Journal of Computer Security 21 (2013) 317–346
DOI 10.3233/JCS-130465
IOS Press

Optimizing mix-zone coverage in pervasive wireless networks [1]

Murtuza Jadliwala [a,*], Igor Bilogrevic [b] and Jean-Pierre Hubaux [b]
[a] *Wichita State University, Wichita, KS, USA*
E-mail: murtuza.jadliwala@wichita.edu
[b] *LCA1, EPFL, Lausanne, Switzerland*
E-mails: {Igor.Bilogrevic, Jean-Pierre.Hubaux}@epfl.ch

Location privacy is a major concern in pervasive networks where static device identifiers enable malicious eavesdroppers to continuously track users and their movements. In order to prevent such identifier-based tracking, devices could coordinate regular identifier change operations in special areas called *mix-zones*. Although mix-zones provide spatio-temporal de-correlation between old and new identifiers, depending on the position of the mix-zone, identifier changes can generate a substantial inconvenience (or "cost") to the users in terms of lost communications and increased energy consumption. In this paper, we address this trade-off between privacy and cost by studying the problem of determining an optimal set of mix-zones such that the degree of mixing in the network is maximized and the overall network-wide mixing cost is minimized. We follow a graph-theoretic approach and model the optimal mixing problem as a generalization of the vertex cover problem, called the *Mix Cover* (*MC*) problem. We propose three approximation algorithms for the MC problem and derive a lower bound on the solution quality guaranteed by them. Additionally, we outline two other heuristics for solving the MC problem. These heuristics are simple, but do not provide any guarantees on the solution quality. By means of extensive empirical evaluation using real data, we compare the performance and solution quality of these algorithms. The combinatorics-based approach used in this work enables us to study the feasibility of determining optimal mix-zones regularly and under dynamic network conditions.

Keywords: Mix-zones, location privacy, pervasive networks

1. Introduction

Rapid proliferation of embedded and portable wireless and mobile computing technology has enabled a variety of pervasive and context-aware data-sharing applications including vehicular safety messaging [41,53], pervasive or local-area social networking [1,10,46], dating [31,32,38], personal safety [44] and micro-blogging [20] applications. Wireless devices in such networks, e.g., mobile phones

[1] An earlier version of this paper titled "Optimizing mixing in pervasive networks: A graph-theoretic perspective" appeared in the *Proceedings of the 16th European Symposium on Research in Computer Security*, Leuven, Belgium, September 12–14, 2011.

[*] Corresponding author: Murtuza Jadliwala, Wichita State University, Wichita, KS 67226, USA. E-mail: murtuza.jadliwala@wichita.edu.

or vehicle on-board units, can communicate directly with each other using wireless peer-to-peer technology or they can communicate with each other or third-party service providers by means of a deployed base station infrastructure.

Users of such pervasive applications continuously face location privacy risks at the hands of malicious eavesdroppers and curious service providers. Users' location information revealed as a result of this threat can be used by malicious parties to track their movements [9], preferences [19] or social affiliations [8]. It can be also used to identify users and their availabilities by inferring their home/work locations [25], which can be later used for accomplishing malicious goals [33]. Third-party service providers, however, are generally trusted and claim to utilize the collected personal and location information to further enhance context-aware services but can inadvertently harm users' privacy if the collected data is improperly shared with commercial partners or leaked in an unauthorized fashion.

One frequently mentioned strategy to overcome these concerns, inspired by Chaum's seminal work on mix networks [13,14], is to regularly mix [40] or change [12,34] device identifiers including application, IP and device MAC addresses. Most cellular networks have adopted a similar approach; they identify a subscriber's device with a Temporary Mobile Subscriber Identity or TMSI which changes every time the subscriber moves to a new geographical area.

In order to maximize location anonymity, mixing or changing of user identifiers should occur in a spatiotemporal region, called *mix-zone* [6,7]. While nodes are in the mix-zone, they refrain from transmitting any information (or identifiers); on leaving the mix-zone, communication resumes with a new identifier or pseudonym for each user or device. Mix-zones serve to mix or provide de-correlation between pseudonyms and device associations, which makes it difficult for an adversary to continuously track users by linking the device and its pseudonym. In most real-life application scenarios, users move on a fixed (pre-defined) network of roads, for example, vehicular or pedestrian hand-held networking scenarios. Thus, it is important to study the effectiveness of mix-zone deployment from the perspective of such road networks. Earlier research [11] has shown that mix-zones are most effective (in protecting location privacy) when they are set at points with higher input and output ports, such as *road intersections*.

Although effective in improving the privacy of users, the pseudonym-change (or mix) operation is not free and induces a cost on the network (and its users). This cost is determined by factors such as the significance of the intersection (at which a mix-zone is located) to users and the network, traffic intensity at the intersection (both entering and leaving) and intersection context, for example, time-of-day. This cost is primarily due to the communication loss caused by routing disruptions [48] or silent periods [34] and the loss of computational resources, such as energy, caused by the pseudonym change operation itself and its related side-effects.

This results in an interesting trade-off between the number of mix-zones that can be deployed on the road network for privacy enhancement and the resulting cost due to such a deployment. An ideal situation from the privacy perspective, although

infeasible from the cost point-of-view, is to deploy a mix-zone at each and every intersection of the road network under consideration. Such a deployment of mix-zones is trivial in theory, but difficult to realize and sustain in practice due to the resulting costs. A more realistic and feasible goal would be to maximize the coverage (of roads) of the deployed mix-zones, and hence the privacy provided by them, and to minimize the associated costs due to such a deployment. Moreover, the goal is not only to determine such an optimal and cost-efficient placement of mix-zones, but also to study if there are algorithms that can find such a solution efficiently (in computation time and space). As pseudonym change costs at intersections are highly dynamic and depend on factors such as intersection context and traffic intensity, which continuously change over time, there is a need to regularly determine the most cost-efficient set of mix-zones. Before designing efficient algorithms, we first need to gain a thorough theoretical understanding of the problem from a combinatorial perspective.

In this paper, we model the problem of optimal mix-zone placement as a graph-based optimization problem where roads are represented as graph edges and intersections as vertices. Each vertex is weighted based on the cost (per device) of mix-zone placement at the corresponding intersection. Each edge is weighted based on the traffic intensity (or mix-zone demand) of the corresponding road in *each direction*. The problem of optimal mix-zone placement – we refer to it as the *Mix Cover problem* (*MC*) – is then to determine a set of intersections (or vertices) for mix-zone placement, such that each road (connecting two intersections) in the network is associated with at least one mix-zone and the network-wide mixing cost due to such a placement of mix-zones is minimized. The mix cover problem is a generalization of a well-known problem in combinatorics, called the *Vertex Cover* (*VC*) problem [37]; the *Facility Terminal Cover* (*FTC*) problem discussed in [52] is a special case of the mix cover problem. To the best of our knowledge, the mix cover generalization, specifically in the setting of pervasive networks, has never been studied before. In this paper, we show that the Mix Cover problem is a combinatorially hard problem and propose three bounded-ratio approximation algorithms for the same. The first algorithm is based on a linear programming relaxation of an Integer Program (IP) formulation of the problem, whereas the remaining two algorithms take advantage of the "divide and conquer" strategy which was used in [52] to solve the FTC problem. We analytically study the solution quality and running-time guarantees of these algorithms by deriving their worst-case approximation ratio and running-time, respectively. These algorithms are able to guarantee a solution quality on specific graph instances of the Mix Cover problem, i.e., graph instances where the demand values are greater than or equal to 2. But in the Mix Cover problem, as the demand values of the graph instances are derived from (or represent) the traffic intensities on roads connecting intersections, they can be accordingly scaled to meet this requirement. We also outline two other heuristics for solving the mix-cover problem; the first is based on a greedy strategy, while the second is based on a simulated annealing strategy. These heuristics are straightforward implementation-wise and run much

more efficiently, but do not provide any guarantees on the solution quality. Finally, we perform an extensive comparative analysis of the performance of the proposed algorithms by evaluating them on real road-traffic data of three US states.

2. Background and related work

In the following section, we provide a brief overview of some concepts from complexity theory and combinatorial optimization that we use throughout the paper. After that, we outline other related research efforts on the optimal mix-zone placement problem.

2.1. *Preliminaries*: *Combinatorial hardness and approximations*

A decision problem S is said to have an *efficiently verifiable proof system* if there exists a polynomial p and a polynomial-time verification algorithm V such that the following two conditions hold:

- Completeness: For every input $x \in S$, there exists y of length at most $p(|x|)$, i.e., polynomial in terms of the size of the input, such that $V(x, y) = 1$.
- Soundness: For every $x \notin S$ and every y, it holds that $V(x, y) = 0$.

The class *NP* is the class of decision problems that have an efficiently verifiable proof system. A polynomial-time computable function f is called a *Karp-reduction* of S to S' (in other words, S is Karp-reducible to S') if, for every x, it holds that $x \in S$ if and only if $f(x) \in S'$. A set S is *NP-complete* if it is in *NP* and every set in *NP* is Karp-reducible to it. A set S is *NP-hard* if every set in *NP* is Karp-reducible to it, but its membership within *NP* is not known. It is not known whether every problem in *NP* can be efficiently (in polynomial time) solved. But, if any single problem in the set of *NP*-hard problems can be solved efficiently, then every problem in *NP* can also be solved efficiently. Thus, *NP*-hard problems are considered "harder" than *NP* problems in general, and are believed to have no polynomial-time exact solutions. Algorithms for such hard problems, also called *optimization problems*, that run in polynomial time and produce a near-exact or sub-optimal solution are called *approximation algorithms*. Approximation algorithms that can guarantee that the solution output by them can be no more (if minimization problem) or less (if maximization problem) than a factor σ times the optimal solution are called σ*-approximation algorithms*. More details on these topics can be found in [21,24].

2.2. *Location privacy and the mix-zone placement problem*

The idea of using mix-zones as a means to improve the location privacy of mobile devices in road networks has been well established in the literature [11,17,34]. Freudiger et al. [18] were the first to study and formulate the problem of optimal

mix-zone placement in road networks. In their work, the authors measure the *effectiveness* of mixing by measuring the probability of error of an adversary in correctly assigning exiting flows to their corresponding entering flows at a mix-zone. By using linear programming, they determine an optimal set of mix-zones that maximize the overall mixing effectiveness. In contrast, our model and solution is more general as we study the trade-off between maximizing the *coverage* of mix-zones and minimizing their *deployment cost*.

In another related effort, Alpcan and Buchegger [2] use game theory to model the attack and optimal defense strategies of the adversary and users in vehicular networks. Humbert et al. [36] also study the problem of optimal mix-zone placement from a game-theoretic perspective. They model the problem of mix-zone placement as a game between mobile users who want to protect their privacy and a local adversary who wants to track them by strategically placing eavesdropping stations. In [36], the authors focus on deriving mix-zone deployment strategies locally at each intersection, whereas in our work, we study the problem of achieving a globally optimal deployment strategy. Palanisamy et al. [43] propose a framework and a suite of algorithms for mix-zone construction, which considers the inherent characteristics of road networks. Similar to earlier results, these mix-zone deployment strategies protect against specific adversarial attacks and only consider local intersection parameters for mix-zone deployment.

There has been extensive research on techniques for analyzing the extent and implications of location privacy loss from collected (or disclosed) location traces. For example, Bindschaedler et al. [9] examine location traces collected from a real experiment consisting of smartphone users in order to study the effect of deterministic identifier mixing on the anonymity of user movements. One of the widely used metric to quantify the trace anonymity of users' is k-anonymity, which guarantees that a particular user's location (or trace) is indistinguishable from k other users. Nergiz et al. [42] adopt the notion of k-anonymity (from databases) to trajectories and propose a novel generalization-based approach for anonymization of trajectories. They show that releasing anonymized trajectories in such a fashion may still have some privacy leaks and propose a randomization-based reconstruction algorithm in order to overcome those leaks. In another related effort, Hoh et al. [29] outline a new metric for quantifying the location privacy of users in location traces which is based on the length of time a user can be successfully tracked. They further propose a privacy-preserving algorithm to maximize (based on this metric) the anonymity of users in the collected location traces. In our work, contrary to the above approaches, we focus more on real-time privacy guarantees rather than guarantees on collected location traces. Gedik et al. [22] propose a system for guaranteeing a personalized level of anonymity in Location-based Services (LBS), but their approach is restricted to real-time location-based queries whereas we focus on a pervasive scenario where any communication sent by a user can be used to infer its position.

In this paper, we make the following novel contributions. We study the problem of optimal mix-zone deployment from a global (network-wide) perspective. Our network model and problem formulation is general enough and can be easily extended

to include other privacy metrics [49,50], in addition to the basic mix-zone coverage guarantee. The analytical results obtained in this paper would help shed light on the feasibility of determining an optimal mix-zone deployment autonomously by mobile devices in dynamic real-time road-network settings. Finally, the results presented in this paper are significant from the combinatorics viewpoint.

3. Problem statement

3.1. System model

We consider a pervasive networking scenario where mobile users (or vehicles), equipped with wireless communication devices that are capable of communicating both in ad-hoc and infrastructure mode, obtain context-based services on their devices. Examples of such networking systems include, but are not limited to, wireless peer-to-peer mobile-phone based social networking platforms, such as Nokia Instant Community (NIC) [46], and vehicle-based wireless communication systems or VANETs [23]. Each mobile device in the network includes some identifying information, such as a MAC address or an application-level identifier, in all of its communications. Such identifiers or pseudonyms are used for identifying the device and for routing communications within the network [45].

In order to prevent trivial tracking by an eavesdropping adversary, wireless devices regularly change their identifiers or pseudonyms. Various techniques for privacy protection, which use multiple pseudonyms or identifiers, have been studied in the literature [7,11,12,40]. In order to prevent trivial linkability of old and new pseudonyms and achieve spatio-temporal de-correlation between users and their identifiers, devices must coordinate their pseudonym changes with other physically co-located or neighboring devices. Such regions for achieving spatial and temporal de-correlation of devices and (old and new) pseudonyms are also referred to as *mix-zones* [7]. In a mix-zone, spatial de-correlation is achieved by *colocated* mobile device(s) changing their pseudonyms in a coordinated fashion while temporal de-correlation is achieved by either (i) remaining silent for a short random period of time [34], (ii) by encrypting communications after the pseudonym change operation [17], or (iii) by means of a mobile proxy [47]. Mix-zones can be *passive* or *active*, depending on the actions taken by the devices immediately after the pseudonym change operation [36]. We assume that an off-line Certification Authority (CA), run by an independent trusted third-party, loads the mobile devices with a set of pseudonyms prior to deployment.

Road *intersections* are considered as good spots for mix-zone deployment in road networks because they provide higher spatio-temporal de-correlation as compared to other areas [18,36]. However, mix-zones incur significant overhead [48] and must be optimally placed (with appropriate parameters [35]) in order to reduce the cost induced on the end-users and to provide high location privacy (or high user-identifier de-correlation). The cost of deploying a mix-zone at any intersection can

be a weighted sum of various factors, including the extra resource requirements of devices for mixing and the resulting communication disruption due to mixing at that intersection. We do not quantify these parameters in this work, but we can use existing results in the literature for representing these costs [36,48].

We assume that all the intersections, over the area under consideration, are connected with each other by a network of road segments. Each road segment can be used to reach either one of the intersection that it connects, i.e., there is a two-way movement of users (or devices, vehicles, etc.) on the road. The *demand* for an intersection on a road segment is the average number of users using that road segment to reach that particular intersection. Thus, each road segment has two demands; one for each intersection connected by that road segment. Accordingly, unidirectional roads have just one demand, i.e., the one in the direction of the intersection; the other demand is zero. For simplicity, we assume that any two intersections are connected only by a single road segment; multiple roads between any two intersection can be combined into a single road by simply adding their respective demands.

3.2. Privacy requirement

Given the system model outlined above, we want to address the problem of determining an optimal selection of intersections for mix-zone deployment such that all the roads in the network are covered and the *overall* cost due to mixing is minimized. In other words, we want to determine the most effective and cost-efficient mixing strategy in large road networks. We say that a road segment is *fully-covered* if and only if *both* the end points (intersections) of the road segment have mix-zones deployed on it, i.e., there is mixing at both intersections of the road. A network is said to be *fully-covered* (or has a *full cover*) if and only if all the road segments in the network are fully-covered.

It is easy to see that in the system model discussed above, a full covering of the network can only be achieved if and only if all the intersections in the network are selected for mixing or mix-zone deployment. Such a mixing or full covering strategy is not only trivial but also ideal from the privacy viewpoint. But from a cost perspective, such a covering may be difficult to achieve (or even infeasible) due to the network size and the network-wide mixing cost.

Let us now define a more general version of the full cover, called the *mix cover*. A network is said to be *mix covered* if and only if each of the road segments in the network have at least one of its intersections where a mix-zone is deployed. A fully-covered network is also mix covered and some of the road segments in a mix covered network may be fully-covered, i.e., both the intersections of the road segment may have mix-zones deployed. From the privacy perspective, a mix covered network can *guarantee* that any user (or device) traversing the road network can traverse *at most two* road segments (or *at most one intersection*) without encountering a mix-zone. From the practical standpoint, a mix cover is a reasonable mixing strategy for most deployment scenarios and adversarial models. We focus on the problem of determining a cost-efficient mix cover by modeling it as a *graph-based optimization problem*, as discussed next.

3.3. Graph-theoretic framework and the Mix Cover (MC) problem

Let us represent the road network described above by an *undirected graph* $G \equiv (V, E, w, d)$. Each intersection on the road network is represented by a vertex $v \in V$ of G, and $|V| = n$ is the total number of intersections (vertices[2]) within the area of the road network under consideration. Each road segment connecting any two intersections u and v is represented by an undirected edge $e \equiv (u, v) \in E$, where E is the set of all edges (or roads) and $|E| = m$ is the total number of roads (edges). There exists only a single edge (u, v) between any pair of vertices u and v in G. Given the undirected graph G, let $w : V \rightarrow \mathbb{R}^+$ be the *cost function* that assigns a positive cost to each vertex. The cost at each vertex represents the average cost (per user) of mix-zone deployment (or mixing) at that intersection; the higher the cost, the higher the amount of communication and device resources spent by each user for mixing at that intersection is. We represent by w_u the cost of a vertex $u \in V$. Let $d : E \rightarrow (\mathbb{R}^+, \mathbb{R}^+)$ be the *demand function* that assigns a pair of positive demands to each edge where each demand value in the pair represents the demand for a particular vertex connected by the edge. This demand pair could represent, in the case of vehicular (or pedestrian) networks, the average traffic (or pedestrian) intensity on the road segment in each direction. For any edge $e(u, v) \in E$, we represent the demand as $d_e = (d_e^u, d_e^v)$, where d_e^u, d_e^v is the demand on edge e for intersections u and v, respectively. The value of $d_e^u = 0$ if u is not one of the end points of the edge e.

Given the above graph representation of the road network, we are interested in the problem of efficiently determining the *optimal* mix cover of the network. Each vertex chosen in the mix cover should be able to handle the demands of all the edges it covers. In other words, each intersection should be able to accommodate even the *largest* demand made at it; we refer to this ability of each intersection as the *capacity* of the mix-zone at that intersection. The capacity at a vertex is zero if there is no mix-zone at that vertex. The optimality criteria is based on an assignment of capacities to vertices or intersections such that the demands of all edges are met and the overall cost minimized. Formally, we can represent the problem of determining the optimal mix cover, referred by us as the *Mix Cover* (*MC*) *problem*, in the graph $G \equiv (V, E, w, d)$ as a generalization of the *Vertex Cover* (*VC*) *problem*. VC is a fundamental problem in graph theory and a vertex cover of an undirected graph $G \equiv (V, E)$ is a subset of vertices $V_C \subseteq V$ which contains at least one vertex of all the edges in E and the VC problem is to determine a vertex cover V_C of the smallest cardinality. The VC problem is *NP-hard* and the decision version of the same is known to be *NP-complete* [37]. The Mix Cover (MC) problem can be formally defined as follows.

Mix Cover. Given an undirected graph $G \equiv (V, E, w, d)$, where w is the cost function associated with the set of vertices and d is the demand function associated with

[2]Readers should note that from this point on we will use "vertex" and "intersections" (similarly, "road segment" and "edge") interchangeably and the intended meaning will be implicit from the context.

the set of edges, determine a subset $V_{MC} \subseteq V$ and a capacity $c(v)$ for each vertex $v \in V_{MC}$ such that for each edge $e \equiv (u, v) \in E$ at least one of the vertices u or v is in V_{MC} and associated with a capacity $c(u) \geqslant d_e^u$ or $c(v) \geqslant d_e^v$ respectively, and the total weighted cost, $\sum_{x \in V_{MC}} c(x)w_x$, of all vertices in V_{MC} is minimized.

Thus, given a graph $G \equiv (V, E, w, d)$ of the road network, the MC problem determines a mix cover of the network such that the overall (network-wide) weighted cost of the mix cover is minimized. The total intersection cost at each intersection v is the intersection mixing cost w_v times the capacity $c(v)$ at v. The capacity at any intersection v is at least the maximum demand at that intersection from all roads covered by it. The overall (network-wide) weighted cost of the mix cover is the sum of all the total intersection costs at each intersection in the mix cover. Figure 1 shows one such feasible solution.

The MC problem is very similar to another generalization of the VC problem called the Facility Terminal Cover (FTC) problem [28,52], but there is an important difference between the two problems. Given a graph $G \equiv (V, E, w, d)$, where $w : V \rightarrow \mathbb{R}^+$ and $d : E \rightarrow \mathbb{R}^+$ (denoted as w_v and d_e for vertex v and edge e, respectively), the FTC problem is to find a set $V_{FTC} \subseteq V$ and a capacity $c(v)$ for each vertex $v \in V_{FTC}$ such that for each edge $e \equiv (u, v) \in E$ at least one of the vertices u and v is in V_{FTC} and associated with a capacity $c(u) \geqslant d_e$, and the total weighted capacity $\sum_{x \in V_{FTC}} c(x)w_x$ is minimized. As we can see from the FTC problem definition, the assumed graph model assigns only a *single* demand value to every edge. As a result, the selected capacity for covering any edge depends only on the demand value of the edge. This is different from the MC problem where each edge has two demand values and the selected capacity for covering any edge depends

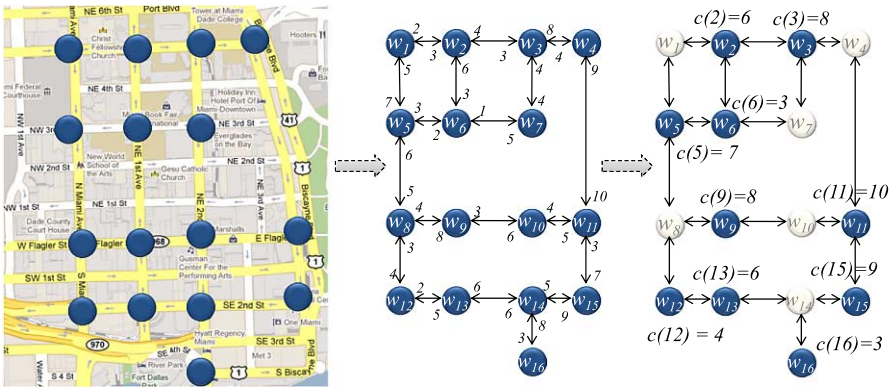

Fig. 1. Mix Cover example on downtown Miami (FL). On the left, the dark circles indicate all intersections where mix-zone placement is possible. The graph representation is shown in the middle and a feasible mix cover is shown on the right, where the dark circles are included in the solution and the shaded circles are not. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-130465.)

on the (demand value associated with the) vertex that is used to cover that edge. The FTC problem can be considered as a special case of the MC problem, i.e., the MC problem reduces to the FTC problem when both the demand values are equal for all the edges in the graph. The formulation and algorithms of the FTC problem cannot be directly used to solve the much more general MC problem; although we will use one of the solution strategies [52] of the FTC problem for solving the MC problem.

There is another generalization of the VC problem called the minimum Generalized Vertex Cover (GVC) problem [27]. In GVC, contrary to VC, an edge incurs a cost (or demand, as in our case) depending on the *number* of its vertices that belong to the solution. Once again, such a generalization of the VC is different from the one that we are interested. In our case, the demand incurred by the edge does not depend on the number of its vertices in the solution, rather it depends on which vertex is included in the solution. To the best of our knowledge, this is the first paper to model and study the problem of optimal mixing or mix-zone placement in pervasive networks as a generalization of the VC problem.

4. Algorithms and combinatorial results

Let us first understand the combinatorial hardness of the MC problem. We derive the following result for the hardness of the MC problem.

Theorem 4.1. *The MC problem is NP-hard.*

The proof of Theorem 4.1 is straightforward, as we can easily reduce any instance of the vertex cover (VC) problem to an instance of the MC problem in polynomial time. This can be done by defining a simple demand function for the graph instance of the vertex cover problem as $d_e \equiv (d_e^u = 1, d_e^v = 1), \forall e$, where $e \equiv (u, v)$ is an edge of the graph instance. Each vertex $u \in V$ can be assigned a weight $w_u = 1$ (or all vertices can also be assigned the same weight value greater than 1). Thus, as the VC problem is NP-hard, we can claim that the MC problem is also NP-hard. The MC problem also seems difficult to approximate and we do not believe it has a Polynomial-Time Approximation Scheme (*PTAS*). This is because the VC problem itself, which is considered to be much simpler than the MC problem, is not believed to have an approximation ratio within 1.3606 unless $P = NP$ [15]. In the following sections, we outline two approximation strategies for the MC problem. The first is based on a linear programming formulation of the problem, whereas the other two algorithms employ a "divide and conquer" strategy by utilizing the round and group approach for solving the FTC problem [52].

4.1. Linear programming algorithm

The MC problem can be easily formulated as an Integer Program (IP), more specifically a 0–1 Program. Let z_e^v be a binary decision variable for each edge e

and its corresponding vertex v which indicates whether the vertex v is included in the mix cover (solution) to cover edge e or not, i.e., $z_e^v = 1$ if edge e is covered by vertex v and $z_e^v = 0$ if not. Let x_v be the decision variable indicating the capacity and w_v indicate the cost of each vertex $v \in V$. Then, the IP formulation of the MC problem can be obtained as follows:

$$\begin{aligned} &\min \sum_{v \in V} w_v x_v, \\ &\quad \text{subject to } z_e^u + z_e^v \geqslant 1, \forall e \equiv (u,v) \in E, \\ &\qquad\qquad x_v \geqslant z_e^v d_e^v, \forall v \in V, e \in E, \\ &\qquad\qquad z_e^v \in \{0,1\}, \forall v \in V, e \in E. \end{aligned}$$

Now, solving an Integer Program is a well-known hard problem [37]. But, efficient (polynomial time) techniques [4] for solving a Linear Program (LP) relaxation of the Integer Program exists. If the LP relaxation has an integral solution then that can also be the solution to the above IP. In general, solving the LP relaxation of the problem can give a fractional feasible solution, from which a feasible (and possibly non-optimal) solution to the above IP can be obtained. The LP relaxation of the problem is as shown below:

$$\begin{aligned} &\min \sum_{v \in V} w_v x_v, \\ &\quad \text{subject to } d_e^v x_u + d_e^u x_v \geqslant d_e^u d_e^v, \forall e \equiv (u,v) \in E, \\ &\qquad\qquad x_v \geqslant 0, \forall v \in V. \end{aligned}$$

Let $(\bar{x}, \{\bar{z}_e \mid \forall e \in E\})$ be an optimal solution to the above LP formulation, where $z_{e,i} = \frac{x_i}{d_e^i}$ is the entry of the vector $\bar{z}_e$ representing the value of the decision variable corresponding to vertex i (to cover edge e), and x_j is the jth entry of $\bar{x}$ and represents the capacity value at the vertex j. The value of $z_{e,i} = 0$ if i is not a vertex in edge e. We can see that any optimal solution $(\bar{x}, \{\bar{z}_e\})$ produced by solving the above LP is a feasible fractional solution to the MC problem. It is also clear that an optimal solution *OPT* to the MC problem is always a feasible solution to the above LP formulation. Thus, the above LP relaxation for the MC problem is correct. Based on this, we can prove the following bound on the approximation quality for the MC problem.

Theorem 4.2. *There exists a polynomial time* 2*-approximation for MC*.

Proof. For the sake of convenience, let us denote the IP formulation of the MC problem as IP-MC and its LP relaxation as LP-MC. It is easy to see that any optimal

solution *OPT* to IP-MC is also a feasible solution to the LP-MC and has the same objective function value. Moreover, LP-MC is indeed a relaxation of IP-MC and an optimal solution $(\bar{x}, \{\bar{z}_e\})$ to LP-MC is a feasible fractional solution to IP-MC. Thus, we can see that the value of the objective function (as the objective functions for both the formulations are the same) of an optimal solution $(\bar{x}, \{\bar{z}_e\})$ to LP-MC is at most that of the optimal solution *OPT* to IP-MC. Now, given the optimal solution $(\bar{x}, \{\bar{z}_e\})$ to LP-MC, we know that for any $e \equiv (u, v) \in E$, as $z_{e,u} + z_{e,v} \geqslant 1$, at least one of the following $z_{e,u} \geqslant \frac{1}{2}$ or $z_{e,v} \geqslant \frac{1}{2}$ is true. Let us apply the following transformation δ to $(\bar{x}, \{\bar{z}_e\})$: If $z_{e,i} \geqslant \frac{1}{2}$, for any e and i, then $\delta(z_{e,i}) = 1$ and if $z_{e,i} < \frac{1}{2}$ then put $\delta(z_{e,i}) = 0$. Also, $\delta(x_i) = 2x_i$ if for any $i \in V$ there is $\delta(z_{e,i}) = 1$.

It is easy to see that $\delta(\bar{x}, \{\bar{z}_e\})$ is a feasible solution to the IP-MC problem. Moreover, the linearity of the objective function guarantees that the objective function value (or the total weighted cost) of $\delta(\bar{x}, \{\bar{z}_e\})$ is at most twice the objective function value of $(\bar{x}, \{\bar{z}_e\})$. As the cost of $(\bar{x}, \{\bar{z}_e\})$ is at most *OPT*, the cost of $\delta(\bar{x}, \{\bar{z}_e\})$ is at most two times the cost of *OPT*, i.e., $\delta(\bar{x}, \{\bar{z}_e\}) \leqslant 2 \cdot \mathit{OPT}$. Thus, $\delta(\bar{x}, \{\bar{z}_e\})$ is a 2-approximation of the MC problem. Moreover, as LP-MC can be solved in polynomial time [4], the proof follows. □

Theorem 4.2 shows that a constant ratio approximation is possible for the MC problem. Efficient algorithms for solving linear programs, such as Ye's Interior Point method [54], require polynomial (in number of constraints) number of iterations and work well in practice. Traditional algorithm such as Simplex (or randomized Simplex) could also be used, but Klee and Minty [39] showed that the number of iterations performed by some variants of the simplex can be exponential. Moreover, there is always a possibility, depending on the demand values, of the method producing an unbounded or an infinite number of solutions. To overcome these problems, we take advantage of a deterministic linear-time (in number of edges) approach for FTC proposed by Xu et al. [52], as discussed next.

4.2. "Divide and conquer" algorithms

In their algorithm, Xu et al. divide the input graph instance into multiple subgraphs by first rounding the edge demands and then grouping them based on the rounded edge values. They then apply the Weighted Vertex Cover (WVC) algorithm on each subgraph to obtain the solution to the FTC problem on the input graph. They show that their algorithm produces a 8-approximation when a 2-approximation algorithm [5,26] is used for WVC.

The main difference between the FTC and the MC problem is that in FTC the input graph instance has all edges with a single demand value, whereas in the MC problem, each edge has two demand values. The edge demand value used in the MC solution depends on the vertex chosen to cover the edge in the solution. Moreover, the MC problem is not directly reducible to the FTC problem unless the two demand values for each edge are equal. Below we outline two algorithms for solving the MC

problem; they utilize the round and group strategy of [52]. In order to take advantage of their approach to solve the MC problem, we first need to transform the input graph instance so that all edges have equal demand values.

4.2.1. Largest Demand First (LDF) algorithm

In our first approach, we transform an input instance $G \equiv (V, E, w, d)$ of the MC problem to an instance $G' \equiv (V, E, w, d')$ such that, for each edge in G', both of the demand values are equal to the larger of the two demand values of the corresponding edge in G. The intuition behind such a transformation is that if a vertex is able to cover the larger demand, then it will definitely be able to cover any demand smaller or equal to the larger demand. Then, the final demand values of the edges in the new graph instance G' are *rounded* off to the closest power of 2 of the larger demand value chosen in the previous step. Lemma 4.3 shows that any solution of the MC problem on such a transformed version (G') of the original graph (G) is also a feasible solution of the original graph. After obtaining G', it is first divided into subgraphs (G_k) based on the rounded edge demands (2^k), with each subgraph containing only edges of the same demand value. A known minimum WVC algorithm (such as [5,26]) is then used to obtain the minimum weighted vertex cover for each subgraph G_k. The mix cover is finally obtained by combining solutions from each of the individual subgraphs in the previous step. The LDF algorithm is outlined in Algorithm 1.

Lemma 4.3. *Any solution to the MC problem on the transformed graph instance* $G' \equiv (V, E, w, d')$ *is also a feasible solution to the MC problem on the original graph instance* $G \equiv (V, E, w, d)$. *Moreover, for large demand values (specifically,* $\geqslant 2$*),* $\mathit{OPT}(G') \leqslant 2\alpha \mathit{OPT}(G)$, *where* $\mathit{OPT}(\cdot)$ *is the optimal solution and* $\alpha = \max\{2, \max\{|d_e^u - d_e^v| \mid \forall e \equiv (u, v) \in G\}\}$, *i.e.,* α *is the greater of* 2 *or the maximum edge demand difference over all edges*.

Proof. The first part of the lemma is straightforward. As the vertex and edge sets of both G and G' are the same, a cover for G' (that covers all edges of G') is also a cover for G. Moreover, as the edge demands are rounded off to the largest and then to the closest power of 2, the selected capacity of the vertices for a solution (or mix cover) in G' will always be greater than the demands of the corresponding edges in G. Thus, a mix cover for G' is also a feasible mix cover for G.

Now, let us prove the second part. Let $V_{\mathit{OPT}(G)}$ be the set of vertices of the optimal solution $\mathit{OPT}(G)$ for the MC problem on the graph instance G and let $C_{\mathit{OPT}(G)} = \{c(v_i) \mid v_i \in V_{\mathit{OPT}(G)}\}$ be the capacities assigned to each vertex in the optimal solution. Consider a solution S such that it has the same set of vertices as $V_{\mathit{OPT}(G)}$ and with capacities $C_S = \{2(c(v_i) + \alpha) \mid v_i \in S\}$, where α is as defined above. We can see that (S, C_S) is always a feasible solution to G'. This is because, firstly, we always select the larger demand value for each edge e. Thus, even in the worst case, where all vertices $v_i \in V_{\mathit{OPT}(G)}$ in the optimal solution $\mathit{OPT}(G)$ are such that $d_e^{v_i} < d_e^{v_j}, \forall e \equiv (v_i, v_j) \in E$, capacities selected in C_S will always overcome the

Algorithm 1. Largest Demand First (LDF) algorithm

input : A graph $G \equiv (V, E, w, d)$.
output: A mix cover $S_{MC} \equiv (v, c(v))$ of G, where $v \in V$ and $c(v)$ is the capacity assigned to v.

for *all* $e \equiv (u, v) \in E$ **do**
  $d'_e \equiv (d'^u_e = \max\{d^u_e, d^v_e\}, d'^v_e = \max\{d^u_e, d^v_e\})$;
  if $2^{k-1} \leqslant d'^u_e = d'^v_e \leqslant 2^k$ **then**
    $d'_e \equiv (d'^u_e = 2^k, d'^v_e = 2^k)$;
  end
end
Let $G' \equiv (V, E, w, d')$;
Let $G_k \equiv (V_k, E_k, w)$ be a subgraph of G' induced by edges $E_k = \{e \in E | d'_e = 2^k\}$;
for *all* G_k **do**
  if $V_k \neq \emptyset$ **then**
    S_{G_k} = `WVC-2Approx(`$G_k \equiv (V_k, E_k, w)$`)`;
  else
    $S_{G_k} = \emptyset$;
  end
end
$S_{MC} = \emptyset$;
for *all* S_{G_k} *such that* $S_{G_k} \neq \emptyset$ **do**
  $c(v) \leftarrow \max\{2^k | \forall k \text{ s.t. } v \in S_{G_k}\}$;
  $S_{MC} \leftarrow (v, c(v))$;
end
return S_{MC};

difference in demands. Secondly, the rounded demands of each edge e is only at most twice that of the original (larger) demand. Thus,

$$\begin{aligned} OPT\big(G'\big) &\leqslant \sum_{v_i \in S} 2\big(c(v_i) + \alpha\big) w_{v_i} \\ &\leqslant \sum_{v_i \in S} 2\alpha c(v_i) w_{v_i} \leqslant 2\alpha OPT(G). \end{aligned}$$

The second inequality above follows from the fact that $c(v_i)\alpha \geqslant c(v_i) + \alpha$ for all $c(v_i), \alpha \geqslant 2$. □

We have the following result for the solution quality and running time of LDF.

Theorem 4.4. *The LDF algorithm is a* $\mathrm{O}(m + n)$ *time* (*linear in terms of the number of edges and vertices*), $4\alpha\beta$*-approximation algorithm for the MC problem on graph instances with large demands* (*specifically,* $\geqslant 2$), *where* $\beta > 1$ *is the approximation ratio of the minimum WVC algorithm used,* α *is as defined in Lemma* 4.3, m *is the total number of edges and* n *is the total number of vertices in the graph.*

Proof. Let WVC-2Approx(G_k) denote the output (overall minimum weight) of applying a β-approximation minimum WVC algorithm to the subgraph G_k of G'. If $OPT(G_k)$ is the corresponding optimal solution, then we have the following inequality:

$$\begin{aligned} &\text{WVC-2Approx}(G_k) \leqslant \beta OPT(G_k), \\ &OPT(G_k) \geqslant \frac{1}{\beta}\text{WVC-2Approx}(G_k). \end{aligned} \tag{1}$$

From Lemma 2 of [52] we know that,

$$OPT\big(G'\big) \geqslant \frac{1}{2}\sum_{k=0}^{K} 2^k OPT(G_k), \tag{2}$$

where K is max. value of the exponent after the rounding. From Lemma 4.3,

$$OPT\big(G'\big) \leqslant 2\alpha OPT(G). \tag{3}$$

From (1) and (2), we have

$$\begin{aligned} OPT\big(G'\big) &\geqslant \frac{1}{2}\sum_{k=0}^{K} \frac{2^k}{\beta}\text{WVC-2Approx}(G_k) \\ &\geqslant \frac{1}{2\beta}\sum_{k=0}^{K} 2^k \text{WVC-2Approx}(G_k). \end{aligned}$$

Combining the above inequality with Eq. (3) we have,

$$\begin{aligned} &2\alpha OPT(G) \geqslant \frac{1}{2\beta}\sum_{k=0}^{K} 2^k \text{WVC-2Approx}(G_k), \\ &\sum_{k=0}^{K} 2^k \text{WVC-2Approx}(G_k) \leqslant 4\alpha\beta OPT(G). \end{aligned} \tag{4}$$

The left-hand side of the inequality in (4) clearly denotes an upper bound of the objective function on the transformed graph instance G' (thus, an upper bound of the objective function on the original graph G) computed from the output of the LDF algorithm. From this inequality, it is clear that the LDF algorithm is a $4\alpha\beta$-approximation of the MC problem.

Now, let us observe the time complexity of the LDF algorithm. Tasks such as determining the larger demand per edge, rounding the demand value (lines 1–5) and

assigning capacities (lines 17–20) can be completed in O($m + n$) time, in the worst-case. Moreover, Bar-Yehuda and Even [5] showed that a 2-approximation can be obtained for a WVC problem in time linear in terms of the number of edges and vertices. Thus, lines 9–15 run in O($m + n$) time in the worst case. Thus, the LDF algorithm runs in linear time in terms of the number of edges and vertices (or quadratic time in terms of the number of vertices only) in the worst case. □

4.2.2. Smallest Demand First (SDF) algorithm

In the LDF algorithm, we transform the input graph instance into an instance where the smaller edge demand is replaced by the larger one. This guarantees that each edge has the same (and a single) demand value and that the mix cover of such a transformed instance is also a feasible mix cover of the original instance. In practice, it is clear that such a strategy will produce a highly sub-optimal solution because there may be vertices in the final solution that may cover edges with much lower actual demand values. In order to overcome this issue, we propose another strategy for solving the MC problem, called the Smallest Demand First (SDF) algorithm, which is based on a transformation that chooses the smaller of the two edge demand values in the input instance G.

The SDF algorithm, as outlined in Algorithm 2, consists of three phases. In the first phase, in contrast to the LDF algorithm, we transform the input graph instance $G \equiv (V, E, w, d)$ of the MC problem into an instance $G'' \equiv (V, E, w, d'')$ where the larger edge demand is now replaced by the smaller one. In this phase, an additional task during edge demands transformation is to remember the largest demand ($d^v_{\max}$) to be covered at each vertex. In the second phase, similar to the LDF algorithm, we use the round and group strategy to obtain a mix cover for the transformed instance. In the final phase, we assign capacities to the vertices based on the output of the previous phase and the largest demand $d^v_{\max}$ determined in the first phase. Lemma 4.5 gives the relationship between the MC problem on the transformed version G'' and the original graph G.

Lemma 4.5. *OPT*(G'') $\leqslant$ 2*OPT*(G), *where* $G'' \equiv (V, E, w, d'')$ *is the shortest demand first transformation of the original graph instance* $G \equiv (V, E, w, d)$ *and* *OPT*($\cdot$) *is the optimal solution of the MC problem on the input graph instance.*

Proof. Let $V_{OPT(G)}$ be the set of vertices of the optimal solution *OPT*(G) for the MC problem on the graph instance G and let $C_{OPT(G)} = \{c(v_i) \mid v_i \in V_{OPT(G)}\}$ be the capacities assigned to each vertex in the optimal solution. Now, consider a solution S such that it has the same set of vertices as $V_{OPT(G)}$ and with capacities $C_S = \{2c(v_i) \mid v_i \in S\}$. We can see that (S, C_S) is always a feasible solution to G''. As in G'', we always select the smaller demand value for each edge e and round it off to the closest power of 2, the demand value of an edge in G'' is at most twice the smaller edge demand of the corresponding edge in the original graph instance G. As the capacity of the vertex v_i in the optimal solution *OPT*(G) should be such that

Algorithm 2. Smallest Demand First (SDF) algorithm

input : Graph $G \equiv (V, E, w, d)$.
output: Mix cover $S_{MC} \equiv (v, c(v))$ of G, where $v \in V$ and $c(v)$ is the capacity assigned to v.

for *all* $v \in V$ **do**
  $d^v_{\max} = 0$;
end
for *all* $e \equiv (u, v) \in E$ **do**
  if $d^u_e > d^u_{\max}$ **then**
    $d^u_{\max} = d^u_e$;
  end
  if $d^v_e > d^v_{\max}$ **then**
    $d^v_{\max} = d^v_e$;
  end
  $d''_e \equiv (d''^u_e = \min\{d^u_e, d^v_e\}, d''^v_e = \min\{d^u_e, d^v_e\})$;
  if $2^{k-1} \leqslant d''^u_e = d''^v_e \leqslant 2^k$ **then**
    $d''_e \equiv (d''^u_e = 2^k, d''^v_e = 2^k)$;
  end
end
Let $G'' \equiv (V, E, w, d'')$;
Let $G_k \equiv (V_k, E_k, w)$ be a subgraph of G'' induced by edges $E_k = \{e \in E | d''_e = 2^k\}$;
for *all* G_k **do**
  if $V_k \neq \emptyset$ **then**
    S_{G_k} = `WVC-2Approx` $(G_k \equiv (V_k, E_k, w))$;
  else
    $S_{G_k} = \emptyset$;
  end
end
$S_{MC} = \emptyset$;
for *all* S_{G_k} *such that* $S_{G_k} \neq \emptyset$ **do**
  $c(v) \leftarrow \max\{\max\{2^k | \forall k \text{ s.t. } v \in S_{G_k}\}, d^v_{\max}\}$;
  $S_{MC} \leftarrow (v, c(v))$;
end
return S_{MC};

it should be greater than or equal to the largest demand on the edges it covers, the capacity of v_i selected in C_S will be able to cover the corresponding edges of v_i in G''. Thus,

$$\begin{aligned} OPT\big(G''\big) &\leqslant \sum_{v_i \in S} 2\big(c(v_i)\big) w_{v_i} \\ &\leqslant 2 \sum_{v_i \in S} c(v_i) w_{v_i} \leqslant 2OPT(G). \end{aligned}$$

□

However, it is easy to see that a feasible solution for the MC problem on G'' may not necessarily be a feasible solution to the MC problem on the original graph

instance G. Moreover, in the worst case, $OPT(G'')$ may include only those vertices that correspond to larger demand values in the original graph. We have the following result for the approximation ratio of SDF.

Theorem 4.6. *SDF is a* O$(m+n)$ (*linear in terms of the number of edges and vertices*), $4\alpha\beta$*-approximation algorithm for the MC problem on graph instances with large demands* (*specifically*, $\geqslant 2$), *where* $\beta > 1$ *is the approximation ratio of the minimum WVC algorithm used*, α *is as defined in Lemma* 4.3, m *is the total number of edges and* n *is the total number of vertices in the graph*.

Proof. We use a similar argument that we use to prove Theorem 4.4. From Eq. (1) in the proof of Theorem 4.4 we know that:

$$OPT(G_k) \geqslant \frac{1}{\beta}\text{WVC-2Approx}(G_k),$$

where, β is the approximation ration of the weighted vertex cover algorithm WVC-2Approx and G_k is the induced subgraph of G'' with edge demands 2^k. Now, let K be the maximum value of the exponent after the rounding the shortest demands on each edge. From Lemma 2 of [52] we know that, $OPT(G'') \geqslant \frac{1}{2}\sum_{k=0}^{K} 2^k OPT(G_k)$. From Lemma 4.5 we have, $OPT(G'') \leqslant 2OPT(G)$.

From the above we have,

$$\begin{aligned}
2OPT(G) &\geqslant \frac{1}{2}\sum_{k=0}^{K} 2^k OPT(G_k) \\
\implies \quad 4OPT(G) &\geqslant \sum_{k=0}^{K} 2^k OPT(G_k) \\
\implies \quad 4\beta OPT(G) &\geqslant \sum_{k=0}^{K} 2^k \text{WVC-2Approx}(G_k) \\
\implies \quad 4\alpha\beta OPT(G) &\geqslant \alpha\sum_{k=0}^{K} 2^k \text{WVC-2Approx}(G_k) \\
\implies \quad 4\alpha\beta OPT(G) &\geqslant \sum_{k=0}^{K} \big(2^k + \alpha\big)\text{WVC-2Approx}(G_k). \qquad (5)
\end{aligned}$$

The right-hand side of Eq. (5) clearly denotes an upper bound on the objective function computed by the SDF algorithm. From this inequality, it is clear that the SDF algorithm is a $4\alpha\beta$-approximation algorithm for the MC problem.

Now, let us observe the time complexity of the SDF algorithm. The SDF algorithm has an additional step, as compared to the LDF algorithm, which is the largest demand determination on each vertex (lines 5–10). In the worst case, this only increases the complexity by a constant factor. Thus the SDF algorithm, similar to the LDF algorithm, runs in linear time in terms of the number of edges and vertices (or quadratic time in terms of the number of vertices only) in the worst case. □

Next, we present two heuristics for the MC problem, which run more efficiently as compared to the LDF and SDF algorithms, but do not provide a guarantee on the solution quality.

5. Other heuristics

We propose two other heuristics for solving the MC problem. The first heuristic employs a greedy strategy to select vertices that go in the solution, while the second heuristic makes use of a simulated annealing-based optimization strategy.

5.1. Greedy heuristic

We propose *two* greedy heuristics based on the parameter of the input graph instance used for making the greedy choice. The first heuristic, shown in Algorithm 3, begins with a vertex $v \in V$ such that v has the smallest incident demand value amongst all vertices in G. In each iteration, the heuristic then greedily selects a vertex (to be added to the solution) with the next smallest incident demand value. This continues until all edges in G are covered.

The second heuristic, shown in Algorithm 4, begins with a vertex $v \in V$ such that v has the smallest mixing cost (w_v) amongst all vertices in G. The capacity of the vertex v is chosen as the largest demand value incident on it. In each iteration, the heuristic then greedily selects a vertex (to be added to the solution) with the next smallest mixing cost. This continues until all edges in G are covered. It is easy to see that the running time of both the greedy heuristics are dictated by lines 4–11 and lines 16–21, which in the worst case enumerates all the edges in the graph. Thus the asymptotic running time of the greedy heuristics is linear in terms of the total number of edges and vertices in the input graph, i.e., polynomial (specifically, quadratic) in terms of the total number of vertices. But, as each greedy heuristic tries to locally maximize either the demand value or the mixing cost, it is difficult to derive a bound on the quality of the solution produced by these heuristics.

5.2. Local Solution Search heuristic

Due to the combinatorial hardness of the MC problem, an exhaustive search for a globally optimal mix cover is infeasible. The greedy heuristics discussed above make locally optimal choices (based on edge demand or mixing cost) at each step,

Algorithm 3. Demand-based greedy heuristic

input : Graph $G \equiv (V, E, w, d)$.
output: Mix cover $S_{MC} \equiv (v, c(v))$ of G, where $v \in V$ and $c(v)$ is the capacity assigned to v.

```
for all v ∈ V do
    d^v_max = 0;
end
for all e ≡ (u, v) ∈ E do
    if d^u_e > d^u_max then
        d^u_max = d^u_e;
    end
    if d^v_e > d^v_max then
        d^v_max = d^v_e;
    end
end
Sort all vertices v ∈ V based on increasing values of d^v_max, i.e., vertex with smallest value of d^v_max
first. Let V' be the sorted set of vertices;
Let S_MC = ∅;
Add (V'[1], c(V'[1]) = d^{V'[1]}_max) in S_MC;
Mark all (V'[1], v) ∈ E as covered, where v ∈ V;
for i = 2 to n do
    if there are more edges in E to be covered then
        Add (V'[i], c(V'[i]) = d^{V'[i]}_max) in S_MC;
        Mark all (V'[i], v) ∈ E as covered, where v ∈ V;
    end
end
return S_MC;
```

and hence can get stuck in a local optima. This results in a globally poor approximation. In order to obtain a better approximation of the global optimum, we propose a probabilistic neighborhood search heuristic based on Simulated Annealing [30].

In this heuristic, we begin from an initial solution (obtained by one of the greedy algorithms described above). Then in each iteration, the current solution is probabilistically replaced by a randomly chosen "nearby" or "neighborhood" solution. The probability of replacing the current solution with a neighborhood solution depends both on the difference between the values of those solutions and a global parameter t, called the temperature, which is gradually decreased in each iteration. The temperature parameter enables the algorithm to escape local optima throughout the simulation, while slowly converging to a near-optimal solution. Next, we describe the neighborhood of a feasible solution by using a *local transformation* technique.

Informally, a local transformation transforms a feasible solution α to another feasible solution β by making some local changes to the specification of α. To define a neighborhood for an instance of the MC problem, we introduce a transformation called the k*-neighborhood* transformation. Give an input graph instance $G \equiv (V, E, w, d)$ of the MC problem and a feasible solution S_{MC}, a k-neighborhood of S_{MC} exists if and only if there is a vertex $v \in S_{MC}$ such that there are exactly

Algorithm 4. Mixing cost-based greedy heuristic

input : Graph $G \equiv (V, E, w, d)$.
output: Mix cover $S_{MC} \equiv (v, c(v))$ of G, where $v \in V$ and $c(v)$ is the capacity assigned to v.

```
1  for all v ∈ V do
2  |  d^v_max = 0;
3  end
4  for all e ≡ (u, v) ∈ E do
5  |  if d^u_e > d^u_max then
6  |  |  d^u_max = d^u_e;
7  |  end
8  |  if d^v_e > d^v_max then
9  |  |  d^v_max = d^v_e;
10 |  end
11 end
12 Sort all vertices v ∈ V based on increasing values of w_v, i.e., vertex with smallest value of w_v
   first. Let V' be the sorted set of vertices;
13 Let S_MC = ∅;
14 Add (V'[1], c(V'[1]) = d^{V'[1]}_max) in S_MC;
15 Mark all (V'[1], v) ∈ E as covered, where v ∈ V;
16 for i = 2 to n do
17 |  if there are more edges in E to be covered then
18 |  |  Add (V'[i], c(V'[i]) = d^{V'[i]}_max) in S_MC;
19 |  |  Mark all (V'[i], v) ∈ E as covered, where v ∈ V;
20 |  end
21 end
22 return S_MC;
```

k edges $(v, u) \in E$ where $u \notin S_{MC}$. If there are no such vertices v in S_{MC}, then S_{MC} does not have a k-neighborhood. Then, a k-neighbor S^k_{MC} of S_{MC} is obtained by replacing one such vertex v in S_{MC} with the k vertices u such that $(v, u) \in E$ and $u \notin S_{MC}$. The first observation we make is that a k-neighbor of any feasible solution has $k - 1$ extra vertices. It is also easy to see that a k-neighbor (if one exists) of a feasible solution is also a feasible solution to the MC-problem. The Local Solution Search heuristic is outlined in Algorithm 5.

The asymptotic time complexity of the Local Solution Search heuristic is dictated by the complexities of the *Greedy* algorithm (line 1) and the Neighbor procedure (line 5). As seen earlier, the Greedy algorithm runs in linear time in terms of the total number of vertices and edges (quadratic in the number of vertices) in the worst case. Provided $c_{\max}$ and t are of the same order as the number of vertices, the worst-case asymptotic runtime of executing lines 4–13 is quadratic in the total number of vertices. Thus, the overall asymptotic time complexity of the Local Solution Search heuristic is still quadratic in the number of vertices. In practice, the Local Solution Search is expected to run slower than the Greedy heuristic because of the extra simulated annealing step. We now evaluate the practical efficiency of the proposed approaches by executing them on real vehicular road-network data.

Algorithm 5. Local Solution Search using simulated annealing

input : Graph $G \equiv (V, E, w, d)$, maximum evaluation count $c_{\max}$, initial temperature $t_{initial}$ and temperature decrease factor $j \in [0, 1]$
output: Mix cover $S_{MC} \equiv (v, c(v))$ of G, where $v \in V$ and $c(v)$ is the capacity assigned to v.

$S_{MC} \leftarrow$ `Greedy` (G), $C = \sum_{v \in S_{MC}} c(v)w_v$; `/*Initial solution and total weighted capacity*/`
$S_{MC}^{best} \leftarrow S_{MC}$, $C_{best} \leftarrow C$; `/*Best solution and corresponding capacity*/`
$c \leftarrow 0$; `/*Initial evaluation count*/`
while $c < c_{\max}$ **do**
  $S_{MC}^{new} \leftarrow$ `Neighbor` (S_{MC}, k), $C_{new} = \sum_{v \in S_{MC}^{new}} c(v)w_v$;
  $t \leftarrow$ `Temperature` $(t_{initial}, c, j)$, $c \leftarrow c + 1$;
  if `Random ()` $<$ `AcceptProbability` (C, C_{new}, t) **then**
    `/*Randomly move to a new feasible solution */`
    $S_{MC} \leftarrow S_{MC}^{new}$, $C \leftarrow C_{new}$;
  end
  if $C_{new} > C_{best}$ **then**
    `/*Solution is better than current best */`
    $S_{MC}^{best} \leftarrow S_{MC}^{new}$, $C_{best} \leftarrow C_{new}$
  end
end
return S_{MC}^{best};

Procedure Neighbor(S_{MC}, k)

`/*Determine the neighborhood of the current solution */`
input : Feasible solution S_{MC} and maximum value of neighborhood parameter k
output: A l-neighbor S_{MC}^{l} of S_{MC} where $l \leqslant k$

$S_{MC}^{l} = S_{MC}$;
Let v be a randomly selected vertex in S_{MC}^{l};
for $l = 1$ *to* k **do**
  if *l-neighbor of v exists* **then**
    Replace v (and its capacity $c(v)$) by its l-neighbor in S_{MC}^{l};
    return S_{MC}^{l};
  end
end
return S_{MC}^{l};

6. Empirical evaluation

We evaluate the performance of the proposed algorithms by implementing them in Matlab on a multi-core desktop computer. For our experiments, we construct the input graph instances using real road-traffic data (intersections, road segments and bi-directional AADT traffic intensities) from the official transportation databases for the US states of Florida [16], Virginia [51] and Arizona [3].

Procedure Temperature($t_{initial}$, c, j)

```
/*Compute the simulation temperature                    */
```
input : Initial temperature $t_{initial}$, evaluation count c and temperature decrease factor $j \in [0, 1]$
output: New simulation temperature value
1 return $t_{initial} \cdot (j)^c$;

Procedure AcceptProbability(C, C_{new}, t)

```
/*Compute the acceptance probability of the new solution  */
```
input : Solution quality or total weighted capacities C and C_{new} of feasible solutions and current simulation temperature value t
output: Probability of accepting solution S_{MC}^{new}
1 return $min(1, e^{(C_{new}-C)/t})$;

For each state, we consider three different sizes of the respective road network graphs: a small graph that corresponds to 25% of the total number of municipalities, a medium (65–85%) and a full state graph, except for Arizona, where we were able to obtain only limited traffic data that is comparable in size with the small graphs of Virginia and Florida. For each such road network graph, we evaluate the performance of the proposed algorithms for three vertex weight distributions: (i) Constant, (ii) uniform and (iii) positive Gaussian. The constant distribution assigns the same weight (=1) to all vertices, the uniform draws the weights uniformly at random from the interval [1, 100], whereas the Gaussian has an expected value of 50 and a standard deviation of 10. The edge demand values are derived from the traffic intensity data (in each direction) on the roads. For the simulated annealing heuristic, we choose the parameter values such that the probability of exiting a local optima, if any, is high. To this end, we select the parameters $c_{\max} = 1000$, $t_{initial} = 100$, the temperature decrease factor $j = 0.9$ and the maximum value of the neighborhood parameter $k = 400$.

We measure the performance of each algorithm by using three metrics. First, we record the number of vertices in the MC solution produced by the algorithms, as shown in Table 1. The number of vertices in the solution is an important performance metric in this work, because it is indicative of the number of mix-zones to be deployed. Second, we compute the ratio of the objective function value of the solution produced by the algorithm to the worst-case (naïve) solution (which includes all vertices of the graph), as shown in Table 2. Third, we measure the duration of the simulation (or execution time) in seconds, as shown in Table 3. All values are averaged over 100 runs of the algorithms. For clarity, we plot the values for the full-size graphs of the states of Florida and Virginia in Figs 2 and 3, respectively.

From Table 1 and Figs 2(a), 3(a), we can see that the LDF and SDF algorithms perform better than the greedy and the simulated annealing heuristics in terms of the number of vertices in the final solution. There is no major performance difference

Table 1

Performance of the algorithms in terms of the number of vertices in the solution

	SMALL SIZE GRAPH 25% of municipalities			MEDIUM SIZE GRAPH 65-85% of municipalities			FULL SIZE GRAPH Entire State		
Total number of *vertices* / *edges* Florida	2557 / 2640			6326 / 6960			7557 / 8310		
Virginia	2408 / 2514			5726 / 6728			5881 / 6952		
Arizona	1571 / 1408								
	Constant	**Uniform**	**Gaussian**	**Constant**	**Uniform**	**Gaussian**	**Constant**	**Uniform**	**Gaussian**
LDF Florida	1243	1267	1237	2891	2990	2909	3481	3604	3503
SDF Florida	1217	1243	1214	2863	2950	2876	3452	3547	3452
Demand-based greedy Florida	2494	2494	2494	6082	6082	6082	7341	7341	7341
Mix-cost based greedy Florida	2465	2452	2459	6079	6033	6027	7336	7289	7282
Simulated annealing Florida	2465	2452	2459	6079	6033	6027	7336	7289	7282
LDF Virginia	1346	1373	1351	3328	3402	3344	3523	3589	3531
SDF Virginia	1376	1387	1364	3376	3423	3375	3550	3605	3550
Demand-based greedy Virginia	2090	2090	2090	5020	5020	5020	5264	5264	5264
Mix-cost based greedy Virginia	2089	2088	2088	5020	5019	5018	5264	5263	5262
Simulated annealing Virginia	2089	2088	2088	5020	5019	5018	5264	5263	5262
LDF Arizona	1180	1148	1144						
SDF Arizona	1184	1151	1147						
Demand-based greedy Arizona	1420	1420	1420						
Mix-cost based greedy Arizona	1420	1420	1419						
Simulated annealing Arizona	1420	1420	1419						

Table 2

Performance of the algorithms in terms of the solution quality

	SMALL SIZE GRAPH 25% of municipalities			MEDIUM SIZE GRAPH 65-85% of municipalities			FULL SIZE GRAPH Entire State		
Total number of *vertices* / *edges* Florida	2557 / 2640			6326 / 6960			7557 / 8310		
Virginia	2408 / 2514			5726 / 6728			5881 / 6952		
Arizona	1571 / 1408								
	Constant	**Uniform**	**Gaussian**	**Constant**	**Uniform**	**Gaussian**	**Constant**	**Uniform**	**Gaussian**
LDF Florida	0.510	0.394	0.475	0.488	0.380	0.456	0.489	0.384	0.460
SDF Florida	0.445	0.346	0.416	0.429	0.338	0.403	0.426	0.336	0.399
Demand-based greedy Florida	0.695	0.695	0.695	0.692	0.693	0.692	0.695	0.694	0.695
Mix-cost based greedy Florida	0.694	0.677	0.686	0.697	0.686	0.687	0.696	0.687	0.688
Simulated annealing Florida	0.694	0.677	0.686	0.697	0.686	0.687	0.696	0.687	0.688
LDF Virginia	0.822	0.785	0.813	0.820	0.785	0.812	0.823	0.788	0.814
SDF Virginia	0.791	0.762	0.784	0.793	0.766	0.787	0.798	0.770	0.792
Demand-based greedy Virginia	0.699	0.699	0.699	0.687	0.686	0.686	0.689	0.689	0.689
Mix-cost based greedy Virginia	0.699	0.699	0.699	0.687	0.686	0.686	0.689	0.689	0.689
Simulated annealing Virginia	0.699	0.699	0.699	0.687	0.686	0.686	0.689	0.689	0.689
LDF Arizona	0.913	0.861	0.882						
SDF Arizona	0.908	0.857	0.876						
Demand-based greedy Arizona	0.701	0.702	0.701						
Mix-cost based greedy Arizona	0.701	0.702	0.701						
Simulated annealing Arizona	0.701	0.702	0.701						

Note: These values are the ratio of the solution quality of each algorithm to the solution quality of the naïve solution that uses all vertices for the mix cover.

between LDF and SDF, and between the greedy and simulated annealing strategies. From the evaluation results we can observe that depending on the size of the input graph and the demand values associated with the edges, the number of mix-zones to be deployed is between 46% (for Florida) and 58% (for Virginia) of the total number of vertices. In Florida, SDF performs slightly better than LDF, i.e., SDF produces around 2–3% or approximately 102 fewer mix-zones. This observation is consistent

Table 3
Performance of the algorithms in terms of the execution time (in seconds)

		SMALL SIZE GRAPH 25% of municipalities			**MEDIUM SIZE GRAPH** 65-85% of municipalities			**FULL SIZE GRAPH** Entire State		
Total number of *vertices* / *edges*	Florida	2557 / 2640			6326 / 6960			7557 / 8310		
	Virginia	2408 / 2514			5726 / 6728			5881 / 6952		
	Arizona	1571 / 1408								
		Constant	**Uniform**	**Gaussian**	**Constant**	**Uniform**	**Gaussian**	**Constant**	**Uniform**	**Gaussian**
LDF	Florida	7.70	8.70	8.50	37.02	37.00	36.72	49.04	42.81	49.15
SDF	Florida	8.55	9.73	9.48	39.48	38.57	39.18	54.66	45.50	53.79
Demand-based greedy	Florida	0.29	0.32	0.31	1.90	1.87	1.82	2.56	2.36	2.63
Mix-cost based greedy	Florida	0.25	0.28	0.27	1.64	1.65	1.62	2.20	2.06	2.30
Simulated annealing	Florida	0.87	1.02	0.97	3.22	3.18	3.13	3.96	3.67	4.14
LDF	Virginia	7.20	7.96	7.81	29.95	30.07	30.51	31.64	28.52	32.20
SDF	Virginia	7.53	8.35	8.22	31.85	31.51	31.70	33.37	29.80	33.62
Demand-based greedy	Virginia	0.26	0.29	0.28	1.65	1.70	1.65	1.72	1.61	1.77
Mix-cost based greedy	Virginia	0.27	0.28	0.27	1.62	1.67	1.64	1.66	1.58	1.74
Simulated annealing	Virginia	0.89	0.95	0.94	3.16	3.23	3.18	3.29	3.00	3.35
LDF	Arizona	4.57	4.71	4.69						
SDF	Arizona	4.59	4.73	4.69						
Demand-based greedy	Arizona	0.11	0.11	0.11						
Mix-cost based greedy	Arizona	0.09	0.10	0.10						
Simulated annealing	Arizona	0.58	0.60	0.58						

across all graph sizes. In Virginia, on the contrary, LDF performs slightly better than SDF (up to 30 fewer mix-zones). This indicates that, although relatively small, the performance of the two algorithms are influenced by the road network topology, and further investigations are needed to determine the effects of the road topology on the performance of the proposed algorithms. The greedy and simulated annealing heuristics do not perform very well in this regard. According to the solutions produced by these algorithms, the number of mix-zones required can be as high as 90% of the total number of vertices (or possible mix-zone deployments), as it is the case for the Florida data.

In terms of the solution quality, we can see from Table 2 and Figs 2(b), 3(b) that SDF performs slightly better than LDF for all graph sizes and for all vertex weight distributions. This is in line with the analytical results. There is another interesting trend that we can see from these plots. For the states of Virginia and Arizona, the greedy and simulated annealing heuristics *slightly* outperform the LDF and SDF algorithms in terms of the solution quality, while the opposite is true for the state of Florida. Both greedy and simulated annealing heuristics perform equally well for all states and for all weight distributions. This gives an impression that the performance of the greedy and simulated annealing heuristics depends on the road network topology and the edge distribution which is, obviously, different in Florida and Virginia. Thus, in terms of solution quality, there is no clear winner amongst these algorithms.

The experimental results confirm that, as compared to the naïve solution, the proposed algorithms achieve a lower mix-zone deployment cost; sometimes as low as 34% of the cost of the naïve solution. For the LDF and SDF algorithms, and for all combinations of parameters, the uniform distribution achieves the best (lowest) objective function ratio, followed by the Gaussian and the constant distributions. In the

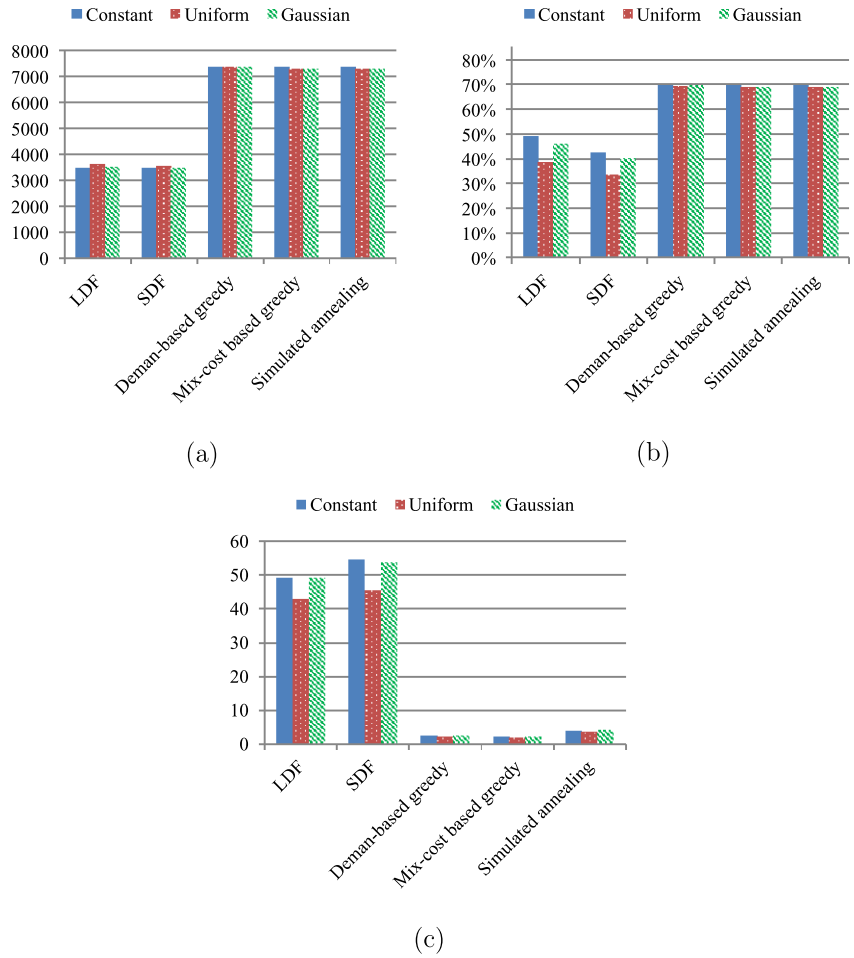

Fig. 2. Performance of the algorithms on traffic data for Florida. (a) Number of vertices in the solution. (b) Solution quality (ratio between the objective function values). (c) Execution time (in seconds). (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-130465.)

uniform distribution, which assigns (on average) the same weight to an equal number of vertices, it is possible to determine lower weight vertices to cover the same set of edges. In the Gaussian scenario, as most of the weights will be close to the mean, this will be more difficult. In the constant weight scenario, where all vertices have the same weights, there is no chance of finding alternative vertices (with lower weights) to cover the same set of edges and thus the assigned capacity would depend solely on the edge demands.

Another important metric for evaluating the performance of the proposed algorithms is the running time or execution efficiency. As the traffic patterns evolve dur-

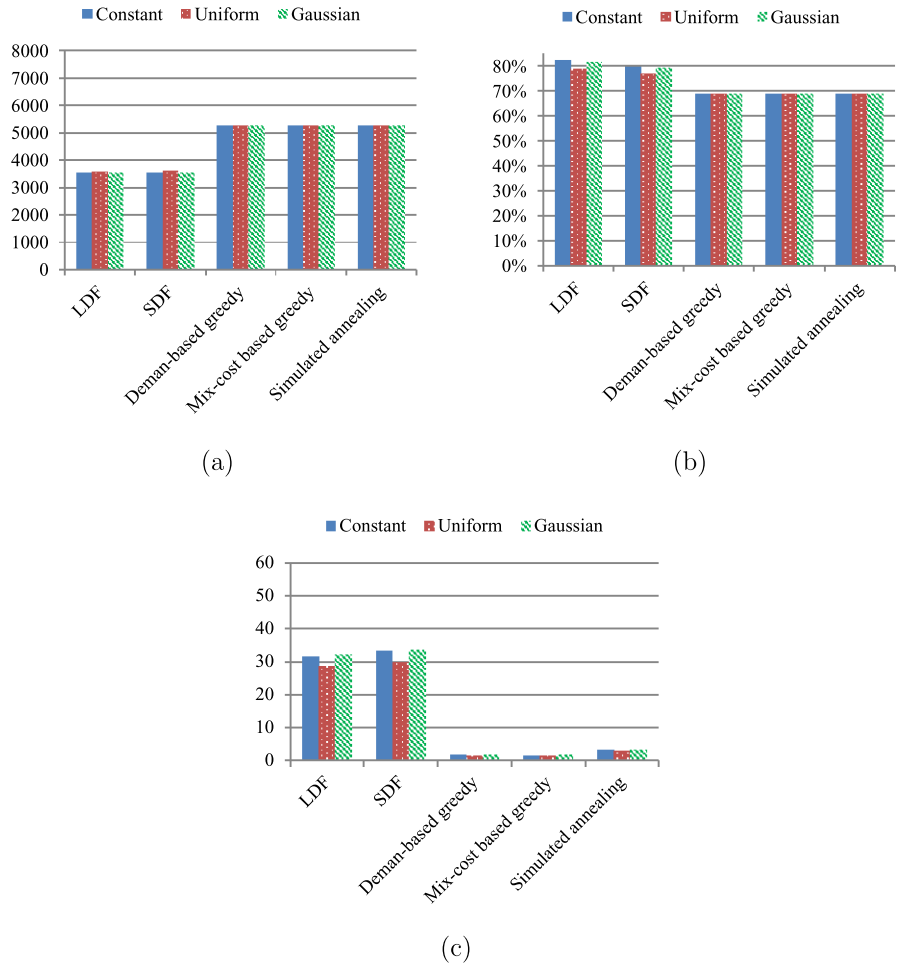

Fig. 3. Performance of the algorithms on traffic data for Virginia. (a) Number of vertices in the solution. (b) Solution quality (ratio between the objective function values). (c) Execution time (in seconds). (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-130465.)

ing the day in each region, these algorithms would be executed multiple times per day, in order to adapt the solutions to the dynamic traffic intensities throughout the day. Moreover, these algorithms may be executed on small and resource constrained devices such as mobile phones or vehicle on-board units. We can see from Table 3 and Figs 2(c), 3(c) that the greedy and simulated annealing heuristics execute *considerably* faster (around 40 time faster) than the LDF and SDF algorithms, albeit with a lower solution quality. Despite similar asymptotic time complexities, the *divide and conquer* strategy of the LDF and SDF algorithms adds significant extra computation

as compared to the simple enumeration strategy followed by the heuristics. Even for the largest graph, the execution duration of the LDF and SDF algorithms does not exceed more than 55 s (<1 min) while the running time of the greedy and simulated annealing heuristics is always under 4 s. In summary, these results show that the proposed algorithms execute in a reasonable amount of time by using only standard computing infrastructure.

7. Conclusion

An optimal placement of mix-zones is critical for improving the mixing effectiveness and location privacy of mobile users in pervasive networks. In this paper, we addressed the problem of optimal mix-zone placement in pervasive road networking scenarios by formulating it as a graph-based optimization problem. We refer to this problem as the Mix Cover or MC problem. The Mix Cover problem that we studied in this paper is a generalization of the Weighted Vertex Cover and the Facility Terminal Cover (FTC) problem, and has not been addressed in the literature before. We proposed several deterministic bounded-ratio approximation algorithms to solve the MC problem. The first algorithm is based on a LP relaxation of the problem and the remaining two approaches take advantage of a "divide and conquer" strategy proposed by Xu et al. [52]. We proved the solution quality and running-time guarantees for the proposed approaches on input graph instances with demand values greater than or equal to 2. We also proposed two additional algorithms; one based on a greedy heuristic and the other based a simulated annealing approach. The later two algorithms are simple and efficient, but do not provide any guarantees on the quality of the produced solution. We performed extensive experiments to evaluate the execution efficiency and solution quality of all the proposed algorithms on real road network and traffic data. Results from these experiments confirmed the analytical results and highlighted the trade-off between solution quality and execution cost of the proposed algorithms. These results showed that, while the LDF and SDF algorithms are able to provide good quality solutions in a reasonable time, heuristics such as greedy and simulated annealing also perform reasonably well while running in an order of magnitude less time.

References

[1] A. Ahtiainen, K. Kalliojarvi, M. Kasslin, K. Leppanen, A. Richter, P. Ruuska and C. Wijting, Awareness networking in wireless environments: Means of exchanging information, *IEEE Vehicular Technology Magazine* **4**(3) (2009), 48–54.

[2] T. Alpcan and S. Buchegger, Security games for vehicular networks, *IEEE Transactions on Mobile Computing* **10**(2) (2011), 280–290.

[3] Arizona State traffic data, available at: http://www.azdot.gov/mpd/data/aadt.asp.

[4] B. Aspvall and R. Stone, Khachiyan's linear programming algorithm, *Journal of Algorithms* **1**(1) (1980), 1–13.

[5] R. Bar-Yehuda and S. Even, A linear time approximation algorithm for the weighted vertex cover algorithm, *Journal of Algorithms* **2**(2) (1981), 198–203.
[6] A.R. Beresford and F. Stajano, Location privacy in pervasive computing, *Pervasive Computing, IEEE* **2**(1) (2003), 46–55.
[7] A.R. Beresford and F. Stajano, Mix zones: User privacy in location-aware services, in: *PerCom Workshop*, 2004.
[8] I. Bilogrevic, M. Jadliwala, I. Lam, I. Aad, P. Ginzboorg, V. Niemi, L. Bindschaedler and J.-P. Hubaux, Big brother knows your friends: on privacy of social communities in pervasive networks, in: *Proceedings of the 10th International Conference on Pervasive Computing (PERVASIVE)*, 2012.
[9] L. Bindschaedler, M. Jadliwala, I. Bilogrevic, I. Aad, P. Ginzboorg, V. Niemi and J.-P. Hubaux, Track me if you can: on the effectiveness of context-based identifier changes in deployed mobile networks, in: *Proceedings of the 19th Annual Network & Distributed System Security Symposium (NDSS)*, 2012.
[10] S. Buchegger, D. Schiöberg, L.-H. Vu and A. Datta, Peerson: P2P social networking: early experiences and insights, in: *EuroSys SNS Workshop*, 2009, pp. 46–52.
[11] L. Buttyán, T. Holczer and I. Vajda, On the effectiveness of changing pseudonyms to provide location privacy in VANETs, in: *ESAS*, 2007.
[12] L. Buttyán, T. Holczer, A. Weimerskirch and W. Whyte, Slow: A practical pseudonym changing scheme for location privacy in VANETs, in: *IEEE VNC*, 2009.
[13] D. Chaum, Untraceable electronic mail, return addresses and digital pseudonyms, *Comm. ACM* **24**(2) (1981), 84–88.
[14] D. Chaum, The dining cryptographers problem: Unconditional sender and recipient untraceability, *J. Cryptology* **1**(1) (1988), 66–75.
[15] I. Dinur and S. Safra, On the hardness of approximating minimum vertex-cover, *Annals of Mathematics* **162**(1) (2005), 439–485.
[16] Florida State traffic data, available at: http://www.dot.state.fl.us/planning/statistics/gis/trafficdata.shtm.
[17] J. Freudiger, M. Raya, M. Felegyhazi, P. Papadimitratos and J.-P. Hubaux, Mix zones for location privacy in vehicular networks, in: *Win-ITS*, 2007.
[18] J. Freudiger, R. Shokri and J.-P. Hubaux, On the optimal placement of mix zones, in: *PETS*, 2009.
[19] J. Freudiger, R. Shokri and J.-P. Hubaux, Evaluating the privacy risk of location-based services, in: *Financial Cryptography*, 2011.
[20] S. Gaonkar, J. Li, R.R. Choudhury, L.P. Cox and A. Schmidt, Micro-blog: sharing and querying content through mobile phones and social participation, in: *MobiSys*, 2008, pp. 174–186.
[21] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, NY, USA, 1979.
[22] B. Gedik and L. Liu, Location privacy in mobile systems: A personalized anonymization model, in: *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, ICDCS'05*, IEEE Computer Society, 2005, pp. 620–629.
[23] E. Giordano, A. Tomatis, A. Ghosh, G. Pau and M. Gerla, C-VeT: an open research platform for VANETs: evaluation of peer to peer applications in vehicular networks, in: *IEEE VTC*, 2008.
[24] O. Goldreich, *Computational Complexity: A Conceptual Perspective*, Cambridge Univ. Press, New York, NY, USA, 2008.
[25] P. Golle and K. Partridge, On the anonymity of home/work location pairs, in: *Pervasive '09*, 2009.
[26] T.F. Gonzalez, A simple LP-free approximation algorithm for the minimum weight vertex cover problem, *Information Processing Letters* **54**(3) (1995), 129–131.
[27] R. Hassin and A. Levin, The minimum generalized vertex cover problem, *ACM Trans. Algorithms* **2**(1) (2006), 66–78.
[28] D. Hochbaum and A. Levin, The multi-integer set cover and the facility terminal cover problem, *Networks* **53** (2009), 63–66.

[29] B. Hoh, M. Gruteser, H. Xiong and A. Alrabady, Preserving privacy in GPS traces via uncertainty-aware path cloaking, in: *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '07*, 2007, pp. 161–171.
[30] J. Hromkovič, *Algorithms for Hard Problems*, Springer-Verlag, Berlin, Germany, 2004.
[31] http://en.wikipedia.org/wiki/Lovegetty.
[32] http://en.wikipedia.org/wiki/Bluedating.
[33] http://pleaserobme.com/.
[34] L. Huang, K. Matsuura, H. Yamane and K. Sezaki, Enhancing wireless location privacy using silent period, in: *IEEE WCNC*, 2005.
[35] L. Huang, H. Yamane, K. Matsuura and K. Sezaki, Towards modeling wireless location privacy, in: *PETS*, 2006.
[36] M. Humbert, M.H. Manshaei, J. Freudiger and J.-P. Hubaux, Tracking games in mobile networks, in: *GameSec*, 2010.
[37] R.M. Karp, *Complexity of Computer Computations*, Plenum Press, New York, NY, USA, 1972.
[38] M. Khiabani, Metro-sexual, available at: http://bit.ly/theranMetroSexual, 2009.
[39] V. Klee and G.J. Minty, How good is the simplex algorithm?, in: *Inequalities*, Vol. III, O. Shisha, ed., Academic Press, New York, NY, USA, 1972, pp. 159–175.
[40] M. Li, K. Sampigethaya, L. Huang and R. Poovendran, Swing & swap: user-centric approaches towards maximizing location privacy, in: *WPES*, 2006.
[41] C.J. Merlin and W.B. Heinzelman, A study of safety applications in vehicular networks, in: *MASS '05*, 2005.
[42] M.E. Nergiz, M. Atzori, Y. Saygin and B. Güç, Towards trajectory anonymization: a generalization-based approach, *Trans. Data Privacy* **2**(1) (2009), 47–75.
[43] B. Palanisamy and L. Liu, Mobimix: Protecting location privacy with mix zones over road networks, in: *ICDE '11*, 2011.
[44] E. Paulos and E. Goodman. The familiar stranger: anxiety, comfort, and play in public places, in: *CHI*, 2004, pp. 223–230.
[45] A. Pfitzmann and M. Köhntopp, Anonymity, unobservability, and pseudeonymity – a proposal for terminology, in: *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, 2001.
[46] Rhiain, Nokia instant community gets you social, available at: http://conversations.nokia.com/2010/05/25/nokia-instant-community-gets-you-social/.
[47] K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Matsuura and K. Sezaki, CARAVAN: Providing location privacy for VANET, in: *ESCAR*, 2005.
[48] E. Schoch, F. Kargl, T. Leinmüller, S. Schlott and P. Papadimitratos, Impact of pseudonym changes on geographic routing in VANETs, in: *ESAS*, 2006.
[49] R. Shokri, J. Freudiger, M. Jadliwala and J.-P. Hubaux, A distortion-based metric for location privacy, in: *ACM WPES*, 2009.
[50] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec and J.-P. Hubaux, Quantifying location privacy, in: *IEEE S&P*, 2011.
[51] Virginia State traffic data, available at: http://www.virginiadot.org/info/2009_traffic_data.asp.
[52] G. Xu, Y. Yang and J. Xu, Linear time algorithms for approximating the facility terminal cover problem, *Networks* **50** (2007), 118–126.
[53] Q. Xu, T. Mak, J. Ko and R. Sengupta, Vehicle-to-vehicle safety messaging in DSRC, in: *VANET*, 2004.
[54] Y. Ye, *Interior Point Algorithms: Theory and Analysis*, Wiley Interscience, 1997.