Column Generation-Based Heuristic for the Sweep Coverage Problem

Fabrizio Marinelli
Università Politecnica delle Marche
Ancona, Italy
fabrizio.marinelli@staff.univpm.it

Andrea Pizzuti
Università degli Studi eCampus
Novedrate, Italy
andrea.pizzuti@uniecampus.it

Nicola Ronchini
Università Politecnica delle Marche
Ancona, Italy
n.ronchini@pm.univpm.it

Abstract

The sweep coverage problem concerns the planning of routes for mobile sensors, such as drones, to periodically visit a set of points of interest while minimizing the total travelled distance. Although several approaches have been proposed, existing methods are mostly heuristic or approximate and rarely consider sensing ranges and coverage redundancy simultaneously. In this work, we propose a cycle-based integer programming formulation that explicitly incorporates both features. The problem is approached through a master-pricing decomposition: the master selects feasible routes subject to redundancy constraints, while the pricing problem generates profitable ones under a length constraint. These routes are computed using a GRASP-based heuristic and, to certify optimality, a TSP-like formulation with profits and sensing ranges. Primal and dual bounds are obtained via a price-and-branch scheme. Computational experiments on randomly generated instances demonstrate the effectiveness of the proposed approach.

Keywords

Sweep Coverage, Integer Programming, Column Generation, Heuristic

1 Introduction

The planning of mobile sensors, such as drones or unmanned aerial vehicles, has gained increasing attention in recent years. This class of problems arises in a wide range of application domains, including indoor agriculture, emergency response planning, and wireless sensor networks, and involves deploying a fleet of mobile sensors to perform tasks such as data collection from points of interest (POIs) or collection and distribution of goods from and to clients, while minimizing the total traveled distance or operational cost. Among these problems, the coverage problem, where a fleet of drones is required to ensure complete coverage of a set of POIs within a monitored area, has become particularly relevant.

Three main types of coverage problems have been investigated: full coverage, barrier coverage, and sweep coverage [2]. This work focuses on the latter, which specifically aims to determine routes of minimum total length or, alternatively, the minimum number of mobile sensors required to periodically visit a set of POIs. Typical applications of the sweep coverage (SCP) include police patrolling, environmental monitoring, and farming field surveillance. Several procedures have been proposed for the SCP under both homogeneous and heterogeneous sweep period assumptions, including two centralized approximation algorithms and a distributed heuristic in [2], as well as an approximation algorithm and a greedy heuristic presented in [7].

More recently, the SCP has been extended to incorporate practical requirements, such as drone sensing radii and coverage redundancy. The former accounts for the fact that a mobile sensor can cover POIs within a non-zero sensing range around its ground projection, as discussed in [8]. In that work, the authors propose two approximation algorithms and a distributed solution to minimize the trajectory length of a single mobile sensor. The latter extension, introduced in [13], requires that each POI be covered by multiple mobile sensors to enhance robustness against sensor failures. For this variant, called the (t, K)-SCP, where t denotes the sweep period and K the redundancy requirement, the authors propose a time- and position-based partitioning procedure. Additional variants of the SCP have also been studied, including the chargeable-SCP, which introduces charging stations to allow mobile sensors to replenish their batteries while performing routes that start and end at the depot [12]. Comprehensive surveys of coverage problems are available in [5, 17]. As highlighted in [19], the SCP can be viewed as a variant of the classical vehicle routing problem (VRP) [20]. The authors addressed the SCP heuristically by adopting a VRP-based framework that combines insertion heuristics with simulated annealing. Nevertheless, the classical VRP does not account for vehicle sensing ranges or coverage redundancy requirements. The first aspect is partially captured by a generalized version of the VRP, in which customers are grouped into clusters and a cluster is considered served (covered) if at least one customer within the cluster is visited [10]. In fact, this generalized VRP can be interpreted as a discrete version of the SCP with sensing radii.

To the best of our knowledge, the sweep coverage with heterogeneous sensing ranges and explicit redundancy requirements has not yet been addressed in the literature. Moreover, existing solution approaches mainly rely on heuristic (approximation) algorithms with no (coarse) performance guarantees. Motivated by this gap, we propose a cycle-based integer programming formulation for the SCP that simultaneously incorporates sensing range and redundancy constraints. Similarly to integer programming-based exact approaches for the VRP, the problem is decomposed into a master–pricing scheme: the master problem selects a minimum-length set of sensor routes that satisfy redundancy requirements, whereas the pricing problem generates promising routes by trading off travel cost against profits from POIs falling within the sensor sensing range. The pricing problem, formulated as a mixed-integer nonlinear program, represents a novel variant of the traveling salesman problem (TSP) with profits, incorporating *i*) the objective structure of the profitable tour problem [6], *ii*) the autonomy (resource) constraint typical of the orienteering problem [9], and *iii*) non-zero sensing ranges, as in close-enough extensions of the TSP [1, 3, 16] and the orienteering problem [18, 21].

Dual bounds to the SCP are obtained by solving the linear relaxation of the cycle-based formulation via a column generation (CG) scheme. Primal solutions are then computed through price-and-branch (P&B). Moreover, as the exact solution of the pricing

INOC '26, Liège (Belgium)

10.48786/inoc.2026.17

problem quickly becomes computationally demanding for large instances, a heuristic procedure combining GRASP and local search is employed to accelerate its solution.

2 CYCLE-BASED FORMULATION

We consider a scenario consisting of a set of n POIs $P = \{1, \dots, n\}$, the i-th of which is located at coordinates $(\bar{x}_i, \bar{y}_i) \in \mathbb{R}^2_+$. Each POI must be covered by mobile sensors at least d times in order to ensure redundancy. The sweep coverage is performed by a fleet of k mobile sensors, such as quadrotors or other multirotor drones, equipped with onboard sensing and computing capabilities that enable fully autonomous operation. At any given time, each mobile sensor can cover a circular area of radius r centred at the ground projection of its position. Therefore, to cover the POI i, the sensor must be located at a point within the region

$$D^i := \left\{(x, y) \in \mathbb{R}^2_+ : \|(x, y) - (\bar{x}_i, \bar{y}_i)\|_2 \le r\right\}.$$

All mobile sensors are assumed to move at a constant and identical velocity, maintain the same altitude, and operate within a maximum allowable travel distance l. This distance limit depends on factors such as battery capacity, energy consumption rate, and the required coverage frequency of each POI.

The SCP asks to select at most k feasible cyclic paths, one for each mobile sensor, that minimize the total travel length while ensuring that each POIs is covered the number of times specified by the redundancy requirement. The initial cost for a mobile sensor to reach the starting position of its route is considered negligible; therefore, no depot node is defined. Each route is a simple cycle $\mathbf{t} = \langle p_1, \dots, p_m, p_1\rangle$ described by an ordered sequence of visited POIs $p_i \in P$, $1 \le i \le m$, with $3 \le m \le n$ and $p_i \ne p_j$ for all $1 \le i < j \le m$. Each cycle is embedded in the plane as an ordered sequence of points $\langle (x_{p_1}, y_{p_1}), \dots, (x_{p_m}, y_{p_m}), (x_{p_1}, y_{p_1})\rangle$ with $(x_{p_i}, y_{p_i}) \in D^{p_i}$. Moreover, the total length of the cycle must not exceed the maximum allowable travel distance l, namely,

$$l_{\mathbf{t}} = \sum_{i=1}^{m} \|(x_{p_i}, y_{p_i}) - (x_{p_{i+1}}, y_{p_{i+1}})\|_2 \le l,$$

with $(x_{p_{m+1}}, y_{p_{m+1}}) := (x_{p_1}, y_{p_1})$.

Let T be the set of feasible routes, and let $x_{\mathbf{t}} \in \mathbb{Z}_+$ be the integer variable indicating the number of mobile sensors moving along route $\mathbf{t} \in T$. Moreover, let $a_{i\mathbf{t}} = 1$ if the i-th POI is covered by route $\mathbf{t} \in T$, and $a_{i\mathbf{t}} = 0$ otherwise. The cycle-based integer linear programming formulation reads as follow:

$$[\mathcal{M}] \quad \min \sum_{\mathbf{t}\in T} l_{\mathbf{t}} x_{\mathbf{t}} \tag{1}$$

$$\text{s.t.} \sum_{\mathbf{t}\in T} a_{i\mathbf{t}} x_{\mathbf{t}} \ge d \qquad \forall i \in P \tag{2}$$

$$\sum_{\mathbf{t}\in T} x_{\mathbf{t}} \le k \tag{3}$$

$$x_{\mathbf{t}} \in \mathbb{Z}_+ \qquad \forall \mathbf{t} \in T \tag{4}$$

The objective function (1) minimizes the total distance traveled by the mobile sensors. Constraints (2) ensure that each POI is covered at least d times, while constraints (3) enforce the limit on the maximum number of mobile sensors deployed for the coverage.

The cardinality of T generally grows exponentially with the size of the problem. Therefore, a CG approach is required to dynamically generate the routes as needed [14].

Let $[\mathcal{R}]$ denote the restricted master problem, i.e., the master problem $[\mathcal{M}]$ restricted to a (possibly empty) subset of columns $T_{\mathcal{R}} \subseteq T$, with the integrality constraints (4) relaxed. Let $\boldsymbol{\lambda} \in \mathbb{R}^{|P|}_+$ and $\omega \in \mathbb{R}_-$ denote the dual variables associated with constraints (2) and (3), respectively.

Let $(\bar{\boldsymbol{\lambda}}, \bar{\omega})$ be an optimal dual solution of the restricted master problem $[\mathcal{R}]$. The reduced cost associated with cycle $\mathbf{t}$ is then given by

$$l_{\mathbf{t}} - \sum_{i\in\mathbf{t}} \bar{\lambda}_i - \bar{\omega} \tag{5}$$

The pricing problem computes a profitable cycle $\mathbf{t}$ in a complete undirected graph $G = (P, E)$, where the nodes correspond to POIs. Considering the maximum allowable cycle length l, edges can be filtered as:

$$\bar{E} := \{\{i, j\} \in E : \|(\bar{x}_i, \bar{y}_i) - (\bar{x}_j, \bar{y}_j)\| - 2r \le \frac{l}{2}\}.$$

Let $v_i \in \{0, 1\}$ be a binary variable associated with POI i, where $v_i = 1$ if i is covered by the cycle $\mathbf{t}$, and $v_i = 0$ otherwise. Furthermore, for all $\{i, j\} \in \bar{E}$, let $z_{ij} \in \{0, 1\}$ denote the binary variable indicating that edge $\{i, j\}$ is in the cycle. Let $x_i, y_i \in \mathbb{R}_+$ be continuous variables representing the point in D^i chosen to cover POI i, and let $l_{ij} \in \mathbb{R}_+$ be the Euclidean distance between points (x_i, y_i) and (x_j, y_j) along the cycle. Finally, let $\delta(S)$ be the cut induced by $S \subset P$, i.e., the set of edges with exactly one endpoint in S.

$$\delta(S) := \{\{i, j\} \in \bar{E} : i \in S, j \notin S\}.$$

The pricing problem is as follow:

$$[\mathcal{P}] \quad \min \sum_{\{i,j\}\in\bar{E}} l_{ij} z_{ij} - \sum_{i\in P} \bar{\lambda}_i v_i - \bar{\omega} \tag{6}$$

$$\text{s.t.} \sum_{j:\{i,j\}\in\bar{E}} z_{ij} = 2v_i \qquad \forall i \in P \tag{7}$$

$$\sum_{\{u,w\}\in\delta(S)} z_{uw} \ge 2 \cdot (v_i + v_j - 1) \qquad \forall S \subset P,\ i \in S, j \in P \setminus S \tag{8}$$

$$\sum_{\{i,j\}\in\bar{E}} l_{ij} z_{ij} \le l \tag{9}$$

$$\|(x_i, y_i) - (\bar{x}_i, \bar{y}_i)\|_2 \le r^2 \qquad \forall i \in P \tag{10}$$

$$\|(x_i, y_i) - (x_j, y_j)\|_2 \le l^2_{ij} \qquad \forall \{i, j\} \in \bar{E} \tag{11}$$

$$v_i \in \{0, 1\} \qquad \forall i \in P \tag{12}$$

$$z_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in \bar{E} \tag{13}$$

$$l_{ij} \in \mathbb{R}_+ \qquad \forall \{i, j\} \in \bar{E} \tag{14}$$

$$x_i, y_i \in \mathbb{R}_+ \qquad \forall i \in P \tag{15}$$

The objective function (6) minimizes the reduced cost defined in (5), namely the difference between the cycle length and the profits collected from covered POIs. Constraints (7) and (8) ensure that the solution corresponds to a simple cycle; notice that the subtour elimination inequalities (8) are active only when the relevant POIs i and j are covered. Inequality (9) limits the total length of the cycle. Finally, for each POI, constraints (10) ensure that the selected point (x_i, y_i) lies within the corresponding coverage region D^i, whereas (11) compute the distance between each pair of selected points. Note that setting $v_i = 0$ makes constraint (8) redundant, allowing $z_{ij} = 0$. Consequently, the variables l_{ij} do not affect either constraint (11) or the objective function (6), i.e., the choice of point (x_i, y_i) becomes irrelevant when node i is not part of the cycle.

If the optimal value of $[\mathcal{P}]$ is negative, then the corresponding cycle is profitable, i.e., it corresponds to a columns with negative reduced cost that can be added to $[\mathcal{R}]$ to potentially improve its optimal value. When no improving cycle exists, the CG procedure terminates with the optimal solution of the linear relaxation of $[\mathcal{M}]$, which serves as a valid dual bound. A primal bound is then computed using a P&B procedure: the integrality of the restricted master problem variables is restored, and an integer primal solution is obtained via branch-and-bound.

The generalization of $[\mathcal{M}]$ to accommodate heterogeneous mobile sensors can be obtained by typifying the sensors and partitioning the set of feasible cycles accordingly. Consequently, multiple pricing problems $[\mathcal{P}_h]$, one for each sensor type h, are solved at each iteration, with only minor differences. When mobile sensors differ in autonomy and sensing ranges, constraints (9) and (10) in $[\mathcal{P}]$ are adapted to account for the maximum travel distance l_h and sensing radius r_h of sensor type h, respectively. Moreover, if sensor type h can perform only specific measurements, the pricing problem $[\mathcal{P}]$ considers only the POIs requiring those measurements.

2.1 An Outer-Approximation Approach

The pricing problem $[\mathcal{P}]$ is a generalization of the profitable tour problem and is therefore strongly NP-hard [6]. Moreover, it must be solved multiple times during the CG process.

To accelerate the exact solution of $[\mathcal{P}]$, the mixed-integer nonlinear program is tackled via an outer-approximation approach $\mathcal{A}$ [4]. First, bilinear terms in (6) and (9) are linearized through standard McCormick technique [15]. Then, conic constraints (10) and (11) are removed to obtain a mixed-integer linear programming relaxation $[\tilde{\mathcal{P}}]$. Solving $[\tilde{\mathcal{P}}]$ provides $(\tilde{\mathbf{v}}, \tilde{\mathbf{z}}, \tilde{\mathbf{l}}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, namely a cycle $\tilde{\mathbf{t}}$, and a valid lower bound on the optimal value of $[\mathcal{P}]$. The optimal positions and distances are computed by solving the following second-order cone program (SOCP):

$$\min \quad \sum_{i=1}^{|\tilde{\mathbf{t}}|-1} l_{i,i+1} - \sum_{i \in P} \bar{\lambda}_i \tilde{v}_i - \bar{\omega} \tag{16}$$

$$\text{s.t.} \quad \begin{cases} \|(x_i, y_i) - (\bar{x}_i, \bar{y}_i)\|_2 \le r^2 \\ \|(x_i, y_i) - (x_{i+1}, y_{i+1})\|_2 \le l_{i,i+1}^2 \\ l_{i,i+1} \in \mathbb{R}_+ \\ x_i, y_i \in \mathbb{R}_+ \end{cases} \quad \forall i = 1, \ldots, |\tilde{\mathbf{t}}| - 1. \tag{17}$$

A SOCP solution $(\mathbf{l}, \mathbf{x}, \mathbf{y})$ yields a valid upper bound for $[\mathcal{P}]$ only when $\mathbf{l} \cdot \tilde{\mathbf{z}} \le l$. If the upper bound does not exceed the lower bound obtained from $[\tilde{\mathcal{P}}]$, then $(\tilde{\mathbf{v}}, \tilde{\mathbf{z}}, \mathbf{l}, \mathbf{x}, \mathbf{y})$ is an optimal solution to $[\mathcal{P}]$. Otherwise, linear inequalities are enforced in $[\tilde{\mathcal{P}}]$ for outer-approximating (10) and (11) via first-order linearization at the point $(\mathbf{l}, \mathbf{x}, \mathbf{y})$. Moreover, an integer cut is added to $[\tilde{\mathcal{P}}]$ to exclude $\tilde{\mathbf{t}}$.

3 HEURISTIC FOR SOLVING $[\mathcal{P}]$

A heuristic $\mathcal{H}$ is designed to rapidly compute feasible, although possibly suboptimal, solutions of $[\mathcal{P}]$. The proposed heuristic incorporates two distinct phases: a construction phase, aimed at generating an initial feasible solution, and an improvement phase, used to refine both the initial solution and the positions (x_i, y_i) of covered POIs.

The construction phase is performed using a GRASP-based procedure, see Algorithm 1.

Algorithm 1 GRASP-based heuristic

Require: $G = (P, \bar{E})$, $\bar{\boldsymbol{\lambda}}$, l, N_G
Ensure: $\mathbf{t}_G$

```
l_{t_G} ← +∞, Λ_{t_G} ← 0
p_1 ← arg max_{i∈P}{λ̄_i}
for κ = 1 to N_G do
    (α, l̂) ←setParams(κ, N_G, l)
    (t̂, m) ←init(λ̄)
    repeat
        s ←computeScore(G, λ̄)
        RCL← populateRCL(s, α, l̂)
        p_m ~ Uniform(RCL)
        t̂ ← ⟨t̂, p_m⟩
        m ← m + 1
    until RCL = ∅
    t ← ⟨t̂, p_1⟩
    while l_t > l do
        (p_h, t) ←removeNode(G, t, λ̄)
    end while
    repeat
        (h, p_h, t) ←addNode(G, t, λ̄, l)
    until p_h = 0
    t ←2Exchange(G, t, l)
    if l_t − Λ_t < l_{t_G} − Λ_{t_G} then
        t_G ← t
    end if
end for
```

The procedure starts by selecting the POI $p_1 = \arg\max_{i \in P}\{\bar{\lambda}_i\}$, where $\bar{\lambda}_i$ are the optimal dual variables of the restricted master problem $[\mathcal{R}]$ regarded as profits associated with the nodes P. Then, the current partial tour $\hat{\mathbf{t}}$ is iteratively extended by adding one node at a time while preserving feasibility with respect to the tour length constraint. Let $\hat{\mathbf{t}} = \langle p_1, \ldots, p_{m-1} \rangle$ denote the current partial tour and $\Lambda_{\hat{\mathbf{t}}}$ be its total profit. Note that $\Lambda_{\hat{\mathbf{t}}}$ is not only the sum of the profits of the nodes in $\hat{\mathbf{t}}$, but also includes the profits collected from POIs whose covering regions D^i intersect the edges of the cycle. At each iteration, candidate nodes $i \in \hat{P} = \{j \in P \setminus \hat{\mathbf{t}} : \{p_{m-1}, i\}, \{i, p_1\} \in \bar{E}\}$ are evaluated according to the score

$$s_i = \frac{\bar{\lambda}_i}{d(i, p_{m-1})} \cdot \bar{s}_i, \tag{18}$$

which resembles the classical profit-cost opportunity ratio. Here, $d(i, p_{m-1})$ is the Euclidean distance between nodes i and p_{m-1}, and $\bar{s}_i$, computed as

$$\bar{s}_i = \frac{1}{\sum_{j:\{i,j\} \in \bar{E}} \bar{d}(i, j)},$$

measures the closeness of node i to the other nodes, allowing nodes with lower profit but better connectivity to be preferred over highly profitable yet relatively isolated nodes. The term $\bar{d}(i, j)$ is the distance $d(i, j)$ normalized by the maximum length of any edge in $\bar{E}$.

A restricted candidate list (RCL) is constructed by retaining only the candidate nodes i whose score is greater than

$$(1 - \alpha) \max_{j \in \hat{P}}\{\bar{s}_j\} + \alpha \min_{j \in \hat{P}}\{\bar{s}_j\},$$

and such that appending the path $\langle i, p_1 \rangle$ to $\hat{\mathbf{t}}$ results in a cycle whose total length $l_{\hat{t}}$ does not exceed $\hat{l}$. Parameter $\alpha \in [0, 1]$ controls the degree of greediness of the construction phase. As

α approaches 1, a larger number of low-quality candidates is included in the RCL, thereby fostering diversification. Conversely, as α approaches 0, the procedure converges to a purely greedy algorithm. On the other hand, parameter $\hat{l} \geq l$ allows, at this stage, infeasible cycles that can later potentially be turned into feasible ones.

Once the RCL has been generated, node p_m is randomly selected from the list and appended to $\hat{\mathbf{t}}$. The procedure for extending the partial tour is repeated as far as new candidate nodes are identified, and the corresponding RCL is not empty. Then, a simple cycle $\mathbf{t}$ is obtained by connecting p_m to p_1. If $\mathbf{t}$ is infeasible due its length, nodes are removed iteratively according to the following criterion [11]:

$$p_h = \text{argmin}_{i \in \mathbf{t} \setminus \{p_1\}} \frac{\bar{\lambda}_i}{d(i,i+1) + d(i,i-1) - d(i-1,i+1)}. \quad (19)$$

Here, the denominator measures the marginal increment yield by node i to $l_{\mathbf{t}}$. Thus, at each step, the node selected for removal is the one whose exclusion most improves the feasibility of the tour while minimizing the loss of collected profit.

Once feasibility is restored, insertion moves are evaluated using the same marginal increment criterion, but this time with the goal of increasing the total collected profit $\Lambda_{\mathbf{t}}$: for each candidate node not in the current cycle, ratios in (19) are computed for all insertion positions. The move that yields the largest ratio while maintaining cycle feasibility is performed. The insertion phase ends when no feasible insertion is found. A final 2-exchange local search is performed to find improved feasible solutions.

After N_{G} iterations, the best cycle $\mathbf{t}_{\text{G}}$ according to (6) is returned and serves as input to the improvement phase (see Algorithm 2) which aims to find cycles with higher total profit and shorter length, primarily by exploiting the covering regions D^i. Indeed, Algorithm 1 assumes $(x_i, y_i) = (\bar{x}_i, \bar{y}_i)$, but significantly shorter cycles can be obtained by allowing $(x_i, y_i) \in D^i$.

Algorithm 2 SOCP-based heuristic

Require: $G = (P, \bar{E})$, $\mathbf{t}_{\text{G}}$, $\bar{\boldsymbol{\lambda}}$, l, r, N_{S}
Ensure: $\mathbf{t}_{\text{S}}$
$\mathbf{t}_{\text{S}} \leftarrow \mathbf{t}_{\text{G}}$
for $\kappa = 1$ to N_{S} **do**
  $\mathbf{t} \leftarrow \mathbf{t}_{\text{S}}$
  $(p_h, \mathbf{t}) \leftarrow$ removeNode$(G, \mathbf{t}, \bar{\boldsymbol{\lambda}})$
  $(q, p_q, \mathbf{t}) \leftarrow$ addNode$(G, \mathbf{t}, \bar{\boldsymbol{\lambda}}, +\infty)$
  $F \leftarrow$ updateTL(F, τ, p_h, p_q)
  $\mathbf{t} \leftarrow$ 2Exchange$(G, \mathbf{t}, l)$
  if $\Lambda_{\mathbf{t}_{\text{S}}} < \Lambda_{\mathbf{t}}$ **then**
    $\mathbf{t} \leftarrow$ solveSOCP$(G, \mathbf{t}, r)$
    if $l_{\mathbf{t}} \leq l \wedge l_{\mathbf{t}} - \Lambda_{\mathbf{t}} < l_{\mathbf{t}_{\text{S}}} - \Lambda_{\mathbf{t}_{\text{S}}}$ **then**
      $\mathbf{t}_{\text{S}} \leftarrow \mathbf{t}$
    end if
  end if
end for

Removal and insertion moves, disregarding the length constraint (9), are applied to the current best cycle $\mathbf{t}$ to obtain cycles with higher total profit. 2-exchange moves are then performed to reduce the cycle length. To avoid local minima, selected nodes are stored in a tabu-list F and forbidden for τ iterations.

In the final step, the optimal positions (x_i, y_i) for each POI i in the cycle $\mathbf{t}$ are computed by solving the SOCP program (16) - (17).

4 COMPUTATIONAL EXPERIMENTS

All computational experiments were conducted on an Intel i5-11400H processor running at 2.70 GHz, equipped with 6 cores and 8 GB of RAM. All algorithms were implemented in C++, and all mathematical formulations were solved using Gurobi Optimizer 12.0.2. The subtour elimination constraints (8) were separated on the fly and enforced as lazy constraints.

The proposed method was tested on 100 randomly generated instances, divided into 20 groups $\{G_1, G_2, \ldots, G_{20}\}$ according to the combinations of the following parameters (see Table 1): the number of POIs to be covered $n \in \{10, 15, 20, 25, 30\}$, the number of available mobile sensors $k \in \{5, 7\}$, and the maximum cycle length $l \in \{30, 40\}$. Parameters r and d are both set equal to 1. Positions of the POIs were uniformly sampled from the squared domain $[0, 25] \times [0, 25] \subset \mathbb{R}^2_+$.

Regarding the heuristic parameters, the values of α are selected from $\{0.2, 0.4, 0.8\}$: $\alpha = 0.2$ and $\alpha = 0.8$ are each used for 15% of the total number of iterations, namely $N_{\text{G}} = 100$, while $\alpha = 0.4$ is used in the remaining 70%. Moreover, $\hat{l}$ takes values in $\{l, \frac{3}{2}l, 2l\}$, each used for one third of the iterations. Finally, in the SOCP-based heuristic, N_{S} and τ are set to 40 and $\lfloor n/10 \rfloor$, respectively.

Primal and dual bounds values are collected at two different timestamps TS, that is 300 and 900 seconds of running time. A time limit TL of 3600 seconds applies to the entire column generation, with a limit of 900 seconds for the outer approximation algorithm $\mathcal{A}$.

4.1 Numerical Results

Table 1 reports computational results aggregated by instance group and for different running times. Each row shows average values over five instances per parameter combination; a dash ("–") denotes a not applicable entry due to early termination of the algorithm.

Column UB[1] reports the averaged primal bound obtained after 300 seconds of running time by solving the integer version of the current restricted master problem. Columns $\text{UB}^{i,j}_{\%}$ list the mean percentage improvement of the primal bound between timestamps i and j, computed as

$$\text{UB}^{i,j}_{\%} = \frac{\text{UB}^i - \text{UB}^j}{\text{UB}^j} \cdot 100$$

Timestamps 2 and 3 correspond to TS = 900 and to TL, respectively.

Column TIME reports the average of the running times for terminated P∂B executions, whose number of cases is reported in column $\overline{\text{CG}}$. If no instances in a given group terminate within the timestamp, either due to lack of CG convergence or excessive time spent solving exactly $[\mathcal{P}]$, the entry LIMIT is reported.

Columns $\text{COL}_{\%}$ and $\text{TIME}_{\%}$ report, respectively, the percentage of columns generated and the percentage of total computational time spent both by the heuristic $\mathcal{H}$ and by the outer-approximation exact approach $\mathcal{A}$. Finally, column $\text{OPT}_{\%}$ reports the optimality gap, computed only for instances where the CG procedure successfully converged, using the primal solution computed by the P∂B (whose running times are negligible).

For $n \leq 20$, the CG algorithm converged within 300 seconds in 86.67% of the cases, increasing to 95.00% before the second timestamp and reaching 98.33% at the third timestamp (TL was reached by only one instance). In particular, the CG terminated within 10 seconds for $n = 10$, within 6 minutes for $n = 15$, and slightly over 15 minutes for $n = 20$, except for one instance. No improvement in the primal bound is observed for instances in

				TS = 300			TS = 900			TL = 3600			$\mathcal{H}$		$\mathcal{A}$		
Group	n	l	k	UB^1	$\overline{CG}$	TIME	$UB^{1,2}_{\%}$	$\overline{CG}$	TIME	$UB^{2,3}_{\%}$	$\overline{CG}$	TIME	$TIME_{\%}$	$COL_{\%}$	$TIME_{\%}$	$COL_{\%}$	$OPT_{\%}$
G_1	10	30	5	54.61	5	2.07	-	-	-	-	-	-	39.66	25.57	59.63	74.43	0.00
G_2	10	40	5	52.86	5	4.98	-	-	-	-	-	-	51.94	44.41	47.84	55.59	0.00
G_3	10	30	7	54.61	5	2.66	-	-	-	-	-	-	35.92	32.46	63.70	67.54	0.00
G_4	10	40	7	52.86	5	5.62	-	-	-	-	-	-	53.63	46.61	46.18	53.39	0.00
G_5	15	30	5	65.15	4	11.96	0.00	1	302.87	-	-	-	32.44	31.03	67.39	68.97	1.07
G_6	15	40	5	64.04	4	109.58	0.00	1	309.12	-	-	-	26.82	50.42	73.12	49.58	0.00
G_7	15	30	7	65.01	4	12.20	0.00	1	306.10	-	-	-	29.91	23.33	72.89	76.67	1.11
G_8	15	40	7	62.10	5	45.64	-	-	-	-	-	-	33.89	49.07	66.03	50.93	0.00
G_9	20	30	5	73.16	5	104.37	-	-	-	-	-	-	8.42	36.84	91.55	63.16	0.00
G_{10}	20	40	5	75.32	2	211.78	9.88	1	449.37	5.33	1	905.66	4.36	60.39	95.67	39.61	0.00
G_{11}	20	30	7	74.03	4	39.84	13.52	1	318.77	-	-	-	30.58	50.63	69.17	49.37	0.17
G_{12}	20	40	7	72.60	4	147.80	52.50	0	LIMIT	0.00	1	957.37	7.09	56.71	92.89	43.29	0.84
G_{13}	25	30	5	89.85	0	LIMIT	1.88	1	596.42	0.01	3	2258.21	2.69	54.02	97.30	45.97	0.11
G_{14}	25	40	5	87.42	0	LIMIT	0.67	0	LIMIT	0.77	0	LIMIT	0.73	54.85	99.27	45.15	-
G_{15}	25	30	7	81.54	2	238.71	0.00	2	591.31	0.00	1	2461.75	2.25	40.06	97.74	59.94	0.38
G_{16}	25	40	7	87.95	0	LIMIT	6.73	0	LIMIT	0.67	0	LIMIT	3.53	59.80	96.46	40.20	-
G_{17}	30	30	5	106.55	0	LIMIT	7.02	0	LIMIT	4.23	0	LIMIT	4.41	74.19	95.58	25.81	-
G_{18}	30	40	5	121.07	0	LIMIT	4.27	0	LIMIT	9.53	0	LIMIT	0.51	92.04	99.49	7.96	-
G_{19}	30	30	7	94.17	0	LIMIT	4.37	1	847.62	5.47	0	LIMIT	0.91	64.74	99.08	35.26	1.24
G_{20}	30	40	7	123.18	0	LIMIT	5.83 (105.11 †)	0	LIMIT	13.48	0	LIMIT	0.56	90.37	99.44	9.63	-

Table 1: Results of the P&B at different timestamps.
†: UB^2 of the instance where UB^1 is unavailable.

groups G_5-G_7, suggesting that the additional running time was primarily spent on certifying the convergence of the CG. For $n \leq 20$, optimality was proved in 86.67% of the instances, while the mean (maximum) optimality gap for the unsolved instances was 2.66% (5.56%).

The running time of the CG procedure grows significantly for $n \geq 25$. In fact, the linear relaxation of $[\mathcal{M}]$ was optimally solved in nine out of twenty cases when $n = 25$, with just five instances spending less than 900 seconds; for $n = 30$, only in one case the CG converged. However, the mean improvements $UB^{1,2}_{\%}$ and $UB^{2,3}_{\%}$ are consistent across the timestamps, showing that the quality of the primal bound steadily improves for increasing computational time. Feasible solutions obtained by the P&B resulted to be optimal in seven out of the ten cases, recording an optimality gap of at most 1.24%.

The CG algorithm generally struggles to converge as the number of POIs grows, suggesting that the current implementation should be refined by incorporating acceleration, stabilization, and early termination techniques [14].

While no statistically significant effect is observed for different values of k, instances become more challenging when l is larger. On the average, solving the instances with $l = 40$ required roughly 5 times more computational time than those with $l = 30$ when $n \leq 20$, whereas for $n \geq 25$ the linear relaxation is solved to optimality only in cases with $l = 30$. This behaviour can be ascribed to the additional time spent to exactly solve $[\mathcal{P}]$ when $l = 40$, which also resulted in fewer generated columns.

As expected, the average percentage of time spent by $\mathcal{H}$ is substantially lower than that required by $\mathcal{A}$, respectively accounting for 18.17% and 81.75% of the total running time. The gap becomes more pronounced as the instance size increases, with $\mathcal{H}$ spending less than 5% of the total computational time when $n \geq 25$. Nevertheless, $\mathcal{H}$ contributed to 51.66% of the overall profitable columns, with an increasing share for larger values of n, where exactly solving $[\mathcal{P}]$ becomes more challenging, thereby demonstrating its significant impact within the P&B scheme.

Additional experiments were performed with $r = 2$ and $d = 2$. Only instances up to $n = 20$ were considered, as the CG consistently reached the CPU time limit for larger instances. When $r = 2$, the problem becomes significantly more challenging due to the increased complexity of the pricing problem. In particular, column generation converged in only 45% of instances, compared to 95% of those with $n = 20$ when $r = 1$. For redundancy level $d = 2$, tests were performed using 6 or 8 mobile sensors, with redundancy assigned to half of the sensors selected randomly. The results show no significant increase in total computational time, suggesting that the algorithm is reliable when solving higher-redundancy instances.

5 CONCLUSIONS

We addressed a sweep coverage problem (SCP) in which a set of POIs must be monitored by a fleet of mobile sensors, with the objective of minimizing the total travel length of the fleet. Coverage requirements enforce measurement redundancy, and each mobile sensor is characterized by a circular sensing region of fixed radius and a maximum allowable route length.

A cycle-based integer programming formulation was proposed, in which the SCP is decomposed into a master problem $[\mathcal{M}]$, selecting a minimum-length subset of feasible cyclic routes that satisfy the redundancy requirements, and a pricing problem $[P]$, defined as a novel variant of the TSP with profits, which generates promising routes by trading off travel cost against profits collected from covered POIs.

To efficiently generate promising routes, we designed a two-phase heuristic combining a GRASP-based construction phase and an improvement phase based on a second-order cone program. The heuristic is embedded within a price-and-branch scheme to compute valid primal and dual bounds.

The computational study on randomly generated instances assessed that *i*) on limited-size instances, the procedure is able to identify optimal solutions in most cases, exhibiting small optimality gap otherwise; *ii*) for larger instances, the column generation procedure fails to converge within a time limit of one hour, highlighting a need for improvement in scalability; *iii*) solving the SCP becomes more challenging as the value of l grows.

As perspective, we aim to improve the efficiency of the price-and-branch by refining the implementations of the column generation and the two-phase pricing heuristic. Moreover, we plan to strengthen the formulation $[\mathcal{P}]$ through the separation of problem-specific valid inequalities, including cover and path inequalities, and by investigating decomposition strategies, such as Benders decomposition. Finally, further computational experiments will be conducted to broaden the analysis of the impact of the sensing radius r and the redundancy coefficient d on solving SCP, as well as to extend the computational campaign to instances derived from real-world applications.

References

[1] Behnam Behdani and J. Cole Smith. 2014. An Integer-Programming-Based Approach to the Close-Enough Traveling Salesman Problem. *INFORMS J. Comput.* 26, 3 (2014), 415–432. doi:10.1287/IJOC.2013.0574

[2] Wei-Fang Cheng, Kebin Liu, Yunhao Liu, Xiang-Yang Li, and Xiang-Ke Liao. 2011. Sweep Coverage with Mobile Sensors. *IEEE Trans. Mob. Comput.* 10 (2011), 1534–1545. doi:10.1109/TMC.2010.237

[3] Walton Pereira Coutinho, Roberto Quirino do Nascimento, Artur Alves Pessoa, and Anand Subramanian. 2016. A Branch-and-Bound Algorithm for the Close-Enough Traveling Salesman Problem. *INFORMS J. Comput.* 28, 4 (2016), 752–765. doi:10.1287/IJOC.2016.0711

[4] Marco A. Duran and Ignacio E. Grossmann. 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming* 36, 3 (1986), 307–339. doi:10.1007/BF02592064

[5] Riham Elhabyan, Wei Shi, and Marc St-Hilaire. 2019. Coverage protocols for wireless sensor networks: Review and future directions. *Journal of Communications and Networks* 21, 1 (2019), 45–60. doi:10.1109/JCN.2019.000005

[6] Dominique Feillet, Pierre Dejax, and Michel Gendreau. 2005. Traveling Salesman Problems with Profits. *Transp. Sci.* 39, 2 (2005), 188–205. doi:10.1287/TRSC.1030.0079

[7] Yuchen Feng, Xiaofeng Gao, Fan Wu, and Guihai Chen. 2015. Shorten the Trajectory of Mobile Sensors in Sweep Coverage Problem. In *2015 IEEE Global Communications Conference (GLOBECOM)*. 1–6. doi:10.1109/GLOCOM.2015.7417353

[8] Xiaofeng Gao, Zhiyin Chen, Jianping Pan, Fan Wu, and Guihai Chen. 2020. Energy Efficient Scheduling Algorithms for Sweep Coverage in Mobile Sensor Networks. *IEEE Transactions on Mobile Computing* 19, 6 (2020), 1332–1345. doi:10.1109/TMC.2019.2910074

[9] Bruce L. Golden, Larry Levy, and Rakesh Vohra. 1987. The orienteering problem. *Naval Research Logistics (NRL)* 34, 3 (1987), 307–318. doi:10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D

[10] Ali Aghadavoudi Jolfaei, M. Alinaghian, Roghayeh Bahrami, and E. B. Tirkolaee. 2023. Generalized vehicle routing problem: Contemporary trends and research directions. *Heliyon* 9 (2023). doi:10.1016/j.heliyon.2023.e22733

[11] Gorka Kobeaga, María Merino, and Jose A. Lozano. 2018. An efficient evolutionary algorithm for the orienteering problem. *Computers & Operations Research* 90 (2018), 42–59. doi:10.1016/j.cor.2017.09.003

[12] Dieyan Liang, Baojian Feng, and Xuefeng Liao. 2024. A Path Planning Method for Chargeable Sweep Coverage With Multiple Charging Stations. *IEEE Access* 12 (2024), 34931–34941. doi:10.1109/ACCESS.2024.3373543

[13] Chuang Liu and Hongwei Du. 2021. t, K-Sweep Coverage With Mobile Sensor Nodes in Wireless Sensor Networks. *IEEE Internet of Things Journal* 8, 18 (2021), 13888–13899. doi:10.1109/JIOT.2021.3070062

[14] Marco Lübbecke and Jacques Desrosiers. 2005. Selected Topics in Column Generation. *Operations Research* 53 (2005), 1007–1023. doi:10.1287/opre.1050.0234

[15] Garth P. McCormick. 1976. Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming* 10, 1 (1976), 147–175. doi:10.1007/BF01580665

[16] William Mennell. 2009. *Heuristics for Solving Three Routing Problems: Close-Enough Traveling Salesman Problem, Close-Enough Vehicle Routing Problem, Sequence-Dependent Team Orienteering Problem.* Ph. D. Dissertation. University of Maryland (College Park, Md.).

[17] Shaimaa M. Mohamed, Haitham S. Hamza, and Iman Aly Saroit. 2017. Coverage in mobile wireless sensor networks (M-WSN): A survey. *Computer Communications* 110 (2017), 133–150. doi:10.1016/j.comcom.2017.06.010

[18] Qiuchen Qian, Yanran Wang, and David Boyle. 2024. On solving close enough orienteering problems with overlapped neighborhoods. *European Journal of Operational Research* 318, 2 (2024), 369–387. doi:10.1016/j.ejor.2024.05.032

[19] Li Shu, Wei Wang, Feng Lin, Zhonghao Liu, and Jiliu Zhou. 2013. A Sweep Coverage Scheme Based on Vehicle Routing Problem. *TELKOMNIKA Indonesian Journal of Electrical Engineering* 11 (2013). doi:10.11591/telkomnika.v11i4.2380

[20] Paolo Toth and Daniele Vigo. 2002. *The Vehicle Routing Problem.* Society for Industrial and Applied Mathematics. doi:10.1137/1.9780898718515 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780898718515

[21] Petra Štefaníková, Petr Váňa, and Jan Faigl. 2020. Greedy randomized adaptive search procedure for close enough orienteering problem. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 808–814. doi:10.1145/3341105.3374010