Multiple Heterogeneous Data Ferry Trajectory Planning in Wireless Sensor Networks

Lirong Xue[*], Donghyun Kim[†], Yuqing Zhu[‡], Deying Li[*][¶], Wei Wang[§], and Alade O. Tokuta[†]
[*] School of Information, Renmin University of China, Beijing, China. E-mail: {flandrex,deyingli}@ruc.edu.cn.
[†] Department of Mathematics and Physics, North Carolina Central University, Durham.
E-mail: {donghyun.kim, atokuta}@nccu.edu.
[‡] Department of Computer Science, University of Texas at Dallas, Richardson, USA.
E-mail: yuqing.zhu@utdallas.edu.
[§] Department of Mathematics, Xi'an Jiaotong University, Xi'an, China. E-mail: wang_weiw@163.com.
[¶] Corresponding Author.

***Abstract*—This paper investigates two new groups of trajectory optimization problems which stem from networked multi-robotic systems. In particular, we study how to efficiently collect data from stationary sensor nodes using multiple robotic vehicles such as data ferries under different circumstance. The first group includes two new problems which aim to find the tours and the paths, respectively, of k robot vehicles with different mobilization conditions to collect data from ground sensor nodes with minimum latency. The second group consists of one new problem whose goal is to determine the quality tours of k robot vehicles with different speeds, where each of which follows its corresponding tour to repeatedly collect data from stationary sensors. We prove the three problems are NP-hard and propose constant factor approximation strategies for them. Through a simulation, an analytical study is conducted to evaluate the average performance of our core contribution.**

***Index Terms*—Approximation algorithm, graph theory, wireless sensor network, mobile elements, traveling salesperson problem, trajectory control**

I. Introduction

Thanks to the recent advances in the field of robotics, various mobile robots with superior capabilities such as unmanned aerial vehicles (UAVs) and drones have been introduced. Unlike stand alone industrial robots which received huge attention in the past years, these newer generations of robots can deal with a wider range of applications by collaboratively executing missions, which are well beyond the ability of a single robot. The term "robotic vehicle" usually refers a robot which can move on the ground, in the air, undersea, or in space autonomously [21]. The vehicle is unmanned and utilizes onboard resources such as sensors, energy source, and computing device to navigate itself. Occasionally, it uses some form of integrated wireless communication onboard device to obtain high-level human oversight of its motion and task execution.

Roughly speaking, a networked multi-robot system is a wireless network of multiple robotic vehicles with various sensors. Despite of various challenges [5], [6], [7], the wireless sensor networks are considered as a promising solution for various surveillance and monitoring applications. Depending on the applications, a networked multi-robot system executes a mission independently or by interacting with surrounding infrastructure. For instance, a group of UAVs, which can be considered as fully controllable data ferries in the context of sensor networking, can collect environmental data from the stationary ground sensor nodes in an isolated mountain area for an extensive period to monitor a gradual climate change. It can form an instantaneous scout system to discover the wounded civilians in an area hard-hit by a critical disaster such as an earthquake or a tsunami.

The trajectory management of the mobile robots in such multi-robotic systems is known to be very significant since it is directly related to the performance such as latency, energy-efficiency etc. of the systems [3], [4], [8], [9], [12], [13], [14], [15], [16], [26], [17], [18], [19], [20], [22]. Recently, Kim et al. have investigated the problem of computing the trajectories of multiple data ferries to collect data from a group of stationary sensor nodes with minimum delay [1], [2]. In detail, given a group of n stationary sensor nodes and k data ferries, two different problems, namely the *k-traveling salesperson problem with neighborhood (k-TSPN)* and the *k-rooted path cover problem with neighborhood (k-PCPN)* are considered. The common goal of the two problems is to determine the trajectories of the k data ferries such that

(a) each trajectory originates from the location of the corresponding data ferry which will move around following the trajectory,
(b) each sensor node is visited by a data ferry on move when the distance between the sensor node and the ferry becomes no greater than the transmission range, and
(c) the length of the longest trajectory is minimized.

Despite the similarity, they are differ in the definition of trajectory. That is, k-TSPN assumes each mobile element can transmit data to the user only at its original (starting) location, and therefore pursues k-rooted tours, and k-PCPN assumes each mobile element is connected to the user at any location and thus seeks for k-rooted paths. Kim et al. have also introduced a constant factor approximation algorithm for each of k-TSPN and k-PCPN, respectively.

In this paper, we further investigate two multiple data ferry trajectory optimization problems in wireless sensor networks.

In particular, we consider the following two problems.

(a) k asynchronous TSPN (k-ATSPN) and k asynchronous PCPN (k-APCPN): Given a set of stationary sensor nodes and k homogenous (i.e. with the same physical capability) data ferries, which will depart the base station at possibly different times, how to collect data from all of the sensors using the data ferries with minimum delay?
(b) k inhomogeneous TSPN (k-ITSPN): Given a set of stationary sensor nodes and k (inhomogeneous) data ferries with different speed, which are the best tours of the data ferries such that when the ferries repeatedly circulate following the tours and routinely collect data from the sensors, the data refresh rate (the maximum inter-arrival time of a data ferry to collect data from a sensor node) is minimized?

The motivations of k-ATSPN and k-APCPN are similar to k-TSPN and k-PCPN, which aim to determine the trajectories of k data ferries for an one-time urgent data collection from sensor nodes. Therefore, the formulations of k-ATSPN and k-APCPN are similar to k-TSPN and k-PCPN in a sense that their common goal is to compute k-trajectories of the data ferries such that each sensor node will have a chance to communicate with a data ferry following a trajectory and the length of the longest trajectory is minimized. However, the assumption that all data ferries will be available from the beginning is removed since it is less realistic. For instance, in case of the urgent search and rescue operation in a disaster-hard-hit area, it is unlike that all of the available data ferries are immediately available.

Unlike k-ATSPN and k-APCPN, on the other hand, k-ITSPN aims to compute the trajectories of (inhomogeneous) data ferries with various speed for routine data collection applications with a focus on minimizing data refresh rate (or equivalently, worst case data delivery latency). One good example is the long-term environmental monitoring system, in which we sparsely deploy a number of stationary sensor nodes over a vast mountain area and employ various data ferries to follow pre-computed tours and routinely extract accumulated data from each sensor node over 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.

The summary of contributions of this paper is as follow.

(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 reduction technique to reduce the problems into a known NP-hard problem, which has a constant factor approximation algorithm. 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 called, the weighted k-tree cover problem (Wk-TCP), whose goal is 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 a constant factor approximation algorithm for Wk-TCP. Using this algorithm, we obtain a constant factor approximation algorithm for k-ITSPN.
(d) We perform a simulation and evaluate the core contribution of this paper, the constant factor approximation for Wk-TCP by comparing the average cost of the output of this algorithm and the lower bound. Our results show the algorithms work better as the size of network grows and k increases.

The rest of this paper is organized as follow. Section II presents some related work. We introduce some notations and definitions in Section III. Our main contributions, the constant factor approximations for k-ATSPN and k-APCPN are in Section IV, and the constant factor approximation for k-ITSPN is in Section V. Section VI presents the simulation results and corresponding discussions. Finally, we conclude this paper in Section VII.

II. Related Work

In the literature, the mobility-assisted data collection schemes are categorized into the following three classes [23]: random mobility, predictable mobility, and controlled mobility. The works in the first class such as Data Mobile Ubiquitous LAN Extensions (MULEs) [24] uses mobile nodes whose trajectories are not controllable and not predictable to opportunistically deliver data from sensor nodes. In [25], 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 [3]. 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 [3], [4], [8], [9], [12], [13], [14], [15], [16], [26], [17], [18], [1], [19], [20], [2].

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 [35]. 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], [26], [27]. 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 [28]. They also proposed $(1 + \epsilon)$-approximation algorithm for this TSPN in which the neighborhood areas of the nodes are disjoint and arbitrary fat-regions [31]. In [36], 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.

The k-traveling salesman problem (k-TSP) is to find k-tours starting from the same origination such that the length of the longest tour is minimized, and is NP-hard. The first constant factor approximation algorithm for k-TSP, namely k-SPLITOUR, is proposed by Frederickson et al [29]. However, this result is not directly applicable if the starting point of the mobile nodes are distinct due to the structure of the approximation algorithm, as a result, more effort should be made. In [1], [2], Kim et al. introduced the k-traveling saleman 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 (3) 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 since the assumption that all data ferries will be available from the beginning is removed.

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 [30], 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 [33], [34]. In [1], [2], 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.

III. Notations and Definitions

A. Notations

$V = \{v_1, \cdots, v_n\}$ is a set of n wireless sensor nodes. $R = \{r_1, \cdots, r_k\}$ is a given set of k roots (data ferris). For each node $v_i \in V$, $N(v_i)$ is the neighborhood area of v_i, which is a disk centered at v_i with normalized radius 1. $G = (V, E)$ 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. $Len(G)$ 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 the 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.

B. Definitions

The formal definition of k-TSPN and k-PCPN in [1], [2] are as follows.

Definition 1 (k-TSPN)**.** *Given a set V of n nodes 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 2 (k-PCPN)**.** *Given a set V of n nodes 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.

Now, 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 3 (k-ATSPN)**.** *Given a set V of n stationary wireless sensor nodes and a set R of k data ferries $\{r_1, \cdots, r_k\}$ 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 data ferries, the k* asynchronous TSPN (k-ATSPN) *is to find the tours of the k data ferries, $U = \{U_1, \cdots, U_k\}$ 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 4 (k-APCPN)**.** *Given a set V of n stationary wireless sensor nodes and a set R of k data ferries $\{r_1, \cdots, r_k\}$ 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 data ferries, the k* asynchronous (k-APCPN) *is to find the paths of the k data ferries, $P = \{P_1, \cdots, P_k\}$ 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 data ferry 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 [1], [2] is not directly applicable to solve the new problems. It is see 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 5 (k-ITSPN)**.** *Given a set V of n stationary wireless sensor nodes and a set R of k data ferries $\{r_1, \cdots, r_k\}$ with unique speed $\{s_1, \cdots, s_k\}$, respectively, the k* inhomogeneous TSPN (k-ITSPN) *is to find the tours of the k data ferries, $U = \{U_1, \cdots, U_k\}$ 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 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 data ferry r_i is divided by its speed s_i, which incurs additional challenge and therefore the result of [1], [2] 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.

IV. 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, $\langle V = \{v_1, \cdots, v_n\}, \{t_1, \cdots, t_k\}, R = \{r_1, \cdots, r_k\}, s \rangle$ for some positive integers n and k, and a positive constant s. 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 $V_2 \leftarrow \{w_1, \cdots, w_k\}$. 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 $Euc(v_i, v_j)$.
(c) For each w_i, v_j pair in $\widehat{V}$, add an edge (w_i, v_j) to $\widehat{E}$ and set the edge weight $c_E(v_i, v_j)$ to be $Euc(r_i, v_j) + s \cdot t_i$. This is because while in the original definition of the problem, the cost of a tour U_i is described as $\frac{Len(U_i)}{s} + t_i$ which is in terms of latency, we can replace this to $Len(U_i) + t_i \cdot s$, which is in terms of travel distance.

The constant factor approximation algorithm $\mathcal{A}$ for k-TSPN (or k-PCPN) in [1], [2]. $\mathcal{A}$ consists of the following steps.

- Step 1: Given a k-TSPN (or k-PCPN) problem instance $\langle V, R \rangle$, apply the $(4+\epsilon)$-approximation algorithm for k-RTCP in [30]. As a result, we have k rooted-trees $Q = \{Q_1, Q_2, \cdots, Q_k\}$ of V such that each tree includes a single distinct root and a subset of V such that $Q_i \bigcap Q_j = \emptyset$ for all i and j pairs.
- Step 2: For each $Q_i \in Q$, a neighborhood tree Q_i^N rooted at r_i and spanning over the neighborhood areas of $V(Q_i) \setminus \{w_i\}$ is computed as done in [1], [2].
- Step 3: Each neighborhood tree Q_i^N is converted to a tour for k-TSPN (or a path for k-PCPN) using Christofides's 1.5-approximation algorithm for TSP [32].

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

- Step 1: Given a k-ATSPN (or k-APCPN) problem instance $\langle V = \{v_1, \cdots, v_n\}, \{t_1, \cdots, t_k\}, R = \{r_1, \cdots, r_k\}, s \rangle$, 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 [30] over $\widehat{G}$ with $\{w_1, \cdots, w_k\} \subset V(\widehat{G})$ as the set of k roots. This approximation algorithm does not use any geometric, and works on any graph. As a result, we have k rooted-trees $Q = \{Q_1, Q_2, \cdots, Q_k\}$ of V such that each tree includes a single distinct root w_i and a subset of V such that $Q_i \bigcap Q_j = \emptyset$ for all i and j pairs.
- Step 2: For each $Q_i \in Q$, a neighborhood tree Q_i^N rooted at r_i and spanning over the neighborhood areas of $V(Q_i) \setminus \{r_i\}$ is computed.
- Step 3: Each neighborhood tree Q_i^N is converted to a tour for k-ATSPN (or a path for k-APCPN) using Christofides's 1.5-approximation algorithm for TSP [32].

Theorem 1. *There exist constant approximation algorithms for k-ATSPN and k-APCPN.*

Proof: In [2], 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. ■

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 $c_E(e) \leq b\}$, and $w_E$ is the edge-weight function from $G = (V, E, w_E)$.
2:  Suppose $R$ is the set of edges in an MST of $G'$. If $G[R]$, the induced subgraph of $G'$ by $R$, is disconnected, return $\langle fail, null \rangle$.
3:  Set $m \leftarrow 1$.
4:  while $Weight(R) = \sum_{e \in R} w_E(e) \geq \alpha b w_m$ and $m \leq k-1$ do
5:      Edge-decompose a tree $T_m$ from $G[R]$ such that
            $Weight(T_m) = \sum_{e \in E(T_m)} w_E(e) \in [\alpha b w_i, 2\alpha b w_i]$.
6:      $R \leftarrow R \setminus E(T_m)$.
7:      $m \leftarrow m + 1$.
8:  end while
9:  if $Weight(R) > 2\alpha b w_m$ then
10:     return $\langle fail, null \rangle$.
11: end if
12: $T_m \leftarrow R$.                          /* $Weight(T_l) \leq 2\alpha b w_m$ */
13: Return $\langle succ, T = \{T_1, T_2, \cdots, T_m\} \rangle$.     /* $m \leq k$ */
```

V. A New Constant Factor Approximation for k-ITSPN

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

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

This sections introduces a new optimization problem, namely the weighted k-tree cover problem (Wk-TCP) and its constant factor approximation algorithm. The definition of the Wk-TCP is as follow.

Definition 6 (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 $w_E : V \rightarrow \mathbb{Z}^+$ is the weight function over the edges. Then, the Wk-TCP is to find a set of $k \geq 2$ trees $T = \{T_1, \cdots, T_k\}$ such that*

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

(b) the cost of T, which is

$$Cost(T) = \max_{1 \leq j \leq k} [\frac{\sum_{e \in E(T_j)} c_E(e)}{w_j}]$$

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

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

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

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

We first introduce Wk-TCPA-SUB, which is a polynomial time algorithm, for Wk-TCP. Once successful, the algorithm outputs m trees $T = \{T_1, T_2, \cdots, T_m\}$ for some $m \leq k$ from $\langle G, W, k \rangle$ with two input positive constants b and α such that

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

for all $1 \leq i \leq m$. Otherwise, it returns 'fail". Algorithm 1 is the formal description of Wk-TCPA-SUB. The details of this algorithm is as follow.

(a) Given an edge-weighted graph $G = (V, E, c_E)$, 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 $T = \{T_1, \cdots, T_m\}$ for some $m \leq k$. The detail of this decomposition process is as below. Remind that R is the set of edges in an MST of G'. This process starts from selecting a random node r in $G[R]$, the induced subgraph of G' by R. To make our discussion simpler, for each resulting subtree T_i, let us call the tree be

    (i) "light" if $Weight(T_i) \in [0, \alpha b w_i)$,
    (ii) medium if $Weight(T_i) \in [\alpha b w_i, 2\alpha b w_i]$, and
    (iii) heavy if $Weight(T_i) \in (2\alpha b w_i, \infty)$.

    Edge-decomposition of Tree. First, since $Weight(R) \geq \alpha b w_i$, $G[R]$ cannot be a light tree. If r is the root of a medium subtree in $G[R]$, 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 $G[R]$. If such r' exists, we consider it as a new root and repeat this procedure. 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 R. This means that the root can be the root of the extracted subtree, and the root is still in the residual $G[R]$ after the edges in the extracted subtree is removed from R. Now, we prove that from a heave tree, it is possible to extract a medium tree as long as $Weight(R) \geq \alpha B w_i$.

Lemma 1. *It is always possible to split a medium tree T_i from $G[R]$ in Lines 4-8 of Algorithm 1.*

Proof: By the condition, $Weight(R) \geq \alpha b w_i$, $G[R]$ cannot be a light tree. If it is a medium tree, then the algorithm will consider $G[R]$ as a medium subtree and this lemma is true. Therefore, we need to prove this lemma for the case that $G[R]$ is a heavy tree. Note that as long as R is not empty, $G[R]$ 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 $G[R]$ is a heavy tree, there exists at least one subtree T_j such that $Weight(T_i) \in [\alpha b w_i, 2\alpha b w_i]$ as long as $Weight(R) \geq \alpha B w_i$. In fact, such a medium tree T_i can always be obtained as follows:
First, randomly choose a leaf node x of $G[R]$. Let $T^*(x)$ be the subtree that is rooted at x. If $T^*(x)$ is light, we identify its parent node y in $G[R]$ and set $x \leftarrow y$, and check if $T^*(x)$ is still light. This is repeated until $T^*(x)$ is not light. If $T^*(x)$ is medium, we decompose it. Otherwise, $T^*(x)$ is heavy, and

(i) if x has a sibling y such that $T^*(y)$ is heavy, then we set $x \leftarrow y$ and continue,
(ii) if x has a sibling y such that $T^*(y)$ is medium, we decompose $T^*(y)$, and
(iii) if all of the subtrees $T^*(y_1), T^*(y_2), \cdots, T^*(y_l)$ 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 $T^*(y_1), T^*(y_2), \cdots T^*(y_i)$ is a medium tree. We decompose this tree from $G[R]$.

As a result, this lemma is true. ■

(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 $E(G')$, but may share some nodes in $V(G')$, and
   (iii) for each $T_i \in T$, $Weight(T_i) \leq 2\alpha b w_i$.

Note that depending on the value of b, Wk-TCP-SUBA may be successful in computing $T = \{T_1, \cdots, T_m\}$ such that $Weight(T_i) \leq 2\alpha b w_i$ for each $1 \leq i \leq m \leq k$, or returns fail. Next, we provide three lemmas: Lemma 2 and Lemma 3 together will be used to prove the correctness of the algorithm (Corollary 1). 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).

1) Correctness of Wk-TCP-SUBA:

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 solutions consists of m trees for some $m \leq k$. Suppose G' is connected, and we have R after Line 8 of Algorithm 1. If $b^* \leq b$, then $Weight(R) \leq 2\alpha b w_k$, where $w_k = 1$.*

Proof: Consider $m \leq k$, the number of trees generated by Algorithm 1 once successful. If $m < k$, R in Line 8 is empty and this lemma is true. Therefore, we only consider $m = k$.

For simplicity, let $T_1^*, ..., T_k^*$ 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:

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

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

$$\begin{aligned} &Weight(MST(G)) = \textstyle\sum_{i=1}^{k-1} w_i Weight(T_i) + Weight(T_k) \\ &\geq \alpha b(\textstyle\sum_{i=1}^{k} w_i + 1) + (Weight(R) - 2\alpha b). \end{aligned} \quad (4)$$

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

$$\begin{aligned} &Weight(T_k) = \max_{1 \leq i \leq k} \tfrac{Weight(T_i)}{w_i} \\ &= \max(\max_{1 \leq i \leq k-1} \tfrac{Weight(T_i)}{w_i}, \tfrac{Weight(T_k)}{w_k}) \\ &\leq \max(2\alpha b, Weight(R)) \end{aligned} \quad (5)$$

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

$$Weight(R) - 2\alpha b \leq \sum w_i \cdot b + (k-1)b - \alpha b(\sum w_i + 1) = 0.$$

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. ■

Corollary 1. *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 every edge whose weight is greater than b is removed, is still connected.*

Proof: This corollary naturally follows from Lemma 2 and Lemma 3. ■

2) Performance Analysis of Our Strategy: Now, we show that Wk-TCP-SUBA can be used to obtain a constant factor approximation algorithm.

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

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

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

As a result, this lemma is true. ∎

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

Proof: By Corollary 1, 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. By Theorem 3, in fact we can perform a binary search for b. 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 2, 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 3. *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:

$$Weight(MST(G')) \geq Weight(MST(G))$$
$$= 4\alpha b(w_1+\cdots+w_k) > 2\alpha b(w_1+\cdots+w_k),$$

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, an result that ranges in $[0, b^* + \epsilon]$ can be get 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.*

B. 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 V-A to I, and obtain k trees $T = \{T_1, T_2, \cdots, T_k\}$.
(c) For each tree $T_i \in T$, apply Step 3 of GMSTNA in Section 4.1 [2] and convert each T_i into a neighborhood tour T_i^N. In this way, the neighborhood area of each node in $V \setminus I$ is visited by some tour T_j^N for some j.

The performance ratio of this algorithm follows from Section 4.4 [2] by replacing the $(4+\epsilon)$-approximation algorithm for the k-rooted tree cover algorithm in [30] with the 2α approximation algorithm for the weighted k-tree cover problem in Section V-A. By Corollary 4.1 in [2], 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 V-A is strongly polynomial.

VI. 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 prepare

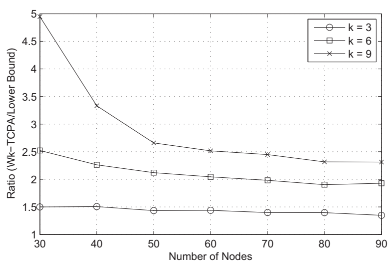

Fig. 1. Effect of k and n with $w_i \in [0, 5]$.

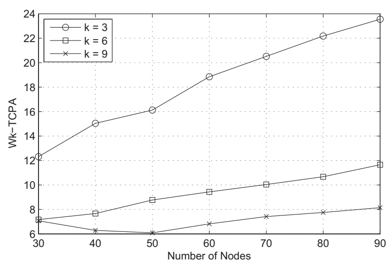

Fig. 3. Performance of Wk-TCPA with $w_i \in [0, 5]$.

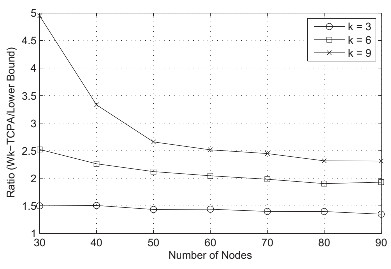

Fig. 2. Effect of k and n with $w_i \in [0, 10]$.

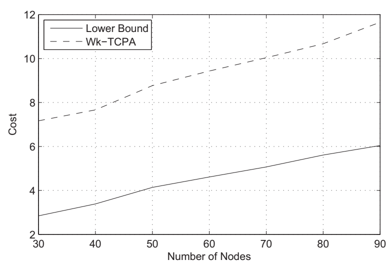

Fig. 4. Comparison between Wk-TCPA and the lower bound with $w_i \in [0, 5]$.

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.

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 an 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. 1, 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. 2, we use the interval $[0, 10]$ and do the same comparison. By comparing Fig. 1 and Fig. 2, we can learn that the maximum difference among the weight of the tree does not affect the average performance ratio.

In Fig. 3, 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 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. 4, 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. 1 and Fig. 2.

VII. Conclusion and Future Work

In this paper, we have studied some important variations of the known problems introduced by Kim et al. in [1]. The two problems k-ATSPN and k-APCPN were relatively easier to approximate using existing result in [1]. 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 research was supported in part by National Natural Science Foundation of China under grants 61070191 and 91124001, and Shenzhen Strategic Emerging Industries Program with Grant No.ZDSY20120613125016389. This work was also partly supported by US National Science Foundation (NSF) CREST No. HRD-1345219 and by US Army Research Office (ARO) No. W911NF-0810510.

References

[1] 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," *Proc. of the 31st IEEE International Conference on Computer Communications (INFOCOM 2012)*, pp. 504-512, March 2012.

[2] 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)*, 2013.

[3] 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.

[4] 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.

[5] S. Cheng, J. Li, and Z. Cai, "$O(\epsilon)$-Approximation to Physical World by Sensor Networks," *Proc. of the 32st IEEE International Conference on Computer Communications (INFOCOM 2013)*, pp. 3084-3092, 2013.

[6] Y. Li, C. Ai, Z. Cai, and R. Beyah, "Sensor Scheduling for p-percent Coverage in Wireless Sensor Networks," *Cluster Computing*, vol. 14, issue 1, pp. 27-40, 2011.

[7] S. Ji and Z. Cai, "Distributed Data Collection in Large-scale Asynchronous Wireless Sensor Networks under the Generalized Physical Interference Model," *IEEE/ACM Transactions on Networking (ToN)*, vol. 21, issue 4, pp. 1270-1283, 2013.

[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," *Proc. 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] T. Akimoto and N. Hagita, "Introduction to a Network Robot System," *International Symposium on Intelligent Signal Processing and Communications (ISPACS)*, 2006.

[22] 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.

[23] 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.

[24] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks," *Proc. of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[25] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using Predictable Observer Mobility for Power Efficient Design of Sensor Networks," *Proc. of ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, vol. 2634, pp. 552-568, 2003.

[26] 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.

[27] 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.

[28] A. Dumitrescu and J.S.B. Mitchell, "Approximation Algorithms for TSP with Neighborhoods in the Plane," *Proc. of the 12nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '01)*, Washington DC, USA, January 7-9, 2001.

[29] 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, 1978.

[30] 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.

[31] 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.

[32] N. Christofides, "Worst-case Analysis of a New Heuristic for the Travelling Salesman Problem," *Report 388*, Graduate School of Industrial Administration, CMU, 1976.

[33] N. Guttmann-Beck and R. Hassin, "Approximation Algorithms for Min-Max Tree Partition," *Journal of Algorithms*, pp. 266-286, 1997.

[34] E.M. Arkin, R. Hassin, and A. Levin, "Approximations for Min-max Vehicle Routing Problems," *Journal of Algorithms*, vol. 59, issue 1, April 2006.

[35] C.H. Papadimitriou, "The Euclidean Traveling Salesman Problem is NP-Complete," *Theoretical Computer Science (TCS)*, vol. 4, no. 3, pp. 237-244, 1977.

[36] 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.