Approximation Algorithms for Permanent Dominating Set Problem on Dynamic Networks

Subhrangsu Mandal(✉) and Arobinda Gupta

Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur, Kharagpur, India
{subhrangsum,agupta}@cse.iitkgp.ernet.in

Abstract. A temporal graph is a graph whose node and/or edge set changes with time. Many dynamic networks in practice can be modeled as temporal graphs with different properties. Finding different types of dominating sets in such graphs is an important problem for efficient routing, broadcasting, or information dissemination in the network. In this paper, we address the problems of finding the minimum permanent dominating set and maximum k-dominant node set in temporal graphs modeled as evolving graphs. The problems are first shown to be NP-hard. A $\ln(n\tau)$-approximation algorithm is then presented for finding a minimum permanent dominating set, where n is the number of nodes, and τ is the lifetime of the given temporal graph. Detailed simulation results on some real life data sets representing different networks are also presented to evaluate the performance of the proposed algorithm. Finally, a $(1 - \frac{1}{e})$-approximation algorithm is presented for finding a maximum k-dominant node set.

Keywords: Permanent dominating set
Maximum k-dominant node set · Approximation algorithm
Temporal graph

1 Introduction

Increased use of various mobile devices has introduced communication networks where the network topology changes frequently. Vehicular networks, delay/disruption tolerant networks, low earth orbiting systems etc. are some examples of such dynamic networks. Traditional graph models are not always sufficient to model and analyse such highly dynamic networks, and several models have been proposed recently to represent them such as temporal graphs [11], dynamic graphs [17] etc. In the rest of this paper, we will informally refer to all such graphs as temporal graphs, as all of them represent graphs with time-dependent topologies.

For network with predictable mobility of the mobile entities, Ferreira et al. [4] proposed *evolving graphs* model to represent these types of networks. In this model, the temporal graph is represented as a finite sequence of static graphs,

A. Negi et al. (Eds.): ICDCIT 2018, LNCS 10722, pp. 265–279, 2018.
https://doi.org/10.1007/978-3-319-72344-0_22

each static graph being the graph at a discrete timestep. The nodes in each static graph represent the nodes in the network, and an edge in the static graph signifies that a link exists between the corresponding nodes at that timestep. The dynamicity of the topology is captured by the changing node/edge sets of the static graphs in the sequence. The total number of timesteps is called the *lifetime* of the temporal graph. The node at a single timestep is called an *instance* of that node.

Finding different types of dominating sets in a graph has important applications in routing, broadcasting, and information dissemination in static as well as dynamic networks [10,16,19] which may be centralized or distributed in nature. A dominating set (a set of nodes such that every node in the graph is either in the set or has at least one neighbor in the set) provides a subset of nodes such that if those nodes possess some information, they can disseminate that information to all other nodes in the next round of information dissemination. Using similar strategy, dominating set provides an efficient method for broadcasting, routing etc. in a distributed environment. Various definitions of dominating set for dynamic networks are available in the literature. Casteigts et al. [1] have proposed three variations of the dominating set problem for temporal graphs, *temporal dominating set*, *evolving dominating set* and *permanent dominating set*. A *temporal dominating set* is the set of nodes which dominates every other node of the given temporal graph in at least one timestep in the sequence of static graphs defining the evolving graph. An *evolving dominating set* is the set of dominating sets for the static graphs at each timestep. A *permanent dominating set* is the set of nodes which is a dominating set for every static graph at each timestep.

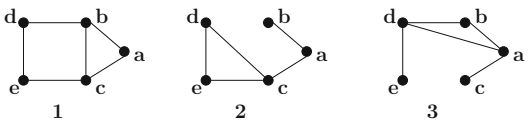

Fig. 1. A temporal graph at different timesteps

In Fig. 1, for the temporal graph $G = \{1, 2, 3\}$ a *temporal dominating set* is $\{c\}$, an *evolving dominating set* is $\{\{b, e\}, \{b, c\}, \{c, d\}\}$ where $\{b, e\}$, $\{b, c\}$ and $\{c, d\}$ are dominating sets for static graphs at timesteps 1, 2 and 3 respectively, and a *permanent dominating set* is $\{a, e\}$. The problem of finding an *evolving dominating set* has been addressed in [8,20]. To the best of our knowledge the problem of finding a *permanent dominating set* has not been addressed till now.

In this paper, we address the problem of finding a *permanent dominating set* for a given temporal graph modeled using the *evolving graphs* model. A solution for the problem of finding the *minimum permanent dominating set* (permanent dominating set of minimum cardinality) for a given temporal graph is proposed first. Then the problem of finding the *maximum k-dominant node set* for a given

temporal graph is addressed. The *maximum k-dominant node set* is the set of nodes with a given cardinality k, which covers the maximum number of node instances of the temporal graph among all such sets of size k. It has been assumed for both the problems that only the edge set of the temporal graph changes with time. We show that the first problem is NP-Complete, the second problem is known to be NP-Hard [14]. We first propose a $\ln(n\tau)$-approximation algorithm for finding a minimum permanent dominating set, where n is the number of nodes, and τ is the lifetime of the given temporal graph. Detailed simulation results on some real life data sets for different networks are presented to evaluate the performance of the proposed algorithm. A $(1 - \frac{1}{e})$-approximation algorithm is then presented for finding the maximum k-dominant node set.

The rest of the paper is organised as follows. Section 2 discusses some related works. Section 3 presents the system model used in this paper. Section 4 first shows that the permanent dominating set problem is NP-Complete. It then describes a greedy approximation algorithm to solve the permanent dominating set problem. It also presents detailed simulation results on some real life data sets for different networks to evaluate the performance of the proposed algorithm. In Sect. 5, a greedy approximation algorithm to find a maximum k-dominant node set is given. Finally, Sect. 6 concludes the paper.

2 Related Works

Finding minimum dominating set for static graphs has been a well researched problem because of its applications in various problems of communication networks. Extension of this problem from static graphs to temporal graphs is not straightforward, and various versions of this problem are solved using different approaches.

One approach is the maintenance of a minimum dominating set for a given temporal graph when changes in underlying graph topology occur. Whitbeck et al. [20] solve this problem in an on-demand basis. They compute the dominating set for a graph first and after every edge addition or deletion, recompute it, if the previous one is no longer a dominating set for the changed graph. In [8], Guibas et al. solve the problem of finding a minimum connected dominating set for geometric graphs under node insertion and deletion in a similar fashion. There are other works by Gao et al. [5], and Harshberger [9] which address the problem of finding a dominating set for temporal graphs with a similar approach where nodes follow some continuous trajectory to enter or leave the graph.

Another version of this problem is finding the *temporal dominating set* [1] for a given temporal graph. This version of the problem basically maps to the problem of finding a minimum dominating set for the *underlying graph* which is the union of all graphs at every timestep of the lifetime of the given temporal graph. Hence prior knowledge of changes in the graph topology is required to solve this problem. In [16] Ros et al. have used a graph structure similar to *connected temporal dominating set* (a temporal dominating set which is connected) to propose a backbone structure to address the problem of information dissemination in the context of vehicular networks. In [3], Dubois et al. have solved a

slightly altered version of this problem. They have computed the dominating set for the underlying graph after eliminating edges which do not occur infinitely often in the temporal graph.

In this paper, we have proposed an approximation algorithm to solve the *minimum permanent dominating set* problem for a temporal graph. To the best of our knowledge, there is no other existing work which addresses the problem.

3 System Model and Assumptions

In this paper, we represent a temporal graph by the *evolving graphs* [4] model. In this model the node set of the temporal graph remains same, the edge set changes with time and all information about the changes of the edge set is available for a certain time period which is called the lifetime of the given temporal graph. All the changes in the graph topology are known apriori. The node set of the temporal graph is denoted by V and the lifetime is denoted by τ. The time interval for which the temporal graph is available is $(0, \tau]$. Throughout this paper all time intervals are in discrete time system. The edge set is denoted by E where every element $e \in E$ is represented by $e(u, v, (s_e^1, d_e^1), (s_e^2, d_e^2), \cdots)$, where $u, v \in V$ and edge e connects nodes u and v. Each pair of (s_e^i, d_e^i) denotes a half open time interval $(s_e^i, s_e^i + d_e^i]$ for which edge e is connecting u and v in the given temporal graph, s_e^i, $0 \leq s_e^i < \tau$, denotes the starting time of that time interval and d_e^i, $0 < d_e^i \leq \tau$ is the corresponding duration for which e is available, $(s_e^i + d_e^i) \leq \tau$. There may be multiple time intervals for an edge. In such a condition, for any two time intervals say (s_e^i, d_e^i) and (s_e^j, d_e^j), $s_e^i \neq s_e^j$ and if $s_e^i < s_e^j$ then $s_e^i + d_e^i < s_e^j$. Thus the maximum number of time intervals for an edge can be $\lfloor \frac{\tau}{2} \rfloor$. An edge $e \in E$ connecting nodes $u, v \in V$ can be denoted by e_{uv}. As G is an undirected graph the ordering of u, v does not matter. For a node $u \in V$ the sum of all durations of all edges incident on it is called *temporal degree* of u, denoted by TD_u. The temporal graph is denoted by $G(V, E)$.

An instance of a node u at time t $(0 < t \leq \tau)$ is denoted by u^t. All nodes are present for the whole lifetime of G. So each node has τ instances and in total there are $n\tau$ node instances in G, where $|V| = n$.

The set of *neighbouring node instances*, N_u, of a node u in G is defined as the set of node instances in G which has an instance of u as its neighbour. Instances of node u are also included in N_u. Set of neighbouring node instances, N_S, of a set of nodes $S \subseteq V$ is given by $N_S = \bigcup_{v \in S} N_v$. If any instance of a node u does not have any neighbour then u is called an *isolated* node in G.

4 Finding Permanent Dominating Set for Temporal Graphs

The *dominating set* problem [6] for static graphs is a well known NP-Complete problem. We know that every static graph can be represented as a temporal graph of lifetime 1. Using this it is quite straightforward to prove that the *dominating set* problem is polynomially reducible to the *permanent dominating set* problem. From this we get the following theorem:

Theorem 1. *Permanent dominating set problem is NP-Complete.*

We next present a greedy approximation algorithm to find a minimum permanent dominating set for a given temporal graph. The algorithm will add nodes one by one to the permanent dominating set. Every node selection will be done greedily based on some parameter. Once a node is added to the permanent dominating set, it will not be removed thereafter. We first define the following.

Definition 1. ***Dominance of node*** u ***at time*** t *(*CD_u^t*): Let* D^t *be the already constructed partial permanent dominating set till timestep* t*,* $0 \leq t \leq \tau$*. The dominance of node* u *at time* t CD_u^t*, is equal to* $|N_u \setminus N_{D^t}|$*.*

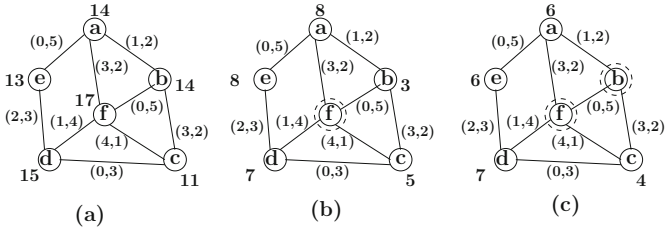

Fig. 2. A temporal graph with dominance information

Figure 2 shows a temporal graph with lifetime $\tau = 5$ and set of nodes $\{a, b, c, d, e, f\}$. Figure 2(a) shows dominance when no node is member of permanent dominating set. Figure 2(b) and (c) show dominance when node f and nodes $\{f, b\}$ are members of permanent dominating set respectively.

Let there be an edge $e(u, v, s_e^1, d_e^1)$ connecting two nodes u and v in the given temporal graph. If u gets added to the permanent dominating set, then we say that node u dominates node v for the duration $(s_e^1, s_e^1 + d_e^1]$. We also say that d_e^1 instances of node v are dominated or covered by node u.

The proposed greedy algorithm works as follows. At any point of execution of the algorithm, all the nodes of the given temporal graph are coloured with any one of the colours *white, black,* or *grey.* A node is coloured *black* if it is a member of the permanent dominating set. A node is coloured *grey* if all instances of it are dominated by at least one node in the permanent dominating set formed so far. A node is coloured *white* if at least one instance of it is not dominated by any node in the permanent dominating set. Initially all nodes are *white.* If a node becomes *grey* or *black* it never becomes *white* again. This implies that the dominance of a node either decreases or remains the same with each round of node selection as a member of the permanent dominating set. Higher dominance of a node at a certain point of time means very few neighbours of this node are *black* and it will dominate more uncovered node instances if it is selected as a member of the permanent dominating set. Thus the node with the highest value of dominance becomes a good choice for selection as a member of the permanent

dominating set. The proposed algorithm *GreedyPDS* uses this greedy strategy to construct a permanent dominating set. At any step, the algorithm will choose the node with the highest dominance value, add it to the permanent dominating set, recompute the dominance of all other nodes and then proceed to the next step. The pseudo code shown in Algorithm 1 describes the basic steps of the algorithm. In Algorithm 1, I is the set of isolated nodes in G, $colour_u^t$ is the colour of a node u at time t, CV_u^t denotes the total number of instances of u with black neighbours till time t, TD_u is the temporal degree of node u. In Algorithm 1, initializations are done in Lines 1–3. Then Lines 4 and 5 add all isolated nodes to the permanent dominating set, D. Finally Lines 7–9 add nodes with maximum dominance at the time of node selection to D. It stops when all nodes in G are coloured with black or grey.

Algorithm 1. GreedyPDS(G)

Input: A temporal graph G with node set V edge set E, $|V| = n$, lifetime τ.
Output: A permanent dominating set D.

```
1: D := ∅
2: ∀u ∈ V colour_u^0 := white,
3: ∀u ∈ V CV_u^0 := 0, CD_u^0 := TD_u + τ
4: for all i, u_i ∈ I do
5:     AddToDomSet(u_i)
6: end for
7: while ∃ v ∈ V such that colour_v^t = white do
8:     u := max {CD_u^t | u ∈ V}
9:     AddToDomSet(u)
10: end while
11: return (D)
```

The algorithm *GreedyPDS* calls the subroutine *AddToDomSet* to add each node to D and update the dominance information of the other nodes in G. To recalculate the dominance information of each node after each node selection, we maintain a list of time intervals, *cvg* list, sorted by the starting time of time interval, per node in G. After adding a node u to D, the subroutine *AddToDomSet* colours it with black and inserts the associated time intervals of edges incident on u to the *cvg* lists of the nodes which are connected with u by those edges. During insertion, it merges all the overlapping time intervals into a single time interval. Then, if any neighbour of u has neighbours in D throughout the lifetime, it colours that node with grey. As u is added to D the neighbouring nodes of u will no longer dominate any node instances of u, *AddToDomSet* subtracts that value from the dominance information of the neighbours of u.

As some instances of neighbouring nodes of u get dominated by u, some changes may happen in the dominance information of two hop neighbours of u. Let node v and node w be one hop and two hop neighbours of u respectively, such that $e_{uv}, e_{vw} \in E$ and $|(s_{e_{uv}}, s_{e_{uv}} + d_{e_{uv}}] \cap (s_{e_{vw}}, s_{e_{vw}} + d_{e_{vw}}]| = \lambda \neq 0$. Since u is a black node and it dominates v for the duration $(s_{e_{uv}}, s_{e_{uv}} + d_{e_{uv}}]$, then w

is no longer going to dominate node v for this λ duration. So *AddToDomSet* subroutine subtracts λ from the dominance of w.

Figure 3 shows the execution of the proposed greedy algorithm *GreedyPDS* on a given temporal graph with lifetime 5. Figure 3(a) shows the given temporal graph with dominance information at the beginning of the execution. At first, as node f has the highest dominance of 17, it gets added to the permanent dominating set and it is coloured black. As all instances of node b are dominated by f, it is coloured grey. Figure 3(b) depicts this scenario. After that, by similar greedy strategy, node a and node c are coloured black, then rest of the nodes are coloured grey, and the algorithm terminates. Figure 3(c) and (d) show the steps of selection of node a and node c as members of the permanent dominating set. $\{f, a, c\}$ is the resultant permanent dominating set.

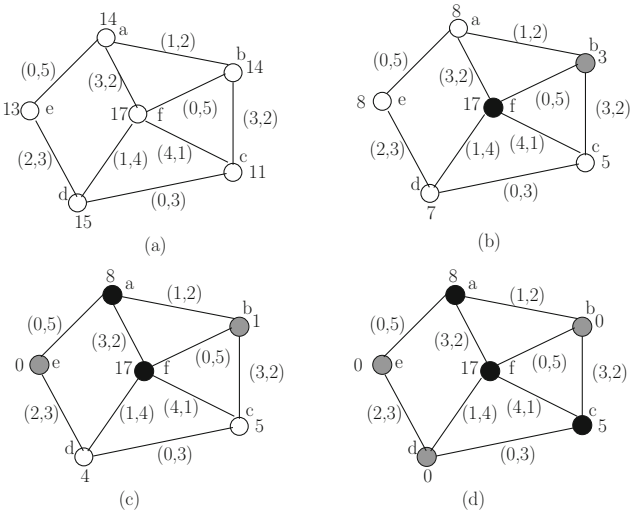

Fig. 3. Execution of *GreedyPDS* on a temporal graph with lifetime 5

Theorem 2. *GreedyPDS correctly computes a permanent dominating set D for a temporal graph G in $O(m\tau^2 n^2)$ time.*

Proof. At first *GreedyPDS* selects all isolated nodes as members of D. Then it adds a node with the highest dominance at that point of time to D. This procedure continues until all nodes are coloured with black or grey. This means that *GreedyPDS* terminates only when all nodes of the given temporal graph are either a member of D or have neighbours in D throughout the lifetime of G. This implies that D is a permanent dominating set for G.

For the dominance information, one sorted list per node is maintained and the maximum number of time intervals per edge is $\lfloor \frac{\tau}{2} \rfloor$. Insertion of a time interval to the list of a node takes $O(\frac{\tau}{2})$ time. So after each node selection, update of

the dominance information of its one and two hop neighbours takes $O(\frac{\tau^2 n}{4})$ and $O(\frac{\tau^2 n^2}{4})$ time respectively, as any node can have maximum n neighbours. Hence the algorithm takes $O(m\tau^2 n^2)$ time to find D for G, where $m = |D|$. □

To prove the approximation ratio of *GreedyPDS* we need following lemmas.

Lemma 1. *Let G be a temporal graph with lifetime τ, P be the set of uncovered node instances at time t, $0 \leq t < \tau$. Let D be any permanent dominating set for P. Then there must be at least one node u in P, such that $CD_u^t \geq \frac{|P|}{|D|}$.*

Proof. We prove the lemma by contradiction. Suppose that there are no such node in G with dominance at time t greater than or equal to $\frac{|P|}{|D|}$. So every node has dominance less than $\frac{|P|}{|D|}$ at time t. All $|D|$ nodes of the permanent dominating set are chosen from P. As dominance of any node does not increase with time so the total number of node instances dominated by D is less than $|P|$ which shows that D does not dominate all $|P|$ node instances. This is a contradiction. Hence the lemma holds. □

Lemma 2. *Let G be a temporal graph, OPT be the optimal permanent dominating set for G. We run algorithm GreedyPDS on G and let N_k be the uncovered node instances remaining after $(k-1)$ rounds of GreedyPDS. Let v_k be the selected node at the k^{th} round then $CD_{v_k}^{t_k} \geq \frac{|N_k|}{|OPT|}$.*

Proof. We prove this lemma by induction.

- ***Base Case:*** Base case of this lemma is the first round of node selection. At first round of node selection by *GreedyPDS*, as all node instances of G are uncovered, from Lemma 1, we know that there exists at least one node which has dominance greater than or equal to $\frac{|N_1|}{|OPT|}$. As *GreedyPDS* selects the node with maximum dominance for any $k = 1, 2, \cdots, l$ where l is the size of permanent dominating set, so $CD_{v_1}^{t_1} \geq \frac{|N_1|}{|OPT|}$.
- ***Inductive Step:*** Let this result hold upto m rounds. We have to show that this lemma holds for $(m+1)^{th}$ round as well. At $(m+1)^{th}$ round we have the set $N_{m+1} \subset N_1$ of uncovered node instances and m nodes which are already selected by *GreedyPDS* as member of the permanent dominating set. According to the algorithm these m nodes do not dominate any instances from N_{m+1}. Now as $N_{m+1} \subset N_1$ and OPT is a dominating set for N_1, it is also a dominating set for N_{m+1}. There are three possible cases:
  - **Case 1:** *No node from the m nodes are in OPT*: For this case all nodes of OPT are selected from unselected nodes and it is currently dominating N_{m+1}. So from Lemma 1 it holds that $CD_{v_{m+1}}^{t_{m+1}} \geq \frac{|N_{m+1}|}{|OPT|}$.
  - **Case 2:** *Some nodes of OPT are from m selected nodes and some from unselected nodes:* For this case let there be p nodes of OPT which are from the m selected nodes. So $|OPT| - p$ nodes of OPT dominate all N_{m+1} uncovered node instances as m selected nodes do not dominate any instances of N_{m+1}. So from Lemma 1, $CD_{v_{m+1}}^{t_{m+1}} \geq \frac{|N_{m+1}|}{|OPT|-p}$. This shows that $CD_{v_{m+1}}^{t_{m+1}} \geq \frac{|N_{m+1}|}{|OPT|}$ as $p > 0$.

Case 3: *All nodes of OPT are from m selected nodes:* This case is not possible because, then it will not be able to dominate N_{m+1} uncovered node instances which is a contradiction.

In all cases the stated lemma holds. □

Theorem 3. *GreedyPDS is a* $\ln(n\tau)$*-approximation algorithm.*

Proof. Let there be n nodes in the given temporal graph G and its lifetime be τ. So after the construction of the permanent dominating set D, for all $n\tau$ node instances of G, either those node instances or their neighbours should be in D. Let OPT be the optimal permanent dominating set and $GPDS$ be the permanent dominating set returned by *GreedyPDS* for G and $|GPDS| = l$.

GreedyPDS selects a node at each round and this process continues upto l rounds. Let the i^{th} round of node selection happens at time t_i. Let before the selection at k^{th} round the number of uncovered node instances be n_k. So $n_1 = n\tau$ and $n_{l+1} = 0$. At the first iteration which occurs at t_1, according to our greedy strategy, the node with the highest dominance is added to $GPDS$. As OPT optimally dominates $n\tau$ node instances, the size of the optimal permanent dominating set for uncovered node instances at t_1 is $|OPT|$. Then the average number of node instances dominated by each node of OPT is $\frac{n\tau}{|OPT|}$. So from Lemma 1 there must be at least one node with dominance at that point of time greater than or equal to $\frac{n\tau}{|OPT|}$. So if the selected node with the maximum dominance is v_1 and dominance of the node v_1 at time t_1 is $CD_{v_1}^{t_1}$, then:

$$CD_{v_1}^{t_1} \geq \frac{n\tau}{|OPT|} \implies \frac{1}{CD_{v_1}^{t_1}} \leq \frac{|OPT|}{n\tau}$$

Since OPT is also a permanent dominating set for n_k number of uncovered node instances therefore, from Lemmas 1 and 2, there is at least one node u such that CD_u greater than or equal to $\frac{n_k}{|OPT|}$. So for the k^{th} round of node selection if the selected node is v_k, it can be written that:

$$\begin{aligned}\frac{1}{CD_{v_k}^{t_k}} &\leq \frac{|OPT|}{n_k} \\ 1 &\leq \frac{CD_{v_k}^{t_k}}{n_k}.|OPT| \\ &\leq \frac{n_k - n_{k+1}}{n_k}.|OPT|\end{aligned}$$

Taking summation from $k = 1$ to l on both sides of the inequality.

$$\sum_{k=1}^{l} 1 \leq \sum_{k=1}^{l} \frac{n_k - n_{k+1}}{n_k}.|OPT| \tag{1}$$

$$\begin{aligned} l &\leq |OPT|.\sum_{k=1}^{l}\left(\frac{1}{n_k} + \frac{1}{n_k - 1} + \cdots + \frac{1}{n_{k+1} + 1}\right) \\ &= |OPT|.\sum_{i=1}^{n\tau}\frac{1}{i} = \ln(n\tau).|OPT| \end{aligned} \tag{2}$$

Equation 2 comes from Eq. 1 from the fact that $\frac{1}{n_k} \leq \frac{1}{n_k - i}$ for each $0 \leq i < n_k$. The final result shows that *GreedyPDS* is a $\ln(n\tau)$-approximation algorithm. □

4.1 Experimental Results

We have evaluated the proposed algorithm by running it on some real life data sets representing different types of networks that capture the mobility patterns of mobile devices [2], contacts between people [7,18], and communications between different autonomous systems [12]. The work in [2] reports experiments done to find the communication opportunities between mobile devices distributed between the participants of INFOCOM'06. A permanent dominating set on this network will identify the devices that can be used to disseminate information to all other devices. Similarly, the works in [7,18] represent networks to model contacts between students and [12] reports data transmission between autonomous systems in the internet.

Each data set contains contact information for a total of T time, where T varies between the data sets. For each such data set, we have considered several values of $\rho < T$, and divided the total time into $\tau = \frac{T}{\rho}$ subintervals, each of which is taken as a single timestep for the temporal graph to be constructed. The static graph for each timestep is constructed with edges added between two nodes if the nodes come into contact at least once in that timestep (i.e., in the subinterval ρ). This gives a temporal graph with lifetime τ. The permanent dominating set of the resultant temporal graph is then computed. Note that an obvious lower bound on the size of the permanent dominating set of any temporal graph is the number of isolated nodes in the graph, as such nodes must be included in the set. A lower value of ρ increases the number of isolated nodes, and hence the values of ρ are chosen such that the number of isolated nodes are not very high.

The first data set contains $T = 342915$ s of contact information between 98 imotes distributed among the participants of INFOCOM'06 [2]. The values of ρ considered for this data set are 25000, 30000, 35000 and 40000 s. The second data set from SocioPatterns contains $T = 61960$ s of contact information between students in a primary school [7,18]. The values of ρ considered for this data set are 5400, 7200, 9000, 10800 s. The third and fourth data sets are collected from SNAP [13] autonomous systems graph, as-733 [12] and as-caida [12]. The as-733 data set has 733 daily instances of communication data between different nodes of different autonomous systems on the internet. We have taken $T = 50$ days of communication data, with $\rho = 1$, 2, 3, and 4 days (T was taken to be 51 and 48 days for $\rho = 3$ and 4 respectively to get an integer number of timesteps). The as-caida data set has 122 instances of communication data between different nodes of different autonomous systems on the internet. Each communication instance contains the communication data on a particular day over the years 2004 to 2007. The values of ρ considered for this data set are 1, 2, 3, and 4 months. We have merged all available data in a single month to create communication data for a single month. Thus we have got $T = 47$. For $\rho = 2$, 3 and 4, the last timestep contains data of 1, 2 and 3 months respectively. Thus, for each data set, four

Table 1. Results of running *GreedyPDS* on three different data sets

Data sets	Interval length (ρ)	No. of nodes	No. of edges	Lifetime (τ)	No. of isolated nodes	Size of PDS		$\ln(n\tau)$
						Naive approach	GreedyPDS	
INFOCOM'06	25000 s	98	4398	14	53	75	60	7.224
	30000 s	98	4398	12	48	72	56	7.069
	35000 s	98	4398	10	36	59	47	6.887
	40000 s	98	4398	9	37	62	47	6.782
SocioPatterns	5400 s	242	8317	12	91	203	128	7.973
	7200 s	242	8317	10	87	189	123	7.791
	9000 s	242	8317	7	121	216	139	7.434
	10800 s	242	8317	6	62	135	92	7.281
as-733, SNAP	1 day	3328	7167	50	626	1530	1422	12.022
	2 days	3328	7167	25	407	1315	1238	11.329
	3 days	3328	7167	17	383	1283	1211	10.943
	4 days	3328	7167	12	352	1244	1174	10.595
as-caida, SNAP	1 month	31379	101945	47	20512	23235	21946	14.204
	2 months	31379	101945	24	18364	21257	19949	13.531
	3 months	31379	101945	16	17874	20716	19460	13.126
	4 months	31379	101945	12	17463	20274	19040	12.838

different temporal graphs are constructed with different lifetime (τ) values. The first six columns of Table 1 shows the details of the temporal graphs formed.

The size of the permanent dominating set obtained by running the *GreedyPDS* algorithm on these temporal graphs is compared with two other values, the lower bound obtained from the number of isolated nodes, and the size of a permanent dominating set obtained by using a greedy approach for computing dominating set for a static graph based on the most number of uncovered neighbours of a node [15]. In this approach, the dominating set for each static graph at every timestep of the lifetime of the given temporal graph is computed first. Then the union of all these static dominating sets is taken to construct the permanent dominating set of the temporal graph. Note that the lower bound using isolated nodes is a loose bound; however, it can still serve as a worst case reference for comparing the size of the sets obtained.

Table 1 shows the results of our experiments. The results show that for the first two and last data sets, the size of the permanent dominating set obtained by *GreedyPDS* is within twice the lower bound, while for the third data set it is within four times the lower bound. The size of the permanent dominating set obtained for all cases is much better than the worst case size obtained from the theoretical bounds proved. The results also clearly show that the *GreedyPDS* algorithm outperforms the naive algorithm in all cases.

The reason for the relatively higher size of the permanent dominating set for the third data set is as follows. As can be seen from the table, the edge to node ratio for the third data set is 2.153, while for the first two data sets, this ratio

44.877 and 34.367 respectively. This shows that the graphs generated from the first two data sets are more densely connected than those generated from the third data set. The relatively sparse graphs of the third data set result in the relatively larger size of the permanent dominating set. Though the temporal graphs generated from the last data set are also relatively sparse in nature with edge to node ratio 3.249, but for this data set, in all cases, more than half of the nodes are isolated and all are included in the permanent dominating set. This results in a permanent dominating set of relatively smaller size for the last data set.

5 Finding Approximate Maximum *k*-Dominant Node Set for Temporal Graphs

In this section, the problem of finding maximum k-dominant node set for a temporal graph $G(V, E)$ has been addressed. It can be seen that if we select a single node, $v \in V$ as a member of the maximum k-dominant node set it will dominate set of its neighbouring node instances N_v, we refer to it as *dominance set* (DS_v) of v and $|DS_v|$ is the total dominance of v. Similarly the dominance set of a set $S \subseteq V$ is $DS_S := \bigcup_{v \in S} DS_v$ and total dominance of S is $|DS_S|$.

Total dominance can also be expressed as a function $f : \{A : A \subseteq V\} \rightarrow \mathbb{Z}_+$, $f(A) = |\bigcup_{v \in A} DS_v|$. Then the problem reduces to finding a set A with cardinality of a given positive integer $k \leq |V|$, for which $f(A)$ is maximum in the given temporal graph G. This problem is a NP-Hard problem [14].

The proposed greedy approximation algorithm uses the same steps as algorithm of *GreedyPDS* with the only difference that this time the algorithm stops either when the number of selected nodes becomes k or all node instances of G are dominated. We refer to this algorithm as *GreedyKDS*.

Lemma 3. *If $A \subset B \subseteq V$ for a given temporal graph $G(V, E)$ then, $DS_A \subseteq DS_B$.*

Proof. We prove this lemma by contradiction. Given that $A \subset B \subseteq V$, let $DS_A \nsubseteq DS_B$. This means that there is atleast one node instance u^t, where $0 < t \leq \tau$, such that $u^t \in DS_A$ and $u^t \notin DS_B$. From the definition of dominance set we know that either $u^t \in A$ or a neighbour of u^t belongs to A. This implies that there is atleast one instance of a node belonging to set A that does not belong to B. This contradicts the given statement $A \subset B$ because when one node is included in a set, all instances of that node are included in that set. Hence $DS_A \subseteq DS_B$. □

Lemma 4. *f is a nondecreasing submodular function.*

Proof. It is straight forward to show from Lemma 3 that f is a nondecreasing function. Next we prove that f is a submodular function.

Let there be two sets A and B such that $A \subset B \subset V$ and u be a node, such that $u \in V$ and $u \notin B$. We need to prove that $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$. The following cases are possible:

– ***Case 1:*** $(DS_u \cap DS_B) = \emptyset$ ***:***
For this case,

$$DS_u \cap DS_B = \emptyset \Rightarrow DS_u \cap DS_A = \emptyset$$

So,

$$\begin{aligned} f(B \cup \{u\}) - f(B) &= |DS_u \cup DS_B| - |DS_B| \\ &= |DS_u| + |DS_B| - |DS_B| \\ &= |DS_u| \end{aligned}$$

$$\begin{aligned} f(A \cup \{u\}) - f(A) &= |DS_u \cup DS_A| - |DS_A| \\ &= |DS_u| + |DS_A| - |DS_A| \\ &= |DS_u| \end{aligned}$$

So for this case,

$$f(A \cup \{u\}) - f(A) = f(B \cup \{u\}) - f(B) \tag{3}$$

– ***Case 2:*** $(DS_u \cap DS_B) \neq \emptyset$ ***:***
For this case, the following subcases are possible:

- ***Case 2a:*** $(DS_u \cap DS_B) = (DS_u \cap DS_A)$ ***:***
For this scenario,

$$\begin{aligned} f(A \cup \{u\}) - f(A) &= |DS_u \cup DS_A| - |DS_A| \\ &= |DS_u| + |DS_A| - |DS_u \cap DS_A| - |DS_A| \\ &= |DS_u| - |DS_u \cap DS_A| \end{aligned}$$

$$\begin{aligned} f(B \cup \{u\}) - f(B) &= |DS_u \cup DS_B| - |DS_B| \\ &= |DS_u| + |DS_B| - |DS_u \cap DS_B| - |DS_B| \\ &= |DS_u| - |DS_u \cap DS_B| \\ &= |DS_u| - |DS_u \cap DS_A| \end{aligned}$$

So for this case,

$$f(A \cup \{u\}) - f(A) = f(B \cup \{u\}) - f(B) \tag{4}$$

- ***Case 2b:*** $(DS_u \cap DS_B) \supset (DS_u \cap DS_A)$ ***:***
For this scenario,

$$\begin{aligned} f(A \cup \{u\}) - f(A) &= |DS_u \cup DS_A| - |DS_A| \\ &= |DS_u| + |DS_A| - |DS_u \cap DS_A| - |DS_A| \\ &= |DS_u| - |DS_u \cap DS_A| \end{aligned}$$

$$\begin{aligned} f(B \cup \{u\}) - f(B) &= |DS_u \cup DS_B| - |DS_B| \\ &= |DS_u| + |DS_B| - |DS_u \cap DS_B| - |DS_B| \\ &= |DS_u| - |DS_u \cap DS_B| \end{aligned}$$

As $(DS_u \cap DS_B) \supset (DS_u \cap DS_A) \Rightarrow |DS_u \cap DS_B| > |DS_u \cap DS_A|$, then,

$$f(A \cup \{u\}) - f(A) > f(B \cup \{u\}) - f(B) \tag{5}$$

- ***Case 3:*** $(DS_u \cap DS_B) \subset (DS_u \cap DS_A)$ ***:***
  As $A \subset B$, from Lemma 3 this scenario is not possible.
- ***Case 4:*** $(DS_u \cap DS_B) \not\supseteq (DS_u \cap DS_A)$ ***:***
  As $A \subset B$, from Lemma 3 this is also an impossible scenario.

So from (3), (4) and (5), it can be said that that,

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \tag{6}$$

Hence f is a nondecreasing submodular function. □

Theorem 4. *Algorithm GreedyKDS is a* $(1-\frac{1}{e})$*-approximation algorithm, where* e *is the base of natural logarithm.*

Proof. According to Lemma 4, f is a nondecreasing submodular function. Algorithm *GreedyKDS* maximizes the function f with the constraint that the cardinality of the resultant set is bounded by k. It has been proved (Proposition 4.1, 4.2 and 4.3, Lemma 4.1 and Theorem 4.1 and 4.2) in [14] that if a nondecreasing submodular function maximization problem with cardinality constraint is solved with greedy approach, then the approximation ratio of the greedy algorithm is $(1 - \frac{1}{e})$. Hence, using the results in [14] we can say that *GreedyKDS* is a $(1 - \frac{1}{e})$-approximation algorithm. If the algorithm terminates before k rounds the resultant solution is the optimal one. □

6 Conclusion

In this paper, we have investigated permanent dominating sets and its applications in highly dynamic networks. In particular, we have presented a $\ln(n\tau)$-approximation algorithm for finding a minimum permanent dominating set and a $(1 - \frac{1}{e})$-approximation algorithm for finding the maximum k-dominant node set for a given temporal graph. The first algorithm has also been simulated on four real life data sets for performance evaluation which supports the theoretical results.

References

1. Casteigts, A., Flocchini, P.: Deterministic algorithms in dynamic networks: Problems, analysis, and algorithmic tools, Technical report, DRDC 2013-020 (2013)
2. Chaintreau, A., Hui, P., Scott, J., Gass, R., Crowcroft, J., Diot, C.: Impact of human mobility on opportunistic forwarding algorithms. IEEE Trans. Mobile Comput. **6**, 606–620 (2007)
3. Dubois, S., Kaaouachi, M., Petit, F.: Enabling minimal dominating set in highly dynamic distributed systems. In: Stabilization, Safety, and Security of Distributed Systems - 17th International Symposium, SSS, 18–21 August, pp. 51–66 (2015)
4. Ferreira, A.: On models and algorithms for dynamic communication networks: the case for evolving graphs. In: 4e Rencontres Francophones sur les Aspects Algorithmiques des Telecommunications (ALGOTEL), pp. 155–161 (2002)
5. Gao, J., Guibas, L.J., Hershberger, J., Zhang, L., Zhu, A.: Discrete mobile centers. In: 17th Annual Symposium on Computational Geometry, 3–5 June, pp. 188–196 (2001)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
7. Gemmetto, V., Barrat, A., Cattuto, C.: Mitigation of infectious disease at school: targeted class closure vs school closure. BMC Infect. Dis. **14**, 695 (2014)
8. Guibas, L.J., Milosavljevic, N., Motskin, A.: Connected dominating sets on dynamic geometric graphs. Comput. Geom. **46**, 160–172 (2013)
9. Hershberger, J.: Smooth kinetic maintenance of clusters. In: 19th ACM Symposium on Computational Geometry, 8–10 June, pp. 48–57 (2003)
10. Jain, S., Fall, K.R., Patra, R.K.: Routing in a delay tolerant network. In: ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 145–158 (2004)
11. Kostakos, V.: Temporal graphs. Phys. A **388**, 1007–1023 (2009)
12. Leskovec, J., Kleinberg, J.M., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 21–24 August, pp. 177–187 (2005)
13. Leskovec, J., Krevl, A.: SNAP datasets: stanford large network dataset collection, June 2014. http://snap.stanford.edu/data
14. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions-i. Math. Program. **14**, 265–294 (1978)
15. Parekh, A.K.: Analysis of a greedy heuristic for finding small dominating sets in graphs. Inf. Process. Lett. **39**, 237–240 (1991)
16. Ros, F.J., Ruiz, P.M.: Minimum broadcasting structure for optimal data dissemination in vehicular networks. IEEE Trans. Veh. Technol. **62**, 3964–3973 (2013)
17. Siljak, D.: Dynamic graphs. Nonlinear Anal. Hybrid Syst. **2**, 544–567 (2008)
18. Stehl, J., Voirin, N., Barrat, A., Cattuto, C., Isella, L., Pinton, J., Quaggiotto, M., Van den Broeck, W., Rgis, C., Lina, B., Vanhems, P.: High-resolution measurements of face-to-face contact patterns in a primary school. PLOS ONE **6**, e23176 (2011)
19. Stojmenovic, I., Seddigh, M., Zunic, J.: Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. IEEE Trans. Parallel Distrib. Syst. **13**, 14–25 (2002)
20. Whitbeck, J., de Amorim, M.D., Conan, V., Guillaume, J.: Temporal reachability graphs. In: 18th Annual International Conference on Mobile Computing and Networking, Mobicom 2012, 22–26 August, pp. 377–388 (2012)