On Theoretical Trajectory Planning of Multiple Drones To Minimize Latency in Search-and-reconnaissance Operations

Donghyun Kim, *Senior Member, IEEE,* Lirong Xue, Deying Li,
Yuqing Zhu, *Member, IEEE,* Wei Wang, Alade O. Tokuta, *Member, IEEE*

Abstract—Following the recent advances in drone technologies, various algorithmic optimization problems related to the effective operation of drones are drawing lots of attentions. This paper considers two interesting multiple-drone-assisted search-and-reconnaissance scenarios, in each of which, the trajectory optimization of multiple drones is of great significance to minimize the latency in the system. In the first scenario, multiple drones, whose moments of mobilization are not necessarily same, are trying to urgently collect intelligence from a given points of interest, and we would like to minimize the task completion time, i.e. the time period between the moment that the first drone commences its operation to the moment that the intelligence from all of the points are collected, by optimizing their trajectories. In the second scenario, multiple drones with different speed, are hovering around the same routes to regularly collect intelligence from highly geographically-diversified points of interest over an extended time period, and we would like to minimize the worst-case data refreshment rate, the largest time gap between two consecutive observations over the same point of interest. In this paper, we formally define each problem, prove its NP-hardness, and propose an approximation algorithm for it. We also conduct a simulation to study the performance of our result.

Index Terms—Drone, trajectory planning, mobile sensors, approximation algorithm, traveling salesman problem.

1 Introduction

The recent developments in robotic technologies realized various autonomous search-and-reconnaissance systems, in which drones are in charge of critical operations such as monitoring a number of geographically dispersed points of interest. There are a number of important applications of such systems. For instance, a group of drones can be used by the first responders who just arrived at a serious disaster scene such as Fukushima nuclear disaster site in Japan from 2011, and are looking for any remain survivals in the area who are too difficult or dangerous to find using a traditional way. Another good example is a long term environmental monitoring of climate change by using multiple drones hovering around a vast area of interest over an extended time period. The trajectory planning of the drones in those systems is known to be very significant since it is directly related to the performance such as latency, energy-efficiency etc. of the systems [4], [7], [8], [9], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [25].

In theory, the goal of travelling salesman problem (TSP) is to find a minimum cost tour for a traveler to visit all of the points of interest and come back to the origin. The TSP is a well-known NP-hard problem, which means that there is no efficient (polynomial running time) algorithm to find the best possible solution for it. Due to the reason, many efforts are dedicated to introduce a polynomial running time sub-optimal algorithm with reasonable performance guarantee, which is well-known as an approximation algorithm. One variation of TSP is the travelling salesman problem with neighborhood (TSPN). Unlike TSP, in TSPN, a point of interest is visited by the traveler once their distance is no greater than a certain threshold value. Frequently, TSPN is considered as a good abstraction of a drone trajectory optimization problem whose goal is to minimize the completion time (latency) to collect the knowledge from all of the sites of interest using a drone with a camera sensor, where the threshold value represents the effective sensing range of the camera sensor. When the sensing range goes to zero, TSPN becomes equivalent to TSP, and therefore, TSPN can be considered as a general case of TSP. This means that if TSPN is not NP-hard, then TSP should not be. As a result, TSPN is also NP-hard.

Recently, Kim et al. have investigated the problem of computing the trajectories of multiple drones to collect knowledge from a given set of sites of interest with

- *D. Kim is with Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA. E-mail: donghyun.kim@kennesaw.edu.*
- *L. Xue is with Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, USA. Email: lirongx@princeton.edu.*
- *D. Li is with Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University), MOE, School of Information, Renmin University of China, P.R. China. Email: deyingli@ruc.edu.cn.*
- *Y. Zhu is with Department of Computer Science, California State University, Los Angeles, CA USA. Email: yuqing.zhu@calstatela.edu.*
- *W. Wang is with School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China. Email: wang_weiw@163.com*
- *A.O. Tokuta is with Department of Mathematics and Physics, North Carolina Central University, Durham, NC 27707. E-mail: atokuta@nccu.edu.*
- *D. Li is the corresponding author.*
- *A preliminary version of this paper has been appeared in INFOCOM 2014 [1].*

the optimization goal of minimizing the latency [2], [3]. Compared to the trajectory optimization problems with a single drone, the case with multiple drones is even more challenging. This is because now we have to optimize the trajectories of each drone and determine the workload of each drone, i.e. which drone will cover which target, at the same time, but any decision on one aspect affects the decision on the other aspect. Apparently, this problem is also NP-hard as it is a general case of TSPN. In their work, Kim et al. have introduced approximation algorithms for two different variations of the problem. In detail, given a group of n targets and k drones, two different problems, namely the k-traveling salesman problem with neighborhood (k-TSPN) and the k-rooted path cover problem with neighborhood (k-PCPN) are defined. The common goal of the two problems is to determine the trajectories of the k drones such that

(a) each trajectory originates from the location of the corresponding drone which will move around following the trajectory,
(b) each ground target is covered by a drone when the distance between the target and the drone becomes no greater than a given threshold level, and
(c) the length of the longest trajectory among the k trajectories is as small as possible as it will determine the latency (the task completion time) of the whole system.

Despite the similarity, k-TSPN and k-PCPN are not necessarily same in the definition of trajectory. That is, k-TSPN assumes each drone can transmit data to the operator only at its basement (starting location), and therefore pursues k-rooted tours, and k-PCPN assumes each drone is connected to the operator at any location while moving and thus seeks for k-rooted paths.

Outline of Problems. In this paper, we consider the following two different scenarios in which multiple drones are used for search-and-reconnaissance and the optimization of the trajectories of drones has a critical impact on the performance (latency) of the whole system.

In the first scenario, multiple drones, whose moments of mobilization are not necessarily same, are trying to urgently collect intelligence from a given set of points of interest, and we would like to minimize the task completion time, i.e. the time period between the moment that the first drone commences its operation to the moment that the intelligence from all of the points are collected, by optimizing their trajectories. This new trajectory optimization issue is formally defined into two specifics problems, the k asynchronous travelling salesman problem with neighborhood (k-ATSPN) and the k-rooted asynchronous path cover problem with neighborhood (k-APCPN). Given a set of n stationary targets and k homogenous (i.e. with the same physical capability) drones, which may be mobilized at possibly different time, the common goal of both problems is to collect the knowledge from all of the points with minimum latency. However, they differ in the assumption on when the operator can extract the collected information from the drone. Clearly, an algorithm for k-TSPN (and k-PCPN) cannot be applied to k-ATSPN (and k-APCPN), directly.

In the second scenario, multiple drones with possibly different speed, are hovering around the same routes to routinely collect intelligence from highly geographically-dispersed points of interest over an extended time period, and we would like to minimize the worst-case data refreshment rate, the largest time gap between two consecutive observations over the same point of interest. We formally define this problem as the k inhomogeneous travelling salesman problem with neighborhood (k-ITSPN). In detail, given a set of stationary targets and k (inhomogeneous) drones with different speed, k-ITSPN aims to find the tour for each drone such that the data refresh rate (the maximum inter-arrival time of a drone to collect intelligence from the same target) is minimized. One good example of such scenario is the long term environmental monitoring system, in which we operate various drones to follow tours and repeatedly observe the points of interest over a vast area over a long time period. Note that in such a case, the trajectory of each data ferry becomes a tour, which does not necessarily include the starting point (base station) of each data ferry.

Summary of Contributions. The summary of contributions of this paper is as follows.

(a) We identify new problems k-ATSPN, k-APCPN, and k-ITSPN, and show they are NP-hard.
(b) To solve k-ATSPN and k-APCPN, we introduce a new graph theory based technique to reduce the problems into known NP-hard problems, each of which has a constant factor approximation algorithm for it. As a result, we obtain a constant factor approximation of each of k-ATSPN and k-APCPN.
(c) To solve k-ITSPN, we introduce a new NP-hard problem, namely the weight k-tree cover problem (Wk-TCP), whose goal is to find to find k-trees spanning over a given set of nodes such that the maximum of the cost of a tree over the weight of the tree is minimized. Then, we propose an approximation algorithm for Wk-TCP. Using this algorithm, we obtain an approximation algorithm for k-ITSPN.
(d) We perform a simulation and evaluate the core contribution of this paper, the approximation algorithm for Wk-TCP, by comparing the average cost of the output of the algorithm with the lower bound. Our result shows the algorithm works better as the size of network grows and k increases.

The rest of this paper is organized as follow. Section 2 presents some related work. We introduce some notations and definitions in Section 3. Our main contributions, the constant factor approximations for k-ATSPN and k-APCPN are in Section 4, and a new approximation

algorithm for k-ITSPN is in Section 5. Section 6 presents the simulation results and corresponding discussions. Finally, we conclude this paper in Section 7.

2 Related Work

In the literature, the mobility-assisted knowledge collection schemes are categorized into the following three classes [22]: random mobility, predictable mobility, and controlled mobility. The works in the first class such as Data Mobile Ubiquitous LAN Extensions (MULEs) [23] uses mobile nodes whose trajectories are not controllable and not predictable to opportunistically deliver data from sensor nodes. In [24], Chakrabarti et al. used mobile nodes with predictable mobility for message routing to conserve the energy of wireless sensor network. The concept of fully controllable mobile data collector in wireless sensor networks is initially introduced by Ma and Yang [4]. In this work, a fully controllable mobile node called SenCar is used to collect data from sensor nodes. The authors also argued about the importance of the trajectory optimization issue for the mobile node to maximize its efficiency. Over years, due to the reason, the trajectory optimization issue of mobile data collectors in wireless sensor networks has attracted a lot of attentions by network research community [2], [3], [4], [5], [6], [7], [8], [9], [12], [13], [14], [15], [16], [17], [18], [19], [20], [25].

In theory, the problem of computing minimum length tour of a mobile node visiting a given set of nodes is known as the traveling salesman problem (TSP), which is NP-hard [34]. Its well-known variation, traveling salesman problem with neighborhood (TSPN), whose goal is, given a set of nodes each of which has a uniform circular neighborhood with radius 1, to find a tour visiting the neighborhood areas of all nodes with minimum total length, is also NP-hard. Frequently, TSP and TSPN are used to abstract a single data collector trajectory optimization problems which aim to minimize the latency to collect data from a given set of remote sensors using the mobile node [9], [25], [26]. Dumitrescu and Mitchell proposed a polynomial time $(\pi + 8)(1 + \epsilon)$-approximation algorithm for TSPN in which the neighborhood areas of the nodes are uniform circular shaped and not necessarily disjoint, where ϵ is a very small positive integer [27]. They also proposed $(1+\epsilon)$-approximation algorithm for this TSPN in which the neighborhood areas of the nodes are disjoint and arbitrary fat-regions [30]. In [35], Mitchell proposed a constant factor approximation algorithm for TSPN in which the neighborhood areas of the nodes are pairwise disjoint and arbitrary shaped fat-regions. In [10], he also introduced a $(1+\epsilon)$-approximation algorithm for TSPN in which the neighborhood areas of the nodes are not necessarily disjoint and arbitrary fat-regions. Note that these results on single mobile node is not directly applicable to multiple mobile nodes case since the problems in the later class requires the optimizations of trajectories and workload distribution, i.e. which mobile node should be which sensor node, to be performed at the same time, which is more challenging.

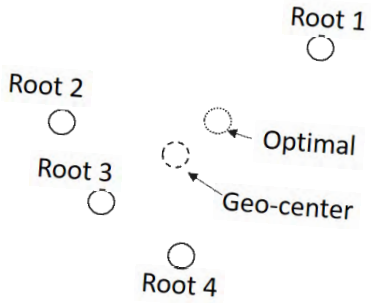


Fig. 1. Once an approximation algorithm for the k-TSP is applied, we can obtain a feasible solution for k-TSPN by extending each tour from the orientation to each root. However, the quality of this approach is unbounded with respect to k-TSPN as the distance between the optimal location and the geo-center can be arbitrary large depending on the locations of mobile nodes (roots).

The k-traveling salesman problem (k-TSP) aims to find k-tours starting from the same orientation such that the length of the longest tour is minimized, and is a well-known NP-hard problem. The first constant factor approximation algorithm for k-TSP, namely k-SPLITOUR, is proposed by Frederickson et al [28]. However, this result is not directly applicable to our problems (k-TSPN and k-PCPN) because k-TSP assumes the starting points of the k tours are same while this is not true for k-TSPN and k-PCPN. One may attempt to design an approximation algorithm for k-TSPN (or k-PCPN) based on k-SPLITOUR. In this case, it is necessary to convert a given k-TSPN (or k-PCPN) instance to a k-TSP instance first, apply an approximation algorithm for k-TSP for this problem instance, then convert the output of the previous step. Most importantly, the error rate caused by each step has to be bounded so that an approximation algorithm for k-TSPN (or k-PCPN) can be obtained. This requires to determine where to put a virtual (common) orientation for all mobile nodes (roots). Apparently, the size of the candidate positions for the orientation is unlimited as the orientation can be located in any position in the 2-D space. Therefore, it is very difficult to determine the best possible position for the orientation, and a simple heuristic algorithm does not work well. For instance, consider the geo-center location for the given roots as the orientation. However, as the k-TSPN (and k-PCPN) is a min-max time problem, this will make any algorithm using this orientation determination strategy with a good approximation algorithm for k-KSP (such as k-SPLITOUR) works arbitrarily bad even after we assume we have an exact algorithm for k-TSP (see Figure 1).

In [2], [3], Kim et al. introduced the k-traveling salesman problem with neighborhood (k-TSPN), whose goal

is to find k rooted tours such that (a) each tour starts from a distinct root, (b) the neighborhood area of each node, which is a uniform circular area and not pairwise disjoint with each other, is visited by some tour, and (c) the length of the longest tour is minimized. This work proposed a constant factor approximation algorithm for k-TSPN, as well as the k-rooted path cover problem with neighborhood (k-PCPN) whose goal is similar to k-TSPN but it is looking for k-paths instead of k-tours. The problems of our interest, k-ATSPN and k-APCPN are similar to k-TSPN and k-PCPN. However, the approximation algorithms for k-TSPN and k-PCPN are not directly applicable to k-ATSPN and k-APCPN, respectively. This is because while k-TSPN and k-PCPN assume all drones are available at the beginning of a given mission and all drones have the same speed, these assumptions are removed for their "asynchronous" versions, i.e. k-ATSPN and k-APCPN.

Given a set of nodes and k roots, the k-rooted tree cover problem (k-RTCP) is to find a set of trees each of which spans over a distinct root and a subset of nodes such that each node is visited by a tree and the total edge length of the heaviest tree is minimized. In [29], the authors has proposed a $(4+\epsilon)$-approximation algorithm for k-RTCP as well as another $(4+\epsilon)$-approximation algorithm for k-tree cover problem (k-TCP), which is a variation of k-RTCP without the concept of roots. This unrooted version is also investigated by [32], [33]. In [2], [3], the approximation algorithm for k-RTCP is used to design approximation algorithm for k-TSPN and k-PCPN, respectively. In this paper, similarly, we design an approximation algorithm for k-ITSPN by first designing a variation of k-TCP in which each data ferry has a different speed, and second, use this as a basis of the algorithm for k-ITSPN.

3 Notations and Definitions

$V = \{v_1, \cdots, v_n\}$ is a set of n targets (or equivalently sites, or points) of interest. $R = \{r_1, \cdots, r_k\}$ is a given set of k roots (i.e. drones in our context, each of which is with a fully rotational camera). For each $v_i \in V$, $N(v_i)$ is the neighborhood area of v_i, which is a disk centered at v_i with normalized radius (the sensing range of the camera sensor) 1. That is, once a drone is within the neighborhood area of v_i (or equivalently if the drone is within the disk centered at v_i), then it can cover (observe) v_i.

$G = (V, E)$ is a graph with a node set $V = V(G)$ and an edge (which can be either a straight line or a curve segment) set $E = E(G)$, respectively. Given a node set V', $E_{V'}$ denotes the set of edges connecting the nodes in V'. Given a subgraph G', $Len(G')$ is the gross sum of the length of the edges in G', i.e.

$$\sum_{(v_i,v_j)\in E(G')} Len(v_i, v_j),$$

where $Len(v_i, v_j)$ is the length of the edge between two nodes v_i and v_j in $V(G')$. $Eucdist(v_i, v_j)$ is the distance between two nodes v_i and v_j in the Euclidean space. Any neighborhood areas (disks) of two nodes are said to "touch" with each other if the borders of the neighborhood areas are adjacent with each other. Now, we introduce the following definitions.

Definition 1 (TSP). *Given a set V of n targets, the* traveling salesman problem (TSP) *is to find a tour U such that*

(a) the tour U starts from and ends at the same node $v \in V$,
(b) U visits the rest of the nodes in $V \setminus \{v\}$, and
(c) $Len(U)$ is minimized.

Definition 2 (TSPN). *Given a set V of n targets, the* traveling salesman problem with neighborhood (TSPN) *is to find a tour U such that*

(a) the tour U starts from and ends at the same node $v \in V$,
(b) for each $v \in V$, $\exists u \in V(U)$ such that u is in (or on the border of) $N(v)$, and
(c) $Len(U)$ is minimized.

Definition 3 (k-TSPN). *Given a set V of n targets and a positive integer k, the k-*traveling salesman problem with neighborhood (k-TSPN) *is to find a set of k rooted-tours $U = \{U_1, U_2, \cdots, U_k\}$ such that*

(a) each tour U_i begins from and ends at r_i,
(b) for each $v_j \in V$, $\exists u \in V(U_i)$ for some $U_i \in U$ such that u is in (or on the border of) $N(v_j)$, and
(c) $Cost(U) = \max_{1\le i\le k} Len(U_i)$ is minimized.

Definition 4 (k-PCPN). *Given a set V of n targets and a positive integer k, the k-*rooted path cover problem with neighborhood (k-PCPN) *is to find a set of k rooted-paths $P = \{P_1, P_2, \cdots, P_k\}$ such that*

(a) each path P_i begins from r_i,
(b) for each $v_j \in V$, $\exists u \in V(P_i)$ for some $P_i \in P$ such that u is in (or on the border of) $N(v_j)$, and
(c) $Cost(P) = \max_{1\le i\le k} Len(P_i)$ is minimized.

Next, we introduce the definitions of two new trajectory optimization problems k-APCPN, k-APCPN, and k-ITSPN and clarify the differences among the problems.

Definition 5 (k-ATSPN). *Given a set V of n stationary targets and a set R of k drones $\{r_1, \cdots, r_k\}$ with uniform speed s, each of which is mobilized after $\{t_1, \cdots, t_k\}$ time units later, respectively from the initial deployment of the drones, the k* asynchronous TSPN (k-ATSPN) *is to find the tours of the k drones, $U = \{U_1, \cdots, U_k\}$ such that*

(a) each tour U_i begins from and ends at r_i,
(b) for each $v_j \in V$, $\exists u \in V(U_i)$ for some $U_i \in U$ such that u is in (or on the border of) $N(v_j)$, and
(c) $Cost(U) = \max_{1\le i\le k} [\frac{Len(U_i)}{s} + t_i]$ is minimized.

Definition 6 (k-APCPN). *Given a set V of n stationary targets and a set R of k drones $\{r_1, \cdots, r_k\}$ with uniform speed s, each of which is mobilized after $\{t_1, \cdots, t_k\}$ time units later, respectively from the initial deployment of the drones, the k* asynchronous (k-APCPN) *is to find the paths of the k drones, $P = \{P_1, \cdots, P_k\}$ such that*

(a) each path P_i begins from r_i,
(b) for each $v_j \in V$, $\exists u \in V(P_i)$ for some $P_i \in P$ such that u is in (or on the border of) $N(v_j)$, and
(c) $Cost(P) = \max_{1 \le i \le k} [\frac{Len(P_i)}{s} + t_i]$ is minimized.

k-ATSPN and k-APCPN are similar to k-TSPN and k-PCPN, respectively. However, the cost function of those new problems becomes different since each drone r_i will be mobilized at t_i from the initialization of the whole system. As a result, the result from our previous work in [2], [3] is not directly applicable to solve the new problems. It is shown that k-ATSPN and k-APCPN are NP-hard since by setting $t_i = 0$ for every i, the problems become equivalent to their counterparts, k-TSPN and k-PCPN, respectively, which are NP-hard.

Definition 7 (k-ITSPN)**.** *Given a set V of n stationary targets and a set R of k drones $\{r_1, \cdots, r_k\}$ with unique speed $\{s_1, \cdots, s_k\}$, respectively, the k* inhomogeneous TSPN (k-ITSPN) *is to find the tours of the k drones, $U = \{U_1, \cdots, U_k\}$ such that*
(a) for each $v_j \in V$, $\exists u \in V(U_i)$ for some $U_i \in U$ such that u is in (or on the border of) $N(v_j)$, and
(b) $Cost(U) = \max_{1 \le i \le k} [\frac{Len(U_i)}{s_i}]$ is minimized.

k-ITSPN assumes that the targets of interest are spread over a vast area and therefore, no two targets are within the sensing range of the same drone at the same time. In reality, if some ground sensors are connected with each other (i.e. within the communication rage of each other), we assume one of them is selected as a cluster-head for them, and we will only consider those cluster-heads and isolated ground sensors for the problem. In most cases, the sensor range of each node is no greater than the communication range of the node. Therefore, this assumption also means that the ground sensors which are used as the input of the problem have disjoint sensing ranges. As a result, k-ITSPN is different from the problems we discuss so far since the tour does not include any root. Furthermore, in the cost function, the cost incurred by the trajectory of each drone r_i is divided by its speed s_i, which incurs additional challenge and therefore the result of [2], [3] is not directly applicable. Now, we show k-ITSPN is NP-hard. Clearly, k-ITSPN is NP-hard since by setting $k = 1$, k-ITSPN becomes equivalent to TSPN, which is known to be NP-hard.

4 Constant Factor Approximations for k-ATSPN and k-APCPN

In this section, we propose constant factor approximations for k-ATSPN and k-APCPN. For this purpose, we first propose an induction strategy from a k-ATSPN problem instance (and k-APCPN) to k-TSPN (and k-PCPN), respectively. In detail, consider a k-APCPN instance, $V = \{v_1, \cdots, v_n\}$ (the set of n points of interest), $R = \{r_1, \cdots, r_k\}$ (the set of k roots, or equivalently drones), $T = \{t_1, \cdots, t_k\}$ (the moment at which each drone commences its operation), s (the uniform speed of the drones). Then, a new edge weighted graph $\widehat{G} = (\widehat{V}, \widehat{E}, c_E)$ is constructed as follows.

(a) Construct two empty node set V_1 and V_2. Set $V_1 \leftarrow V = \{v_1, \cdots, v_n\}$ and $V_2 \leftarrow \{r_1, \cdots, r_k\}$. Set $\widehat{V} \leftarrow V_1 \bigcup V_2$.
(b) For each v_i, v_j pair in $\widehat{V}$, add an edge (v_i, v_j) to $\widehat{E}$ and set the edge weight $c_E(v_i, v_j)$ to be their Euclidean distance $Eucdist(v_i, v_j)$.
(c) For each r_i, v_j pair in $\widehat{V}$, add an edge (r_i, v_j) to $\widehat{E}$ and set the edge weight $c_E(r_i, v_j)$ to be $Eucdist(r_i, v_j) + s \cdot t_i$.

As a result, $\widehat{G}$ is an edge-weighted complete graph with a node set $\widehat{V} \bigcup \widehat{R}$. Our key observations are that

(a) each drone will spend exactly "the distance between two locations divided by its speed" unit time to move from one location to another location, but
(b) such travel time increases if there exists a delay.

In detail, if a drone (with a speed s unit distance per unit time) departs t_i unit time late, it will cost additional $s \cdot t_i$ unit time to complete a given travel. Thanks to $\widehat{G}$ which was constructed with this key observations in mind, we can effectively eliminate the condition that each drone may be available at a different timing, and apply existing approximation algorithms for the case in which all drones are available from the initial moment, i.e. the constant factor approximation algorithm for k-TSPN to solve k-ATSPN and the constant factor approximation algorithm for k-PCPN to solve k-APCPN [2], [3].

Briefly speaking, the constant factor approximation algorithm for k-TSPN (and k-PCPN) consists of the following steps.

- Step 1: Given a k-TSPN (and k-PCPN) problem instance V, R, apply the $(4+\epsilon)$-approximation algorithm for k-RTCP in [29]. As a result, we have k rooted-trees $X = \{X_1, X_2, \cdots, X_k\}$ of V such that (a) each tree includes a single distinct root in R and a subset of V and (b) $X_i \bigcap X_j = \emptyset$ for every i and j pair, $i \neq j$.
- Step 2: For each $X_i \in X$, a neighborhood tree X_i^N rooted at r_i and spanning over the neighborhood areas of $V(X_i) \setminus \{r_i\}$ is computed as done in [2], [3].
- Step 3: Each neighborhood tree X_i^N is converted to a tour for k-TSPN (and a path for k-PCPN) using Christofides's 1.5-approximation algorithm for TSP [31].

To obtain a constant factor approximation for k-ATSPN and k-APCPN, we perform followings.

- Step 1: Given a k-ATSPN (and k-APCPN) problem instance $V = \{v_1, \cdots, v_n\}, R = \{r_1, \cdots, r_k\}, T = \{t_1, \cdots, t_k\}, s$, construct a new edge weighted graph $\widehat{G} = (\widehat{V}, \widehat{E}, c_E)$ as explained above. Apply the $(4+\epsilon)$-approximation algorithm for k-RTCP in [29] over $\widehat{G}$ (this approximation algorithm does not use any geometry, and thus works on any graph including $\widehat{G}$ and its performance ratio is preserved) with

$\{r_1, \cdots, r_k\} \subset V(\widehat{G})$ as the distinct k roots. As a result, we have k rooted-trees $X = \{X_1, X_2, \cdots, X_k\}$ of V such that each tree includes a single distinct root r_i and a subset of V such that $X_i \bigcap X_j = \emptyset$ for all i and j pairs.

- Step 2: For each $X_i \in X$, a neighborhood tree X_i^N rooted at r_i and spanning over the neighborhood areas of $V(X_i) \setminus \{r_i\}$ is computed.
- Step 3: Each neighborhood tree X_i^N is converted to a tour for k-ATSPN (and a path for k-APCPN) using Christofides's 1.5-approximation algorithm for TSP [31].

Theorem 1. *There exists a constant factor approximation algorithms for k-ATSPN and k-APCPN.*

Proof. In [3], there exists a polynomial time constant factor approximation algorithm for each of k-TSPN and k-PCPN, respectively. In this section, we have introduced polynomial time reduction from k-ATSPN and k-APCPN to k-TSPN and k-PCPN, respectively. Therefore, the constant factor approximation algorithms for k-TSPN and k-PCPN can be used as the constant factor approximation algorithms for k-ATSPN and k-APCPN, and thus this theorem is true. □

5 A New Constant Factor Approximation for k-ITSPN

In this section, we propose a new approximation algorithm for k-ITSPN. In Section 5.1, we introduce a new optimization problem, namely the weighted k-tree cover problem (Wk-TCP), and propose a new approximation algorithm for it. In Section 5.2, we use this algorithm to design a new approximation algorithm for k-ITSPN.

5.1 Weighted k-tree cover problem (Wk-TCP) and its constant factor approximation

The definition of the Wk-TCP is as follow.

Definition 8 (Wk-TCP)**.** *Consider an edge-weighted graph $G = (V, E, w_E)$, where V and E are the node set and edge set of G, respectively and $w_E : V \to \mathbb{Z}^+$ is the weight function over the edges. Then, the Wk-TCP is to find a set of $k \geq 2$ trees $X = \{X_1, \cdots, X_k\}$ such that*

(a) for each node $v_i \in V$, $\exists X_j \in X$ such that $v_i \in V(X_j)$, and

(b) the cost of X, which is

$$Cost(X) = \max_{1 \leq j \leq k} [\frac{\sum_{e \in E(X_j)} w_E(e)}{w_j}]$$

is minimized, where

$W = \{w_1, \cdots, w_k\}$ be the set of inverted normalized weights (the heavier, the less significant) of the resulting trees such that

$$w_1 \leq w_2 \leq \cdots \leq w_k = 1. \quad (1)$$

Algorithm 1 Wk-TCP-SUBA (G, W, k, b, α)

1: Compute $G' = (V', E', w_E)$ such that $V' \leftarrow V(G)$, $E' \leftarrow \{e | e \in E(G)$ and $w_E(e) \leq b\}$, and w_E is the edge-weight function from $G = (V, E, w_E)$.
2: If G' is disconnected, then return $(fail, null)$. Otherwise, set $\hat{E}'$ to be the set of edges in an MST of G'.
3: Set $m \leftarrow 1$.
4: **while** $Weight(\hat{E}') = \sum_{e \in \hat{E}'} w_E(e) \geq \alpha b w_m$ and $m \leq k-1$ **do**
5: Edge-decompose a tree X_m from $MST(G', \hat{E}')$, the induced subgraph of the MST of G' by $\hat{E}'$ such that

$$Weight(X_m) = \sum_{e \in E(X_m)} w_E(e) \in [\alpha b w_m, 2\alpha b w_m].$$

6: $\hat{E}' \leftarrow \hat{E}' \setminus E(X_m)$.
7: $m \leftarrow m + 1$.
8: **end while**
9: **if** $Weight(\hat{E}') > 2\alpha b w_m$ **then**
10: return $(fail, null)$.
11: **end if**
12: $X_m \leftarrow \hat{E}'$. /* $Weight(X_l) \leq 2\alpha b w_m$ */
13: Return $(succ, X = \{X_1, X_2, \cdots, X_m\})$. /* $m \leq k$ */

Remark 1. *The proposed algorithm only considers the Wk-TCP instances with non-extreme inverted normalized weight difference, which can be defined as $w_k/w_1 \leq \alpha$, where*

$$\alpha = \frac{k - 1 + w_1 + \cdots + w_k}{1 + w_1 + \cdots + w_k}.$$

Based on this assumption, Eq. (1) can be rewritten as

$$\frac{1}{\alpha} \leq w_1 \leq w_2 \leq \cdots \leq w_k = 1. \quad (2)$$

To better understand the notion of "extreme", consider the case in which all the weights are same (or equivalently the speed of all drones are same in our context), i.e. $w_1 = w_2 = \cdots = w_k = 1$, and $\alpha = \frac{k-1+k}{1+k} \simeq 2$. Apparently, $w_k/w_1 \leq \alpha$ is always true. On the other hand, if some of them are really fast and the other some are very slow, $w_k/w_1 = 1/\epsilon \geq \alpha$ is highly possible, where ϵ is a really small positive constant.

We first introduce Wk-TCP-SUBA, which is a polynomial time sub-routine, for Wk-TCP. Once successful, the algorithm outputs m trees $X = \{X_1, X_2, \cdots, X_m\}$ for some $m \leq k$ from (G, W, k) with two input positive constants b and α such that

$$Weight(X_i) = \sum_{e \in E(X_i)} w_E(e) \leq 2\alpha b w_i$$

for each $X_i \in X$. Otherwise, it returns "fail". Algorithm 1 is the formal description of Wk-TCP-SUBA. The details of this algorithm is as follow.

(a) Given an edge-weighted graph $G = (V, E, w_E)$, a positive integer k, the inverted normalized weights of the resulting trees $W = \{w_1, \cdots, w_k\}$, and two positive constants α and b, this algorithm induces a graph $G' = (V', E', w_E)$ from G such that $V' \leftarrow V(G)$ and $E' \leftarrow \{e | e \in E(G)$ and $w_E(e) \leq b\}$. That is E' is the set of edges in E whose weights are no greater than b. If the choice of b makes G' disconnected, or equivalently makes an MST of G' disconnected, the algorithm stops and returns fail.

(b) Through Lines 4-8, the algorithm tries to decompose the edges of an MST of G' into m trees $X = \{X_1, \cdots, X_m\}$ for some $m \leq k$. The detail of this decomposition process is as below. Initially, we set $\hat{E}'$ to be the set of edges in an MST of G'. We start by selecting a random root node r in $MST(G', \hat{E}')$, the induced subgraph of the MST of G' by $\hat{E}'$. Then, each child of r in $MST(G', \hat{E}')$ is a root node of a subtree X_i. To make our discussion simpler, for each resulting subtree X_i, let us categorize the subtree be
   (i) "light" if $Weight(X_i) \in [0, \alpha b w_i)$,
   (ii) "medium" if $Weight(X_i) \in [\alpha b w_i, 2\alpha b w_i]$, and
   (iii) "heavy" if $Weight(X_i) \in (2\alpha b w_i, \infty)$.

   By Line 4 of the algorithm, if $Weight(\hat{E}') < \alpha b w_i$, $MST(G', \hat{E}')$ is a single light tree which is an trivial output of Algorithm 1. If r is the root of a medium subtree in $MST(G', \hat{E}')$, then we are done. Otherwise, r is the root of a heavy subtree. We first look for a child node r' of r which is the root of a heavy subtree in $MST(G', \hat{E}')$. If such r' exists, we consider it as a new root and repeat this procedure (Lines 4-8). Otherwise, the root collectively merges the light subtrees whose roots are the children of it until the merged tree becomes a medium subtree. It is important to notice that when a medium subtree is extracted, it extracts edges from the edge set $\hat{E}'$. This means that r can be the root of the extracted subtree, and at the same time, it still can be used for further processing of $MST(G', \hat{E}')$ even after the edges in the extracted subtree is removed from $\hat{E}'$. After $\hat{E}'$ is updated, we repeat the tree extraction processing to obtain another trees (up to $k-1$ more trees).

(c) If the algorithm is successful (Line 13), it will return m trees such that
   (i) $m \leq k$,
   (ii) no two trees share the same edge in $MST(G', \hat{E}')$, but may share some nodes in $V(G')$, and
   (iii) for each $X_i \in X$, $Weight(X_i) \leq 2\alpha b w_i$.

Note that depending on the value of b, Wk-TCP-SUBA may be successful in computing $X = \{X_1, \cdots, X_m\}$ such that $Weight(X_i) \leq 2\alpha b w_i$ for each $1 \leq i \leq m \leq k$, or returns fail. Next, we prove the correctness of the algorithm. In the following, we explain how to find choose a b and use Wk-TCP-SUBA to obtain a constant factor approximation algorithm for Wk-TCP (Lemma 4).

5.1.1 Correctness of Wk-TCP-SUBA

First, we prove that given a heavy tree, it is possible to extract a medium tree as long as $Weight(\hat{E}') \geq \alpha b w_i$.

Lemma 1. *It is always possible to split a medium tree X_i from $MST(G', \hat{E}')$ in Lines 4-8 of Algorithm 1.*

Proof. By the condition, $Weight(\hat{E}') \geq \alpha b w_i$, $MST(G', \hat{E}')$ cannot be a light tree. If it is a medium tree, then the algorithm will consider the whole $MST(G', \hat{E}')$ as a medium subtree and this lemma is true. Therefore, we need to prove this lemma for the case that $MST(G', \hat{E}')$ is a heavy tree. Note that as long as $\hat{E}'$ is not empty, $MST(G', \hat{E}')$ has at least one light subtree. That is, for any edge $e \in E(G)$, we have $w_E(e) \leq b = \alpha \cdot b \cdot \frac{1}{\alpha} \leq \alpha b w_i$ since $\frac{1}{\alpha} \leq w_i$ by our assumption Eq. (2).

Therefore, to complete the proof of this lemma, we need to show that when $MST(G', \hat{E}')$ is a heavy tree, there exists at least one subtree X_j such that $Weight(X_i) \in [\alpha b w_i, 2\alpha b w_i]$ as long as $Weight(\hat{E}') \geq \alpha B w_i$. In fact, such a medium tree X_i can always be obtained as follows:

First, randomly choose a leaf node x of $MST(G', \hat{E}')$. Let $X^*(x)$ be the subtree that is rooted at x. If $X^*(x)$ is light, we identify its parent node y in $MST(G', \hat{E}')$ and set $x \leftarrow y$, and check if $X^*(x)$ is still light. This is repeated until $X^*(x)$ is not light. If $X^*(x)$ is medium, we decompose it. Otherwise, $X^*(x)$ is heavy, and

(i) if x has a sibling y such that $X^*(y)$ is heavy, then we set $x \leftarrow y$ and continue,
(ii) if x has a sibling y such that $X^*(y)$ is medium, we decompose $X^*(y)$ (Line 5 or Algorithm 1), and
(iii) if all of the subtrees $X^*(y_1), X^*(y_2), \cdots, X^*(y_l)$ where $y_1, y_2, \cdots, y_l$ the list of sibling of x are light, then, there exists i, $1 \leq i \leq l$ such that the tree rooted at the parent node of x and connecting $X^*(y_1), X^*(y_2), \cdots X^*(y_i)$ is a medium tree. We decompose this tree from $G[R]$.

As a result, this lemma is true. □

By this lemma, if $MST(G', \hat{E}')$ is initially a heavy tree, it will be split into at most $k-1$ medium trees. For the last residual tree, we have the following lemma.

Lemma 2. *Suppose b^* is the cost of an optimal solution of this Wk-TCP instance, the cost of the heaviest tree in the optimal solution consists of m trees for some $m \leq k$. Suppose G' is connected, and we have $\hat{E}'$ after Line 8 of Algorithm 1. If $b^* \leq b$, then $Weight(\hat{E}') \leq 2\alpha b w_k$, where $w_k = 1$.*

Proof. Suppose Algorithm 1 is executed and $m \leq k$ trees are generated. If $m < k$, $\hat{E}'$ in Line 8 is empty and this lemma is true. Therefore, we only consider $m = k$.

For simplicity, let $X_1^*, ..., X_k^*$ denote the trees that cover G, which is the optimal solution of our problem. We assume G is connected after all the edges of weights greater than b are removed. So, by adding at most $k-1$ edges, we can connect these k trees to obtain a tree that

spans over G. Since the cost of each such edge is at most b, we obtain:

$$\sum_{i=1}^{k} w_i Weight(X_i^*) + (k-1)b \geq Weight(MST(G)), \quad (3)$$

where $Weight(MST(G))$ is the total edge weight of a MST (minimum spanning tree) of G. Since $Weight(X_i^*) \leq w_i b^* \leq b$, we obtain that:

$$b(k-1) + b\sum_{i=1}^{k} w_i \geq Weight(MST(G)).$$

Also, since the algorithm is a decomposition of the minimum spanning tree of G, we have:

$$\begin{aligned} &Weight(MST(G)) = \textstyle\sum_{i=1}^{k-1} w_i Weight(X_i) + Weight(X_k) \\ &\geq \alpha b(\textstyle\sum_{i=1}^{k} w_i + 1) + (Weight(\hat{E}') - 2\alpha b). \end{aligned} \quad (4)$$

The cost of the resulting tree cover by our algorithm is at most $2\alpha b$ as long as $Weight(T_k) = Weight(R) \leq 2w_k \alpha b = 2\alpha b$, since $w_k = 1$. This is because

$$\begin{aligned} &Weight(X_k) = \max_{1\leq i\leq k} \frac{Weight(X_i)}{w_i} \\ &= \max(\max_{1\leq i\leq k-1} \frac{Weight(X_i)}{w_i}, \frac{Weight(X_k)}{w_k}) \\ &\leq \max(2\alpha b, Weight(\hat{E}')) \end{aligned} \quad (5)$$

From Eq. (3), Eq. (4), Eq. (5), and the definition of α, we obtain that:

$$Weight(\hat{E}') - 2\alpha b \leq \sum w_i \cdot b + (k-1)b - \alpha b(\sum w_i + 1) = 0.$$

As a result, the cost is at most $2\alpha b$ and this lemma is proved. □

Lemma 3. *If Algorithm 1 returns fail, then either G' is disconnected or $b^* > b$.*

Proof. This is the inverse proposition of Lemma 2, and thus can be easily proved by contradiction. □

Theorem 2. *Wk-TCP-SUBA returns a feasible solution of Wk-TCP only if for a selected b, $b^* \leq b$ and G', the subgraph of G, which is obtained after removing all edges whose weight is greater than b is removed, is still connected.*

Proof. This corollary naturally follows from Lemma 1, Lemma 2 and Lemma 3. □

5.1.2 Performance Analysis of Our Strategy

Now, we show that Wk-TCP-SUBA can be used to obtain an approximation algorithm for Wk-TCP.

Lemma 4. *When successful, the cost of a weight k-tree cover of G by Algorithm 1 is bounded by $2\alpha b$.*

Proof. By construction, the weight of each tree in T returned by the algorithm is bounded by $\alpha b \leq Weight(T_i) \leq 2\alpha b$. It follows that

$$Weight(X) = \max_{X_i \in X} \frac{Weight(X_i)}{w_i} \leq \max\{\frac{1}{w_i} \cdot 2\alpha b w_i\} \leq 2\alpha b.$$

As a result, this lemma is true. □

Theorem 3. *By applying binary search for b^*, a 2α-approximation algorithm can be obtained.*

Proof. By Corollary 2, Wk-TCP-SUBA returns a feasible solution of Wk-TCP only if $b^* \leq b$ and G' is connected. Since $Weight(MST(G)) > 0$, when b is sufficiently small ($i.e., b < \frac{Weight(MST(G))}{4\alpha(w_1+\cdots+w_k)}$), W$k$-TCP-SUBA will return fail. Meanwhile, when b is sufficiently large ($i.e., b > Weight(MST(G))$), Wk-TCP-SUBA will return success. In fact, we can perform a binary search for b as follows. Suppose the result of the binary search is b_0. We now prove that $b_0 \in (0, b^* + \epsilon)$. Relating to the process of binary search, suppose in the n_{th} step, the range of b_0 is contracted to $S_n = [b_n, b'_n]$. Since the algorithm always fails with b_n, by Lemma 3 we have $b_n < b^*$. S_n is a nested interval sequence and converges to point b_0. Therefore:

$$b_0 = \lim_{n\to\infty} b_n \leq \lim_{n\to\infty} b^* = b^*.$$

So the approximation ratio of Algorithm 1 is no bigger than 2α, i.e.

$$\leq \frac{2\alpha b}{b^*} \leq 2\alpha$$

□

As stated in the proof of Theorem 3, we can set the initial left and right bound of our binary search as

$$[\frac{Weight(MST(G))}{4\alpha(w_1+\cdots+w_k)}, Weight(MST(G))]$$

and derive a $(2\alpha+\epsilon)$-approximation for the weighted k-tree cover that runs in polynomial time.

Theorem 4. *For every $\epsilon > 0$, there is a $(2\alpha+\epsilon)$-approximation for the weighted k-tree cover that runs in time polynomial in log of the size of G and $\log(\frac{1}{\epsilon})$.*

Proof. We start by proving that the left and right bound of our binary search are valid, which is:

$$b \in [\frac{Weight(MST(G))}{4\alpha(w_1+\cdots+w_k)}, Weight(MST(G))].$$

When setting $b = \frac{Weight(MST(G))}{4\alpha(w_1+\cdots+w_k)}$, the k trees we derived will not be able to cover the whole minimum spanning tree of G, that is:

$$\begin{aligned} &Weight(MST(G')) \geq Weight(MST(G)) \\ &= 4\alpha b(w_1+\cdots+w_k) > 2\alpha b(w_1+\cdots+w_k), \end{aligned}$$

where G' is the subgraph of G without the edges whose weight is greater than b. So we will have $Weight(R) > 2\alpha b$.

The algorithm will return fail for this smaller b. When setting $b = Weight(MST(G))$, $MST(G)$ keeps unchanged after removing edges that weights greater than b. This is because no edges in $MST(G)$ is removed. The k_{th} tree will always be able to cover the remaining graph because:

$$2\alpha b w_k > Weight(MST(G)) = Weight(MST(G')) \geq w(R).$$

So the algorithm will return SUCCESS for this greater b.

The difference between the left and right bound is smaller than $Weight(MST(G))$. By applying a binary search, a result that ranges in $[0, b^* + \epsilon]$ can be obtained within $\log_2(\frac{Weight(MST(G))}{\epsilon})$ iterations. So that binary search within this range is polynomial in log of the size of G and $\log(\frac{1}{\epsilon})$. The approximation ratio for it is $2\alpha(b^* + \epsilon)/b^* = 2\alpha + \epsilon$. □

Remark 2. *By the design of Wk-TCP-SUBA, this algorithm fails if G′ in Line 2 is disconnected. During the binary search process, the algorithm will always find a feasible and smallest b for Wk-TCP-SUBA. But the approximation ratio can be proved only when the graph is connected after removing all the edges greater than b*. This property of the graph holds true for most of the times.*

5.2 A New Constant Factor Approximation for k-ITSPN

In this section, we propose a constant factor approximation algorithm for k-ITSPN. This algorithm consists of the following steps.

(a) Apply two coloring among the neighborhood areas of the nodes in V and obtain a subset I of nodes V' such that the neighborhoods of the nodes in I is pairwise-disjoint.
(b) Apply the 2α-approximation algorithm for the weighted k-tree cover problem in Section 5.1 to I, and obtain k trees $X = \{X_1, X_2, \cdots, X_k\}$.
(c) For each tree $X_i \in X$, apply Step 3 of GMSTNA in Section 4.1 [3] and convert each X_i into a neighborhood tour X_i^N. In this way, the neighborhood area of each node in $V \setminus I$ is visited by some tour X_j^N for some j.

The performance ratio of this algorithm follows from Section 4.4 [3] by replacing the $(4 + \epsilon)$-approximation algorithm for the k-rooted tree cover algorithm in [29] with the 2α approximation algorithm for the weighted k-tree cover problem in Section 5.1. By Corollary 4.1 in [3], the approximation ratio of our strategy for k-ITSPN is

$$1.5 \cdot 2\pi \cdot 2\alpha \cdot (1 + 20/\pi).$$

It is easy to see this is a polynomial time algorithm since our algorithm for the weighted k-tree cover problem in Section 5.1 is strongly polynomial.

6 Simulation Results and Analysis

In this section, we evaluate the average performance of our constant factor approximation algorithm for Wk-TCP, which is the main technical contribution of this paper. Let us call this algorithm by Wk-TCPA. For this simulation, we implemented the codes using C++, and prepare a 30×30 virtual space and deploy $n = 30$ to 90 nodes by increasing the number by 10. The weight of each tree w_i is a random real number either from the interval $[0, 5]$ or $[0, 10]$. For each parameter setting, we create 100 graph instances, execute the simulation, and obtain an averaged result.

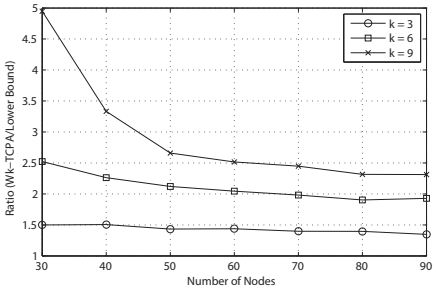


Fig. 2. Effect of k and n with $w_i \in [0, 5]$.

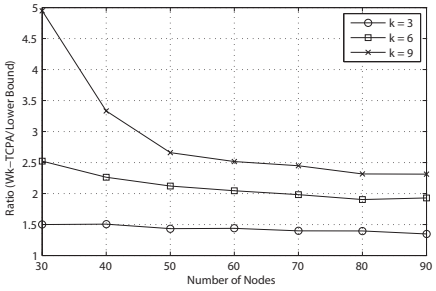


Fig. 3. Effect of k and n with $w_i \in [0, 10]$.

Given a Wk-TCPA instance, $\langle G = (V, E), W = \{w_1, \cdots, w_k\}, k\rangle$, we compute the lower bound of the cost of any feasible solution for Wk-TCP as follows. Without loss of generality, we assume $w_1 \leq \cdots \leq w_k$.

(a) Compute an MST T of G.
(b) Remove $k - 1$ heaviest edges from T, and obtain the residual graph T', which may consist of at most k components.
(c) Divide the weight of T', the total weight sum of the remaining edges in T', and again divide it with w_1.

In Fig. 2, we use the interval $[0, 5]$ to pick a uniformly distributed random weight for each tree and vary k to $3, 6, 9$ and n from 30 to 90. Then, we compare the performance of Wk-TCPA against the lower bound in terms of the ratio of the cost of the output of Wk-TCPA against the lower bound of the cost. This figure show (a) as we have a larger k, the average performance ratio becomes higher, and (b) as the number of the nodes increases in V, the performance gap is getting smaller and eventually stabilized. With $k = 3$, the cost of an output of Wk-TCPA is around 1.5 times larger and with $k = 9$, the cost is roughly 2.5 times larger. Given that we are using a lower bound instead of an optimal solution cost, Wk-TCPA performs reasonably well in a larger network. In Fig. 3, we use the interval $[0, 10]$ and do the same comparison. By comparing Fig. 2 and Fig. 3, we can learn that the maximum difference among the weight of the tree does not affect the average performance ratio.

In Fig. 4, we use the interval $[0, 5]$ to pick a uniformly distributed random weight for each tree and vary k to $3, 6, 9$ and n from 30 to 90. Then, we observe the average

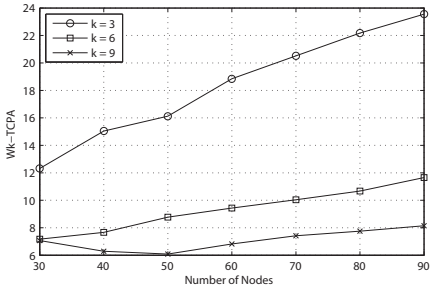


Fig. 4. Performance of Wk-TCPA with $w_i \in [0, 5]$.

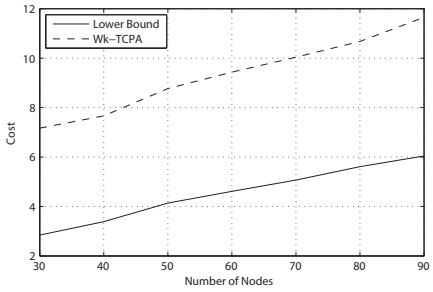


Fig. 5. Comparison between Wk-TCPA and the lower bound with $w_i \in [0, 5]$.

performance of Wk-TCPA. The result shows that (a) as the number of nodes grows, the cost of an output of Wk-TCPA grows, and (b) with a larger k, the cost decreases. Those two observations are very easy to understand. One very interesting phenomena is that with sufficient large k, until the size of network reaches to some point (around $n = 50$), the cost is getting smaller and after the point, it increases again. This shows that within a limited space of 30×30 size, when the nodes are evenly distributed, the cost overhead for to the trees assigned by our algorithm is also very even.

Finally, using Fig. 5, we compare the actual average cost of the outputs of Wk-TCPA against the lower bound. The lines representing the cost of each algorithms, respectively, increase with the same speed. This make their ratio become smaller as we saw from Fig. 2 and Fig. 3.

7 Conclusion and Future Work

In this paper, we have studied some important variations of the known problems introduced by Kim et al. in [2]. The two problems k-ATSPN and k-APCPN were relatively easier to approximate using existing result in [2]. On the other hand, despite the similarity, k-ITSPN is very challenging to design a constant factor approximation algorithm. Especially, our approximation algorithm for Wk-TCP, which is the core contribution of this paper to design the constant factor approximation algorithm for k-ITSPN, has an assumption like Eq. (1) and thus is not applicable under some extreme cases. As a future work, we plan to design more general approximation strategy for this problem. In addition, we will further investigate more practical versions of the problems.

Acknowledgement

This work was supported in part by US National Science Foundation (NSF) CREST No. HRD-1345219. This work is partly supported by National Natural Science Foundation of China under grant 11671400.

References

[1] L. Xue, D. Kim, Y. Zhu, D. Li, W. Wang, and A.O. Tokuta, "Multiple Heterogeneous Data Ferry Trajectory Planning in Wireless Sensor Networks," *Proceedings of the 33rd IEEE International Conference on Computer Communications (INFOCOM 2014)*, April 27, 2014 - May 2, 2014, Toronto, Canada.

[2] D. Kim, B.H. Abay, R.N. Uma, W. Wu, W. Wang, and A.O. Tokuta, "Minimizing Data Collection Latency in Wireless Sensor Network with Multiple Mobile Elements," *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM 2012)*, pp. 504-512, March 2012.

[3] D. Kim, R.N. Uma, B.H. Abay, W. Wu, W. Wang, and A.O. Tokuta, "Minimum Latency Multiple Data MULE Trajectory Planning in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing (TMC)*, vol. 13, no. 4, pp. 838-851, April 2014.

[4] M. Ma and Y. Yang, "SenCar: an Energy-efficeint Data Gathering Mechanism for Large-scale Multihop Sensor Networks," *IEEE Transactions on PArallel and Distributed Systems (TPDS)*, vol. 18, no. 10, pp. 1476-1488, 2007.

[5] M. Ma, Y. Yang and M. Zhao, "Tour Planning for Mobile Data-Gathering Mechanisms in Wireless Sensor Networks," *IEEE Transactions on Vehicular Technology*, vol. 62, issue 4, pp. 1472-1483, 2013.

[6] G. Xing , T. Wang , W. Jia, and M. Li, "Rendezvous Design Algorithms for Wireless Sensor Networks with a Mobile Base Station," *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'08)*, pp. 231-240, 2008.

[7] H. Jun, W. Zhao, M.H. Ammar, E.W. Zeura, and C. Lee, "Trading Latency for Energy in Densely Deployed Wirless Adhoc Networks using Message Ferrying," *Ad Hoc Networks*, vol. 5, pp. 441-461, May 2007.

[8] A. Jenkins, D. Henkel, and T. Brown, "Sensor Data Collection through Gateways in a Highly Mobile Mesh Network," *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, 2007.

[9] B. Yuan, M. Orlowska, and S. Sadiq, "On the Optimal Robot Routing Problem in Wireless Sensor Networks," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 19, no. 9, pp. 1252-1261, 2007.

[10] J.S.B. Mitchell, "A PTAS for TSP with Neighborhoods Among Fat Regions in the Plane," *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 11-18, 2007.

[11] Y. Yang, M. Lin, J. Xu, Y. Xie, "Minimum Spanning Tree with Neighborhoods," *Proc. of the 3rd International Conference on Algorithmic Aspects in Information and Management (AAIM '07)*, Portland, OR, USA, June 6-8, 2007.

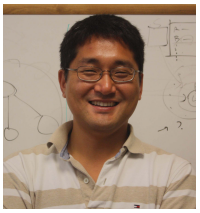
[12] D. Henkel and T.X. Brown, "Towards Autonomous Data Ferry Route Design through Reinforcement Learning," *Proc. of the 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM)*, pp. 1-6, 2008.

[13] O. Tekdas, J. Lim, A. Terzis, and V. Isler, "Using Mobile Robots to Harvest Data from Sensor Fields," *IEEE Wireless Communications Special Issue on Wireless Communications in Networked Robotics*, vol. 16, pp. 22-28, 2008.

[14] L. Boloni and D. Turgut, "Should I Send Now or Send Later?, a Decision-theoretic Approach to Transmission Scheduling in Sensor Networks with Mobile Sinks," *Wireless Communications and Mobile Computing*, vol. 8, no. 3, pp. 385-403, 2008.

[15] G. Anastasi, M. Conti, and M.D. Francesco, "Reliable and Energy-efficient Data Collection in Sparse Sensor Networks with Mobile Elements," *Journal of Performance Evaluation*, vol. 66, pp. 791-810, 2009.

[16] R. Sugihara and R.K. Gupta, "Optimizing Energy-latency Trade-off in Sensor Networks with Controlled Mobility," *Proceedings of the 28st IEEE International Conference on Computer Communications (INFOCOM 2009)*, pp. 1476-1488, 2009.
[17] B. Pearre and T.X. Brown, "Model-free trajectory optimization for wireless data ferries among multiple sources," *IEEE Globecom 2010 Workshop on Wireless Networking for Unmanned Aerial Vehicles (Wi-UAV 2010)*, 2010.
[18] B. Pearre and T.X. Brown, "Fast, Scalable, Model-free Trajectory Optimization for Wireless Data Ferries," *Proc. of IEEE International Conference on Computer Communications and Networks (ICCCN)*, pp. 370-377, 2011.
[19] B. Pearre and T.X. Brown, "Energy Conservation in Sensor Network Data Ferrying: a Reinforcement Metalearning Approach," *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2012)*, December 3-7, 2012.
[20] B. Pearre and T.X. Brown, "Model-Free Trajectory Optimisation for Unmanned Aircraft Serving as Data Ferries for Widespread Sensors," *Remote Sensing*, vol. 4, pp. 2971-3000, 2012.
[21] M. Todd, D. Mascarenas, E. Flynn, T. Rosing, B. Lee, D. Musiani, S. Dasgupta, S. Kpotufe, D. Hsu, R. Gupta, G. Park, T. Overly, M. Nothnagel, C. Farrar, "A Different Approach to Sensor Networking for SHM: Remote Powering and Interrogation with Unmanned Aerial Vehicles," in Proc. of *6th International Workshop on Structural Health Monitoring*, 2007.
[22] R. Sugihara and R.K. Gupta, "Speed Control and Scheduling of Data Mules in Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 7, issue 1, August 2010.
[23] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks," in Proc. of IEEE SNPA Workshop, 2003.
[24] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using Predictable Observer Mobility for Power Efficient Design of Sensor Networks," in Proc of *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, vol. 2634, pp. 552-568, 2003.
[25] D. Ciullo, G.D. Celik, and E. Modiano, "Minimizing Transmission Energy in Sensor Networks via Trajectory Control," *Proc. of the 8th International Symposium on Modeling and Optimization in Mobile, Ad HHoc and Wireless Networks (WiOpt)*, pp. 132-141, 2010.
[26] O. Tekdas, V. Isler, J. Lim, and A. Terzis, "Using Mobile Robots to Harvest Data from Sensor Fields," *IEEE Wireless Communications*, vol. 16, no. 1, pp. 22-28, Feb. 2009.
[27] A. Dumitrescu and J.S.B. Mitchell, "Approximation Algorithms for TSP with Neighborhoods in the Plane," in the Proc. of *the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '01)*, Washington DC, USA, January 7-9, 2001.
[28] G.N. Frederickson, M.S. Hecht, and C.E. Kim, "Approximation Algorithms for Some Routing Problems," *SIAM Journal of Computing*, vol. 7, pp. 178-193 (16 pages), 1978.
[29] G. Even, N. Garg, J. Konemann, R. Ravi, and A. Sinha, "Min-Max Tree Covers of Graphs," *Operations Research Letters*, vol. 32, issue 4, pp.309-315, 2004.
[30] A. Dumitrescua and J.S.B. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," *Journal of Algorithms*, vol. 48, issue 1, pp. 135-159, 2003.
[31] N. Christofides, "Worst-case Analysis of a New Heuristic for the Travelling Salesman Problem," *Report 388*, Graduate School of Industrial Administration, CMU, 1976.
[32] N. Guttmann-Beck and R. Hassin, "Approximation Algorithms for Min-Max Tree Partition," *Journal of Algorithms*, pp. 266-286, 1997.
[33] E.M. Arkin, R. Hassin, and A. Levin, "Approximations for Minmax Vehicle Routing Problems," *Journal of Algorithms*, vol. 59, issue 1, April 2006.
[34] C.H. Papadimitriou, "The Euclidean Traveling Salesman Problem is NP-Complete," *Theoretical Computer Science (TCS)*, vol. 4, no. 3, pp. 237-244, 1977.
[35] J.S.B. Mitchell, "A Constant-factor Approximation Algorithm for TSP with Pairwise-disjoint Connected Neighborhoods in the Plane," *Proc. of the Annual Symposium on Computational Geometry (SoCG)*, 2010.



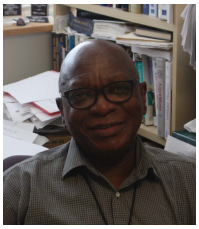
Donghyun Kim received the BS degree in electronic and computer engineering from the Hanyang University, Ansan, Korea (2003), and the MS degree in computer science and engineering from Hanyang University, Korea (2005). He received the PhD degree in computer science from the University of Texas at Dallas, Richardson, USA (2010). Currently, he is an assistant professor in the Department of Computer Science at Kennesaw State University, Marietta, USA. From 2010 to 2016, he was an assistant professor in the Department of Mathematics and Physics at North Carolina Central University, Durham, USA. His research interests include security and privacy, social computing, mobile computing, cyber physical systems, wireless and sensor networking, and algorithm design and analysis. He is an associate editor of Discrete Mathematics, Algorithms and Applications. He is a member of ACM and a senior member of IEEE.

Lirong Xue is a graduate student of the Operations Research and Financial Engineering Department at Princeton University. He received his Bachelor degree from Renmin University of China in Beijing in 2015. In his undergraduate study, he participated in research in graph theory, approximation algorithm, bioinformatics and machine learning.

Deying Li received the MS in mathematics from Huazhong Normal University, China (1988) and the PhD degree in computer science from City University of Hong Kong (2004). She is currently a professor in the Department of Computer Science from Renmin University of China. Her research includes wireless networks, mobile computing and algorithm design and analysis.

Yuqing Zhu is an assistant professor in Department of Computer Science, California State University Los Angeles. He obtained his PhD degree from University of Texas at Dallas, and his M.S. and B.S. degrees from Institute of Computing Technology, Chinese Academy of Science and Renmin University of China respectively. He is interested in Big Data related areas like MapReduce optimization, social networks and network management.

Wei Wang received the BS degree in applied mathematics from ZheJiang University, Hangzhou, China (1991). He received the MS degree in computational mathematics (1994) and PhD degree in mathematics from Xian Jiaotong University, Xian, China (2006). He is currently a professor at School of Mathematics and Statistics, Xian Jiaotong University. His research interests include algebraic graph theory and approximation algorithm design and analysis.



Alade O. Tokuta received the BS and MS degrees in electrical engineering from Duke University, Durham, the EE degree in electrical engineering from Columbia University, NY, and the PhD degree in electrical engineering and computer science from University of Florida, Gainesville. Currently, he is a professor in the Department of Mathematics and Physics at the North Carolina Central University, Durham, USA. His research interests include robotics; computer image synthesis/vision; networking, and algorithm design.