2.2.3. Chaotic neural network based on memory of cities

This tabu search neural network is also extended to the chaotic version in a similar way to the two-dimensional case. Namely, the output function for detecting the maximum output are implemented in dynamics of the internal states 文字(t). Then the chaotic neural network based on the tabu search memorizing cities can be realized by the following equations:

(12) (13) (14) (15)

If xi(t + 1)> 1/2, the city I is connected with the city j corresponding to the maximum of {} in Eq. (12), using the 2-opt exchange.

For numerical calculation, Eq. (14) can be reduced to the following equations: if t < s, (16)(17) where R = θ(1―kr). It should be noted again that we assume that xij(u) = 0 for u < 0.

In the following, we set s ― 1 in Eq. (14) same as the original chaotic neural network, s ― 1 = t, then the mode becomes the same as the original chaotic neuron model (Aihara, 1990; Aihara et al., 1990). Thus, chaotic dynamics can be realized by this neural network. Since this chaotic searching method requires only n chaotic neurons for an n-city TSP as shown in Fig. 3, we call this the one-dimensional method. This is applicable to much described in Section 2.1.

2.3. Reduction of computational workload

 In the case of applying the proposed methods to very large TSPs, such as 10⁴ or 10⁵ order problems, it still requires heavy calculation, even if the one-dimensional method is adopted. Then, we should reduce the computa-tional complexity of the methods in order to make it easier to apply them to very large TSPs. The basic idea is that, if we can reduce the cost for unimportant parts, we can spend more computational costs on important parts for finding better solutions. In this sub-section, we discuss a simple method for reducing the computational workload of the proposed for reducing the computational workload of the proposed chaotic search.

 In the case of Euclidean TSPs, it is very rare to connect very long paths.

Most of good TSP solutions have relatively shorter paths. Since it is rare for good solutions to include long paths, we can omit calculations for such paths. In the two-dimensional method, we reduce neurons which corre-wpond to very long paths. In the one-dimensional method, the workload for detection of the best j in Eq. (12) is reduced by omitting scanning j corresponding to cities which are far from the city i.

 Figs. 4 and 5 show the performance of the one-dimen-sional chaotic neural network (Section 2.2.3), whose computational amount is reduced, for two problems, KroA200 and Pcb1173, from TSPLIB (Reinelt, on line). In these figures, the left show the relation between the reduction rates of scanning for selecting the best j and the solutions which are evaluated by the mean values of the percentages of the gaps between the optimum solutions and the obtained solutions. The right figures show the rela-tion between the reduction rates and the CPU time. These experiments are done by Sum Ultra Sparc 5. In those figures, every run is cut at 10,000 iterations (t = 10,000). The mini-mum number of cities scanned as the city j in Eq. (12) is 20. From figs. 4 and 5, we find that although there is no signi-ficant difference on qualities of obtained solutions even if we reduce the computational complexity, the CPU time can be much decreased. Namely, when not all paths are taken into account, the reduction has such a big advantage that the calculation amount can be much decreased. Then, further search is available with the same CPU time.

2.2.3.都市の記憶に基づくカオスニューラルネットワーク

このタブーサーチニューラルネットワークも、2次元の場合と同様にカオスバージョンに拡張されます。すなわち、最大出力を検出する出力関数は、内部状態文字(t)のダイナミクスに実装されている。次に、タブー探索記憶都市に基づくカオス ニューラル ネットワークは、次の方程式によって実現できます。

(12) (13) (14) (15)

xi(t + 1)> 1/2 の場合、都市 I は式 (1) の {} の最大値に対応する都市 j に接続されます。 (12) 2-opt 交換を使用します。

数値計算の場合、Eq. (14) は、次の式に還元できます。t < s の場合、(16)(17) ここで、R = θ(1―kr)。 u < 0 の場合、xij(u) = 0 であると仮定していることに注意してください。

以下では、式に s ― 1 を設定する。 (14) 元のカオス神経回路網と同じ s ― 1 = t とすると、モードは元のカオスニューロンモデルと同じになる (相原, 1990; 相原ほか, 1990)。このように、このニューラルネットワークによってカオスダイナミクスを実現することができます。このカオス探索法は、図 3 に示すように、n 個の都市 TSP に対して n 個のカオスニューロンしか必要としないため、これを 1 次元法と呼ぶ。これは、セクション 2.1 で説明されている多くの場合に当てはまります。

2.3.計算負荷の軽減

提案手法を 10 点以上の TSP や 10 個のセット ライジャ オーダー問題のような非常に大きな TSP に適用する場合、1 次元の手法を採用しても計算量が多くなります。次に、メソッドの計算の複雑さを軽減して、非常に大きな TSP への適用を容易にする必要があります。基本的な考え方は、重要でない部分のコストを削減できれば、より良い解決策を見つけるために重要な部分により多くの計算コストを費やすことができるということです。このサブセクションでは、提案されたカオス検索の計算負荷を軽減するために、提案された の計算負荷を軽減する簡単な方法について説明します。

ユークリッド TSP の場合、非常に長いパスを接続することは非常にまれです。

優れた TSP ソリューションのほとんどは、比較的短いパスを持っています。良い解に長いパスが含まれることはまれであるため、そのようなパスの計算は省略できます。二次元法では、非常に長い経路に対応するニューロンを削減します。 1 次元法では、式 (1) の最良の j を検出するための作業量は次のようになります。 (12) 都市 i から遠い都市に対応するスキャン j を省略することで削減される．

イチジク図 4 と図 5 は、計算量が削減された 1 次元カオス ニューラル ネットワーク (セクション 2.2.3) の、TSPLIB (Reinelt、オンライン) の 2 つの問題、KroA200 と Pcb1173 についての性能を示しています。これらの図において、左は、最適解と得られた解とのギャップの百分率の平均値で評価された、最良のjを選択するための走査の縮小率と解との関係を示している。右の図は、削減率とCPU時間の関係を示しています。これらの実験は Sum Ultra Sparc 5 によって行われています。これらの図では、すべての実行が 10,000 回の反復 (t = 10,000) でカットされています。式で都市 j としてスキャンされた都市の最小数。 (12) は 20 です。図から。図 4 と図 5 から、計算量を減らしても、得られる解の品質に大きな違いはありませんが、CPU 時間は大幅に削減できることがわかります。すなわち、全てのパスを考慮しない場合、計算量を大幅に削減できるという大きなメリットがある。その後、同じ CPU 時間でさらに検索を行うことができます。