Clever Ant Colony System for Key Covering Problem in the Group Rekeying

Sunyan Hong¹*, Jianhua Adu¹, Yunfei Zhang¹

¹College of Information Engineering, Kunming University, Kunming, Yunnan 650214, China
*hongsunyan@kmu.edu.cn

Abstract. The key covering problem (KCP) in the group rekeying for secure multicast communication is NP-Hard. We try to introduce the ant colony optimization (ACO) idea to solve the KCP, but the KCP can’t be solved directly by the existing ant algorithms. So we design a new kind of heuristic algorithm, called clever ant colony system (CACS), for the KCP. Firstly, we establish a map from a secure multicast group to an auxiliary key graph (AKG). Secondly, according to the idea of the ant colony system (ACS), we design the CACS over AKG. Finally, using simulation, we verify that the CACS is feasible and keeps high performance with a very short time.

1. Introduction

1.1. Ant colony optimization (ACO)
Ant colony optimization was presented by Dorigo et al. [1,2] as a multi-agent approach to hard combinatorial optimization problem. Recently, it is widely applied to the subject fields such as, vehicle routing, sequential ordering, routing in communication networks [3,4].

The idea of ant colony optimization is as follows: The ants in the animal world can find the shortest path from a food source to the nest by depositing pheromone. Ants drop pheromone while walking to find food, and use it to exchange information with each other. Pheromone is a kind of secretion that can be recognized by other ants and can influence the behavior of ants. Ants distinguish pheromone and, when choosing their way, tend to choose, in probability, the path marked by strong pheromone concentrations. Due to the pheromone trail, ants can establish and maintain the shortest path. There will be more the pheromone when many ants select the path, so that the density of pheromone will be bigger and the probability of selecting the path for ants will be higher. The more the ants selecting a path, the more attractive that path becomes for being selected. The process is characterized by a positive feedback loop. As a result, all the ants will choose the shorter path finally.

Following the ACO idea many kinds of ant-like algorithms are presented. Among them, ant colony system (ACS) outperforms the others. So we design our algorithm CACS based on ACS in section 2. The details of ACS are described in the listed literature[3-4].

1.2. Secure multicast group
To expound key covering problem, we recall basic definitions in secure multicast group [5,6]. A secure multicast group is a triple \((U, K, R)\), where \(U\) is a finite and nonempty set of users, \(K\) is a finite and nonempty set of keys, \(R\) is a binary relation between \(U\) and \(K\), \(R \subseteq U \times K\), called the
user-key relation of the secure multicast group. User \( u \) has key \( k \) if and only if \( (u,k) \in R \).

In a secure multicast group, each user \( u \) has a group key \( k_G \) and an individual key \( I_u \). The \( k_G \) is shared by all the users in \( U \). The \( I_u \) is used by the user \( u \) and the key server. And there are several auxiliary keys \( K_a \) in the secure multicast group, which are used by the key server to facilitate rekeying. To communicate with each other the members in the secure multicast group use the group key \( k_G \) to encrypt message.

For secure group communication, the key server performs the key management functions, among them the critical design issue is the group rekeying, i.e. the updating of group key and auxiliary keys. When rekeying, the key server creates a new key firstly, which can be a substitute for old key (group key or auxiliary key). Then the key server selects the auxiliary keys that are not leaked, to encrypt the new key. Those keys serve as key encryption keys (\( KEKs \)). Therefore the critical issue of rekeying is the selection of \( KEKs \).

Two functions, keyset () and userset (), are defined as follows for each secure multicast group \( (U,K,R) \):

\[
\forall u \in U, \text{keyset}(u) = \{k|(u,k) \in R\}, \quad \forall S \subset U, \text{keyset}(S) = \bigcup_{u \in S} \text{keyset}(u).
\]

\[
\forall k \in K, \text{userset}(k) = \{u|(u,k) \in R\}, \quad \forall M \subset K, \text{userset}(M) = \bigcup_{k \in M} \text{userset}(k).
\]

Obviously, keyset(\( u \)) is the set of keys that are held by user \( u \) in \( U \), and userset(\( k \)) is the set of users that hold key \( k \) in \( K \).

1.3. Key Covering Problem

In secure multicast group, when a user leaves a secure multicast group \( (U,K,R) \), the keys that have been held by \( u \) and shared by other users in \( U \) should be changed. Let \( k \) be such a key. To update \( k \), the key server randomly generates a new key \( k_{\text{new}} \) and sends it to every user in userset(\( k \)) except \( u \). To secure the process, the key server needs to find a subset \( K' \) of keys such that userset(\( K' \)) = userset(\( k \)) \( \setminus \{u\} \), and use keys in \( K' \) to encrypt \( k_{\text{new}} \) for distribution. To minimize communication overload, the key server would like to find a minimal size set \( K' \). The processing of rekeying is generalized as follows: Given a secure multicast group \( (U,K,R) \), and a subset \( S \) of \( U \), find a minimal size subset \( K' \) of \( K \) such that userset(\( K' \)) = \( S \).

When a user \( u \) leaves a secure multicast group \( (U,K,R) \), to update the key \( k \), let \( U_u = \text{userset}(k) \setminus \{u\} \), \( K_u = \text{keyset}(U_u) \setminus \text{keyset}(u) \). So our goal is to find a minimal size subset \( K' \) of \( K_u \), then use the keys in \( K' \) to encrypt \( k_{\text{new}} \). \( K' \) is the set of keys that are not leaked, so the keys in \( K' \) can be regarded as \( KEKs \) to encrypt the key required to update. The \( U_u \) and \( K_u \) can be obtained by computing, therefore the discussion of rekeying can be limited in the new secure multicast group \( (U_u,K_u,R_u) \) \( (R_u \subset U_u \times K_u) \). Now we present the definition of key covering set and key covering problem (KCP).

Definition of key covering set: Given a secure multicast group \( (U,K,R) \), \( K' \in K \) is the key covering set of \( (U,K,R) \), if and only if userset(\( K' \)) = \( U \).

Definition of KCP: Given a secure multicast group \( (U,K,R) \), find a minimal size key covering
set of \((U, K, R)\). (As a matter of fact, \((U, K, R)\) in the definition is just the secure multicast group \((U_u, K_u, R_u)\) that is discussed in the above paragraphs.)

When the user join the secure multicast group, the KCP can be easily modeled just as the user leave it. So the KCP is a core problem in the group rekeying, especially in large-scale VSS[8]. Unfortunately, the KCP in general is NP-hard [5-7]. KCP is a core problem in large-scale VSS.

2. The heuristic algorithm for KCP

2.1. The structure of heuristic algorithm for KCP

It is difficult to give the heuristic algorithm for KCP directly. We divide the heuristic algorithm into three portions.

The first portion (Algorithm 1) is simple. It deals with the special secure multicast group \((U, K, R)\), where \(K\) is the set of individual keys. The input of Algorithm 1 is the special secure multicast group \((U, K, R)\), and its output (the key covering set) is just \(K\).

Algorithm 1. Simple algorithm for special secure multicast group.
Input: \(U\), \(K\), and \(R\)
Output: \(K\)

The second portion (Algorithm 2), called clever ant colony system (CACS) for KCP, is pivotal. It is designed for searching the key covering set \(C\) of normal secure multicast group, in which each user holds two keys (the individual key and some auxiliary keys) at least. We will describe the details of Algorithm 2 in subsection 2.2.

Algorithm 2. Clever ant colony system for KCP in normal secure multicast group.
Input: \(U\), \(K\), and \(R\)
Output: \(C\)

The last portion (Algorithm 3) is an integration of Algorithm 1 and Algorithm 2. Algorithm 3 handles KCP in any secure multicast group. Firstly, find all the users who only hold individual key. The users and their individual keys formed the special secure multicast group, use Algorithm 1 to find the key covering set \(C_1\) of the special secure multicast group. Secondly, use Algorithm 2 to find the key covering set \(C_2\) of the normal secure multicast group, which is formed by the rest users and keys.

Finally, output \(C_1 \cup C_2\).

Algorithm 3. The heuristic algorithm for KCP.
Input: \(U\), \(K\), and \(R\)
\(U_1\) : =All the users who only hold individual key
\(K_1\) : =keyset\((U_1)\)
\(C_1\) : =the key covering set of \((U_1, K_1, R_1)\) found by Algorithm 1
\(U_2\) : =\(U \setminus U_1\)
\(K_2\) : =\(K \setminus K_1\)
\(C_2\) : =the key covering set of \((U_2, K_2, R_2)\) found by Algorithm 2
Output: \(C_1 \cup C_2\)

2.2. Clever Ant colony system (CACS)

Firstly, we construct a map \(f : (U, K, R) \rightarrow (V, E; w)\). \(f\) maps the secure multicast group \((U, K, R)\) to the auxiliary key graph \(G_u = (V, E; w)\), and satisfies the following conditions:

(1). \(G = (V, E)\) is a complete graph.
(2). There is a one-to-one correspondence between \( k \) in \( K \) and the set of \( v \) in \( V \), denotes as \( v \leftrightarrow k \)

(3). \( w(e=(v_1,v_2)) = \begin{cases} 1 & v_1,v_2 \in K, v_1 \leftrightarrow k_1, v_2 \leftrightarrow k_2, \text{userset}(k_1) \cap \text{userset}(k_2) \neq \emptyset \\ 0 & \text{Other} \end{cases} \)

Having established the map \( f \), we design the clever ant colony system (CACS) over the auxiliary key graph (AKG). The CACS is based on the ant colony system(ACS). Informally, in CACS, we let a set of agents, called ants, searching parallel in the auxiliary key graph \( G_a \) for heuristic solutions to KCP. The ants cooperate through pheromone-mediated indirectly and global communication. The ant deposits pheromone on the vertexes of \( G_a \) (Note: The ant in ACS[3] deposit the pheromone on the path). When one ant arrives at the vertex \( v \in V \), let \( d(e) = \sum_{u \in V} w((v,u)) \), it deposits the pheromone that is in proportion to \( d(e) \). Upon leaving the vertex \( v \in V \), it set \( w(e)=0 (e=(v,u), \forall u \in V) \). When \( w(e)=0 (\forall e \in E) \), we say that the ant complete one whole tour. The visited vertex set is similar to the visited edge set in ACS for TSP[3]. So the visited vertex set makes up of the ant’s path in CACS (Note: In ACS for TSP, the ant agent completes a tour, which means it travels all the cities. But the ant in CACS completes a tour, which doesn’t mean it visits all the vertexes). The size of the visited vertex set is the length of the ant’s path. To sum up, the CACS differs from ACS in two main aspects: (1).The path on which the ant agent travels in CACS is not an edge sequence but a vertex set. (2). The pheromone is deposited on the vertex.

The CACS works as follows: \( m \) ants are initially posited on \( n \) nodes according to some initialization rule. Each ant chooses the next node by itself according to the state transition rule. While the ant is in the tour, it modifies the amount of pheromone on the visited nodes by applying the local updating rule. Once all ants have terminated their tour, the pheromone is modified again by applying the global updating rule. The ants are guided by heuristic information and the pheromone. The pheromone updating rule are designed so that they tend to give more pheromone to the nodes that should be visited by ants.

In the following we discuss the state transition rule, the global updating rule, and the local updating rule.

The state transition rule:

The state transition rule used by CACS is given by Eq.(1), which gives the probability with which ant \( k \) choose to move to the node \( j \).

\[
p_j^k = \begin{cases} \frac{1}{\text{num}} , & q < q_0 , j \in J \\ 0 , & q \geq q_0 \end{cases} \quad (1)
\]

Where \( j \) is the pheromone, \( \eta_j = d(e) = \sum_{u \in V} w((v,u)) \), \( A_k \) is the set of nodes that remain to be visited by ants, \( J = \{ r | r \eta_j^k = \max_{\eta_{j,r}} \{ r \eta_j^k , r \in A_k \} \} \) (Namely, \( J \) is the set of the best next hops), \( \text{num} = |J| \). Generally, \( \text{num}=1 \). \( \beta \) is a parameter that determines the relative importance of pheromone \( (\beta > 0) \), \( q \) is a random number uniformly in \( [0..1] \), \( q_0 \) is a parameter \( (0 \leq q_0 \leq 1) \).
The global updating rule:
In CACS, the ant, which constructs the best tour, is allowed to deposit pheromone. Global updating is performed after all ants have completed their tours. The pheromone level is updated by applying the global updating rule of \( Eq.(2) \).

\[
\tau_j = (1 - \alpha) \tau_j + \alpha \Delta \tau_j \quad (2)
\]

Where \( \Delta \tau_j = \left( \frac{(L)_{gb}}{l} \right)^{-1} \) if \( j \in \text{global-best-tour} \)

\[
0 \quad \text{otherwise}
\]

Where \( 0 < \alpha < 1 \) is the pheromone decay parameter, and \( L_{gb} \) is the size of the visited vertex set. \( Eq.(2) \) dictates that only the pheromone on those nodes belong to the best tour will increase.

The local updating rule:
While searching a solution to the KCP, ants visit nodes and change their pheromone level by the local updating rule of \( Eq.(3) \).

\[
\tau_j = (1 - \rho) \tau_j + \rho \Delta \tau_j \quad (3)
\]

Where \( 0 < \rho < 1 \) is a parameter, \( \Delta \tau_j = \tau_j - (mL_m^{-1}) \), where \( L_m \) is the very rough approximation of the size of key covering set.

**Theorem:** In CACS, Let \( f \) maps the secure multicast group \((U, K, R)\) to the auxiliary key graph \( G_a = (V, E; w) \). Let \( V_c \) is the set of nodes that one ant agent visited in one whole tour, and \( K_c \subset K, f(V_c) = K_c \). Then \( K_c \) is \((U, K, R)\)'s key covering set.

**Proof:** \( \forall u \in U, \exists l_{k_i}, k_i \in K_c \), s.t. \( u \in \text{userset}(k_i) \cap \text{userset}(k_j) \neq \emptyset \) (Algorithm2 premise any user \( u \) holds two keys at least).

According to the definition of the map \( f \), \( \exists v_1, v_2 \in V : v_1 \leftrightarrow k_i, v_2 \leftrightarrow k_j \), s.t. \( w((v_1, v_2)) = 1 \). The \( V_c \) is the set of nodes that the ant visited in its tour, so \( v_1 \in V_c \) or \( v_2 \in V_c \) (Because the ant should set \( w(e)(\forall e \in E) \) to 0 to complete its tour). We suppose that \( v_1 \in V_c \), and \( k_i \leftrightarrow v_1 \), it follows that \( k_i \in K_c \), because of \( V_c = f(K_c) \).

This proves that \( \forall u \in U, \exists k_i \in K_c, \text{s.t.} u \in \text{userset}(k_i) \), namely \( K_c \) is \((U, K, R)\)'s key covering set.

**3. Simulation and result**
We model the CACS of KCP using Java, and use CACS to search the solution to the KCP. The numeric parameters are as follows: \( \beta = 0.25 \), \( q_0 = 0.9 \), \( \alpha = 0.1(\alpha = 0.2) \), \( \rho = 0.1(\rho = 0.2) \), \( \tau_1 = 1/(nL_m) \), where we set \( L_m = 2/5|K| \), \( |U|/|K| = 0.6 \). Table 1 and 2 show the simulation experiment results of CACS.

| The Size of Key Set | The Number of ants | Key Cover | Time (sec) | \( \alpha \) | \( \rho \) |
|---------------------|-------------------|-----------|------------|-----------|---------|
| Best               | Worst             | Average   |            |           |         |
| 256                | 4                 | 104       | 117        | 111.15    | 1.820   | 0.2     | 0.1     |
| 256                | 8                 | 105       | 114        | 109.92    | 3.417   | 0.2     | 0.1     |
| 256                | 16                | 105       | 113        | 109.79    | 6.752   | 0.2     | 0.1     |
Table 2. The experiments with different size of secure multicast groups
(The platform: CPU:1.0GHz; RAM:256M; OS:WinXP; Java:JDK 1.4.2)

| Size of Key Set | The Number of ants | Key Cover | Time (sec) | α | ρ |
|-----------------|--------------------|----------|------------|---|---|
|                 |                    |          | Best       | Worst | Average |          |       |     |
| 200             | 32                 | 80       | 86         | 84.47 | 5.003   | 0.1      | 0.1   |
| 256             | 32                 | 105      | 110        | 107.92| 14.098  | 0.1      | 0.1   |
| 300             | 32                 | 121      | 129        | 125.88| 24.046  | 0.1      | 0.1   |

Table 1 shows that the CACS with α=0.1 ρ=0.1 get best average key-cover size. Table 2 shows that the CACS can get good result for different size of secure multicast groups.

4. Conclusion

ACO is widely used to solve the hard combinatorial optimization problems. We use the idea of ACS to design the new algorithm CACS to solve the key covering problem in the group rekeying. Performance simulation results show that the CACS is a feasible heuristic algorithm for KCP.

Acknowledgments

This work was supported by a grant from SCIENCE FUND OF DEPARTMENT OF EDUCATION IN YUNNAN PROVINCE( No.2018JS391).The authors would like to thank Professor Jun He of Kunming University for the revision and suggestion of the paper.

References

[1].Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. Proc. of the 1st Euro. Conf. on A rtificial Life, France: Elsevier Publishing, 1991. 134~ 142.
[2].Colorni A , Dorigo M ,Maniezzo V. An investigation of some properties of an ant algorithm. Proc. of the Parallel Problem Solving from Nature Conference (PPSN ’92) ,Belgium: Elsevier Publishing, 1992. 509~ 520
[3].Dorigo and L.M.Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1):53-66,1997.
[4] Marco Dorigo and Gianni Di Caro. Ant Algorithms for Discrete Optimization. In Artifical Life, MIT Press, 1999.
[5].C.K.Wong, M.Gouda and S.Lam. Secure Group Communication Using Key Graphs. In Proceedings of ACM SIGCOMM’98, volume28, number 4,1998.
[6] C.K.Wong, M.Gouda and S.Lam. Secure Group Communication Using Key Graphs[J]. IEEE/ACM Trans on Networking. 2000,8(1):16-30.
[7] Zhengfu Lu, Sunyan Hong. On the Proof of NP-Completeness of Key Covering Problem in Group Rekeying. Journal of Yunnan University(Natural Sciences Edition), Vol.28, No.3, 2006(In Chinese, with English abstract)
[8] Zheng-Fu Lu ,  Fei Du. Optimization for large-scale VSS based on secure multicast. Journal of Yunnan University(Natural Sciences Edition), 2012.