Skeleton Construction in Mobile Social Networks: Algorithms and Applications

Zongqing Lu[†], Xiao Sun[†], Yonggang Wen[‡] and Guohong Cao[†]
[†]The Pennsylvania State University, [‡]Nanyang Technological University
[†]{zongqing,xxs118,gcao}@cse.psu.edu, [‡]ygwen@ntu.edu.sg

***Abstract*—Mobile social networks have emerged as a new frontier in the mobile computing research society, and the commonly used social structure (i.e., community) has been exploited to facilitate the design of network protocols and applications, such as data forwarding and worm containment. However, community based approaches may not be accurate when applied for predicting node contacts and may separate two frequently contacted nodes into different communities. In this paper, to address these problems, we propose *skeleton*, a tree structure specially designed for organizing network nodes, as the underlying structure in mobile social networks. We address the challenges on how to uncover skeleton from network, how to adapt skeleton with dynamic network and how to leverage skeleton for network protocol designs. Skeleton is constructed based on best friendship and skeleton construction is simple and efficient (e.g., less computational complexity than community detection). Algorithms are also designed to adapt skeleton construction to dynamic network. Moreover, a data forwarding algorithm and a worm containment strategy are designed based on skeleton. Trace-driven simulation results show that the skeleton based data forwarding algorithm and worm containment strategy outperform existing schemes based on community.**

I. Introduction

With the proliferation of smart devices and the merging of online social networks that link humans, computers and the Internet, mobile social networks have emerged as a new frontier in the mobile computing research society [1][2][3][4][5]. A current trend for online social networking, such as Facebook and Twitter, is to develop mobile apps to provide users instant access through their mobile devices. In addition, more and more mobile social networks have been created like Foursquare, Instagram, and Path. Smart devices equipped with Bluetooth or Wi-Fi Direct bring numerous ad-hoc communication opportunities based on user mobility and social contacts. Social relations are deeply rooted in such mobile social networks.

It is a challenge to identify the network structure (i.e., how nodes are organized) in mobile social networks. Since the network structure represents a long-term property of the network, it can facilitate the design of network protocols. For example, better data forwarding can be achieved by locating nodes through the network structure and then accurately predicting future node contacts; worm containment can be achieved by disconnecting some nodes based on the network structure. However, due to the huge amount of node contacts, mobile social networks are extremely complex and it is a challenge to identify the underlying structure.

Community has been widely used as the underlying structure of social networks, where nodes inside the community have more internal connections than external connections [6][7][8][9]. Although community-based data forwarding and worm containment [10][11][12][13] outperform other social based schemes [14][15][16][17][18][19], community based approaches suffer from two major problems: the result may not be accurate when applied for predicting future node contacts; two frequently contacted nodes may be separated into different communities.

To address these problems, in this paper, we propose *skeleton*, a tree structure specially designed for organizing network nodes, as the underlying structure in mobile social networks. Skeleton is simple and easy to be uncovered, and we propose a lightweight algorithm to construct skeleton based on best friendship. With skeleton, nodes can be easily connected or separated, which better facilitates protocol design. The contribution of the paper is two-fold and summarized as follows:

- We propose skeleton as the network structure in mobile social networks. We design a lightweight algorithm for skeleton construction, which has less computational complexity than that of community detection. We also design algorithms to adapt skeleton construction to dynamic networks.
- We exploit skeleton for designing a data forwarding algorithm and a worm containment strategy in mobile social networks. Trace-driven simulation results show that skeleton based schemes outperform community based schemes.

The rest of this paper is organized as follows. Section II gives an overview of this work. Section III presents the skeleton construction. Section IV shows the skeleton evolution with dynamic network. The skeleton based applications are presented in Section V, followed by performance evaluations in Section VI. Section VII reviews related work and Section VIII concludes the paper.

II. Overview

A. Motivation

In mobile social networks, node contacts can be categorized into two types: *frequent contacts* and *intermittent contacts*.

This work was supported in part by Network Science CTA under grant W911NF-09-2-0053.

For community detection, every contact is taken into consideration for uncovering communities. This is specially true for community detection in binary networks [10], where frequent contacts and intermittent contacts are treated equally as edges in the network. For community detection in weighted networks [12], although frequent contacts and intermittent contacts are differentiated by edge weights, intermittent contacts are still important criteria for nodes to form communities. However, since intermittent contact between two nodes happens occasionally, we cannot rely on it to predict node contacts in the future. Nevertheless, such prediction is essential for designing network protocols in mobile social networks.

In community detection, since the criterion for a node to be included in a community is the connections between the node and the existing community rather than the connections between nodes, community structure might separate two frequently connected nodes into different communities and hence undermines its usefulness for protocol designs. Let us give an example. In a network, there are two neighboring nodes, u and v, where node v contacts with u more frequently than with other nodes; however, u and v are separated into community A and B, respectively. According to data forwarding schemes based on community, to forward a message to v from community A, the message is firstly forwarded into community B, then the message is continuously relayed within B before received by node v. However, obviously, the forwarding through node u from A will be more efficient to reach node v. Similarly, for worm containment, the separation of frequently connected nodes will make worms spread quickly between communities if they are infected.

To address these problems, we propose *skeleton*, a tree structure specially designed for organizing network nodes based on best friendship (i.e., one contacts with one's best friend more frequently than with others). The skeleton tree is structured by hierarchically grouping nodes or groups connected by best friendships. Since skeleton is constructed based on the most frequent and reliable connections, it is more accurate and easier to predict future node contacts based on skeleton than community. Without loss of generality, we develop data forwarding algorithm and worm containment strategy based on skeleton to demonstrate how skeleton better facilitates protocol designs.

TABLE I: MIT Reality trace summary

Trace	MIT Reality
No. of nodes	97
No. of contacts	114046
Duration (days)	246
Pairwise contact frequency (per day)	0.10

TABLE II: Facebook trace summary

Trace	Facebook
No. of nodes	18559
No. of contacts	296002
Duration (days)	365
Pairwise contact frequency (per day)	0.002

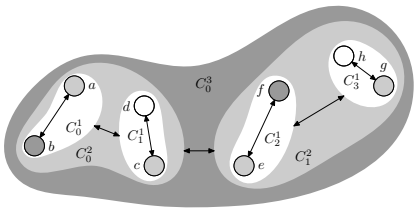


Fig. 1: Illustration of skeleton, where arrow lines represent best friendships between nodes or between groups, and the arrow points from a node or group to its best friend.

B. Traces

In this paper, we study skeleton construction and evaluate the proposed protocols based on two traces in mobile social networks: the MIT Reality trace [20] and the Facebook trace [21].

The MIT Reality trace contains contacts among users carrying Bluetooth devices. Bluetooth devices periodically discover the peers in the neighborhood and record their contacts when two devices move close to each other. The detail of the trace is summarized in Table I.

The Facebook trace contains the friendship information and the wall posts among Facebook users in the New Orleans regional network for more than four years. In this paper, we choose a partial trace which spans one year from 1/1/2007 to 1/1/2008. The Facebook trace is summarized in Table II, where contact between two nodes is the wall post.

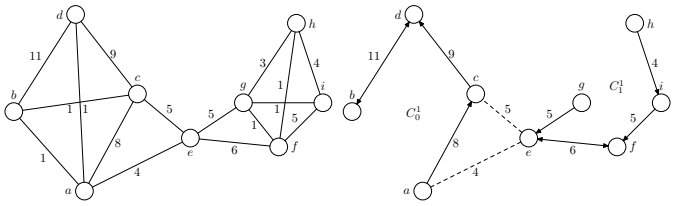
III. Skeleton Construction

In this section, we address the following problem: *How to construct the skeleton tree based on best friendships?*

A. Preliminary

Let $G = (V, E)$ represent a weighted and undirected network, where V denotes the set of nodes and E denotes the set of edges. For two neighboring nodes $u, v \in V$, w_{uv} denotes the weight of the edge. The edge weight indicates the contact frequency between two nodes and the time unit of frequency is consistent for all the edges. We denote C as a *group* (a set of nodes), we also denote $\mathcal{C}$ as a set of groups, e.g., $\mathcal{C} = \{C_0, C_1, C_2...\}$.

The skeleton tree is denoted by S, $S = \{\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_l\}$. $\mathcal{C}_i$ denotes the set of groups at level i of S and C_j^i denote a group in $\mathcal{C}_i$. The bottom level of the skeleton tree (i.e., $\mathcal{C}_0$) consists of individual nodes and we also denote each node as a group in the skeleton tree (e.g., C_m^0) for convenience. For any group at the level above the bottom level, e.g. C_j^i and $i > 0$, it has a subgroup set $\{C_m^{i-1}, C_n^{i-1}, C_o^{i-1}, ...\}$. Moveover, there is only one group in $\mathcal{C}_l$ and the group is called the root of S. We also denote $S_{C_j^i}$ as a subtree of S, where the root of $S_{C_j^i}$ is C_j^i and $\mathcal{C}_{C_j^i}^k$ denotes the set of groups at level k of subtree $S_{C_j^i}$. For any given set of groups at the same level of the skeleton tree $\mathcal{C} = \{C_m^i, C_n^i, C_o^i, ...\}$, we denote $\theta_{\mathcal{C}}$ as the lowest common group (ancestor) in S, where $\mathcal{C}_{\theta_{\mathcal{C}}}^i$ of subtree $S_{\theta_{\mathcal{C}}}$ includes all the groups in $\mathcal{C}$.



(a) weighted network (b) extracted best friendships

Fig. 2: Best friendships extraction

Let us give an example using Fig. 1, where $S = \{\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$. We have $\mathcal{C}_0 = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}, \{h\}\}$, where we also denote $\{a\}$ as C_0^0, $\{b\}$ as C_1^0, $\{c\}$ as C_2^0, etc. We also have $\mathcal{C}_1 = \{C_0^1, C_1^1, C_2^1, C_3^1\}$, $\mathcal{C}_2 = \{C_0^2, C_1^2\}$, and $\mathcal{C}_3 = \{C_0^3\}$, where C_0^3 is the root of S. For $\mathcal{C} = \{\{a\}, \{c\}\}$, the lowest common group of $\mathcal{C}$ is C_0^2. Note that the notation of the skeleton tree is convenient for algorithm designs in the rest of the paper.

B. Best Friendship Stability

In mobile social networks, nodes opportunistically contact with each other, however two nodes connected by best friendship contact each other frequently and regularly. For a node u, $u \in V$, the best friend of u is the node that u contacts with more frequently than with other neighboring nodes, it is denoted by β_u. For node u and v, where $v = \beta_u$, $u \to v$ also denotes that the best friend of u is v. For example, in Fig. 2a, we have $a \to c$, $b \to d$, $d \to b$, etc. As best friendship in social network is uni-directional, in the network model, best friendship is also uni-directional that is if $u \to v$, there is not necessarily $v \to u$.

We use the MIT Reality trace and the Facebook trace to show the stability of best friendships. We set 20 and 50 observation points for the MIT Reality trace and the Facebook trace. That is, the traces are divided by 20 and 50 based on the time span of traces. At each observation point, we calculate the stability ratio of best friendships as

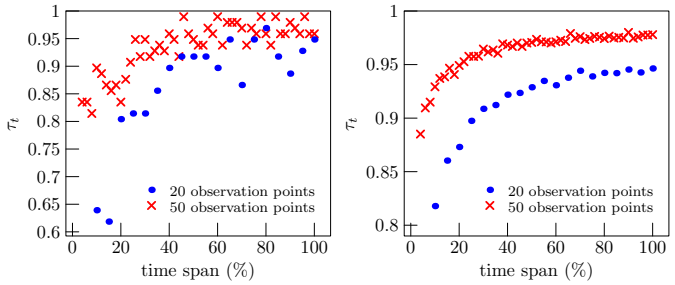
$$\tau_t = \frac{\sum_{u \in V} \lambda(\beta_u^t, \beta_u^{t-1})}{|V|},$$

where λ-function yields one when the best friend of node u at observation point t (β_u^t) is the same with that at $t-1$ (β_u^{t-1}); otherwise, it is zero.

As shown in Fig. 3, after initial few observation points, the stability ratio is more than 80% and 90% in the MIT Reality trace and the Facebook trace, respectively, for both 20 and 50 observation points. Therefore, best friendships between nodes are much stable and hence we can rely on them to build the skeleton tree.

C. The Skeleton Construction Algorithm

To construct the skeleton tree, best friendships between nodes are extracted and then the nodes connected by best friendships form a group. For example, in Fig. 2a where $a \to c$, $c \to d$, $b \to d$ and $d \to b$, a, b, c and d will form a group. Note that each node has one and only one best friend.



(a) MIT Reality trace (b) Facebook trace

Fig. 3: Stability ratio of best friendships in the MIT Reality trace and the Facebook trace.

For a group, it at least includes two nodes which are the best friend of each other.

Fig. 2b shows the extraction of best friendships between nodes for a given weighted network in Fig. 2a. After the extraction, the network is partitioned into $C_0^1 = \{a, b, c, d\}$ and $C_1^1 = \{e, f, g, h, i\}$, and within each group nodes are connected by best friendships, i.e., arrow lines in Fig. 2b, where the arrow points from a node to its best friend.

Best friendships connect individual nodes in the skeleton tree, and similarly, we also identify best friendship between groups to connect groups in the skeleton tree. Considering connections between groups as the sum of the edge weights between individual nodes in two groups, for a group C_i, the best friend of C_i is the group that C_i has more connections with than with other groups, and this group is denoted by β_{C_i}. $C_i \to C_j$ also denotes the best friend of C_i is C_j.

Based on best friendships between nodes and between groups, the skeleton tree is constructed as follows. Starting with individual nodes at the bottom level of the skeleton tree, nodes connected by best friendships form a group. This partitions the network into one or more groups, which form the next level up of the skeleton tree. Then, best friendships between groups are identified and similarly groups connected by best friendships are grouped together. The process is iterated until there is only one group. The pseudo code of the skeleton construction algorithm is detailed in Algorithm 1.

Unlike best friendship between nodes, e.g., $u \to v$, best friendship between groups is built on the sum of the edge weights between two groups. To bridge two groups connected by best friendship, we define a pair of *joint nodes*. For $C_i \to C_j$, considering connections between a node and a group as the sum of the edge weights between the node and nodes in the group, the joint node that connects C_j from C_i is the node in C_i, which has more connections with C_j than other nodes in C_i. Similarly, we have the joint node that connects C_i from C_j. For example, in Fig. 2b, where the connections between two groups are the dashed lines, the joint nodes are node c and e.

Skeleton is eventually built as a tree. Let us use Fig. 1 as an example of skeleton. a and b form C_0^1, c and d form C_1^1, e and f form C_2^1, g and h form C_3^1. Further, C_0^1 and C_1^1 form C_0^2,

Algorithm 1: Skeleton construction

Input : $G = (V, E)$
Output: S

$\mathcal{C} = \{\{u\} : u \in V\}$;
while $|\mathcal{C}| \neq 1$ **do**
  $\mathcal{C}$ =`Skeleton(`$\mathcal{C}$`)`;
  $S = S \cup \{\mathcal{C}\}$;
end

`Skeleton(`$\mathcal{C}$`)`
begin
  $\mathcal{C}' = \emptyset$;
  while $\mathcal{C} \neq \emptyset$ **do**
    $\mathcal{C}'' = \{C_i\}$, $C_i \in \mathcal{C}$;
    $C_j = \beta_{C_i}$;
    while $C_j \notin \mathcal{C}''$ **do**
      $\mathcal{C}'' = \mathcal{C}'' \cup \{C_j\}$;
      $C_j = \beta_{C_j}$;
    end
    forall the $C_i \in \mathcal{C} \backslash \mathcal{C}''$ **do**
      if $\beta_{C_i} \in \mathcal{C}''$ **then**
        $\mathcal{C}'' = \mathcal{C}'' \cup \{C_i\}$;
      end
    end
    $\mathcal{C} = \mathcal{C} \backslash \mathcal{C}''$;
    $\mathcal{C}' = \mathcal{C}' \cup \{\mathcal{C}''\}$;
  end
  return $\mathcal{C}'$;
end

C_2^1 and C_3^1 are grouped as C_1^2, C_0^3 consists of C_0^2 and C_1^2. a and c are the joint nodes between C_0^1 and C_1^1, e and g are the joint nodes between C_2^1 and C_3^1, b and f are the joint nodes between C_0^2 and C_1^2 (Fig. 1 shows best friendships between nodes and between groups, and other edges between nodes are omitted). Thus, by skeleton construction we can organize all the nodes into different groups and each level of the skeleton tree has different number of groups.

D. Computation Complexity

Skeleton construction starts with $|V|$ individual nodes and ends with one group. As a newly formed group at least contains two members at each grouping step, there are at most $\log_2 |V|$ grouping processes. Since, at each grouping step, the groups connected by best friendships form together, the time complexity of ith grouping process is $(\frac{1}{2})^{i-1}|V| \times d \times 2^{i-1}$, where $(\frac{1}{2})^{i-1}|V|$ denotes the number of groups and $d \times 2^{i-1}$ denote the time to compute the best friend for each group (2^{i-1} denotes the number of nodes in each group, and d denotes the average number of neighbors for each node and is a constant). Thus, the time complexity of skeleton construction is $O(|V| \log |V|)$, which is less than $O(|V|^3)$ [8] and $O(|V|^2)$ [10] for community detection.

IV. Skeleton Evolution

In this section, we address the following issue: *How to adapt skeleton construction to dynamic network?* Although best friendship of each node is much stable in mobile social networks, as shown in Fig. 3, it still varies over time. As skeleton is constructed based on best friendships, skeleton should be able to evolve over network when best friendships change and when nodes are added into or removed from the network.

Algorithm 2: Update best friendship

Input : S_{t-1}
Output: S_t

`//` $u \to v \Rightarrow u \to w$, `where` $v \in C_m^i$, $w \in C_n^i$
$\mathcal{C} = \{C_n^i, C_m^i\}$;
$C^j = \theta_{\mathcal{C}}$;
$C_n^i = C_n^i \cup \{u\}$;
forall the $k \in C_m^i$ **do**
  if $\beta_k \in C_n^i$ **then**
    $C_n^i = C_n^i \cup \{k\}$;
    $C_m^i = C_m^i \backslash \{k\}$;
  end
end
for $k = i$ **to** $j - 1$ **do**
  $\mathcal{C}_{C^j}^{k+1}$ =`Skeleton(`$\mathcal{C}_{C^j}^k$`)`;
  if $|\mathcal{C}_{C^j}^{k+1}| = 1$ **then**
    break;
  end
end

A. Update Best Friendship

When the best friendship changes between either nodes or groups, the skeleton tree needs to adapt accordingly. For example, suppose node or subgroup a, b and c are in group C_m^i, d and e are in group C_n^i. At time $t-1$ we have $c \to b$. At time t, $c \to a$. Since a, b and c are all in group C_m^i, there is no change needed for the skeleton tree. However, if at time t, $c \to e$, then we will have $C_m^i = \{a, b\}$ and $C_n^i = \{c, d, e\}$. Since $\mathcal{C}_{i+1}$ is constructed based on $\mathcal{C}_i$, we have to update upper levels of the skeleton tree. Let $\mathcal{C} = \{C_m^i, C_n^i\}$ and suppose the lowest common group of $\mathcal{C}$ is C^j. As C^j contains all the nodes or subgroups included in C_m^i and C_n^i, the change of C_m^i and C_n^i does not affect C^j, and thus it does not impact $\mathcal{C}_j$ and the levels above. Therefore, when the best friendship changes at C_m^i and C_n^i, we need to update subtree S_{C^j} from level i to $j-1$. Algorithm 2 shows the details of updating best friendship.

B. Node Insertion

First, let us consider the case that a person joins a group in a social network. When the person becomes a member of a social group, he must have contacted with another member in the group. Thus, for the insertion of node $u \notin V$ that has the contact with node $v \in V$, the skeleton is updated with $C_m^1 = C_m^1 \cup \{u\}$, where $v \in C_m^1$, and the change of C_m^1 will not affect the rest of the skeleton tree.

Then we consider another more complex situation, i.e, node u has multiple edges with nodes in the network. In real world, this situation happens when a person returns back after a long trip, and his social relations still remain. For this case, node u will be attached to group $C_n^1 = C_n^1 \cup \{u\}$, where $v = \beta_u$ and $v \in C_n^1$. However, this change may affect the skeleton tree since node u may have connections with other nodes and hence change the best friendship of other nodes. Thus, we have to update the skeleton tree for the insertion of node u. Algorithm 3 describes the procedure of handling node insertion.

C. Node Removal

When a node u is removed from the network, all edges connected with u will be removed. Also, the nodes, who have

Algorithm 3: Node insertion

Input : S_{t-1}
Output: S_t
// u is inserted
$v = \beta_u$;
$C^1 = C^1 \cup \{u\}$, where $v \in C^1$;
if $|N_u| \neq 1$ **then** // N_u is neighbor set of u
  forall the $k \in N_u$ **do**
    $\mathcal{C} = \mathcal{C} \cup \{C\}$, where $k \in C$;
  end
  $C^i = \theta_{\mathcal{C}}$;
  for $k = 1$ **to** $i - 1$ **do**
    $\mathcal{C}_{C^i}^{k+1} =$Skeleton$(\mathcal{C}_{C^i}^{k})$;
    if $|\mathcal{C}_{C^i}^{k+1}| = 1$ **then**
      break;
    end
  end
end

Algorithm 4: Node removal

Input : S_{t-1}
Output: S_t
// u is removed
$C_m^1 = C_m^1 \backslash \{u\}$, where $u \in C_m^1$;
forall the $v \in N_u$ **do** // N_u is neighbor set of u
  if $u = \beta_v$ **then**
    $C_m^1 = C_m^1 \backslash \{v\}$;
    update β_v;
    $C^1 = C^1 \cup \{v\}$, where $\beta_v \in C^1$;
    $\mathcal{C} = \mathcal{C} \cup \{C^1\}$;
  else
    $\mathcal{C} = \mathcal{C} \cup \{C^1\}$, where $v \in C^1$;
  end
end
$\mathcal{C} = \mathcal{C} \cup \{C_m^1\}$;
$C^i = \theta_{\mathcal{C}}$;
for $k = 1$ **to** $i - 1$ **do**
  $\mathcal{C}_{C^i}^{k+1} =$Skeleton$(\mathcal{C}_{C^i}^{k})$;
  if $|\mathcal{C}_{C^i}^{k+1}| = 1$ **then**
    break;
  end
end

the same best friend u, need to update their best friendship. To update the skeleton tree after node removal, we need to track which groups are directly affected. For example, after node u, $u \in C_m^1$, is removed, which is the best friend of node v, v will be connected to its new best friend w and included in C_n^1, $w \in C_n^1$, and hence C_m^1 and C_n^1 are directly affected. Therefore, for node removal, a subtree of the skeleton tree needs to be updated, where the root of the subtree is the lowest common group of these directly affected groups. Algorithm 4 gives the details.

V. Skeleton based Applications

In this section, we exploit skeleton for two applications: data forwarding and worm containment in mobile social networks.

A. Skeleton based Data Forwarding

Different from data forwarding strategies based on community [10][11], we leverage skeleton to facilitate data forwarding in mobile social networks. The basic idea of skeleton based routing for data forwarding is to route messages along best friendships and through joint nodes.

In the skeleton tree, the shortest routing path (for simplicity we also call it routing path) can be established for nodes or groups connected by best friendships. For example, as shown in Fig. 2b, the routing path from node f and h is denoted as (f, h), $|(f, h)|$ denotes the routing distance, where $|(f, h)| = \frac{1}{5} + \frac{1}{4}$ (the distance between two nodes connected by best friendship is the reciprocal of the edge weight), and similarly $|(C_0^1, C_1^1)| = \frac{1}{9}$. Moreover, for the routing path, e.g., (f, h) as shown in 2b, i is called the *next hop node*, and similarly we also have *next hop group* for the routing path between groups. To route messages between groups, joint nodes are used. For example, as shown in Fig. 2b, to forward a message from node f to d, the message is firstly forwarded to the joint node e that connects C_0^1 from C_1^1, then from e to nodes in C_0^1, e.g., from e to c, finally from c to d. We can see there are two cases in skeleton based routing: the routing between nodes within group (e.g., from f to e), where nodes are connected by best friendships, referred to as *intra-group routing*, and the routing between nodes in different groups (e.g., from e to c), referred to as *inter-group routing*.

With the establishment of routing paths for nodes or groups connected by best friendships, the overall strategy of skeleton based routing is as follows. Suppose a node (sender) carries a message destined to another node (receiver). When the sender encounters a node (relay), it will forward the message to the relay:

- if the relay has shorter routing distance to the receiver or to the joint node that connects the next hop group than the sender, when the routing path can be established for the sender, relay and receiver or joint node;
- if the relay's group has shorter routing distance to the receiver's group or to the joint node's group that connects the next hop group than the sender's group, when the routing path can be established for the sender's, relay's and receiver's or joint node's group.

Let us give an example of of skeleton based routing. In Fig. 1, node d has a message for node h. First, we identify the lowest common group of d and h, which is C_0^3. From the top of subtree $S_{C_0^3}$ down, b is the joint node from C_0^2 to C_1^2, and c is the joint node from C_1^1 to C_0^1. Thus, node d will forward the message if it encounters c since c is the joint node that connects C_0^1, or b and a since they are either the joint node that connects C_1^2 or in the same group with the joint node, or e, f, g and h since they are either the receiver or in the same group with the receiver. For another example, as shown in Fig. 2, node i sends a message to node d. Node i will relay the message if it encounters g and f since they have shorter routing distance to the joint node e that connects C_0^1 than i. After node e receives the message, it will relay the message when it encounters any node in C_0^1. Finally, the message is relayed to d with C_0^1. The pseudo code of skeleton based routing is detailed in Algorithm 5.

In skeleton based routing, best friendships between nodes are used to build the routing path locally within group, where messages are forwarded by intra-group routing; joint nodes are used to bridge the gap between groups, where messages are

Algorithm 5: Skeleton based routing

Input : $u, v, w \in V$

// u is sender, v is receiver and w is relay.

$C^i = \theta_{\mathcal{C}}$, where $\mathcal{C} = \{\{u\}, \{v\}\}$;
$j = i$;
while $j \geq 1$ **do**
  $C_u = C_m^{j-1}$, where $u \in C_m^{j-1}$ and $C_m^{j-1} \in \mathcal{C}_{C^i}^{j-1}$;
  $C_v = C_n^{j-1}$, where $v \in C_n^{j-1}$ and $C_n^{j-1} \in \mathcal{C}_{C^i}^{j-1}$;
  $C_w = C_o^{j-1}$, where $w \in C_o^{j-1}$ and $C_o^{j-1} \in \mathcal{C}_{C^i}^{j-1}$;
  if $C_u = C_v = C_w$ **then**
    if $j = 1$ **then**
      if $|(w, v)| < |(u, v)|$ **then**
        w.**AddMessage**();
      end
      break;
    else
      $j = j - 1$;
      continue;
    end
  else if $C_v = C_w \neq C_u$ **then**
    w.**AddMessage**();
    break;
  else if $C_u = C_v \neq C_w$ **then**
    break;
  else if $C_u = C_w \neq C_v$ **then**
    $v = N_{C_u C_{next}}$, where C_{next} is next hop group of (C_u, C_v) and $N_{C_u C_{next}}$ denotes the joint node that connects C_{next} from C_u;
    $j = j - 1$;
    continue;
  else if $C_u \neq C_w \neq C_v$ **then**
    if $|(C_w, C_v)| < |(C_u, C_v)|$ **then**
      w.**AddMessage**();
    end
    break;
  end
end

forwarded by inter-group routing.

Theorem 1. *With skeleton constructed as a complete binary tree, the expected hop count of intra-group routing path $E(H)$ and the expected number of inter-group routing $E(D)$ between any pair of nodes are $\frac{3^{n-1}}{2^n-1}$ and $\frac{\sum_{k=1}^{n-1} 2^k 3^{n-1-k}}{2^n-1}$, respectively, where $n = \log_2 |V|$ (see Appendix A for the proof).*

From Theorem 1, we have $E(H) = \frac{3^{n-1}}{2^n-1}$ and $E(D) = \frac{\sum_{k=1}^{n-1} 2^k 3^{n-1-k}}{2^n-1}$. That is, for example, $E(H) \approx 19$ and $E(D) \approx 37$ for a network with 1024 nodes. However, these are the worst case of skeleton based routing. Since message forwarding in skeleton based routing does not stick to the route path, it should be much less than $E(H)$ and $E(D)$.

B. Skeleton based Worm Containment

Mobile social networks become popular recently with the rapid development of smartphone such as iPhone, Android phone and Window phone, where people can keep in touch with friends and family, share news, photos and videos with mobile devices. However, due to the openness of API of iOS, Android and Windows phone OS, it is also easier for hackers to write malicious software that can control the smartphone and propagate viruses and worms to other mobile devices quickly by mobile social network apps.

Social-based worm containment for cellular networks and online social networks has been proposed in [18], [10] and [22]. The intuition behind these schemes is to contain worms within the infected community before they can spread out widely. Like these schemes, skeleton based patching for worm containment also aims to isolate the infected group, however, based on skeleton.

As discussed in Section III, in skeleton construction, after the first grouping process, network nodes are separated into different groups which are referred to as *fine-grained groups*. Inside group, nodes are connected by best friendships and hence node contacts are much frequent. If a node in a fine-grained group is infected, it is difficult to keep other nodes inside the group from being infected. Therefore, the basic idea of skeleton based patching is to patch the nodes which can isolate the fine-grained groups as much as possible.

Based on the fine-grained groups, we define the external connectivity of each node u as the sum of the edge weights between u and nodes outside u's fine-grained group. Since distributing patches to all nodes at the same time may not always be feasible, e.g., due to the bandwidth limitation in cellular networks, a proper patching order is needed for all the nodes. In skeleton based patching, the patching sequence is determined by the external connectivity, i.e., the higher its external connectivity is, the sooner the node will be patched.

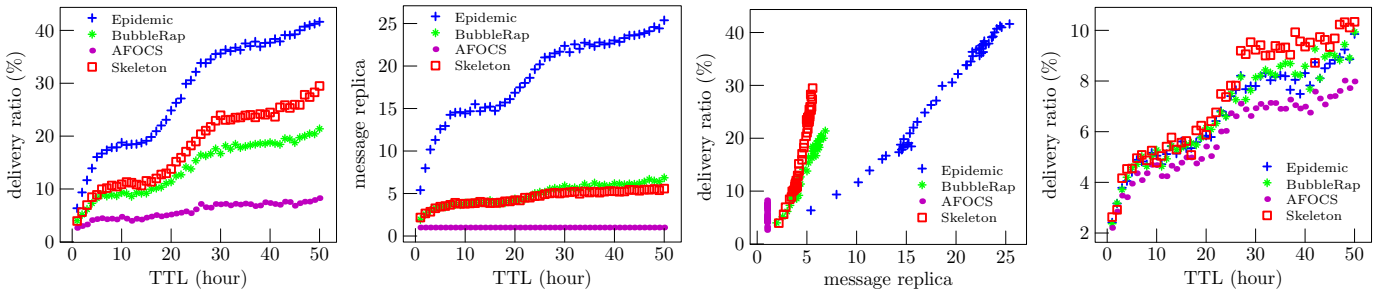
The intuition of skeleton based patching is to disconnect communications and connections among fine-grained groups. For example, there are four laboratories in computer science department, networking labs a and b are neighboring in one building, computer vision labs c and d are also neighboring and in another building, where each lab is an individual group. Moreover, labs a and b are further composed of networking lab; labs c and d are further composed of computer vision lab. In skeleton based patching, instead of networking lab and computer vision lab, we focus on the connections among lab a, b, c and d so as to contain worms within small groups. By patching nodes with high external connectivity built on fine-grained groups, skeleton based patching can better isolate infected groups than community based schemes.

VI. Performance Evaluations

In this section, we evaluate the performance of skeleton based data forwarding and worm containment algorithms.

A. Data Forwarding

The evaluations are conducted based on the MIT Reality trace. In the simulation, each node sends 500 messages to other randomly selected nodes, and messages will be discarded if they are not successfully delivered within the Time-To-Live (TTL), which varies from 1 hour to 50 hours. We compare skeleton based routing (*Skeleton*) to other three forwarding strategies: *Epidemic* [23], *BubbleRap* [11] and *AFOCS* [10], where half trace is used as warmup to perform the skeleton construction in *Skeleton* or the community detection for *BubbleRap* and *AFOCS*, and another half is used to evaluate data forwarding algorithms. For skeleton based routing, skeleton keeps evolving with the network after warmup.



(a) delivery ratio (b) message replica (c) frontier (d) delivery ratio without replication

Fig. 4: Performance of data forwarding algorithms **Epidemic**, **BubbleRap**, **AFOCS** and **Skeleton** on the MIT Reality trace.

We evaluate all the algorithms based on two message forwarding modes: forwarding with message duplication and forwarding without message duplication. For forwarding with message duplication, the algorithms are compared in term of data delivery ratio, the number of message replica, and frontier between delivery ratio and message replica, where *Epidemic* is the upper bound for delivery ratio and the lower bound for message replica since in *Epidemic* messages are duplicated and forwarded to any encountered node. For the forwarding without message duplication, data delivery ratio is compared. As message duplication that incurs the storage cost can be seen as the data forwarding cost, delivery ratio achieved in this mode indicates the efficiency of the forwarding algorithm.

Fig. 4 shows the results of the data forwarding on the MIT Reality trace. As analyzed above, *Epidemic* has the highest delivery ratio and message replicas as expected in Fig. 4a and 4b, meanwhile *AFOCS* has the lowest delivery ratio and message replicas. The delivery ratio of *Skeleton* is, up to 25%, better than that of *BubbleRap*. Moreover, *Skeleton* has much less message replicas than *Epidemic* and up to 30% less than *BubbleRap* as shown in Fig. 4b. Fig. 4c gives the frontier of each algorithm in terms of delivery ratio and message replicas. We can see *Skeleton* and *BubbleRap* sit in the same region, however, *Skeleton* is much better than *BubbleRap*, e.g., the delivery ratio of *Skeleton* is almost 30% with five message replicas, but the delivery ratio of *BubbleRap* is below 20%. Although *Epidemic* can achieve better delivery ratio than *Skeleton*, the message replica of *Epidemic* is almost 5 times of *Skeleton*. Fig. 4d shows the delivery ratio of data forwarding algorithms without message duplication, where *AFOCS* still has the worse delivery ratio, *Epidemic* and *BubbleRap* are equivalent, and *Skeleton* is the best. Moreover, *Skeleton* increasingly outperforms other algorithms with the increase of TTL.

To summarize, there exists a tradeoff between data delivery ratio and message replica. *Epidemic* represents the upper bound of the performance and cost of data forwarding, and *AFOCS* represents the lower bound of the performance and cost of data forwarding as shown in Fig. 4. *Skeleton* and *BubbleRap* span between them as illustrated in Fig. 4c. *Skeleton* achieves a good balance between performance and cost, i.e., it has the second best delivery ratio and the second best message replica, and it is also the most efficient algorithm. Since skeleton based routing relies on the most frequent connections in network (best friendships) for data forwarding, it outperforms other community-based forwarding algorithms *BubbleRap* and *AFOCS*.

B. Worm Containment

Worm containment algorithms are evaluated on the Facebook trace. Without loss of generality, we use similar worm propagation model in [10] that mimics the behaviors of the famous worm *Koobface* that once spread out on Facebook. We assume that the worms are able to explore the friendship information for propagation (such as sending out messages including malicious links). The probability of node activating worm received from friends is proportional to the contact frequency between them. The time taken for the worm to propagate from one user to his friend is inversely proportional to the contact frequency between them. Finally, the worm starts to propagate right after it successfully infects the user.

Initially, we randomly choose 0.05% of nodes as the seed set of worm sources to initiate the infection process. When the infection rate (the fraction of infected nodes over all nodes) reaches the predefined alarm threshold α, the patching process will be initiated. The experiments are conducted with $\alpha = 5\%$, 10%, and 15%.

We compare skeleton based patching (*Skeleton*) with *AFOCS*, *iWander* [22] and the cluster based scheme [18] (*Clustering*). Among all these schemes, *AFOCS* selects a particular part of nodes to be patched (the number of patched nodes is 985 for the Facebook trace), meanwhile all other algorithms determine the patching sequence of all nodes. In order to compare the performance among different schemes, we first choose the set of patched nodes V_p, $|V_p| = 985$, for each scheme, then choose $|V|p = 2000$ for all the schemes except *AFOCS*. For *iWander*, *Clustering* and *Skeleton*, nodes are selected according to their patching sequence.

Similar to the evaluations for data forwarding, half trace is used for warmup to perform the skeleton construction for *Skeleton*, the community detection for *AFOCS*, the clustered partitioning for *Clustering*, or the random walk for *iWander*. Another half trace is used for evaluation. Also, the skeleton keeps evolving with the network after warmup. The worm

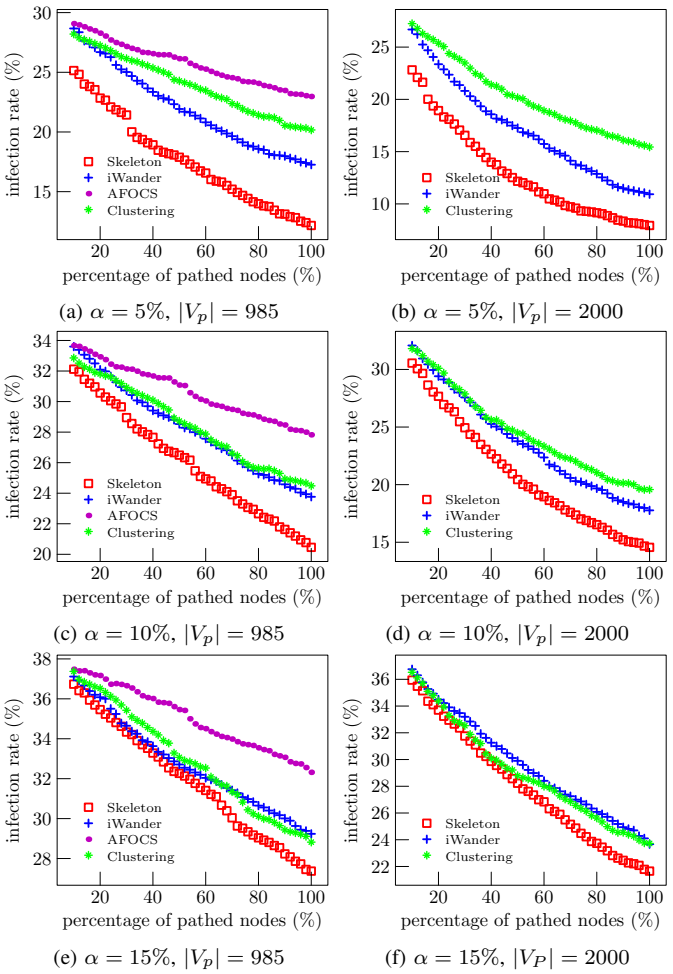


Fig. 5: Performance of worm containment algorithms – **Skeleton**, **iWander**, **AFOCS** and **Clustering** in terms of infection rate on the Facebook trace.

propagation is continually simulated for 30 days after the alarm threshold is reached.

Fig. 5 shows the infection rates achieved by different patch schemes with alarm threshold $\alpha = 5\%$, 10%, and 15%, respectively, for both 985 and 2000 patching nodes. As shown in Fig. 5, the infection rates of all the schemes decrease as the percentage of patch nodes increases and the infection rates increase with the increase of alarm threshold. For $\alpha = 5\%$, 10%, and 15%, *Skeleton* outperforms other schemes and it increasingly outperforms other algorithms with the increase of patched nodes, where *AFOCS* is the worst, and *iWander* and *Clustering* are comparable. In addition, the difference between *Skeleton* and other algorithms narrows down from $\alpha = 5\%$ to $\alpha = 15\%$, because with higher alarm threshold, more nodes are infected before the patching process and the infected nodes can distribute all over the communities, groups, clusters, which degrade the performance of all strategies.

To summarize, skeleton based patching outperforms all other strategies, which demonstrates that patching nodes with higher external connectivity can effectively contain the worm propagation, and isolating the fine-grained groups in skeleton works better than communities or clusters.

VII. Related Work

Hui *et al.* [11] proposed BubbleRap for data forwarding based on community, where the rankings of local centrality and global centrality were used to forwarding data within community and between communities. Chen *et al.* [24] designed Social Map, where data forwarding was carried out by social map within community and node degree between social maps. However, the nodes with high global centrality [11] or active degree [24] may not effectively connect communities or social maps. Nguyen *et al.* [10] investigated the detection of overlapped communities in dynamic binary networks and selected overlapping nodes to relay messages between communities and to be patched for worm containment. However, as the detected communities in binary networks may not accurately reflect the network structure, this design has some limitations. Han *et al.* [22] identified node's influence as the betweenness centrality computed by random walk and used the node influence as a metric for data dissemination and disease control. However, the betweenness centrality is a global characteristic, which cannot be used to improve performance of data dissemination and contain disease locally. Zhu *et al.* [18] proposed a cluster based patching scheme for worm containment in cellular networks, where separators (i.e., key nodes that separate clusters) were patched. However, worms may still spread within clusters, and the performance may be degraded when there are many nodes within clusters.

VIII. Conclusions

In this paper, we proposed skeleton as the network structure of mobile social networks and exploited it for protocol designs. We designed algorithms for skeleton construction and skeleton evolution. As skeleton is constructed based on best friendship, which is the most frequent and reliable connection, it can be used to accurately predict node contacts. Based on skeleton, we designed skeleton based routing for data forwarding and skeleton based patching for worm containment. Trace-driven simulations show that skeleton based routing has better performance than existing community based algorithms and skeleton based patching also outperforms existing schemes.

References

[1] W. Peng, F. Li, X. Zou, and J. Wu, "A privacy-preserving social-aware incentive system for word-of-mouth advertisement dissemination on smart mobile devices," in *Proc. of IEEE SECON*, 2012.

[2] W. Hu, G. Cao, S. V. Krishanamurthy, and P. Mohapatra, "Mobility-assisted energy-aware user contact detection in mobile social networks," in *Proc. of IEEE ICDCS*, 2013.

[3] T. Ning, Z. Yang, H. Wu, and Z. Han, "Self-interest-drive incentives for ad dissemination in autonomous mobile social networks," in *Proc. of IEEE INFOCOM*, 2013.

[4] J. Wu, M. Xiao, and L. Huang, "Homing spread: Community home-based multi-copy routing in mobile social networks," in *Proc. of IEEE INFOCOM*, 2013.

[5] Z. Lu, Y. Wen, and G. Cao, "Information diffusion in mobile social networks: The speed perspective," in *Proc. of IEEE INFOCOM*, 2014.

[6] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, p. 7821, 2002.
[7] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.
[8] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.
[9] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.
[10] N. Nguyen, T. Dinh, S. Tokala, and M. Thai, "Overlapping communities in dynamic networks: their detection and mobile applications," in *Proc. of ACM MobiCom*, 2011.
[11] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 11, pp. 1576–1589, 2011.
[12] Z. Lu, Y. Wen, and G. Cao, "Community detection in weighted networks: Algorithms and applications," in *Proc. of IEEE PerCom*, 2013.
[13] X. Zhang and G. Cao, "Transient community detection and its application to data forwarding in delay tolerant networks," in *Proc. of IEEE ICNP*, 2013.
[14] P. Costa, C. Mascolo, M. Musolesi, and G. Picco, "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 5, pp. 748–760, 2008.
[15] E. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proc. of ACM MobiHoc*, 2007.
[16] Q. Yuan, I. Cardei, and J. Wu, "Predict and relay: an efficient routing in disruption-tolerant networks," in *Proc. of ACM MobiHoc*, 2009.
[17] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: a social network perspective," in *Proc. of ACM MobiHoc*, 2009.
[18] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci, "A social network based patching scheme for worm containment in cellular networks," in *Proc. of IEEE INFOCOM*, 2009.
[19] N. Nguyen, Y. Xuan, and M. Thai, "A novel method for worm containment on dynamic social networks," in *Proc. of IEEE MILCOM*, 2010.
[20] N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 255–268, 2006.
[21] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks*, August 2009.
[22] B. Han and A. Srinivasan, "Your friends have more friends than you do: identifying influential mobile users through random walks," in *Proc. of ACM MobiHoc*, 2012.
[23] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.
[24] K. Chen and H. Shen, "Smart: Lightweight distributed social map based routing in delay tolerant networks," in *Proc. of IEEE ICNP*, 2012.

APPENDIX

A. Proof of Theorem 1

We prove the theorem by induction.

When $n = 1$, clearly the hop count of intra-group routing path is $E_1(H) = 1$ and the number of inter-group routing $E_1(D) = 0$.

When $n = 2$, we use group C_0^2 in Fig. 1 as an example, where a and c are the joint nodes for group $\{a, b\}$ and $\{c, d\}$, by using node a as an example, we have the expected hop count from a to b $H(a, b) = 1$, $H(a, c) = \frac{1}{2}$, $H(a, d) = \frac{1}{2}$. Note that as we assume a has the same probability to encounter c or d for inter-group routing, $H(a, c)$ and $H(a, d)$ are $\frac{1}{2}$. By being similarly applied to other nodes, we have $E_2(H) = 6/\binom{4}{2} = 1$ for all pairs of nodes. As inter-group routing is needed for pair of nodes from different groups, for $n = 2$, we have $E_2(D) = 4/\binom{4}{2} = \frac{2}{3}$.

When $n = 3$, as shown in Fig. 1, for node a, we have $H(a, e) = H(a, b) + H(b, e)$, where $H(b, e) = \frac{1}{4}(H(e, f) + H(e, g) + H(e, h))$. As the skeleton tree is symmetric, $H(b, e) = \frac{3}{4}$. This is similar for any other nodes. All the paths of pair nodes in the skeleton tree $n = 3$ can be split into four times of all the paths of pair nodes in the skeleton tree $n = 2$ and $\binom{8}{2} - 2 \times \binom{4}{2}$ number of paths with $\frac{3}{4}$ hop count. So we have $E_3(H) = \left(E_2(H) \times 4 \times \binom{4}{2} + \left(\binom{8}{2} - 2 \times \binom{4}{2}\right) \times \frac{3}{4}\right) / \binom{8}{2} = \frac{9}{7}$. Similarly, for inter-group routing, we have $D(a, e) = D(a, b) + D(b, e)$, where $D(b, e) = \frac{D(b,e)+D(b,f)+D(b,g)+D(b,h)}{4} = \frac{3}{2}$. Thus, $D_3(P) = \left(D_2(P) \times 4 \times \binom{4}{2} + \left(\binom{8}{2} - 2 \times \binom{4}{2}\right) \times \frac{3}{2}\right) / \binom{8}{2} = \frac{10}{7}$.

When $n = 4$, similarly, for intra-group routing we have $E_4(H) = ((E_3(H) \times 4 \times \binom{8}{2} + \left(\binom{16}{2} - 2 \times \binom{8}{2}\right) \times \frac{9}{8}) / \binom{16}{2} = \frac{27}{15}$. For inter-group routing, we have $E_4(D) = ((E_3(D) \times 4 \times \binom{8}{2} + \left(\binom{16}{2} - 2 \times \binom{8}{2}\right) \times \frac{9}{4}) / \binom{16}{2} = \frac{38}{15}$.

By induction, we have

$$E_n(H) = \frac{4E_{n-1}(H)\binom{2^{n-1}}{2} + \left(\binom{2^n}{2} - 2\binom{2^{n-1}}{2}\right)\frac{3^{n-2}}{2^{n-1}}}{\binom{2^n}{2}}, \quad (1)$$

$$E_n(D) = \frac{4E_{n-1}(D)\binom{2^{n-1}}{2} + \left(\binom{2^n}{2} - 2\binom{2^{n-1}}{2}\right)\frac{3^{n-2}}{2^{n-2}}}{\binom{2^n}{2}}. \quad (2)$$

Assuming $E_n(H) = \frac{3^{n-1}}{2^n-1}$, for $n+1$ we have the following according to Eq. 1

$$\begin{aligned} E_{n+1}(H) &= \frac{4 \times E_n(H) \times \binom{2^n}{2}}{\binom{2^{n+1}}{2}} \\ &\quad + \frac{\left(\binom{2^{n+1}}{2} - 2\binom{2^n}{2}\right) \times \frac{3^{n-1}}{2^n}}{\binom{2^{n+1}}{2}} \\ &= \frac{E_n(H) \times 2 \times (2^n - 1) + 3^{n-1}}{2^{n+1} - 1} \\ &= \frac{\frac{3^{n-1}}{2^n-1} \times 2 \times (2^n - 1) + 3^{n-1}}{2^{n+1} - 1} \\ &= \frac{3^n}{2^{n+1} - 1}. \end{aligned}$$

Assuming $E_n(D) = \frac{\sum_{k=1}^{n-1} 2^k 3^{n-1-k}}{2^n-1}$, for $n+1$ we have the following according to Eq. 2

$$\begin{aligned} E_{n+1}(D) &= \frac{4 \times E_n(D) \times \binom{2^n}{2}}{\binom{2^{n+1}}{2}} \\ &\quad + \frac{\left(\binom{2^{n+1}}{2} - 2\binom{2^n}{2}\right) \times \frac{3^{n-1}}{2^{n-1}}}{\binom{2^{n+1}}{2}} \\ &= \frac{E_n(P) \times 2 \times (2^n - 1) + 2 \times 3^{n-1}}{2^{n+1} - 1} \\ &= \frac{\frac{\sum_{k=1}^{n-1} 2^k 3^{n-1-k}}{2^n-1} \times 2 \times (2^n - 1) + 2 \times 3^{n-1}}{2^{n+1} - 1} \\ &= \frac{\sum_{k=1}^{n} 2^k 3^{n-k}}{2^{n+1} - 1}. \end{aligned}$$

□