ANNs FOR COPs

人工神经网络解决组合优化问题

Contents

Introduction

ANNs: ANNs provide a fundamentally new and different approach to information processing. ANNs are capable of internally developing information processing capabilities for solving a problem when fed with appropriate information about the problem.

ANNs for COPs: Hopfield and Tank solved small instances of the Traveling Salesman Problem (TSP) with a Hopfield neural network in 1985. Many other types of COPs have since been tackled with ANNs in different application areas: routing and transportation, scheduling, timetabling, and many others.

Models: ANNs applied to COPs are mostly based on three alternative models: **Hopfield-Tank (H–T) and its variants**, **the elastic net (EN)** and the **self-organizing map (SOM).**

Hopfield neural networks

Discrete and Stochastic Hopfield Network

Continuous and Deterministic Hopfield Network

W_{ij}

- > 0 excitatory
- < 0 inhibitory

internal state u_i

$$u_i(t+1) = \sum_{j=1}^{n} W_{ij} v_j(t) + I_i \tag{1}$$

external state v_i

$$v_i(t+1) = f(u_i) = \begin{cases} 1 & \text{if } u_i > 0 \\ 0 & \text{if } u_i \leq 0 \end{cases} \tag{2}$$

$$E_d = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} v_i v_j - \sum_{i=1}^{n} I_i v_i$$

Hopfield neural networks

Discrete and Stochastic Hopfield Network

Continuous and Deterministic Hopfield Network

$$\frac{du_i}{dt} = \sum_{j=1}^{n} W_{ij} v_j - \frac{u_i}{\tau} + I_i \qquad (4)$$

$$v_i = f(u_i) \quad \text{and} \quad \tau = R_i C_i \qquad (5)$$

$$v_i = f(u_i) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i}{T}\right)\right) \qquad (6)$$

$$E_c = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} v_i v_j - \sum_{i=1}^{n} I_i v_i + \int_0^{v_i} f_i^{-1}(\mathbf{v})\, d\mathbf{v} \qquad (7)$$

Hopfield neural networks for COPs

Hopfield and Tank (H–T) realized that networks of neurons with this basic organization could be used to compute solutions to **specific optimization problems** by selecting weights and external inputs which appropriately represent the function to be minimized and the desired states of the problem. The updating of the neurons according to the differential equations given by (4) and (5) (or even the discrete versions (1) and (2)) ensures that both the energy function and the optimization problem are simultaneously minimized over time.

The network energy function is made equivalent to the objective function of the optimization problem needing to be minimized, while the constraints of the problem are included in the energy function as penalty terms.

Hopfield neural networks for TSP

$$X_{ij} = \begin{cases} 1 & \text{if city } i \text{ is in position } j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{minimise} \sum_{i=1}^{N} \sum_{\substack{k=1 \\ k \neq i}}^{N} \sum_{j=1}^{N} d_{ik} X_{ij} (X_{k,i+1} + X_{k,i-1}) \tag{8}$$

$$\text{subject to} \sum_{i-1}^{N} X_{ij} = 1 \quad \text{for all } j \tag{9}$$

$$\sum_{j=1}^{N} X_{ij} = 1 \quad \text{for all } i \tag{10}$$

$$X_{ij} \in \{0, 1\} \quad \text{for all } i, j \tag{11}$$

Hopfield neural networks for TSP

$$E = \frac{A}{2}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N} X_{ij}X_{kj} + \frac{B}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{l=1\\l\neq j}}^{N} X_{ij}X_{il} + \frac{C}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij} - N\right)^2$$

$$+ \frac{D}{2}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N}\sum_{j=1}^{N} d_{ik}X_{ij}(X_{k,i+1} + X_{k,i-1}) \qquad (12)$$

$$E_{\mathrm{d}} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} W_{ijkl}X_{ij}X_{kl} - \sum_{i=1}^{N}\sum_{j=1}^{N} I_{ij}X_{ij} \qquad (13)$$

$$E_{\mathrm{d}} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}v_i v_j - \sum_{i=1}^{n} I_i v_i \qquad (3)$$

pseudocode

Step 0: Preliminary Tasks
0.1 Construct an energy function using a penalty parameter approach.
0.2 Expand energy function and infer network weights and external inputs.
Step 1: Initialization Tasks
1.1 Initialize neuron states to random values.
1.2 Select A,B,C,D.
1.3 Select T, the parameter of the continuous transfer function
Step 2: If energy function has converged to local minimum proceed to Step 5, otherwise proceed to step 3
Step 3: Repeat n times:
3.1 Randomly choose a neuron i to update (if using discrete time dynamics).
3.2 Update and using equations (1)-(2) or (3)-(4).
Step 4: Go back to Step 2.
Step 5: Examine final solution matrix and determine feasibility and optimality.
Step 6: Adjust parameters A,B,C,D if necessary to obtain a satisfactory solution, re-initialize neuron states, and repeat from Step 2.

Extensions

- **the valid subspace approach of Aiyer and the subsequent work of Gee**
- **deterministic approaches**
  - problem specific enhancements
  - the use of alternative neuron models within the Hopfield network
- **stochastic approaches**
  - replace sigmoidal transfer function with a stochastic decision-type function;
  - add noise to the weights of the network;
  - add noise to the external inputs of the network;
  - any combination of the above methods.

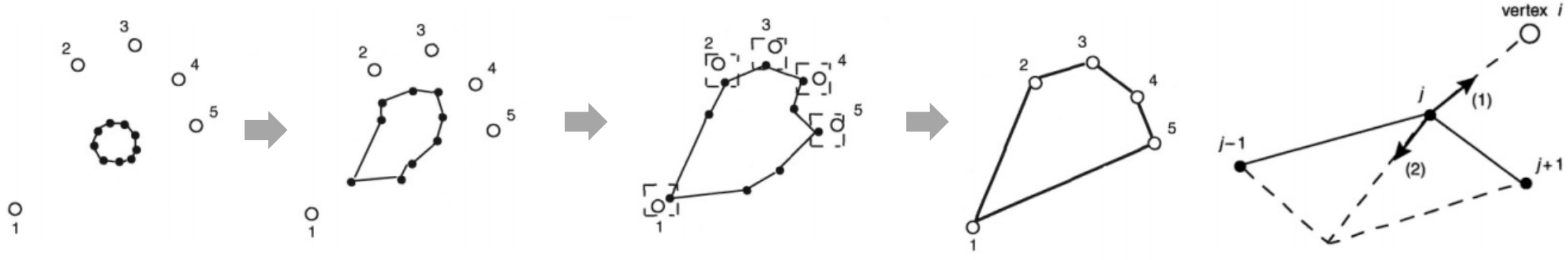
Practical Issues

- Efficiency Issues
  - Problem Representation X_{ij}
  - Energy Function
  - Penalty Factors
- Problematic Issues
  - Diagonal Weights
  - Simulation
  - Initial States
  - Updating Mode
  - Solution Integrality
  - Termination Criteria

$$u_i(t+1) = u_i(t) + \Delta t \frac{du_i}{dt} = u_i(t)\left[1 - \frac{\Delta t}{\tau}\right] + \sum_{j=1}^{n} W_{ij} v_j(t) + I_i \qquad (24)$$

Euler Approximation

Elastic Net



Step 0: $K \leftarrow K_0; Y_j \leftarrow Y_j^0,\ j = 1, \ldots, M;$

Step 1: Repeat rep times

1.1 Update the coordinates Y_j of ring point $j, j = 1, \ldots, M;$

1.2 If $\min_{j=1,\ldots,M} d_{X_i Y_j} \le \varepsilon, i = 1, \ldots, N,$ then STOP;

Step 2: $K \leftarrow \alpha K (0 < \alpha < 1);$

Step 3: Go back to Step *1*.

$$Y_j \leftarrow Y_j + \Delta Y_j$$

$$\Delta Y_j = \alpha \sum_{i=1,\ldots,N} w_{ij} (X_i - Y_j) + \beta K (Y_{j+1} + Y_{j-1} - 2Y_j)$$

$$w_{ij} = \frac{\phi(d_{X_i Y_j}, K)}{\sum_{k=1,\ldots,M} \phi(d_{X_i Y_k}, K)}, \quad i = 1, \ldots, N$$

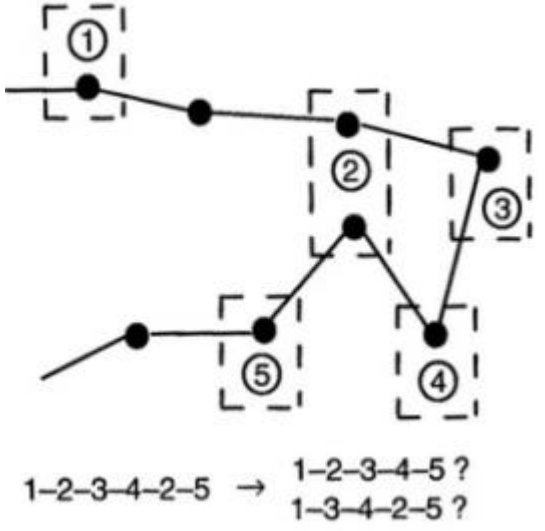
$$\phi(d, K) = e^{-d^2/2K^2}$$

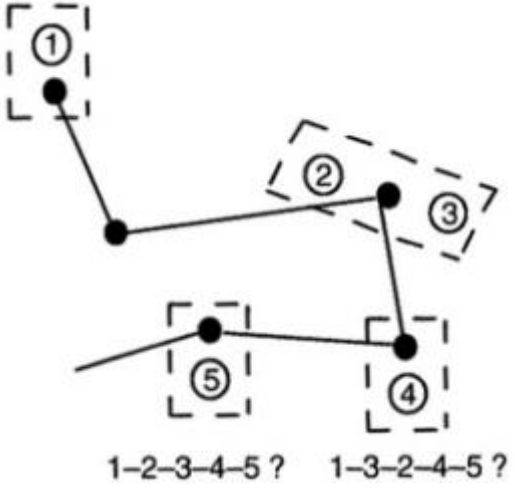
$$\Delta Y_j = -K \frac{\partial E_K}{dY_j}$$

$$E_K = -\alpha K \sum_{i=1,\ldots,N} \ln \sum_{j=1,\ldots,M} \phi(d_{X_i Y_j}, K) + \frac{\beta}{2} \sum_{j=1,\ldots,M} d^2_{Y_j Y_{j+1}}$$

Practical Issues

- Efficiency Issues
  - Filtering
  - Hierarchical EN
- Problematic Issues
  - Normalization of Coordinate Vectors
  - Initial Ring
  - Ring Migration
  - Interpretation of the Final Configuration





Self-organizing Map

Step 0: Initialization. $i \leftarrow 0$; $Y_j \leftarrow Y_j^0$; $j = 1, \ldots, M$;

Step 1: Competition.

1.1 $i \leftarrow (i + 1) \bmod (N + 1)$;

1.2 $o_j \leftarrow d_{X_i Y_j}, j = 1, \ldots, M$;

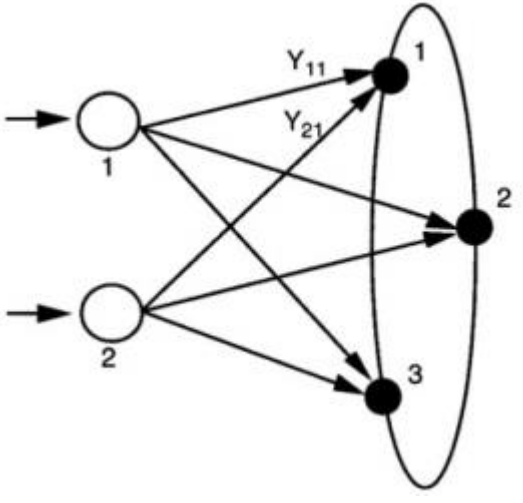
1.3 $o_j \leftarrow \min_{j=1,\ldots,M}\{o_j\}$;

Step 2: Weight adjustment.

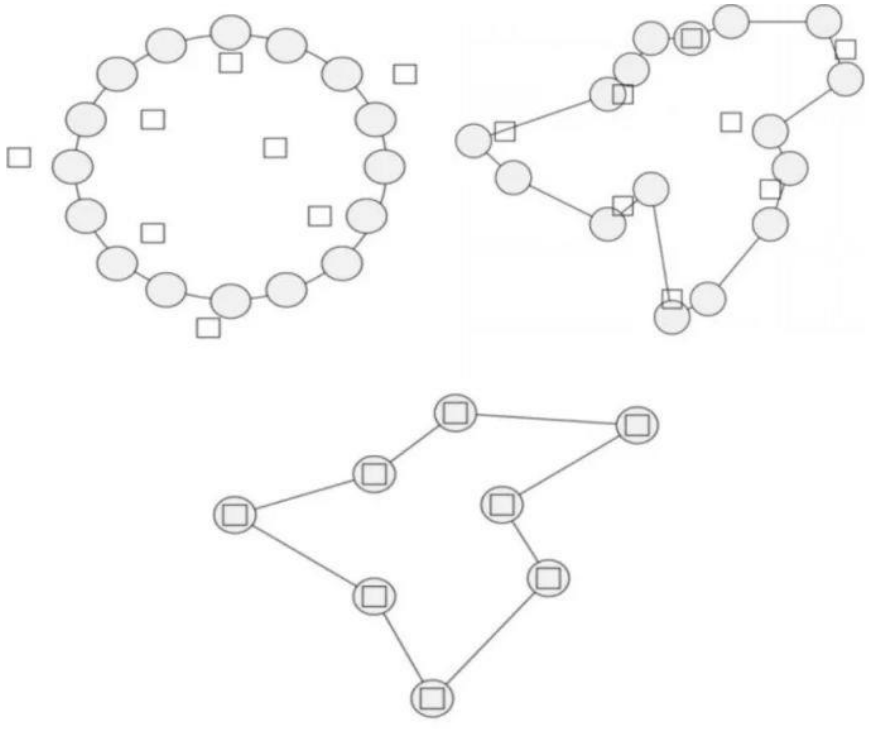
$Y_j \leftarrow Y_j + \mu f(j, j^*)(X_i - Y_j), j = 1, \ldots, M, 0 < \mu < 1$;

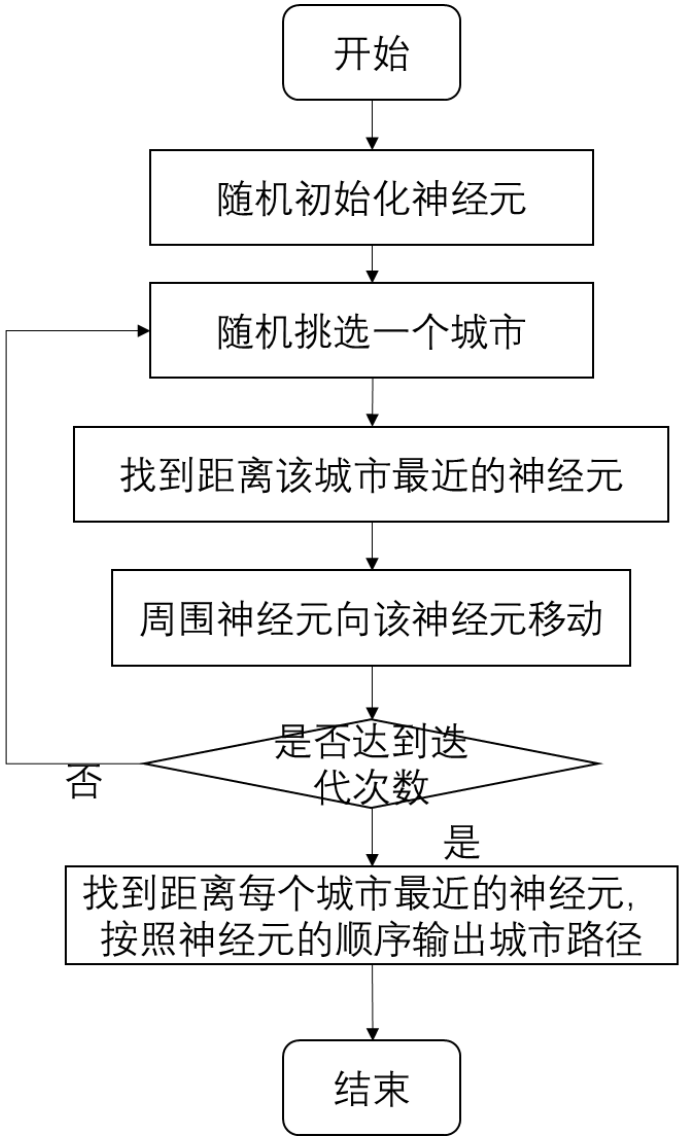
Step 3: If $\min_{j=1,\ldots,M} d_{X_i Y_j} \leq \varepsilon, i = 1, \ldots, N$, then STOP;

Step 4: Go back to Step 1.



Self-organizing Map





Practical Issues

- Efficiency Issues
  - the update mechanism

$$f(j, j^*) = \left(1 - \frac{d''_{jj^*}}{L}\right)^{\beta}, \quad if\ d''_{jj^*} < L$$
$$= 0, \qquad otherwise \qquad (25)$$

where $d''_{jj^*} = \min(|j - j^*|, m - |j - j^*|)$.

- Problematic Issues
  - Ring Point Freezing

Performance

- Both SOM and EN are closely related. They both involve migration of a ring towards vertices, although the mechanism for updating the location of the ring points is different.
- In the case of EN, the update is defined through the minimization of an appropriate energy function. The SOM does not require such a function.
- Overall, Both EN and SOM outperform the Hopfield model on the TSP. For problems with up to a few hundred cities, EN often ends up with slightly better solutions than SOM. However, it is also more computationally expensive, as it requires more iterations to converge. SOM scales up better and has been applied on much larger problems.

conclusions

From a metaheuristics viewpoint, neural networks can be seen as an alternative technique with the current potential to match the performance of better known algorithms such as tabu search and simulated annealing.

The Hopfield network method generalizes to a broad range of combinatorial problems, but the cost of this generalization is a reduction in efficiency and scalability.