# Energy-Constrained Bi-Objective Data Muling in Underwater Wireless Sensor Networks

Ke Li, *Student Member, IEEE,* Chien-Chung Shen, *Member, IEEE,* and Guaning Chen, *Member, IEEE*

**Abstract**—For underwater wireless sensor networks (UWSNs), data muling is an effective approach to data gathering, where sensor data are collected when a mobile data mule travels within the wireless communication range of the sensors. However, given the constrained energy available on a data mule and the energy consumption of its motions and communications a data mule may be prevented from visiting every deployed sensor in a tour. We formulate the tour planning of a data mule collecting sensor data in UWSNs as an energy-constrained bi-objective optimization problem termed the Underwater Data Muling Problem (UDMP). UDMP has the two conflicting objectives of maximizing the number of sensors contacted and minimizing the length of a tour, while satisfying the energy constraint on the data mule at all times. We design two heuristic algorithms to solve one special case and one generalized case of this NP-hard problem, respectively. Each algorithm computes a set of Pareto-efficient solutions addressing the tradeoff between the two optimization objectives to facilitate tour planning. Simulation results validate the effectiveness of both algorithms.

**Index Terms**—Underwater wireless sensor networks, data muling, tour planning, heuristic algorithm.

## 1 Introduction

*Data muling* [1] has been shown to be an effective approach to data collection in underwater wireless sensor networks (UWSNs) [2]. A data mule is a mobile robot equipped with data storage, battery, and wireless communications capability. A data mule starts off from a sink (or depot), traverses the network to collect data (via direct wireless communications) from individual sensors, and transports the collected data back to the sink. Ideally, we would like to plan a tour to collect data from *all* the sensors, whereas the overall tour length is minimized. However, the *finite* battery energy available on a mobile mule may not support the expedition of reaching all the deployed sensors along a data collecting tour.

This paper investigates the tour planning for underwater data muling as an energy-constrained bi-objective optimization problem, termed the *Underwater Data Muling Problem* (UDMP), which is proved to be NP-hard. Given a constraint on a data mule's finite battery energy, two competing optimization objectives are considered at the same time: the maximization of the number of visited sensors (*cover-objective*) and the minimization of the tour length (*length-objective*). In particular, to sustain the long-running data muling operations inside deep ocean to cover large geographical volumes, underwater "energy stations" that harvest ocean currents for renewable energy would be deployed within the ocean where a data mule may *dock* to be recharged. With the usage of energy (or docking) stations, we design two heuristic algorithms to solve one special case called UDMP-AD (all-docking, where all the energy stations are to be visited) and one generalized case called UDMP-DoD (docking-on-demand, where some energy stations are to be visited on a needed basis), respectively. Simulation results show that a computed heuristic UDMP-AD tour that covers all the sensors is much closer to the corresponding optimal single length-objective tour, and often outperforms the optimal solution of another closely related NP-hard problem, and the UDMP-DoD heuristic could further decrease the tour length in comparison to UDMP-AD solutions.

The paper proceeds in Section 2 with a review of the related tour planning problems and solutions. Section 3 introduces the visit/cover concepts and the proposed energy consumption model of the data mule. Section 4 formulates UDMP, together with two problem cases. Section 5 describes heuristic tour planning algorithms for the two UDMP cases, together with analyses of their space and time complexity. Simulation results are discussed in Section 6, and Section 7 concludes the paper.

- *K. Li and C.-C. Shen are with the Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716. Emails: {kli, cshen}@cis.udel.edu.*
- *G. Chen is with the Department of Information Management, Kainan University, Taiwan. Email: gchen@mail.knu.edu.tw.*

*A preliminary version of this paper appeared in the 7th IEEE Int'l Conf. on Mobile Adhoc and Sensor Syst. (MASS), San Francisco, CA, Nov 8–12, 2010.*

## 2 Related Work

### 2.1 Single Objective Tour Planning

Among existing tour planning problems, the Traveling Salesman Problem (TSP) has been studied extensively, which computes a shortest tour of given cities starting from an origin city, visiting each other city exactly once, and returning to the city of origin. The problem is NP-hard even in the special case of vertices on the $2D$ Euclidean plane [3]. Fortunately, the Euclidean TSP can be approximated in polynomial time by ratio $(1+\varepsilon)$ $(\forall \varepsilon > 0)$ [4]. Given expected data traffic in stationary sparse ad hoc networks, Zhao et al. [5], [6] formulated the message ferry problem to minimize the average delivery delay, which is TSP when the traffic matrix is symmetric. Tariq et al. [7] futher designed ferry routes for mobile node cases by adding a wait time at each stop so that every node is contacted with certain probability and the total time of traveling and waiting is minimized. Note that the message ferry problem assumes unlimited energy for a ferry.

A number of NP-hard tour planning problems have been proposed as generalizations of TSP. One direction suitable for data muling with wireless communications is to remove the assumption of TSP that all cities must be on the tour. One effort of this kind is the Covering Salesman Problem (CSP) [8], where a covering distance $D$ is introduced, with the goal of finding a minimum cost tour through only a subset of the given vertices such that every vertex is either on the tour or within distance $D$ of some vertex on the tour. Another similar effort is the Covering Tour Problem (CTP) [9], where two vertex sets are specified—a set $V$ of candidate tour stops and a set $W$ of targets with covering distance $D$, and the aim is to cover all vertices in $W$ by a subset of $V$ chosen as actual tour stops on a minimum cost tour. Recently both problems have been applied to mobile data gathering in wireless sensor networks (WSNs). Rao et al. [10] modeled the trajectory computation problem for the mobile data harvester as CSP. Ma and Yang [11] formulated a single-hop data gathering problem as CTP. Zhao and Yang [12] further proposed a bounded relay hop mobile data collection problem as a variation of CSP, to balance between the tour length and the relay hop count of local data aggregation. One common heuristic used for solving CSP/CTP is to carefully choose a subset of vertices, usually in a greedy fashion, as tour stops to cover all the given vertices and then find a TSP tour through these chosen covering vertices. Although these algorithms often generate good results in practice, it is hard to analyze their approximation ratios. Note that the formulation of CSP/CTP does not take advantage of the fact that data exchange with sensors could proceed while a mule is traveling between tour stops, even when sensors are not covered by the mule stationarily at any tour stop. In such cases, sensors within the communication range *along a tour* could be covered by the data mule traveling between consecutive tour stops.

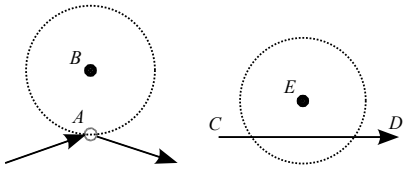
Arkin and Hassin [13] first studied TSP with neighborhoods (TSPN) as a geometric version of CSP. Given a set of connected regions (called neighborhoods), TSPN computes a minimum cost tour that intersects all regions. Constant-ratio approximation algorithms were proposed in [13] for several special cases of neighborhoods, *e.g.*, parallel unit segments (ratio $3\sqrt{2}+1$) and translate regions (ratio $\sqrt{11^2+3^2}+1$). The basic idea of these algorithms is to select a representative point for each region and then apply a known TSP algorithm on these points. Dumitrescu and Mitchell [14] gave an $11.15$-approximation algorithm for neighborhoods of equal-size disks, which relates to sensors' wireless communication regions in data muling. However, the resulting tours include curves rather than simple straight lines, and visit same points and lines multiple times, which is neither easily nor efficiently applicable to the data muling problem. Recently, Elbassioni et al. [15] presented constant-ratio approximation algorithms for both disjoint (ratio $9.1\alpha+1$) and lightly intersecting (ratio $O(\alpha^3)$) convex fat regions ($\alpha=4$ for disks). While neighborhoods each containing exactly one vertex may overlap, the algorithm does not allow the existence of any vertex in the intersections. Nevertheless in data muling, it is not rare for arbitrary overlapping of sensors' neighborhoods, *e.g.*, some sensors within the communication range of multiple sensors. For general case of continuous neighborhoods with arbitrary overlapping, the best approximation ratio known so far is $O(\log n)$ [15]. Yuan et al. [16] defined a robot routing problem for data muling in WSNs as a special case of TSPN where the neighborhoods are disjoint disks. Since other TSPN algorithms only gave theoretical approximation ratios—often large and loose bounds, [16] proposed an evolutionary algorithm (EA) based solution that could often yield empirically better results. However, such solution does not address the general case of overlapping disks, and the performance of the solution depends on the proper choice of an EA algorithm.

### 2.2 Multiple Objectives Tour Planning

Similar to TSP, most tour planning problems only optimize the single objective of tour length minimization. However, real world applications usually require multiple desirable and competing objectives to be optimized at the same time. Current and Schilling [8] presented a bi-objective formulation of CSP, where a cost was associated with each tour stop, and both the tour length and the stop-over cost were to be minimized. In [17], they defined another two bi-objective problems, the median tour problem (MTP) and the maximal covering tour problem (MCTP), both with a pre-specified number $p$ out of $n$ vertices to be selected as tour stops. Both problems also have one common objective of minimization of the tour length. With respect to the second objective, MTP aims to minimize the average distance from any node not on the tour to a nearest node on the tour, and MCTP aims to maximize the number of vertices covered at tour stops within a given covering distance. Recently, [18] proposed a bi-objective version of CTP which shared similar objectives as MTP, but with no specified number of tour stops $p$. Although formulating similar bi-objectives as MCTP, UDMP does not need to specify an a priori $p$ value.

### 2.3 Uniqueness of UDMP

To sum up, UDMP differs from existing problems in four aspects. (1) Compared with CSP/CTP and bi-objective extensions, UDMP allows sensors to be covered by a data mule moving between consecutive tour stops provided that the trajectory cuts across sensors' communication regions. This is defined as *line-cover* in UDMP, which may shorten the tour length as well as improving the tour coverage. Section 6 validates these benefits by comparing heuristic UDMP tours against the optimal tours of CSP with and without line-cover. (2) In comparison to TSPN with 'continuous' neighborhood, UDMP adopts *straight* lines connecting a sequence of tour stops to form a tour, rather than curves which are hard to prescribe and difficult for a data mule to navigate in realistic scenarios. Moreover, UDMP sets no limit on the relative relationship among disk regions that represent different sensors' wireless communication ranges, which may *arbitrarily* overlap with one another. These two features make UDMP much more practical and easily applicable to realistic situations. (3) In terms of the optimization objectives, UDMP tries to optimize both tour length and tour coverage simultaneously, which differs from the majority of existing tour planning problems that consider only the single length-objective. In comparison




(a) Point-cover (b) Line-cover

Figure 1. Wireless-visit or cover.

to MCTP with similar bi-objectives, UDMP does not need to pre-designate a specific number of stops on the tour, which is too restrictive and hard to determine in advance. (4) Unlike all existing work, UDMP specially addresses the limited energy issue of a data mule in underwater scenarios by modeling and tracking energy consumptions, trading off the full coverage requirement, and enforcing energy constraint in tour planning.

## 3 Model and Assumptions

### 3.1 Types of Visit and Cover

UDMP defines two types of visit by a data mule. The first type is termed **wireless-visit** (or **cover**). Rather than traveling all the way to a sensor's exact location, a data mule only needs to move within a sensor's wireless communication range in order to collect the sensor's data wirelessly. We assume that a data mule's communication range is no less than that of the sensor, which is usually true in data muling applications. The transmission radius of a sensor thus represents the distance threshold for the sensor itself to be covered by a data mule. Wireless-visits could happen either stationarily at tour stops—called **point-cover**, or during movement between two consecutive tour stops—called **line-cover**, which may effectively increase the coverage without increasing the tour length. Figure 1(a) shows that sensor $B$ is point-covered by a data mule "at" position $A$ (as a tour stop) on (or inside) the boundary of $B$'s communication range. Figure 1(b) shows that sensor $E$ is line-covered by a data mule "with" line segment $(C, D)$ which cuts through $E$'s communication region. The second type of visit is termed **docking-visit**, where a data mule does need to visit the exact location of a node. For instance, a data mule docking-visits an energy station to be recharged.

The above types of visit and cover by a data mule are formally defined as follows. Let $D_i$ denote node $i$'s **distance threshold to be covered** by a data mule (*e.g.*, node's $i$'s wireless communication range), and $c$ denote the Euclidean distance either between two points or between a point and a line segment[1] in the Euclidean space. A node $i$ is "point-covered" by a data mule at position $j$ if the point-point distance $c(i, j) \leq D_i$, and "line-covered" with line segment $(j, k)$ if the point-line-segment distance $c(i, (j, k)) \leq D_i$. The relation of "point-cover" by a data mule between a node/position pair of $i$ and $j$ is symmetric[2] when $c(i, j) \leq \min(D_i, D_j)$. In addition, docking-visit may be regarded as a special case of point-cover, where a node's 'distance threshold to be covered' equals zero.

### 3.2 Energy Consumption Model

In UWSNs, a data mule consumes its battery energy in movement and communications [19]. For example, propulsion power consumption may range from 2 watts ($W$) for low speed ($0.2\ m/s$–$0.4\ m/s$) electric-propelled gliders to more than a hundred watts in high speed (up to $2.9\ m/s$) REMUS-class autonomous underwater vehicles. In addition, underwater acoustic communications consume much more energy than terrestrial radio communications, which makes the communication energy non-negligible even compared with propulsion. For example, an acoustic modem may use about $50W$ for packet transmission, and $0.2W$ to $2W$ for packet reception and decoding depending on the data rates. As a result, we divide the energy consumption of a data mule into two parts—$E_{move}$ for movement, and $E_{comm}$ for communications.

(1) For underwater movement, let $F_p$ be a data mule's propulsion force, $L$ be the moving distance, and $v$ be the settling speed which is a constant. Then the energy consumed for moving distance $L$ is the 'work' done by $F_p$, *i.e.*,

$$E_{move} = F_p L. \tag{1}$$

For simplicity, we ignore underwater current so that a data mule's speed relative to surrounding water remains $v$. This assumption is proper for the lakes or the deep ocean environments. Although the speed of wind-driven ocean surface current could reach $2.5\ m/s$, *e.g.*, the Gulf Stream, the deep sea current, mainly caused by density gradient from temperature and salinity, is relatively static, varying from $0.02\ m/s$ to $0.10\ m/s$ or less [20]. Assume the weight of a data mule is adjusted (like a submarine in equilibrium state vertically) so that its gravity can be counter-balanced by the buoyancy from water, which means $F_p$ only needs to counteract the water drag force $F_d$ in the reverse direction so as to reach the constant speed $v$. In addition, we assume a slow moving speed for data mule, say, less than $1\ m/s$, which is appropriate for extended missions of weeks long. By Stoke's drag equation [21], when $v$ is small, $F_d$ is linear with $v$ but opposite in direction.

$$F_d = -bv, \tag{2}$$

where $b$ is a constant dependent on properties of water and dimensions of the data mule. Combining (1) and (2), we have

$$E_{move} = bvL = \alpha L, \tag{3}$$

where $\alpha = bv$ is a constant coefficient for moving distance $L$.

(2) For underwater communications, let $P_{comm}$ be a data mule's average communication power for collecting data from

1. Note that the point-line segment distance is meaningful only if the point can be orthogonally projected onto the line segment—otherwise the distance is assigned to be $\infty$.

2. In short, '$i$ and $j$ point-cover each other' means that node $i$ (at position $i$) is point-covered by a data mule at position $j$, and node $j$ (at position $j$) is point-covered by a data mule at position $i$. For simplicity, in the remainder of this paper we will no longer differentiate among node, point, vertex, and its position, which may be used interchangeably within appropriate contexts. Note that the data mule is the *sole* agent for all visits/covers.

sensors, which is a weighted sum of both transmission and receiving/decoding power of the data mule depending on the data retrieving scheme. For example, a data mule could sequentially poll each nearby sensor within the sensor's communication range by first sending a small request message and then receiving data from the polled sensor. For simplicity, we assume a uniform amount of data, denoted by $a$, to be exchanged at every sensor[3]. Let $N$ be the data transfer rate, and $C$ be the number of visited sensors including both point-cover and line-cover (termed **coverage**). We then have

$$E_{comm} = P_{comm}(\frac{a}{N})C = \beta C, \tag{4}$$

where $\beta = P_{comm}(\frac{a}{N})$ is a constant coefficient for $C$.

### 3.3 Energy Consumption within a Tour Segment

In UDMP, a data mule needs to visit a docking station to get recharged before running out of battery energy. Specifically, the data mule's energy usage is tracked within each **tour segment** (or **docking segment**), which is the sub-tour between two consecutive docking-visits. A tour segment starts at a docking station, passes a set of non-docking tour stops, and terminates at a next docking station. Note that the depot is also regarded as a docking station. The straight line segment traversed by a data mule between any two consecutive tour stops is termed a **tour link**. Therefore within each tour segment, the data mule operating on its finite battery energy both moves along tour links and collects data from sensors before recharging itself at the next docking station on the segment border. Let $e(S)$ denote a data mule's energy consumption in a tour segment $S$, $L_S$ be the sum of tour link length within segment $S$, and $C_S$ be the coverage inside the segment (excluding those sensors point-covered at the two bordering docking stations). Based on (3) and (4), we have

$$e(S) = \alpha L_S + \beta C_S, \tag{5}$$

which is linear to the segment's length and coverage.

## 4 Problem Formulation

We formulate the energy-constrained bi-objective UDMP as follows. Let $G = (V, E)$ be a complete graph in an Euclidean space (*e.g.*, $3D$), with the vertex set $V = \{d\} \cup Y \cup Z$, where $d$ is the sink (or depot), $Y$ is a set of docking stations that *may* be **docking-visited**, with $Y_1 \subseteq Y$ as the subset that *must* be docking-visited ($d \in Y_1$), and $Z$ denotes a set of sensors that *may* be **wireless-visited** within the wireless communication range $D_i$ of each sensor $i \in Z$, with $Z_1 \subseteq Z$ as the subset that *must* be wireless-visited. Let the non-negative cost function $c(i, j)$ be the Euclidean distance between any two points $i$ and $j$ in the same Euclidean space where $G$ resides. Let $e_0$ be the battery capacity of a data mule which is fully supplied at the depot and can later be recharged at a docking station.

3. Uniform or low-variance in the amount of collected data among sensors could be achieved through low sampling rate or in-sensor historical data fusion/compression.

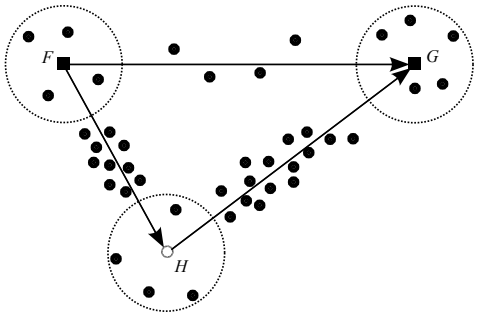



Figure 2. Tradeoff between length and and cover objectives. The shorter route $\langle F, G\rangle$ covers fewer sensors, while the longer route $\langle F, H, G\rangle$ covers more sensors.

### 4.1 UDMP

The goal of UDMP is to find a tour $T = \langle t_1, t_2, \cdots, t_{|T|}\rangle$, which is an ordered list, for the data mule such that:

1) Each tour stop $t_i \in T$ is different, with the depot as both the start and the end of the tour, *i.e.*, $t_1 = t_{(|T|+1)} = d$;
2) A tour stop could be at either the location of a docking-visited station or any position to cover a wireless-visited sensor, *i.e.*, $\forall t_i \in T$, either $t_i \in Y$ or $\exists z_k \in Z$ such that $t_i$ can point-cover $z_k$ within $z_k$'s communication range $D_k$; Based on those stops at docking stations, tour $T$ could be further divided into a consecutive sequence of tour segments $S_1, S_2, \cdots$, whose *borders* are marked by docking-visited stations;
3) All vertices in $Y_1$ are docking-visited during the tour $T$, *i.e.*, $\forall y_j \in Y_1$, $y_j \in T$;
4) All vertices in $Z_1$ are wireless-visited during the tour $T$, *i.e.*, $\forall z_k \in Z_1$, $\exists t_i \in T$ such that either $z_k$ is point-covered at $t_i$, or line-covered with $(t_i, t_{i+1})$—the line segment between the two adjacent tour stops $t_i$ and $t_{i+1}$;
5) The data mule's energy usage in any tour segment $S_i$ satisfies the upper bound $e_0$, *i.e.*, $e(S_i) \leq e_0$, based on Equation (5)—energy constraint;
6) At the same time optimize the following two objectives: (i) the maximization of $C_T$ as the number of nodes in set $Z$ wireless-visited by the tour (cover-objective), and (ii) the minimization of the tour length $L_T = \sum_{i=1}^{|T|} c(t_i, t_{i+1})$ (length-objective).

In the above formulation, Item 1 defines the concept of a tour with *distinct* stops, which may be relaxed in the generalized UDMP-DoD case later in this section. Item 2 specifies the selection of tour stops as well as the division of a tour into tour segments. Items 3 and 4 further qualify the tour with the sets of nodes which must be docking- or wireless-visited. Item 5 enforces the data mule's energy constraint for all tour segments. Finally, Item 6 states the cover-objective and the length-objective for tour optimizations, which are competing against each other as illustrated in Figure 2, and have to be compromised.

*Theorem 1:* UDMP is NP-hard.

*Proof:* Since TSP is NP-complete, we will show that UDMP is polynomial-time reducible from TSP. Given an instance of TSP, which includes a complete graph $G = (V, E)$

and a cost function $c$ mapping each edge in $E$ to the Euclidean distance between two endpoints. We construct an instance of UDMP as follows. We define a complete graph $G' = (\{d\} \cup Y \cup Z, E)$, where $d$ is a vertex randomly picked from $V$ as the depot, $Y = V$, and $Z = V \setminus \{d\}$ with each vertex in $Z$ associated with a non-negative 'distance threshold to be covered'. Apparently, $G'$ shares the same vertex set and edge set as $G$. In addition, let $Y_1 = Y$ so that all vertices in $Y$ must be docking-visited. Therefore every vertex in $G'$ must be on a feasible tour of the UDMP instance, which means all vertices in $Z$ must be wireless-visited—the maximum coverage. We also define the cost function $c'$ to be the Euclidean distance between any two points, and the segment energy constraint $e_0 = \infty$. Apparently, there exists a minimum length TSP tour in $G$ *iff* the same tour in $G'$ is the UDMP tour with the minimum length for the maximum coverage and satisfied energy constraint. This reduction can be done in polynomial time. Hence UDMP is NP-hard. □

### 4.2 UDMP-AD (All-Docking)

One special case of UDMP, termed UDMP-AD (All-Docking), is defined when all docking stations must be visited (*i.e.*, $Y_1 = Y$). UDMP-AD is also NP-hard as the above proof can be straightforwardly applied.

We also define a non-energy-constrained single length-objective version of UDMP-AD, which optimizes only the length-objective while covering all sensors (*i.e.*, $Z_1 = Z$) with no energy constraint (*i.e.*, $e_0 = \infty$). We term this problem the *Covering Salesman Problem with Line-Cover* (CSP-LC) to differentiate it from CSP that does not utilize line-cover. In Section 6, we will evaluate the goodness of heuristic UDMP-AD tours by comparing with the optimal tours of CSP-LC and CSP, respectively. NP-hard as CSP, CSP-LC is formulated as an *Integer Linear Programming* (ILP) problem as follows.

Given a complete graph $G = (V, E)$, with $V = \{d\} \cup Y \cup Z$, $Y_1 = Y$, and $Z_1 = Z$. Let $c_{ij}$ be a non-negative cost associated with each edge $(i, j) \in E$. The objective of CSP-LC is to

$$\text{minimize} \sum_{i,j \in V, i \neq j} c_{ij} x_{ij}, \tag{6}$$

subject to

$$\sum_{j \in V} P_{ij} I_j + \sum_{(k,l) \in E} L_{i,(k,l)} x_{kl} \geq 1, \forall i \in Z, \tag{7}$$

$$I_k = 1, \forall k \in Y, \tag{8}$$

$$\sum_{i \in V, i \neq k} x_{ik} = I_k, \forall k \in V, \tag{9}$$

$$\sum_{j \in V, j \neq k} x_{kj} = I_k, \forall k \in V, \tag{10}$$

$$y_{ij} \leq (|V| - 1) x_{ij}, \forall i, j \in V, i \neq j, \tag{11}$$

$$\sum_{k \in V, k \neq j} y_{jk} - \sum_{i \in V, i \neq j} y_{ij} = I_j, \forall j \in V \setminus \{d\}, \tag{12}$$

$$\sum_{i \in V, i \neq d} y_{id} - \sum_{j \in V, j \neq d} y_{dj} = \sum_{k \in V \setminus \{d\}} I_k, \tag{13}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ is on the tour,} \\ 0 & \text{otherwise;} \end{cases}$$

$$I_k = \begin{cases} 1 & \text{if } k \in V \text{ is a tour stop,} \\ 0 & \text{otherwise;} \end{cases}$$

$y_{ij} \geq 0$ : the integer flow from $i$ to $j$ via $(i, j) \in E$;

$c_{ij} \geq 0$ : the cost from $i$ to $j$ via $(i, j) \in E$;

$$P_{ij} = \begin{cases} 1 & \text{if } i \in V \text{ is point-covered by } j \in V, \\ 0 & \text{otherwise;} \end{cases}$$

$$L_{i,(j,k)} = \begin{cases} 1 & \text{if } i \in V \text{ is line-covered by } (j,k) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

In the objective function (6), the sole optimization objective is to minimize the total tour cost, where the binary variable $x_{ij}$ determines whether or not the tour passes through the edge (or line segment) $(i, j) \in E$, and the coefficient parameter $c_{ij}$ denotes the travel cost over $(i, j)$ (*e.g.*, the Euclidean distance between $i$ and $j$). Constraint set (7) enforces all sensors to be covered by the tour (*i.e.*, $Z = Z_1$), where the binary variable $I_j$ determines whether node $j$ is on the tour, and binary coefficients $P_{ij}$ and $L_{i,(k,l)}$ indicate whether sensor $i$ is point-covered by a data mule at point $j$ and whether sensor $i$ is line-covered by a data mule with line segment $(k, l)$, respectively. Constraint set (8) mandates that all the docking stations be on the tour (*i.e.*, $Y = Y_1$). Next, constraint sets (9)–(13) ensure the validity of the tour based on characteristics of a Hamiltonian cycle. In particular, (9) and (10) enforce the degree restriction for tour stops such that the tour should only enter a stop via one single edge and leave the stop via another single edge, which prohibits revisits of tour stops. Constraint sets (11)–(13) further eliminate cases of sub-tours (*i.e.*, tours with disconnected components) by introducing the integer decision variable $y_{ij}$ denoting, say, the unit of collected data flowing through edge $(i, j)$. Let the data mule start out at depot $d$ with zero data, and collect only one unit of data at each tour stop. Constraint set (11) enforces the upper limit on data flows and confines them to only edges on the tour. Constraint set (12) ensures that only one additional unit of data is added to the flow at each tour stop except for the depot. Finally, constraint (13) specifies that the net data flow into depot $d$ should be equal to the number of all the other tour stops, signifying $d$ as both the start and the end of a tour which passes through all stops, and thus prohibiting sub-tours.

### 4.3 UDMP-DoD (Docking-on-Demand)

In comparison to UDMP-AD where *all* the docking stations must be docking-visited, we study the general scenarios where docking stations are *selectively* docking-visited based on the actual charging need. In contrast to UDMP-AD that assumes $Y_1 = Y$, we term this general case (without the assumption of $Y_1 = Y$) UDMP-DoD (Docking-on-Demand). In practice, UDMP-DoD could further decrease tour length by avoiding unnecessary docking-visits in UDMP-AD as illustrated in Figure 3. In general, UDMP-DoD tours are shorter than UDMP-AD tours in scenarios which lack an optimal deployment of docking stations, as will be verified in Section 6.2.

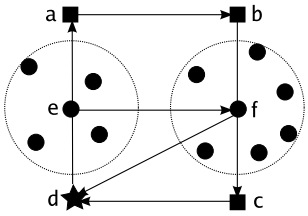
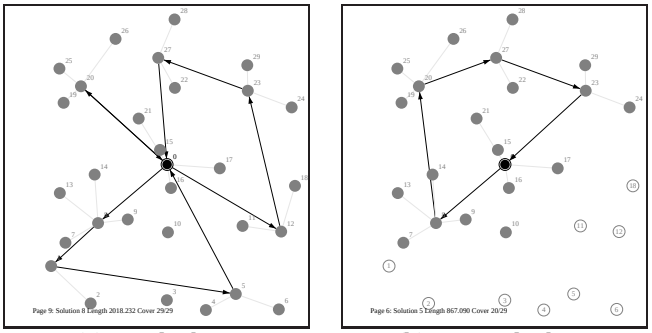



Figure 3. Comparison between UDMP-AD and UDMP-DoD tours to cover all sensors in a sample network, with depot $d$ and $Y = \{a, b, c, d\}$. (1) AD tour: $\langle d, a, b, c, d\rangle$ as $Y = Y_1$; (2) DoD tours: $\langle d, e, f, d\rangle$ or $\langle d, e, f, c, d\rangle$—depending on whether the data mule needs to be recharged—both being shorter than the AD tour.



(a) Re-docking (b) No re-docking

Figure 4. Comparison between UDMP tours (a) with and (b) without re-docking: (a) covers all, while (b) does not.

Moreover, UDMP-DoD generalizes UDMP by relaxing the tour requirement of "distinct stops" in Item 1 of the UDMP formulation (Section 4.1). Note that UDMP-DoD allows only re-visits of docking stations (termed *re-docking*), while still maintaining the uniqueness of all the other tour stops at the same time. In UDMP-DoD, each tour stop $t_i \in T$ is different *except for* those that are docking stations (*i.e.*, $t_i \in Y$, including the depot $d$). This extended feature of re-docking in UDMP-DoD accommodates circumstances where insufficient number of docking stations are deployed in some region(s) of the network, such that the data mule has to re-visit some docking station(s) for re-charging in order to collect data from as many sensors as possible in a tour. For example, when no docking station other than the depot exists (*i.e.*, $Y = \{d\}$ as the central black node shown in Figure 4), the data mule may re-visit the depot repetitively to renew battery energy. Consequently, the data mule could cover all sensors in one tour, as illustrated in Figure 4(a). However, without re-docking the mule would have to complete the tour with only a subset of the sensors visited (shown as gray nodes in Figure 4(b)). UDMP-DoD is also NP-hard, as the proof for Theorem 1 can be straightforwardly applied.

## 5 Heuristic Tour Planning Algorithms

In general, the solution to a multi-objective optimization problem is not a single instance, but a set of *Pareto-efficient* ones that represent the best compromise among all objectives in different tradeoff situations, *i.e.*, none of the objectives can be further improved without the degradation of other objective(s). All Pareto-efficient solutions to a multi-objective optimization problem form the *Pareto frontier*. Formally, a solution $T_1$ *dominates* another solution $T_2$ *iff* $T_1$ is at least as good as $T_2$ with respect to all objectives, and there exists at least one objective where $T_1$ is strictly better than $T_2$ with respect to that objective. A solution is Pareto-efficient if it is *not* dominated by any other solution.

Given the NP-hardness of UDMP, we propose two greedy heuristic algorithms to calculate the Pareto frontier for UDMP-AD and UDMP-DoD, respectively. For simplicity, we assume non-prioritized coverage requirement among sensors (*i.e.*, $Z_1 = \emptyset$)[4]. Both algorithms share the same basic framework which iteratively (1) adds a tour stop to cover sensors, and (2) applies a known Euclidean TSP algorithm on the tour stops to compute a shortest tour, until all the sensors to be covered are covered by the tour. We define the *neighbor set* of a covering position as the set of sensors point-covered at this position minus those point-covered at existent tour stops other than this position. The selection of a covering position as a newly added tour stop depends on the 'weight' of its neighbor set, to be detailed in the algorithms. Note that both algorithms are applicable to $3D$ and any dimensional Euclidean space, because they (1) deal with *logical* sets that are not necessarily formed based on geometric relationship in any dimension, and (2) use the Euclidean distance which is dimension-independent. The key differences between UDMP-AD and UDMP-DoD heuristics are manipulation of docking segments to enforce the energy constraint. The UDMP-AD heuristic maintains a fixed number (equal to the size of $Y$) of docking segments, and tries to adjust segment borders in case of energy constraint violation. In contrast, the UDMP-DoD heuristic keeps a variable number of docking segments with the selection of docking-visits based on the actual energy need. The two algorithms are described in detail as follows.

### 5.1 Heuristic Algorithm for UDMP-AD

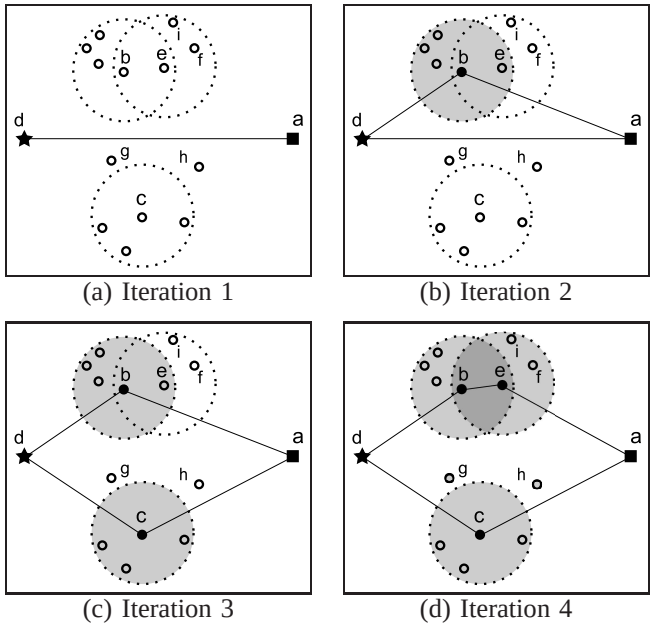
Input: Docking station set $Y = Y_1$, sensor set $Z$, wireless communication range $D_i$ for each sensor $i \in Z$, energy constraint $e_0$ for each tour segment.

Output: The set $\Pi$ of Pareto-efficient tours that satisfy the segment energy constraint with both tour length and tour coverage objectives optimized.

Step 1: Initialize the tour stop set $R = Y_1$, the candidate tour stop set $H = Z$, and the candidate Pareto-efficient solution set $\Pi = \{TSP(R)\}$, where $TSP(R)$ is the TSP tour over $R$ calculated by a known TSP heuristic algorithm. Update $Z$ by removing those point-covered at vertices in $R$. If $Z$ is empty or each vertex in $Z$ can be line-covered with some edge in $TSP(R)$, return $\Pi$ and terminate.

Step 2: For each vertex $i \in H$, calculate its neighbor set $nSet(i) = \{j \mid c(j,i) \le D_j, \forall j \in Z\}$, and its weight $w_i = |nSet(i)|$. Remove from $H$ any vertex with zero weight. Initialize the tabu set $B = \emptyset$.

4. It is trivial to solve cases with $Z_1 \neq \emptyset$ by prioritizing the covering order of $Z$ and applying our algorithms twice—first on $Z_1$, and second on $Z \setminus Z_1$.




(a) Iteration 1 (b) Iteration 2

(c) Iteration 3 (d) Iteration 4

Figure 5. An example execution of UDMP-AD algorithm.

Step 3: If $H \setminus B$ is empty, return $\Pi$ and terminate. Otherwise find the vertex $k = \text{argmax}_{i \in (H \setminus B)} w_i$—select the one with the shortest average distance to all vertices in $R$ in case of a tie, and add vertex $k$ into $R$.

Step 4: Construct *TSP*$(R)$ and compute its length and coverage, counting both point-coverage and line-coverage. Check each segment's energy consumption (based on Equation (5)) against the energy constraint $e_0$, and try 'border adjustment' to correct violations if any. If all the segments are able to satisfy $e_0$, set $H = H \setminus \{k\}$ and $\Pi = \Pi \cup \{$*TSP*$(R)\}$, check dominance between the new tour *TSP*$(R)$ and previous solutions in $\Pi$ so as to remove whomever dominated, and go to Step 5. Otherwise, discard *TSP*$(R)$, set $R = R \setminus \{k\}$ and $B = B \cup \{k\}$, and go to Step 3.

Step 5: Update $Z$ by removing those point-covered at stop $k$. If $Z$ is empty or each vertex in $Z$ can be line-covered with some edge in *TSP*$(R)$, return $\Pi$ and terminate. Otherwise, for those in $H$ whose neighbor sets overlap $k$'s neighbor set, update their neighbor sets as well as weights, remove vertices with zero weights from $H$, reset $B = \emptyset$, and go to Step 3.

The algorithm is illustrated via an example shown in Figure 5, where $Y_1 = Y$ includes depot $d$ and docking station $a$, both of which must be docking-visited, and $Z$ includes 13 non-prioritized sensors to be covered as many as possible with a uniform 'distance threshold to be covered' shown by the equal-radius dotted circles. For simplicity, energy constraint $e_0$ is assumed to be large enough so that the energy consumption of each computed tour segment during the UDMP-AD algorithm execution satisfies the energy constraint—we will demonstrate 'border adjustment' in case of energy constraint violation later. This example works in four iterations, each producing one tour. Figure 5(a) shows that the initial tour $\langle d, a, d \rangle$ consists of only depot $d$ and docking station $a$, with the tour link $(d, a)$

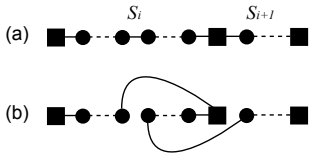



Figure 6. Connectivity (a) before and (b) after successful 'border adjustment' between segment $S_i$ and its right segment $S_{i+1}$ in order to correct $e_0$ violation in $S_i$.

line-covering two sensors $g$ and $h$ (Step 1). Since no sensor is point-covered by the initial tour yet, $Z$ equals the candidate stop set $H$ which contains all 13 sensors. Next, a weight for every candidate stop is calculated to be the size of its neighbor set, which includes sensors in $Z$ point-covered at the candidate stop, as shown inside the circle centered at a candidate stop (Step 2), *e.g.*, $w_b = 5$, $w_e = w_c = 4$. Sensor $b$ is chosen with the largest weight of 5 (Step 3), and the second tour $\langle d, b, a, d \rangle$ is produced (Step 4) as depicted in Figure 5(b). Assume that both segments $\langle d, b, a \rangle$ and $\langle a, d \rangle$ satisfy the energy constraint. Then those sensors in the new tour stop $b$'s neighbor set, which have been *permanently* covered at tour stop $b$ (via point-cover), are removed from $Z$, as highlighted by the shadow area. Notice that sensors line-covered in the current tour (such as $g$ and $h$) may not still be covered in the newly computed tour due to the change of tour links. Therefore these sensors are only *temporarily* covered and should not be removed from $Z$. The neighbor sets of the other candidate stops in $H$ are also updated so that the new neighbor sets contain only sensors in $Z$, and the candidate stops with empty neighbor sets are removed from $H$ (Step 5). Notice that some sensors already covered and deleted from $Z$ can still be candidate tour stops in $H$, as long as their neighbor sets are not empty, *e.g.*, sensor $e$ with a weight of 2. In the third iteration, sensor $c$ is selected with the largest weight of 4, to produce another tour $\langle d, b, a, c, d \rangle$ shown in Figure 5(c), with both segments $\langle d, b, a \rangle$ and $\langle a, c, d \rangle$ satisfying the energy constraint. Now, sensors $g$ and $h$ are line-covered with tour links $(c, d)$ and $(a, c)$, respectively, rather than with $(d, a)$ which no longer belongs to the current tour. After sensor $c$'s neighbor set is removed from $Z$, the neighbor sets of candidate stops in $H$ are updated as before. In the fourth iteration, there is a tie in the largest weight of 2 among the candidate tour stops $e$, $f$, and $i$. Then the distances between each candidate and the current tour stop set are compared, and sensor $e$ is selected as the closet one into the tour stop set (Step 3). Therefore the fourth tour $\langle d, b, e, a, c, d \rangle$ is produced as shown in Figure 5(d), with both segments $\langle d, b, e, a \rangle$ and $\langle a, c, d \rangle$ satisfying the energy constraint. Since the updated set $Z$ only contains two sensors $g$ and $h$, which can be line-covered with current tour links $(c, d)$ and $(a, c)$, respectively, the algorithm terminates and returns these four tours as heuristic Pareto-efficient solutions.

When some segment in a computed TSP tour violates the energy constraint, 'border adjustment' is used to correct such violation (Step 4), if possible, as illustrated in Figure 6, where squares denote docking stations as segment boundaries, circles represent tour stops within segments, and edges (direct solid lines or curves) reflect only the connectivity relationship rather

than the real shape of tour links. Let $S_i$ be a segment with a negative battery energy balance, *i.e.*, $e_0 - e(S_i) < 0$. By Equation (5), the segment energy consumption is proportional to both the length and the coverage of the segment, both of which decrease with fewer tour stops in the segment. Therefore, in order to lower $e(S_i)$, 'border adjustment' tries to shift out a minimum number of close-to-boundary tour stops from segment $S_i$ into an adjacent segment—either $S_{i+1}$ in the right or $S_{i-1}$ in the left, whichever has a positive energy balance, so that both $S_i$ and the adjacent shift-in segment could satisfy the energy constraint. Notice that the boundary tour stops of the segments are always docking stations, which stay unchanged during the adjustment. Since $e(S_i)$ is decreased at the cost of increasing $e(S_{i+1})$ or $e(S_{i-1})$, 'border adjustment' can not solve the energy violation problem of segment $S_i$ when $S_i$'s two adjacent segments deplete their respective energy before digesting enough shift-in tour stops to eliminate $S_i$'s energy deficiency. In such case, the tour fails the energy constraint and is simply discarded. Another side-effect of 'border adjustment' is the increase of the total tour length due to the detour cross from edge exchange, as will be evidenced in Figure 8(c) of Section 6.1. Therefore, 'border adjustment' is only applied between adjacent segments without propagating to farther segments.

We now analyze the worst-case computational complexity of the algorithm. Suppose there are $n$ sensors and $m$ docking stations ($n \gg m$). Therefore, $n = |Z|$ and $m = |Y|$. For the space complexity, it takes $\sum_{i=0}^{n}(m+i) = O(nm + n^2)$ space to store all candidate Pareto-efficient solutions, $O(n^2)$ for neighbor sets of all the sensors as well as the point-point distances, and at most $O(n^3)$ to buffer calculated line-coverage results of tour links in Step 4. Hence the overall space complexity of the algorithm is $O(n^3)$.

For the time complexity, the algorithm takes $O(nm)$ time to calculate the point-coverage of $m$ docking stations as well as the line-coverage of $m$ edges in Step 1, and $O(n^2)$ time to compute the neighbor sets and weights for $n$ sensors in Step 2. For Step 3 through Step 5 which form two nesting loops, we count each step's overall running time by combining all possible repetitions. In the worst case, each addition of a new tour stop has to try every candidate stop due to repeated failure to satisfy the energy constraint (*i.e.*, from Step 4 going back to Step 3 repeatedly for a maximal number of times before going forward to Step 5). Step 3 takes $O(n^2 \log n)$ time to find the maximum weight for every trial by sorting all the candidate stops at the start of each iteration, and at most $O(nm + n^2)$ time to select the candidate closest to the set of current tour stops to break ties, in the worst case that all the candidate stops share the same weight, *e.g.*, all have the weight of one with non-overlapping neighbor sets. Step 4 takes $O(n^2)$ time to incrementally compute the length of all the tour segments, $O(n^2)$ time to add up the point-coverage of tour stops, $O(n^3)$ time to calculate and add up the line-coverage of possible tour links, $O(n^2 m)$ time to check the segment energy constraint, at most $O(n^3)$ time to adjust segment borders in case of energy constraint violations, and $O(n^2)$ time to check dominance between the new solution and the existent solutions. Step 5 takes $O(n^2)$ time to remove sensors point-covered at every new tour stop from the set of uncovered sensors, $O(n)$ to check the line-coverage result of every new tour computed in Step 4 for termination condition test, and $O(n^3)$ to update the neighbor sets that overlap with each new tour stop. Finally, let $T(k)$ be the running time of the heuristic TSP algorithm over $k$ vertices employed in the proposed UDMP algorithm. Then the running time of TSP in Step 1 is $T(m)$, and the worst-case running time in Step 4 is at most $\sum_{i=1}^{n}(n-i+1)T(m+i)$. Therefore the overall time complexity of the algorithm in the worst case is $O(n^3) + T(m) + \sum_{i=1}^{n}(n-i+1)T(m+i)$.

### 5.2 Heuristic Algorithm for UDMP-DoD

Input: Docking station set $Y \supseteq Y_1$, sensor set $Z$, wireless communication range $D_i$ for each sensor $i \in Z$, energy constraint $e_0$ for each tour segment.

Output: The set $\Pi$ of Pareto-efficient tours that satisfy the segment energy constraint with both tour length and tour coverage objectives optimized.

Step 1: Initialize the tour stop set $R = Y_1$, the candidate tour stop set $H = Z$, and the candidate Pareto-efficient solution set $\Pi = \{\mathit{TSP}(R)\}$. Update $Z$ by removing those point-covered at vertices in $R$. If $Z$ is empty or each vertex in $Z$ can be line-covered with some edge in $\mathit{TSP}(R)$, return $\Pi$ and terminate.

Step 2: For each vertex $i \in H$, calculate its neighbor set $\mathit{nSet}(i) = \{j \mid c(j,i) \le D_j, \forall j \in Z\}$, and its weight $w_i = |\mathit{nSet}(i)|$. Remove from $H$ any vertex with zero weight. Initialize the tabu set $B = \emptyset$.

Step 3: If $H \backslash B$ is empty, return $\Pi$ and terminate. Otherwise find the vertex $k = \mathrm{argmax}_{i \in (H \backslash B)}\, w_i$—select the one with the shortest average distance to all vertices in $R$ in case of a tie, and add $k$ into $R$.

Step 4: Construct the current tour $\pi = \mathit{TSP}(R)$, and compute its length and coverage, counting both point-coverage and line-coverage. Starting at the depot with full battery energy $e_0$, track the remaining energy $E_r$ along $\pi$, and use 'docking on-demand' to recharge at a selected docking station (added as a tour stop) whenever $E_r < 0$. If the depot is finally reached with $E_r \ge 0$ after passing all tour stops and links in $\pi$, set $H = H \setminus \{k\}$ and $\Pi = \Pi \cup \{\pi\}$, check dominance between $\pi$ and the previous solutions in $\Pi$ so as to remove whomever dominated, and go to Step 5. Otherwise (*i.e.*, without enough energy to return to the depot), discard $\pi$, set $R = R \setminus \{k\}$ and $B = B \cup \{k\}$, and go to Step 3.

Step 5: Update $Z$ by removing those point-covered at tour stop $k$. If $Z$ is empty, or each vertex in $Z$ can be either point-covered at a newly inserted docking station or line-covered with an edge in the current tour $\pi$, return $\Pi$ and terminate. Otherwise, for those in $H$ whose neighbor sets overlap $k$'s neighbor set, update their neighbor sets as well as weights, remove those vertices with zero weights from $H$, reset $B = \emptyset$, and go to Step 3.

In UDMP-DoD, the number of docking segments varies and the selection of docking-visits depends on the actual


(a) Mature docking (b) Pre-mature docking

Figure 7. Two cases of 'docking on-demand' with $(i-1)$ as the on-demand stop and $y$ as the selected on-demand docking station. In (a), $y$ is visited immediately after the on-demand stop. In (b), backtrack to preceding tour stops when no $y$ is reachable from stop $(i-1)$, until finding $y$ to be visited between stops $(i-j)$ and $(i-j+1)$.

energy (recharging) need. This is achieved by incrementally building tours to cover sensors without considering the energy constraint at first (Steps 1, 2, 3, and 5), similar to the example in Figure 5. Meanwhile at each iteration, starting from the depot the algorithm keeps track of mule's the energy remainder $E_r(i)$ at each tour stop $i$ (indexed by its sequential order in a segment) along the tour. Specifically, the data mule's battery starts with full capacity $e_0$ at the beginning of a tour segment (a docking station), and is consumed within the tour segment over links (with communication energy from line-cover plus movement energy) and at stops (with communication energy from point-cover), based on Eq (5).

In case of $e_0$ violation (*i.e.*, $E_r < 0$), 'docking on-demand' is used to select a docking station for recharging (Step 4, illustrated in Figure 7) in three steps. First, locate the *on-demand stop* as the *last* stop that a data mule may visit with enough energy, *e.g.*, stop $(i-1)$ in Figure 7, with $E_r(i-1) \geq 0$ and $E_r(i) < 0$—without recharging, the data mule could *not* reach the next stop $i$. Second, try to find an *on-demand docking station* $y \in Y$ which is both reachable with the remaining energy from on-demand stop $(i-1)$ and incurring minimal increase in tour length, termed **mature docking** (Figure 7(a), with the dotted circle surrounding the tour stop representing the search scope based on $E_r$ at the stop). If mature docking is not available from $(i-1)$, *backtrack* to the preceding stops one by one in the same segment until finding an on-demand docking station $y$, termed **pre-mature docking** (Figure 7(b)). Apparently, the search scope at each further backtracked stop increases since $E_r$ increases while 'moving backward.' Mature and pre-mature dockings also differ in terms of $E_r$ at the stop from which an on-demand docking station $y$ is selected—mature only with the minimal $E_r > 0$ corresponding to an on-demand stop. Note that the segment length might increase more with mature docking (immediately after on-demand stop $(i-1)$) than with pre-mature docking (after some backtracked stop $(i-j)$), as compared between Figures 7(a) and 7(b). Notwithstanding the possibility of a locally larger increase in tour length, this greedy heuristic always chooses mature docking whenever possible, because by postponing recharging as late as possible, the heuristic may reduce the total number of recharging in the tour, which would potentially lower the overall tour length. Third, the newly selected on-demand docking station $y$ is inserted into the tour, and the original tour segment splits into two with the new segment before $y$ having already passed the energy constraint check and the new segment after $y$ to be examined next. Notice that the candidate pool for on-demand docking stations remains unchanged as $Y$ which permits re-docking. 'Docking on-demand', together with re-docking, eliminates the need for 'border adjustment' (used in UDMP-AD algorithm) to enforce the energy constraint. Since on-demand docking stations are selected based on the locations of on-demand stops in a specific tour, the selected on-demand docking stations may vary with different tours between iterations. Therefore, sensors point-covered by the mule at the on-demand docking stations in one iteration may not be covered any more in other iterations due to the possible changes of on-demand docking stations. Therefore, these sensors are treated as *temporarily* covered (like line-coverage) in Step 5.

For the worst-case computational complexity, the UDMP-DoD algorithm is comparable to the UDMP-AD algorithm. The overall space complexity of UDMP-DoD is $O(n^3)$ due to the same line-coverage buffer requirement as in UDMP-AD. For the time complexity, we just examine the on-demand docking in Step 4. There are at most $n$ iterations, with each iteration adding at most $n$ on-demand docking stations considering re-docking, each successful on-demand docking taking at most $n$ backtracks of tour stops, and each tour stop testing exactly $m$ candidate docking stations. Therefore, the on-demand docking takes at most $O(n^3 m)$ time.

## 6 Performance Evaluations

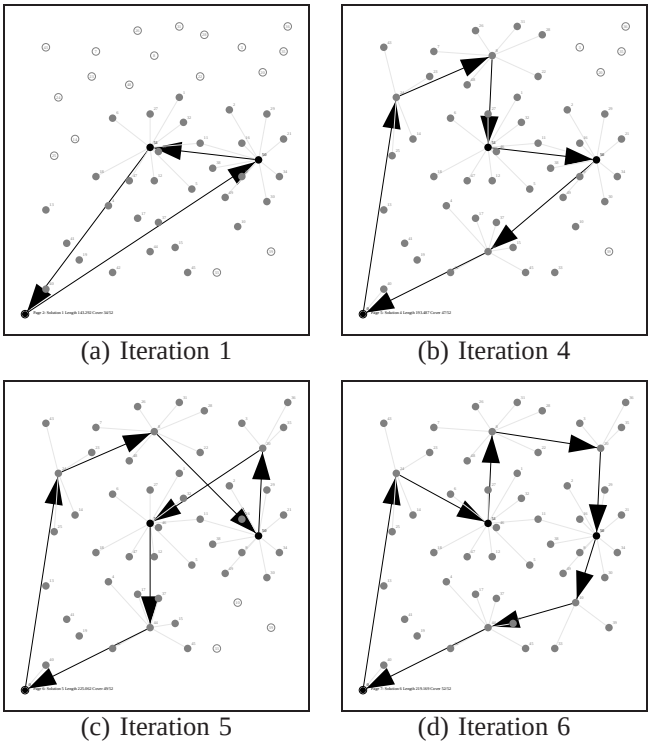
We have conducted extensive simulation to validate the effectiveness of the proposed UDMP-AD and UDMP-DoD algorithms. For the TSP subroutine employed in both algorithms, we use Heldsgaun's implementation of the Lin-Kernighan TSP heuristic [22], [23], considered to be one of the best heuristics for Euclidean TSP. In all simulation, we assume a uniform communication range $r$ for each sensor. The network topologies are either adapted from the test instances in TSPLIB [24], or generated with random distribution.

### 6.1 Results of UDMP-AD

We vary the communication range, the energy constraint, and the network size[5] in the simulation, and present tour snapshots pictorially and tour statistics quantitatively.

Figure 8 shows the snapshots of four tours out of totally six tours generated for a network topology adapted from the TSPLIB instance "$eil51$." The "$eil51$" instance consists of $51$ vertices on a terrain of size $70 \times 70$ $m^2$. We add one more vertex at $(0,0)$ (the lower left corner) as the depot, and randomly pick two other vertices as docking stations. The three vertices (shown in black) form the set $Y_1 = Y$ which must be docking-visited (as the initial tour stops in Figure 8(a)), while all the remaining vertices compose the sensor set $Z$. We set $r = 15m$ as the uniform communication range for all sensors, which is selected to fit the actual terrain size in the "$eil51$" instance, and could certainly be scaled (together with the terrain size and the nodes' coordinates) to

5. In this paper, network size refers to the total number of nodes in the network, which is not to be confused with the terrain size of the network.



(a) Iteration 1 (b) Iteration 4

(c) Iteration 5 (d) Iteration 6

Figure 8. Snapshots with coverage ratio $C$ and tour length $L$ of selected iterative UDMP-AD tours for adapted "$eil51$" with $e_0 = 20$. (a) $C = 31/49, L = 143.292m$; (b) $C = 44/49, L = 193.487m$; (c) $C = 46/49, L = 225.062m$, with segment border adjustment to satisfy $e_0$; (d) $C = 49/49, L = 219.169m$.

match the realistic underwater acoustic transmission range—typical in the hundreds of meters up to kilometers. We also enforce a smaller $e_0 = 20$ in order to demonstrate the impact of energy constraint with the effect of 'border adjustment.' Figure 8(a) depicts the initial tour constructed with all three docking stations in $Y$, which become the boundaries of three tour segments. The length of tour segments increases with newly added tour stops in subsequently generated tours, while the boundary docking stations remain unchanged. Notice that the tour 'direction' denoted by the arrows is only *nominal*, as it makes no difference between two opposite directions in a cycle with regard to both objectives as well as energy constraint. After initialization, the initial tour extends by one tour stop at each subsequent iteration to cover more sensors. Figure 8(b) shows the tour at the fourth iteration, with three more tour stops added and thirteen more sensors covered than Iteration 1. So far the energy constraint is satisfied in all three tour segments. Figure 8(c) depicts the tour at Iteration 5 after successfully adjusting the segment border to correct one energy constraint violation. The tour stop at the top right is shifted to the next tour segment so that both segments' energy consumption is less than $e_0$. As a result, the tour length is increased to $225.062m$ due to the cross detour formed at the top right portion of the tour. Figure 8(d) shows the tour at Iteration 6, which covers all 49 sensors with a total length of $219.169m$. Notice that although one more tour stop is added,

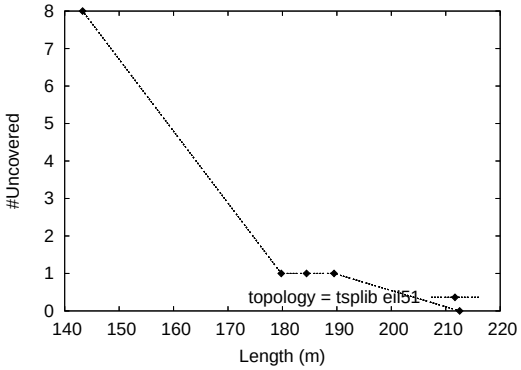



Figure 9. Cover vs. length objectives in UDMP-AD solutions for adapted "eil51" with $e_0 = 100$.

the computed TSP tour satisfies the energy constraint without a need for any 'border adjustment.' Moreover, we find that this tour dominates the previous one with three more sensors added in coverage and shorter length. The algorithm then terminates, and returns five tours as Pareto-efficient solutions, except for the dominated one from Iteration 5.

Figure 9 depicts the tradeoff between cover ($y$-axis) and length ($x$-axis) objectives for the same network topology as Figure 8, except that the energy constraint is relaxed ($e_0 = 100$, with no need for 'border adjustment') and a larger communication range is used ($r = 25m$). We represent the cover-objective by the number of *uncovered* (*i.e.*, not yet covered) nodes so that both objective values are to be minimized. Each plotted data point represents a tour produced by one iteration of the algorithm, with the top leftmost point denoting the initial tour at Iteration 1 with only the three members of $Y_1$ that must be docking-visited. Each next point toward the bottom right signifies the addition of a new tour stop which increases the tour length in order to, hopefully, cover more sensors. Notice that the two tour points representing Iterations 3 and 4 are both dominated by the tour from Iteration 2, and thus are not Pareto-efficient. We plot them in the curve just to show all iterations. The Pareto frontier computed for this network should only include tours from Iterations $1, 2,$ and 5. In general, as the length of the tour increases, the number of uncovered nodes decreases until finally becoming zero when all sensors are covered. We find that at Iteration 2, there is only one sensor not yet covered with a tour length of $180m$. However, it takes three additional tour stops and about $20\%$ increase in tour length in order to cover the last sensor at Iteration 5. In practice, we may decide to give up visiting the very last sensor by choosing the tour generated at Iteration 2 to trade coverage for tour length. Therefore, the Pareto frontier computed by the algorithm could help make better tour decisions that balance both objectives in practical applications.

Next, we compare the heuristic UDMP-AD tours to the optimal solutions of CSP with and without line-cover, denoted as CSP-LC and CSP, respectively. Both CSP-LC and CSP may be regarded as special cases of UDMP-AD with $Y_1 = Y$, $Z_1 = Z$, and $e_0 = \infty$, except that CSP uses only point-cover without line-cover. With a sufficiently large $e_0$ value in UDMP-AD,

| | Point-cover | | Line-cover | |
|---|---|---|---|---|
| | Usage | Optimality | Usage | Optimality |
| CSP-LC | yes, explicit | optimal | yes, **explicit** | optimal |
| UDMP | yes, explicit | sub-optimal | yes, **implicit** | sub-optimal |
| CSP | yes, explicit | optimal | **no** | |

Table 1

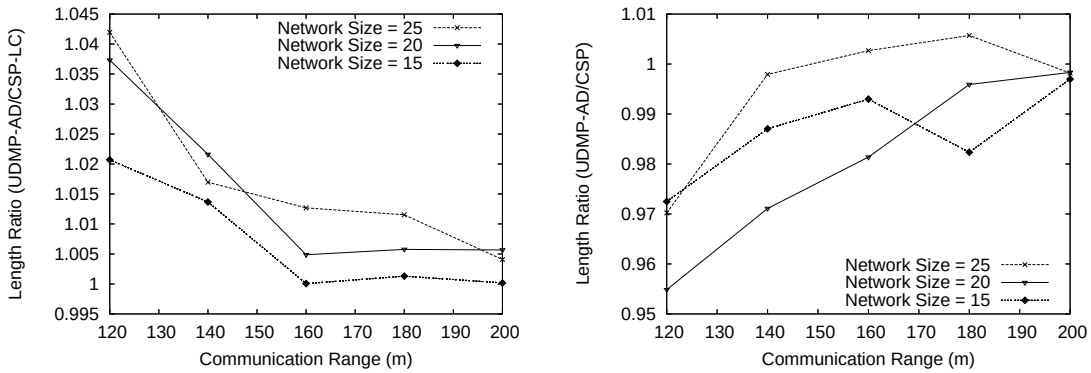
we compare the tour length of the last iteration in UDMP-AD (which should cover all sensors in $Z$) to the corresponding optimal solutions of CSP-LC and CSP which optimize only the single length-objective with full sensor coverage. Note that under this comparison scenario, CSP-LC and UDMP-AD addresses *exactly* the same problem both allowing line-cover. Table 1 summarizes the usage of point-cover and line-cover for the length-objective under full coverage in the optimal tours of CSP-LC and CSP, and in the heuristic tour of UDMP (including both UDMP-AD and UDMP-DoD). Apparently, point-cover has been explicitly used in all three solutions to select tour stops, except that UDMP heuristics may not achieve optimality. Notice that the usage level of line-cover decreases in the order of CSP-LC, UDMP, and CSP. First, CSP-LC *explicitly* specifies line-cover (together with point-cover) as part of the coverage constraint (Equation (7)) in its ILP problem formulation, based on both optimal tours are constructed. Next, UDMP *implicitly* uses line-cover by incorporating line-coverage into the total coverage only *after* a TSP tour has been constructed with tour stops selected solely based on point-cover (Steps 3 and 4 of both algorithms in Section 5). Finally, CSP totally omits line-cover.

Due to the NP-hardness of CSP-LC and CSP, we could only obtain optimal solutions for small networks within a reasonable amount of execution time and memory usage. By formulating CSP-LC and CSP in ILP models (as in Section 4.2), we solve both problems optimally using the GNU linear programming kit (GLPK) [25] for the network size of $15$, $20$, and $25$ nodes, and the communication radius ranging from $120m$ to $200m$. In each of these configurations, twenty networks are generated with nodes randomly distributed over a terrain of size $500 \times 500\ m^2$. Each network includes one depot and two docking stations that must be visited, with the remaining nodes as sensors to be covered. Each data point in the following figures is an average of simulation results from these random networks.

Figures 10(a) and 10(b) depict the length ratios of the UDMP-AD solution over the corresponding optimal solutions of CSP-LC and CSP, respectively, with different communication ranges and network sizes. As observed in Figure 10(a), the proposed UDMP-AD algorithm could produce tours of length much close to that of the optimal solutions—no more than $4.2\%$ longer in all simulated network sizes and communication ranges, which validates the effectiveness of the UDMP-AD algorithm. For each network size, the difference in length between the UDMP-AD and the optimal CSP-LC solutions generally increases (*i.e.*, deviates from the ratio of $1$) as the communication range decreases (corresponding to a *sparser* neighborhood for each node), due to the following reasons. First, the relative benefits of line-cover versus point-cover for the length-objective (under full coverage) increases in sparser networks, as fewer nodes could be point-covered at the tour stops. One extreme case is when every node is outside any other node's communication range, *i.e.*, each node as a tour stop could only point-cover itself, which will force all nodes to become tour stops if only point-cover is used for full coverage. In this case, line-cover could potentially lower tour length by *not* selecting all the nodes as tour stops, *i.e.*, with some nodes being line-covered by tour links between adjacent tour stops. Second, UDMP-AD does not explicitly use line-cover for selecting tour stops to decrease tour length as CSP-LC does, although the effect of line-cover is indeed counted in UDMP-AD to increase coverage after each tour decision, as compared in Table 1. Therefore, in sparser networks (*e.g.*, with shorter communication range)where line-cover could potentially have a bigger impact, UDMP-AD performs relatively poorer compared to CSP-LC optima due to the *under-usage* of line-cover.

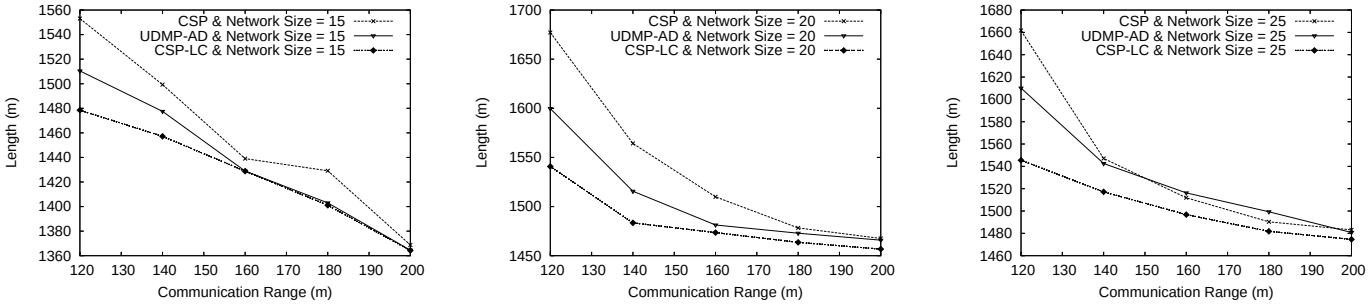
Compared to CSP, UDMP-AD could produce tours with length close to (less than $0.6\%$ worse) and mostly shorter (up to $4.5\%$ better) than the optimal solutions of CSP, as seen in Figure 10(b). This is due to the usage (although implicit) of line-cover in UDMP-AD compared to no usage of line-cover in CSP, which helps improve the coverage without sacrificing the tour length. In particular, UDMP-AD tours are shorter than CSP optima in networks with shorter communication range and/or fewer number of nodes representing sparser neighborhoods, where the added benefit of line-cover in UDMP-AD dominates the optimality of point-cover in CSP (Table 1). Moreover, similar to Figure 10(a), the difference in length between tours of heuristic UDMP-AD and optimal CSP generally increases (*i.e.*, deviates from the ratio of $1$) as the communication range decreases for all network sizes, due to the increasing benefit of line-cover in sparser networks, which is also evidenced in the sparser network with $15$ nodes (compared to the other two denser networks) at the communication range of $180m$. UDMP-AD solutions are slightly inferior to CSP optima though for networks with more nodes ($25$) and longer communication ranges ($160m$ and $180m$) representing denser neighborhoods, because of the relative bigger effect of point-cover over line-cover in dense networks and the suboptimal usage of both point-cover and line-cover in UDMP-AD.

Figures 11(a), 11(b), and 11(c) compare the absolute values of tour length computed by the proposed UDMP-AD algorithm and by the optimal CSP-LC and CSP for different communication ranges with network sizes of $15$, $20$ and $25$ nodes, respectively. In all three figures, as the communication range increases, the length of all corresponding tours decreases, since a larger 'distance threshold to be covered' enables more sensors to be covered without traveling longer distance. In particular, Figures 11(a) and 11(b) show that the UDMP-AD solutions are better than the CSP optima in sparse neighborhood (*i.e.*, smaller network size and/or shorter communication range) due to the benefit of using line-cover, as has been evidenced in Figure 10(b). Moreover, the length of the UDMP-AD tour generally lies between that of CSP-LC (at below) and CSP (at above), ordered by the usage level of line-cover




(a) UDMP-AD vs. CSP-LC (b) UDMP-AD vs. CSP

Figure 10. Tour length ratio of heuristic UDMP-AD over optimal CSP-LC and CSP solutions with varying radii.




(a) $N = 15$ (b) $N = 20$ (c) $N = 25$

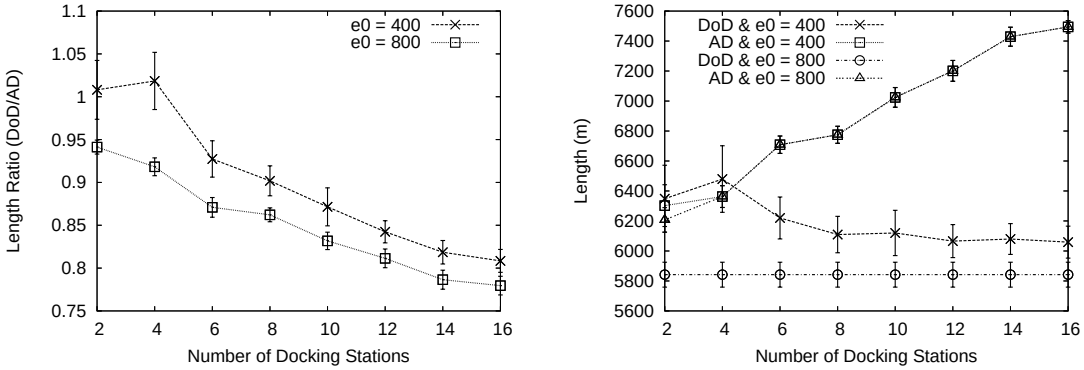
Figure 11. Tour length values of heuristic UDMP-AD and optimal CSP and CSP-LC vs. radius and network size $N$.

as listed in Table 1. Notice that CSP does not always bound UDMP-AD from above as the network becomes denser with larger network size and longer communication range (shown in Figure 11(c)), because the relative benefit of point-cover versus line-cover is more significant in dense networks than in sparse networks, which causes the optimality of point-cover in CSP to dominate the sub-optimality of both line-cover and point-cover in UDMP-AD.

### 6.2 Results of UDMP-DoD

We also evaluate the performance of UDMP-DoD algorithm in comparison to UDMP-AD, in terms of the length of the tour that covers all nodes in the sensor set $Z$, which is generated in the last iteration in both algorithms. In the following simulation, the $Z$ set consists of $300$ sensors randomly deployed over a terrain of size $1000 \times 1000\ m^2$, each with a fixed communication radius of $100m$. The docking set $Y$ includes a depot located at the center of the terrain and a varying number of docking stations randomly deployed across the network. For UDMP-DoD, only the depot must be docking-visited (*i.e.*, $Y_1$ includes the depot only), while all other randomly deployed docking stations (which form the set $Y \backslash Y_1$) may be selectively docking-visited on-demand to satisfy the energy constraint. For UDMP-AD, however, both the depot and all other docking stations must be docking-visited since $Y_1 = Y$. In this study, we vary the number of docking stations from $2$ to $16$ with an increment of $2$, and the energy constraint over $400$, $500$, $600$, $700$, and $800$. Each measurement in the following figures is the average of $20$ different randomly generated networks for a given configuration, and is drawn with $95\%$ confidence interval.

Figures 12(a) and 12(b) compare the tour lengths produced by the algorithms of UDMP-DoD and UDMP-AD, in terms of ratios and absolute values, respectively, when the number of docking stations varies over $\{2, 4, 6, \ldots, 16\}$ with fixed energy constraints of $400$ and $800$. As expected, UDMP-DoD typically produces shorter tours than UDMP-AD when more docking stations are randomly deployed in the network, due to better chances for *less deviate* on-demand recharging (i.e., to insert on-demand docking stations in tour with less increases of tour length), and vice versa. Specifically, Figure 12(a) shows that UDMP-DoD could decrease tour length (in comparison to UDMP-AD) by $22\%$ in scenarios of two or more docking stations with a large battery capacity of $e_0 = 800$, and by $19\%$ with a docking set of more than four and a small battery capacity of $e_0 = 400$. However, for the cases of $e_0 = 400$ with four or fewer docking stations, the UDMP-DoD tours are slightly worse than the UDMP-AD tours on average (by no longer than $1.8\%$), as fewer docking stations may cause longer detour for on-demand docking statistically. In general, the length ratio of UDMP-DoD over UDMP-AD decreases with the increasing number of docking stations, except for a slight 'bump' occurred at the case of four docking stations with $e_0 = 400$ to be elaborated by Figure 12(b) next. The length ratio also decreases when $e_0$ becomes larger, since a battery with larger capacity has less need for recharging during




(a) Length ratio (b) Length value

Figure 12. Comparison between DoD and AD tours vs. number of docking stations with $e_0 = 400$ and 800.

one complete tour to cover all sensors.

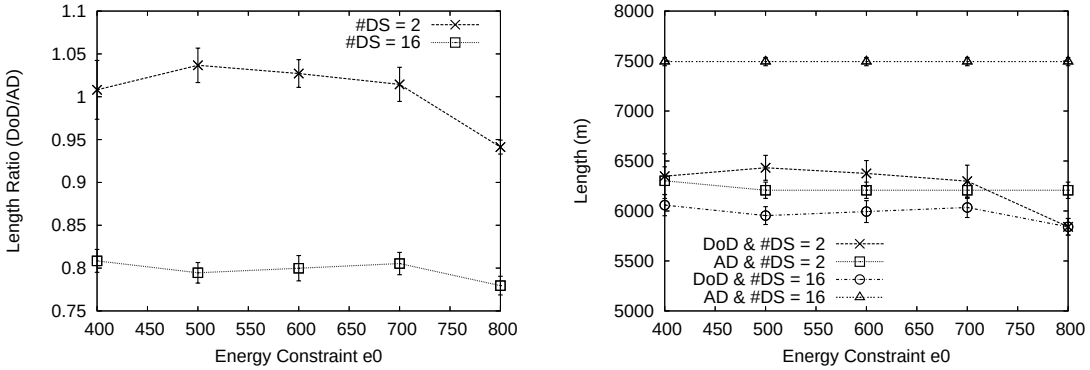
Figure 12(b) shows that the tour length of UDMP-AD increases almost linearly with the increasing number of docking stations, since all the docking stations in the network have to be docking-visited no matter it is necessary (for recharging) or not. Moreover, the UDMP-AD tour lengths with different energy constraints coincide for each docking set size except for the small set of 2, where the tour with a small battery capacity of $e_0 = 400$ is about $1.5\%$ longer than the tour with a large battery capacity of $e_0 = 800$ on average. This is due to the fact that with only two docking stations, the small battery case requires 'border adjustment' to correct energy constraint violations while the large battery case does not, and with more docking stations in the tour (thus more smaller segments) both cases satisfy the constraint without any violation. Next, for UDMP-DoD with $e_0 = 800$, the tour length remains the shortest ($5841m$) and constant across different docking set sizes, since the large capacity battery does not require any on-demand charging during the complete tour. For UDMP-DoD with $e_0 = 400$, the tour shortens at first and then stabilizes in length when more docking stations are added. Notice that there is a small rise of $2\%$ in length when the docking set size increases from 2 to 4 with $e_0 = 400$, because (a) mature docking is available immediately after the on-demand stop and thus selected by the greedy heuristic in the cases of four docking stations, rather than in the cases of two which only have pre-mature docking available from some backtracked stop; and (b) mature docking happens to add more length to tour (*i.e.*, more deviate) than pre-mature docking in the above cases (illustrated by Figure 7 in Section 5.2). The reasons also explain the 'bump' in Figure 12(a).

Figures 13(a) and 13(b) depict the length ratios and values of UDMP-DoD and UDMP-AD tours with varying energy constraints over $\{400, 500, \ldots, 800\}$ for 2 and 16 docking stations in the network. As observed in Figure 13(a), the length ratio of UDMP-DoD over UDMP-AD generally decreases slowly with increasing value of $e_0$ with two docking stations, due to less recharging need with an enlarging battery capacity, except for a slight rise at $e_0 = 500$ to be elaborated by Figure 13(b) next. With a larger docking set of 16, the slope of the length ratio becomes more or less flat under varying energy constraints, since enough docking stations may lead to *convenient* docking near any on-demand stop with less deviation from the tour. Thus the tour length remains short and stable regardless of locations or total times of recharging in the tour. In particular, with sixteen docking stations randomly deployed in the network, UDMP-DoD always achieves shorter tours than UDMP-AD by about $20\%$ on average for all $e_0$ values, as is also evidenced by Figure 12(a) before.

In Figure 13(b), the tour lengths of UDMP-AD remain unchanged across varying $e_0$ values for each docking set size, except for the small set of two docking stations with a small capacity battery of $e_0 = 400$, which requires 'border adjustment' in order to satisfy energy constraint and thus increases the tour length, as is also seen in Figure 12(b). Due to the almost constant UDMP-AD tour lengths across different $e_0$ values for each docking set size, the trend of either UDMP-DoD length curve resembles the related ratio curve in Figure 13(a). Notice that there is a $1.3\%$ increase in UDMP-DoD tour length with the small docking set of two when $e_0$ increases from 400 to 500. This is due to the fact that a small increase of battery capacity may just change the location(s) of on-demand stop(s) without reducing the total number of recharging in the tour, and the docking station(s) selected in the greedy fashion (*i.e.*, postponing recharging as late as possible) may cause more *deviation* with $e_0 = 500$ than with $e_0 = 400$ especially when the docking set is small. As a result, the increase of UDMP-DoD tour length at $e_0 = 500$ with two docking stations, together with the decrease of UDMP-AD tour length in the same scenario, leads to the corresponding ratio increase in Figure 13(a). Finally, when provided with the large capacity battery of $e_0 = 800$, UDMP-DoD tours of both docking set sizes converge to the shortest length of $5841m$, where no recharging is needed during a complete tour of visiting all sensors, as is also evidenced in Figure 12(b).

## 7 Conclusion

In this paper, we formulate the tour planning of a data mule to collect sensor data in UWSNs as an energy-constrained bi-objective underwater data muling problem (UDMP). UDMP defines two types of visit and two types of cover. In particular, the line-cover helps improve the cover-objective without sacrificing the length-objective. We propose two heuristic




(a) Length ratio (b) Length value

Figure 13. Comparison between DoD and AD tours vs. energy constraint with 2 and 16 docking stations.

algorithms to solve one special case of all-docking (UDMP-AD) and one generalized case of docking-on-demand (UDMP-DoD). Each algorithm computes a set of Pareto-efficient solutions addressing the tradeoff between the two optimization objectives. Extensive simulation validates the effectiveness of both algorithms. In particular, UDMP-AD performs better than the corresponding optimal solutions of the Covering Salesman Problem (CSP) in sparse neighborhoods (*e.g.*, smaller network size and/or shorter communication range), and closer to the optimal solutions of CSP with line-cover (CSP-LC) in dense networks (*e.g.*, larger network size and/or longer communication range). This is because the relative benefit of line-cover versus point-cover increases as the network becomes sparser, and the usage level of line-cover increases from CSP (no usage at all), UDMP (implicit usage with sub-optimal solutions), and to CSP-LC (explicit usage with optimal solutions). Moreover, UDMP-DoD often decreases the tour length compared to UDMP-AD, by avoiding unnecessary docking-visits in scenarios with enough number of randomly deployed docking stations and/or large battery capacity.

## REFERENCES


[1] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling a three-tier architecture for sparse sensor networks," in *IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, May 2003.

[2] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ser. SenSys '05. New York, NY, USA: ACM, 2005, pp. 154–165.

[3] C. H. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," *Theoretical Computer Science*, vol. 4, no. 3, pp. 237–244, 1977.

[4] S. Arora, "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems," *J. ACM*, vol. 45, no. 5, pp. 753–782, 1998.

[5] W. Zhao and M. H. Ammar, "Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks," in *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, ser. FTDCS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 308–.

[6] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *IEEE INFOCOM*, 2005, pp. 1407–1418.

[7] M. M. Bin Tariq, M. Ammar, and E. Zegura, "Message ferry route design for sparse ad hoc networks with mobile nodes," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '06. New York, NY, USA: ACM, 2006, pp. 37–48.

[8] J. Current and D. Schilling, "The covering salesman problem," *Transportation Sciences*, vol. 23, pp. 208–213, 1989.

[9] M. Gendreau, G. Laporte, and F. Semet, "The covering tour problem," *Operations Research*, vol. 45, pp. 568–576, 1997.

[10] J. Rao, T. Wu, and S. Biswas, "Network-Assisted Sink Navigation Protocols for Data Harvesting in Sensor Networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, March 31-April 3 2008, pp. 2887–2892.

[11] M. Ma and Y. Yang, "Data gathering in wireless sensor networks with mobile collectors," in *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, April 14-18 2008, pp. 1–9.

[12] M. Zhao and Y. Yang, "Bounded relay hop mobile data gathering in wireless sensor networks," in *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, October 12-15 2009, pp. 373–382.

[13] E. M. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *Discrete Appl. Math.*, vol. 55, no. 3, pp. 197–218, 1994.

[14] A. Dumitrescu and J. S. B. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," *J. Algorithms*, vol. 48, no. 1, pp. 135–159, 2003.

[15] K. Elbassioni, A. V. Fishkin, and R. Sitters, "Approximation algorithms for the Euclidean traveling salesman problem with discrete and continuous neighborhoods," *International J. of Computational Geometry and Applications*, vol. 19, no. 2, pp. 173–193, 2009.

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

[17] J. Current and D. Schilling, "The median tour and maximal covering tour problems: Formulations and heuristics," *European Journal of Operational Research*, vol. 73, pp. 114–126, 1994.

[18] N. Jozefowiez, F. Semet, and E.-G. Talbi, "The bi-objective covering tour problem," *Comput. Oper. Res.*, vol. 34, no. 7, pp. 1929–1942, 2007.

[19] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," in *Proceedings of the 1st ACM International Workshop on Underwater Networks (WUWNet)*, September 2006, pp. 17–24.

[20] J. Adams, "Ocean currents," Redmond: Microsoft, 1999, Microsoft Encarta. Vol. II. CD-ROM.

[21] G. Batchelor, *An Introduction to Fluid Dynamics, Second Edition*. Cambridge University Press, 2000.

[22] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, pp. 498–516, 1973.

[23] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[24] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[25] Free Software Foundation, "GNU Linear Programming Kit (GLPK) package," http://www.gnu.org/software/glpk/.