Probing a Set of Trajectories to Maximize Captured Movement

Sándor P. Fekete[1], Alexander Hill[1], Dominik Krupke[1], Tyler Mayer[2], Joseph S. B. Mitchell[2], Ojas Parekh[3], and Cynthia A. Phillips[3]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fekete,a.hill,d.krupke}@tu-bs.de
2 **Department of Applied Mathematics and Statistics, Stony Brook University, USA.**
tmayer93@gmail.com, jsbm@ams.sunysb.edu
3 **Sandia National Labs, USA**
{odparek,caphill}@sandia.gov

Abstract

We study the *Trajectory Capture Problem* (TCP), in which, for a given input set $\mathcal{T}$ of trajectories in the plane, and an integer $k \geq 2$, we seek to compute a set of k points ("portals") to maximize the total length of all subtrajectories of $\mathcal{T}$ between pairs of portals. This problem naturally arises in trajectory analysis and summarization.

We show that TCP is polynomial time solvable in 1D and NP-hard in 2D and then focus on tackling the TCP with integer linear programming to solve instances to provable optimality. We analyze this method on different classes of data, including benchmark instances that we generate. We demonstrate that we are able to compute provably optimal solutions for real-world instances.

1 Introduction

We study the Trajectory Capturing Problem (TCP): Given a set $\mathcal{T}$ of trajectories and an integer $k \geq 2$, determine a set of k *portals* (points) to maximize the sum of the lengths of the inter-portal subtrajectories in $\mathcal{T}$; such subtrajectories are said to be "captured" by the set of portals. This problem arises in placing a limited number of toll booths, cameras for average speed measurement, various other types of sensors, or abstract collections of focus points for sampling trajectories. For example, consider the scenario shown in Figure 1, which corresponds to more than 500,000 data points that arise from the trajectories of over 250 taxi cabs in San Francisco. Our goal is to identify a small subset of locations that allow us to capture as much of the movement pattern as possible.

Our main results are as follows. We mention proofs that the TCP is NP-hard and allows constant-factor approximation, based on geometric parameters. Our main focus is on demonstrating that even relatively large instances of the TCP can be solved to provable optimality with the help of Integer Programming.

Related Work. Related to the TCP is the well-studied geometric hitting set problem, in which one seeks a smallest cardinality set of points to hit a given set of lines, segments, or trajectories. These NP-hard problems are hard to approximate (below a threshold), and special cases have improved (constant-factor) approximation algorithms; see, e.g., the references in [5]. The difference to the TCP is the need to place at least two hit points on a trajectory in order to get (partial and weighted) credit for hitting it.

Buchin et al. [4] provide a set of positive and negative results for the similarly motivated problem of covering the most 'sites' in a weighted graph by a tree or a path of limited weight.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Figure 1 A set of taxi trajectories in the San Francisco Bay Area.

Limiting the weight and maximizing the covered vertices instead of limiting the spanning vertices and maximizing the covered weight in between, however, makes it fundamentally different to ours.

There is a vast literature on problems of analyzing, clustering, and summarizing a set of trajectories. In particular, notions of "flocks" and "meetings" have been formalized and studied algorithmically [2, 6, 9, 17]. Gudmundsson, van Kreveld, and Speckmann [7] define *leadership, convergence, and encounter* and provide exact and approximate algorithms to compute each. Andersson et al. [1] show that the *Leader-Problem* (LP) variants (*LP-Report-All, LP-max-Length, LP-Max-Size*) are all polynomially solvable and provide exact algorithms. Buchin et al. [3] present a framework to fully categorize trajectory grouping structures (grouping, merging, splitting, and ending of groups). They model grouping patterns in trajectory data using the Reeb graph in 3D and show several combinatorial bounds. Other approaches to trajectory summarization naturally include cluster analysis, of which there is a large body of related work. Li, Han, and Yang [12] consider rectilinear trajectories and show how to cluster with bounding rectangles of a given size. Several approaches (e.g., [14, 11, 10, 8]) consider density based methods for clustering sub-trajectories; Lee, Han, and Li [10] take it one step further by considering a two-level clustering hierarchy that first accounts for regional density and then considers lower-level movement patterns. Li, Ding, Han, and Kays [13] consider a problem (related to [7]) in which they seek to identify all *swarms* or groups of entities moving within an arbitrary shaped cluster for a certain, possibly disconnected, duration of time. Also, Uddin, Ravishankar, and Tsotras [16] consider finding what they call *regions of interest* in a trajectory database.

2 Preliminaries

We are given a set of trajectories $\mathcal{T}$, each specified by a sequence of points, e.g., in the Euclidean plane. We seek a set $P = \{p_1, \ldots, p_k\}$ of k *portals*, i.e., selected points that lie on some of the trajectories. While our practical study focuses on instances in which the trajectories $\mathcal{T}$ are purely spatial, e.g., given as polygonal chains or line segments in the plane, our methods apply equally well to more general portals and to trajectories that include the temporal component and live in space-time. More generally, we are given a graph $\mathcal{G}$, with length-weighted edges, and a set of (simple) paths within $\mathcal{G}$. We wish to determine a subset of k of the nodes of $\mathcal{G}$ that maximizes the sum of the (weighted) lengths of the subpaths (of the input paths) that link consecutive portals along the input paths.

We seek to compute a P that maximizes the total *captured weight* of subtrajectories between pairs of portals. For a trajectory $\tau \in \mathcal{T}$, if there are two or more portals of P that lie along τ, say $\{p_{i_1}, \ldots, p_{i_q}\}$ (for $q \geq 2$), then the subtrajectory, $\tau_{p_{i_1}, p_{i_q}}$, between p_{i_1} and p_{i_q} is *captured* by P, and we get credit for its weight $f(\tau_{p_{i_1}, p_{i_q}})$. (For many of our instances, $f(\tau_{p_{i_1}, p_{i_q}})$ corresponds to the Euclidean distances, denoted by $|\tau_{p_{i_1}, p_{i_q}}|$, but our methods generalize to other types of weights.) Let $f_P(\tau)$ denote the captured weight of trajectory τ by the portal set P. The Trajectory Capture Problem (TCP) is then to compute, for given $\mathcal{T}$ and k, a set of k portals $P = \{p_1, \ldots, p_k\}$ to maximize $\sum_{\tau \in \mathcal{T}} f_P(\tau)$.

3 One-Dimensional TCP

In the one-dimensional setting, the underlying graph $\mathcal{G}$ is a path, and the input trajectories $\mathcal{T} = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ are a set of subpaths of $\mathcal{G}$, specified by pairs of integers, a_i, b_i. A solution to the TCP then consists of k points, $P = \{p_1, \ldots, p_k\}$, w.l.o.g. indexed in sorted order, $p_1 < p_2 < \cdots < p_k$.

▶ **Theorem 1.** *The one-dimensional TCP can be solved exactly in polynomial time.*

Proof. For $i = 1, 2, \ldots, k-1$, let $V_i(x)$ be the maximum possible length of $\mathcal{T}$ captured by points $(p_i, \ldots, p_k)$, with $p_i = x \in \{a_1, \ldots, a_n, b_1, \ldots, b_n\}$; let $V_k(x) = 0$, for any x. Then, the value functions V_i satisfy the following dynamic programming recursion, for $i = 1, 2, \ldots, k-1$, and each $x \in \{a_1, \ldots, a_n, b_1, \ldots, b_n\}$:

$$V_i(x) = \max_{x' \in \{a_1, \ldots, a_n, b_1, \ldots, b_n\}, x' > x} \{V_{i+1}(x') + \sum_{j:(x,x') \subseteq (a_j, b_j)} (x' - x)\}.$$

The summation counts the length $(x' - x)$ once for each input interval that contains the interval (x, x'). We can compute the $O(nk)$ values $V_i(x)$ in time $O(n^2 k)$ by incrementally updating the summation as we consider values of x' in increasing order. ◀

4 Two-Dimensional TCP

The TCP is NP-hard (and hard to approximate) in the general graph setting (in which there is a general graph $\mathcal{G}$ given, and a set of paths within $\mathcal{G}$ is given as the input trajectories), even in the case of unweighted (weight-1) edges: We can simply give as input paths the set of single-edge paths in $\mathcal{G}$; thus, the TCP that asks if we can achieve total capture weight of $\binom{k}{2}$ is asking if there is a clique of size k in $\mathcal{G}$.

For TCP with input trajectories $\mathcal{T}$ given by (straight) line segments, we can show that the problem is also NP-hard by a reduction from Hitting Lines, see Fig. 2.

▶ **Theorem 2.** *The TCP for an input set $\mathcal{T}$ of n line segments, arbitrarily overlapping in the plane, is NP-hard.*

By a more intricate construction, we can show hardness even for axis-aligned segments.

On the other hand, we can provide the following approximation results, based on specific geometric parameters of the instances. The proofs are omitted from this short abstract.

▶ **Theorem 3.** *The TCP for an input set $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_K$, with each subset $\mathcal{T}_i$ having no crossing points, has a polynomial-time K-approximation algorithm.*

▶ **Theorem 4.** *The TCP for an input set $\mathcal{T}$ of arbitrarily overlapping/crossing trajectory paths in the plane having bounded depth D (i.e., no point of $\Re^2$ lies in more than D input trajectories) has a polynomial-time D-approximation algorithm.*

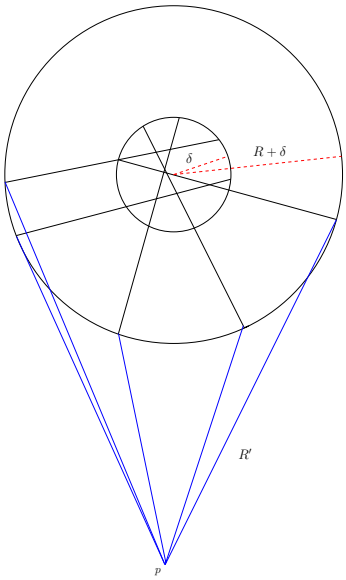


Figure 2 Reduction from Hitting Lines in which we have to decide whether we can hit all n (black) lines with l pins. There exists a circle that contains all intersections. For the reduction we create a second significantly larger circle and extend the lines to the lower half. We add n blue segments at the ends of the extended hitting lines to a point far below. The n blue lines are very long and captured by $n+1$ portals. We can additionally capture the part between the two circles with $l = k - (n+1)$ portals if there exists a hitting set with l pins.

5 Practical Results

5.1 Integer Linear Programming

We assume an input graph $\mathcal{G} = (V, E)$, and a collection of trajectories $\mathcal{T}$. Each trajectory $\tau \in \mathcal{T}$ is represented as a path in $\mathcal{G}$, and we refer to the two interchangeably. An edge e in a trajectory τ represents a portion of τ that is captured by any pair of nodes (portals) $p, q \in V$ that induces a subpath of τ containing e. Capturing an edge e earns an associated weight $f(e)$, and we seek to maximize the total weight earned by capturing edges subject to selecting at most k nodes. An IP can now be stated as follows.

$$\max \textstyle\sum_{\tau \in \mathcal{T}, e \in E(\tau)} f(e) x_{\tau,e}$$

$$\textstyle\sum_{v \in V} y_v \leq k \qquad \text{(Constraint 1)}$$

$$\forall \tau = (v_0, \ldots, v_l) \in \mathcal{T}:$$

$$\forall i \in 0, \ldots, l-1: \quad \begin{cases} x_{\tau,v_i v_{i+1}} \leq y_i & \text{if } i = 0, \\ x_{\tau,v_i v_{i+1}} \leq y_i + x_{\tau,v_{i-1} v_i} & \text{else} \end{cases} \qquad \text{(Constraint 2)}$$

$$\forall i \in 1, \ldots, l: \quad \begin{cases} x_{\tau,v_{i-1} v_i} \leq y_i & \text{if } i = l, \\ x_{\tau,v_{i-1} v_i} \leq y_i + x_{\tau,v_i v_{i+1}} & \text{else} \end{cases} \qquad \text{(Constraint 3)}$$

$$\forall v \in V, \tau \in e \in \tau: \quad x_{\tau,e}, y_v \in \{0,1\}$$

We have two types of Boolean variables: y_v, for $v \in V$, which indicates if node v is one of the k selected portals, and $x_{\tau,e}$, for edge $e \in E$ on trajectory τ, which indicates if the portion e of trajectory τ is captured by selected portals. For an edge e, there are distinct variables, $x_{\tau,e}, x_{\tau',e}$, for trajectories $\tau \neq \tau'$ because e may be captured in τ but not in τ'.

Our objective function maximizes the weighted sum of captured trajectory edges, where

$E(\tau)$ denotes the edges of τ in $\mathcal{G}$, and $f(e)$ is the weight (i.e. length) of edge e. (Optionally, we could have trajectory-dependent weights on edges.) Constraint 1 limits the number ($\leq k$) of selected portals. Constraints 2 and 3 enforce that, in order for an edge to be captured as part of trajectory τ, there must be a selected portal in each direction; either there is a selected portal at the next node, or the following trajectory edge is also captured. In the latter case, because τ has no cycle (it is a simple path), there must be a selected portal on τ at some point in that direction if any portion of τ is to be captured.

Figure 3 shows an example; using "x_i" as shorthand for the IP variables $x_{\tau,v_iv_{i+1}}$ that correspond to the edges along trajectory path $\tau = (v_0, v_1, \ldots, v_6)$. Constraints 2 are: $x_0 \leq y_0$, $x_1 \leq y_1 + x_0$, $x_2 \leq y_2 + x_1, \ldots, x_5 \leq y_5 + x_4$. Constraints 3 are: $x_5 \leq y_6$, $x_4 \leq y_5 + x_5$, $x_3 \leq y_4 + x_4, \ldots, x_0 \leq y_1 + x_1$. With portals selected at v_1 and v_4 (i.e., with $y_1 = y_4 = 1$, $y_i = 0$ otherwise), we get constraints $x_0 \leq 0$, $x_5 \leq 0$, and $x_4 \leq y_5 + x_5 = 0$, implying that only the subpath (v_1, v_2, v_3, v_4) of τ contributes to the objective function.

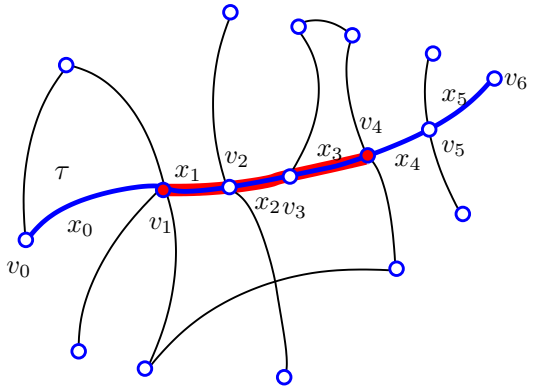


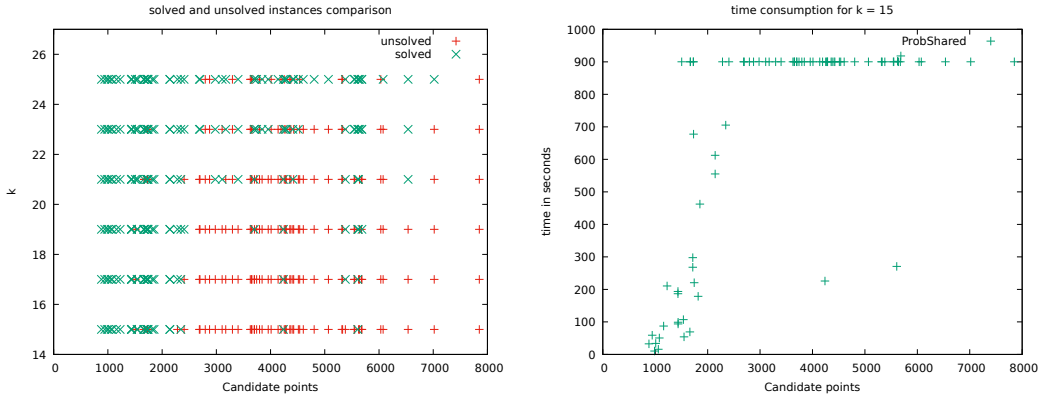
Figure 3 Formulating the IP: Trajectory $\tau = (v_0, v_1, \ldots, v_6)$ is highlighted in blue. With selected portals at v_1 and v_4, the portion highlighted in red is captured.

5.2 Experimental Evaluation

We used three kinds of instances for experimental evaluation: **(1)** Instances based on 10-20% of the segments/edges in a straight line embedding of a complete graph with 35-55 random points (likely to have high degree while most intersections only have degree 4) where the trajectories are the full edges, **(2)** random non-overlapping axis-parallel segments, and **(3)** trajectories based on real-world taxi tracking data. For all instances, we used CPLEX (V12.7.1 with default settings) with a time limit of 900 s on Intel(R) Core(TM) i7-4770 with 32 GB. The code and data is available at `https://github.com/ahillbs/trajectory_capturing`.

For type **(1)** we see in Figure 4 that instances with up to ~ 2500 candidate points (i.e., nodes at intersection points in the graph, where selected portals can be placed) can be solved for $15 \leq k \leq 23$ to provable optimality within the time limit. Instances with more than 2500 candidate points are most often not solved for $k < 23$. For instances with $k \geq 23$, the problem seems to be easier to solve. In Figure 4b we see that for $k = 15$ and between 1500 and 2000 candidate points, instances start to become very difficult to solve. On the other hand, for $k \geq 23$ instances are still solvable for more than 2500 candidate points.

For **(2)** axis-parallel non-overlapping segments (see Fig. 5), we do not know the hardness but one can devise a simple 2-approximation via dynamic programming. Indeed, these



(a) Instances solved within 900 s.

(b) Time used for instances with $k = 15$.

Figure 4 Instances with a percentage of segments between 35-55 random points. The problem is harder for placing only 15 portals than placing 25 portals.

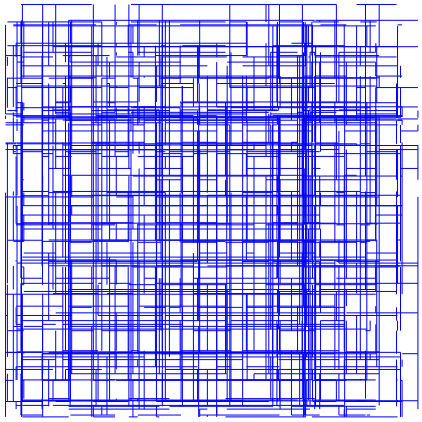


Figure 5 An instance with 1100 axis-parallel segments and 8200-8500 candidate points.

instances are much easier to solve, see Fig. 7. We have solved instances with 2000 segments and 19,000 candidate points. Furthermore, for instances with up to 20,000 candidate points, the integrality gap was never larger than 20% for $k = 5$, 7% for $k = 10$, 5% for $k = 15$ and less than 2.5% for larger k.

For **(3)** we studied taxi cab trajectories in the San Francisco Bay Area. Figure 6 shows our results for $k = 5$ optimal portals. (The data is based on 375 vehicles, sampled every 5 minutes, 288 times per day, for one week [15]; the satellite imagery is courtesy of Planet Labs.) Our experiments included runs on 30 instances, with k ranging from 5 to 11, on sets of 10 to 120 trajectories of varying lengths (comprised of 1300 to 3700 edges, and 600 to 1800 vertices). The measurements of the trajectories are snapped to a regular grid graph to counteract GPS inaccuracy. Solution times were up to 200 seconds of computation, with most instances taking less than 10 seconds.

6 Conclusion

The main theoretical questions that remain open are: (1) Is TCP NP-hard for axis-aligned segments that are allowed to intersect at single points (endpoints or crossing points) but

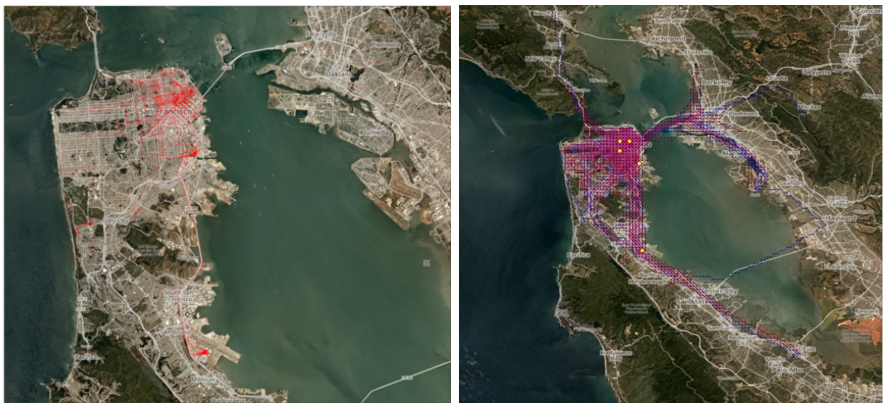


Figure 6 (Left) Data points before processing. (Right) Solution to real-world TCP instance: An optimal set of $k = 5$ portals are highlighted by yellow dots. The captured parts of the trajectories are highlighted in red while not captured parts are shown in blue.

not to overlap in subsegments? (our construction relied on overlapping segments); and, (2) Can we improve the approximation factor (of K) for a set of trajectories that is the union of K subsets, each of which is noncrossing? On the practical side, there is a wide range of real-world classes of instances that we can explore.

References

1 Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12(4):497–528, 2008.

2 Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.

3 Kevin Buchin, Maike Buchin, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Trajectory grouping structure. In *Algorithms and data structures*, pages 219–230. Springer, 2013.

4 Kevin Buchin, Sergio Cabello, Joachim Gudmundsson, Maarten Löffler, Jun Luo, Günter Rote, Rodrigo I Silveira, Bettina Speckmann, and Thomas Wolle. Finding the most relevant fragments in networks. *J. Graph Algorithms Appl.*, 14(2):307–336, 2010.

5 Sándor P Fekete, Kan Huang, Joseph SB Mitchell, Ojas Parekh, and Cynthia A Phillips. Geometric hitting set for segments of few orientations. *Theory of Computing Systems*, 62(2):268–303, 2018.

6 Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proc. of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, pages 35–42. ACM, 2006.

7 Joachim Gudmundsson, Marc van Kreveld, and Bettina Speckmann. Efficient detection of patterns in 2d trajectories of moving points. *Geoinformatica*, 11(2):195–215, 2007.

8 Marios Hadjieleftheriou, George Kollios, Dimitrios Gunopulos, and Vassilis J Tsotras. Online discovery of dense areas in spatio-temporal databases. In *Advances in Spatial and Temporal Databases*, pages 306–324. Springer, 2003.

9 Patrick Laube, Matt Duckham, and Thomas Wolle. Decentralized movement pattern detection amongst mobile geosensor nodes. In *Geographic Information Science*, pages 199–216. Springer, 2008.

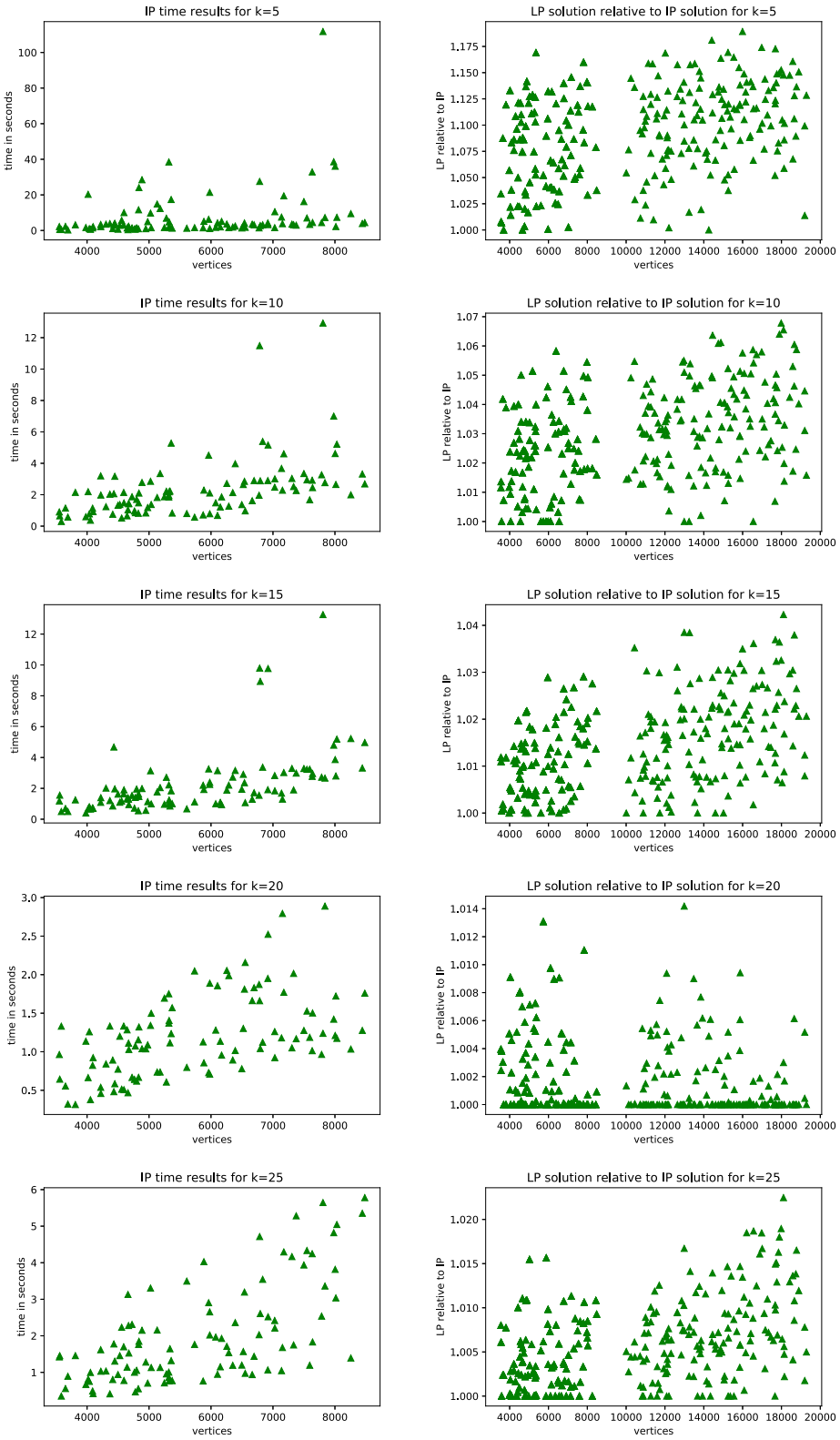


Figure 7 IP runtime and integrality gap for axis-parallel instances and $k = 5, 10, \ldots, 25$.

10 Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

11 Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proc. of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 593–604. ACM, 2007.

12 Yifan Li, Jiawei Han, and Jiong Yang. Clustering moving objects. In *Proc. Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 617–622. ACM, 2004.

13 Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2):723–734, 2010.

14 Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.

15 Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24), doi:10.15783/c7j010, February 2009.

16 Md Reaz Uddin, Chinya Ravishankar, and Vassilis J Tsotras. Finding regions of interest from trajectory data. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 39–48. IEEE, 2011.

17 Marcos R Vieira, Petko Bakalov, and Vassilis J Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proc. of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 286–295. ACM, 2009.