x)$ と対等な $B(x)$ が存在して,$deg(b_{ij}) < deg(a_{11})$ なる b_{ij}をもつ.」

例えば,a_{i1}が a_{11}で割り切れないとき,

$$a_{i1} = qa_{11} + r,\ 0 < \deg r < \deg a_{11}$$

となる 2 つの多項式 q, rが存在する. 第 1 行を q倍してそれを i 行から引けば, $deg(b_{i1}) < deg(a_{11})$ を満たす $B(x)$ が得られる.

V. 「a_{ij}を $A(x)$ の 0 でない最小次数の成分とする. $d(A(x)) = \deg a_{ij}$とおく. a_{ij}で割り切れない成分 a_{kl}が存在するならば, $A(x)$ と対等な $B(x)$ が存在して,$d(B(x)) < d(A(x))$ となる.」

行, 列を入れ換えても $d(A(x))$ は変化しないので, (II) より $d(A(x)) = \deg a_{11}$ と仮定してよい. 第 1 行 (第 1 列) に a_{11} で割り切れない成分があるならば, (IV) から (V) がいえる. 従って, 第 1 行と第 1 列の全ての成分が a_{11} で割り切れると仮定する. $a_{i1} = q_i a_{11}$ とせよ. 各 $i = 2, \ldots, n$ に対し, i 行から第 1 行の q_i 倍を引いて得られる行列を $C(x)$ とする. すると $c_{11} = a_{11}$, $c_{i1} = 0$ $(i = 2, \ldots, n)$ となる. $C(x)$ の (k, l) 成分は

$$c_{kl} = a_{kl} - q_k a_{1l}$$

となる. 仮定より, a_{kl} は a_{11} で割り切れず, かつ a_{1l} は a_{11} で割り切れるので, c_{kl} は a_{11} で割り切れない. $C(x)$ の第 1 行に第 k 行を加えてできる行列を $D(x)$ とする. $D(x)$ は $A(x)$ と対等であり, $d_{1l} = c_{1l} + c_{kl} = a_{1l} + c_{kl}$ は $d_{11} = c_{11} = a_{11}$ で割り切れない. よって, (IV) により, $D(x)$ と対等で, $\deg b_{ij} < \deg d_{11} = \deg a_{11} = d(A(x))$ となるような成分 b_{ij} を持つ $B(x)$ が存在することがいえた. $B(x)$ は $A(x)$ と対等であり, $d(B(x)) \leq \deg b_{ij} < d(A(x))$ であるから, (V) が証明された.

VI. 「$A(x)$ と対等な行列 $B(x)$ が存在して, $B(x)$ の全ての成分 b_{kl} を割り切る b_{ij} をもつ.」

a_{ij} を $d(A(x)) = \deg a_{ij}$ なる $A(x)$ の成分とせよ. a_{ij} が $A(x)$ の全ての成分を割り切るならば, $B(x) = A(x)$ とすればよい. それ以外の場合は, (V) より, $d(B(x)) < d(A(x))$ となる $A(x)$ と対等な $B(x)$ が存在する. $B(x)$ がその全ての成分を割り切る成分をもっていれば, 証明は終り. そうでない場合は, 再び (V) により $d(C(x)) < d(B(x))$ となる $B(x)$ と対等な $C(x)$ が存在する. この議論を繰り返す. $d(A(x))$ が自然数であるので, この議論は有限ステップで終了する. これで (VI) が証明された.

VII. 「$A(x)$ と対等な $B(x)$ が存在して, 次を満足する.

(1) $b_{11} \neq 0$

(2) $b_{i1} = b_{1j} = 0$ $(i > 1, j > 1)$

(3) b_{11} は b_{ij} $(i > 1, j > 1)$ を割り切る.

(4) b_{11}の主係数は 1 である.

」

(VI) より,$A(x)$ と対等な $C(x)$ が存在して, その全ての成分を割り切る成分 c_{ij}を持つ. $C(x)$ の行や列の入れ換えを行い, $d_{11} = c_{ij}$となるように行列 $D(x)$ を作る. $D(x)$ は $A(x)$ と対等であり,d_{11}が $D(x)$ の全ての成分を割り切る. (III) より,$D(x)$ と対等であり, (VII1) と (VII2) を満たす $B(x)$ が存在する. このとき,$B(x)$ は (VII3) も満足する. 何故ならば, 例えば (III) において d_{1j} $(j > 1)$ を消去するとき, 結果として得られる成分は,

$$d_{ij} - q_j d_{i1} \ (i > 1, j > 1)$$

となる. ここに,$d_{1j} = q_j d_{11}$で, 各成分は d_{11}で割り切れる (d_{i1} $(i > 1)$ を消去する場合も同様). 従って,b_{ij}は $b_{11} = d_{11}$で割り切れる. さらに,$B(x)$ の第 1 行を b_{11}の主係数で割れば, (VII4) も満たされる.

VIII. (VII1)–(VII4) を満たす $B(x)$ を半標準形と呼ぶことにする. $B(x)$ は以下のように書ける.

$$B(x) = \begin{bmatrix} b_{11} & 0 \\ 0 & B_1(x) \end{bmatrix}$$

$B_1(x)$ を (VII) における $A(x)$ に置き換えて考えると, $B_1(x)$ は以下の半標準形と対等である.

$$C_1(x) = \begin{bmatrix} b_{22} & 0 \\ 0 & B_2(x) \end{bmatrix}$$

b_{11}は $B_1(x)$ の全ての成分を割り切るので, b_{11}が $C_1(x)$ の全ての成分をも割り切る. $B_1(x)$ から $C_1(x)$ への基本変形を $B(x)$ にも行うと以下の行列を得る.

$$C(x) = \begin{bmatrix} b_{11} & & \\ & b_{22} & \\ & & B_2(x) \end{bmatrix}$$

この手続きを繰り返すことにより, 定理のスミス標準形が有限ステップで得られる.

noindent 証明終り

定理 3の証明でのステップ (I) – (VIII) から成るアルゴリズムを以下 *smith* で表す.

4.3.2 不安定性

アルゴリズム *smith* の不安定性の原因となるいくつかの側面について議論する. まず,*smith* において, 多項式 fが多項式 gを割り切るかどうかの判定が頻繁に起きる. ステップ (III), (IV), (VI) がそうである. この判定は gを fで割った余りが, 0 多項式 (全ての係数が 0 である多項式) であるかどうかに帰着される. 多項式が 0 多項式であるかどうかという述語は不連続点 0 をもち, これがアルゴリズム *smith* に不安定性を生じさせる可能性がある. 次に, 多項式から主項または主係数を選ぶという操作に, 暗に不連続性が含まれている. 多項式の主項とは, 係数が 0 でない単項式で次数が最大のものをいう. この操作は, 多項式同士の除算を行い, 剰余を計算するときに常に必要となる. 実際,(III), (IV), (VI), (VII) において現れる. この操作のもつ不連続性について説明しする. 主項を選ぶためには係数が 0 であるかどうかの判定が必要となる. 前述のように, この述語は不連続点 0 をもつ. 主係数についても同様である. 詳細は文献 [32] にある.

結論として,*smith* は不連続点 0 の代数的アルゴリズムであることがわかる.

次にアルゴリズム *smith* の不安定性の現象が具体的に起きる例を示す. 以下,$A \in M_n(R)$ に対し,A の特性行列 $A - xE \in M_n(R[x])$ を $A(x)$ と書く. よく知られているように,A が体 Rに多重固有値αをもつとき, $smith(A(x))$ は不安定となりやすい. より詳しく言えば, $A(x)$ の最後尾の単因子 $e_n(x)$ が $(x-\alpha)^k$ $(k \geq 2)$ という多重因子をもつときである. この現象は,A のジョルダン標準形の不安定性とも深く関連している. 実際, 任意の行列 $A \in M_n(R)$ に対し, ある「摂動方向行列」$F \in M_n(R)$ (Fの成分は 0 か± 1) が存在して, $0 < |\epsilon| \leq 1$ なる任意のϵに対し, $A + \epsilon F$が n 個の相異なる固有値を持つ. (実は, ほとんどの摂動方向行列がこの性質をもつ.) 詳細は, 文献 [18] を参照されたい. 以降に示す例では, いずれの場合も行列 A が全て体 Rに多重固有値をもつという上記の性質を満たしている.

例 1 次のような $M_3(\mathbf{Q})$ の行列 A を考える．ただし,$\mathbf{Q}$ は有理数全体の集合である．

$$A = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 1 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

このとき,$smith(A(x))(A - xE$のスミス標準形) は以下のようになる:

$$smith(A(x)) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & x-1 & 0 \\ 0 & 0 & x^2-2x+1 \end{bmatrix}$$

ここで, 摂動方向行列として次の Fを考える:

$$F = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & -1 \end{bmatrix}$$

摂動の大きさϵを 0.000001 とする．そして摂動された行列 $A + \epsilon F$を$\tilde{A}_\epsilon$で表す．すると,$\tilde{A}_\epsilon$は次のスミス標準形をもつ:

$$smith(\tilde{A}(x)) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1.0000 & 0 \\ 0 & 0 & p(x) \end{bmatrix}$$

ここに, $p(x) = 1.0x^3 - 3.0x^2 + 3.0x - 0.999997$ である．

どんなに小さなϵに対しても, スミス標準形の「形」は正確なものと一致しない．言い換えると,$\epsilon \to 0$ であっても $smith(\tilde{A}_\epsilon(x)) \to smith(A(x))$ とはならない．

4.4 スミス標準形計算の安定化

4.4.1 $Int(smith)_R$

4.2節で説明したアルゴリズムの安定化手法を古典的なアルゴリズム $smith$ に適用する[11].

$Int(smith)$ について説明しする. まず $Int(smith)$ のデータ領域は, 区間係数多項式を成分とする行列の集合である. ここで区間係数多項式とは,1 変数で係数が区間である多項式をいう. 区間演算が行われるのは, 区間係数多項式の行列に基本変形を行うとき, および区間係数多項式同士で除算を行い剰余を求めるときである. 0 書換えが行われるのは,4.3.2節で説明したように, 剰余多項式が 0 かどうかを判定する述語を評価する直前, および, 多項式の係数が 0 かどうかを判定する述語を評価する直前である. しかし,4.3.2節で述べたように, アルゴリズムの記述に述語が陽に現れていない場合があるため, 実装上は, 行列の基本変形や多項式の除算を行った直後で 0 書換えを行うことにする[12]. さらに $Int(smith)_R$は,$Int(smith)$ の出力行列の各成分の各係数に 0 書換えを施し, その結果の各区間係数の第一要素を取り出すことによって, 実係数多項式行列に戻すアルゴリズムである (4.2節の $Int(\mathcal{A})_R$の定義を見よ).

定理 2より以下が成り立つ.

定理 4 ($smith$ の安定化) 行列 $A(x) = [a_{kl}] \in M_n(R[x])$ に対して, 区間係数多項式の行列の列 $Int(A(x))_j = [Int(a_{kl})_j]$ で, 成分ごとに $A(x)$ に収束するものを考える. すなわち各 (k,l) について,$Int(a_{kl})_j \to a_{kl}$となる行列の列を考える. このとき,

1. $Int(smith)_R(Int(A(x))_j) \to smith(A(x))$ が行列成分ごとに成り立つ.

[11]スミス標準形を計算するアルゴリズムとして, より効率的で実用的なものが多く存在する [16, 15, 35]. 効率性を追求するならば, これらについても安定化を行い, その有用性を示すことは重要であろう. 文献 [29] は, 古典的なアルゴリズムの安定性について議論し, また有限精度の p 進近似計算でスミス標準形を計算する方法を提案している. これは, われわれの研究に密接に関係する.

[12]実装上の議論は文献 [36] に詳しい

2. ある Nが存在して,$j \geq N$ならば

$$Supp(Int(smith)_R(Int(A(x))_j) = Supp(smith(A(x))).$$

が成り立つ.

4.4.2 例

$Int(smith)_R$の効果を見るために, 具体的な実行例を挙げる. 実験は,$smith$ と $Int(smith)_R$を Maple V Release 3[7] で実装し, 計算機は HP9000/735 を使って行った. Maple は linear algebra というパッケージに "smith" という組み込み関数をもっているが, これは浮動小数点や代数的数の係数を受けつけない.

例 2 例 *1*を再掲する.

$$A = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 1 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & -1 \end{bmatrix}$$

これより

$$A(x) = \begin{bmatrix} 2-x & 0 & 1 \\ -1 & 1-x & -1 \\ -1 & 0 & -x \end{bmatrix}$$

となる.$A(x)$ に収束する区間係数多項式の行列の列として, 例えば "$[A+10^{-j}F, 10^{-j}]-x[E,0]$" を考える. すなわち,$Int(A(x))_j$として,

$$\begin{bmatrix} [2-e_j, e_j]-[1,0]x & [0,e_j] & [1,e_j] \\ [-1+e_j, e_j] & [1,e_j]-[1,0]x & [-1+e_j, e_j] \\ [-1,e_j] & [-e_j,e_j] & [-e_j,e_j]-[1,0]x \end{bmatrix}$$

を考える. ここに

$$e_j = 10^{-j}$$

である. 明らかに $Int(A(x))_j \to A(x)$ である. $Int(smith)$ に $\{Int(A(x))_j\}_j$を入力すると, 次のようになる. $j=2$ に対しては, $Int(smith)_R(Int(A(x))_2) =$

$$\begin{bmatrix} 1.0 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0 & 1.0x^3 - 3.0x^2 + 3.0x - 1.0 \end{bmatrix}$$

である. この結果は $smith(A(x))$ と台が一致しない. しかし $j=3$ に対しては, $Int(smith)_R(Int(A(x))_3)) =$

$$\begin{bmatrix} 1.00 & 0 & 0 \\ 0 & 1.00x - 1.00 & 0 \\ 0 & 0 & p(x) \end{bmatrix}$$

となる. ここに,$p(x) = 1.00x^2 - 2.00x + 0.997$ である. この結果は,$smith(A(x))$ と台が一致している. また $j = 4, \ldots, 10$ で各係数の収束性も実験で観測した.

次に摂動方向行列 Fをあらかじめ指定せず, A を単に浮動小数点近似する場合について考える.

例 3 $A =$

$$\begin{bmatrix} \frac{107}{153} & \frac{202}{153} & \frac{8}{153} & -\frac{61}{153} \\ -\frac{29}{153} & \frac{421}{306} & \frac{25}{153} & -\frac{37}{306} \\ -\frac{71}{306} & \frac{577}{612} & \frac{325}{306} & -\frac{175}{612} \\ -\frac{22}{51} & \frac{19}{51} & \frac{26}{51} & \frac{44}{51} \end{bmatrix}$$

のとき,

$smith(A(x)) =$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & (x-1)^2 & 0 \\ 0 & 0 & 0 & (x-1)^2 \end{bmatrix}$$

となる. A の5桁の浮動小数点近似$\tilde{A}$を考える.

$\tilde{A} =$

$$\begin{bmatrix} 0.69935 & 1.3203 & 0.052288 & -0.39869 \\ -0.18954 & 1.3758 & 0.16340 & -0.12092 \\ -0.23203 & 0.94281 & 1.0621 & -0.28595 \\ -0.43137 & 0.37255 & 0.50980 & 0.86275 \end{bmatrix}$$

単に $smith$ に$\tilde{A}(x)$ を入力すると, 正確な形の行列を返さない.

$smith(\tilde{A}(x)) =$

$$\begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & p(x) \end{bmatrix}$$

ここに, $p(x) = 1.0000x^4 - 3.9598x^3 + 5.9237x^2 - 3.9677x + 1.0040$ である.

具体的には以下の様にして, 正確でない計算結果が導かれる. 定理3で説明したアルゴリズムのVIIIにより以下の半標準形が導かれる.

$$\tilde{B}(x) = \begin{bmatrix} 1.3203 & 0 & 0 & 0 \\ 0 & -0.0012504 & 0 & 0 \\ 0 & 0 & p_1(x) & p_2(x) \\ 0 & 0 & p_3(x) & p_4(x) \end{bmatrix}$$

ここで, $p_1(x) = 241.49 - 483.00\,x + 241.50\,x^2$, $p_2(x) = -173.21 + 346.42\,x - 173.21\,x^2$, $p_3(x) = 799.78 - 1599.5\,x + 799.76\,x^2$, $p_4(x) = -571.12 + 1142.2\,x - 571.10\,x^2$ である.

この行列が計算されるまでは, 厳密に定理 3のアルゴリズムを実行した時と同じ形の行列が計算課程の途中においても計算される. 次に定理 3の VIIIにより, $\tilde{B}$ の部分行列

$$\tilde{B}_1 = \begin{bmatrix} p_1(x) & p_2(x) \\ p_3(x) & p_4(x) \end{bmatrix}$$

に対して,I,II,III... の操作が順に実行される. I,II は$\tilde{B}_1$を変化させない. IIIにおいては $p_1(x) = \tilde{B}_{3,3}(x)$ が $p_2(x) = \tilde{B}_{3,4}(x), p_3(x) = \tilde{B}_{4,3}(x)$ を割りきるか, の判定が行われる. つまり除算の剰余の 0 判定が行われる. 浮動小数近似を行っていないアルゴリズムにおいては, 割りきる, という判定行う. そして定理 3の IIIにより, 正確なスミス標準形を導く. しかし, 浮動小数近似においては, 割りきれない行列要素がある, という判定になり, 定理 3の IVの操作に入ってしまう.

桁数を高めて $j = 1000$ まで計算したが, 正しい形の行列は得られなかった. この原因は, つまり, 単純な浮動小数点近似では, 正しい 0 判定が出来ないことにある. $Int(smith)_R$はどうか? $A(x)$ に収束する区間係数多項式の列 $\{Int(A(x))_j\}_j$ として "$[float_j(A(x)), error_j(A(x))]$" を考える. すなわち行列成分は, それに対応する $A(x)$ の行列成分の j桁での区間 (浮動少数点近似とその誤差の上界) である. すると, $Int(smith)_R(Int(A(x))_2) =$

$$\begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0x^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

となる. まだ, 精度桁が足りない.

$Int(smith)_R(Int(A(x))_3) =$

$$\begin{bmatrix} 1.00 & 0 & 0 & 0 \\ 0 & 1.00 & 0 & 0 \\ 0 & 0 & p_1(x) & 0 \\ 0 & 0 & 0 & p_2(x) \end{bmatrix}$$

$$\begin{aligned} p_1(x) &= 1.00x^2 - 2.00x + 1.01 \\ p_2(x) &= 1.00x^2 - 2.00x + 0.989 \end{aligned}$$

$j = 3$ とすると正確な形の行列を得る. $j = 4, \ldots, 10$ に対して, $\{Int(smith)_R(Int(A(x))_j)\}_j$ が $smith(A(x))$ に成分ごとに台収束することを実験により観測した.

最後に無理数を含む場合を考える.

例 4 A として $M_4(\mathbf{Q}(\sqrt{2}))$ の要素を考える. ただし, $\mathbf{Q}(\sqrt{2})$ は集合 $\{a + b\sqrt{2} \mid a, b \in \mathbf{Q}\}$ である.

$$A = (3608\sqrt{2} - 8545)^{-1} \times$$

$$\left(\begin{bmatrix} 6322 & -186 & -408 & 959 \\ -1788 & 6844 & -816 & 1918 \\ -9834 & -2046 & 2728 & 10549 \\ -5364 & -1116 & -2448 & 12970 \end{bmatrix} \right.$$

$$\left. + \sqrt{2} \begin{bmatrix} -8545 & 252 & -24 & 40 \\ 0 & -8041 & -48 & -80 \\ 0 & 2772 & -8809 & -440 \\ 0 & 1512 & -144 & -8785 \end{bmatrix} \right)$$

このとき,

$smith(A(x)) =$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & x-\sqrt{2} & 0 & 0 \\ 0 & 0 & x-\sqrt{2} & 0 \\ 0 & 0 & 0 & \left(x-\sqrt{2}\right)^2 \end{bmatrix}$$

である.

同様に, 単に $A(x)$ の浮動小数点5桁による近似 $\tilde{A}(x)$ を $smith$ に入力すると,

$smith(\tilde{A}(x)) =$

$$\begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & p(x) \end{bmatrix}$$

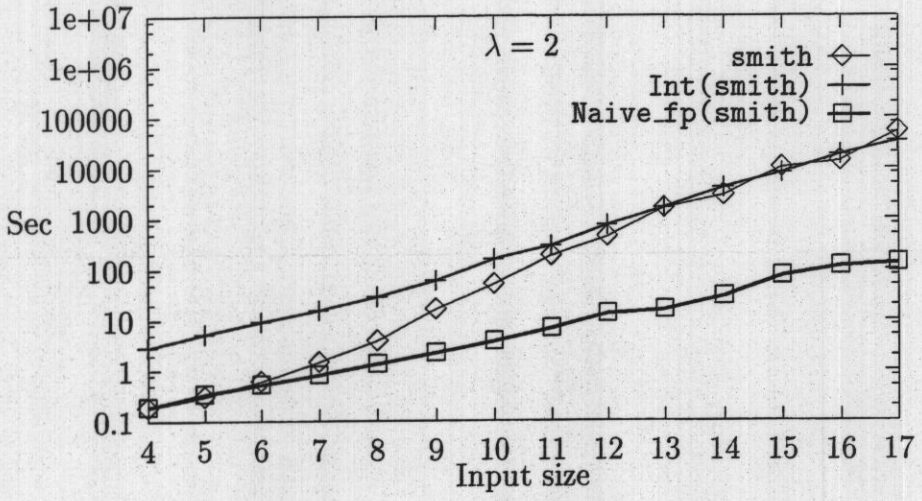
ここに, $p(x) = 1.0000x^3 - 7.8284x^2 + 16.141x - 9.9985$ である. この場合の不安定性の原因も, 例3と同様の形の行列が導かれた後の定理3のIIIでの割りきるかどうかの判定, が正しく行われていないことによる. この原因は単純な浮動小数点近似では, 正しい0判定が出来ないことにある.

これは, 不安定である.$j = 1000$ でも安定な結果が得られなかった. ところが,$Int(smith)_R$によれば,$j = 5$(かつ5より大きいいくつかの j) に対して, 以下のような正確な結果が得られた.

$Int(smith)_R([Int(A(x))_5]) =$

$$\begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & p_1(x) & 0 & 0 \\ 0 & 0 & p_2(x) & 0 \\ 0 & 0 & 0 & p_3(x) \end{bmatrix}$$

ここに, $p_1(x) = 1.0000x - 1.4144$, $p_2(x) = 1.0000x - 1.4143$, $p_3(x) = 1.0000x^2 - 2.8282x + 1.999$ である.


1e+07
1e+06
100000
10000
Sec 1000
100
10
1
0.1
$\lambda = 2$
smith
Int(smith)
Naive_fp(smith)
4 5 6 7 8 9 10 11 12 13 14 15 16 17
Input size

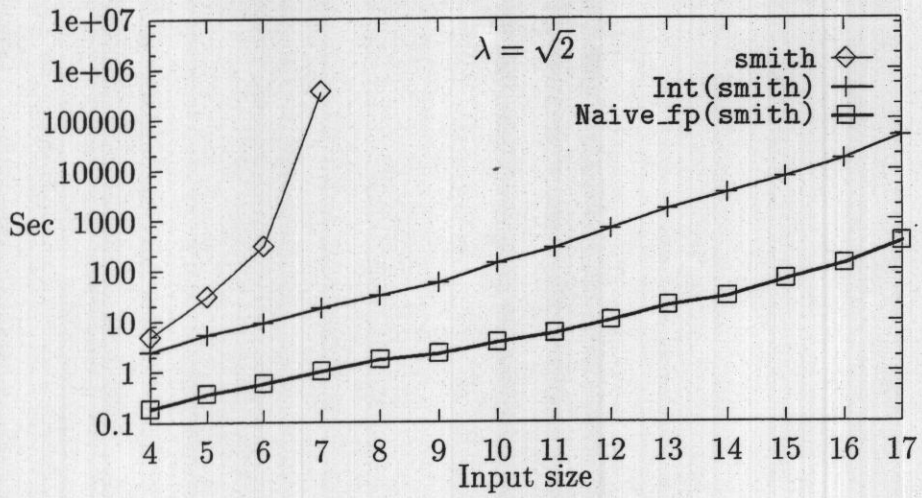

1e+07
1e+06
100000
10000
Sec 1000
100
10
1
0.1
$\lambda = \sqrt{2}$
smith
Int(smith)
Naive_fp(smith)
4 5 6 7 8 9 10 11 12 13 14 15 16 17
Input size

図 13 計算時間

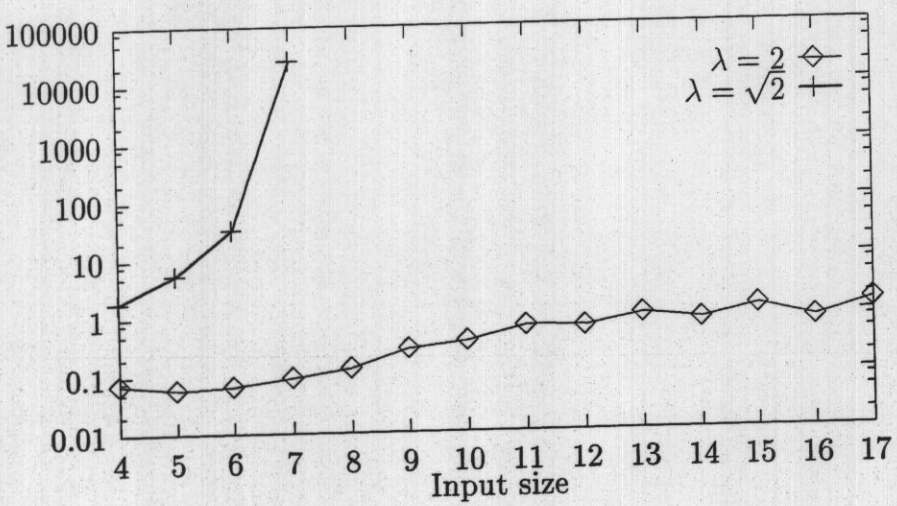


図 14 (smith の計算時間)/(Int(smith) の計算時間) の比

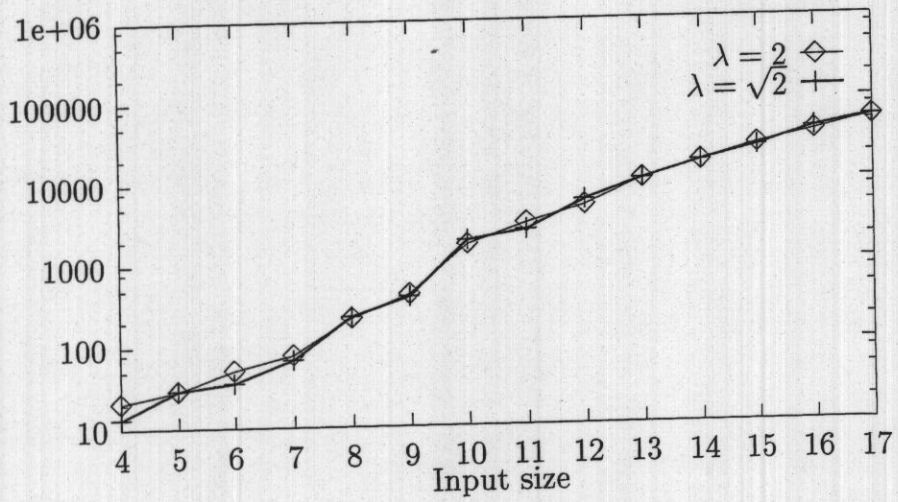


図 15 0 に書き換えられた区間係数の個数

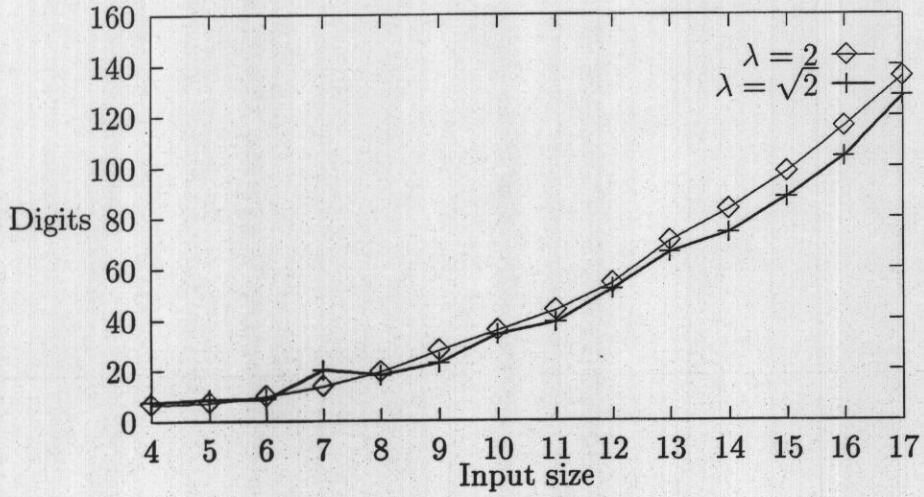


図 16 正確な台を与える最小精度桁

4.4.3 実験結果

前節では, 二三の例で $Int(smith)_R$の安定性を観察してきた. この節では, より大きな行列の大量な例について, 次のデータをグラフによって示す.

- 計算時間
- 素朴な浮動小数点計算と厳密計算との比較.
- 提案手法と厳密計算との計算時間の比
- $Int(smith)_R$の一回の実行中,0 書換えによって 0 に置きかわった区間の個数.
- 正確な台を与える最小の桁数 (定理 4の N).

入力のサンプルは, 多重固有値をもつようにランダムに生成した. すなわち,D を同じ実数が 2 回以上現れる対角行列, Pをランダムな正則整数行列として $A = P^{-1}DP$を入力とした. Pの生成には,Maple のランダム関数を使った. 簡単のため, 次の 2 つの場合について実験を行った. (1) A の全ての固有値が 2 の場合, (2)

Aの全ての固有値が$\sqrt{2}$の場合. (2) の場合には,Pも$\sqrt{2}$をランダムに含むようにした. それぞれのグラフは, 各サイズに対し, 10 個の異なるサンプルについての平均値をとっている.

図について説明する. 全ての図で, 横軸は,$n \times n$ の入力行列のサイズ n を表している. 縦軸は, 最後の図を除き, 底 10 の対数目盛で書いてある. なお, グラフに不足している点や線は, 与えられた計算機環境内で耐えられる時間内に出力できなかったため計算を打ち切り, 対応するデータが得られていないことを示す.

図 13では, 縦軸は計算 CPU 時間 (秒) を表す. (1) 固有値λが 2 の場合, (2) 固有値λが$\sqrt{2}$の場合の両方について記している. **smith**, **Int(smith)**, **Naive_fp(smith)** はそれぞれ,*smith* の厳密計算,$Int(smith)_R$の計算, *smith* の (0 の判定を行わない) 素朴な浮動小数点計算を示す. **Int(smith)** の精度は, 正確な台を与える最小の桁数を使った. **Naive_fp(smith)** に対してもそれと同じ桁数を使って計算した. 桁数については, 図 16を見よ.

図 14は, 図 13から導かれるもので, **smith** の **Int(smith)** に対する計算時間の比を表す.

図 15は, 図 13での **Int(smith)** の一回の実行において, C_jは 0 ではないが $|C_j| \leq \gamma_j$であるため $[C_j, \gamma_j]$ が $[0,0]$ に書き換えられたその $[C_j, \gamma_j]$ の個数を示す.

図 16は, 正確な台が最初に出現する桁数, すなわち, 定理 4における Nを示す.

考察: 図 13を見れば, 予期していた通り, 素朴な浮動小数点計算 **Naive_fp(smith)** が最も速いことが分かる. しかし,$j = 1,000$ 桁の精度で計算を行っても正確な台は得ることはできなかった. 0 書換えを行わない素朴な区間解析法でも **Naive_fp(smith)** と同様であった. すなわち 0 書換えなしでは 1,000 桁の精度で計算をしても正確な台を得ることはできない. 一方, 提案手法 **Int(smith)** では, 図 16で示すように比較的小さな桁数の計算で安定な結果が得られる. 従って 0 書換えが実際に効いていることがわかる. このことは, 図 15によっても確認される. 大まかにいえば, 計算時間は,**smith**, **Int(smith)**, **Naive_fp(smith)** はどれも, 対数目盛上で行列サイズと共に線形に増加しているように見える. すなわち通常の目盛上では指数関数的に増加している. 図 13と図 14からいえることは, 有理数係数の場合,**Int(smith)** は **smith** より必ずしも速くはない (少なくとも中くらいのサイズの行列に対して

は) のに対し, 係数が無理数を含むような複雑な場合には, Int(smith) は smith より, はるかに速いことである. さらに,Int(smith) は,Naive_fp(smith) より区間演算と 0 書換えの計算コストの分だけ計算が遅くなるが, はるかに安定である.

4.5 結論

スミス標準形を, 有限精度の浮動小数点計算を用いても安定に計算できるための手法を提案し, 実際にその効果を実証した. 今後の課題としては, 行列が与えられたとき, 本手法がスミス標準形の正確な台を与えるのにどれだけの桁数が必要であるかを理論的に見積もることが挙げられる. また, 安定化手法をより効率的で実用的なアルゴリズムに適用し, スミス標準形の高速で信頼性の高い計算方式を実現することも重要である.

5. まとめ

本研究では, ニューラルネットワークによる新たな組合せ最適化問題の解法を2つ提案した.

提案した1つめの手法は, カオスニューラルネットを用いた最適化問題の解法を, 座標変換によって改良する手法である. この手法は, 目的関数の滑らかさを示すラプラシアンの空間平均を座標変換によって小さくし目的関数が持つポテンシャルバリヤを取り除くことで, カオスによる非平衡ダイナミクスが, このポテンシャルバリヤを乗り越えやすくし, より広い空間を探索できるようにする手法である. この座標変換をQAPLIBの密で大きなベンチマークに適用した結果,20%程度ラプラシアンの空間平均を小さくすることができた. この座標変換による手法を, カオス的DCN(CDCN)に適用した結果, 計算機実験で, より多くの局所最適解を発見できるようになることを確認した. この座標変換手法は計算時間は2倍程度にしか増加させない手法であり, 問題の大きさNが100程度の大きな2次割当て問題にも適用可能である. また, この手法の基本的な考え方はCDCN以外の勾配法と類似点をもつ手法にも適用可能である.

提案した2つめの手法は,λ-optヒューリスティクスのアナログ版による基づく手法である. λ-opt近傍をアナログ的に表現し, そのλ-opt近傍の解を全て探索するのではなくアナログ的な判定基準で選び出された解のみを探索することで, 少ない計算時間でより広いλ-opt近傍を探索することが可能となった. この手法をQAPLIBのベンチマークに適用したところ2つのベンチマークについては, 従来のチャンピオンである手法よりも良い解を発見できた. また, 他の大きく対称で密なベンチマークについても, 従来のチャンピオンである手法と同程度の解を発見できた.

さらに, これらニューラルネットによる手法を高速化するための1つの手法として数式処理に注目し, 本研究で提案したアルゴリズムで良く使われる行列演算の高速化を行う為に必要な基礎的研究を行った. 多くの行列演算は部分空間ごとの演算に分割することで, 高速化することができる. その為には行列のジョルダン標準形やスミス標準形を求める必要がある. ジョルダン標準形はスミス標準形から求めることができるが, 安定化理論による手法で大きな行列のスミス標準形

を求める時に必要となるパラメーターを見積もる為の計算機実験を行った.

以上の研究により, 大規模な 2 次割当て問題のより良い近似解を, より高速で求めることが可能となった. しかし, 本研究で提案した手法でλ-DCN および,λ-interior DCN は, まだ改良の余地のあるアルゴリズムである. 今後の課題は, さらに良い解を発見できるようにλ-DCN および,λ-interior DCN を改良することにある. また, 本研究で提案した QAP の解法はいずれも非対称な QAP に対しては, あまり力を発揮できていない. QAPLIB における, 非対称なベンチマークに対しても実用的解を計算できる様にアルゴリズムを改良することも今後の課題である.

謝辞

論文の執筆にあたり，多大な力を貸して頂いた 石井 信助教授に心から感謝いたします．プログラムで協力して頂いた後藤 大介君に心から感謝いたします．伊藤研究室のみなさんに心から感謝いたします．

参考文献

[1] T. Aihara, K. Takabe and M. Toyoda. Chaotic neural networks. *Physics Letters A*, 144:333–340, 1990.

[2] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Computer Science and Applied Mathematics. Academic Press, 1983.

[3] S. Amin. Simulated jumping. *Annals of Operations Research*, to appear, 1998.

[4] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on computing*, 6:126–140, 1994.

[5] G. Bilbro, R. Mann, T.K. Miller, W.E. Snyder, D.E. Van den Bout, and M. White. *Optimization by mean field annealing.* 1988.

[6] S. Burkard, R.E. Karisch and F. Rendl. Qaplib - a qadratic assignment problem library. *European Journal of Operational Research*, 55:115–119, 1991.

[7] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *First Leaves: A Tutorial Introduction to Maple V.* Springer-Verlag, 1992.

[8] C. Fleurent and J.A. Ferland. Genetic hybrids for the quadratic assignment problem. *Quadratic assingnment and related problems*, 16:173–187, 1994.

[9] M. Hasegawa, T. Ikeguchi, and K. Aihara. Harnessing of chaotic dynamics for solving combinatorial optimization problems. *In Proceedings of the Fifth International Conference on Neural Information Processing*, 2:749–752, 1998.

[10] T. Hasegawa, M. Ikeguchi and K. Aihara. Combination of chaotic neurodynamics with the 2-opt algorithm to solve traveling salesman problems. *Physical Review Letters*, 79:2344–2347, 1997.

[11] J.J. Hopfield and D.W. Tank. "neural" computations of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[12] S. Ishii and M. Sato. Doubly constrained network for combinatorial optimization. *ATR Technical Report*, H-193, Kyoto:ATR, 1996.

[13] S. Ishii and M. Sato. Chaotic potts spin model for combinatorial optimization problem. *Neural Networks*, 10:941–963, 1997.

[14] S. Ishii and M. Sato. Constrained neural approaches to quadratic assignment problems. *Neural Networks*, 11:1073–1082, 1998.

[15] E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM Journal on Algebraic and Discrete Methods*, 8:683–690, 1987.

[16] R. Kannan. Polynomial-time algorithms for solving systems of linear equations over polynomials. *Theoretical Computer Science*, 39:69–88, 1985.

[17] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[18] T. Kato. *Perturbation Theory for Linear Operators.* Springer-Verlag, 2nd edition, 1976.

[19] S. Lang. *Introduction to Linear Algebra.* Addison-Wesley, 1965.

[20] S. Lang. *Algebra.* Addison-Wesley, 3rd edition, 1993.

[21] Y. Li. *Heuristic and exact algorithms for the quadratic assignment problem.* Ph.D.thesis. The Pennsylvania State University, 1992.

[22] Y. Li, M. Panos, and G.C.R. Maruricio. A greedy randomized adaptive search procedure for the quadratic assignment problem. *Quadratic assingnment and related problems*, 16:237–261, 1994.

[23] S. Lin and B.W. Kernigham. An efficient heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[24] D.G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, Reading Massachusetts, 1984.

[25] W.S. McCulloch and W. Pits. A logical clculs of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1988.

[26] H. Nozawa. A neural network model as a globally coupled map and applications based on chaos. *CHAOS*, 2:377–386, 1992.

[27] K. Ohta, M. Matsumiya and K. Fukunaga. An analysis on minimum searching principle of chaotic neural network. *IEICE Transactions on Fundamentals*, E79-A:363–369, 1996.

[28] C. Peterson and B. Söderberg. Parallel distributed approaches to combinatorial optimization problems onto neural networks. *International Journal of Neural Systems*, 1:3–22, 1986.

[29] V. Ramachandran. Exact reduction of a polynomial matrix to the Smith normal form. *IEEE Transactions on Automatic Control*, AC-24(4):638–641, 1979.

[30] K. Shirayanagi. An algorithm to compute floating point Gröbner bases. In T. Lee, editor, *Mathematical Computation with Maple V: Ideas and Applications*, pages 95–106. Birkhäuser, 1993.

[31] K. Shirayanagi. Floating point Gröbner bases. *Mathematics and Computers in Simulation*, 42(4-6):509–528, 1996.

[32] K. Shirayanagi and M. Sweedler. A theory of stabilizing algebraic algorithms. Technical Report 95-28, Mathematical Sciences Institute, Cornell University, 1995.

[33] E.D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–255, 1991.

[34] E.D. Taillard. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3(2):87–105, 1995.

[35] G. Villard. Computation of the Smith normal form of polynomial matrices. In *Proceedings of ISSAC'93*, pages 208–217, Kiev, Ukraine, 1993.

[36] 白柳 潔. アルゴリズムの安定化理論. 数式処理, 5(2):2–21, 1997.

[37] 白柳 潔. 不安定なアルゴリズムを安定化する. 情報処理, 39(2):111–115, 1998.

[38] 杉浦光夫. *Jordan* 標準形と単因子論 *II*. 岩波講座 基礎数学. 岩波書店, 1977.

[39] 関川 浩 白柳 潔. ゼロ書換えに基づいた区間法と sturm のアルゴリズムへの応用. 電子情報通信学会論文誌, J80-A(5):791–802, 1997.

付録

A. ボルツマンマシンとニューラルネットワーク

ここでは式 (1.17) の導出を行う. 式 (1.9) のシステムをσ_iではなくスピン S_iを使った表記に書き換えると

$$p(S_i \to S_i') = f_\beta(-\triangle E_{obj}) = \frac{1}{1+\exp(\beta \triangle E_{obj})} \tag{A.1}$$

$$\triangle E_{obj} = E_{obj}(S') - E_{obj}(S) \tag{A.2}$$

となる. $p(S_i \to S_i')$ は,S_iの値が変化する確率を示す. $\triangle E_{obj}$は S_iが変化した場合の目的関数 E_{obj}の変動量を示す. スピン S_iが変化した時の式 (1.16) の E_{obj}を考えるために, 式 (1.16) で S_iの関係する項のみをとりだすと

$$S_i[\sum_j W_{i,j}S_j + J_i]$$

となる. $S_i = 0$ の時に S_iが変化して $S_i = 1$ となる確率は

$$p(S_i = 0 \to 1) = f_\beta(-[\sum_j W_{i,j}S_j + J_i])$$

となる. ここで簡単のために $W_{i,i} = 0, W_{i,j} = W_{j,i}$とした. $S_i = 0$ の時に S_iが変化しない確率は

$$p(S_i = 0 \to 0) = 1 - f_\beta(-[\sum_j W_{i,j}S_j + J_i]) = f_\beta([\sum_j W_{i,j}S_j + J_i])$$

ここで

$$\begin{aligned} 1 - f_\beta(x) &= 1 - \frac{1}{1+\exp(\beta x)} \\ &= \frac{\exp(\beta x)}{1+\exp(\beta x)} \\ &= \frac{1}{1+\exp(\beta - x)} \\ &= f_\beta(-x) \end{aligned}$$

の関係を使った. $S_i = 1$ の時の S_iが変化する確率, 変化しない確率も同様に計算できる. まとめると

$$\begin{aligned} p(S_i = 0 \to 0) &= f_\beta([\sum_j W_{i,j} S_j + J_i]) \\ p(S_i = 0 \to 1) &= f_\beta(-[\sum_j W_{i,j} S_j + J_i]) \\ p(S_i = 1 \to 0) &= f_\beta([\sum_j W_{i,j} S_j + J_i])) \\ p(S_i = 1 \to 1) &= f_\beta(-[\sum_j W_{i,j} S_j + J_i]) \end{aligned} \tag{A.3}$$

となる. 式 (A.3) を見ると, 計算された確率は変化後の状態のみによって決まっていることがわかる. したがって, 式 (A.3) を次のように記述することもできる.

$$\begin{aligned} p(S_i = 1) &= f_\beta([\sum_j W_{i,j} S_j + J_i]) \\ p(S_i = 0) &= f_\beta(-[\sum_j W_{i,j} S_j + J_i]) \end{aligned} \tag{A.4}$$

これで, 式 (1.17) を導くことができた.

B. λ-exchange と N^*近傍

λ-exchange は, 順列πの近傍として次の $N_\lambda(\pi)$ を考える手法である.

$$N_\lambda(\pi) = \bigcup_{i=0}^{k} N_2(\pi_i, G_i) \tag{B.1}$$

G_iは置換操作を行いたい順列中の位置の集合

$$G_i = \{ \text{置換操作を行いたい位置のペア } (n, m)\}$$

である. $N_2(\pi_i, G_i)$ は,G_iに含まれるペアの集合について 2-exchange 操作を行った順列の集合である. このペアの集合 G_iは, 次の繰り返し計算の結果できる集合を考える.

$$G_{k+1} = G_k - \{(i,j) | i \text{ or } j \in \bigcup_{l=0}^{k} \delta(\pi_l, \pi_{l+1}), i, j = 1, ..., N\} \tag{B.2}$$

この繰り返し計算は $E_{obj}(\pi_k) > E_{obj}(\pi_0)$ か $G_k = \emptyset$ となるまで続ける. 図 17に λ-exchange アルゴリズムを示す.

式 (B.2) を

$$G_{k+1} = G_k - \{(i,j)|i \text{ and } j \in \bigcup_{l=0}^{k} \delta(\pi_l, \pi_{l+1}), i, j = 1, ..., N\} \qquad \text{(B.3)}$$

に入れ替えた同様の近傍が,N*近傍である. 図 18に N*近傍に基づくアルゴリズムを示す.

C. 自由エネルギーの最小原理

スピン$\boldsymbol{S}$の生じる確率が, 式 (1.11) で与えられるボルツマン分布に従うとする. 変数として, さらにパラメーターxを考えるとする. この時, パラメーターが$x = x_1$となる確率は

$$P(x_1) = \frac{1}{Z} \sum_{\boldsymbol{S} \in C(x_1)} \exp(-E_{obj}(\boldsymbol{S})/T) = Z(x_1)/Z$$

で与えられる. ここで $C(x_1)$ はパラメーター x が x_1という値になるようなスピンベクトル$\boldsymbol{S}$の集合である. 従って, 最も生じる確率の大きいパラメーター x の値は

$$Z(x) = \sum_{\boldsymbol{S} \in C(x)} \exp(-E_{obj}(\boldsymbol{S})/T)$$

を最大にするような x の値である. $Z(x)$ の最大値を求める問題は

$$\phi(x) = -T\log(Z(x))$$

を最小化する問題ともみれる. この$\phi(x)$ は, 熱統計力学では自由エネルギーと呼ばれる. 従って自由エネルギーを最小となるようなスピンベクトルとは最も生じる確率の高いスピンベクトルである. これは自由エネルギーの最小原理と呼ばれる.

D. DCN 方程式

ここでは式 (1.56),(1.57),(1.58) で示される DCN 方程式の導出を行う. 式 (1.55) のラグランジュ関数の停留条件は次で与えられる.

procedure λ-exchange(N, π)

I. Loop=true;

II. do Loop==true $\rightarrow$

III. $\pi_0 = \pi$; $G_0 = G$; Done=false;

IV. do Done==false and $G_k \neq \emptyset \rightarrow$

V. $\mathbf{N}_2(\pi_k, G_k)$ 内の最良解$\pi_{k+1} \in \mathbf{N}_2(\pi_k, G_k)$ を見つける;

VI. $G_{k+1} = G_k - \{(i,j) | i \text{ or } j \in \bigcup_{l=0}^{k} \delta(\pi_l, \pi_{l+1}), i, j = 1, ..., N\}$;

VII. if $E_{obj}(\pi_{k+1}) \geq E_{obj}(\pi_0) \rightarrow$ Done=true fi;

VIII. k=k+1;

IX. od;

X. $\pi = \text{argmin}\{E_{obj}(\pi_1), ..., E_{obj}(\pi_k)\}$;

XI. if $E_{obj}(\pi) == E_{obj}(\pi_0) \rightarrow$ Loop=false fi;

XII. od;

XIII. return(π)

図 17 λ-exchange

procedure $\mathbf{N}^*$neighborhood(N, π)

I. Loop=true;

II. do Loop==true $\rightarrow$

III. $\pi_0 = \pi$; $G_0 = G$; Done=false;

IV. do Done==false and $G_k \neq \emptyset \rightarrow$

V. $\mathbf{N}_2(\pi_k, G_k)$ 内の最良解$\pi_{k+1} \in \mathbf{N}_2(\pi_k, G_k)$ を見つける;

VI. $G_{k+1} = G_k - \{(i,j) | i \text{ and } j \in \bigcup_{l=0}^{k} \delta(\pi_l, \pi_{l+1}), i, j = 1, ..., N\}$;

VII. if $E_{obj}(\pi_{k+1}) \geq E_{obj}(\pi_0) \rightarrow$ Done=true fi;

VIII. k=k+1;

IX. od;

X. $\pi = \text{argmin}\{E_{obj}(\pi_1), ..., E_{obj}(\pi_k)\}$;

XI. if $E_{obj}(\pi) == E_{obj}(\pi_0) \rightarrow$ Loop=false fi;

XII. od;

XIII. return(π)

図 18 $\mathbf{N}^*$neighborhood

$$\frac{\partial L}{\partial V_{a,n}} = \frac{\partial E(\boldsymbol{V})}{\partial V_{a,n}} + T(\log V_{a,n} + 1) + \tilde{\lambda}_n + \tilde{\mu}_a = 0 \tag{D.1}$$

$$\frac{\partial L}{\partial \tilde{\lambda}_n} = \sum_a V_{a,n} - 1 = 0 \tag{D.2}$$

$$\frac{\partial L}{\partial \tilde{\mu}_a} = \sum_n V_{a,n} - 1 = 0 \tag{D.3}$$

さらに, 次の補助変数を導入する

$$U_{a,n} = \exp(\frac{\partial E(\boldsymbol{V})}{\partial V_{a,n}}/T) \tag{D.4}$$

補助変数 $U_{a,n}$ を使うと式 (D.1) は次の様に表せる.

$$V_{a,n} = \frac{U_{a,n}}{\mu_a \lambda_n} \tag{D.5}$$

ここで

$$\mu_a = \exp(\tilde{\mu}_a/T + 1) \tag{D.6}$$

$$\lambda_n = \exp(\tilde{\lambda}_n/T) \tag{D.7}$$

である. この式 (D.5) と式 (D.2) から次の式を得られる.

$$\mu_a = \sum_n \frac{U_{a,n}}{\lambda_n} \tag{D.8}$$

式 (D.8) を式 (D.5) に代入すると

$$V_{a,n} = \frac{U_{a,n}/\lambda_n}{\sum_m U_{a,m}/\lambda_m)} \tag{D.9}$$

を得る. 式 (D.9) と式 (D.3) から,λ_nについての次の式も得る.

$$\lambda_n = \sum_a \frac{U_{a,n}}{\sum_m (U_{a,m}/\lambda_m)} \tag{D.10}$$

以上で DCN 方程式を導くことができた.

E. 勾配法と DCN

ここでは, 式 (2.14) の導出を行う. 式 (1.60) は, 次のように書き換えられる.

$$V_{a,n}(t+1) = \frac{U_{a,n}(t+1)}{\mu_a(t+1)\lambda_n(t+1)} \tag{E.1}$$

ここで, 式 (D.8) と同様に

$$\mu_a(t+1) = \sum_n \frac{U_{a,n}(t+1)}{\lambda_n(t+1)} \tag{E.2}$$

とした. 両辺の対数をとると, 式 (D.1) と同様の次の式が導ける.

$$\begin{aligned} \log V_{a,n}(t+1) &= \log U_{a,n}(t+1) - \log \mu_a(t+1) - \log \lambda_n(t+1) \\ &= \frac{\partial E(\boldsymbol{V}(\mathrm{t}))}{\partial V_{a,n}}/T + \tilde{\lambda}_n(t+1)/T + \tilde{\mu}_a(t+1)/T - 1 \end{aligned} \tag{E.3}$$

ここで, 式 (D.6),(D.7) と同様に

$$\mu_a(t+1) = \exp(\tilde{\mu}_a(t+1)/T + 1) \tag{E.4}$$

$$\lambda_n(t+1) = \exp(\tilde{\lambda}_n(t+1)/T) \tag{E.5}$$

とした. また式 (1.59) を使った. ところで, 式 (1.36) で与えた自由エネルギーϕについて, 次の計算ができる.

$$\frac{\partial \phi(\boldsymbol{V}(\mathrm{t}))}{\partial V_{a,n}} = \frac{\partial E(\boldsymbol{V}(\mathrm{t}))}{\partial V_{a,n}} + T(\log V_{a,n}(t) + 1) \tag{E.6}$$

この式を使って, 式 (E.3) から$\frac{\partial E(\boldsymbol{V}(\mathrm{t}))}{\partial V_{a,n}}$を消去すると

$$\log V_{a,n}(t+1) = (\frac{\partial \phi(\boldsymbol{V}(\mathrm{t}))}{\partial V_{a,n}} + \tilde{\lambda}_n(t+1) + \tilde{\mu}_a(t+1))/T + \log V_{a,n}(t) \tag{E.7}$$

となり, 式 (2.14) が導ける.

F. λ-DCN 方程式

ここでは式 (3.15),(3.16),(3.17),(3.18),(3.19) で示されるλ-DCN 方程式の導出を行う.

式 (3.12),(3.13),(3.14) の制約のもとで$\phi^*(\boldsymbol{X};\boldsymbol{S}) = E^*(\boldsymbol{X};\boldsymbol{S}) - TH(\boldsymbol{X})$ を最小化するために, 次のラグランジュ関数を導入する.

$$L = \phi^* + \sum_{a=1}^{N} A_a(\sum_{b=1}^{N} X_{a,b} - 1)$$
$$+ \sum_{b=1}^{N} B_b(\sum_{a=1}^{N} X_{a,b} - 1) + C(\sum_{a=1}^{N} X_{a,a} - (N - \lambda))$$

ここで,A_a $(a = 1, ..., N)$, B_b $(b = 1, ..., N)$, Cはラグランジュの未定係数である. このラグランジュ関数に対する停留条件は

$$\frac{\partial L}{\partial X_{a,b}} = \frac{\partial E^*}{\partial X_{a,b}} + T(logX_{a,b} + 1)$$
$$+ A_a + B_b + C\delta_{a,b} = 0 \qquad \text{(F.1)}$$

及び, 式 (3.12),(3.13),(3.14) になる. 式 (F.1) を解いたものは次の様になる.

$$X_{a,b} = \exp(-\frac{1}{T}\frac{\partial E^*}{\partial X_{a,b}})/(\alpha_a\beta_b\gamma^{\delta_{a,b}}) \qquad \text{(F.2)}$$

ここで$\alpha_a, \beta_b, \gamma$は$\alpha_a \equiv \exp(A_a/T + 1), \beta_b \equiv \exp(B_b/T), \gamma \equiv \exp(C/T)$ であり, ラグランジュの未定係数を変換したものである. 式 (3.16) を使って, 式 (F.2) を書き換えると, 式 (3.15) を得ることができる.

制約条件 (3.12) を満たすためには,

$$1 = \sum_a X_{a,b} = \sum_a \frac{U_{a,b}}{\alpha_a\beta_b\gamma^{\delta_{a,b}}}$$

でなければいけない. この式をβ_bについて解くと式 (3.18) を得る. 同様にして, 式 (3.17),(3.19) も得られる.

G. λ-interior DCN 方程式

ここでは, 式 (3.26),(3.27) の導出を行う. 式 (3.12),(3.13) の制約のもとで$\phi^*(\boldsymbol{X};\boldsymbol{S}) = E^*(\boldsymbol{X};\boldsymbol{S}) - TH(\boldsymbol{X}) + K^*(\boldsymbol{X})$ を最小化するために次のラグランジュ関数を導入する.

$$L = \phi^* + \sum_{a=1}^{N} A_a(\sum_{b=1}^{N} X_{a,b} - 1) + \sum_{b=1}^{N}(B_b \sum_{a=1}^{N} X_{a,b} - 1)$$

ここで,A_a $(a=1,...,N)$ B_b $(b=1,...,N)$ はラグランジュの未定係数である. このラグランジュ関数に対する停留条件は

$$\frac{\partial L}{\partial X_{a,b}} = \frac{\partial E^*}{\partial X_{a,b}} + T[logX_{a,b} + 1 + \delta_{a,b}(log(\sum_{a=1}^{N} X_{a,a} - M) + \theta)] + A_a + B_b = 0 \quad \text{(G.1)}$$

及び, 式 (3.12),(3.13) になる. 式 (G.1) は次の様に解ける.

$$X_{a,b} = exp(-\frac{\partial E^*}{T\partial X_{a,b}}/(\alpha_a\beta_b)) \quad \text{(G.2)}$$

$$X_{a,a}(\sum_{c=1}^{N} X_{c,c} - M) = exp(-\frac{\partial E^*}{\partial X_{a,a}}/(\alpha_a\beta_b) - \theta) \quad \text{(G.3)}$$

ここでα_a, β_bは$\alpha_a \equiv exp(A_a/T+1), \beta_b \equiv exp(B_b/T)$ であり, ラグランジュの未定係数を変換したものである. 式 (G.2) は $a \neq b$ の時, 式 (3.15),(3.26) と等価である. 式 (G.3) の添字 a についての和を計算すると$\sum_{a=1}^{N} X_{a,a}$についての 2 次式となる. この 2 次式を解くと

$$\sum_{a=1}^{N} X_{a,a} = \frac{M + \sqrt{M^2 + 4\sum_a \frac{U_{a,a}(t+1)}{\alpha_a\beta_a}}}{2}$$

となる. $\gamma = \sum_{a=1}^{N} X_{a,a} - M$とすると式 (3.29) を導ける. $a=b$ の時の式 (G.2) は, 式 (G.3) とγの定義を使うと直接に導ける.

H. ベンチマーク tai100a,tai80a の最良解

本研究ではベンチマーク tai100a,tai80a について, QAPLIB で公開されている最良解よりも良い解を計算した. 次に, 本研究で求めた最良解を示す.

I. tai80a の最良解 $E_{obj} = 1.35497e+07$ 0.060%の改善
π=(26,75,4,19,58,67,38,32,74,34,63,22,31,68,5,29,2,27,35,64,21,40,
17,70,44,18,25,54,73,48,12,3,24,14,71,51,6,16,33,56,47,46,72,10,66,61,
60,77,80,11,23,20,79,9,55,49,42,69,36,43,37,1,53,78,65,15,50,41,57,30,
28,45,7,13,8,52,76,59,62,39)

II. tai100a の最良解 $E_{obj} = 2.1108e+07$ 0.082%の改善

π=(59,16,58,56,84,9,81,67,98,64,38,1,45,60,22,29,10,72,57,55,83,47,
71,43,11,79,96,7,3,86,75,66,65,35,12,92,62,20,4,76,44,78,100,80,91,52,
69,18,54,94,63,37,48,31,42,33,53,61,13,99,21,32,90,95,40,15,93,28,41,
68,36,74,2,49,82,70,77,25,26,6,46,87,24,39,30,5,97,14,8,50,34,73,23,27,
88,89,19,51,85,17)