Minimizing Data Collection Latency in Wireless Sensor Network with Multiple Mobile Elements

Donghyun Kim*, Baraki H. Abay*, R.N. Uma*, Weili Wu†, Wei Wang‡, and Alade O. Tokuta*
* Department of Mathematics and Computer Science, North Carolina Central University, Durham, NC 27707, USA
E-mail: donghyun.kim@nccu.edu, babay@eagles.nccu.edu, {ruma, atokuta}@nccu.edu
† Department of Computer Science, The University of Texas at Dallas, Richardson, Texas 75080, USA
E-mail: weiliwu@utdallas.edu
‡ Department of Mathematics, Xi'an Jiaotong University, Xi'an, P.R. of China, 710049
E-mail: wang_weiw@163.com

***Abstract*—This paper considers the problem of computing the optimal trajectories of multiple mobile elements (e.g. robots, vehicles, etc.) to minimize data collection latency in wireless sensor networks (WSNs). By relying on slightly different assumption, we define two interesting problems, the k-traveling salesperson problem with neighborhood (k-TSPN) and the k-rooted path cover problem with neighborhood (k-PCPN). Since both problems are NP-hard, we propose constant factor approximation algorithms for them. Our simulation results indicate our algorithms outperform their alternatives.**

***Index Terms*—Approximation algorithm, graph theory, wireless sensor network, mobile elements, traveling salesperson problem with neighborhood, k-rooted tree cover problem.**

I. Introduction

A *wireless sensor network (WSN)* consists of spatially distributed autonomous sensor nodes to cooperatively monitor physical events or environmental conditions. In most WSNs, each node has a limited energy source, and thus energy-efficiency is a crucial issue. Typically, the data produced by a node is sent to a designated data collector called a sink via either long-range single-hop or short-range multi-hop routing path. In wireless communication, the amount of energy consumed to send a signal increases super-linearly proportional to the travel distance of the signal. Therefore, the single-hop communication tends to be very energy-exhaustive in WSN. Meanwhile, the multi-hop communication makes the nodes near the sink deplete much faster than the other nodes, which results in shortening the lifetime of the WSN.

To address this issue, several mobile element (e.g. robot, vehicles, etc.) based data collection strategies are introduced to WSN [1], [2], [3], [4], [5], [6], [7], [8], in which no sensor node performs multi-hop routing and one or more mobile elements move around the sensor nodes deployed over the area of interest. Once a mobile element is within the communication range of a sensor node, the data accumulated in the sensor node is forwarded to the mobile element. In case that the mobile element has a long distance direct communication channel to a sink, this data can be sent to the sink immediately. Otherwise, the element must move to some location to be connected to the sink. Clearly, such mobile element based strategy alleviates the problem of the multi-hop communication scheme since no node needs to be involved in energy-exhaustive multi-hop message forwarding. Furthermore, such method enables us to extract data even from a disconnected WSN. However, while this approach can extend the lifetime of WSN greatly, it generally suffers from huge data latency due to the slow speed of the mobile elements. Therefore, it is crucial to make the trajectories of the elements shorter to minimize the latency.

In the target WSN of our research, data is delivered from each node to the sink only through the mobile elements. We assume there are k available mobile elements possibly located at different places. We consider following two probable cases: each mobile element is 1) only connected to the sink at their original position and 2) directly connected to the sink at any location. We also assume the speed of the mobile elements is fixed to some constant. In such models, the worst case data latency is heavily dependent on the length of the trajectories of the mobile elements. Such a model has a wide range of commercial and military applications [1], [9], [10]. Finally, we assume the neighborhoods of any two nodes may overlap with each other. Now, we list the contributions of this paper.

1) We propose two versatile problems, the *k-traveling salesperson problem with neighborhood (k-TSPN)* and the *k-rooted path cover problem with neighborhood (k-PCPN)*. The common goal of both problems is to find the trajectories of the k mobile elements to collect data from the n wireless sensor nodes on 2-D euclidean space such that the data collection latency (the length of the longest trajectory among k trajectories) is minimized. In detail,
   - k-TSPN assumes each mobile element can transmit data to the sink only at its original (starting) location, and therefore pursues k-rooted tours.
   - k-PCPN assumes each mobile element is connected to the sink at any location and thus seeks for k-rooted paths.
   - The neighborhood area of a node is defined as a circle centered at the node with a constant radius no smaller than one unit distance. Any two neighborhood areas may overlap with each other. In both k-TSPN and k-PCPN, a mobile element collects data from a node only if the neighborhood of the node is visited by

the element.

Both problems are NP-hard since their subclasses are NP-hard (see Theorem 3.1 and Theorem 3.3). We would like to emphasize that we assume each mobile element may start from different positions unlike [8], in which all mobile elements are assumed to start from the same position.

2) We introduce the *k-rooted tree cover problem with neighborhood (k-TCPN)*, whose objective is similar to k-PCPN and k-TSPN, but actually is to find k rooted trees instead of paths (k-PCPN) or tours (k-TSPN). k-TCPN is NP-hard since its subclass is NP-hard (see Theorem 3.2). By Remark 1 and Remark 2, an 1.5α approximation for k-PCPN and k-TSPN can be obtained from an α-approximation for k-TCPN by utilizing the 1.5-approximation of the traveling salesperson problem (TSP) by Christofides [17].
3) We propose a constant factor approximation algorithm for the *general minimum spanning tree with neighborhood (GMSTN)* problem, whose goal is, given a set of nodes, to find a *minimum spanning tree (MST)* of a set of the uniform circular overlapperable neighborhoods of the nodes. Based on this algorithm, we propose a constant factor approximation for k-TCPN. As a result, we obtain a constant factor approximation for k-PCPN and k-TSPN.
4) Given k roots and n nodes, the goal of the k-rooted tree cover problem[1] is to find k rooted trees such that the weight of the heaviest tree is minimized [13]. Naturally, any algorithm for this problem can be used to produce a feasible solution of the k-TCPN, and thus is useful to solve k-PCPN and k-TSPN. In simulation, we compare the outputs of our approximation strategies for k-PCPN and k-TSPN with the k-paths and k-tours computed by utilizing the $(4 + \epsilon)$-approximation algorithm for the k-rooted tree cover problem [13]. Our simulation results indicate our algorithms outperform these alternative methods.

We claim k-TSPN and k-PCPN versatile since they can be used to model various problems outside wireless network research community. For example, the problem of computing the trajectories of multiple ground vehicles with video cameras (note that it is sufficient to be in the neighborhood of each point of interest to take a picture) such that the time for the operator to collect the information of the area of interests can be minimized can be modeled using k-PCPN if the vehicles have long range radio communication devices, otherwise using k-TSPN since the vehicles must return back to their base station.

The rest of this paper is organized as follows. Section II introduces related work. In Section III, we introduce several important notations, the definitions of the problems, and their NP-hardness proofs. Our major contributions which include several constant factor approximation algorithms are introduced in Section IV. The simulation results are presented in Section V. Finally, Section VI concludes this paper.

[1]In [13], this problem is referred as the R-rooted tree cover problem. In this paper, we will use k instead of R to refer the number of available mobile elements (roots), while R is used to refer the set of k mobile elements.

II. Related Work

Largely, existing mobility-assisted data collection strategies are classified into following three categories [2]: random mobility, predictable mobility, and controlled mobility. Data Mobile Ubiquitous LAN Extensions (MULEs) is one of the seminary works in the random mobility class, in which several mobile nodes move around and collect data from the sensor nodes [3]. Eventually, they meet a sink and deliver the data to the sink. The authors in [4] used a queuing system to model the data collection process using mobile elements with predictable mobility. In [5], the SenCar, a fully controllable mobile sink, is introduced to collect the data from sensor nodes. The work also demonstrated the importance of finding a proper trajectory for the mobile element to accomplish its job successfully. Note that our work is in the third category.

In the *traveling salesperson problem with neighborhood (TSPN)*, a set of nodes each of which has a uniform circular neighborhood with radius 1 is given and a tour visiting the neighborhood areas of all nodes with minimum total length is sought. In [6], [7], the problem of finding the shortest tour of a single mobile element to collect the data from a set of uniform sensor nodes is modeled as TSPN and heuristic algorithms are proposed. Dumitrescu and Mitchell proposed a polynomial time $(\pi + 8)(1 + \epsilon)$-approximation algorithm for TSPN with overlappable circular neighborhoods, where ϵ is a small positive constant [11]. In [19], Mitchell also introduced a constant factor approximation algorithm for TSPN with pairwise disjoint arbitrary-shaped connected neighborhoods satisfying some condition. However, the existing algorithms for TSPN cannot be directly applied to our k-TSPN since we may need to find $k \geq 1$ tours.

The *k-traveling salesperson problem (k-TSP)* is to find k-tours originated from the same spot such that each node is visited by at least one of the tours and the length of longest tour is minimized. The k-SPLITOUR by Frederickson et al. is the first constant factor approximation for k-TSP [12]. In [8], k-SPLITOUR is used as a heuristic to find the tours of k-robots to collect the data from sensor nodes. Note that our k-TSPN is a generalization of their problem since in k-TSPN, the initial location of each robot may be different.

In the *k-rooted tree cover* problem, a set of nodes and k roots, which are possibly at different locations, are given, and k rooted trees are sought such that each node is in some tree and the total edge weight of the heaviest tree is minimized. The authors in [13] proposed a polynomial time $(4+\epsilon)$-approximation algorithm for this problem. The unrooted version of the *k-rooted tree cover* problem is studied in [14], [15]. However, it is conjectured that a solution for the unrooted version cannot be used for the rooted version [13]. Clearly, the $(4 + \epsilon)$-approximation algorithm can be used to obtain a feasible solution of our k-TCPN problem since a tree visiting the centers of a set of nodes also visits the neighborhoods of all of the nodes.

In the *minimum spanning tree with neighborhood (MSTN)* problem, a set of nodes each of which has a uniform circular neighborhood is given and an MST of the neighborhood areas of all nodes is sought. The authors of [16] assumed *no two neighborhood areas overlap with each other* and proposed a 7.4-approximation algorithm, two 3-approximation algorithms, and one polynomial time approximation scheme (PTAS) for MSTN. This algorithm cannot be used for k-TCPN since the neighborhoods of nodes may overlap with each other in k-TCPN and we are interested in k rooted trees of the neighborhoods for some $k > 1$.

III. Notations and Problem Definitions

In this paper, $V = \{v_1, v_2, \cdots, v_n\}$ is the set of n sensor nodes and $R = \{r_1, r_2, \cdots, r_k\}$ is the set of k mobile elements (or roots). For any $v_i \in V$, $N(v_i)$ is the disk (circular neighborhood area) which is centered at v_i and whose radius is $d \geq 1$. $G = (V, E)$ is a graph with a node set $V = V(G)$ and an edge (a straight line or a curve segment) set $E = E(G)$. $Len(G) = \sum_{(v_i,v_j) \in E(G)} L(v_i, v_j)$, where $L(v_i, v_j)$ is the length of the edge between $v_i, v_j \in V(G)$. $Eucdist(v_i, v_j)$ is the eculidean distance of v_i and v_j. In this paper, we say two disks (or neighborhoods) are "touching" with each other if their borders are adjacent with each other.

Definition 3.1 (k-TSPN): Given a $\langle V, R \rangle$ pair, the *k-traveling salesperson problem with neighborhood (k-TSPN)* is to find a set of k tours $U = \{U_1, U_2, \cdots, U_k\}$ such that 1) each U_i starts from and ends at r_i, 2) for each $v_j \in V$, $\exists u \in V(U_i)$ for some i such that u is in (or on the border of) $N(v_j)$, the neighborhood area of v_j, and 3) $Cost(U) = \max_{1 \leq i \leq k} Len(U_i)$ is minimized.

Definition 3.2 (k-PCPN): Given a $\langle V, R \rangle$ pair, the *k-rooted path cover problem with neighborhood (k-PCPN)* is to find a set of k paths $P = \{P_1, P_2, \cdots, P_k\}$ such that 1) each P_i is rooted at r_i, 2) for each $v_j \in V$, $\exists u \in V(P_i)$ for some i such that u is in (or on the border of) $N(v_j)$, and 3) $Cost(P) = \max_{1 \leq i \leq k} Len(P_i)$ is minimized.

Definition 3.3 (k-TCPN): Given a $\langle V, R \rangle$ pair, the *k-rooted tree cover problem with neighborhood (k-TCPN)* is to find a set of k trees $T = \{T_1, T_2, \cdots, T_k\}$ such that 1) each T_i is rooted at r_i, 2) for each $v_j \in V$, $\exists u \in V(T_i)$ for some i such that u is in (or on the border of) $N(v_j)$, and 3) $Cost(T) = \max_{1 \leq i \leq k} Len(T_i)$ is minimized.

Theorem 3.1: k-TSPN is NP-hard.

Proof: This is true since TSPN, which is a special case of k-TSPN in which $k = 1$, is NP-hard [11], [18]. ■

Theorem 3.2: k-TCPN is NP-hard.

Proof: Since a special case of k-TCPN is NP-hard in which $k = 1$, $d = 1$, and no two different $N(v_i)$ and $N(v_j)$ overlaps with each other [16], k-TCPN is NP-hard. ■

Theorem 3.3: k-PCPN is NP-hard.

Proof: Consider a special case of k-PCPN in which k is 1 and d, the radius of the neighborhood area of each node, goes to zero. Under the restrictions, k-PCPN is equivalent to the problem of finding a minimum total length path rooted at the only root r and visiting all nodes in V. Now, from the k-PCPN instance, we construct a directed graph G_s with $\{r\} \bigcup V$ as its vertex set by establishing directed edges

1) from r to each $u \in V$ with edge cost $Eucdist(r, u)$,
2) from each node $u \in V$ to r with edge cost 0, and
3) from $u \in V$ to $v \in V$ with edge cost $Eucdist(u, v)$ for every possible u, v pair.

Clearly, an exact algorithm for the special case of k-PCPN (i.e. $k = 1$ and $d = 0$) would solve the asymmetric TSP in G_s, a very well-known NP-hard problem, within polynomial time. As a result, the special case of k-PCPN has to be NP-hard, and so does k-PCPN in general. ■

IV. Main Results

Before presenting our main results, we would like to emphasize we assume that any pair of neighborhoods may overlap with each other. If we assume all neighborhoods are completely pairwise-disjoint, constant factor approximations of k-TCPN, k-TSPN, and k-PCPN can be easily obtained as follows.

1) Given a set V of n nodes and a set R of k roots, apply the $(4+\epsilon)$-approximation algorithm for the k-rooted tree cover problem in [13]. As a result, we have k-rooted trees $T_c = \{T_1, T_2, \cdots, T_k\}$ such that each tree T_i is rooted at $r_i \in R$ and each node in V is visited by some tree in T_c.
2) For each rooted tree $T_i \in T_c$, apply the $(1 + \epsilon)$-approximation algorithm for MSTN in [16] to r_i and $V(T_i)$. As a result, we obtain a tree T_i' spanning over r_i and the neighborhoods of all nodes in T_i.
3) Convert each T_i' into a tour or a path using the famous 1.5-approximation for TSP by Christofides [17]. (See Remark 1 and Remark 2 for details).

One can easily see the performance ratio of this strategy is roughly $4 \cdot 1 = 4$ for k-TCPN and $4 \cdot 1 \cdot 1.5 = 6$ for k-TSPN and k-PCPN. However, it is not practical to assume that the neighborhood areas are always pairwise-disjoint. This is because many applications of WSN consider scenarios, in which a number of sensor nodes are randomly deployed. If some pairs of neighborhoods overlaps with each other, this approximation strategy cannot be applied since the performance analysis in [13], [16] heavily relies on the pairwise-disjointness of neighborhoods.

In [16], the authors studied MSTN where no two neighborhoods overlap with each other. Let us call a variation of MSTN where the neighborhood areas of nodes in V may overlap and the radius of each circular neighborhood is a constant d as the *general MSTN (GMSTN)* problem. In Section IV-A, we introduce GMSTNA, a constant factor approximation of the GMSTN problem. In Section IV-B, we exploit GMSTNA to obtain a constant factor approximation algorithm of k-TCPN, and further optimize this algorithm in Section IV-C. Finally, the constant factor approximations of k-TSPN and k-PCPN are proposed in Section IV-D.

Algorithm 1 General Minimum Spanning Tree with Neighborhood Algorithm (GMSTNA) ($V = \{v_1, \cdots, v_n\}$)

```
1: Let $D = \{N(v_1), \cdots, N(v_n)\}$. We compute $I \subset D$, the set of maximal independent (pairwise-disjoint) neighborhoods of the nodes in $V$ as follows (also, see Fig 1(a)).
   (a) Color all disks in $D$ white.
   (b) Color one white disk $N(v_i)$ black and every white disk $N(v_j)$ which are overlapping (or touching) with $N(v_i)$ gray.
   (c) Repeat Step (c) until there is no white disk left.
   (d) Add all black disks to $I$.
2: Compute an MST $T_I^{center}$ of the centers of the disks in $I$ (see Fig 1(b)).
3: Let $T_{out}$ be an empty graph. For each edge $(v_i, v_j)$ in $T_I^{center}$, add the external segment (type 1 edge) of the edge connecting $N(v_i)$ and $N(v_j)$ as an edge of $T_{out}$. Also, for each $N(v) \in I$, add the round tour (type 2 edge) following $N(v)$ as an edge of $T_{out}$. For each point where a type 1 edge and a type 2 edge meet, we add this point as a vertex of $T_{out}$ (see Fig 1(c)).
4: Output $T_{out}$.
```

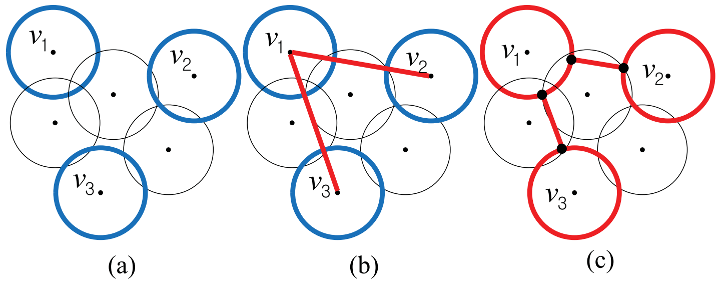

Fig. 1. These figures illustrate how GMSTNA works. In Figure (a), a maximal pairwise-disjoint disks, each of which is centered at one of $\{v_1, v_2, v_3\}$ are selected. In Figure (b), an MST T_I^{center} of $\{v_1, v_2, v_3\}$ is computed. In Figure (c), for each $N(v_i)$ in I, the internal segments of each edge connected to v_i is removed from T_I^{center} and the boundary of each disk selected in the first step is added to T_I^{center}.

A. GMSTNA: A Constant Factor Approximation of GMSTN

In this section, we propose a constant factor approximation algorithm for the GMSTN problem, namely *the general minimum spanning tree with neighborhood algorithm (GMSTNA)*. We would like to emphasize that GMSTN is for computing an MST of the neighborhoods of a set of nodes and there is no concept of root node. GMSTNA consists of following three major steps. In the first step, a set I of the pair-wise disjoint neighborhood areas of the nodes in V is identified by using a 2-coloring strategy. In the second step, an MST T_I^{center} of the center of the disks in I is computed. At this point, T_I^{center} is spanning over every $N(v_i) \in I$, but may not cover some $N(v_j) \notin I$. Due to this reason, in the final step, we modify T_I^{center} to T_{out}, which touches the neighborhood area of every node in V. Algorithm 1 is the description of GMSTNA. Now, we show this is a constant factor approximation of the GMSTN problem.

Lemma 4.1: $Len(T_{out}) \leq (d(2\pi - 1) + 1)Len(T_{mst-I}^{center}) + 2\pi d$, where T_{mst-I}^{center} is the MST of the centers of the disks in I in the first step of GMSTNA, T_{out} is an output of GMSTNA, and d is the radius of the circular neighborhood area (disk) of each node.

Proof: We first assume $|T_{mst-I}^{center}| \geq 2$. Consider the center v_i of a disk $N(v_i)$ in I. Then, v_i is connected to a set of nodes $\{w_1, w_2, \cdots, w_{l_i}\}$ in T_{mst-I}^{center} (see Fig 2). Let $\{(v_i, w_1), (v_i, w_2), \cdots, (v_i, w_{l_i})\}$ and $\{p_1, p_2, \cdots, p_{l_i}\}$ be the set of points where each edge in $\{(v_i, w_1), (v_i, w_2), \cdots, (v_i, w_{l_i})\}$ intersects with the boundary of $N(v_i)$. Let $\{q_1, q_2, \cdots, q_{l_i}\}$ be the points on the middle of each edge in $\{(v_i, w_1), (v_i, w_2), \cdots, (v_i, w_{l_i})\}$. Then, T_{mst-I}^{center} can be partitioned into a set of clusters $C(v_i)$ of edges $\{(v_i, q_1), (v_i, q_2), \cdots, (v_i, q_{l_i})\}$ such that $\bigcup_{\forall i} C(v_i) = T_{mst-I}^{center}$ and for any different i and j, there is no edge segment shared by $C(v_i)$ and $C(v_j)$. Also, let $\{a_1, a_2, \cdots, a_{l_i}\}$ be the set of arcs such that a_i is the portion of the boundary of $N(v_i)$ from a_i to $a_{(i+1) \mod l_i}$. Note that the union of the l_i arcs is the whole boundary of $N(v_i)$.

Now, let $\alpha_i = \sum_{1 \leq j \leq l_i} Eucdist(v_i, q_j)$. Since no two disks in I overlap with each other, $d \leq Eucdist(v_i, q_j)$ for all j. Also, from our assumption, $d \geq 1$. As a result, we have $l_i \leq l_i \cdot d \leq \alpha_i$. Then,

$$\frac{Len(T_{out})}{Len(T_{mst-I}^{center})} = \frac{\sum_i(\alpha_i - l_i \cdot d + 2\pi d)}{\sum_i \alpha_i}. \tag{1}$$

For each i, we have

$$\begin{aligned} &\frac{\alpha_i - l_i \cdot d + 2\pi d}{\alpha_i} = 1 + d(\frac{2\pi - l_i}{\alpha_i}) \\ &\leq 1 + d(\frac{2\pi - l_i}{l_i}) \quad \text{/* if } 2\pi - l_i \geq 0\text{*/} \\ &\leq d(2\pi - 1) + 1, \quad \text{/* reach maximum when } l_i = 1\text{*/} \end{aligned} \tag{2}$$

since we assumed $|T_{mst-I}^{center}| \geq 2$, which implies $l_i \geq 1$. Note that in Equation (2), if $2\pi - l_i < 0$, then $1 + d(\frac{2\pi - l_i}{\alpha}) \leq 1 \leq d(2\pi - 1) + 1$. Next, suppose $|T_{mst-I}^{center}| = 1$. Then, we have

$$Len(T_{out}) \leq 2\pi d. \tag{3}$$

By combining Equation (1)$\sim$(3), we can conclude that $Len(T_{out}) \leq (d(2\pi - 1) + 1)Len(T_{mst-I}^{center}) + 2\pi d$. ■

Lemma 4.2: [16] Given a set V of nodes $\{v_1, v_2, \cdots, v_n\}$, let T_{mst-I}^{center} and T_{mst-I}^{disk} be an MST of V and an MST of a set of "non-overlapping" disks $\{N(v_1), \cdots, N(v_n)\}$. Then, $Len(T_{mst-I}^{center}) \leq (1 + \frac{20}{\pi})Len(T_{mst-I}^{disk}) + 2d$, where d is the uniform radius of the disks.

Theorem 4.3: Suppose T_{out} is an output of GMSTNA and T_{mst}^{disk} is an MST of the neighborhood areas of all nodes in V. Then, $Len(T_{out}) \leq (d(2\pi - 1) + 1)((1 + \dfrac{20}{\pi})Len(T_{mst}^{disk}) + 2d) + 2\pi d$.

Proof: From Lemma 4.1, we have $Len(T_{out}) \leq (d(2\pi - 1) + 1)Len(T_{mst-I}^{center}) + 2\pi d$. Since all of the disks in I are pair-wise disjoint with each other, by Lemma 4.2, we have

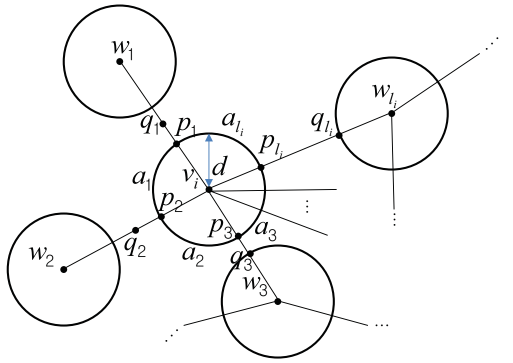

Fig. 2. A maximal pair-wise disjoint disks with radius d and associated notations for the performance analysis.

$Len(T^{center}_{mst-I}) \leq (1+\frac{20}{\pi})Len(T^{disk}_{mst-I}) + 2d$, where T^{disk}_{mst-I} is an MST of the disks in I. In addition, $Len(T^{disk}_{mst-I}) \leq Len(T^{disk}_{mst})$ since T^{disk}_{mst-I} is spanning over less number of disks than T^{disk}_{mst} and both of them are MSTs. In conclusion, we have

$$\begin{aligned}
&Len(T_{out}) \leq (d(2\pi-1)+1)Len(T^{center}_{mst-I}) + 2\pi d \\
&\leq (d(2\pi-1)+1)((1+20/\pi)Len(T^{disk}_{mst-I}) + 2d) + 2\pi d \\
&\leq (d(2\pi-1)+1)((1+20/\pi)Len(T^{disk}_{mst}) + 2d) + 2\pi d.
\end{aligned}$$

■

B. *k-TCPNA: A Constant Factor Approximation of k-TCPN*

Now, we introduce our constant factor approximation algorithm for k-TCPN, namely *the k-tree cover problem with neighborhood algorithm (k-TCPNA)*. In the first step, k-TCPNA computes a maximal pair-wise disjoint set I of the circular neighborhood areas (disks) of all nodes in V. This can be done by using a variation of the 2-coloring strategy (see Step 1 of Algorithm 2 for more details). Our coloring strategy offers the following two important properties to the disks in I. First, no two disks in I overlap (or touch) with each other. Second, no disk in I overlaps with the neighborhood of any root $N(r_i)$.

In the second step, the $(4+\epsilon)$-approximation algorithm for the k-rooted tree cover problem in [13] is applied over the centers of the disks in I and R. As a result, we have k rooted trees, $T_M = \{T_1, T_2, \cdots, T_k\}$. We would like to emphasize

1) each T_i is rooted at r_i,
2) the neighborhoods of any pair of nodes in $V(T_i)$ including $N(r_i)$ do not overlap (or touch) with each other,
3) most importantly, there are some nodes in V whose neighborhoods are not covered by any T_i since their neighborhoods are overlapping with some disk(s) in I.

In the third step, the third step of GMSTNA in Section IV-A (Algorithm 1) is applied to $V(T_i)$ for each $T_i \in T_M$. Apparently, after this step, the neighborhood area of each node in V has to be visited by some tree, and therefore, we can obtain a feasible solution of k-TCPN. Note that each tree resulted from the third step of k-TCPNA may have a curve in its edge set, which can be replaced with a shorter direct line.

Algorithm 2 k-rooted Tree Cover Problem with Neighborhood Algorithm (k-TCPNA) $(R = \{r_1, \cdots, r_k\}, V)$

```
1: Suppose $D = \{N(r_1), \cdots, N(r_k), N(v_1), \cdots, N(v_n)\}$.
   Compute a subset $I \subseteq D$ as follows.
   (a) Color all disks in $D$ white.
   (b) Color all $N(r_i)$s in $D$ black and the $N(v_j)$s overlapping
       (or touching) with any $N(r_i)$ gray.
   (c) Find a white $N(v_i)$ which is overlapping (or touching)
       with the most number of white $N(v_j)$s for some $j$, and
       color the $N(v_i)$ black and the $N(v_j)$s gray.
   (d) Repeat Step (c) until there exists no white disks in $D$.
   (e) Add all black $N(v_i)$s into $I$ (not $N(r_j)$s).
   As a result, $I$ is a maximal independent (pairwise-disjoint)
   neighborhoods of the nodes in $D \setminus \{N(r_1), \cdots, N(r_k)\}$.
2: Apply the $(4+\epsilon)$-approximation algorithm for the $k$-rooted
   tree cover problem in [13] on the centers of the disks in
   $I$ and the $k$ mobile elements in $R$. As a result, we have a
   set $T_M = \{T_1, T_2, \cdots, T_k\}$ of $k$ rooted trees, where each
   $T_i$ is rooted at $r_i$. For each $T_i \in T_M$, if $T_i$ is not an MST
   of $V(T_i)$, we force it.
3: For each $T_i$ (an MST with root $r_i$), apply Step 3 of
   GMSTNA in Section IV-A on $V(T_i)$ (including $r_i$) and
   obtain $T'_i$.
4: Suppose $T_{out} = \{T'_1, T'_2, \cdots, T'_k\}$. Apply our optimiza-
   tion strategy in Section IV-C on $T_{out}$ to eliminate all of
   curves from $E(T'_i)$ for each $T'_i \in T_{out}$ and connect each
   $T'_i$ to $r_i$.
```

Therefore, in the last step, the algorithm further optimizes the output. More details of this step will be introduced in the following subsection.

Algorithm 2 is the formal definition of k-TCPNA. Now, we show that Algorithm 2 is a constant factor approximation of k-TCPN.

Observe that after Step 1.(b), the neighborhood areas of two different r_i and r_j may overlap with each other. However, this does not hinder us from applying our performance analysis of Algorithm 1 to the performance analysis of Algorithm 2 since in each T_i in Step 2, the neighborhood areas of all nodes in T_i are disjoint with each other.

Lemma 4.4: Consider a set R of k roots and a set of nodes $V = \{v_1, v_2, \cdots, v_n\}$ such that their circular neighborhoods with radius d are "pairwise disjoint" with each other. Let $OPT^{center-I}$ and OPT^{disk-I} be the optimal solution of the k-rooted tree cover problem and k-TCPN defined over $\langle V, R\rangle$, respectively. Then, $Cost(OPT^{center-I}) \leq (1 + 20/\pi)Cost(OPT^{disk-I}) + 2d$.

Proof: Suppose we have OPT^{disk-I}. Now, from OPT^{disk-I}, construct a feasible solution $S = \{\tilde{T}_1, \cdots, \tilde{T}_k\}$ of the k-rooted tree cover problem instance defined over $\langle R, V\rangle$ as follows: for each T_i in OPT^{disk-I}, let $W = \{w_1, w_2, \cdots, w_l\}$ be the set of nodes whose neighborhoods are touched by T_i. Then, we can construct an MST $\tilde{T}_i$ of

$W \bigcup \{r_i\}$. Clearly, $Cost(OPT^{center-I}) \leq Cost(S)$ since $OPT^{center-I}$ is an optimal solution.

Now, suppose T^{disk-I}_{max} is the tree in OPT^{disk-I} with maximum total edge weight. Then, by Lemma 4.2, for any $\tilde{T}_j \in S$, we have $Len(\tilde{T}_j) \leq (1+\frac{20}{\pi})Len(T^{disk-I}_{max})+2d$, and therefore, we have

$$\begin{aligned} &Cost(OPT^{center-I}) \leq Cost(S) = \max_{\forall j} Len(\tilde{T}_j) \\ &\leq (1+\tfrac{20}{\pi})Len(T^{disk-I}_{max}) + 2d \\ &= (1+\tfrac{20}{\pi})Cost(OPT^{disk-I}) + 2d. \end{aligned}$$

■

Theorem 4.5: k-TCPNA is a constant factor approximation of k-TCPN.

Proof: Let T^{disk-V}_{max} and $T^{center-I}_{max}$ be the tree $T'_i \in T_{out}$ with the maximum total weight and the tree $T_j \in T_M$ with the maximum total weight after the completion of k-TCPNA, respectively. We would like to remind you that T_{out} is covering the neighborhoods of all nodes, while T_M is covering the centers of the disks in I. From Lemma 4.1, we have

$$Len(T^{disk-V}_{max}) \leq (d(2\pi-1)+1)Len(T^{center-I}_{max})+2\pi d. \quad (4)$$

Now, suppose V_I is the set of centers of disks in I. Let $OPT^{center-I}_{max}$ be the maximum cost tree in an optimal solution of the k-rooted tree cover problem defined over V_I. Since in Algorithm 2, we used a $(4+\epsilon)$-approximation algorithm for the k-rooted tree cover problem in [13], we have

$$Len(T^{center-I}_{max}) \leq (4+\epsilon)Len(OPT^{center-I}_{max}). \quad (5)$$

Furthermore, from Lemma 4.4, we have

$$\begin{aligned} &Len(OPT^{center-I}_{max}) \\ &\leq (1+20/\pi)Len(OPT^{disk-I}_{max})+2d \quad (6) \\ &\leq (1+20/\pi)Len(OPT^{disk-V}_{max})+2d, \end{aligned}$$

where OPT^{disk-I}_{max} is the maximum cost tree in an optimal solution of the k-TCPN defined over I and OPT^{disk-V}_{max} is the maximum cost tree in an optimal solution of the k-TCPN defined over V. By combining Equations (4)∼(6), we have

$$\begin{aligned} Cost(T_{out}) &= Len(T^{disk-V}_{max}) \\ &\leq (d(2\pi-1)+1)Len(T^{center-I}_{max})+2\pi d \\ &\leq (d(2\pi-1)+1)(4+\epsilon)Len(OPT^{center-I}_{max})+2\pi d \\ &\leq (4+\epsilon)\cdot(d(2\pi-1)+1)\cdot \\ &\qquad ((1+20/\pi)Cost(OPT^{disk-V}_{max})+2d)+2\pi d. \end{aligned} \quad (7)$$

■

C. Optimization of k-TCPNA

Each tree in T_{out}, an output of Algorithm 2 may contain curves (i.e. neighborhood boundaries). Therefore, we can further reduce the cost of the output by carefully replacing the curves with straight lines.

In detail, consider $T_{out} = \{T'_1, T'_2, \cdots, T'_k\}$ and $T_M = \{T_1, T_2, \cdots, T_k\}$ in Algorithm 2. Suppose $A = V \bigcap (T_1 \bigcup T_2 \bigcup \cdots \bigcup T_k)$ and $B = V \setminus A$. Now, we construct a set of subsets $X = \{X_1, X_2, \cdots, X_k\}$ as follows (see Fig 3):

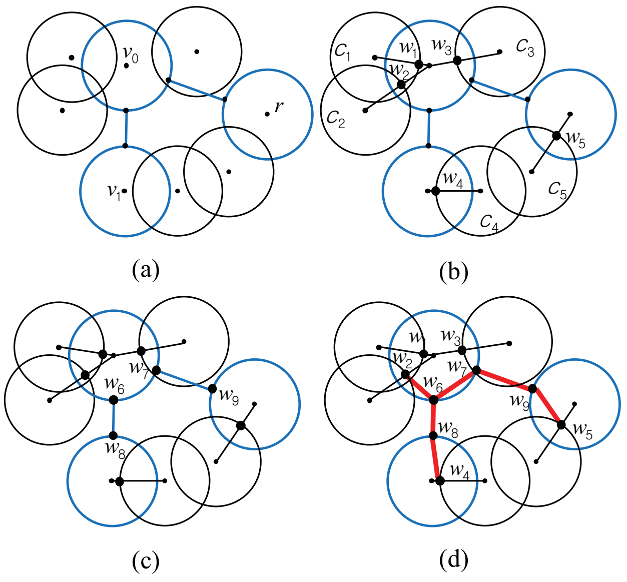

Fig. 3. (a): a set of k rooted trees are obtained (the thick lines and curves) by executing Algorithm 2. (b): a set of points $\{w_1, w_2, w_3, w_4, w_5\}$ are selected. Note that, each of these points are shared by a disk centered at a node outside the rooted trees and a disk centered at a node inside some rooted tree. Also, observe that a point w_j for some c_j will be added to X_i only if $V(T_i)$ has the node which is nearest to the center of c_j (i.e. v_0 for w_1 and v_1 for w_4) than another other rooted tree in T_{out}, the output of Algorithm 2. (c): a set of points (i.e. $\{w_6, w_7, w_8, w_9\}$ in the figure) are selected and added to X_i for each T_i. Note that an MST over them connects all the neighborhoods of the nodes in $V(T_i)$. (d): for each X_i, we remove some useless points using the set-cover approximation strategy (i.e. w_2 is preferred over w_1 since w_2 covers $\{c_1, c_2, N(v_0)\}$ while w_1 only covers $\{c_1, N(v_0)\}$) and induce $\tilde{X}_i \subseteq X_i$, and calculate an MST of $\tilde{X}_i$.

1) For each node $u \in B$, find its nearest node $v \in A$. Then, $v \in V(T_j)$ for some $1 \leq j \leq k$. Note that $N(u)$ and $N(v)$ must overlap (or touching) with each other. Consider an edge e connecting u and v. Let w be the point where e and $N(u)$ meet (the set of big bold points $\{w_1, w_2, w_3, w_4, w_5\}$ in Fig 3(b)). Add w to X_j.
2) For each T'_i, suppose $\tilde{e}$ is an edge in T'_i connecting two disks $N(u)$ and $N(v)$ such that $u, v \in T_i$ (the set of big bold points $\{w_6, w_7, w_8, w_9\}$ in Fig 3(c)). Add the two points where T'_i and $N(u)$ meet and T'_i and $N(v)$ meet to X_i for every $\tilde{e}$ in T'_i.
3) After the first step, we have a set of nodes X_i for each i such that each disk centered at a node in T_i includes at least one point in X_i. In fact, we can think of this as an instance of the set-cover problem and thus further reduce the size of X_i using a famous set-cover approximation algorithm. In detail, let D_i be disk which is centered at some node in T_i. Next, prepare an empty set $\tilde{X}_i$ and we repeat the following until D_i is empty.
   a) Pick a point $x \in X_i$ which covers the most number of disks in D_i (x covers a disk if x is in the disk).
   b) Add x to $\tilde{X}_i$ and remove all the disks covered by x.

4) Compute an MST XT_i of each $\tilde{X}_i$.
5) Finally, outputs $XT_{out} = \{XT_1, XT_2, \cdots, XT_k\}$.

Clearly, XT_{out} is a feasible solution of k-TCPN since it includes k rooted trees and the neighborhood of each node in V is visited by one of the trees. Also, $Cost(XT_{out}) \leq Cost(T_{out})$ since a) with our optimization method, each circular neighborhood added in the second step of Algorithm 1 can replaced by a set of straight lines connecting some points on the circular neighborhood, and b) the cost of the final MST is even smaller than this.

Observe that each XT_i may not connected to r_i and the distance from r_i to its nearest node in $V(XT_i)$ is at most d. Strictly speaking, Step 4) above should be "Add r_i to $\tilde{X}_i$ and compute an MST XT_i of each $\tilde{X}_i$", and we have $Cost(XT_{out}) \leq Cost(T_{out}) + d$. After this necessary modification, from Equation (7), we finally have

$$\begin{aligned} Cost(XT_{out}) &\leq (4+\epsilon)\cdot(d(2\pi-1)+1)\cdot \\ &\quad ((1+20/\pi)Cost(OPT^{disk-V}_{max}) + 2d) + (2\pi+1)d. \end{aligned} \tag{8}$$

D. Constant Factor Approximations of k-TSPN and k-PCPN

We first present following remark (Remark 1) demonstrating an interesting relationship between k-TCPN and k-PCPN, which leads to a constant factor approximation of k-PCPN (Theorem 4.6). We would like to make a note that a concept similar to Remark 1 is used in [13], but has not been formally proven. Similarly, a constant factor approximation algorithm for k-TSPN can be obtained (Corollary 4.1) from the relationship between k-TCPN and k-TSPN (Remark 2).

Remark 1: An α-approximation algorithm for k-TCPN can be modified to obtain an 1.5α-approximation for k-PCPN in polynomial time.

Proof: Note that a problem instance of both k-TCPN and k-PCPN consists of a set $R = \{r_1, r_2, \cdots, r_n\}$ of k roots and a set $V = \{v_1, v_2, \cdots, v_n\}$ of n nodes. Suppose we have applied an α-approximation algorithm for k-TCPN and obtained a feasible solution $T = \{T_1, \cdots, T_k\}$. Let $T_{max} = \max_{1\leq i\leq k} Len(T_i) = Cost(T)$.

Since the performance ratio of this algorithm is α, we have

$$Len(T_i) \leq Len(T_{max}) \leq \alpha Len(T^{opt}_{max}), \tag{9}$$

for all i, where T^{opt}_{max} is a tree in an optimal solution of k-TCPN with the maximum cost.

Next, convert each tree T_i rooted at r_i into a tour T'_i of all the nodes in $V(T_i)$ using the famous 1.5-approximation for TSP by Christofides [17]. At last, convert each tour U_i into a path P_i by removing an edge between r_i and a node in the tour. Then, we have

$$Len(P_i) \leq Len(U_i) \leq 1.5Len(T_i) \tag{10}$$

for every $1 \leq i \leq k$. By combining Equation (9) and Equation (10), we have

$$Len(P_i) \leq 1.5\alpha Len(T^{opt}_{max}). \tag{11}$$

Furthermore, we have

$$Len(T^{opt}_{max}) \leq Len(P^{opt}_{max}), \tag{12}$$

since a path is a special type of a tree, where P^{opt}_{max} is a path with the maximum cost in an optimal solution of k-PCPN. From Equation (11) and Equation (12), we have

$$Len(P_i) \leq 1.5\alpha Len(P^{opt}_{max}),$$

for all $1 \leq i \leq k$.

In summary, by utilizing an α-approximation algorithm for k-TCPN and the 1.5-approximation for TSP, we can obtain 1.5α-approximation for k-PCPN. In addition, the procedure would take $O(n^2)$ time. Therefore, this theorem holds true. ■

Remark 2: An α-approximation algorithm for k-TCPN can be modified to obtain an 1.5α-approximation for k-TSPN in polynomial time.

Proof: From Equation (9) and Equation (10), we have

$$Len(U_i) \leq 1.5\alpha Len(T^{opt}_{max}).$$

By combining this with

$$Len(T^{opt}_{max}) \leq Len(U^{opt}_{max}),$$

where U^{opt}_{max} is the maximum length tour in an optimal solution of k-TSPN, we can prove this theorem. ■

Theorem 4.6: There exists a constant factor approximation algorithm for k-PCPN.

Proof: From Theorem 4.5 and Remark 1, we have a polynomial approximation algorithm for k-PCPN whose cost is bounded by

$$\begin{aligned} &(4+\epsilon)\cdot \\ &\quad ((d(2\pi-1)+1)((1+\tfrac{20}{\pi})Cost(OPT_T)+2d)+(2\pi+1)d) \\ &\leq 1.5\cdot(4+\epsilon)\cdot \\ &\quad ((d(2\pi-1)+1)((1+\tfrac{20}{\pi})Cost(OPT_P)+2d)+(2\pi+1)d), \end{aligned}$$

where $Cost(OPT_T)$ the cost of an optimal solution of k-TCPN, $Cost(OPT_P)$ is the cost of an optimal solution of k-PCPN, $d \geq 1$ is the radius of the circular neighborhood area of each node in V, and ϵ is a small positive constant. ■

Corollary 4.1: There exists a constant factor approximation algorithm for k-TSPN.

Proof: This naturally follows from Theorem 4.6 after we replace OPT_P to OPT_U, where OPT_U is an optimal solution of k-TSPN. ■

V. Simulation Results and Discussion

As we mentioned before, the $(4+\epsilon)$-approximation algorithm for the k-rooted tree cover problem in [13], can compute feasible solutions for k-TCPN, k-TSPN, and k-PCPN. To the best of our knowledge, this is the only algorithm to compete with our algorithms for solving k-TCPN, k-TSPN, and k-PCPN, where the mobile elements are randomly deployed. In the rest of this section, we call this algorithm "k-TCPA".

In this simulation, we first prepare a $x \times x$ virtual 2-D space and randomly deploy a number of nodes, where x is 10, 15, or 30. Then, we randomly deploy n nodes and k mobile

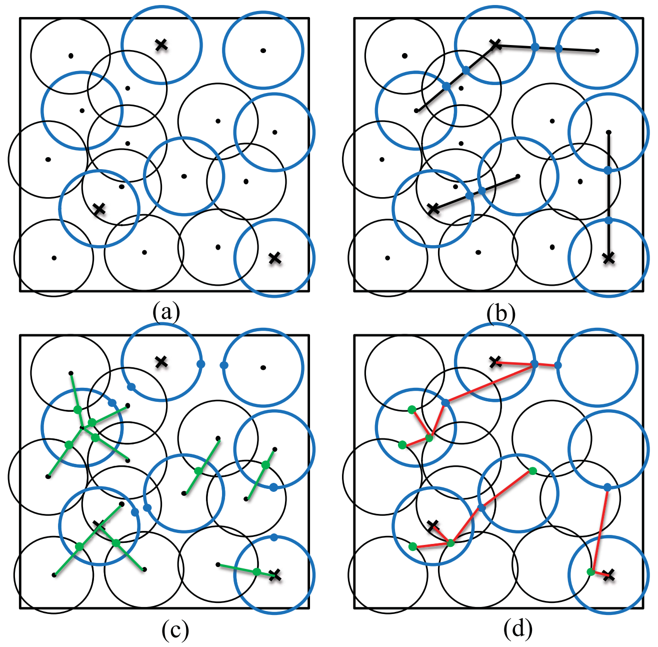

Fig. 4. These figures illustrate how Algorithm 2 work. The bold crosses are the locations of the mobile elements and the small dots are sensor nodes. In Figure (a), the set of pairwise-disjoint circles are selected (those with bold border lines). Note that the neighborhoods of mobile elements are always selected. In Figure (b), the $(4+\epsilon)$-approximation algorithm for the k-rooted tree cover problem [13] is applied to the union of selected centers and mobile elements from Figure (a). As a result, we have three rooted-trees (each connected with black lines). For each rooted-tree, each point where the neighborhood of selected node (or center) and the line in the tree meet is marked. In Figure (c), each node v which is not selected in Figure (a) identifies a node u which is 1) closest to v and 2) selected in Figure (a). Then, draw a line l between v and u, and mark the point where the neighborhood of v meets l. In Figure (d), for each rooted-tree, an MST of the marked points from Figure (b) and Figure (c) are constructed. Note that not all of them are necessary to be connected and thus some of them are not considered for this final tree construction as discussed in Section IV-C.

TABLE I
THE SIMULATION RESULTS INDICATES THAT OUR ALGORITHM (k-TCPNA) OUTPERFORMS THE $(4+\epsilon)$-APPROXIMATION ALGORITHM FOR THE k-ROOTED TREE COVER PROBLEM IN [13] (k-TCPA) IN TERMS OF SOLVING k-TCPN, k-TSPN, AND k-PCPN. THE PERFORMANCE GAP GETS LARGER AS THE DENSITY OF THE NETWORK GOES UP.

(a) 10×10 space with $k = 3$

n	Tree-IR	Tree-DR	Tour-IR	Tour-DR	Path-IR	Path-DR
30	0.313	0.687	0.251	0.749	0.263	0.737
50	0.355	0.645	0.308	0.692	0.320	0.680
70	0.435	0.565	0.381	0.619	0.389	0.611

(b) 10×10 space with $k = 6$

n	Tree-IR	Tree-DR	Tour-IR	Tour-DR	Path-IR	Path-DR
30	0.182	0.818	0.132	0.868	0.135	0.865
50	0.188	0.812	0.147	0.853	0.177	0.823
70	0.270	0.730	0.239	0.761	0.255	0.745

(c) 15×15 space with $k = 3$

n	Tree-IR	Tree-DR	Tour-IR	Tour-DR	Path-IR	Path-DR
30	0.263	0.737	0.210	0.790	0.215	0.785
50	0.276	0.724	0.241	0.759	0.26	0.74
70	0.322	0.678	0.273	0.727	0.282	0.718

(d) 15×15 space with $k = 6$

n	Tree-IR	Tree-DR	Tour-IR	Tour-DR	Path-IR	Path-DR
30	0.192	0.808	0.132	0.868	0.135	0.865
50	0.219	0.781	0.175	0.825	0.191	0.809
70	0.275	0.725	0.234	0.766	0.255	0.745

(e) 30×30 space with $k = 3$

n	Tree-IR	Tree-DR	Tour-IR	Tour-DR	Path-IR	Path-DR
30	0.166	0.834	0.129	0.871	0.118	0.882
50	0.156	0.844	0.126	0.874	0.123	0.877
70	0.188	0.812	0.156	0.844	0.160	0.840

(f) 30×30 space with $k = 6$

n	Tree-IR	Tree-DR	Tour-IR	Tour-DR	Path-IR	Path-DR
30	0.177	0.823	0.118	0.882	0.101	0.899
50	0.187	0.813	0.128	0.872	0.121	0.879
70	0.190	0.810	0.159	0.841	0.152	0.848

elements, where n is 30, 50, or 70, and k is 3 or 6. For each parameter setting with x, k, and n fixed, we generate 100 problem instances and calculate the average of the cost of the outputs of k-TCPA and our k-TCPNA (optimized version). In detail, we first apply both k-TCPA and our k-TCPNA and generate solutions of k-TCPN. Then, we convert their outputs to solutions of k-PCPN and k-TSPN.

Table I summarizes our simulation results. In the table, we use the following two metrics for the performance comparison. The first one is the improvement ratio (IR), which is defined as

$$\frac{\text{cost of output of } k\text{-TCPA} - \text{cost of output of } k\text{-TCPNA}}{\text{cost of output of } k\text{-TCPA}}.$$

The second metric is the direct ratio (DR), which is defined as

$$\frac{\text{cost of output of } k\text{-TCPNA}}{\text{cost of output of } k\text{-TCPA}}.$$

Note that k-TCPNA outperforms k-TCPA as IR increases and as DR decreases. In the table, Tree-IR, Tour-IR, and Path-IR refer to the IR in the calculation of k-TCPN, k-TSPN, and k-PCPN, respectively. Tree-DR, Tour-DR, and Path-DR are also defined similarly.

Now, we make following three claims and explain how each claim is supported by our simulation results.

- Claim 1: As the number of nodes increases, k-TCPNA outperforms k-TCPA.
- Claim 2: As the size of the virtual space decreases, k-TCPNA outperforms k-TCPA.
- Claim 3: As we have less number of mobile elements, k-TCPNA outperforms k-TCPA.

To see the correctness of Claim 1, we compare the rows within the same table. In Table II(a), when $n = 30$, Tour-IR is 0.251 and Path-IR is 0.263. The IR value becomes much bigger with $n = 70$, in which Tour-IR is 0.381 and Path-IR is 0.389. Since a larger IR value implies k-TCPNA outperforms k-TCPA, this claim is supported by this table. Since such a trend is consistently observed in Table II(b) to Table II(f), our claim is supported by the simulation results.

To see the correctness of Claim 2, we compare Table II(a), Table II(c), and Table II(e) (say Group 1). Also, we need to compare Table II(b), and Table II(d), and Table II(f) (say Group 2). In Group 1, k is fixed to 3. When $n = 30$, Tour-

IR are 0.251, 0.210, 0.129, and Path-IR are 0.263, 0.215, 0.118 in the space with size 10×10 (Table II(a)), 15×15 (Table II(c)), and 30×30 (Table II(e)), respectively. In addition, when $n = 70$, Tour-IR are 0.381, 0.273, 0.156, and Path-IR are 0.389, 0.282, 0.160 in the space with size 10×10 (Table II(a)), 15×15 (Table II(c)), and 30×30 (Table II(e)), respectively. As we can see, independent of n, Claim 2 is true since when the size of the virtual space is small, IR goes up. In fact, the similar trend can be observed in the tables in Group 2. Therefore, Claim 2 is supported by the simulation results.

Lastly, to see the correctness of Claim 3, we compare Table II(a) with Table II(b), Table II(c) with Table II(d), and Table II(e) with Table II(f), respectively. In general, with n and the size of virtual space fixed, the performance gap between them becomes larger with small k, and this claim is true. This is because with more number of mobile nodes, the workload among the available mobile elements is better distributed, and that helps to reduce the performance gap between k-TCPNA and k-TCPA.

Clearly, from Claim 1 and Claim 2, we can conclude that our algorithm k-TCPNA outperforms k-TCPA in denser WSN. The fact, k-TCPA ignores the neighborhood of each node and visits each node directly while k-TCPNA visits the neighborhood of each node in an intelligent way, could be one contributing factor. However, from the simulation results, we can learn that the design of k-TCPNA, to visit non overlapping neighborhoods first and expand the tree minimally works very effectively.

VI. Conclusions and Future Work

In this paper, we considered the problem of computing the optimal trajectories of multiple mobile elements (e.g. robots, vehicles, etc.) to minimize data collection latency in wireless sensor networks (WSNs). Depending on slightly different assumption, we propose two versatile problems. Since both of them are NP-hard, we propose constant factor approximation algorithms for them. Our simulation results show our algorithms outperform their competitors on average. Still, our results pose several interesting future work. First, since our algorithms for k-TSPN and k-PCPN produce solutions through several steps of conversions, their approximation ratios are not small. While our simulation result indicate the performance of our algorithms are much better than any existing solution on average, the investigation of direct approaches to obtain approximations with smaller worst case bound for the k-TSPN and k-PCPN are of great theoretical interest. Second, our formulations did not reflect some of the application specific constraints such as the data ratio of each sensor node. Thus, integrating such new requirements into our models would be of great network research interest.

VII. Acknowledgement

This work was supported in part by US National Science Foundation (NSF) CREST HRD-0833184. This work was supported in part by US Army Research Office (ARO) W911NF-0810510. It is also partly supported by National Natural Science Foundation of China (NSFC) 11071191. It is also supported in part by NSF under grants CNS-1016320 and CCF-0829993.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] D. Ciullo, G.D. Celik, and Eytan Modiano, "Minimizing Transmission Energy in Sensor Networks via Trajectory Control," in *Proc. of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2010, pp. 132-141, Avignon, France, May 31 2010-June 4 2010.

[8] 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.

[9] "NEPTUNE Canada," [Online]. Available: http://www.neptunecanada.ca

[10] "Underwater Robots Track Oil and Ocean Life," [Online]. Available: http://spectrum.ieee.org/podcast/robotics/industrial-robots/ underwater-robots-track-oil-and-ocean-life.

[11] 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.

[12] 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.

[13] 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.

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

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

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

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

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

[19] J.S.B. Mitchell, "A Constant-Factor Approximation Algorithm for TSP with Pairwise-Disjoint Connected Neighborhoods in the Plane," in the Proc. of *the 2010 Annual Symposium on Computational Geometry (SoCG 2010)*, Snowbird, Utah, USA, June 13-16, 2010.