Minimum set of controlled nodes to fully control dynamic networks

***Abstract*—Network controlling is a problem of great importance, for its wide range of applications. Although lots of work have been devoted to gain insight into this problem, there still lacks a work to study the minimum set of controlled nodes to fully control an arbitrary network, given its transmission matrix. Therefore, we initiate the study of minimum set of controlled nodes of dynamic networks. First of all, we give the formulation of the problem of the minimum set of controlled nodes (MSCN) for finding a minimum set of controlled nodes to fully control an arbitrary network. We show that MSCN is a NP-hard problem by reducing Set Covering Problem to it. Then, we propose an exponential algorithm which can find the minimum set of control nodes and an $O(n^4)$ approximation algorithm of MSCN problem. We also propose an greedy algorithm to solve MSCN. Finally, we propose an Ant Colony Optimization (ACO) based approximation algorithm to solve MSCN. Extensive simulations have been conducted to corroborate our analysis.**

***Index Terms*—Controlled nodes, Network control, Fully control, Controlling matrix, Ant Colony Optimization**

I. Introduction

There are many networks both in the real world, such as food web, transportation networks, communication networks and gene networks. Network controlling involves both network science and control theory.

There are linear networks and nonlinear networks in the real world. Controlling nonlinear networks is more complicated than controlling linear networks. But they have similarities in many aspects. So studying linear network controlling can help us gain some insight into nonlinear network controlling. In this paper, we focus on linear network controlling.

We use the classic linear, time-invariant model to describe the process of dynamic networks:

$$\overrightarrow{x(t+1)} = A \cdot \overrightarrow{x(t)} + B \cdot \overrightarrow{u(t)} \tag{1}$$

where $\overrightarrow{x(t)}$ represents the state values of all the N nodes in the network, while $\overrightarrow{u(t)}$ represents all the input signals of M controllers. A is a $N \times N$ transmission matrix whose elements indicate the weight of links between two nodes. B is a $N \times M$ controlling matrix, which indicates the weight of each link from a controller to a control node. The elements in matrix A and B is fixed during the process under the assumption of linear, time-invariant model.

In control theory, we say a system is controllable if given a final state, we can use input signals to drive the system to it in finite time. Control theory gives us some conditions to judge whether a linear, invariant system (A, B) is controllable or not, such as Gram matrix condition, Kalman rank condition, PBH rank condition and Jordan canonical form condition. In this paper, we mainly use the PBH rank condition and Jordan canonical form to get some results. The PBH rank condition tells us that if a linear, invariant system (A, B) is controllable, then

$$rank(cI_N - A, B) = N \tag{2}$$

holds for any complex number c. Here, I_N is a $N \times N$ identity matrix. It has a equivalent form as below : A linear, invariant system (A, B) is controllable, then

$$rank(\lambda I_N - A, B) = N \tag{3}$$

holds for every eigenvalue of matrix A.

In the recent years, some theories about network controllability have been proposed, such as structural controllability theory and exact controllability theory. Structural controllability theory was proposed by Liu *et al.* It introduces the concept of structural controllability and gives a method to find a minimum set of driver nodes, employing maximum matching algorithm. Exact controllability was proposed by Yuan *et al.* This theory is based on the PBH rank condition. It gives the minimum number N_D of controllers to fully control an arbitrary network :

$$N_D = \max_i \mu(\lambda_i) \tag{4}$$

where λ_i (i ranges from 1 to N) is the $i-th$ eigenvalue of A and $\mu(\lambda_i)$ is the geometric multiplicity of eigenvalue λ_i, defined as:

$$\mu(\lambda_i) = dimV_{\lambda_i} = N - rank(\lambda_i I_N - A). \tag{5}$$

Although many researchers have devoted great effort to explore the nature of network controllability, to our best knowledge, there still lacks any work focused on the minimum number of control nodes needed to fully control an arbitrary network. This problem is of great importance in some scenarios. Imagine that there is a social network, and we need to lead the direction of public opinion on a particular topic. It's very straightforward that we need to influence some people in the social network first, then they will change the opinion of the people related to them, an so on. For simplicity, the fewer people we need to directly influence, the better.

In this paper, we propose the problem of the minimum set of controlled nodes (MSCN), which is to find out a minimum set of controlled nodes to fully control an arbitrary network. The main challenge of the problem is to minimize the number of controlled nodes. We also study the relationship between the minimum number of driver nodes (N_D) and the minimum number of control nodes (N_C).

The main contributions of this paper are:

1) We formulate the problem of the minimum set of controlled nodes (MSCN) and prove that it is an NP-hard problem;
2) We propose an exponential algorithm to find a set of control nodes to fully control a network, whose size is exactly the minimum number of control nodes;
3) We propose an $O(n^4)$ approximation algorithm of MSCN problem;
4) We propose an greedy approximation algorithm to solve MSCN problem;
5) We propose an ACO based approximation to solve MSCN problem;

The remainder of this paper is organized as follows. In the following section, we give the system model and the formulation of MSCN problem. Section III highlights some related works on the MSCN problem. In Section IV, we explore the relationship between N_D and N_C. We prove the NP-hardness of the MSCN problem in Section V and propose an exponential algorithm that can get the optimal solution and an $O(n^4)$ approximation algorithm and a greedy approximation algorithm of MSCN problem. We propose an ACO based algorithm to solve MSCN in Section VI. Extensive simulations have been conducted by us and Section VII gives the results and analyses. Section VIII concludes the paper.

II. Preliminaries

A. System Model

We take a dynamic directed network $G(A)$ into consideration. Suppose that there are N nodes $V = v_1, v_2, \cdots, v_N$ in $G(A)$. In the linear, time-invariant model, $\overrightarrow{x(t)}$ represents the state values of all the N nodes in the network at time t. Matrix A is called the transmission matrix of network $G(A)$. A is a $N \times N$ matrix. If the element a_{ij} of A is nonzero, then there must exist a directed link from vertex v_j to vertex v_i, and the value of a_{ij} is the weight of the directed link. For undirected networks, if a_{ij} is zero, then a_{ji} must be zero as well. If a_{ij} is nonzero, then a_{ji} is nonzero and $a_{ji} = a_{ij}$.

This paper is mainly focused on controlling the dynamic network $G(A)$ by employing a set of control nodes. We need to design a controlling matrix B, which has N rows. And the number of columns in B indicates the number of controllers. B gives us the information of the connection from the controllers to the control nodes. Every nonzero row is related to a control node in $G(A)$. The network $G(A, B)$ is called a controlled network. We can divide time into equal slots and control the network in a discrete way. We assume $G(A, B)$ fits the linear, time-invariant model well, and have the formulation as :

$$\overrightarrow{x(t+1)} = A \cdot \overrightarrow{x(t)} + B \cdot \overrightarrow{u(t)}.$$

B. Control Nodes

To explicitly explain our research result, we give the definition of controller and control node here.

Definition 2.1: A controller is a external node that can offer signals to change the state of nodes in the network.

Definition 2.2: A controlled node of a network is a node which is directly controlled by one controller or more.

C. Problem Formulation

In this paper, we want to find a minimum set of controlled nodes to fully control an arbitrary dynamic network. Therefore, we formulate the problem of Minimum Set of Controlled Nodes as follows:

Problem 1: For any dynamic network $G(A)$, find a set S of controlled nodes to fully control it and the size of S is minimum.

III. Related Works

A. PBH rank condition

A linear, time-invariant network $G(A, B)$ is said to be *controllable* if we can drive it from any start state to any final state that we desire in finite time by imposing external signals which is time-invariant. It is shown in [1] that $G(A, B)$ is controllable if and only if it satisfies the PBH rank condition that :

$$rank(cI_N - A, B) = N$$

holds for any complex number c. Here, I_N is a $N \times N$ identity matrix. It has a equivalent form as below : A linear, invariant system (A, B) is controllable, then

$$rank(\lambda I_N - A, B) = N$$

holds for every eigenvalue of matrix A. PBH rank condition is one of the conditions that control theory gives us, and it is very useful in the scenario of MSC problem.

B. Jordan Canonical Condition

Jordan canonical condition is another condition to judge if a network $G(A, B)$ is fully controllable. Jordan canonical form is very important in linear algebra. Any $N \times N$ square matrix A can be transformed into a matrix

$$J = \begin{bmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_l \end{bmatrix}$$

where

$$J_i = \begin{bmatrix} J_{i1} & & & \\ & J_{i2} & & \\ & & \ddots & \\ & & & J_{i\alpha_i} \end{bmatrix}$$

where

$$J_{ik} = \begin{bmatrix} \lambda_i & 1 & & & \\ & \lambda_i & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & & & & \lambda_i \end{bmatrix}$$

The coupling matrix A can also be transformed to get $J = P^{-1}AP$ where P is an invertible matrix. And $Q = P^{-1}B$ is the transformed matrix of B.

The Jordan canonical form condition is: for every J_i, the corresponding rows of the last rows of $J_{i1}, J_{i2}, \cdots, J_{i\alpha_i}$ in matrix Q are linearly independent. The Jordan canonical form condition is very important in our later discuss.

C. *Exact Controllability*

Exact controllability theory was introduced by Yuan *et al.* The exact controllability theory gives the minimum number N_D of controllers to fully control an arbitrary network :

$$N_D = \max_i \mu(\lambda_i)$$

where λ_i (i ranges from 1 to N) is the $i-th$ eigenvalue of A and $\mu(\lambda_i)$ is the geometric multiplicity of eigenvalue λ_i, defined as:

$$\mu(\lambda_i) = dimV_{\lambda_i} = N - rank(\lambda_i I_N - A).$$

In fact, the geometric multiplicity of eigenvalue λ_i is the maximum number of independent eigenvectors belong to eigenvalue λ_i.

And for undirected networks, the geometric multiplicity $\mu(\lambda_i)$ of eigenvalue λ_i equals the algebraic multiplicity $\delta(\lambda_i)$ of eigenvalue λ_i, which is also the eigenvalue degeneracy. Therefore, for undirected networks,

$$N_D = \max_i \delta(\lambda_i)$$

also holds.

IV. Minimum Number of Controlled Nodes

In this section, we focus on the relationship between the minimum number of controllers or independent driver nodes (N_D) and the minimum number of controlled nodes (N_C).

A. *Lower and Upper bounds of N_C/N_D*

It is obvious that $N_C \geq N_D$, for that if there are less controlled nodes than controllers, for each controlled node, we can use a controller to control it to guarantee the controllability of the network. Then there are redundant controllers. Therefore, the number of controllers is not minimum, which is inconsistent with the assumption. Consider the following example:

$$A = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

It is easy to compute that the matrix A has a single eigenvalue 0. And for eigenvalue $\lambda = 0$, $\lambda I_N - A = -A$. According to the exact controllability theory, $N_D = N - rank(-A) = N - (N-1) = 1$. And $N_C = N_D = 1$. Therefore, the lower bound of N_C/N_D is 1.

Then we consider the upper bound of N_C/N_D. It is straightforward that $N_D \geq 1$, because we need at least one controller to control a network. If N_D is set to 1, we consider how large N_C could be. Suppose N is even. Consider the following matrix A:

$$A = \begin{bmatrix} 0 & 1 & & & & & & \\ 1 & 0 & & & & & & \\ & & 0 & 2 & & & & \\ & & 2 & 0 & & & & \\ & & & & & \ddots & & \\ & & & & \ddots & & & \\ & & & & & & 0 & N/2 \\ & & & & & & N/2 & 0 \end{bmatrix}$$

It is easy to compute that the eigenvalues of matrix A are: $\pm1, \pm2, \cdots, \pm N/2$, and $N_D = 1$. And to make the PBH condition satisfied for every eigenvalue, at least $N/2$ nodes need to be directly controlled. Therefore, $N_C = N/2$. This result is in agreement with the structural controllability theory [2]. According to the structural controllability theory, many element cycles can be controlled by a single controller. And given N nodes, the maximum number of element cycles is $N/2$, because we need at least two nodes to construct a element cycle. If a network is not structurally controllable, then it is not controllable. If $N_D \geq 2$, and we know that $N_C \leq N$ at any time, thus $N_D/N_C \leq N/2$. Then we have proved that $N/2$ is the upper And use the same method, we can show that $\lceil \frac{N}{2} \rceil$ is the upper bound of N_C/N_D. So the upper bound of N_C/N_D is $N_C = \lceil \frac{N}{2} \rceil$.

Combine the results together, we get:

$$1 \leq N_D/N_C \leq \left\lceil \frac{N}{2} \right\rceil$$

B. *Lower and Upper bounds of N_C*

From the exact controllability theory, we know that $N_D = \max_i \mu(\lambda_i)$. In the last subsection, we have shown that $N_C/N_D \geq 1$. Take them together, we can conclude that $N_C \geq 1$.

To guarantee the fully controllability of $G(A)$, we need to ensure that for each eigenvalue of matrix A, the PBH rank condition is satisfied. To satisfy the PBH rank condition for eigenvalue λ_i, $\mu(\lambda_i)$ nodes must be directly controlled by controllers, where $\mu(\lambda_i)$ is the geometric multiplicity of eigenvalue λ_i. Therefore, the total number of nodes that need to be directly controlled could reach $\sum_{i=1}^{l} \mu(\lambda_i)$ at most. In other words, the upper bound of N_C is $\sum_{i=1}^{l} \mu(\lambda_i)$.

Combine these together, we have:

$$\max_i \mu(\lambda_i) \leq N_C \leq \sum_{i=1}^{l} \mu(\lambda_i)$$

V. Design of Minimum Set of Controlled Nodes

In this section, we will prove that the MSCN problem is a NP-hard problem. Then we give an exact algorithm to find out the exact minimum set of controlled nodes. And we propose an $O(n^4)$ approximation algorithm and a greedy algorithm of MSCN problem.

A. NP-hardness of MSCN Problem

We will reduce Set Cover Problem to MSCN to show that MSCN is a NP-hard problem. Set Cover Problem is a well-known NP-complete problem. The definition of Set Cover Problem is: Given a set of N elements $U = \{u_1, u_2, \cdots, u_N\}$ and a set of M subsets of U,$V = \{V_1, V_2, \cdots, V_M\}$, the objective is to find a collection C of the sets from V with the least size. That is, the number of sets in C is minimum.

We construct a matrix E, in which each row corresponds to an element in U and each column corresponds to a subset in V. Hence, E is an $M \times N$ matrix. If subset V_j contains the element u_i of U, let the element E_{ij} be 1, else let it be 0. Let $L = M + N$, then we construct an $L \times L$ identity matrix and put the elements of matrix E to the last M rows and left N columns. Denote the new matrix as P^{-1}. Because $|P^{-1}| = 1$, P^{-1} is an invertible matrix. Then we compute the inverse matrix P of P^{-1}. Let matrix

$$J = \begin{bmatrix} 1 & 1 & & & & & \\ & 1 & \ddots & & & & \\ & & \ddots & 1 & & & \\ & & & 1 & & & \\ & & & & 2 & & \\ & & & & & \ddots & \\ & & & & & & M \end{bmatrix}$$

where J is $L \times L$ and the last M rows have 1 on the diagonal. Let matrix $A = PJP^{-1}$.

If C is a minimum set of controlled nodes to fully control network $G(A)$, then we construct matrix B using method 1. Let matrix $Q = P^{-1}B$. From linear algebra we know it is to use elementary column transformation on P^{-1} to get matrix Q. If column j of B is nonzero, the column j of matrix P^{-1} is reserved, else it will be a zero column. By Jordan canonical form condition, if $G(A, B)$ is fully controllable, the last M rows of matrix Q must be all nonzero. If any node $i > N$ was chosen, replace it with node j if P_{ij}^{-1} is nonzero. We can ensure that such a node must exist, otherwise element u_{i-N} can never be covered by the subsets. Suppose C is changed into C^*, then the corresponding subsets of C^* in V can successfully cover the set U.

Hence, we successfully reduced Set Cover Problem into the MSCN problem and verified that MSCN problem is NP-hard.

B. Exact Algorithm of MSCN Problem

Then we propose an exact algorithm to find out the exact minimum set of control nodes of an arbitrary network.

From the discussion in the last section, we know that the time cost of Step.9 can reach exponential level, because the number of different maximal linear independent rows can be exponential. And in Alg.1, we need space to store the sets of indexes of maximal linear independent rows. Therefore, both the time complexity and space spatial complexity are exponential order. Then we show that Alg.1 can find the optimal solution of the MSC problem.

Algorithm 1 Exact algorithm of MSCN

```
1: **Input:** the coupling matrix of the network: $A$, the number of nodes in the network: $N$;
2: **Output:** $C = \{c_1, c_2, \cdots, c_m\}$;
3: $i := 1, m := 0, min := N, C := \emptyset$;
4: Compute all the eigenvalues of $A$: $\{\lambda_1, \lambda_2, \cdots, \lambda_l\}$;
5: Construct $l$ empty sets: $S_1, S_2, \cdots, S_l$;
6: **while** $i \leq l$ **do**
7:     Compute matrix $D = \lambda_i I_N - A$;
8:     Compute $r = rank(D)$;
9:     Find out all the combinations of linear independent rows in $D$. For each combination of linear independent rows, construct a set $V = \{v_1, v_2, \cdots, v_r\}$, where $v_i$ is index of the $i-th$ row of the combination in $D$. Compute the complement set $V^*$ of $V$ and add it to $S_i$;
10: **end while**
11: Find all the combinations of $\{s_1, s_2, \cdots, s_l\}$, where $s_i$ is an element of $S_i$. Compute $M = s_1 \cup s_2 \cup \cdots \cup s_l$; If $|U| < min$, $C := U, min := |U|$.
```

Theorem 1: Alg. 1 can find one of the exact minimum set of control nodes for an arbitrary network to be fully controlled.

Proof: First we prove the sufficiency. We know that a network $G(A, B)$ is controllable if and only if the PBH rank condition is satisfied. If we design a controlling matrix B to make all the rows which are pointed by the minimum set of control nodes linear independent. Imagine we construct an $N \times N$ matrix B, and for the $i-th$ element in the minimum control nodes set, we make the $i-th$ element in the $i-th$ row 1, and all the other elements zero. Then for every eigenvalue λ_i of matrix A, $rank(\lambda_i I_N - A, B) = N$ is satisfied. Because the linear independent rows in the matrix $\lambda_i I_N - A$ are still linear independent in the matrix $(\lambda_i I_N - A, B)$. And the left rows in matrix B are linear independent according to our design, so the left rows in the matrix $(\lambda_i I_N - A, B)$ are linear independent as well. Therefore, all the rows in the matrix $(\lambda_i I_N - A, B)$ are linear independent, so $rank(\lambda_i I_N - A, B) = N$ is satisfied for every eigenvalue. So the PBH rank condition is satisfied, and the network $G(A)$ is controllable.

Then we prove the necessity. If we make the number of control nodes less than the size of the minimum set of control nodes that Ala.1 finds. And if for every eigenvalue, the chosen set of control nodes contains one configuration of its control nodes. From the process of Alg.1, we know that the Alg.1 finds the minimum union of sets of control nodes for every eigenvalue. Therefore, we get a contradiction. So there doesn't exist such set of control nodes. And the necessity is proved.

Combine these together, the Theorem 1 has been proved. ■

C. Approximation algorithm

Although Alg.1 can find the optimal solution of the MSC problem, it is prohibited by time when the network has a big size. In this section, we propose an $O(n^4)$ approximation algorithm of MSC problem.

Algorithm 2 Approximation Algorithm of MSCN

```
1: Input: the coupling matrix of the network: $A$, the number of nodes in the network: $N$;
2: Output: the set of controlled nodes: $C = \{c_1, c_2, \cdots, c_m\}$;
3: $i := 1, m := 0, C := \emptyset$;
4: Compute all the eigenvalues of $A$: $\{\lambda_1, \lambda_2, \cdots, \lambda_l\}$;
5: while $i \leq l$ do
6:     Compute matrix $D = \lambda_i I_N - A$;
7:     Invoke Gauss-Jordan elimination to get column canonical form of matrix $D$.
8:     Find out an index set of maximal linear independent rows $V = \{v_1, v_2, \cdots, v_m\}$ in matrix $D$;
9:     Compute the complement set $V^*$ of $V$;
10:    $C := C \cup V^*$;
11: end while
```

Then we discuss the time and spatial complexity of Alg.2. It is easy to find that there are mainly matrix operations in Alg.2. Therefore, the spatial complexity of Alg.2 is $O(N^2)$. And the time complexity of the SVD algorithm is $O(N^3)$. Suppose A has l eigenvalues, then the while loop will be performed l times. And the time complexity of the algorithm of transforming a matrix into column canonical form is $O(N^3)$. So the time complexity of Alg.2 is $O(lN^3)$. And because $l \leq N$, the time complexity of Alg.2 is $O(N^4)$.

In some cases, Alg.2 can find the exact minimum set of control nodes. For example, when the matrix A has only one eigenvalue, then the minimum number of control nodes and driver nodes or controllers are the same. Or in another case, when A is a diagonal matrix, there must be N distinct eigenvalues of A. And we the minimum number of control nodes is N. In this scenario, Alg.2 will find N control nodes, equal to the minimum number of controlled nodes.

For an arbitrary network, in the worst case, Alg.2 will find $\sum_{i=1}^{l} \mu(\lambda_i)$ control nodes. Therefore, the approximate ratio of Alg.2 is $\sum_{i=1}^{l} \mu(\lambda_i)/\max_i \mu(\lambda_i) \leq N/1 = N$.

D. Greedy Approximation Algorithm

Because MSCN problem has some similarities with Set Cover Problem, we next design a greedy algorithm to solve MSCN problem.

VI. ACO BASED ALGORITHM

A. Ant Colony Optimization

Ant Colony Optimization (ACO) algorithmis a very efficient meta heuristic algorithm to solve combinatorial optimization problems such as TSP problem. ACO algorithm is originated from the behavior of the ants in the real word.

B. ACO based approximation algorithm

We next propose an approximation algorithm based on ACO. In this algorithm, we imitate the real ants' behavior

Algorithm 3 Greedy approximation algorithm of MSCN

```
1: Input: the coupling matrix of the network: $A$, the number of nodes in the network: $N$;
2: Output: the set of controlled nodes: $C = \{c_1, c_2, \cdots, c_m\}$;
3: Initialize an zero matrix $Q$.
4: Initialize empty array $W$ with length $N$.
5: Compute the Jordan canonical form $J = P^{-1}AP$ of $A$ and get the matrix $P^{-1}$
6: Find all the rows in $J$ corresponding to the last rows of every Jordan block and store them in the matrix $D = d_{ij}$.
7: for $1 \leq j \leq N$ do
8:     for every eigenvalue do
9:         If there exists nonzero element in the corresponding elements of eigenvalue $\lambda_i$ in column vector $d_j$, add W[j] by 1.
10:    end for
11: end for
12: while there exists nonzero elements in $W$ do
13:    Find the maximum element in $W$, denote its index as $q$.
14:    Add column vector $D_q$ to $Q$.
15:    Let $W[q]$ be 0.
16:    for every nonzero element in W do
17:        Denote the index as $j$.
18:        for every eigenvalue do
19:            If the corresponding subvector in $D_j$ is linear independent to the corresponding subvectors in $Q$, W[j]=W[j] - 1.
20:        end for
21:    end for
22: end while
```

of searching food to solve MSCN problem. Ant k selects a node by probability P. Denote D as the current solution set.

$$P_j^k = \begin{cases} argmax|\tau_j^\alpha||\eta_j^\beta| & ifq \leq q_0 \\ \frac{\tau_j^\alpha * \eta_j^\beta}{\sum \tau_j^\alpha * \eta_j^\beta} & else \end{cases} \tag{6}$$

Heuristic information η_j is the probability that node j is in the optimal solution set. Here we let $\eta_j = \frac{W[j]}{N_\lambda}$, where $W[j]$ is defined in Alg.3 and N_λ denotes the number of distinct eigenvalues. Pheromone τ_j is the core of ACO algorithm. Pheromone is updated by the following mode:

$$\tau_j(t+1) = (1-p)\tau_j(t) + \sum_M \Delta\tau_j^k \tag{7}$$

where

$$\Delta\tau_j^k = \begin{cases} Q/N & ifantkchosenodej \\ 0 & else \end{cases} \tag{8}$$

Q is a constant and q is the evaporation rate of phermone, $0 \leq q \leq 1$.

To enhance the performance of the algorithm, we use MMAS method to control the amount of phermone to prevent early stagnation:

$$\tau_i = \begin{cases} \tau_{max} & if \tau_i > \tau_{max} \\ \tau_{min} & if \tau_i < \tau_{min} \end{cases} \tag{9}$$

If the ACO algorithm can't get a better solution after many iterations, then

$$\tau_i = \begin{cases} \tau_{min} & if \tau_i = \tau_{max} \\ \tau_{max} & if \tau_i = \tau_{min} \end{cases} \tag{10}$$

C. ACO based Approximation Algorithm

Because MSCN problem has some similarities with Set Cover Problem, we next design a greedy algorithm to solve MSCN problem.

Algorithm 4 ACO based approximation algorithm of MSCN

```
1: Initialize empty array $W$ with length $N$
2: Initialize a zero matrix $Q$
3: Compute the Jordan canonical form $J = P^{-1}AP$ of $A$ and get the matrix $P^{-1}$
4: Find all the rows in $J$ corresponding to the last rows of every Jordan block and store them in the matrix $D = d_{ij}$.

5: for $1 \le j \le N$ do
6:     for every eigenvalue do
7:         If there exists nonzero element in the corresponding elements of eigenvalue $\lambda_i$ in column vector $d_j$, add W[j] by 1.
8:     end for
9: end for
10: while the optimal condition is not satisfied or the iteration number is less than some given number do
11:     Assign the $m$ ants uniformly to the $N$ nodes.
12:     Initialize boolean variable $b$ = ture.
13:     for k=1 to m do
14:         Initialize set $X = 1, 2, \cdots, N$
15:         while b do
16:             Choose a node $j$ from $X$ according to (6)
17:             $D = D \cup j$
18:             $Q = Q \cup d_j$
19:             $X = X - j$
20:             If the rank of all the corresponding parts of $Q$ are equal the number of Jordan blocks of every eigenvalue,$b = flase$.
21:         end while
22:         Record the solution found by ant $k$ and its size.
23:     end for
24:     Record the best solution of the $m$ ants.
25:     if The solution hasn't been updated for $n$ iterations then
26:         Change phermone according to (10).
27:     end if
28:     Change phermone according to (7),(8),(9)
29: end while
```

VII. Simulation

VIII. Conclusion

IX. Acknowledgements

References

[1] M. L. J Hautus. Controllability and observability conditions of linear autonomous systems. *Nederl.akad.wet.proc.ser.a*, 72(5):443–448, 1969.

[2] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *Nature*, 473(7346):167–173, 2011.