Revisiting Robotic Sensor Networks: A Budget-Constrained Covering Salesman Approach

Soham Patil, Bin Tang
Department of Computer Science, California State University Dominguez Hills
spatil1@toromail.csudh.edu, btang@csudh.edu

***Abstract*—We focus on robotic sensor networks (RSNs) and present a new algorithmic framework, the *budget-constrained covering salesman problem* (i.e., BC-CSP). We propose a suite of algorithmic solutions, including integer linear programming (ILP), approximation, and heuristic algorithms. Using commonly adopted mobility models of the robots and realistic battery power measurements from robotic applications, we show that a) with the same amount of battery power, our algorithms can collect up to 56.6% more data packets than the most representative CSP-based work and b) our algorithms perform within 91.0% to 97.3% of the ILP-based optimal solution while taking two orders of magnitude less amount of time than the optimal.**

Keywords – robotic sensor networks, budget-constrained covering salesman problem, algorithms.

I. Introduction

Background and Motivation. Robotic sensor networks (RSNs) are wireless sensor networks consisting of static sensors and mobile robots that collectively perform sensing, communication, and actuation in physical environments [14]. By combining robots' mobility and autonomy with sensor networks' real-time sensing capabilities, RSNs significantly enhance our ability to monitor and interact with the physical world. Traditional RSN applications include industry automation [15] and environmental monitoring and disaster relief [26]. In recent years, two new forms of RSNs that use autonomous underwater vehicles (i.e., AUVs) for deep water exploration [28] or unmanned aerial vehicles (i.e., UAVs and drones) for sensing data collection [16] have attracted much attention. The existing literature uses various names for mobile robots, such as drones or UAVs [16], AUVs [6], and mobile data collectors [18]. We use mobile robots throughout the paper.

Research Question. Due to the above intrinsic weakness of the modern battery and its resulting insufficient battery power, it could happen that the robot does not have enough battery power to visit all parts of the sensing area in an RSN application. This is especially true for many large-scale sensing applications, such as disaster relief and underwater exploration, in which robots are dispatched to a vast area for a relatively long period (e.g., one day). In this paper, we ask the following question: *how to find a path for a battery-constrained robot to maximize its data-collection performance in a large-scale RSN application?*

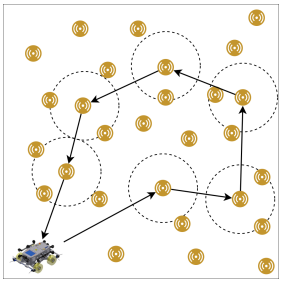


Fig. 1. Battery-constrained data collection in the RSN.

Our Contributions. In our RSN model, the sensor nodes have already generated various data packets from the environment. We use the number of data packets to indicate the value of information available at a node (and thus the importance of visiting this node). A robot with limited battery power is dispatched from an RSN depot into the network to collect these data packets. The depot consists of a base station, where the robot can upload collected data for further analysis and actuation, and a charging station, where the robot can be fully recharged. The robot collects the data packets wirelessly from the sensor nodes it visits and from all the sensor nodes within its wireless data-collecting range, as shown in Fig. 1. We say the robot *covers* the sensor nodes from which it collects packets. The goal is for the robot to visit a sequence of sensor nodes (i.e., *data-covering route*) to collect as many data packets as possible before it runs out of battery power and returns to the depot, where it uploads its collected data packets and fully recharges its battery. We refer to this problem as *data collection in RSNs* (DCR).

Underlying DCR is a new graph-theoretical problem, which we call *budget-constrained covering salesman problem* (i.e., BC-CSP). Given a graph in which each node has a prize, each edge has a cost, and a salesman with a limited budget, the BC-CSP aims to find a *Hamiltonian covering tour* that collects the maximum number of prizes within the budget. Here, "covering" means that the salesman collects prizes from both the node he visits and those within *covering distance* from a visited node. To the best of our knowledge, the network community has not yet identified or solved BC-CSP.

We design a suite of algorithms, including optimal integer linear programming (ILP), approximation, and heuristic algorithms to solve the BC-CSP. Using commonly adopted mobility models of the robots and realistic battery power measurements from robotic applications, we show that a) with the same amount of battery power, our algorithms can collect up to 56.6% more data packets than the most representative CSP-based work and b) our algorithms perform within 91.0% to 97.3% of the ILP-based optimal solution while taking two orders of magnitude less amount of time than the optimal.

II. Related Work

Covering Salesman Problem (CSP). CSP was initially proposed by Current and Schilling [10]. Arkin et al. [4] designed the first approximation algorithm for the geometric version of the CSP, in which each node specifies a neighborhood from which the salesman can visit. They provided performance ratios for various neighborhood shapes, including parallel unit segments, translations of a polygonal region, and circles. Other approaches to solving the CSP include branch-and-cut algorithms [19] and a generalized version of the CSP [12]. Recently, Naji-Azimi et al. [3], [20] studied the time-constrained maximal covering salesman problem and proposed several mathematical programming models and heuristic algorithms. Both works maximize the number of covered nodes, implicitly assuming all the nodes have the same prizes. Our work considers that different nodes have different prizes (i.e., data packets), and the goal is to find a Hamiltonian cycle that maximizes the total prize collected by the covering salesman. We propose a 2-approximation algorithm for a special case of BC-TSP. In contrast, [3], [20] proposed ILP solutions (which lack scalability) and heuristic algorithms (which lack performance guarantees).

BC-CSP differs from the orienteering problem (OR) [25], also known as the budget-constrained traveling salesman problem (BC-TSP). In BC-TSP, nodes have varying prizes and the salesman has a budget (indicating the maximum distance he can travel), the goal is to visit a sequence of nodes to collect the maximum number of packets within his budget. OR is a special case of BC-CSP where the covering distance is zero.

Robotic Sensor Networks (RSNs). Extensive research has achieved various data collection objectives in the RSN [14]. One group aims to optimize *resource provisioning objectives* of data collection [17], [18], such as minimizing the delay or energy consumption of the data-covering tour or maximizing the network lifetime. The other group achieves *admission control objectives* [8], [7], [24], which require retrieving data within a deadline or time duration. Some RSN applications have recently emerged that use UAVs or drones [28], [16]. In contrast, we focus on the robot's limited battery power and adopt a graph-theoretical approach (i.e., BC-CSP) to solve the data-collection problem. Patil et al. [21] formulated the data collection in RSN as a BC-TSP, in which the robot can collect data only from the nodes it visits. As the robot collects packets (i.e., prizes) from visited nodes and nodes within its wireless data-covering range, BC-CSP provides much better data-collection performance than BC-TSP.

III. Problem Formulation of DCR

System Model. We model the RSN as an area of len meter $\times$ wid meter, where n sensor nodes $V_s = \{1, 2, ..., n\}$ are randomly placed. Sensor node $i \in V_s$ is located at (x_i, y_i), $0 \leq x_i \leq len$, $0 \leq y_i \leq wid$, and it has generated $d_i > 0$ data packets, each is of b-bit. A depot, denoted as $s = (0, 0)$, is located at one corner of the RSN, as shown in Fig. 1. The depot serves as both a base station for uploading collected packets and a charging station for the robot upon its return from a data-collecting trip. Let $c(i, j)$ denote the Euclidean distance between any two nodes $i, j \in V$, where $V = V_s \cup \{s\}$; $c(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. We assume the robot has a wireless sensing range of T_r; that is, it can sense and collect packets from any sensor nodes within T_r. We refer to T_r as the robot's *data-covering range*.

Mobility and Data Collection Model for the Robot. To collect data packets, the robot moves directly to the location of a sensor node, turns on its wireless sensing radio interface, and senses and collects data packets from this sensor node and all other nodes within T_r distance from this sensor node. After collecting the data, the robot turns off its wireless radio interface, moves to another unvisited sensor node, and repeats this process. Our model differs from the "lawn mowing" model [5], which finds the shortest tour for the mower such that every point in a region is covered by the mower along the tour. In our data collection, lawn mowing always requires turning on the robot's wireless radio, which is not energy-efficient and further depletes the robot's limited battery capacity. The robot turns on its sensing radio interface only when collecting data to conserve battery power.

Mobility Graph and Data Collection Graph. The above model gives rise to two different graphs for the RSN. First, the robot's movement can be characterized by a complete graph $G(V, E)$, where the weight $w(i, j)$ of any edge $(i, j) \in E$ is $c(i, j) \times \mu$, the robot's mobility energy consumption moving from i to j. We refer to this complete graph as the *mobility graph* of the RSN. On the other hand, the robot's data-collecting behavior is modeled by a *data collection graph* $G_1(V, E_1)$, where an edge $(i, j) \in E_1$ if $c(i, j) \leq T_r$, $\forall i, j \in V$. For any node $i \in V$, denote i's *1-covered nodes* as $N_i^1 = \{j | (i, j) \in E_1\}$. When the robot visits i, it collects packets from i and from all its 1-covered nodes N_i^1 (if any). We say that both node i and its 1-covered nodes N_i^1 are *covered* by the robot. Note that G_1 is a subgraph of G, as $E_1 \subseteq E$. We thus use the mobility graph G as the input graph for the DCR.

Energy Model. There are two primary components of a robot's energy consumption during its data-collecting process [27]. One is its *mobility energy*, the energy required to overcome friction between its wheels and the terrain. The maximum distance a wheeled robot of battery power $\mathcal{E}$ can move is $d = \frac{\mathcal{E}}{w \times C_{crr}}$ [27]. Here, C_{crr} is the rolling friction coefficient of the terrain, and w is the robot's weight. We define a robot's *mobility coefficient* as the battery power consumed per unit of traveled distance and denote it as μ; $\mu = w \times C_{crr}$. For a robot to move l meters, its mobility energy consumption, denoted as E_m, is $E_m = \mu \times l$.

The other is *robotics energy* E_c, which powers the robot's sensing, computing, and communication capabilities. In data collection, E_c primarily represents the robot's wireless energy consumption while sensing and collecting data packets from sensor nodes. For a robot to collect a data packet of b-bit within

T_r, its robotics energy $E_c = \epsilon_e \cdot b$, where $\epsilon_e = 100$nJ/bit is the energy consumption per bit on the circuit of the robot [13].

Problem Formulation of DCR. Given the mobility graph G, let $R = \{s, t_1, t_2, ..., t_a, s\}$ be a *data-covering tour* of the robot, where the robot starts from depot s, visits a sequence of a distinct sensor nodes $t_j \in V_s$, $1 \leq j \leq a$, $1 \leq a \leq n$, to collect the data packets from t_j and its 1-covered nodes, and returns at depot s before running out of battery. The mobility energy of the robot along R is thus $E_m = \mu \times \big(c(s, t_1) + \sum_{i=1}^{a-1} c(t_i, t_{i+1}) + c(t_a, s)\big)$. Let $\mathcal{N}_R = \bigcup_{j=1}^{a} N_{t_j}^1$ denote all the sensor nodes that are 1-covered by at least one node in R. Denote the total number of data packets the robot collects when moving along R as D_R; $D_R = \sum_{j \in R \cup \mathcal{N}_R} d_j \cdot b$. Thus the robotics energy of the robot is $E_c = \epsilon_e * D_R$. Denote the total battery power spent by the robot along R as E_R; $E_R = E_m + E_c$. Given the robot's initial battery power $\mathcal{E}$, the goal of the DCR is to find an *optimal data-covering tour* R that maximizes D_R subject to $E_R \leq \mathcal{E}$.

Mobility Energy vs. Robotics Energy. We compare the mobility and robotics energy using a typical data collection scenario in the RSN. Following [27]; we assume a robot's weight w is around 600Kg while a typical terrain type $C_{crr} = 0.17$. Thus, the mobility coefficient $\mu = w \times C_{crr} \approx 100$ Joule/m. Assume a data packet is 1024 B; the energy required to collect one packet is around 10^6 nJ. With one Jouel of battery power, a robot can travel 0.01m in movement or collect 1000 packets in data-collecting. In a large-scale RSN (kilometers in size), the robot's robotics energy is negligible.

Next, we formally define BC-CSP and show that it is equivalent to the DCR. BC-CSP generalizes CSP, which generalizes the classic traveling salesman problem (TSP), wherein the covering distance in CSP is zero.

BC-CSP. Given a complete graph $G'(V', E')$ where a node $i \in V'$ has a prize $p_i \geq 0$ and an edge $(u, v) \in E'$ has a weight $w(u, v) \geq 0$. Each node $i \in V'$ can cover a subset of nodes $S_i \subset V'$, referring to i's *covering set*. A traveling salesman is located at $r \in V'$ and has a budget of $\mathcal{B}$, the maximum distance he can travel before returning to r. When the salesman visits a node i, he can collect prizes from i and all nodes in S_i (if they are still available). We assume each prize can be collected at most once. Given a *prize-covering cycle* $R = \{r = v_1, v_2, v_3, ..., v_x = r\}$, the total prize it collects is $P_R = \sum_{i \in R \cup \{j | j \in S_i \wedge i \in R\}} p_i$ and the cost along R as $C_R = \sum_{i=1}^{x-1} w(v_i, v_{i+1})$. The BC-CSP aims to find an R that maximizes P_R subject to $C_R \leq \mathcal{B}$. BC-CSP is NP-hard as its special case of CSP, where $\mathcal{B} = +\infty$, is NP-hard [10]. We give the theorem below without proof.

Theorem 1: DCR on the mobility graph $G(V, E)$ is a BC-CSP on $G'(V', E')$.

IV. Optimal and Approximation Algorithms

A. Integer Linear Programming (ILP).

We formulate DCR as an ILP, as shown in ILP (A). We introduce four decision variables: $x_{i,j}$ indicating if edge (i, j) is on the data-covering tour of the robot; y_i indicating if node i is visited by the robot (i.e., on the data-covering tour); z_i indicating if node i's data packets are collected by the robot (i.e., i is either visited by the robot or 1-covered by a node visited by the robot). Finally, u_i is a position variable showing the order in which the node i is visited. $u_s = 1$ shows that s, being the depot, is both a starting and an ending node.

$$(A) \quad \max \sum_{j \in V_s} d_i \cdot z_i \tag{1}$$

s.t.

$$x_{i,j}, y_i, z_i \in \{0, 1\}, \quad \forall i, j \in V \tag{2}$$

$$\sum_{j \in V_s} x_{s,j} = \sum_{i \in V_s} x_{i,s} = 1 \tag{3}$$

$$\sum_{j \in V} x_{i,j} = \sum_{j \in V} x_{j,i} = y_i \leq 1, \quad \forall i \in V_s \tag{4}$$

$$\sum_{i \in V} \sum_{j \in V} (d_{i,j} \cdot x_{i,j} \cdot \mu) \leq \mathcal{E}, \tag{5}$$

$$z_i \geq y_i + \sum_{j : c(i,j) \leq T_r} y_j, \quad \forall i \in V_s \tag{6}$$

$$2 \leq u_i \leq n, \quad \forall i \in V_s \tag{7}$$

$$u_i - u_j + 1 \leq n \cdot (1 - x_{i,j}). \quad \forall i, j \in V_s \tag{8}$$

Objective function 1 is to maximize the total number of data packets collected from all the sensor nodes covered by the robot. Constraint 2 shows the integer constraints of $x_{i,j}$, y_i, and z_i. Constraint 3 ensures that the data-covering tour starts from and ends at depot s. Constraint 4 ensures that each sensor node in V_s is visited by the robot at most once. Constraint 5 enforces the battery power constraint of the robot. Constraint 6 is the *covering constraint*, indicating that if a node is visited or within T_r of any visited node, it is covered by the robot, and the robot collects its packets. Constraints 7 and 8 are Miller–Tucker–Zemlin (MTZ) formulations for subtour elimination [2] that guarantee that the final data-covering tour is a single tour rather than multiple tours.

B. Approximation Algorithm.

We consider a special DCR case wherein $T_r = 0$ and $d_i = d_j$, $\forall i, j \in V_s$, and refer to it as DCR-0. We propose a primal-dual-based approximation algorithm for DCR-0 viz. Algo. 1. The main idea of our algorithm is to reduce a weighted optimization problem to an unweighted one by following techniques proposed by Goemans and Williams [11], wherein given an ILP of the problem and its feasible dual solution y, it iteratively finds a feasible primal solution x that obeys the complementary slackness conditions [23] w.r.t. y.

As ILP(A) is based on the MTZ formulation, which makes it challenging to derive its dual, we present below another primal ILP and its dual based on the cycle-cut intersection of a graph. Given a cut $S \subseteq V$, let x_S indicate if the robot will tour S, and y_e denote if an edge $e \in E$ is on the tour. Let

$\delta(S)$ denote the set of edges with exactly one endpoint in the set S. The primal ILP of DCR-0 is:

$$(B) \quad \max \sum_{S \subseteq V} |S| \times x_S \tag{9}$$

s.t.

$$x_S = \{0,1\}, \quad \forall S \subseteq V \tag{10}$$

$$y_e = \{0,1\}, \quad \forall e \in E \tag{11}$$

$$\sum_{e \in \delta(S)} y_e \geq 2 \times \sum_{T:S \subset T} x_T, \quad \forall S \subset V \tag{12}$$

$$\sum_{e \in E} c_e \times y_e \times \mu \leq \mathcal{E}, \tag{13}$$

$$\sum_{S \subseteq V} x_S = 1. \tag{14}$$

Constraint 12 states that if the robot tours a subset T such that $S \subset T$, there must be at least two edges across the cut S. Constraints 13 and 14 enforce the battery constraint and ensure that one tour is selected. Its dual linear program is:

$$(C) \quad \min \lambda_1 \times \mathcal{E} + \lambda_2 \tag{15}$$

s.t.

$$2 \times \sum_{T \subset S} y_T + \lambda_2 \geq |S|, \quad \forall S \subset V \tag{16}$$

$$\sum_{e \in \delta(S)} y_S \leq \lambda_1 \times c_e, \quad \forall e \in E \tag{17}$$

$$\lambda_1, \lambda_2, y_S \geq 0, \tag{18}$$

where each primal variable and constraint in ILP(B) becomes a dual constraint and variable in ILP(C), respectively.

Approximation algorithm for DCR-0. Next, we design the primal-dual method viz. Algo. 1 to construct the data-covering tour. Given the above ILP(B) for DCR-0 and its feasible dual solution λ_1 in ILP(C), Algo. 1 iteratively finds a feasible primal solution y that obeys the complementary slackness conditions [23] w.r.t. λ_1.

Algorithm 1 An Approximation Algorithm for DCR-0.

1: First, it sets the value of λ_1 and then uses it to construct a full dual solution and corresponding potential tours. These tours may not be feasible with respect to the battery constraint.
2: Next, it adjusts λ_1 to find a feasible solution with a bounded approximation ratio. It sets all y_S to zero and the collection of active sets to all singleton nodes.
3: Then, in each iteration, it increases y_S in the collection of active sets (*growth phase*) until either a dual constraint for an edge between two sets becomes neutral, where a subset $S \subseteq V$ is neutral if $2\sum_{T \subseteq S} y_T = |S|$.
4: It deletes edges in S to obtain the final path that spans $\mathcal{E}/2$ nodes (*pruning phase*).
5: Finally, the tour is obtained by starting from s of the pruned tree while traversing all the edges at most twice.

Algo. 1 is similar to the 2-approximation algorithm for a related prize-collecting TSP [22]. Its time complexity is $O(|V|^2 \cdot \log|V|)$. We give Theorem 2 without proof that it is a $2+\epsilon$ approximation for DCR-0 [22].

Theorem 2: Let O^* be the optimal subset of vertices for DCR-0. In Algo. 1, the primal problem yields more than $1/2$ times the value of the feasible dual solution. This implies that the resulting tour in Algo. 1 is with a factor of $2+\epsilon$ of O^*.

V. Greedy Data-Covering Algorithms

As the above ILP(A) is time-consuming to compute and the approximation algorithm viz. Algo. 1 is challenging to implement, we present two efficient and easy-to-implement greedy algorithms: the prize-based covering algorithm (PCA) and the spanning-tree-based covering algorithm (SCA).

A. Prize-Based Covering Algorithm (PCA).

To guarantee the salesman's safe return to the depot before running out of budget while maximizing the number of packets collected, we define the following.

Definition 1: (**Battery-Feasible Sensor Nodes.**) Given that the robot is currently located at node $i \in V$ and with battery B, the set of sensor nodes U that have not been visited by the robot, its *battery-feasible sensor nodes*, denoted as $\mathcal{F}(i,B,U)$, is the set of sensor nodes that the robot has not visited and that it has sufficient battery to travel to and then return to s. $\mathcal{F}(i,B) = \{j | \big(c(i,j) + c(j,s)\big) \times \mu \leq B \wedge j \in V_s \wedge U\}$. □

Definition 2: (**Prize at a Sensor Node.**) Given a sensor node $i \in V_s$, its available *prize*, denoted as p_i, is all the data packets that can be collected by the robot when it visits i. I.e., $p_i = \sum_{j \in N_i \cup \{i\}} d_j^c$, where N_i is i's 1-covered nodes, and d_j^c is the current number of data packets available at j. Initially, $d_j^c = d_j$, and it becomes 0 when the robot collects j's packets. We denote a node i's *initial prize* as $p_i^o = \sum_{j \in N_i \cup \{i\}} d_j$. □

Definition 3: (**Covered Prize-Cost Ratio of a Battery-Feasible Sensor Node.**) Given the robot's current location i, for a battery-feasible node $k \in \mathcal{F}(i,B,U)$, its *covered prize-cost ratio*, denoted as $pcr(i,k)$, is the ratio between the prize available at k and the battery consumption of the robot moving from i to k. That is, $pcr(i,k) = \frac{p_k}{c(i,k) \times \mu}$. □

Definition 4: (**2-Covered Nodes.**) Given a sensor node j, its 2-covered nodes are the 1-covered nodes of any 1-covered node of j that are not 1-covered nodes of j, denoted as N_j^2. That is, $N_j^2 = \{i | i \in N_k^1 \wedge k \in N_j^1 \wedge i \notin N_j^1\}$. □

In Fig. 2(a), l is a 2-covered node of j, as l is a 1-covered node of k, which is a 1-covered node of j, while l is not a 1-covered node of j. Note that if l is a 2-covered node of j, j is also a 2-covered node of l. Both 1-covered and 2-covered nodes are critical in the data-collecting and prize-updating process in Algo. 2.

Observation 1: During the data-collecting and prize-updating process, for a sensor node j's packets d_j and prize p_j, if $d_j = 0$, it is still possible that $p_j > 0$; however, if $p_j = 0$, it must be that $d_j = 0$. In particular, there are four possible combinations of d_j and p_j: a) $d_j > 0$ and $p_j = p_i^o$, meaning neither j's nor any of its 1-covered nodes' packets have been

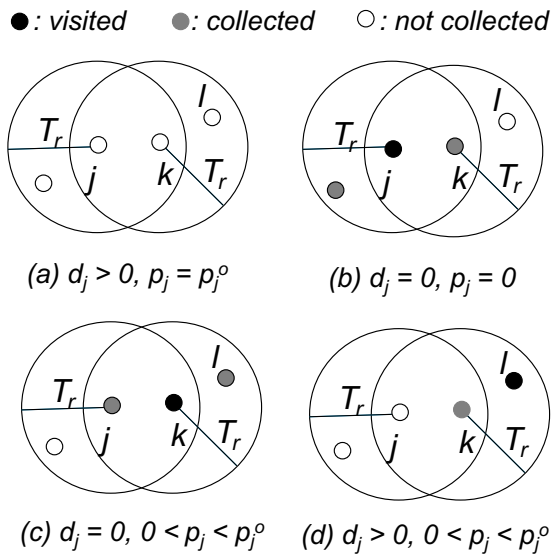


Fig. 2. Four combinations of d_j and p_j. A sensor node j can be in one of three scenarios: visited by the robot, not visited but its packets collected by the robot, or not visited and its packets not collected by the robot.

collected by the robot (Fig. 2(a)); b) $d_j = 0$ and $p_j = 0$, meaning both node j's and all its 1-covered nodes' packets have been collected (Fig. 2(b)); c) $d_j = 0$ and $0 < p_j < p_j^o$, meaning j's packets have been collected but at least one of its 1-covered nodes' packets have not (Fig. 2(c)); and d) $d_j > 0$ and $0 \leq p_j < p_j^o$, meaning i's packets hav not been collected by the robot but one or more of its 1-covered nodes' packets have been collected (Fig. 2(d)). □

Prize-Based Covering Algorithm (PCA). Algo. 2 works as follows. First, all the necessary variables related to the robot's data-covering tour and battery power are initialized (line 1). It computes the initial prizes for all sensor nodes (lines 2-5) and then proceeds in rounds. In each round, located at the current node i, the robot visits a battery-feasible node j with the largest covered prize-cost ratio and updates all the route-related information (lines 7-10). It then collects j's prize p_j by collecting packets from j and its 1-covered nodes, and the prizes of all the sensor nodes involved (lines 11-31), as illustrated next. This continues until it no longer finds a battery-feasible node, at which point it returns to s and outputs the data-covering tour, the collected packets, and the robot's energy cost for this route (lines 34-35). In Algo. 2, the robot visits at most $n = |V_s|$ nodes. At each node j, it updates the prizes by checking all of its 1-covered and 2-covered nodes, which is $O(n^2)$. The time complexity of Algo. 2 is $O(n^3)$.

Prize-Updating. When the robot visits sensor node j, the data-collecting and prize-updating process takes place in two steps. First, it sets p_j to 0, since all packets from j and its 1-covered nodes $k \in N_j$ (if any) will be collected (lines 12 - 20). Note, we also update p_k if node k has packets, as the robot will collect them in this round (lines 17-19). Meanwhile, A_j records all sensor nodes whose packets are collected by the robot during its visit to j.

Second, we further update the prizes of node k and k's 1-covered nodes $l \in N_k$ (lines 21-31). There are two cases. The first case is that l's packets are collected in this round (i.e., $l \in A_j$). In this case, it must be that $l \in N_j^1$, as shown in Fig. 3(a). We thus update k's prize as $p_k = p_k - d_l$ (lines 23-24). The second case is that l's packets are not collected in this round, which is further divided into two subcases. One subcase is that l is j's 1-covered node. As its packets were

Algorithm 2 Prize-Based Covering Algorithm (PCA).

Input: A mobility graph $G(V, E)$, d_i, T_r, μ, $\mathcal{E}$, and depot s,
Output: A data-covering tour R, P_R, and C_R.
Notations: R: the data-covering tour, starting from s;
C_R: the energy cost of the robot on R, initially zero;
P_R: the prizes collected on R, initially zero;
U: the set of unvisited sensor nodes, initially V_s;
i: the node where the robot is located currently;
A_j: nodes whose packets are collected when j is visited;
B: current battery power of the robot, initially $\mathcal{E}$;

1: $i = s$, $R = \{s\}$, $C_R = P_R = 0$, $U = V_s$, $B = \mathcal{E}$;
2: **for** (each $i \in V_s$) **do**
3: $N_i^1 = \{j | j \in V_s \wedge c(i,j) \leq T_r \wedge j \neq i\}$;
4: $p_i = p_i^0 = \sum_{j \in N_i^1 \cup \{i\}} d_j$;
5: **end for**
// if there are still battery-feasible nodes for the robot
6: **while** $(\mathcal{F}(i, B, U) \neq \phi)$ **do**
7: $j = \text{argmax}_{k \in \mathcal{F}(i,B,U)} pcr(i,k)$;
8: $R = R \cup \{j\}$, $P_R = P_R + p_j$;
9: $C_R = C_R + c(i,j) \times \mu$;
10: $B = B - c(i,j) \times \mu$, $U = U - \{j\}$;
11: $A_j = \phi$ (empty set), $p_j = 0$;
12: **if** $(d_j \neq 0)$ **then**
13: $d_j = 0$;
14: $A_j = \{j\}$;
15: **end if**
16: **for** (each $k \in N_j^1$) **do**
17: **if** $(d_k \neq 0)$ **then**
18: $p_k = p_k - d_k, d_k = 0$; $A_j = A_j \cup \{k\}$;
19: **end if**
20: **end for**
21: **for** (each $k \in N_j^1$) **do**
22: **for** (each $l \in N_k^1$) **do**
23: **if** $(l \in A_j)$ **then**
24: $p_k = p_k - d_l$;
25: **else**
26: **if** $(l \in N_j^2 \wedge k \in A_j)$ **then**
27: $p_l = p_l - d_k$;
28: **end if**
29: **end if**
30: **end for**
31: **end for**
32: $i = j$;
33: **end while**
34: $R = R \cup \{s\}$, $C_R = C_R + c(i,s) \times \mu$;
35: **return** R, P_R, C_R.

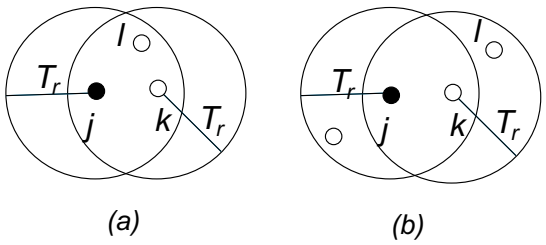
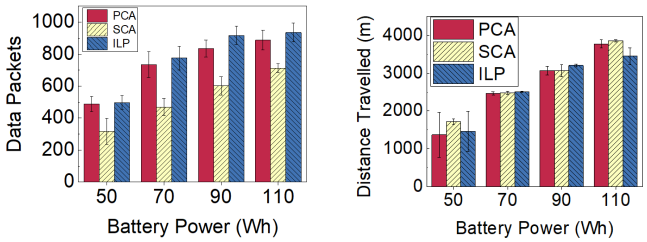


Fig. 3. Prize-updating when node j is visited, where k is j's 1-covered node. (a) l is a 1-covered node of both j and k. (b) l is k's 1-covered node and j's 2-covered node.



(a) Collected Packets. (b) Traveled Distance.

Fig. 4. Comparing PCA, SCA, and ILP.

not collected by the robot in this round, they must have been collected in a previous round, so nothing needs to be updated. The other subcase is that l is j's 2-covered node (i.e., $l \in N_j^2$), as shown in Fig. 3(b), then we need to check if $k \in A_j$ (i.e., if k's packets are collected in this round). If so, we update l's prize as $p_l = p_l - d_k$ (lines 26-28).

B. Spanning-Tree-Based Covering Algorithm (SCA).

SCA is based on the *spanning tree-based covering algorithm* proposed by Ma et al. [18]. Its main goal is to select a set of *polling nodes* from the sensor nodes so that the robot can visit them power-efficiently while covering all sensor nodes in the RSN. First, it uses a minimum-spanning-tree-based greedy algorithm to select polling nodes in increasing order of cost (defined next) until all sensor nodes are covered. Then, it applies the well-known 2-approximation algorithm for the traveling salesman problem [9] to the polling nodes to find a data-covering route for the robot. It finds a minimum spanning tree of all the polling nodes, duplicates all edges of this tree, and then finds an Eulerian cycle of these edges.

However, [18] does not consider battery constraints. We thus introduce a binary-search-based mechanism into SCA to iteratively find a subarea of l^* meter $\times$ wid meter, $0 < l^* \leq len$, where len and wid are the length and width of the entire RSN area, such that the robot's data-covering tour in this subarea is close to but within $\mathcal{E}$, its initial battery power.

Definition 5: (Cost of Selecting a Polling Node.) Let $P(l^*)$ denote the set of polling nodes $P(l^*)$ for the subarea l^* meter $\times$ wid meter. Given $P' \subset P(l^*)$, its 1-covered nodes is $C(P') = \cup_{i \in P'} N_i^1 \cup P'$, and its uncovered nodes is $U(P') = V_s(l^*) - C(P')$, where $V_s(l^*) = \{i \in V_s \wedge 0 \leq x_i \leq l^*\}$. The cost of selecting sensor node $j \in U(P')$ as a polling node w.r.t. P', denoted as $cost(j, P')$, is the ratio between the smallest battery consumption of the robot to move from any existing polling node $i \in P'$ to j and the number of newly covered nodes due to j being a polling node. That is, $cost(j, P') = \frac{\min_{i \in P'} c(i,j) \times \mu}{|U(P') \cap N_j^1|}$. □

SCA works as follows. It starts by finding the energy power C_R of the data-covering route for the entire sensor nodes V_s following [18]. If $C_R \leq \mathcal{E}$, the robot has sufficient battery to cover V_s, and it stops; otherwise, it reduces to half of the area to find the battery power needed to cover $V_s(\frac{len}{2})$. This repeats the binary search until it finds a subarea where the difference between C_R and $\mathcal{E}$ is within the threshold value Δ. Its time complexity is $O(\log \mathcal{E} \times n^2)$. Due to space constraints, we omit its pseudo-code presentation.

VI. Performance Evaluation

Using measurements of battery power and mobility models of robots from real applications, we compare various data-covering algorithms viz. ILP(A) (**ILP**), approximation algorithm viz. Algo. 1 (**PrimalDual**), prize-based covering algorithm viz. Algo. 2 (**PCA**), spanning-tree-based covering algorithm (**SCA**), and finally, the budget-constrained TSP-based algorithm (**TSP**) [21], which is the only work that we are aware of that explicitly considers battery power of the robot as the problem constraint in the RSN data collection. We use CPLEX [1] for ILP computation. For SCA, we set the binary search threshold Δ as the 10% of $|C_R - \mathcal{E}|$.

Experiment Setup. We use small RSNs of $1000\text{m} \times 1000\text{m}$, where 20 sensor nodes are randomly placed when the ILP optimal solutions are computed, and large RSNs of $10,000\text{m} \times 10,000\text{m}$, where 100 sensor nodes are randomly placed. In either case, the depot s is at (0, 0) of the BSN. Each sensor generates a random number of data packets in [0, 100], each of 1024B. We set the mobility energy coefficient μ as 100 Joules/m following [27]. Each data point in our plots is the average of 10 runs, with a different RSN instance constructed and applied to all algorithms for a fair comparison. The error bars indicate 95% confidence intervals. We implement our simulator in Java on Windows 11 with an AMD Ryzen 5 4000 Series 6-Core processor and 24GB of DDR4 Memory.

Comparing PCA, SCA, and ILP. Fig. 4 compares PCA, SCA, and ILP in small RSNs by varying the initial battery $\mathcal{E}$ of the robot from 50Wh to 110Wh. With a mobility coefficient $\mu = 100$ Joules/m, this battery range means the robot can travel 1800m and 3960m, respectively. Fig. 4(a) shows that

TABLE I
Execution Time (ms) of Different Algorithms.

Battery power $\mathcal{E}$ (Wh)	PCA	SCA	ILP
50	4	41	353
70	5	45	575
90	6	52	1507
110	6	60	2053

with the increase of battery power, more packets are collected for all the algorithms. Being an optimal solution, ILP always collects the most packets. PCA performs very close to ILP, though, with the number of packets ranging from 91.0% to 97.3% of optimal. Moreover, PCA significantly outperforms SCA, collecting between 24.7% and 56.6% more packets. Fig. 4(b) shows the distances traveled by the robot, showing that in all the test cases, battery power is a constraint limiting the traveling distance of the robot in all the algorithms. Table I shows the execution time of different algorithms w.r.t. $\mathcal{E}$. The execution time of PCA is one order of magnitude smaller than that of SCA, which is one order of magnitude smaller than that of the ILP. This shows that PCA is the most efficient data-covering algorithm among the three approaches.

Comparing PCA, SCA, and PrimalDual. Next, we compare the PCA and SCA with the approximation algorithm PrimaDual in small RSNs. As the 2-approximation provided by PrimalDual applies only to the special case where the robot's data-covering range T_r is 0 and all sensor nodes have the same number of packets, we set T_r to 0 and d_i to 50 for all sensor nodes. We didn't implement PrimalDual because its design is very complicated for deriving the approximation ratio. Instead, since it is a 2-approximation, we assume that one-half of the packets collected by the optimal ILP(A) are collected by the PrimalDual. Fig. 5 shows that PCA and SCA collect more packets than PrimalDual, showing that both are competitive data-collecting algorithms.

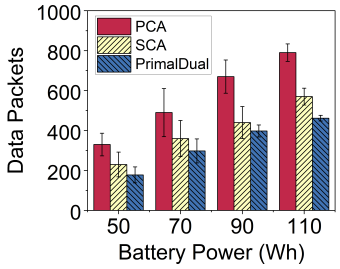


Fig. 5. Comparing with PrimalDual.

VII. Conclusions

We propose a new algorithmic framework called the budget-constrained covering salesman problem (i.e., BC-CSP). BC-CSP is inspired by robotic sensor networks (RSNs), wherein battery-constrained mobile robots, including drones and UAVs, are dispatched to collect sensing data or maintain the network in many sensing applications. We design algorithmic solutions that outperform the state-of-the-art RSN research in solution efficacy and time efficiency. Owing to this theoretical root, our research not only applies to RSNs but also has the potential to impact a wide range of modern robotic systems, where limited battery power severely affects their operations.

Acknowledgment

This work was supported by NSF Grant 2240517.

References

[1] Ibm cplex optimizer. https://www.ibm.com/analytics/cplex-optimizer.
[2] Miller–tucker–zemlin (mtz) subtour elimination constraint. https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html.
[3] A. Amiri and M. Salari. Time-constrained maximal covering routing problem. *OR Spectrum*, 41:415–468, 2019.
[4] E. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *DAMATH: Discrete Applied Math. Combinatorial Operations Res. Comput. Sci.*, 55, 1994.
[5] E. M. Arkin, S. P. Fekete, and J. S.B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1):25–50, 2000.
[6] S. Basagni, L. Boloni, P. Gjanci, C. Petrioli, C. A. Phillips, and D. Turgut. Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle. In *Proc. of INFOCOM*, 2014.
[7] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud. A uav-assisted data collection for wireless sensor networks: Autonomous navigation and scheduling. *IEEE Access*, 8:110446–110460, 2020.
[8] L. Chen, W. Wang, H. Huang, and S. Lin. On time-constrained data harvesting in wireless sensor networks: Approximation algorithm design. *IEEE/ACM Transactions on Networking*, 24(5):3123–3135, 2016.
[9] T. Corman, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2009.
[10] J. R. Current and D. A. Schilling. The covering salesman problem. *Transportation science*, 23:208–213, 1989.
[11] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
[12] B. Golden, Z. Naji-Azimi, S. Raghavan, M. Salari, and P. Toth. The generalized covering salesman problem. *INFORMS Journal on Computing*, 24(4):534–553, 2011.
[13] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of HICSS 2000*.
[14] H. Huang, A. V. Savkin, M. Ding, and C. Huang. Mobile robots in wireless sensor networks: A survey on tasks. *Computer Networks*, 148:1–19, 2019.
[15] H. Li and A. V. Savkin. An algorithm for safe navigation of mobile robots by a sensor network in dynamic cluttered industrial environments. *Robotics and Computer-Integrated Manufacturing*, 54:65–82, 2018.
[16] K. Li, W. Ni, and F. Dressler. Continuous maneuver control and data capture scheduling of autonomous drone in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 21(8):2732–2744, 2022.
[17] X. Liu, T. Qiu, X. Zhou, T. Wang, L. Yang, and V. Chang. Latency-aware path planning for disconnected sensor networks with mobile sinks. *IEEE Transactions on Industrial Informatics*, 16(1):350–361, 2020.
[18] M. Ma, Y. Yang, and M. Zhao. Tour planning for mobile data-gathering mechanisms in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 62(4):1472–1483, 2013.
[19] L. P. Maziero, F. L. Usberti, and C. Cavellucci. Branch-and-cut algorithms for the covering salesman problem. *CoRR*, abs/2104.01173, 2021.
[20] Z. Naji-Azimi and M. Salari. The time-constrained maximal covering salesman problem. *Applied Mathematical Modelling*, 38(15):3945–3957, 2014.
[21] S. Patil, Y. T. Chen, and B. Tang. Revisiting data collection in robotic sensor networks: A budget-constrained traveling salesman perspective. In *Proc. of IEEE MASS*, 2024.
[22] A. Paul, D. Freund, A. Ferber, D. B. Shmoys, and D. P. Williamson. Prize-collecting tsp with a budget constraint. In *25th Annual European Symposium on Algorithms (ESA 2017)*.
[23] K. A. Ravindra, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.
[24] H. Salarian, K. W. Chin, and F. Naghdy. An energy-efficient mobile-sink path selection strategy for wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 63(5):2407–2419, 2014.
[25] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Oper. Res.*, 255(2):315 – 332, 2016.
[26] A. Wichmann, T. Korkmaz, and A. S. Tosun. Robot control strategies for task allocation with connectivity constraints in wireless sensor and robot networks. *IEEE Transactions on Mobile Computing*, 17(6):1429–1441, 2018.
[27] X. Xiao and W. L. Whittaker. Energy considerations for wheeled mobile robots operating on a single battery discharge. Technical Report CMU-RI-TR-14-16, Pittsburgh, PA, August 2014.
[28] C. Yang, C. Gu, T. Zhang, and Y. Gao. Underwater robot sensing technology: A survey. *Fundamental Research*, 1(3):337–345, 2021.