Solving Covering salesman problem using modified Hopfield network

Wenkang Wei
Department of Electrical and
Computer Engineering
Clemson University
Clemson, South Carolina,US

***Abstract*—The primary purpose of this paper is to propose a new modified Hopfield neural network to solve the Covering salesman problem (CSP), which is the generalization of traveling salesman problem (TSP). Covering salesman problem can be described as follows: identify the minimum cost tour of a subset of n given cities such that every city not on the tour is within some predetermined covering distance standard, S, of a city that is on the tour.**

There is a lack of attempts to solve the generalization of TSP by using a more general energy-based method, though Traveling salesman problem and Hopfield network have been studied well for decades. This paper provides a more general version of Hopfield network for solving CSP. The neural network successfully generalizes TSP to CSP and solves CSP problem with nearly 100% rate of finding optimal solutions for all city set we test with. The performances of our method on different city sets are also presented and analyzed in this paper.

***Index Terms*—artificial neural network, Optimization, machine learning, Covering Salesman Problem, Traveling salesman problem, Hopfield network, path planning**

I. Introduction

Traveling salesman problem The traveling salesman problem is a NP-hard problem of finding the shortest path that visits every city in a set of cities once and returns to the first. It is a very well studied problem for decades [23]. There have been many different TSP solutions, such as energy method multi-agent method, genetic algorithm and graph-based solution [10–14].

Though TSP has been studied for decades and many solutions are proposed, many variations of TSP are proposed in recent years, attracting researchers from different fields to study and apply it to real world problem, such as online traveling salesman [22], Covering salesman problem [15], Covering salesman problem with nodes and segments [20], online CSP [18].

Covering salesman problem Covering salesman problem (CSP) is a generalization of traveling salesman problem (TSP). It is stated as follows: to identify the minimum cost tour of a subset of n given cities such that every city not on the tour is within some predetermined covering distance standard, S, of a city that is on the tour [15]. Since CSP is a general version of TSP, solving CSP can leads to solutions to TSP as well as to other similar problems. There is a wide range of applications of CSP. For example, in Disk Covering Tour problem [7], solving CSP can help reduce energy consumption for mobile robots. Moreover, it can contribute to the traffic network system and other scheduling, resource reallocation problems [8], [9].

Hopfield Network Hopfield network is widely used in numerous different optimization problems since the early 1980s. One of the constraint combinatorial optimization problem is TSP. Hopfield and Tank first applied the network to solve TSP successfully [14]. In recent years, Hopfield network is extended to solve more complex problems [1, 5]. In the paper by Z. Uykan [5], it proposes the hopfield network with double sigmoid functions and also analyzes the fast convergence property. In addition, Z. Uykan also extends hopfield network to deal with complex number cases [1]. However, despite that TSP and Hopfield network have been studied well for decades and that various versions of TSP and Hopfield networks are proposed, there is a lack of attempts to generalize Hopfield network for solving the derivatives of TSP.

Contribution Motivated by the extension of Hopfield network and variations of TSP and CSP, this paper aims at contributing to find a general solution addressing CSP. The contributions of this paper are as follows: 1) We re-formulate the Covering Salesman Problem, giving a new insight of generalization of TSP. 2) We propose a more general energy-method, extended Hopfield network, for solving both TSP and CSP successfully, generalizing the solutions of such kind of planning problems.

In this paper, we first formulate the covering salesman problem in section II. Later we introduce the Hopfield-Tank type neural network introduced [16] in section III. After introducing Hopfield-Tank network, we introduce how our modified Hopfield network works in section IV. In section IV, we formulate CSP and show how to apply Hopfield network on it. Then we then introduce the modification of energy function E_3 and additional energy function E_5 in Hopfield network. Based on the modifications, we present the new forms of weight, bias and the overall algorithm procedure. The experiment section discusses how our method generalizes TSP to CSP and how the new parameters we design matter. Finally, this paper ends with the future work and conclusion.

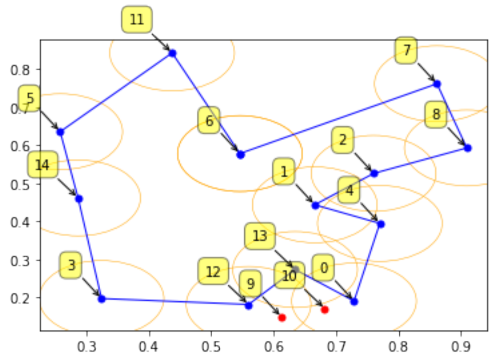

Fig. 1. example pattern with 15 cities with indices range from 0 to 14. Each point represent a city. Red points represent cities covered by other cities on the tour and blue points represent cities on the tour. The yellow circles are the covering area.

II. Problem Formulation

Assume that there is a set of n cities with indices V={0,2,...n-1}, starting from 0. Distance between city i and city j, is denoted as $d_{i,j}$. If i= j, then $d_{i,i} = 0$. The covering distance of the i^{th} city is r_i and $r_i \geq 0$. If distance between city i and j, $d_{i,j}$, is less than covering distance r_i, then city j is covered by city i. Denote that the set of cities that are on the tour is T= $\{a_1$,a_2,...$a_m\}$, where a_m is the index of a city on tour from set V and a set of cities covered by other cities on the tour S = $\{b_1$,b_2,...$b_k\}$ such that $T \cup S$ =V and T should not be an empty set.

Covering salesman problem requires: (1) the salesman starts from the city i and goes back to the city i, (2) the salesman should visit each city at most once, (3) all cities, not on the tour, are within radius r of any visited city on the tour. In other words, the covering salesman problem is to select the cities from set S to set A such that

$$min \sum_{i=1..m-1} d_{a_i,a_{i+1}} + d_{a_m,a_1} \tag{1}$$

where a_i is the index of the city in set T. $|T|$ is the size of set T. City a_i and city a_{i+1} are connected directly in the tour. Since CSP requires the salesman starts from city i and goes back to city i, distance d_{a_m,a_1} is added to represent that salesman goes back to the initial city.

In figure 1, there are 15 cities. The blue line is the shortest valid tour in CSP. Since City 9 and City 10 are in the yellow circles of city 12 and city 0 respectively, they are covered by city 12 and city 0 respectively and hence not on the tour. In addition, if the covering distance r_i of all cities are equal to 0, then CSP will become TSP problem, since it requires each city must be on the tour.

III. Hopfield-Tank neural network

Hopfield network is a type of neural network aiming at finding the output that satisfies the constraints and minimizes the energy functions. When the output from Hopfield network minimizes the energy functions, then it satisfies the constraints in the problem.

In Hopfield-Tank neural network model [16] with n fully-interconnected computation units, or called neurons, the weight connecting i^{th} neuron and the j^{th} neuron is denoted as $W_{i,j}$. Each neuron has two state variables: internal state,or called input potential,u and external state, or called output potential v. Here,both u and v are state vectors and $u, v \in \mathcal{R}^n$. If external state v_i is continuous value, then the relationship between these two states are as follows:

$$v_i = f(u_i) = 1/2[1 + tanh(\frac{u_i}{\mu})] \tag{2}$$

where v_i and u_i are the i^{th} scalar components in vectors. Function $f()$ is sigmoid activation function [5] and μ is a parameter controlling the slope of this function. μ is gain value. If μ is significantly small, v_i can be approximately binary value. When external state v_i is required to be binary decision value, then the relationship between u_i and v_i is

$$v_i = \begin{cases} 1 \text{ , if } f(u_i) \geq \lambda \text{ threshold} \\ \quad 0 \text{ , otherwise} \end{cases} \tag{3}$$

The threshold λ is usually default as 0.5 when activation function is sigmoid function. In continuous Hopfield network, unit dynamic modeled by the first order differential equation [19] is given by

$$\dot{u}(t) = \frac{du}{dt} = -\frac{1}{\tau}u(t) + Wv(t) + i_b \tag{4}$$

For practical computer realization, when applying Euler method and equation (4), the dynamic of input potential becomes :

$$u(t + \triangle t) = u(t) + \triangle t\frac{du}{dt} \tag{5}$$

$\triangle t$ is the change of time. The term $Wv(t) + i_b$ from dynamic is obtained by the first order derivative of energy function subject to minimize. The energy function has a general form:

$$E = -(1/2)\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}v_iv_j - \sum_{i=1}^{n} i_bv_i \tag{6}$$

Or equivalently,

$$E = -(1/2)v^TWv - i_b^Tv \tag{7}$$

where v and u are vectors $\in \mathcal{R}^n$, W is the weight matrix $\in \mathcal{R}^{n\times n}$ and $W_{i,j}$ is the i^{th} row, j^{th} column entry of W. $i_b \in \mathcal{R}^n$ is the bias vector. We can observe that when $\tau = 1$ and $\triangle t$=1 in equation (4), (5), we have

$$u = -\frac{dE}{dv} = Wv + i_b \tag{8}$$

This also leads to the update rule of input potential u in discrete case:

$$u(t + 1) = Wv(t) + i_b \tag{9}$$

Overall, after updating the input potential $u(t)$ by equations (4) and (5), we can compute the output potential $v(t)$ by equation (2). As the internal state is updated by the first-order derivative of the energy function, the update of internal state is converging to the local minima of energy function [16].

IV. Modified Hopfield network in CSP

A. Formulation

In order to apply Hopfield network to CSP, consider the following formulation. Consider n cities and hence n positions corresponding to each city in tour. Then we have a set of cities $V = \{1, 2, ...n\}$, a set of cities on the tour $T = a_1, a_2, ...a_m$ and a set of cities not on the tour, covered by cities on the tour $S = b_1, b_2, ...b_k$, such that $T \cup S = V$. Then there are $n \times n$ neurons and each neuron has external state $o_{x,i}$. Then the vector of all external state is denoted as o. In addition, since neurons in the network are fully connected, then weight matrix $W \in \mathcal{R}^{n^2 \times n^2}$, has n^4 components. Sizes of set T and S are $|T| = m$ and $|S| = k$, respectively, where m + k =n. Then the quadratic formulation of the n-city CSP, gives the binary decision state, or external state:

$$o_{x,i} = \begin{cases} 1 \text{ , if city x is in position i} \\ 0 \text{ , otherwise} \end{cases} \tag{10}$$

Different from the notation v_i in the last section, the new notation of external state $o_{x,i}$ here has its physical meaning. if $o_{x,i}$=1, it means city x is in the i^{th} position on the tour. Otherwise, it means city x may be either covered by other cities, or in another position rather than the i^{th} position on the tour [19]. The internal state is denoted as $u_{x,i}$ corresponding to external state $o_{x,i}$. There is a distance matrix D with component $d_{i,j}$, representing distance between city i and city j.

Example 1: This example illustrates the matrix representation of the path solution for CSP. Assume that there are 4 cities, A, B, C, D and each city has corresponding 4 position options in the table. In addition, distance $d_{B,C} <$ covering distance r. Hence city B and city C cover each other. Each entry in the table, matrix represents an external state $o_{x,i}$. Hence entry (A, 1) has external state $o_{A,i}$=1, meaning that city A is the first city to be visited. This Hopfield network has n×n = 16 neurons and 16 × 16=256 components in weight matrix. In this solution, city B is on the tour and city C is not on the tour, but covered by city B, hence entries in the row of city C are all zero. City D is the last city to be visited since $o_{D,4}$=1. Actually, if $o_{D,3} = 1$ and $o_{D,4}$=0, it still represents the same solution because the final solution is obtained by removing the zero-rows and zero-columns in the matrix and then sorting cities along the step axis in table. Hence, the same solution can be represented in different matrix representations for CSP.

TABLE I
Matrix representation of path in CSP.

	step visited in solution			
city: n=4	1	2	3	4
A	1	0	0	0
B	0	1	0	0
C	0	0	0	0
D	0	0	0	1

Assume all cities have the same covering distance r_i = $r \geq 0$. Then the covering salesman problem can be formulated as follows:

Let $E_1 = \sum_{x=1}^{n} \sum_{y \in N(x)} \sum_{i=0}^{n} o_{y,i}$

$E_2 = \sum_{x=1}^{n} \sum_{y=1, y \neq x}^{n} \sum_{i=1}^{n} d_{xy} \cdot o_{x,i} \cdot (o_{y,i+1} + o_{y,i-1})$

minimize $E_1 + E_2$

$$\tag{11}$$

$$\text{Subject to} \sum_{i=1}^{n} o_{x,i} \leq 1 \text{ for all city x} \tag{12}$$

$$\sum_{x=1}^{n} o_{x,i} \leq 1 \text{ for all position i} \tag{13}$$

$$\sum_{i=1}^{n} \sum_{x=1}^{n} o_{x,i} = m = |T| \tag{14}$$

The CSP problem is to minimize the cost E_1 of the path with several constraints, which is as same as the form in [19]. The constraint (12) is to guarantee each city to be visited at most once while constraint (13) ensures at most one city to be visited at each step, referred to [14, 16, 19]. Constraint (14) ensures exact $|T|$ cities visited in solution, where $m = |T|$ is the amount of cities in the set T, or the amount of cities on the path. Since the amount of element in set T is unknown so far, the technique for finding m will be introduced in the next subsection.

The term E_2 means that for each city x, there is a set of cities covered by it, denoted as $N(x)$. This set includes city x itself. Then we need to minimize the number of cities that are selected to be on tour from this set. This constraint can reduce the number of covered cities on tour. In later section, such constraints will be represented as multiple energy functions so that the modified Hopfield network can find the valid solutions.

B. Compute m, the amount of cities required on tour

In order to calculate the least amount of cities required on tour m, we construct an adjacency matrix in undirected graph to measure the connectivity of all cities and then construct a tree data structure based on the adjacency matrix.

A greedy approach is then applied to add tree nodes and construct the tree. Finally, the number of cities required on tour is equal to the smallest depth of leaf nodes in tree. To measure the connectivity of cities for adjacency matrix, consider following definitions:

Definition 1: Let all cities have the same covering distance r, then two cities are connected if they cover each other. That is, for any city x, if there exists a city y such that $r \geq d_{x,y}$, then city y $\in N(x)$.

Note that city y can be city x itself. In order to use Tree search method to find value of m, the least amount of cities on the tour required to satisfying CSP constraints, let the number m has the following definition:

Definition 2: m is the least amount of cities required on the tour such that a set of cities covered an city on the tour and a set of cities covered by another city on the tour have as less interaction cities as possible, and each city on the tour should cover as many cities as possible.

Equivalently, the problem of finding m can be formulated as follows:

$$\min m \ , \ \max_{i=1,\ldots m} k_i \tag{15}$$

where k_i is the number of cities covered by the i^{th} city that is on tour. To do this, we construct a tree and obtain m from the tree. The steps are as follows:

Step 1: construct adjacency matrix based on definition 1

Step 2: In adjacency matrix, find the cities with the largest degree that cover the most cities.

Step 3: Add a node to current parent for each city with the largest degree value. Note that different cities could have the same largest degree value in the matrix. When this happens, add a node for each one.

Step 4: Each new node stores the new adjacency matrix updated by removing the rows and columns of its corresponding city and the cities connected to this city.

Step 5: repeat steps 2 to 4 until that the adjacency matrix passed to a node becomes empty.

Step 6: Finally, m = z - 1, where z is the smallest depth of leaf node in the tree.

To illustrate this method clearly, considers *Example* 2 and 3.

Example 2: In figure 2, all cities have the same covering distance r=0.1. City 0 and city 1, city 2 and 4 are connected respectively. Then the corresponding adjacency matrix is in Table II.

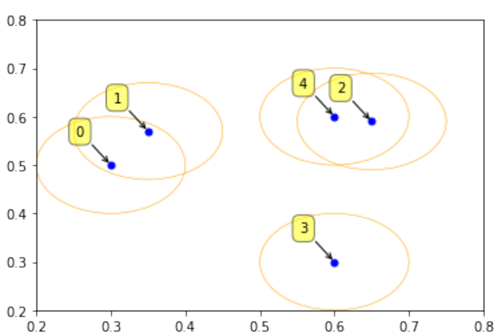

Fig. 2. separate 5 cities into 3 groups: (0,1), (2,3), (4)

TABLE II
ADJACENCY MATRIX OF EXAMPLE 2

city 0	city 1	city 2	city 3	city 4
0	1	0	0	0
1	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	1	0	0

In this case, Connections between city 0 and city 1, city 2 and city 4 are also called density directly reachable [21].

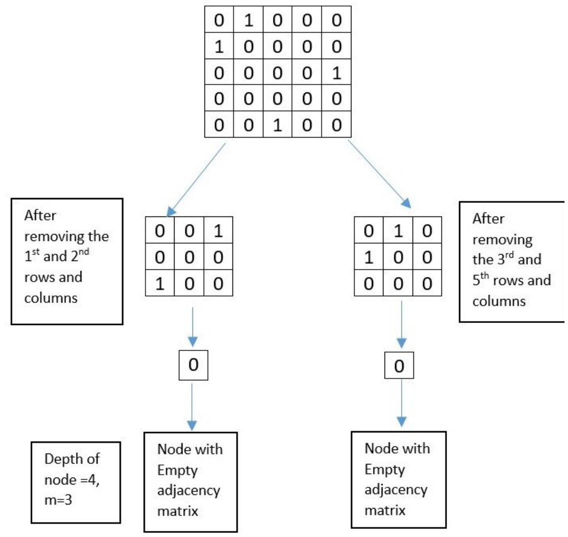

Fig. 3. Construct a tree to compute the amount of cities on the tour m

Figure 3 shows how the tree is constructed. The root node of tree store the adjacency matrix in Table II and then the cities with maximum degree are city 0, 1, 2, 4 with degree equal to 1. For each city with degree 1, add a node to root node. Follow step 4), the upper left adjacency matrix under the root node is obtained by removing city 1 and city 2 since city 2 and city 1 are connected. Hence the first and second rows and columns are removed from original adjacency matrix. Similarly, since city 4 is covered by city 2, then the 1^{st} row, 3^{rd} column associated with city 2 and the 3^{rd} row, 1^{st} column associated with city 4 are removed from the upper left matrix again. Repeat the same operations, the smallest depth of leaf nodes in the tree is 4. Hence the least amount of cities required on tour, m= 3, according to step 6). Finally, cities can be separated into 3 groups, like $\{0, 1\}$, $\{2, 4\}$, $\{3\}$. Then Hopfield network is to select exact one city from each group for the path.

Example 3: Similarly, In example 3, covering distance r=0.1.

TABLE III
ADJACENCY MATRIX OF EXAMPLE 3

city 0	city 1	city 2	city 3	city 4
0	1	0	0	1
1	0	0	0	0
0	0	0	1	0
0	0	1	0	0
1	0	0	0	0

City 0 and city 1 are connected while city 1 and city 4 are connected. Then City 0 and city 4 are so called density reachable [21]. Then m= 3,according to adjacency matrix in Table III. 5 Cities are separated into 3 clusters. Since city 0 has the largest degree 2, then city 0 and the cities covered by it, city 1 and city 4, are removed, then the adjacency matrix in the tree becomes a 2 by 2 matrix with the first row and column represents city 2, the second row and column represent city 3. The depth of leaf node is 3 and hence m= 2 cities on tour.

Remark: This method indeed applies a greedy approach to

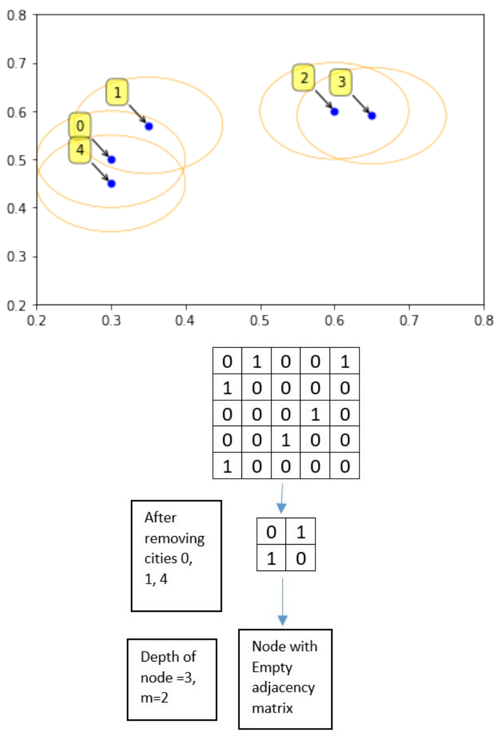

Fig. 4. Example 3

find m when it removes the cities with the largest degree value. When removing the row and column corresponding to a city with the maximum degree, the algorithm is indeed selecting the largest cluster of cities covered by this city from the city set and cutting the edges between this cluster and the remained cities outside this cluster. Then it repeats choosing the largest cluster covered by a city from the remained city set. The depth of leaf node represents the possible number of cities that cover a large groups of different cities. Hence, the smallest depth of leaf nodes in the tree is equal to the least number of cities on the tour.

C. Energy functions

This section introduces energy functions used in modified Hopfield network for CSP. The constraints (12) to (14) will be satisfied when energy functions are minimized. Energy functions used in modified Hopfield network are as follows:

$$\begin{cases} E_1 = (A/2)\sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} o_{x,i}o_{x,j} \\ E_2 = (B/2)\sum_{i=1}^{n}\sum_{x=1}^{n}\sum_{y=1,y\neq i}^{n} o_{x,i}o_{y,j} \\ E_3 = (C/2)(\sum_{x=1}^{n}\sum_{i=1}^{n} o_{x,i} - m)^2 \\ E_4 = (D/2)\sum_{x=1}^{n}\sum_{y=1,y\neq x}^{n}\sum_{i=1}^{n} d_{xy}(o_{x,i})(o_{y,i+1}+o_{y,i-1}) \\ E_5 = (F/2)\sum_{x=1}^{n}[(\sum_{y=1}^{n} g(r_x - d_{x,y})(\sum_{j=1}^{n} o_{y,j}) - 1)^2] \\ E = E_1+E_2+E_3+E_4+E_5 \end{cases} \tag{16}$$

In energy functions E_1 to E_5, A, B, C, D, F are coefficients used to weigh each energy term. Energy functions E_1 and E_2 are as same as energy functions in TSP problem [19]. E_1 guarantees at most one neuron is activated in each row of matrix representation of CSP while E_2 ensures that at most one neuron is activated in each column. E_3 enforces the constraint that exact m neurons are active in Hopfield network.Compared with Hopfield network applied on TSP [16], m in E_3 is the number of cities on the tour, instead of the total number of cities n. Energy function E_1, E_2, E_3 are corresponding to constraints (12), (13), (14). E_4 is the cost of path to minimize,corresponding to constraint (14).

E_5 is the term we design to enforce cities that are not on tour are covered by cities on tour. In E_5,for all city x, covering distance r_x = r ≥ 0. $g(\cdot)$ is a step function:

$$g(x) = \begin{cases} 1 \ \textit{if} \ x \ \geq 0 \\ 0 \ \text{otherwise} \end{cases} \tag{17}$$

Hence, when $r - d_{x,y} \geq 0$ and x≠y, it means that city x and y are connected. Otherwise, if x=y, then city y is city x itself and $d_{x,y} = d_{x,x} = 0$ then $g(r - d_{x,y}) = 1$. When $g(r - d_{x,y})$ is 1, then the term $\sum_{j=1}^{n} o_{y,j}$ matters. Due to the term $(\sum_{y=1}^{n} g(r-d_{x,y})(\sum_{j=1}^{n} o_{y,j}) - 1)^2$, E_5 has value greater than or equal to zero. Besides, if this term is equal to 0, exact one city in the cluster of cities covered by city x, including city x itself, is selected to be on the tour and exact one neuron for this cluster of cities is active. Therefore, as E_5 converges the minimum point, E_5=0, more and more cities not on tour are covered by cities on tour.

In addition, when all covering distances r_x=0, $g(r_x - d_{x,y})$ =1 only when city y is city x itself. That is, the term $\sum_{y=1}^{n} o_{y,j}$ matters only when x=y. In this case, if E_5 =0, it requires that each row in the matrix representation of CSP has exact one active neuron. Each city must be on tour. This leads to the Traveling salesman problem. Hence, when r=0, CSP transits to TSP.

From energy functions above, when $\tau = 1$ and $\triangle t$=1 in equation (4), (5), the update of input potential, internal state and the first-order derivative of total energy $E = E_1 + E_2 + E_3 + E_4 + E_5$ leads to :

$$\begin{aligned} u_{z,i} &= -\frac{dE}{do_{z,i}} \\ &= -A(\textstyle\sum_{j=1,j}^{n} o_{z,j}) - B(\sum_{y=1,y}^{n} o_{y,i}) \\ &-C(\textstyle\sum_{y=1}^{n}\sum_{j=1}^{n} o_{y,j} - m) \\ &-D\textstyle\sum_{y=1}^{n} d_{z,y}(o_{y,i-1}+o_{y,i+1}) \\ &-F\textstyle\sum_{x=1}^{n} g(r-d_{x,z})[\sum_{y=1}^{n} g(r-d_{x,y})(\sum_{j=1}^{n} o_{y,j}) - 1] \end{aligned} \tag{18}$$

However, such form of $\frac{dE}{do_{z,i}}$ is hard to compute and could lead to computation complexity of $O(n^3)$ due to the term from the first-order derivative of E_5. Nevertheless, we know that equation (19) can be expressed by the form in equation (8) once we know the weight matrix W and bias term i_b. From equation (8) and states used in CSP, we obtain

$$u_{z,i} = -\frac{dE}{do_{z,i}} = \sum_{y=1}^{n}\sum_{j=1}^{n} W_{z,i,y,j} v_{y,j} + i_b \tag{19}$$

By observing equation (7) and (9), the relationship between weight and the second-order derivative of energy is

$$W_{zi,kj} = -\frac{d^2E}{do_{z,i}do_{k,j}} = \frac{du_{z,i}}{do_{k,j}} \tag{20}$$

From equations (19) and (20), the modified version of the weights in paper [19] is:

$$\begin{aligned} W_{zi,kj} &= -A\delta_{z,y}(1-\delta_{i,j}) - B\delta_{i,j}(1-\delta_{z,k}) \\ &-C - Dd_{z,k}(o_{k,i-1} + o_{k,i+1}) \\ &-F \textstyle\sum_{x=1}^{n}(g(r-d_{xz})g(r-d_{xk})) \end{aligned} \tag{21}$$

where the last term with coefficient F is an additional term we design and other terms are from the Hopfield network for TSP in [19]. $\delta_{z,k}$, $\delta_{i,j}$ are delta function defined by:

$$\delta_{z,k} = \begin{cases} 1 \text{ , if z=k} \\ 0 \text{ , otherwise} \end{cases} \tag{22}$$

And the bias term is

$$i_b = Cm + F\sum_{x=1}^{n} g(r - d_{x,z}) \tag{23}$$

Proof: Since E_5 is the only term we design, the first-order and second-order derivative of other energy functions can be found in [19] and [14].

$-\frac{dE_5}{do_{z,i}} =$

$(F/2)\sum_{x=1}^{n}[(\sum_{y=1}^{n} g(r-d_{x,y})(\sum_{j=1}^{n} o_{y,j}) - 1)\cdot 2\cdot g(r-d_{x,z})]$

$= (-F)\sum_{x=1}^{n} g(r-d_{x,z})[\sum_{y=1}^{n} g(r-d_{x,y})(\sum_{j=1}^{n} o_{y,j}) -1]$

$=$

$(-F)\sum_{x=1}^{n} g(r-d_{x,z})[\sum_{y=1}^{n} g(r-d_{x,y})(\sum_{q=1}^{n} o_{y\neq k,q\neq j})$

$-1] + (-F)\sum_{x=1}^{n}(g(r-d_{xz})g(r-d_{xk})o_{k,j})$

Where $o_{y\neq k,q\neq j}$ means external states don't include the state $o_{k,j}$. The term associated with $o_{k,j}$ is moved out from original equation. Then from the first-order derivative above, we get

$-\frac{d^2E_5}{do_{z,i}do_{k,j}} = -F\sum_{x=1}^{n}(g(r-d_{xz})g(r-d_{xk}))$

The bias term i_b can be derived from the constant terms in equation (19). In equation (19), it has

From the first derivative of E_3
$-C(\sum_{y=1}^{n}\sum_{j=1}^{n} o_{y,j} - m) =$

$-C(\sum_{y=1}^{n}\sum_{j=1}^{n} o_{y,j}) + Cm$

From the first derivative of E_5
$-F\sum_{x=1}^{n} g(r-d_{x,z})[\sum_{y=1}^{n} g(r-d_{x,y})(\sum_{j=1}^{n} o_{y,j}) - 1]$

$= -F\sum_{x=1}^{n}[\sum_{y=1}^{n} g(r-d_{x,z})g(r-d_{x,y})(\sum_{j=1}^{n} o_{y,j})]$

$+F\sum_{x=1}^{n} g(r-d_{x,z})$

where Cm and $F\sum_{z=1}^{n} g(r-d_{x,z})$ are constant terms, composing the bias.

For consistency of notations, we use notations $o_{x,i}$, $u_{x,i}$, rather than $o_{z,i}$, $u_{z,i}$, or $o_{k,i}$, $u_{k,i}$ in following sections.

Therefore, we can derive the weight and bias from equations (21) and (23). Combining the dynamic equation (4) and equation (5) for updating internal state, we can obtain the update rule for internal state u. Besides, using equations (2), (3) and update rule of internal state lead to the update rule of external state $o_{x,i}$.

D. Algorithm

Algorithm 1 Update of modified Hopfield Network

Input:
n: total amount of cities
N: the number of initial external state o to try
max_itr: maximum iteration for updating state
parameters: A,B,C,D,F, $\mu, u_0, \alpha, r_x,$ distance matrix
Output:
p : matrix representation of path solution

Procedure:
initialization of parameters
Calculate m, the amount of cities required on tour
Calculate weight W and bias i_b
Generate a list of N initial external state o randomly
for *j=1... N* **do**
  select j^{th} initial external state o(t=0)
  Initial internal state u_0 =u(t=0) from list
  for *itr=1.. max_itr* **do**
    $\frac{du}{dt} \leftarrow -\frac{1}{\tau}u + Wo + i_b$
    $u_{x,i} \leftarrow u_{x,i} + \alpha\frac{du_{x,i}}{dt}$
    $o_{x,i} \leftarrow 1$ if $f(u_{x,i}) \geq \lambda, 0$ otherwise For all x, i
    Compute Energy E
    if *o satisfies all constraints* **then**
      add valid o to solution list
    end
  end
  Calculate cost of paths in solutions
  Update p the path solution with minimum cost
end

In the algorithm shown, α denotes the change of time $\triangle t$ in equation (5), which is set to be 10^{-5} as default, as same as the value in paper [16]. τ is set to be 1 as default. λ is the threshold used to convert $f(u_{x,i})$ to binary external state $o_{x,i}$.

Since Hopfield network aims at finding the local minima of energy and starting from different initial external state can lead to different local minimas, a list of different initial external states is first generated to train the network. Later, experiment section discusses details about parameters, initial states and other settings.

V. Experiment

A. Generalizing TSP to CSP problem

To show that the our Hopfield network can successfully solves the generalization of TSP, we apply our networks to both TSP and CSP on the same city set, city set 1 in figure 10 shown in appendix. Two networks applied on TSP and CSP have the same basic settings:

$A = B = 50$, $C = 100$, $D = 60$, $F = 100$, $\mu = 0.02$, $\tau = 1$. $\alpha = \triangle t = 10^{-5}$. The initial internal state $u_0 = 0$. Threshold λ=0.7. max_iter= 1000 and N=30, 30 different initial external states $o(t = 0)$ attempted during training the network.

Each initial external state is generated randomly such that each row in matrix representation of state has at most one active neuron. This method allows the network to find valid solution easier since it lets the initial external state satisfy the constraint 12 directly. Besides, as we find that using N=30, training the network with 30 different initial external states, is already able to let our networks obtain the optimal solutions, we simply select N=30 for all following simulations. Two networks for both TSP and CSP use the same 30 randomly generated initial external states o to start training the networks.

The only different setting between two Hopfield networks in TSP and CSP is that covering distance r=0 in TSP and r= 0.1 in CSP.

After changing covering distance r to 0.1, the covering areas, yellow circles, in CSP are shown in figure 10. Then the corresponding optimal solution to CSP is shown in 6. In TSP with r=0, there is no covering distance and it requires all cities must be visited once. Hence the optimal solution shown in 5 has no yellow circles.

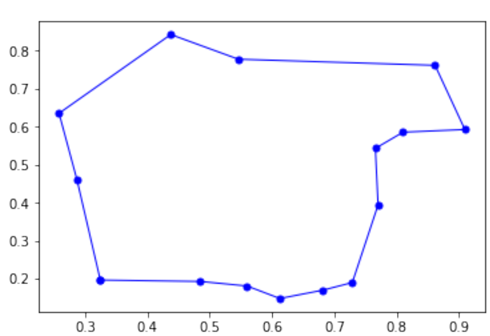

Fig. 5. TSP solution on city set 1 with r=0, A=B=50,C=100, D= 60, F=100

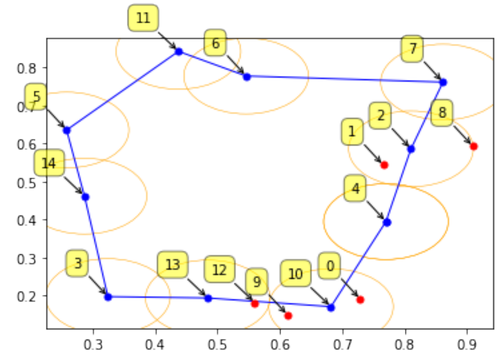

Fig. 6. CSP solution on city set 1 with r =0.1, A=B=50,C=100, D=60, F=100

City set 1 with 15 cities			
Cases Valid Solution Rate	Path Cost		
CSP	worst	best	average
D=60,F=100	3.811	2.154	3.031
D=40,F=100	4.654	2.140	3.397
D=80,F=100	3.793	2.154	2.88
TSP	worst	best	average
D=40,F=100	4.824	2.972	3.807
D=60,F=100	3.522	2.544	3.232
D=80,F=100	3.266	3.266	3.266

TABLE IV
City set 1 performance for CSP and TSP

Table IV shows the best, worst and average path costs in each case. The valid solution rate is obtained by

$$rate = \frac{N_{valid}}{N_{total}} \tag{24}$$

where N_{valid} is the number of initial external state that leads to valid solutions. N_{total} is the total number of initial external state. Since we try 30 different initial external states for all simulations, N_{total}=30. we can observe that the at these settings, the shortest paths explored by the networks for CSP and TSP have cost 2.154 and 2.972 respectively.Moreover, different values of coefficient D are tested to evaluate the effect of weight of path cost on performance of solutions. The results are shown in table IV. From the table, we observe that as D increases, both the solution rate and the cost of the worst solution are decreasing for both CSP and TSP. This implies that as the term of path cost given larger weight, two networks are searching for the paths with lower path cost. Compared with TSP, the network for CSP has much higher solution rate. This is because Constraints in TSP, requiring all cities to be visited once exactly, are more strict than those in CSP. Hence, there less valid solutions in TSP than in CSP, which makes the network in TSP hard to find valid solutions. Hence when solving generalization of TSP, we can simply set covering distance r >0 to enable Hopfield network to solve CSP. Otherwise, r=0 can lead to TSP solutions.

Note that since TSP has more strict constraints than its generalization problem, coefficients A, B, C, D may need changes to achieve better solutions. In addition, if r=0, energy

E_5 becomes an constraint requiring that each row in matrix representation of TSP has exact one active neuron. E_5 becomes a more strict version of energy E_1 and coefficients A, F have similar impact. The performance of network related to coefficients D, F are discussed in the next section of experiment.

B. Analysis of performance

As the effect of coefficients A, B, C on performance has been studied by previous researchers [16, 17, 19], this section performs simulations on 3 city sets, labeled as City set 2, 3, 4, to evaluate the effect of D and F on performance of our network in CSP. All simulations have the similar settings as the first section:

$A = B = 50$, $C = 100$, $r = 0.1$, $\mu = 0.02$,$\tau = 1$, $\alpha = 10^{-5}$. The initial internal state u_0 = 0. Threshold λ=0.7. max_iter= 1000 and N=30, 30 different initial external states o generated randomly in each simulation.

VI. Analysis of coefficients D and F

City set 2			n = 10	$n-m$ =1
coefficient	worst	best	average	valid solution rate
D=60,F=100	3.624	2.680	3.143	0.467
D=40,F=100	4.641	2.685	3.614	0.867
D=80,F=100	3.105	2.767	2.969	0.1
D=60,F=80	3.739	2.680	3.138	0.533
D=60,F=120	3.985	2.940	3.235	0.4
City set 3			n = 10	$n-m$ =3
coefficient	worst	best	average	valid solution rate
D=60,F=100	3.100	1.993	2.407	0.93
D=40,F=100	3.540	2.047	2.618	1.0
D=80,F=100	2.685	1.984	2.278	0.53
D=60,F=80	2.910	2.00	2.447	0.93
D=60,F=120	3.407	2.012	2.454	1.0
City set 4			n = 13	$n-m$ =6
coefficient	worst	best	average	valid solution rate
D=60,F=100	3.135	2.041	2.601	0.96
D=40,F=100	3.283	2.0412	2.570	1.0
D=80,F=100	3.1537	2.323	2.654	0.86
D=60,F=80	3.338	2.064	2.637	0.96
D=60,F=120	3.234	2.120	2.613	1.0
City set 1			n = 15	$n-m$ =10
coefficient	worst	best	average	rate for CSP
D=60,F=100	3.811	2.154	3.031	0.96
D=40,F=100	4.654	2.140	3.397	1.0
D=80,F=100	3.793	2.154	2.88	0.7

TABLE V
PERFORMANCE N CITY SETS 1,2,3,4. IT PRESENTS THE WORST, BEST AND AVERAGE PATH COSTS OF SOLUTIONS FOUND BY THE NETWORK

In table V, n is the total number of cities and m is the number of cities on the tour. Hence $n - m$ is the number of cities covered by cities on tour. The optimal solutions with minimum path costs of three city sets are shown in figures 7, 8 and 9. In order to evaluate the impact of D and F, we choose $D = 60$ and $F = 100$ as base case and then change D, F respectively. The results of performance on City sets 2, 3, 4 are shown in tables V. From table, without changing F, increasing D from 40 to 80 leads to higher valid solution rate and lower path cost, which implies the same conclusion as the last simulation on City set 1. For City set 2, there is only one city (city 8) covered (red point), according to figure 7. With $A = B = 50$, C=100,$D = 60$, as F increases, the rate of finding valid solution is decreasing obvious and the average and the worst costs of path increase. The solution rate decreases from 0.533 to 0.4 as F increases from 80 to 120.

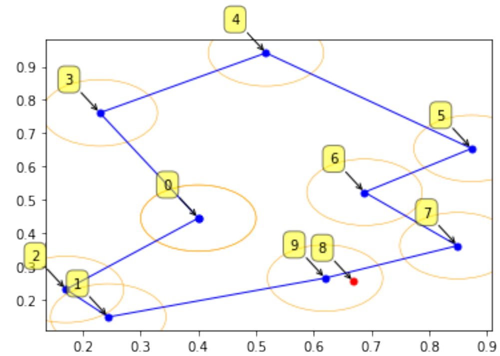

Fig. 7. CSP solution for City set 2

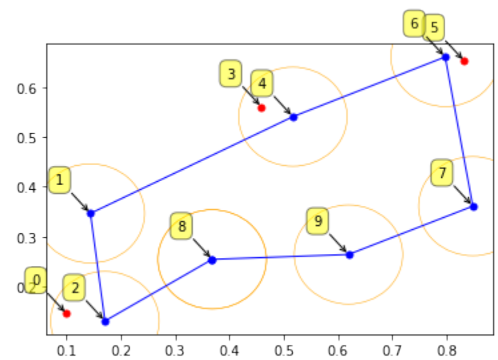

Fig. 8. CSP solution for City set 3 with r=0.1, A=B=50,C=100, D=75, F=100

Compared with City 2, the changes of solution rate and of path cost are slight in City set 3 and 4. There are 3 cities covered in City set 3 and 6 cities covered in City set 4, according to figures 8 and 9. Their rates of obtaining valid solution are almost 1.0. This implies that as more cities are covered by cities on tour, easier it is for the network to find valid solution. As less cities are covered, CSP is more similar to TSP and less solutions can be found. Simulation on City set 1 is the extreme case of this phenomenon.

Overall, from simulations above, we find that as the number of cities covered by other cities on tour increases, the Hopfield network we propose can find valid solutions easier. Coefficient F has the similar impact as coefficient A, but enforces the network to have as less cities selected to be on tour as possible. if the number of cities covered and not on tour is small, increasing F could lead to more difficulty in finding valid solutions.

VII. Future work

As this paper proposes an extension of Hopfield network to solve the Covering salesman problem, one of NP hard

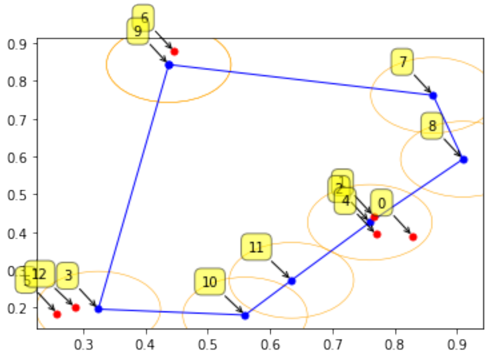

Fig. 9. CSP solution for City set 4 with r=0.1, A=B=50,C=100, D=75, F=100

problems, there are several works worth researching and improving in future.

Covering salesman problem so far, there are several different versions, or more general versions of Covering salesman problem proposed, such as online CSP, CSPNS, and online Traveling salesman problem [18, 20, 22]. Those problems give a more general overview about CSP problem. Inspired by them, an online version of covering salesman problem with node and segments may be worth researching.

Hopfield Network Although we propose a new Hopfield network for solving CSP, there is an obvious problem that as the number of cities n increases, the number of neuron required to model CSP are n^2 and the weight matrix has $n^2 \times n^2$. That is, the memory complexity becomes $O(n^4)$, which is not efficient enough when involving a large of cities. New methods with lower memory complexity and computation complexity are expected.

VIII. Conclusion

In conclusion, this paper re-formulates and extends Hopfield network to solve the generalization of traveling salesman problem, Covering salesman problem(CSP). It gives a more general solution of energy method for solving this kind of problem. Our simulation results show that the proposed Hopfield network is able to solve TSP problem and CSP problem successfully and return the optimal path solutions. This paper shows how to calculate the number of cities required on tour m after modifying the energy term E_3 and also investigates the influence of the new coefficient F and energy function E_5 we introduce. Finally, future work for solving CSP problem is discussed. The Appendix page at the end of this paper shows the positions of cities used in simulations.

References

[1] Z. Uykan, ”Shadow-Cuts Minimization/Maximization and Complex Hopfield Neural Networks,” in IEEE Transactions on Neural Networks and Learning Systems.April.2020.

[2] M. Atencia, G. Joya, and F. Sandoval, “Dynamical analysis of continuous higher-order hopfield networks for combinatorial optimization,” Neural Comput., vol. 17, no. 8, pp. 1802–1819, Aug. 2005.

[3] M. Kobayashi, ”Decomposition of Rotor Hopfield Neural Networks Using Complex Numbers,” in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 4, pp. 1366-1370, April 2018, doi: 10.1109/TNNLS.2017.2657781.

[4] S. Jankowski, A. Lozowski, and J. M. Zurada, “Complex-valued multistate neural associative memory,” IEEE Trans. Neural Netw., vol. 7, no. 6, pp. 1491–1496, 1996.

[5] Z. Uykan, “Fast-convergent double-sigmoid hopfield neural network as applied to optimization problems,” IEEE Trans. Neural Netw. Learn. Syst., vol. 24, no. 6, pp. 990–996, Jun. 2013.

[6] D. Ying, ”Competition Decision for Bottleneck Traveling Salesman Problem Based on Big Data Mining Algorithm with Multi-Segment Support,” 2018 3rd International Conference on Smart City and Systems Engineering (ICSCSE), Xiamen, China, 2018, pp. 725-729.

[7] J. Yang, J. Jiang and Y. Lai, ”A Decreasing k-means algorithm for the Disk Covering Tour Problem in wireless sensor networks,” 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), Hsinchu, 2014, pp. 906-910.

[8] Shih Y. Chin, A. R. P. Neto and E. V. G. Filho, ”Application of the Traveling Salesman Problem Heuristics to the Reallocation of Equipment in a Small-size Bakery Aiming at Minimizing Bread Production Time,” 2007 Winter Simulation Conference, Washington, DC, 2007, pp. 2375-2375, doi: 10.1109/WSC.2007.4419890.

[9] B. Di, R. Zhou, Y. Zhang and J. Che, ”Path planning for unmanned vehicle searching based on sensor deployment and travelling salesman problem,” Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference, Yantai, 2014, pp. 1775-1779

[10] C. Yang and K. Y. Szeto, ”Solving the Traveling Salesman Problem with a Multi-Agent System,” 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 2019, pp. 158-165

[11] S. Birtane Akar and Ö. K. Şahingöz, ”Solving asymmetric traveling salesman problem using genetic algorithm,” 2015 23nd Signal Processing and Communications Applications Conference (SIU), Malatya, 2015, pp. 1655-1659

[12] J. Liu and W. Li, ”Greedy Permuting Method for Genetic Algorithm on Traveling Salesman Problem,” 2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, 2018, pp. 47-51

[13] J. Stastný, V. Skorpil and L. Cizek, ”Traveling Salesman Problem optimization by means of graph-based algorithm,” 2016 39th International Conference on Telecommunications and Signal Processing (TSP), Vienna, 2016, pp. 207-210, doi: 10.1109/TSP.2016.7760861.

[14] J. J. Hopfield and D. W. Tank, “‘Neural’ computation of decisions in optimization problems,” Biol. Cybern., vol. 55, pp. 141–146, Jul. 1985.

[15] Current, John R. and David A. Schilling. “The Covering Salesman Problem.” Transportation Science 23 (1989):

208-213.
[16] Mańdziuk, Jacek. (1996). Solving the Travelling Salesman Problem with a Hopfield-type neural network. Demonstratio Mathematica. 29. 219-231.
[17] Potvin JY., Smith K.A. (2003) Artificial Neural Networks for Combinatorial Optimization(pp. 431-438). In: Glover F., Kochenberger G.A. (eds) Handbook of Metaheuristics. International Series in Operations Research Management Science, vol 57. Springer, Boston, MA.
[18] Zhang, H., Xu, Y. Online covering salesman problem. J Comb Optim 35, 941–954 (2018).
[19] Robert J. Schalkoff. Artificial Neural Network.pp.280-284.1997
[20] Matsuura, T. and Kimura, T. (2017) Covering Salesman Problem with Nodes and Segments. American Journal of Operations Research, 7, 249-262
[21] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1997. Density-connected sets and their application for trend detection in spatial databases. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97). AAAI Press, 10–15.
[22] Wen, X Xu,Y and Zhang,H. Online traveling salesman problem with deadline and advanced information.Computers Industrial Engineering. 2012,1048-1053
[23] Rossi, F , Beek, P and Walsh,T. Handbook of Constraint Programming, in Foundations of Artificial Intelligence,Aug,2006, Ch.2.
[24] M. Sahu, A. V. Singh and S. K. Khatri, "A Classical Constraint Satisfaction Problem and its Solution using Artificial Intelligence," 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 2019, pp. 429-433.
[25] A. Potebnia, "Representation of the greedy algorithms applicability for solving the combinatorial optimization problems based on the hypergraph mathematical structure," 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv, 2017, pp. 328-332,

Appendices

Appendix A
Positions of City sets

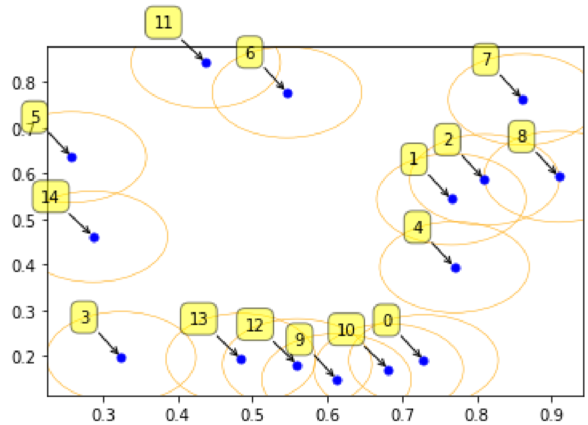

Fig. 10. City Set 1: positions of 15 cities labeled from 0 to 14

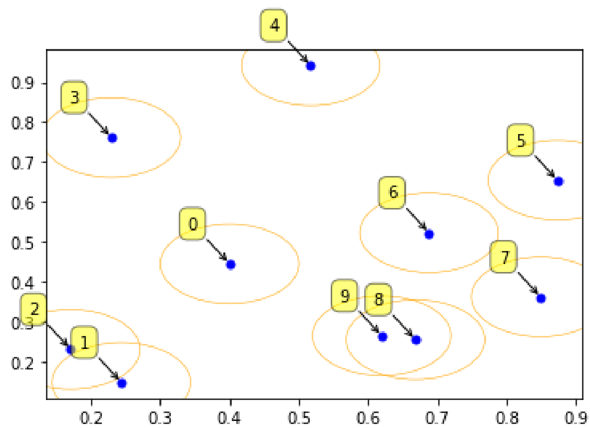

Fig. 11. City Set 2: positions of 10 cities labeled from 0 to 9

city index	x	y
0	0.729	0.19
1	0.766	0.543
2	0.81	0.586
3	0.324	0.196
4	0.77	0.394
5	0.258	0.636
6	0.546	0.778
7	0.862	0.762
8	0.91	0.593
9	0.612	0.148
10	0.682	0.17
11	0.437	0.843
12	0.56	0.181
13	0.484	0.193
14	0.287	0.461

TABLE VI
City set 1

city index	x	y
0	0.4	0.444
1	0.244	0.146
2	0.171	0.229
3	0.229	0.761
4	0.517	0.941
5	0.873	0.654
6	0.688	0.522
7	0.849	0.361
8	0.668	0.254
9	0.62	0.263

TABLE VII
City Set 2

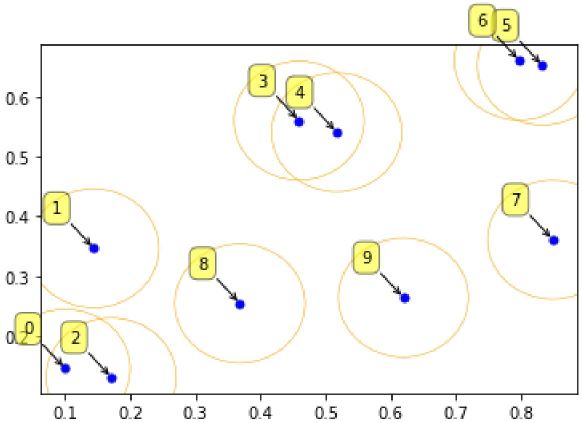

Fig. 12. City Set 3: positions of 10 cities labeled from 0 to 9

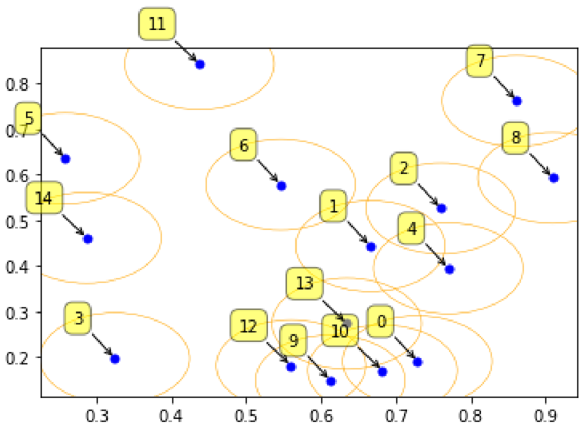

Fig. 13. City Set 4: positions of 13 cities labeled from 0 to 12

city index	x	y
0	0.1	0.144
1	0.144	0.346
2	0.171	0.129
3	0.459	0.561
4	0.517	0.541
5	0.833	0.654
6	0.798	0.662
7	0.849	0.361
8	0.368	0.254
9	0.62	0.263

TABLE VIII
CITY SET 3

city index	x	y
0	0.829	0.39
1	0.766	0.443
2	0.76	0.426
3	0.324	0.196
4	0.77	0.394
5	0.258	0.186
6	0.446	0.878
7	0.862	0.762
8	0.91	0.593
9	0.437	0.843
10	0.56	0.181
11	0.634	0.273
12	0.287	0.201

TABLE IX
CITY SET 4