Mathematical Biosciences and Engineering

MBE, 19(4): 3427–3448.
DOI: 10.3934/mbe.2022158
Received: 12 October 2021
Revised: 09 January 2022
Accepted: 16 January 2022
Published: 25 January 2022

http://www.aimspress.com/journal/MBE

Research article

Solving the TSP by the AALHNN algorithm

Yun Hu and Qianqian Duan*

Department of Electric and Electronic Engineering, Shanghai University of Engineering Science, 333 Longteng Road, Shanghai 201620, China

* **Correspondence:** Email: dqq1019@163.com.

Abstract: It is prone to get stuck in a local minimum when solving the Traveling Salesman Problem (TSP) by the traditional Hopfield neural network (HNN) and hard to converge to an efficient solution, resulting from the defect of the penalty method used by the HNN. In order to mend this defect, an accelerated augmented Lagrangian Hopfield neural network (AALHNN) algorithm was proposed in this paper. This algorithm gets out of the dilemma of penalty method by Lagrangian multiplier method, ensuring that the solution to the TSP is undoubtedly efficient. The second order factor added in the algorithm stabilizes the neural network dynamic model of the problem, thus improving the efficiency of solution. In this paper, when solving the TSP by AALHNN, some changes were made to the TSP models of Hopfield and Tank. Say, constraints of TSP are multiplied by Lagrange multipliers and augmented Lagrange multipliers respectively, The augmented Lagrange function composed of path length function can ensure robust convergence and escape from the local minimum trap . The Lagrange multipliers are updated by using nesterov acceleration technique. In addition, it was theoretically proved that the extremum obtained by this improved algorithm is the optimal solution of the initial problem and the approximate optimal solution of the TSP was successfully obtained several times in the simulation experiment. Compared with the traditional HNN, this method can ensure that it is effective for TSP solution and the solution to the TSP obtained is better.

Keywords: TSP; HNN; Lagrange neural network algorithm; augmented Lagrangian; nesterov acceleration technique

1. Introduction

In a broad sense, combinatorial optimization problem involves finding the "best" object from a limited set of objects. "Best" is measured by a given evaluation function, which maps the object to a score or cost. The goal is to find the object with the highest evaluation score and the lowest cost. Combinatorial optimization often involves sorting, classification, screening and so on. In terms of discrete cop problem, the goal is to find a set, an arrangement or a graph from all feasible solutions.

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem. There are highly similar problems in multiple disciplines. This problem is described as follows: a traveling salesman needs to visit a number of (n) cities; he returns to the starting point after visiting every city once; then the shortest path among all paths that meet this condition is the optimal solution to this problem. The TSP is a NP-hard problem in classical combinatorial optimization, which is characterized by a fact that there is no method with polynomial-time complexity that can accurately find the optimal solution at present. Therefore, the TSP is mostly solved by heuristic algorithms such as genetic algorithm (GA), neural network algorithm and ant colony optimization (ACO). Hence, it is of great significance and theoretical value to study this problem.

The study of artificial neural network had been in a silence period of for more than ten years since the late 1960s, but began to flourish again in the early 1980s. The research boom has continued till today. At that time, the concept of HNN was proposed in the two classic papers of Hopfield and Tank [1,2]. Y. Zhu [3] addresses the problems of synchronization and state estimation for a class of discrete-time hierarchical hybrid neural networks (NNs) with time-varying delays. Wilson and pawleyi [4] found that the energy functions of Hopfield and tank may not get effective solutions. And in [5] the state estimation problem for a class of discrete-time switched neural networks with modal persistent dwell time (MPDT) switching and mixed time delays is investigated.Hopfield neural network is a recursive neural network, which was invented by John Hopfield in 1982. Hopfield network is a neural network combining storage system and binary system. It guarantees the convergence to the local minimum, but it may also converge to the wrong local minimum rather than the global minimum. Hopfield network also provides a model to simulate human memory. The success Hopfield and Tank made in the solution to the TSP by HNN inspired researchers to get down to dig deep into solving the NP-hard problem by neural network. It is found that neural network has many advantages over other methods in solving the TSP. M. Waqas et al. [6] described a neural network optimizer/scheduler. A new initialization rule is proposed to improve the performance of the system and make the system converge to the optimal solution in a shorter time. Therefore, various methods to improve the convergence of HNN were proposed by researchers after various improvements and studies of solving combinatorial optimization problem by HNN.

L. García et al. [7] pointed out CHNN (Continuous Hopfield Neural Network) is used as the optimizer to make the neural network competitive in solving k-opt problem. I. Valova et al. [8] discussed The implementation of some new optimization algorithms and traditional deep learning (DL) optimization algorithms. S. Z. Li [9] transformed the relaxation labeling problem into a constrained optimization problem and solved it by Lagrange multiplier method. D. Kaznachey et al. [10] and Z. Wu et al. [11] solved the combinatorial optimization problem by using neural network and logarithmic descent algorithm respectively. A. Pvy et al. [12] proposed a hybrid artificial neural network and genetic algorithm to predict the hysteresis loop and magnetic properties of alloys. The research shows that artificial neural network, as a powerful calculation technology in nonlinear system modeling, can

be reliably applied to the prediction of nonlinear systems. A. Rachmad et al. [13] and A. Mazeev et al. [14] improved simulated annealing algorithm to solve minimum distance and TSP problem respectively. S. Z. Li [15] proposed the relaxation labeling to improve HNN by Lagrangian multiplier method and hierarchical HNN. Honari-Latifpour et al. [16] proposed the application of three state plane Potts model in combinatorial optimization problem. S. Zhang et al. [17] analyzed in detail a type of neural network for general nonlinear programming, i.e., equality and inequality constraints, etc. This method is based on the optimization of Lagrangian multiplier theory and seeks to provide solutions that meet the necessary conditions for optimality. S. S. Kia [18] discusses how to use the idea of augmented Lagrange method to solve the convex optimization problem with affine constraints, and obtain the distributed solution with convergence guarantee in the network with convex local cost. Y. Hu et al. [19] proposed A bi-directional graph neural network to solve arbitrary symmetric neural networks. U. P. Wen et al. [20] by dividing the application direction of HNN into three categories: linear, nonlinear and mixed integer, that we can better understand the application direction of HNN.

As it is difficult for HNN to converge to an efficient solution when solving the TSP, an AALHNN was designed in this paper. Firstly, the TSP was converted into a neural network model to solve a constrained optimization problem. For the convenience of solving the TSP by AALHNN, some changes were made to the corresponding constraints and objective functions in this paper. Secondly, AALHNN made some amendments to the defect that HNN could not get an efficient solution when solving the TSP. That is, the stability of neural network was improved by adding a quadratic term to AALHNN and the effectiveness and convergence of the algorithm were theoretically proved. Finally, the effectiveness of the algorithm was verified by some experiments of solving the TSP by simulation.

The rest of parts in this paper are included as follows. Part 2 is the model of the TSP to be solved. That is, the description of the problem model is illustrated herein. Part 3 refers to the dynamic iterative process of augmented Lagrangian neural network energy function and proves the convergence of the algorithm. A method based on nesterov technique is proposed to update the Lagrange factor which does not affect the convergence and robustness of ALHNN algorithm. Experimental process and data analysis and summary are provided in Part 4. Part 5 is the summary of this paper.

2. Model of the TSP

2.1. Problem description

The TSP, i.e., the traveling salesman problem, is a classical combinatorial optimization problem. This problem is described as follows. Suppose a traveling salesman has to visit a number of cities, he needs to choose a travel path, the constraints on which are that every city must be traversed once and only once and final arrival city must be the departure one so as to guarantee the salesman's profit. The ultimate goal of path selection is to obtain an efficient one, the total length of which should be the minimum among the all. For the salesman, the route length usually varies according to the sequence he chooses to traverse all the cities. For instance, the traversal length is different according to the two paths, i.e., A-H-G-F-E-D-C-I-B-A and A-H-G-F-E-D-C-B-I-A in Figure 1(a),(b), but both of them satisfy the constraints of the problem, which means they are both efficient solutions. When the number of cities is not great, the optimal path may be obtained by the method of exhaustion. Yet, when cities are considerable enough in number, the TSP is a typical NP-hard combinatorial optimization problem with the characteristic that it is always easy to describe but extremely difficult to solve to the extent

that the optimal solution often cannot be obtained. In such cases, an approximate optimal solution can be obtained only by heuristic algorithm. Among various intelligent methods for solving the TSP, there is no universal optimal solver for all problem instances. This fact greatly facilitates the selection of solution algorithm. Neural networks generate solutions by fully representing a graph with a set of coordinates followed by capturing graphic information from the coordinates. It is more convenient to apply the TSP model on any symmetric graph here in reality [21]. Therefore, solving the TSP by neural networks has been proved to have great potential [22]. Meanwhile, under appropriate parameter settings, the network can be made more competitive by using continuous HNN [23]. Therefore, in this paper, in addition to referring to the previous experimental parameters, some parameters were also set in a reasonable and independent way, thus making the augmented Lagrangian neural network algorithm designed herein for solving the TSP more stable and effective.

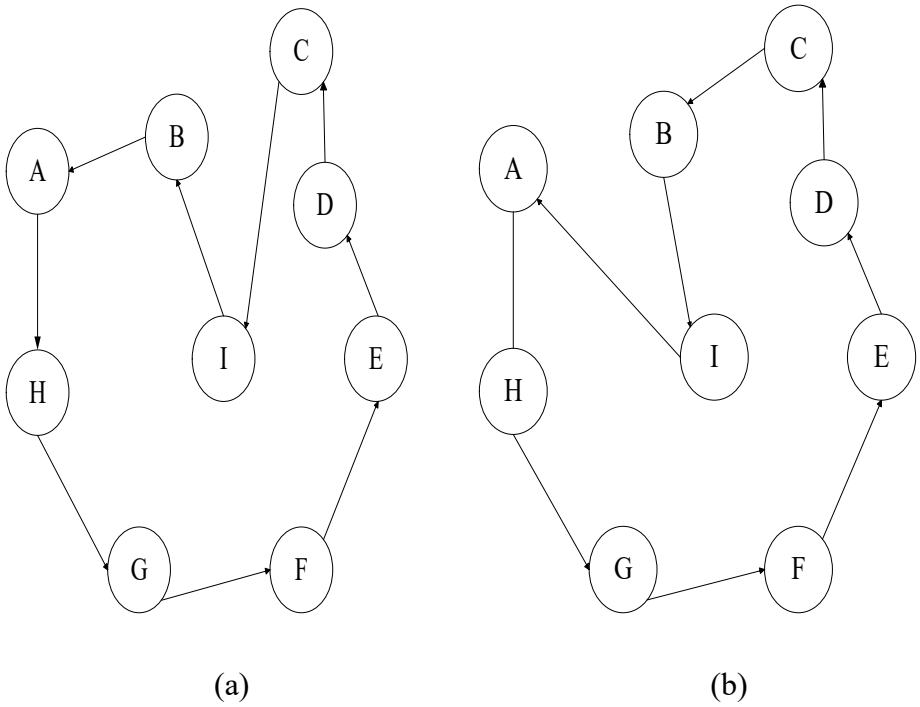


Figure 1. (a) travel paths a; (b) travel paths b.

2.2. Mathematical model of the TSP

When solving the TSP by neural networks, this problem can be regarded as a label assignment problem. Let $S = \{1, \mathrm{L}\ , m\}$ represent the set of m element, and $1 = \{1, \mathrm{L}\ , M\}$ represent the set of M cities, in which $m = M$. An M-dimensional vector $p_i = [p_i(I) \mid I \in 1]$ is used to represent the assignment status of $i \in S$. The actual value $p_i(I) \in [0, 1]$ reflects the status in which i is assigned to label I. The matrix $p = [p_i(I) \mid i \in S, I \in 1]$ is the results of assignment.The Generative Adversarial Network (GAN) is composed of a generator and a discriminator.

Initially, Hopfield and Tank [2] proposed the following energy function for the TSP of m citie

$$
\begin{aligned}
E_{HT}(p) = & \frac{A}{2}\sum_{I}\sum_{J\neq i}\sum_{I} p_i(I)p_j(I) \\
& + \frac{B}{2}\sum_{i}\sum_{I}\sum_{J\neq I} p_i(I)p_j(J) \\
& + \frac{C}{2}\left(\sum_{i}\sum_{I} p_i(I) - m\right)^2 \\
& + \frac{D}{2}\sum_{i}\sum_{I}\sum_{J\neq I} d_{IJ}p_i(I) * \left[p_{i+1}(J) + p_{i-1}(J)\right]
\end{aligned} \tag{1}
$$

where d_{IJ} is the distance between city I and city J, while A, B, C and D are some constants. Variable $p_i(I) \in (0,1)$ represents the neural output of city I at position i (position $m+1$ is the same as 1). The relation between them and the internal variable $u_i(I)$ is $p_i(I) = 1/(1+e^{-u_i(I)/T})$, where T is a parameter for temperature. When $T \to 0$, all $p_i(I)$ are forced to take an extreme value. That is, the value of every $p_i(I)$ is 1 or 0. The first three terms on the right side of the equation are the constraints that generate effective travel paths and the last term determines the path length. The variable parameter is the solution to the TSP. HNN is the main neural network to solve combinatorial optimization problems. The structure of HNN uses three common methods i.e., penalty function, Lagrange multiplier, original method and dual method to construct the energy function. When the energy function converges to a steady state, the approximate optimal solution $p^* = \arg\min_p E(p)$ to this problem is obtained.

When studying the equation above, Wilson and PawleyI [4] found that there is a probability that this energy function cannot get an efficient solution. The reason for this is that the equation above may obtain a local minimum, which corresponds to an invalid travel path. In order to solve this problem, Brandt et al. proposed a corrected energy function as follows:

$$
\begin{aligned}
E_{Brandt}(p) = & \frac{A}{2}\sum_{i}\left(\sum_{I} p_i(I) - 1\right)^2 \\
& + \frac{B}{2}\sum_{I}\left(\sum_{i} p_i(I) - 1\right)^2 \\
& + \frac{C}{2}\sum_{i}\sum_{I} p_i(I)\left(1 - p_i(I)\right) \\
& + \frac{D}{2}\sum_{i}\sum_{I}\sum_{J\neq I} d_{IJ}p_i(I) * \left[p_{i+1}(J) + p_{i-1}(J)\right].
\end{aligned} \tag{2}
$$

Although this made it converge better to travel paths than the equation proposed by Hopfield and Tank does, the quality of these paths is inferior to that of the original paths. In terms of theoretical analysis, it is reasonable because HNN is a classic use of penalty method. In order to make the penalty method converge to a feasible solution, the weighting factor of the penalty term should be big enough. However, as the penalty term becomes stronger and the constraints on the original problem become relatively weaker, the quality of the solution is also deteriorating. Even worse is that as they become ever-greater, the problem itself becomes pathological accordingly. This is a dilemma in penalty method.

Therefore, neither of the above-mentioned methods can ensure that the solution obtained is an efficient and the optimal solution. The potential of HNN for solving combinatorial optimization problems is limited to some extent due to some of its own problems [24]. The AALHNN method designed herein helps the TSP converge to high-quality solutions by improving HNN by the augmented Lagrangian method. The TSP model in this paper is given in Eqs (3) to (5).

In this paper, the TSP is established as HNN dynamic system model by the same steps as HNN:

Step1: Each effective path of the TSP is extracted as a m*m-dimensional 0-1 digital matrix (commutation matrix). There is only one single 1 on each row and each column of the matrix while the others are 0. The length of each corresponding path should be calculated.

Step2: The commutation matrix in Step1 is associated with a neuron network composed of N neurons, and each element in the matrix corresponding to each path is associated with the corresponding neuronal steady-state output.

Step3: An energy function E should be found to solve the constrained optimization of the TSP.

Step4: The moment the minimum value of energy function E is calculated, record the connection weight matrix of the neural network and the bias parameter of the neural network.

The neural network designed by the steps above can solve the TSP. Then, the steady-state output of the network is the regional optimal solution to the TSP, and also the minimal point of the energy function [25]. Therefore, the extremum of the energy function can be obtained through the steady-state output obtained. At this moment, the search time for the neural network to reach the steady state is that for solving the TSP, and the calculating process is the iterative process of the dynamic equation of the neural network. Therefore, the objective function of an optimization problem needs to be at first converted into the energy function of the neural network, and the variables of the initial problem should correspond to the state of the network. In such cases, the combinatorial optimization problem TSP can be solved by HNN.

2.3. Constrained minimization problem of the TSP

Firstly, the original combinatorial optimization problem is modified to the following constrained minimization problem:

$$\min_{p} \quad E(p) = \frac{1}{2}\sum_{i}\sum_{I}\sum_{J\neq I} d_{IJ}p_i(I) * \left[p_{i+1}(J) + p_{i-1}(J)\right] \tag{3}$$

$$subject \quad to \quad C_k(p) = 0 \quad k = 1, \mathrm{L} \quad K \tag{4}$$

$$p_i(I) \geq 0 \quad \forall i \in S, \forall I \in 1 \tag{5}$$

where $E(p)$ is the objective function of the total length of the travel path, K is an integer, C_k is some real functions and the constraint is that the value should be 0. The final solution p_k will be subject to additional constraints:

$$p_i^*(I) = 0 \quad or \quad 1 \quad \forall i, I \tag{6}$$

$C_k(p)$ in Eq (4) is the following equality constraint:

$$C_1^i(p) = \sum_I p_i(I) - 1 = 0 \quad \forall i \tag{7}$$

$$C_2^I(p) = \sum_i p_i(I) - 1 = 0 \quad \forall I \tag{8}$$

$$C_3^{i,I}(p) = \sum_{j \neq i} p_i(I) p_j(I) = 0 \quad \forall i, I \tag{9}$$

$$C_4^{i,I}(p) = \sum_{J \neq I} p_i(I) p_j(I) = 0 \quad \forall i, I \tag{10}$$

$$C_5^{i,I}(p) = p_i(I)(1 - p_i(I)) = 0 \quad \forall i, I. \tag{11}$$

$$K = 2m + 2m * m + m * m = 3m^2 + 2m \tag{12}$$

Therefore, there is a total of a number of (K) equality constraints. They can be classified into three categories: the first category includes terms C_1 and C_2, the second does terms C_3 and C_4, and the third does term C_5. In this paper, when solving the objective function Eq (3) by the AALHNN method, the function C was not simply added to the path length function as a penalty term. Instead, an augmented constraint optimization term was combined to form an augmented Lagrangian function.

3. Solving the TSP by the ALHNN algorithm

The approximate optimal solution of the TSP was obtained by the ALHNN algorithm. The basic structure of neural network is to multiply the output of each neuron by some weights and then send it to the input end of other neurons, thus forming a total feedback structure [26]. The way how algorithm works is as follows. The objective function and constraints of the TSP are mapped to energy functions (e.g., Eqs (3) to (5) in Section 2.3) of a Hopfield model, Iterative updating of neural network by AALHNN. When the iterative convergence of the network reaches a certain stable solution, the energy function reaches some local minimum value. After these local minima were determined to meet the conditions for the approximate optimal solution, a solution that meets the conditions was obtained, i.e., the approximate optimal TSP path.

The algorithm flow chart is as follows:

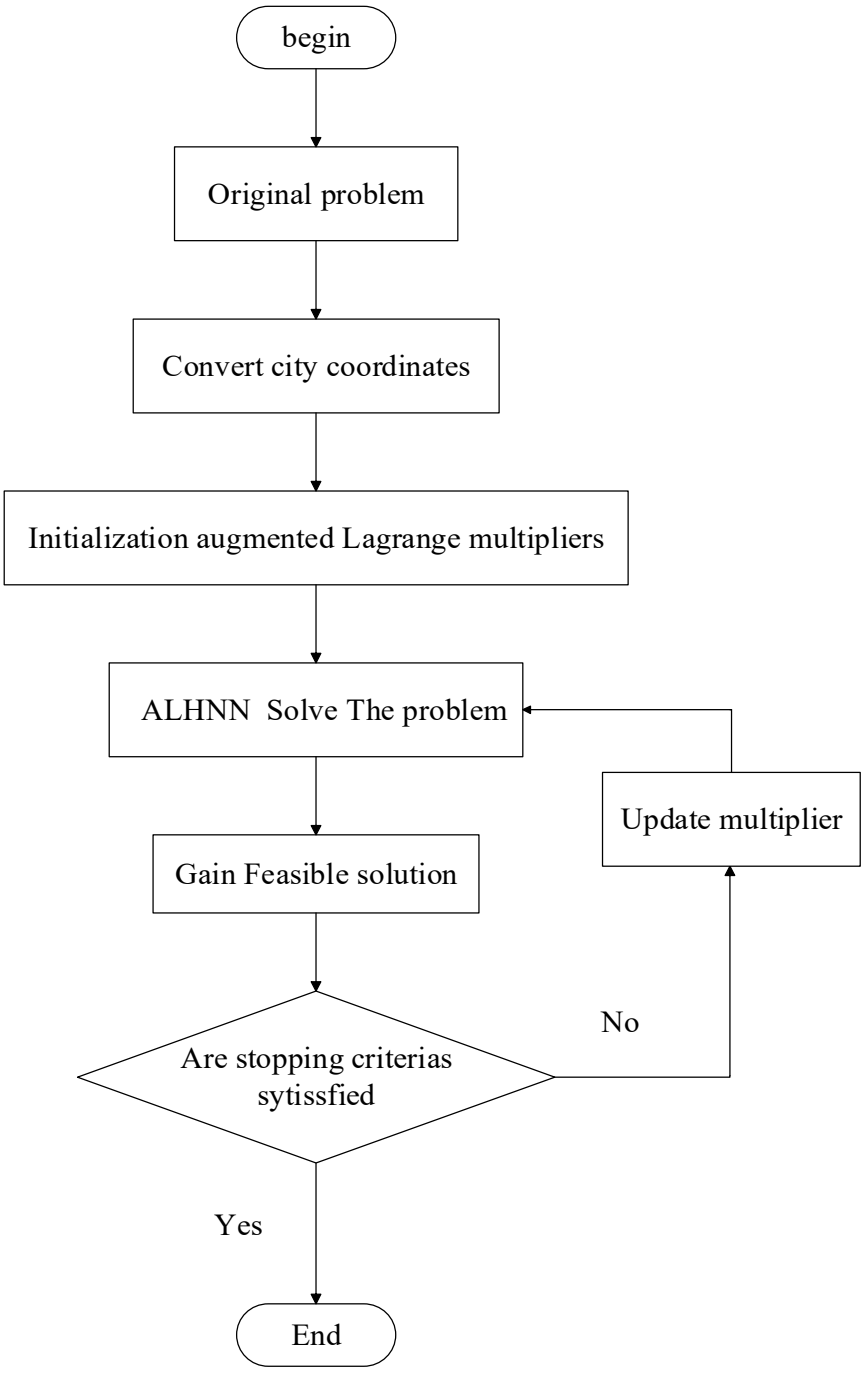


Figure 2. Flowchart of augmented Lagrangian Hopfield neural network.

3.1. Solving the minimum constraint by the Lagrangian neural network

Like the method proposed by Hopfield, the internal variable $u_i(I) \in (-\infty, +\infty)\ (\forall i, I)$ was introduced and associated with $p_i(I)$ through Eqs (13) and (14).

$$p_i(I) = \psi_T(u_i(I)) \tag{13}$$

$\psi_T(X)$ is a Sigmoid Function controlled by the temperature parameter $T > 0$.

$$\psi_T(x) = 1 / [1 + e^{-x/T}]. \tag{14}$$

Because of the internal variable introduced u, the energy function $E(u)$ could be regarded as $E(p(u))$. This method made the value of $p_i(I)$ within the range of (0,1), which satisfies the condition of Inequality Constraint Eq (5). Equality constraint Eq (4) was added to the Lagrangian function by the Lagrangian multiplier method. When $T \to 0^+$, the value of $p_i(I)$ will be 0 or 1 (depending on the positivity or negativity of $u_i(I)$), which meets the condition of constraint Eq (6). Finally, the following Lagrangian function was defined:

$$L(p, \gamma) = E(p) + \sum_k \gamma_k C_k(p) \tag{15}$$

where, γ_k is a Lagrangian multiplier. The complete Lagrangian function of equality constraints Eqs (7) to (11) for solving the travelling salesman problem (TSP) is:

$$\begin{gathered} L(p, \gamma) = E(p) + \sum_i \gamma_1^i C_1^i(p) + \sum_I \gamma_2^I C_2^I(p) + \sum_{i,I} \sum_{k=3}^{5} \gamma_k^i C_k^{i,I}(p) \\ C_k(p^*) = 0. \end{gathered} \tag{16}$$

This function is composed of m^2 p variables and number of K γ variables. Since p^* is the local minimum solution, (p^*, γ^*) must be the stable point of the Lagrangian function, and therefore,

$$\begin{aligned} \nabla_p L(p^*, \gamma^*) &= 0 \\ \nabla_\gamma L(p^*, \gamma^*) &= 0 \end{aligned} \tag{17}$$

Now (p^*, γ^*) satisfies

$$L(p^*, \gamma) \le L(p^*, \gamma^*) \le L(p, \gamma^*) \tag{18}$$

The gradient (first derivative) value of the objective function at this point is 0, but one direction starting from the change point is the maximum point of the function, and the other direction is the minimum point of the function. This point is defined as the saddle point.

It can be seen from Eq (18) that when p is constant, $L(p, \gamma)$ takes the maximum value when $\gamma = \gamma^*$, and when γ is constant, $L(p, \gamma)$ takes the minimum value when $p = p^*$.That is, (p^*, γ^*) is a saddle point.

Therefore, when we know the value of γ^*, we find p^*, and when we find p^*, the value of γ^* can also be obtained. This means that robustness of Lagrange algorithm is very strong. Augmented Lagrange has better robustness and convergence. It is more suitable for equality constraints. For inequality constraints, we need to deform the problem to obtain the intermediate variables, and then discuss and solve the intermediate variables.

According to Eq (18), if (p^*, γ^*) is a saddle point of $E(p)$, it can be seen from Eq (17) that (p^*, γ^*) meets the condition of being the minimum constraint of the Lagrangian function, and thus (p^*, γ^*) is a local minimum point of Lagrangian Function Eq (16) [27].

Such a saddle point can be found by solving the following kinetic equation by a basic differential multiplier:

$$\frac{dp_i(I)}{dt} = -\frac{\partial L(p, \gamma)}{\partial p_i(I)} \tag{19}$$

$$\frac{d\gamma_k}{dt} = +\frac{\partial L(p, \gamma)}{\partial \gamma_k}. \tag{20}$$

Such an equation causes energy to fall as p goes up and rise as γ goes up. This process will be analyzed in detail in Section 3.2.

3.2. Proof of convergence of the augmented Lagrangian neural network algorithm

Lagrangian function Eq (16) could be augmented into the following augmented Lagrangian function by adding a penalty term $[C_k(p)]^2$:

$$L_\beta(p, \gamma) = E(p) + \sum_k \gamma_k C_k(p) + \frac{1}{2}\sum_k \beta_k [C_k(p)]^2 \tag{21}$$

where, $\beta_k > 0$ is a penalty factor.

The kinetic equation used to find the saddle point was changed to:

$$\begin{aligned} \frac{dp_i(I)}{dt} &= -\frac{\partial L_\beta(p, \gamma)}{\partial p_i(I)} \\ &= -\frac{\partial E(p)}{\partial p_i(I)} - \sum_k \gamma_k \frac{\partial C_K(p)}{\partial p_i(I)} - \sum_k \beta_k C_k(p) \frac{\partial C_k(p)}{\partial p_i(I)} \\ \frac{d\gamma_k}{dt} &= +\frac{\partial L_\beta(p, \gamma)}{\partial \gamma_k} \\ &= +C_k(p). \end{aligned} \tag{22}$$

This dynamic equations corresponds to the improved differential multiplier. $p_i(I)$ and γ_k represent the state variables of corresponding neurons. The neurons in the neural network model were classified, according to the different functions of the variables in the neural network model, into two categories, i.e., Lagrangian neuron γ_k and variable neuron $p_i(I)$. It can be seen from the state equation of neural network model (22) that γ_k made the trajectory of the network fall into the feasible region; and $p_i(I)$ reduced the Lagrangian function and made it always decrease with $p_i(I)$ and

increase with γ_k along the trajectory of the neural network model. Finally, when the neural network model iterated to the steady state,the stable point (p^*, γ^*) satisfies Eq (23),

$$\frac{dL}{dt}\Big|_{(p^*,\gamma^*)} = \frac{\partial L}{\partial p} * \frac{dp}{dt}\Big|_{(p^*,\gamma^*)} + \frac{\partial L}{\partial \gamma} * \frac{d\gamma}{dt}\Big|_{(p^*,\gamma^*)} = 0 \tag{23}$$

In the above Eq (23), $\frac{dL}{dt}\Big|_{(p^*,\gamma^*)}$ means the sum of the results after partial derivation of p and γ respectively[28], In other words, the stable point of neural network model (22) is the stable point of the Lagrangian function we constructed. Equation (23) represents the sum of two partial derivatives are equal to zero. Because dynamic equations corresponds to the improved differential multiplier, So the differential of L over t is equal to the partial derivative of L at point P Plus the partial derivative of L at point R.

Lemma 3.2.1. If the following equation is satisfied, (p^*, γ^*) is an equilibrium point of neural network model (22).

$$\begin{aligned} \nabla_p L(p^*, \gamma^*) &= 0 \\ \nabla_\gamma L(p^*, \gamma^*) &= 0. \end{aligned} \tag{24}$$

Theorem 3.2.1. If and only if (p^*, γ^*) is an equilibrium point of neural network model (22), (p^*, γ^*) is a saddle point of the Lagrangian function Eq (21).

Proof: If (p^*, γ^*) is a saddle point of the Lagrangian function Eq (21),Then according to Lemma 3.1.1, p^* is a local minimum point of nonlinear programming problem Eq (21). Therefore, when (p^*, γ^*) is a saddle point of the Lagrangian function Eq (21), (p^*, γ^*) is an equilibrium point of neural network model (22). If (p^*, γ^*) is an equilibrium point of neural network model (22), according to the analysis of the state equation of neural network model (22), the equilibrium point (p^*, γ^*) is a saddle point of the Lagrangian function. The theorem is thus proved.

Theorem 3.2.2. If (p^*, γ^*) is an equilibrium point of neural network model (22), the neural network model (22) is stable at point (p^*, γ^*).

Proof: The energy function that constructs neural network model (22) is

$$E(P, \gamma) = \frac{1}{2}\left\|\nabla_p L(P, \gamma)\right\|^2 + \frac{1}{2}\left\|\nabla_\gamma L(P, \gamma)\right\|^2 \tag{25}$$

The differential of the energy function with respect to time t is

$$\begin{aligned}\frac{dE}{dt} &= \frac{\partial E}{\partial p} * \frac{dp}{dt} + \frac{\partial L}{\partial \gamma} * \frac{d\gamma}{dt} \\ &= [\nabla_p L(p,\gamma)\nabla^2_{pp} L(p,\gamma) + \nabla_\gamma L(p,\gamma)\nabla^2_{p\gamma} L(p,\gamma)]\frac{dp}{dt} \\ &+[\nabla_p L(p,\gamma)\nabla^2_{p\gamma} L(p,\gamma) + \nabla_\gamma L(p,\gamma)\nabla^2_{\gamma\gamma} L(p,\gamma)]\frac{d\gamma}{dt} \\ &= -\nabla_p L(p,\gamma)\nabla^2_{pp} L(p,\gamma)\nabla_p L(p,\gamma).\end{aligned} \tag{26}$$

The following conclusion can be drawn from the stability theory of Lyapunov:

1) If $\nabla^2_{pp} L(p,\gamma)$ is positive definite, the differential of the energy function with respect to t is $\frac{dE}{dt} < 0$. Therefore, the neural network model is asymptotically stable at point (p^*, γ^*).

2) If $\nabla^2_{pp} L(p,\gamma)$ is positive semi-definite, the differential of the energy function with respect to t is $\frac{dE}{dt} \le 0$. Therefore, the neural network model is stable at point (p^*, γ^*).

Since (p^*, γ^*) is an equilibrium point of neural network model (22) and satisfies $\nabla_p L(p,\gamma) = 0$, the differential of the energy function with respect to t at point (p^*, γ^*) is $\frac{dE}{dt} = 0$. Therefore, the conclusion above shows that the neural network is stable at point (p^*, γ^*). That is, the algorithm converges at this point.

The quadratic term introduced $\sum_k \beta_k [C_k(p)]^2$ does not affect the position of the saddle point because of $[C_k(p)] = 0$, and the quadratic term plays a role in stabilizing the system [29]. The penalty term is necessary for resisting the oscillations of the standard Lagrangian method and can improve the convergence performance of numerical calculation.

3.3. Process of accelerate augmenting the Lagrangian neural network algorithm

In the augmented Lagrangian multiplier method, the label assignment object changed from p to u; and neural network model (22) was replaced with

$$\frac{du_i(I)}{dt} = -\frac{\partial L_\beta(p(u),\gamma)}{\partial p_i(I)}\frac{\partial p_i(I)}{\partial u_i(I)} \tag{27}$$

It is noticed that

$$\frac{\partial p_i(I)}{\partial u_i(I)} = \frac{e^{-u_i(I)/T}}{T(1+e^{-u_i(I)/T})^2} > 0 \tag{28}$$

Hence u was updated with

$$\frac{du_i(I)}{dt} = -\frac{\partial L_\beta(p(u), \gamma)}{\partial p_i(I)}. \tag{29}$$

In the TSP, the energy gradient in neural network model (22) is:

$$\frac{\partial E}{\partial p_i(I)} = \sum_{J \neq I} d_{IJ}[p_{i+1}(J) + p_{i-1}(J)] \tag{30}$$

There are (n=) K Lagrangian multipliers γ in the second term of neural network model (22); and the value of K is given in Eq (12). There are three different types of constant β_k, which adapt to constraints Eqs (7) to (11) [30]. The neurons corresponding to travel paths are composed of m^2 $p_i(I)$ neurons. These neurons are associated with internal variable u through Sigmiod Function. Meanwhile,an additional number of $5m^2 + m$ γ neurons for the Lagrange multipliers.

The tradition ALHNN algorithm realizes the kinetic processes described in Eqs (14), (22) and (30) in the following order as a whole:

$$u^{(t+1)}(I) \leftarrow u^{(t)}(I) - \mu\left\{q_i^{(t)}(I) + \sum_k \gamma_k^{(t)} \frac{\partial C_i(p)}{\partial p_i(I)} + \sum_k \beta_k C_k(p^{(t)}) \frac{\partial C_i(p)}{\partial p_i^{(t)}(I)}\right\} \tag{31}$$

$$p_i^{(t+1)}(I) \leftarrow \psi_T(u_i^{(t+1)}(I)) \tag{32}$$

$$\gamma_k^{(t+1)} \leftarrow \gamma_k^{(t)} + \beta_k C_k(p^{(t+1)}). \tag{33}$$

Where, μ is the step length factor. During the update, T may decrease and β_k may increase to increase the convergence rate. All i and I were synchronously updated. The ALHNN algorithm is better than the genetic algorithm (GA) because it does not need the normalized operation required by the GA, which is more convenient for conducting a simulation experiment [29]. The convergence rate of ALHNN is $O(1 / k)$.

In the experiment, we found that the iterative speed of ALHNN is too slow. This is because although ALHNN is globally convergent. Therefore,Using nesterov acceleration technique, we can obtain the following AALM (acceleration of augmented Lagrangian method) iterations.

$$\theta_k^{(t)} = \gamma_k^{(t)} + \lambda C_k(p^{(t+1)}) \tag{34}$$

$$\gamma_k^{(t+1)} = \theta_k^{(t)} + \frac{t-1}{t+2}(\theta_k^{(t)} - \theta_k^{(t-1)}) + \frac{t+1}{t+2}(\theta_k^{(t)} - \gamma_k^{(t)}). \tag{35}$$

Among them,

$$\gamma_k^{(1)} = \theta_k^{(0)} \tag{36}$$

The convergence rate of AALHNN by nesterov acceleration technique is $O(1 / k^2)$ [31], Compared with the traditional ALHNN, it improves ALHNN's convergence speed.

Remarks: In the above proof process, the dynamic model is required to be a convex function [31], which has some good properties:

1) The feasible solution set is a convex set;

2) Any local optimal value of the objective function in the definition domain must be the global optimal solution;

3) The set of optimal solutions is convex (when the optimal solution exists);

if the programming problem to be solved is a nonconvex function, AALHNN probably not suitable.

The model in this paper is Minimized Energy Function $E(p)$. If Maximized Function $F(p)$ needs to be solved in some cases, simply $E(p) = -F(p)$, then the same renewal equation is used.

With the output to TSP written below, and constraints Eqs (7) to (11) as the constraints on the problem, the following augmented Lagrangian algorithm process could be obtained:

$$u_i^{t+1}(I) \leftarrow u_i^t(I) - \mu\left\{D\sum_{J\neq I} d_{IJ}[p_{i+1}^{(t)}(J) + p_{i-1}^{(t)}(J)] + \frac{\partial E_L}{\partial p_i^{(t)}(I)} + \lambda\frac{1}{2}\frac{\partial E_p}{\partial p_i^{(t)}(I)}\right\} \tag{37}$$

$$p_{i+1}^{(t)}(I) \leftarrow \psi_T(u_i^{(t+1)}(I)) \tag{38}$$

$$\begin{aligned}\theta_k^{(t)} &= \gamma_k^{(t)} + \lambda C_k(p^{(t+1)})\\ \gamma_k^{(t+1)} &= \theta_k^{(t)} + \frac{t-1}{t+2}(\theta_k^{(t)} - \theta_k^{(t-1)}) + \frac{t+1}{t+2}(\theta_k^{(t)} - \gamma_k^{(t)}).\end{aligned} \tag{39}$$

In Eqs (37) to (39) above, μ is the step length constant; D is the weighting constant; T is the temperature constant; λ is a non-decreasing factor of penalty term; and the two partial derivatives in neural network model (22) are

$$\frac{\partial E_L}{\partial p_i(I)} = \gamma^{I_1^{(t)}} + \gamma_2^{I^{(t)}} + \gamma^{i,I_3^{(t)}}\sum_{j\neq i} p_j(I) + \gamma_4^{i,I^{(t)}}\sum_{J\neq I} p_j(J) + \gamma_5^{i,I^{(t)}}(0.5 - p_i(I)) \tag{40}$$

$$\begin{aligned}\frac{1}{2}\frac{\partial E_p}{\partial p_i(I)} &= \beta_1(\sum_I p_i(I) - 1) + \beta_2(\sum_i p_i(I) - 1) + p_i(I) * (\sum_{j\neq i} p_j(I))^2\\ &+ p_i(I) * (\sum_{J\neq I} p_i(J))^2 + p_i(I) * (1 - p_i(I)) * (0.5 - p_i(I))\end{aligned} \tag{41}$$

Equation (40) is the partial derivative of Lagrangian multipliers; and Eq (41) is the partial derivative of penalty terms. Here constant β_k is uncorrelated to i or I.

The system of dynamic equation with respect to variable u in [4] is

$$u_i^{(t+1)}(I) \leftarrow u_i^{(t)}(I) - \mu\left\{A(\sum_I p_i^{(t)}(I) - 1) + B(\sum_i p_i^{(t)}(I) - 1) + C(0.5 - p_i^{(t)}(I)) + D\sum_{J\neq I} d_{IJ}\left[p_{i+1}^{(t)}(J) + p_{i-1}^{(t)}(J)\right]\right\}. \tag{42}$$

The $p - u$ relation is

$$p_i^{(t+1)}(I) \leftarrow \frac{1}{1 + e^{-[u^{(t+1)}(I)-0.5]/T}}. \tag{43}$$

$u_i^{(t+1)}(I)$ in Eq (43) shifted by 0.5. In this algorithm, i and j were synchronously updated.

4 Simulation experiment

4.1 Setting the experimental process

Experimental parameters were set as follows:

$$\beta_1 = \beta_2 = 100, \beta_3 = \beta_4 = 1, \beta_5 = 0, D = 10^6, T = 10^7, \mu = 100. \tag{44}$$

Since paths were closed, the city visited first was set as $I = 1$. Updated the algorithm parameters when $i, I = 2, \mathrm{L}\ , m$. During each operation, the initial value of variable p was that recommended by Brandt as below.

$$p_i^{(0)}(I) = 0.5 + \delta \quad i, I = 2, \mathrm{L}\ , m \tag{45}$$

δ was a value randomly taken from the uniform distribution $[-10^6, 10^6]$; and variable $u^{(0)}$ was determined by the respective $u - p$ relation [31]. Finally, in the augmented Lagrangian multiplier, the Lagrangian multiplier $\gamma_k^{(0)}$ was set to 0.

If the following three conditions are met, the iterative process is deemed convergent. Here it was set to solve the termination condition.

$$\left\|p^{(t)} - p^{(t-1)}\right\|_\infty < 0.01 / m \tag{46}$$

$$\left|\sum_I p_i(I) - 1\right| < 0.01 \qquad \left|\sum_i p_i(I) - 1\right| < 0.01 \tag{47}$$

$$1 - p_i(I) < 0.01 \quad or \quad p_i(I) < 0.01. \tag{48}$$

Equation (46) ensures that the final state is stable, Eq (47) ensures the feasibility of the process; and Eq (48) indicates that the process is definite and that the value of p is 0 or 1. The experiments that cannot meet these conditions during the experiment iteration are deemed invalid.

4.2 Solving the TSP by the simulation experiment

In order to verify the actual application effect of the proposed AALHNN, the effect of the AALHNN was compared with that of the traditional Hopfield neural network (HNN) on the same TSP by pathfinding simulation, with a group of random paths set as a reference for the two algorithms. The number of cities was set to 8, 10,12 and 15, respectively. The coordinates for solving the TSP for 15 cities are listed in Table 1 and the TSP of the number of other cities takes the coordinates of the previous corresponding number in the Table 1. Such data can be easily modified in the running code when solving practical problems.

Table 1. Coordinate of 15 citys.

Number	X	Y
1	0.4000	0.4439
2	0.2439	0.1463
3	0.1707	0.2293
4	0.2293	0.7610
5	0.5171	0.9414
6	0.8732	0.6536
7	0.6878	0.5219
8	0.8488	0.3609
9	0.6683	0.2536
10	0.6195	0.2634
11	0.4125	0.4941
12	0.8251	0.7407
13	0.2551	0.8300
14	0.9646	0.9132
15	0.8884	0.5944

The experimental environment is Intel(R) Core(TM) i7-7500U CPU, 8 GB memory, Geforce 940 MX, Windows 10 operating system, The simulation code of the AALHNN in this paper was solved by matlab 2016 b, while the codes of the HNN and random paths were compiled and solved by Phthon 3.7. Since there is a difference in the solving efficiency between the two languages, the solving time was not added as a performance comparison indicator for the two algorithms. The final pathfinding compare results are listed in Table 2. The experimental data settings are provided in Section 4.1.

Figure 3 shows that the HNN may generate a trivial solution due to its limitations; and according to the statistics by multiple experiments, the solution success rate is about 80% when the city number of TSP is 10 and will decreases to about 71% when the city number of TSP is 15. And under the same circumstances AALHNN can always solve TSP as long as the number of cities of TSP is reasonable. Therefore, experiments show that the AALHNN can jump out of the local optimal solution for the same problem and get a better solution result.

Figures 4 to 7 show the solution path diagrams of the three algorithms when the city number of TSP is 8, 10, 12 and 15, respectively. It can be seen from these figures that when the number of TSP cities is certain, the AALHNN solution result is better than HNN, and the two algorithms have great path length optimization compared with random paths, which means that AALHNN has certain application effectiveness and novelty.

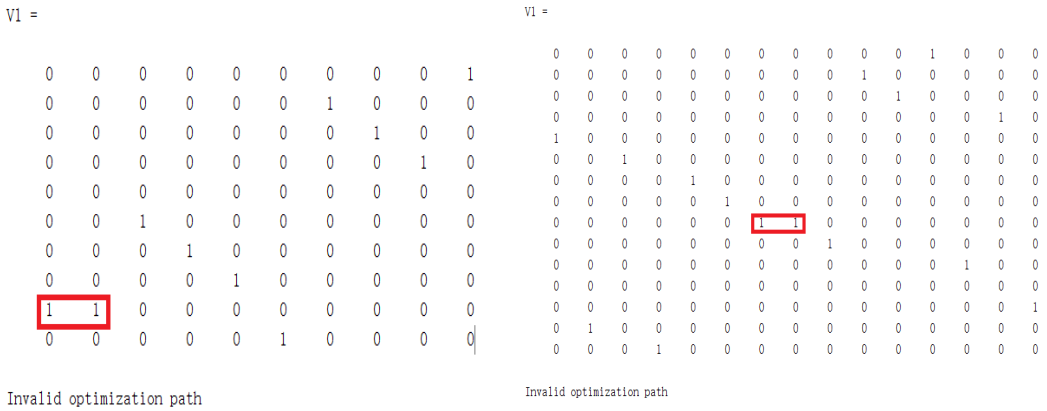


Figure 3. HNN Invalid optimization path for 10 citys and 15 citys.

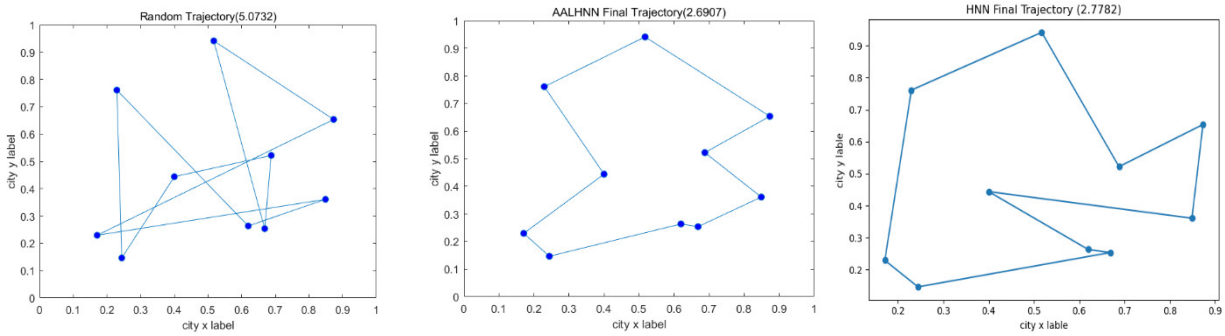


Figure 4. Comparison of TSP paths obtained by three methods in 8 cities.

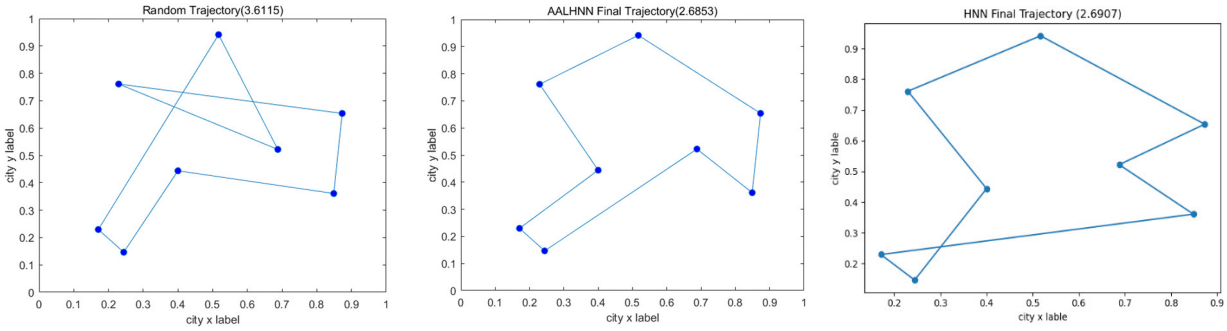


Figure 5. Comparison of TSP paths obtained by three methods in 10 cities.

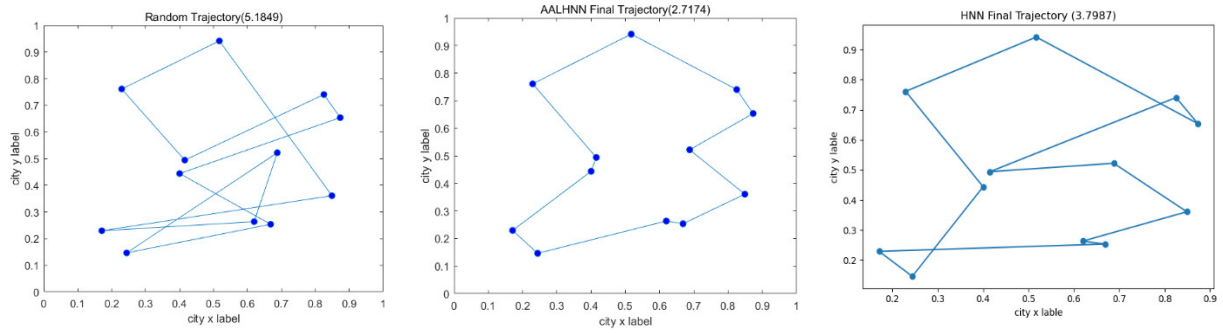


Figure 6. Comparison of TSP paths obtained by three methods in 12 cities.

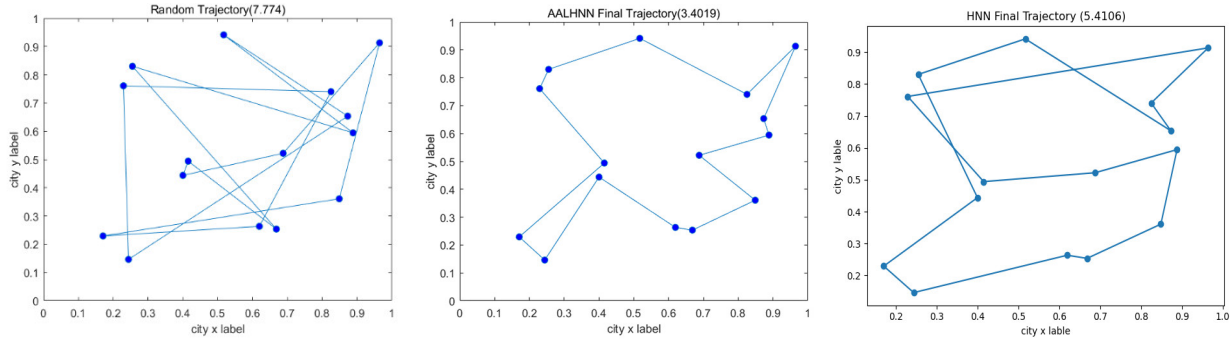


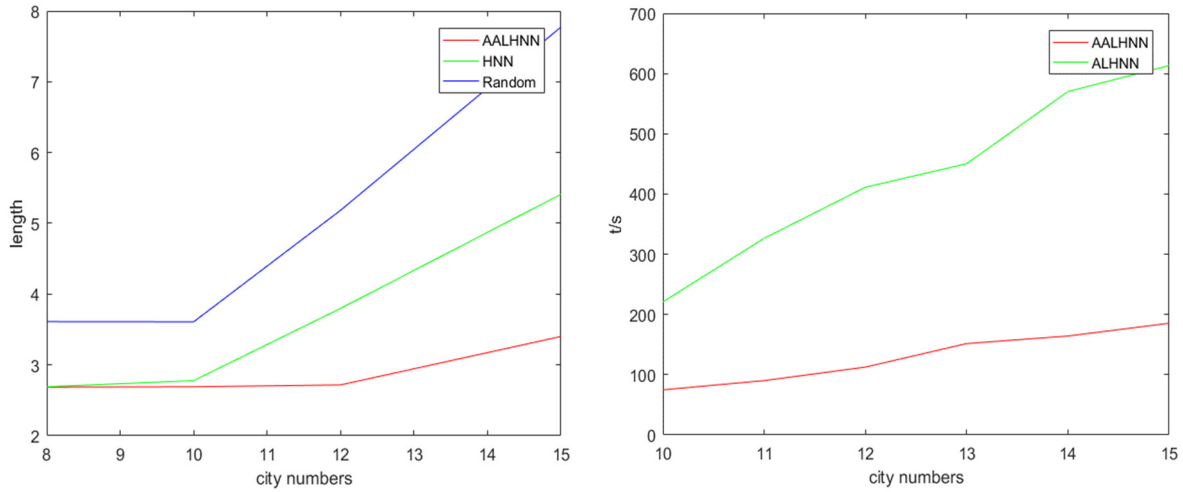
Figure 7. Comparison of TSP paths obtained by three methods in 15 cities.

Table 2. Comparison of TSP solutions of three methods.

City number / method	8	10	12	15
AALHNN	2.6853	2.6907	2.7174	3.4019
HNN	2.6907	2.7782	3.7987	5.4106
Random Path	3.6115	3.6088	5.1849	7.7740

It can be seen from Table 2 that the results obtained by AALHNN change little when the number of TSP cities increases. However, the results obtained by HNN begin to deform when the number of cities increases to 12, and the TSP path length increases greatly at this time. This reflects the robustness of AALHNN. Of course, from Figures 3 to 7, it can be seen that the results of the two neural networks are more optimized than the random path.

From Figure 8(a) it shows the comparison of the effective solution path length obtained by the three methods for TSP with different number of cities. It can be seen from the figure that the result obtained by AALHNN is better than that obtained by HNN, and from the simulation experiment, the success rate of AALHNN is 100%, which is greatly optimized compared with HNN. It is worth noting that the path length obtained by AALHNN is not optimized compared with ALHNN This is because the acceleration technique in this paper only improves the convergence rate of ALHNN, which can be seen from Table 3.



(a) (b)

Figure 8. (a) The results of solving TSP by three methods; (b) Time comparison of ALHNN and AALHNN.

Table 3. Comparison of performance for the 15-city problem.

	Rate of solving feasible solution	Convergence speed of algorithm
HNN	71.43%	fast
ALHNN	100.00%	slow
AALHNN	100.00%	fast

Figure 8(b) shows that the method of updating Lagrange factor by using the acceleration technique of Eq (39) can greatly improve the convergence speed (repeat the experiment 50 times for each coordinate and take the average value) of ALHNN. Table 3 shows that AALHNN improves the convergence rate compared with ALHNN, while AALHNN and ALHNN greatly improve the probability of obtaining feasible solutions compared with HNN.

4.3 Result analysis

Tables 2 and 3 show that the length of random TSP paths was greatly optimized by the HNN. However, according to Figure 3, the HNN has a probability of not finding the efficient solution. Moreover, the AALHNN and ALHNN ensures better results when solving the TSP than the HNN, and greatly improves the solution success rate. It can be noticed that the solution success rate by the HNN decreases as the number of cities increases. The reason is that with the increase in the number of cities in the TSP, there's more probability of obtaining a trivial solution as the corresponding HNN equation gets stuck in the local minimum. It can also be noted that AALHNN greatly improves the convergence rate of ALHNN.The following conclusion is made based on the above simulation experiment and its analysis: the proposed AALHNN has better solution lengths and success rates than the HNN,and also has better solution convergence rate than ALHNN.

5 Conclusions

Based on the basic HNN, an accelerated augmented Lagrangian method was proposed in this paper to solve the TSP. The innovation of this paper is from the existing literature, the paper found that the HNN may not always generate an efficient solution when solving the TSP (i.e., the shortcoming of the penalty method), but the AALHNN can overcome the shortcoming. The AALHNN introduces Lagrangian multipliers to solve the function and introduces quadratic terms to augment the Lagrangian function, providing greater convergence stability, And an iterative method based on nesterov acceleration technique is proposed to update the Lagrange factor. In this paper, the convergence of the AALHNN was theoretically proved. Besides, the performances and success rates in solving the TSP were compared between the HNN and AALHNN by a simulated program. It is concluded that the final solution length is shorter than HNN, and the solution success rate is greatly improved. It was found in experiments that the efficiency of solving the TSP by the AALHNN decreases with the increase in the number of cities. The reason is that when the number of cities increases, the equations with Lagrangian multipliers become more complex; the algorithm iteration becomes more difficult; and the amount of calculation also soars. As a result, the efficiency of obtaining efficient solutions decreases. Therefore, the next research direction should be expanding the application scope of the AALHNN by simplifying the equations.

Acknowledgments

This work is supported by the national key research and development program (2019YFB1705702).

Conflict of interest

The authors declared that they have no conflicts of interest to this work.

References

1. J. J. Hopfield, D. W. Tank, "Neural" computation of decisions in optimization problems, *Biol. Cybern.*, **52** (1985), 141–152. https://doi.org/10.1007/BF00339943
2. J. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, *Proc. Natl. Acad. Sci.*, **81** (1984), 3088–3092. https://doi.org/10.1073/pnas.81.10.3088
3. L. Zhang, Y. Zhu, W. X. Zheng, State estimation of discrete-time switched neural networks with multiple communication channels, *IEEE Trans. Cybern.*, **47** (2017), 1028–1040. https://doi.org/10.1109/TCYB.2016.2536748
4. G. V. Wilson, G. S. Pawley, On the stability of the travelling salesman problem algorithm of Hopfield and Tank, *Biol. Cybern.*, **58** (1988), 63–70. https://doi.org/10.1007/BF00363956
5. Brandt, W. Yao, Laub, Mitra, Alternative networks for solving the traveling salesman problem and the list-matching problem, in *IEEE 1988 International Conference on Neural Networks*, (1988), 333–340. https://doi.org/10.1109/ICNN.1988.23945

6. M. Waqas, A. A. Bhatti, Optimization of n + 1 queens problem using discrete neural network, *Neural Network World*, **27** (2017), 295–308. http://dx.doi.org/10.14311/NNW.2017.27.016
7. L. García, P. M. Talaván, J. Yáez, The 2-opt behavior of the Hopfield Network applied to the TSP, *Oper. Res.*, 2020. https://doi.org/10.1007/s12351-020-00585-3
8. I. Valova, C. Harris, T. Mai, N. Gueorguieva, Optimization of convolutional neural networks for imbalanced set classification, *Procedia Comput. Sci.*, **176** (2020), 660–669. https://doi.org/10.1016/j.procs.2020.09.038
9. S. Z. Li, Relaxation labeling using Lagrange-Hopfield method, in *IEEE International Conference on Image Processing*, **1** (1995), 266–269. https://doi.org/10.1109/ICIP.1995.529697
10. D. Kaznachey, A. Jagota, S. Das, Primal-target neural net heuristics for the hypergraph k-coloring problem, in *Proceedings of International Conference on Neural Networks*, **2** (1997), 1251–1255. https://doi.org/10.1109/ICNN.1997.616213
11. Z. Wu, Jiang, B. Jiang, H. R. Karimi, A logarithmic descent direction algorithm for the quadratic knapsack problem, *Appl. Math. Comput.*, **369** (2020), 124854. https://doi.org/10.1016/j.amc.2019.124854
12. P. V. Yekta, F. J. Honar, M. N. Fesharaki, Modelling of hysteresis loop and magnetic behaviour of fe-48ni alloys using artificial neural network coupled with genetic algorithm, *Comput. Mater. Sci.*, **159** (2019), 349–356. https://doi.org/10.1016/j.commatsci.2018.12.025
13. A. Rachmad, E. M. S. Rochman, D. Kuswanto, I. Santosa, R. K. Hapsari, T. Indriyani, Comparison of the traveling salesman problem analysis using neural network method, in *Proceedings of the International Conference on Science and Technology (ICST)*, 2018. https://doi.org/10.2991/icst-18.2018.213
14. A. Mazeev, A. Semenov, A. Simonov, A distributed parallel algorithm for the minimum spanning tree problem, in *International Conference on Parallel Computational Technologies*, **753** (2017), 101–113. https://doi.org/10.1007/978-3-319-67035-5_8
15. S. Z. Li, Improving convergence and solution quality of hopfield-type neural networks with augmented lagrange multipliers, *IEEE Trans. Neural Networks*, **7** (1996). 1507–1516. https://doi.org/10.1109/72.548179
16. M. Honari-Latifpour, M. A. Miri, Optical potts machine through networks ofthree-photon down-conversion oscillators, *Nanophotonics*, **9** (2020), 4199–4205. https://doi.org/10.1515/nanoph-2020-0256
17. S. Zhang, A. G. Constantinides, Lagrange programming neural networks, *IEEE Trans. Circuits Syst.*, **39** (1992), 441–452. https://doi.org/10.1109/82.160169
18. S. S. Kia, An Augmented Lagrangian distributed algorithm for an in-network optimal resource allocation problem, in *2017 American Control Conference (ACC)*, (2017), 3312–3317. https://doi.org/10.23919/ACC.2017.7963458
19. Y. Hu, Z. Zhang, Y. Yao, X. Huyan, X. Zhou, W. S. Lee, A bidirectional graph neural network for traveling salesman problems on arbitrary symmetric graphs, *Eng. Appl. Artif. Intell.*, **97** (2021), 104061. https://doi.org/10.1016/j.engappai.2020.104061
20. U. Wen, K. M. Lan, H. S. Shih, A review of hopfield neural networks for solving mathematical programming problems, *Eur. J. Oper. Res.*, **198** (2009). 675–687. https://doi.org/10.1016/j.ejor.2008.11.002
21. S. Sangalli, E. Erdil, A. Hoetker, O. Donati, E. Konukoglu, Constrained optimization to train neural networks on critical and under-represented classes, preprint, arXiv:2102.12894v4.

22. B. Aylaj, M. Belkasmi, H. Zouaki, A. Berkani, Degeneration simulated annealing algorithm for combinatorial optimization problems, in *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, (2015), 557–562. https://doi.org/10.1109/ISDA.2015.7489177
23. M. Zarco, T. Froese, Self-modeling in hopfield neural networks with continuous activation function, *Procedia Comput. Sci.*, **123** (2018), 573–578. https://doi.org/10.1016/j.procs.2018.01.087
24. Y. J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, O. Büyüköztürk, Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types, *Comput.-Aided Civ. Infrastruct. Eng.*, **33** (2017), 731–747. https://doi.org/10.1111/mice.12334
25. H. Uzawa, K. Arrow, L. Hurwicz, Studies in linear and nonlinear programming, (1958), 154–165.
26. A. Barra, M. Beccaria, A. Fachechi, A relativistic extension of hopfield neural networks via the mechanical analogy, preprient, arXiv:1801.01743.
27. V. N. Dieu, W. Ongsakul, J. Polprasert, The augmented lagrange hopfield network for economic dispatch with multiple fuel options, *Math. Comput. Modell.*, **57** (2013), 30–39. https://doi.org/10.1016/j.mcm.2011.03.041
28. Y. Arjevani, J. Bruna, B. Can, M. Gürbüzbalaban, S. Jegelka, H. Lin, Ideal: inexact decentralized accelerated augmented lagrangian method, preprient, arXiv:2006.06733.
29. C. Dang, L. Xu, A lagrange multiplier and hopfield-type barrier function method for the traveling salesman problem, *Neural Comput.*, **14** (2002), 303–324. https://doi.org/10.1162/08997660252741130
30. Z. Wu, Q. Gao, B. Jiang, H. R. Karimi, Solving the production transportation problem via a deterministic annealing neural network method, *Appl. Math. Comput.*, **411** (2021), 126518. https://doi.org/10.1016/j.amc.2021.126518
31. M. Kang, S. Yun, H. Woo, M. Kang, Accelerated bregman method for linearly constrained ℓ_1-ℓ_2 minimization, *J. Sci. Comput.*, **56** (2013), 515–534. https://doi.org/10.1007/s10915-013-9686-z

AIMS Press