Zhang, W., Wang, K., Wang, S., Laporte, G., 2020. Clustered coverage orienteering problem of unmanned surface vehicles for water sampling. Naval Research Logistics 67, 353–367.

Clustered coverage orienteering problem of unmanned surface vehicles for water sampling

Wei Zhang
Department of Logistics and Maritime Studies, Hong Kong Polytechnic University, Hong Kong;
wei.sz.zhang@connect.polyu.hk

Kai Wang
Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA;
cwangkai@mit.edu

Shuaian Wang
Department of Logistics and Maritime Studies, Hong Kong Polytechnic University, Hong Kong; wangshuaian@gmail.com

Gilbert Laporte
Department of Decision Sciences, HEC Montréal, Montréal, Québec, Canada; gilbert.laporte@cirrelt.ca

This study investigates a clustered coverage orienteering problem (CCOP), which is a generalization of the classical orienteering problem. The problem is widely motivated by the emerging unmanned techniques (e.g., unmanned surface vehicles and drones) applied to environmental monitoring. Specifically, the unmanned surface vehicles (USVs) are used to monitor reservoir water quality by collecting samples. In the CCOP, the water sampling sites (i.e., the nodes) are grouped into clusters, and a minimum number of nodes must be visited in each cluster. With each node representing a certain coverage area of the water, the objective of the CCOP is to monitor as much as possible the total coverage area in one tour of the USV, considering that overlapping areas provide no additional information. An integer programming model is first formulated through a linearization procedure that captures the overlapping feature. A two-stage exact algorithm is proposed to obtain an optimal solution to the problem. The efficiency and effectiveness of the two-stage exact algorithm are demonstrated through experiments on randomly generated instances. The algorithm can effectively solve instances with up to 60 sampling sites.

Key words: unmanned surface vehicle; clustered coverage orienteering problem; exact algorithm; water sampling.

1. Introduction

The clustered coverage orienteering problem (CCOP) is motivated by the emerging applications of unmanned surface vehicles (USVs) and unmanned aerial vehicles (UAVs) in environmental monitoring, e.g., water pollution monitoring by USVs (Steimle and Hall 2006, Dunbabin et al. 2009, Water Supplies Department 2018) and air pollution monitoring by UAVs (Wivou et al. 2016, Xia et al. 2019). Given a set of nodes representing sampling sites, the CCOP consists of selecting a subset of them to perform monitoring tasks. Each node represents a coverage area and once a node is visited, the information associated with its coverage area is obtained. However, for overlapping

areas associated with visited nodes, no additional information is provided. Moreover, the candidate nodes are grouped into clusters representing regions, and a minimum number of nodes must be visited in each cluster so that efficient information can be gathered in the region. The CCOP aims to design a tour maximizing the information union of the coverage areas associated with the selected nodes, with respect to possible restrictions on the travel time (or distance) and the capacity of the USV.

The CCOP generalizes a problem belonging to the class of the traveling salesman problems with profits (Vidal et al. 2015, Feillet et al. 2005), namely the orienteering problem (OP), which is also known as the selective travelling salesman problems (Laporte and Martello 1990, Gendreau et al. 1998). In the OP, each node has a profit. The aim is to maximize the collected profits associated with the visited nodes subject to a travel time budget. The CCOP differs from the classical OP as follows: (i) the CCOP maximizes the profits of the union of the coverage areas associated with the visited nodes rather than the sum of the profits associated with all coverage areas (overlapping areas count only once); (ii) the CCOP relates to the node clusters, and cluster-dependent restrictions can be imposed. The concept of coverage area is frequently applied in robotics control studies (Choset 2001, Hokayem et al. 2007, Faigl 2010, Fazli et al. 2013, Franco et al. 2015). However, to the best of our knowledge, it has not yet been considered in routing studies.

In this paper, we describe the CCOP to model a USV performing water quality monitoring tasks by taking samples. A detailed illustration of the USV water sampling procedure in practice is provided by Water Supplies Department (2018). The USV departs from the depot, travels through the selected sampling sites (or nodes) to pick up water samples, and then returns to the depot. The CCOP captures the traditional time budget of the OP with respect to the battery depletion of the USV. The capacity budget is included in the CCOP regarding the maximum volume of water samples that can be loaded by the USV. Aside from the budget constraints, two special features are considered. The first feature relates to the "clusters" of sampling sites: the candidate sampling sites belong to different clusters, and at least a given number of sampling sites must be selected in each cluster. Figure 1 represents a water reservoir with three clusters having five, three, and three candidate sampling sites. The second feature relates to the "coverage" of water samples: a water sample collected from each sampling site can represent the water quality information of a circle area called the "coverage (water) area". In Figure 1, repeated information is indicated by the overlapping coverage area of different sampling sites in one region. In particular, three coverage areas in Region 1 overlap with one another, and two coverage areas in Region 3 overlap with each other. The overlapping coverage areas provide no additional water quality information. To maximize the information obtained in one trip performed by the USV, the objective of the CCOP is to maximize the union of the coverage areas of the selected sampling sites.

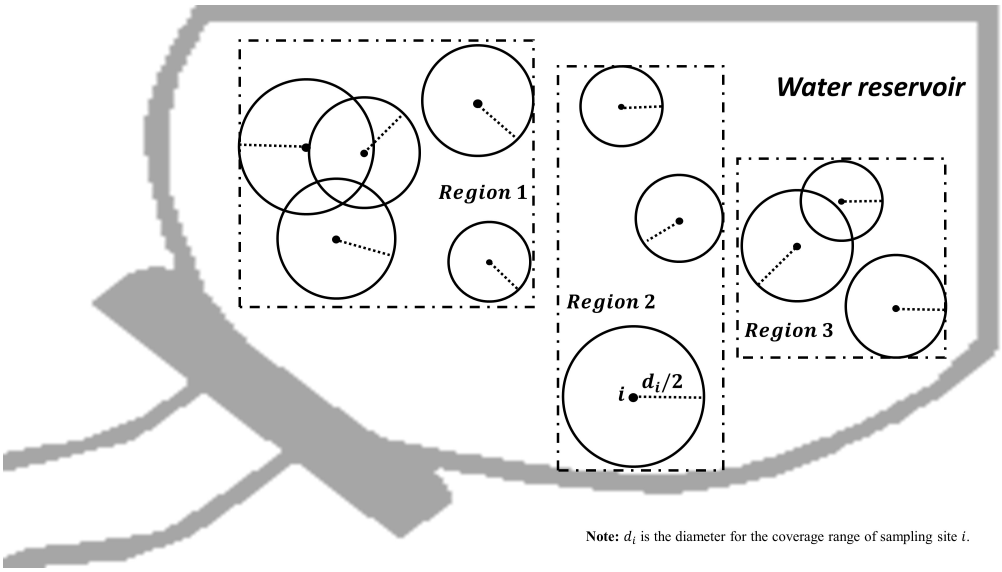

Figure 1 **Example of coverage area of water samplings.**

The concepts of the area coverage and overlapping in the CCOP are self-explanatory. The reason for clustering the candidate sampling sites in the CCOP relates to the spatial characteristics of the reservoir. For instance, eutrophication (i.e., the process by which a body of water becomes enriched in dissolved nutrients that stimulate the growth of aquatic plant life) is the most common problem in a reservoir, whose sensitivity is highly related to the spatial characteristics of the hydrodynamics of the reservoir (e.g., retention ratio and sluggish flow ratio). Even within the same reservoir, the hydrodynamic characteristic varies in different regions. In the Three Gorges Reservoir in China, the flow speed changes from the head region to the tail region: in the head region, the flow speed is much slower, and the nutrition deposits more easily (Zhang et al. 2006). Thus, the head region has a much lower threshold value of eutrophication and algal bloom than the tail region, which yields different nutrition level indices and water quality standards. Henceforth, the reservoir under consideration will be divided into several regions, each having a specific spatial characteristic. A minimum number of sampling sites should be selected in each region to assess the water quality.

1.1. Literature review

The CCOP is related to two mainstreams of studies: the OP for routing and the informative pathing problem for robotics control where the idea of area "coverage" is widely discussed and applied.

The orienteering problem for routing. The CCOP generalizes the OP which was introduced by Tsiligirides (1984) and Golden et al. (1987). Comprehensive reviews of OP have recently been provided by Vansteenwegen et al. (2011) and Gunawan et al. (2016), including recent variants, solution approaches, and applications. The CCOP defined in this paper is most related to the clustered orienteering problem (COP) and the correlated orienteering problem (CorOP).

The COP is an OP variant, in which nodes are clustered. It was investigated by Angelelli et al. (2014) who assumed that the profit associated with a cluster is collected if and only if all nodes of the cluster are visited. The CCOP introduced here generalizes the COP studied by Angelelli et al. (2014). Specifically, not all nodes must be visited in a cluster; in addition, a profit is associated with each node but not with each cluster.

The CorOP was investigated by Yu et al. (2014) and Yu et al. (2016), in which a quadratic score function captures the spatial correlations among points close to each other, and a mixed integer quadratic programming model maximizes the quadratic function, then the proposed model is solved by an off-the-shelf solver. This particular CorOP applies to the planning of a robot tour in order to monitor an environment with spatial correlations. Spatial correlations also exist in the CCOP, captured by overlapping coverage areas of sampling sites of each cluster.

Informative pathing problem for robotics control. Several informative path studies fall in the category of robotics control. Although the focus of the CCOP in this study is routing, we can still draw some insights from the related robotics literature, in which, as in the CCOP, the aim is to monitor the environment.

According to Yu et al. (2016), applications of informative path study belong to several domains, including aerial surveillance (Girard et al. 2004, Nigam and Kroo 2008, Michael et al. 2011), underwater surveillance (Smith et al. 2011), and multidomain surveillance (Grocholsky et al. 2006). Smith et al. (2011) proposed a path planning algorithm and a speed control algorithm for underwater gliders, such that informative trajectories were provided for a glider to persistently monitor a patch of ocean. Information values along the path were maximized, while the deviation from the planned path affected by ocean currents was minimized. As in the CCOP, informative path studies design a path to collect as much as possible of the useful information of the environment.

Informative pathing problems have mostly been defined in the context of monitoring studies, in which several methods are involved (i.e., the graph-based method, minimalism process design, continuously evolving scalar field monitoring, and stochastic data harvesting). The graph-based method is closely related to the CCOP. In graph-based methods of robotics study for path planning, vertices represent regions of interest, edge lengths give travel times between regions, and vertex weights show the importance of each region. Alamdari et al. (2014) depicted the environment as a graph with vertex weights and edge lengths to plan a path for a robot. As the robot repeatedly performs a closed walk on the graph, they define the maximum weighted latency as the maximum time between visits to a vertex, weighted by the importance (vertex weight) of that vertex. The objective is to minimize the "maximum weighted latency" of any vertex of a closed walk of the robot in order to monitor the environment.

Referring to the sampling tasks in the CCOP, a similar approach (i.e., sampling-based approach) is also adopted in informative path studies in robotics area. The sampling-based approach selects observation locations to minimize prediction uncertainty or to maximize some measure of information gain (Karaman and Frazzoli 2011, Hollinger and Sukhatme 2014). For instance, Karaman and Frazzoli (2011) adopted stochastic sampling-based path planning algorithms to measure the quality of solution paths. They analyzed the asymptotic behavior of the solutions at the increase in the number of samples. Hollinger and Sukhatme (2014) found a trajectory that maximizes an information quality metric (e.g., variance reduction, information gain, or mutual information) and that falls within a prespecified budget constraint (e.g., fuel, energy, or time).

The area coverage feature of path planning has been studied in robotics and is also considered in the CCOP. Unlike the conventional point-to-point path planning, the coverage path planning can be applied to demining, floor scrubbing, and inspection, and can also be utilized in the reservoir monitoring discussed in this paper. Choset (2000) determined a path for a robot to pass each point in an area by an exact cellular decomposition approach for the purposes of coverage. Choset (2001) performed a comprehensive review of studies on the coverage for robotics, in which the coverage algorithms are divided into four categories: heuristic, approximate, partial-approximate, and exact cellular decompositions. Additional coverage path problems that mostly emphasized on robotics control can be found in the works of Hokayem et al. (2007), Faigl (2010), Fazli et al. (2013), and Franco et al. (2015).

1.2. Contributions

The scientific contribution of this paper includes the introduction of a new generalization of the OP and the design of a novel exact algorithm to solve it. First, we introduce the CCOP, which is motived by emerging applications in environmental monitoring. It embeds the concepts of clusters and coverage areas into the classical OP and thus generalizes the problem. Second, to solve the CCOP effectively, we design an exact two-stage algorithm based on a new framework. The first stage includes a sampling task generator and two CCOP bounding procedures. The second stage iteratively considers each of the sampling tasks and solves the corresponding travelling salesman problem (TSP) to check whether the travel time required to perform the sampling task exceeds the available budget or not. This is done by applying two bounding procedures before solving the TSP to optimality. On the basis of the feasibility of the current sampling task, each iteration ends running sampling task deleters on the set of sampling tasks not yet considered. This framework can be easily adopted to other problems with a structure similar to that of the CCOP. We test the model and the algorithm on randomly generated instances. Results show that the two-stage exact algorithm generally provides Pareto improvements over the CPLEX implementations, resulting in

superior solutions and shorter computational times. Our algorithm can solve large-size instances with up to 60 nodes and reduce the computational time by over 70% on average compared with CPLEX implementations.

The remainder of this paper is organized as follows. In Section 2, the CCOP is formally defined and is formulated as non-linear and linearized models. The framework of the two-stage exact algorithm designed for the CCOP is described in Section 3. We then specifically explain the first and the second stage of the algorithm in Section 4 and 5, respectively. Section 6 presents computational experiments conducted to assess the effectiveness and the efficiency of the proposed model and algorithm. Section 7 concludes the study.

2. Problem description and model formulation

In this section, we formally define the CCOP for water sampling, and we propose the non-linear and linearized mathematical models.

2.1. Problem description

The CCOP for reservoir water sampling is defined on an undirected graph $G(V^+, A)$. The vertex set $V^+ = V \cup \{s, t\}$ consists of a set of sampling sites V and one depot s with its replica t. A single USV departs from the depot s, travels through the selected sampling sites (to be determined) to take water samples, and then returns to the depot t. An arc $(i, j) \in A$ is the travelling leg between any two vertices for $i \in V \cup \{s\},\ j \in V \cup \{t\},\ i \neq j$, with the travelling time t_{ij}. The travelling times satisfy the triangle inequality. In the CCOP, both time and capacity budgets of the USV are considered:

(i) The time budget refers to the maximum travelling time $T_{\max}$.

(ii) Regarding the capacity budget, we consider that the water sample collected from each site has the same volume, and this budget can be simplified as the maximum number of sampling sites to be selected, denoted as $C_{\max}$.

The detailed descriptions referring to clusters of the CCOP are as follows:

(iii) The whole reservoir is divided into different regions $r \in R$, and each region r has a cluster of candidate sampling sites $V_r \subset V$. Each sampling site belongs to at most one cluster (i.e., $\cup_{r \in R} V_r = V$ and $V_r \cap V_{r'} = \varnothing,\ \forall r, r' \in R,\ r' \neq r$).

(iv) The minimum number of selected sampling sites in each region $r \in R$ is $S_r^{\min}$ to ensure sufficient water samples taken from region r for analysis, where $\sum_{r \in R} S_r^{\min} \leq C_{\max}$ should be satisfied to guarantee the feasibility of the tour. Without loss of generality, we assume that the given time budget $T_{\max}$ is sufficient for the USV to pick up $S_r^{\min}$ water samples in all the regions; otherwise, no feasible solution exists.

Specifications referring to coverage of the CCOP are as follows. As shown in Figure 1, we consider that the water sample collected from each sampling site $i \in V_r$, $r \in R$ indicates the water quality of a certain area, i.e., a circle coverage water area $C_i(d_i)$, with the location of sampling site i as the center and d_i as the diameter for the coverage range. The detailed information of d_i can be collected from historical data. $C_i(d_i)$ represents both the coverage area positionally and the value of the coverage area numerically. Considering that sampling sites $i \in V_r$ jointly monitor one region $r \in R$, the water samples collected from one cluster V_r, $r \in R$ may share common information.

(v) The coverage areas of sampling sites in different regions have no overlap, i.e., no overlap exists between $C_i(d_i)$ of sampling site $i \in V_r$ and $C_i(d_j)$ of $j \in V_{r'}$, $r' \neq r$.

(vi) The coverage area of sampling sites in the same region may overlap with each other, and the overlapping area can only be counted once in the union coverage area. We define $Area(\boldsymbol{y}_r) = \cup_{i \in V_r: \text{if } y_i=1} C_i(d_i)$ as the size of the union coverage area of the selected sampling sites in region r, where binary vector $\boldsymbol{y}_r = (y_i, i \in V_r)$ indicates whether the sampling site $i \in V_r$ is selected or not.

(vii) The objective function can be represented as $\sum_{r \in R} Area(\boldsymbol{y}_r)$ or as $Area(\boldsymbol{y})$, in which vector $\boldsymbol{y} = (y_i, i \in V)$.

2.2. Notations

Some notations are listed as follows.

***Sets and indices*:**

V	Set of all sampling sites.
V^+	Set of all vertices (including the depot s and its replica depot t).
$i,\ j,\ k$	Index of a vertex of V^+.
r	Index of each region.
R	Set of all regions.
V_r	Set of all sampling sites of region $r \in R$.

***Parameters*:**

t_{ij}	Traveling time between vertices i and j.
$T_{\max}$	Time budget.
$C_{\max}$	Capacity budget, maximum number of the selected sampling sites.
$S_r^{\min}$	Minimum number of the selected sampling sites in region $r \in R$.

***Decision variables*:**

x_{ij}	Binary, set to one if vertex i is immediately followed by vertex j in the tour.
y_i	Binary, set to one if vertex i is selected.
$\boldsymbol{y}_r$	Vector defined as $\boldsymbol{y}_r = (y_i, i \in V_r)$, and we define $\boldsymbol{y} = (y_i, i \in V)$.
$Area(\boldsymbol{y}_r)$	Coverage area size determined by $\boldsymbol{y}_r$ of region $r \in R$, and we define $Area(\boldsymbol{y})$ for the covered area size of the reservoir including all regions.
u_i	Position of vertex i on the tour (the depot s is the first vertex).

2.3. Mathematical model

$$[M1] \quad \text{maximize} \sum_{r \in R} Area\left(\boldsymbol{y}_r\right) \tag{1}$$

subject to:

$$\sum_{j \in V \cup \{t\}} x_{sj} = \sum_{i \in V \cup \{s\}} x_{it} = y_s = y_t = 1 \tag{2}$$

$$\sum_{i \in V \cup \{s\}} x_{ik} = \sum_{j \in V \cup \{t\}} x_{kj} = y_k \quad k \in V \tag{3}$$

$$\sum_{i \in V \cup \{s\}} \sum_{j \in V \cup \{t\}} t_{ij} x_{ij} \leq T_{\max} \tag{4}$$

$$\sum_{i \in V} y_i \leq C_{\max} \tag{5}$$

$$\sum_{i \in V_r} y_i \geq S_r^{\min} \quad r \in R \tag{6}$$

$$2 \leq \ u_i \leq |V| + 2 \quad i \in V \cup \{t\} \tag{7}$$

$$u_i - u_j + 1 \leq (|V| + 1)(1 - x_{ij}) \quad i,\ j \in V \cup \{t\},\ i \neq j \tag{8}$$

$$x_{ij} \in \{0, 1\} \quad i \in V \cup \{s\},\ j \in V \cup \{t\},\ i \neq j \tag{9}$$

$$y_i \in \{0, 1\} \quad i \in V. \tag{10}$$

Objective (1) aims to maximize the total coverage area size of the selected sampling sites of all the clusters. Constraints (2) impose the tour to start at the depot and to return to the depot after finishing sampling. Constraints (3) ensure the connectivity of the path, and each sampling location can only be taken once. Constraints (4) and (5) concern the time and capacity budgets, respectively. Constraints (6) ensure that the minimum number of selected sampling sites are collected from one region. Constraints (7) and (8) prevent the subtours (Miller et al. 1960). Constraints (9) and (10) define the domains of the decision variables.

2.4. Model linearization

The overlapping of coverage areas induces complexity in modelling the CCOP, because the objective cannot be easily represented by a linear or a non-linear function. We provide here a linearization procedure toward an integer programming model.

The geographical area of each region can be discretized by using polygons of a given shape. The smaller the polygons, the better the approximation of the area covered by the sampling sites. However, smaller polygons (associated with the granularity of the discretization) do not increase the numbers of variables and constraints of the model if we package the polygons: As indicated in Figure 2, disjoint subareas q including polygons that are covered by the same subset of sampling sites can be considered independently. Let Q_r be the set including all the subareas in region $r \in R$.

Define γ_q as the size of subarea $q \in Q_r$ and let δ_{iq} be a binary coefficient that equals one if subarea $q \in Q_r$ is covered by sampling site $i \in V_r$. We introduce the binary decision variable z_q, which is set to one if subarea $q \in Q_r$, $r \in R$ is covered by one or more than one sampling site that is visited, then we can formulate the model:

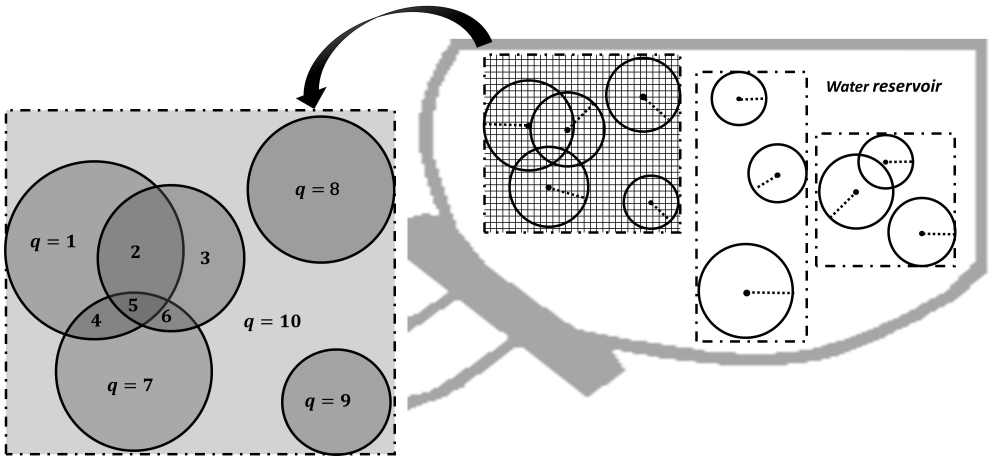

Figure 2 **To package polygons into disjoint subareas $q \in Q_r$ of region r.**

$$[M2] \quad \text{maximize} \sum_{r \in R} \sum_{q \in Q_r} z_q \gamma_q \tag{11}$$

subject to (2)–(10), and

$$z_q \leq \sum_{i \in V_r} y_i \delta_{iq} \quad q \in Q_r,\ r \in R \tag{12}$$

$$z_q \in \{0,1\} \quad q \in Q_r,\ r \in R. \tag{13}$$

Although model $[M2]$ is a linearized model with a reasonable number of decision variables, it can only solve small-scale instances by using off-the-shelf solvers directly, such as CPLEX. However, for large-scale instances, we need a more effective exact solution approach.

2.5. Model Implementations

We consider two additional options as benchmarks in addition to the direct implementation of model $[M2]$. Let continuous variables w_{ij} represent the arrival time at vertex j of the USV when coming from vertex i, and we introduce the following constraints:

$$w_{sj} = t_{sj} x_{sj} \quad j \in V, \tag{14}$$

$$\sum_{j \in V \cup \{t\}} w_{ij} - \sum_{j \in V \cup \{s\}} w_{ji} = \sum_{j \in V \cup \{t\}} t_{ij} x_{ij} \quad i \in V, \tag{15}$$

$$w_{ij} \le (T_{\max} - t_{jt})x_{ij} \quad i \in V \cup \{s\},\ j \in V \cup \{t\},\ i \ne j, \tag{16}$$

$$w_{ij} \ge (t_{si} + t_{ij})x_{ij} \quad i \in V \cup \{s\},\ j \in V \cup \{t\},\ i \ne j, \tag{17}$$

where by convention $t_{ss} = 0$, $t_{tt} = 0$. Constraints (14)–(17) are alternatives to constraints (7)–(8) in model $[M2]$ to prevent subtours (Maffioli and Sciomachen 1997). We implement model $[M2]$, model $[M2']$ obtained from model $[M2]$ by replacing constraints (7)–(8) with (14)–(17), and model $[M2'']$ obtained from model $[M2]$ adding with constraints (14)–(17) as three benchmarks.

3. Framework of a two-stage exact algorithm for the CCOP

We have designed a two-stage exact algorithm to obtain an optimal CCOP solution. The framework of the two-stage exact algorithm consists of procedures **(a)**–**(h)**, which are illustrated in Figure 3 and explained in the following. Note that we aim to provide a general framework to solve the CCOP or problems with a similar structure, so the described procedures can be substituted by other efficient ones with the same functions, e.g., other lower or upper bounding methods. Recall that three resource limits are present in the CCOP, which are (i) the capacity budget $C_{\max}$, (ii) the requirement that at least $S_r^{\min}$ sample sites must be selected in region $r \in R$, and (iii) the time budget $T_{\max}$. In the first stage of this algorithm, we generate a sampling task set, in which all tasks satisfy limits (i) and (ii).

Let sampling task $\boldsymbol{y} = (y_i, i \in V)$ be a selection of sampling sites, which embeds the information as to whether a candidate sampling site $i \in V$ is selected or not. The sampling task generator, i.e., procedure **(a)**, will generate a set Y, in which any $\boldsymbol{y} \in Y$ satisfies two restrictions:

$$\sum_{i \in V} y_i \le C_{\max} \wedge \sum_{i \in V_r} y_i \ge S_r^{\min}, r \in R. \tag{18}$$

Since the visiting sequence of the selected sites is not embedded, a task $\boldsymbol{y} \in Y$ may not be necessarily feasible concerning travel time limit, but all the feasible tasks are covered in Y. Naturally, each sampling task $\boldsymbol{y} \in Y$ yields a profit $Area(\boldsymbol{y})$ given the fixed selected sampling sites. We can search for the optimal ones in the total of $|Y|$ tasks, i.e., by comparing $Area(\boldsymbol{y})$ of those feasible tasks in Y. Note that the travel time feasibility check will be performed in the second stage.

To reduce the number of feasibility checks, especially when $|Y|$ is large, we execute two CCOP bounding procedures in the first stage. The procedure **(b)**, i.e., a CCOP heuristic, is applied to obtain the initial lower bound (incumbent best solution for the CCOP), denoted as $Area^{(L)}$. Meanwhile, a cut generation procedure **(c)** is applied to solve the CCOP relaxation problem to obtain an upper bound, denoted as $Area^{(U)}$. Let $\boldsymbol{y} \in Y'$ be the tasks bounded by $Area^{(L)} \le Area(\boldsymbol{y}) \le Area^{(U)}$, and thus set Y is reduced to set $Y'(Y' \subset Y)$.

In the second stage, we validate the travel time limit for tasks $\boldsymbol{y} \in Y'$. Once the feasibility of one task is validated, the corresponding sampling task deleter, i.e., procedure **(g)** or **(h)**, can be

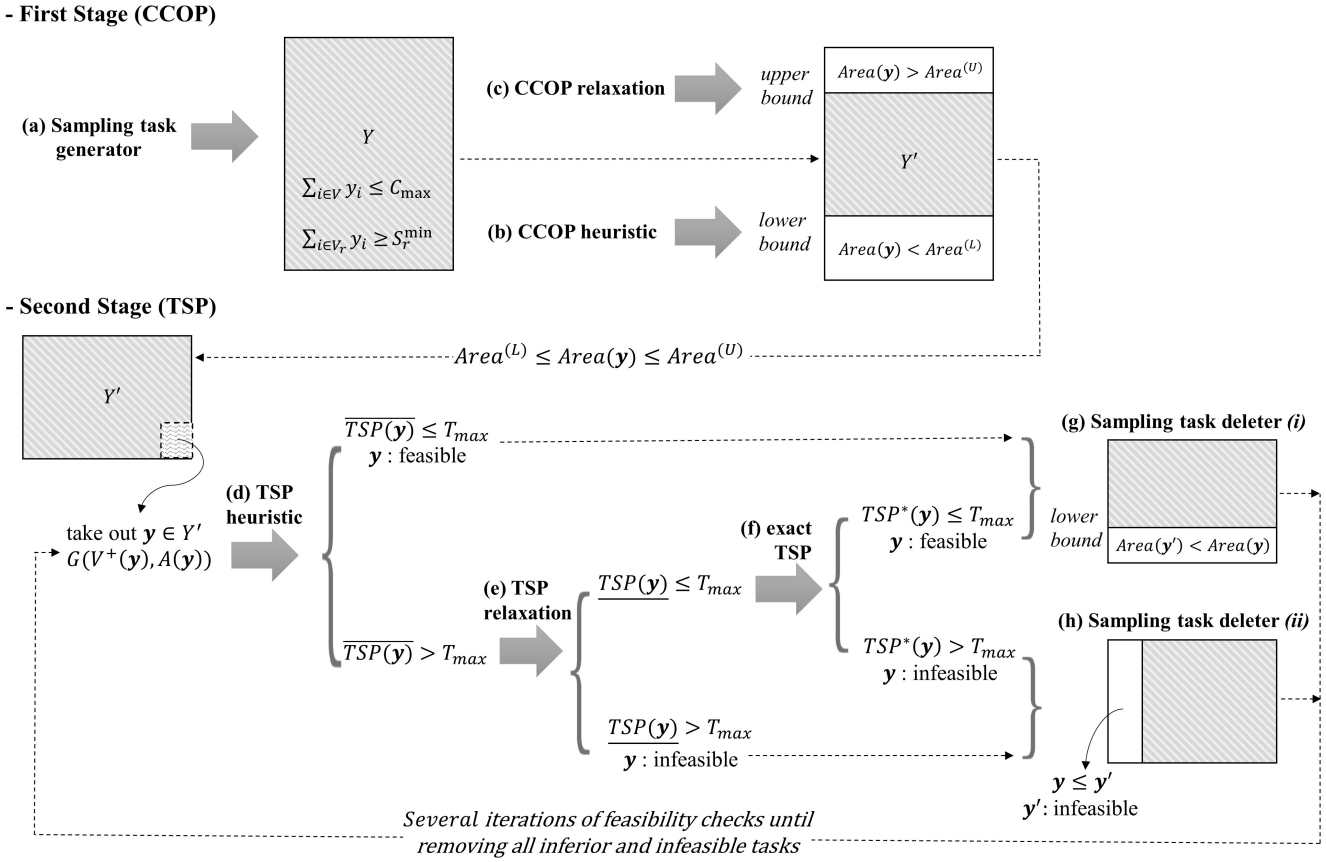

Figure 3 **Framework of the two-stage exact algorithm for the CCOP**[a].

[a] The shaded area in set Y or Y' represents the tasks waiting for feasibility checks, the unshaded area represents the inferior or infeasible tasks which have been removed.

executed to directly remove some other tasks. We introduce the following proposition, in which statements (i) and (ii) correspond to sampling task deleters (i) and (ii), respectively.

PROPOSITION 1. *(i) If a sampling task $\boldsymbol{y}$ is validated to be feasible, inferior tasks $\boldsymbol{y}' \in Y'$ satisfying $Area(\boldsymbol{y}') < Area(\boldsymbol{y})$ can be removed from Y'. (ii) If a sampling task $\boldsymbol{y}$ is validated to be infeasible, all infeasible tasks $\boldsymbol{y}' \in Y'$ satisfying $\boldsymbol{y}' > \boldsymbol{y}$ can be removed from Y'.*

Here, statement (i) implies that if a task $\boldsymbol{y}$ is feasible, its profit $Area(\boldsymbol{y})$ can be updated as the best-known CCOP lower bound (solution), and procedure **(g)** is triggered to remove all tasks $\boldsymbol{y}'$ whose coverage area is less than $Area(\boldsymbol{y})$. Meanwhile, as indicated by statement (ii), when a task $\boldsymbol{y}$ is infeasible, procedure **(h)** is triggered to remove all tasks $\boldsymbol{y}'$ where the selected sites of $\boldsymbol{y}$ is a subset of the selected sites of $\boldsymbol{y}'$. Statement (ii) is valid since an infeasible task $\boldsymbol{y}$ indicates that the travel time limit associated with the corresponding shortest elementary circuit is violated. Given

that the triangular inequality holds, we can conclude that the shortest elementary circuit for task $\boldsymbol{y}' > \boldsymbol{y}$ also violates the travel time limit.

The shortest elementary circuit for task $\boldsymbol{y}$ is checked by applying TSP procedures **(d)**, **(e)** and **(f)**, in which the first two are bounding procedures, and the last one solves the TSP to optimality. Note that the sampling task deleter is triggered once the feasibility or infeasibility of a task is validated by any one(s) of the three TSP procedures.

Procedures **(d)** and **(e)** aim at accelerating the algorithm by skipping the more time-consuming exact procedure **(f)**. We show in Figure 3 that procedure **(d)** is executed before **(e)**, while this precedence order can be reversed. We apply procedure **(d)**, the TSP heuristic, to obtain the TSP upper bound $\overline{TSP(\boldsymbol{y})}$; or procedure **(e)**, the TSP relaxation, for the TSP lower bound $\underline{TSP(\boldsymbol{y})}$, both of which run very fast. If $\overline{TSP(\boldsymbol{y})} \leq T_{\max}$ (i.e., task $\boldsymbol{y}$ is feasible) or $\underline{TSP(\boldsymbol{y})} > T_{\max}$ (i.e., task $\boldsymbol{y}$ is infeasible), we can skip procedure **(f)**. If necessary, we apply the exact TSP procedure **(f)** to obtain the minimum travel time for completing task $\boldsymbol{y}$, denoted as $TSP^*(\boldsymbol{y})$, and we validate the feasibility by checking whether $TSP^*(\boldsymbol{y}) \leq T_{\max}$ or $TSP^*(\boldsymbol{y}) > T_{\max}$.

Procedures from **(d)** to **(h)** of the second stage are iteratively executed, until completing the feasibility checks of all tasks $\boldsymbol{y} \in Y'$. At each iteration, we check the feasibility for one task, then directly determine the feasibility or infeasibility of some other tasks, and execute the sampling task deleter. The last remaining task(s) in Y' not deleted should be the optimal solution(s) $\boldsymbol{y}^*$ of the CCOP with profit $Area(\boldsymbol{y}^*)$.

In summary, our algorithm is built on the fact that given a task $\boldsymbol{y}$, its associated profit $Area(\boldsymbol{y})$ is fixed, and the travel time feasibility of each task $\boldsymbol{y}$ needs to be checked in order to find an optimal feasible one. The algorithmic steps of our exact method are presented in the Appendix. We describe the detailed procedures in the next sections.

4. First stage of the two-stage exact algorithm for the CCOP

The first stage of the two-stage exact algorithm focuses on the original CCOP. We will introduce the sampling task generator **(a)**, the CCOP upper bounding procedure **(b)**, and the CCOP lower bounding procedure **(c)** in Sections 4.1, 4.2 and 4.3, respectively. We will generate a non-necessary feasible task set Y by **(a)**, and reduce Y to Y' by both **(b)** and **(c)**, where the tasks $\boldsymbol{y} \in Y'$ may be infeasible with respect to the travel time limit.

4.1. Sampling task generator

We generate a set of sampling tasks $Y = \left\{\boldsymbol{y} \in \{0,1\}^{|V|} \,|\, \sum_{i \in V} y_i \leq C_{\max},\ \sum_{i \in V_r} y_i \geq S_r^{\min}, r \in R\right\}$, where each task $\boldsymbol{y}$, denoted by a vector $(y_i, i \in V)$, specifies the selected sampling sites to visit, i.e., those $i \in V$ with $y_i = 1$. Since that travel time limit is not guaranteed, $\boldsymbol{y} \in Y$ may be infeasible, while the optimal feasible solution $\boldsymbol{y}^*$ of the CCOP must be included in Y.

The generator includes two steps, where (19) and (20) are ensured respectively in each step, together equivalent to (18). First, we generate the set Λ in which $\boldsymbol{\varepsilon} = (s_r, r \in R)$, $\boldsymbol{\varepsilon} \in \Lambda$ is a feasible combination of different numbers of selected sites of each region $r \in R$, by ensuring that

$$\sum_{r \in R} s_r \leq C_{\max} \wedge s_r \geq S_r^{\min}, r \in R. \tag{19}$$

Then, for each $\boldsymbol{\varepsilon}$, we generate the set $Y_{\boldsymbol{\varepsilon}}$, in which $\boldsymbol{y} = (y_i, i \in V)$, $\boldsymbol{y} \in Y_{\boldsymbol{\varepsilon}}$ is a feasible selection of sampling sites satisfying

$$\sum_{i \in V_r} y_i = s_r, r \in R. \tag{20}$$

Finally, we can obtain $Y = \cup_{\boldsymbol{\varepsilon} \in \Lambda} Y_{\boldsymbol{\varepsilon}}$ as all possibilities of sampling tasks.

We put aside the task set Y, until we complete the two bounding procedures of the CCOP (described in Sections 4.2 and 4.3). After having obtained $Area^{(L)}$ and $Area^{(U)}$, we go back to deal with task set Y, i.e., reduce Y to Y', in which all $\boldsymbol{y} \in Y'$ are bounded by $Area^{(L)} \leq Area(\boldsymbol{y}) \leq Area^{(U)}$.

4.2. CCOP heuristic to obtain the initial lower bound

The CCOP dynamic programming-based heuristic (CCOP heuristic) is designed to quickly produce an initial lower bound $Area^{(L)}$ for the CCOP. This lower bounding procedure is fast as we apply two heuristic components in the dynamic programming (DP) algorithm: (i) As will be described in **State extension on the reduced graph**, this DP will be executed based on a reduced graph instead of the complete graph $G(V^+, A)$ of the CCOP. The reduced graph is defined by discarding some arcs $(i, j) \in A$. (ii) As will be described in **Heuristic dominance rule II**, some states are shortsightedly deleted even though they are not necessarily dominated by others and not exactly fathomed as the non-pareto optimal ones.

The CCOP heuristic aims to find an elementary path associated with two components, i.e., the sampling sites and the travel time (battery usage) of the USV. Let a state $(\boldsymbol{y}, T, i)$ represent a path from the depot s to a vertex $i \in V \cup \{t\}$ (a sampling site or the depot), where T is the time at which the USV visits vertex i; and $\boldsymbol{y} \in \{0,1\}^{|V|}$ is the $|V|$-dimensional binary vector, in which the k^{th} element y_k represents whether a candidate sampling site $k \in V$ is already visited ($y_k = 1$) or not ($y_k = 0$).[1] Each state $(\boldsymbol{y}, T, i)$ corresponds to a profit, a coverage area size $Area(\boldsymbol{y})$, that only depends on the already visited sampling sites. The initial state is defined as $(0^{|V|}, 0, s)$, where the USV is positioned at the depot s at time 0, has yet to visit any candidate site. A state $(\boldsymbol{y}, T, i)$ is feasible only if

$$\sum_{k \in V} y_k \leq C_{\max} \tag{21}$$

[1] The depot s and its replica depot t are fixed as the start and end vertices.

and

$$T \leq T_{\max}. \tag{22}$$

From (21) and the requirement $\sum_{k\in V_r} y_k \geq S_r^{\min}$ for $r \in R$,

$$\sum_{k\in V_r} y_k \leq C_{\max} - \sum_{r'\in R\setminus\{r\}} S_{r'}^{\min} \quad r \in R, \tag{23}$$

should also be satisfied. The feasibility checks of (21), (22) and (23) are conducted once a new state is explored, whereas the feasibility check of (24) described below can only be performed when the state extends to the last vertex, i.e., the depot t. Formally, a state $(\boldsymbol{y}, T, t)$ associated with vertex t is feasible only if

$$\sum_{k\in V_r} y_k \geq S_r^{\min} \quad r \in R. \tag{24}$$

This ensures that the number of selected sampling sites in each region is no less than $S_r^{\min}$.

For a current feasible state $(\boldsymbol{y}, T, i)$, only its feasible successors $(\boldsymbol{y}', T', j)$ can be generated. The state extension of the CCOP heuristic is based on the reduced graph described as follows.

State extension on the reduced graph. A label $(\boldsymbol{y}, T, i)$ of vertex i only extends to its feasible successors $(\boldsymbol{y}', T', j)$ if arc $(i, j) \in A$ satisfies

$$t_{ij} \leq \sigma_1 \bar{t}, \tag{25}$$

where $\bar{t} = \frac{\sum_{i\in V\cup\{s\}} \sum_{j\in V\cup\{t\}} t_{ij}}{|V|^2+|V|}$ as the average travelling time between two vertices, and parameter σ_1 affects the size of the reduced graph. Since the extension discards all arcs $(i, j) \in A$ with $t_{ij} > \sigma_1 \bar{t}$, a smaller value of σ_1 can speed up the algorithm but may yield an inferior solution, and vice versa. To ensure that at least one feasible solution is generated, a very small value of σ_1 should be avoided. Label $(\boldsymbol{y}', T', j)$ is updated from $(\boldsymbol{y}, T, i)$ by

$$\boldsymbol{y}' = \boldsymbol{y} + \boldsymbol{e}^j \text{ if } j \in V, \quad \boldsymbol{y}' = \boldsymbol{y} \text{ if } j \text{ is the depot } t, \tag{26}$$

to forbid the repeated selection of a sampling site, where $\boldsymbol{e}^j$ is a binary vector with $|V|$ elements whose j^{th} component equals one and the others equal zero. Meanwhile, the travel time resource is updated as

$$T' = T + t_{ij}. \tag{27}$$

Once a state is feasibly extended, dominance rules are performed as follows.

Exact dominance rule I. The exact dominance rule exactly fathoms some states that cannot lead to the optimum. If a state is necessarily dominated by another state, then it will be discarded. State $(\boldsymbol{y}^2, T^2, j)$ is dominated by state $(\boldsymbol{y}^1, T^1, j)$ if

(a)$\boldsymbol{y}^1 = \boldsymbol{y}^2$ and

(b) $T^1 \leq T^2$, and at least one of the inequalities is strict.

Note that $\boldsymbol{y}^1 = \boldsymbol{y}^2$ implies $Area(\boldsymbol{y}^1) = Area(\boldsymbol{y}^2)$ because of the natural property of the CCOP, i.e., the corresponding coverage area size of visited sampling sites are equal.

To quickly produce feasible solutions, we apply **Heuristic dominance rule II** before the **Exact dominance rule I** in the CCOP heuristic procedure as follows.

Heuristic dominance rule II. Each state $(\boldsymbol{y}, T, i)$ is mapped onto a new state (ϕ, T, i), where $\phi = \sum_{k \in V} y_k$. Although (ϕ^2, T^2, j) is not necessarily dominated by (ϕ^1, T^1, j), we still delete state (ϕ^2, T^2, j) if

(c) $\phi^1 = \phi^2$,

(d) $T^2 \geq \sigma_2 T^1$, where $\sigma_2 \geq 1$ is a control parameter.

A smaller value of σ_2 speeds up the procedure but may affect the quality of the initial lower bound $Area^{(L)}$ of the CCOP. The sequence of implementing the dominance rules will not affect the quality of $Area^{(L)}$, while applying rule II before rule I can accelerate the procedure.

To the end, we derive the initial lower bound $Area^{(L)}$ of the CCOP, that is the maximum coverage area size of a feasible state $(\boldsymbol{y}, T, t)$ extended to the depot t. The two control parameters σ_1 and σ_2 aim at reducing the number of the states to accelerate this CCOP heuristic procedure. However, excessively small values of σ_1 and σ_2 should be avoided, because they may prevent finding a feasible solution satisfying the requirement that at least $S_r^{\min}$ sample sites must be visited in a region $r \in R$. The control parameters can be adjusted on a case by case basis to balance the computation time and the quality of the initial lower bound $Area^{(L)}$. We could conceivably run an exact implementation of this DP in order to derive an optimal CCOP solution; however, the exact DP implementation is very time-consuming.

4.3. CCOP cut generation for upper bounding

The cut generation procedure is applied to obtain a CCOP upper bound $Area^{(U)}$ by solving its relaxation problem with two classes of additional valid inequalities.

The relaxation problem of the CCOP without additional inequalities is described as model $[M3]$ (i.e., a relaxation of model $[M2]$), in which the integrality restriction of y_i is always retained while only x_{ij} is set to be fractional.

$$[M3] \quad \text{maximize} \sum_{r \in R} \sum_{q \in Q_r} z_q \gamma_q \tag{28}$$

subject to (2)–(6), (10), (12)–(13), (14)–(17), and

$$0 \leq x_{ij} \leq 1 \quad i \in V \cup \{s\},\ j \in V \cup \{t\},\ i \neq j. \tag{29}$$

Constraints (14)–(17) are applied in model $[M3]$ as more efficient alternatives to Constraints (7)–(8), which will be connotated by our experiments. This model can be solved by off-the-shelf solvers (e.g., CPLEX) very fast to obtain an upper bound of the CCOP, but its quality is proven to be unstable in our experiments. Hence, a cut generation procedure is applied to dynamically generate additional valid inequalities that are added back to model $[M3]$ for tightening its formulation.

Two classes of valid inequalities (30) and (31) are used. Inequalities (30) are derived from the cycle relaxation, considering that subtours are not necessarily prevented when x_{ij} is fractional. Thus, inequalities (30) are formulated as

$$\sum_{i \in \Theta} \sum_{j \in \Theta} x_{ij} \leq |\Theta| - 1 \quad \Theta \subset V,\ 2 \leq |\Theta| \leq \sigma_3, \tag{30}$$

where σ_3 is a control parameter, and $\sigma_3 \leq |V| - 1$. A smaller value of σ_3 may affect the quality of the upper bound of the CCOP but let the procedure run faster. Inequalities (31) have been proposed in Fischetti et al. (1998) for the OP and are formulated as

$$\sum_{(i,j) \in \tau} x_{ij} \leq |\tau| - 1 \quad \tau \subseteq A', \tag{31}$$

where $A' = \{\tau' \subseteq A | \sum_{(i,j) \in \tau'} t_{ij} > T_{\max}\}$. These inequalities stipulate that not all the edges of τ can be selected in a feasible path of the CCOP.

We refer to model $[M3]$ augmented by valid inequalities associated with (30) and (31) as to augmented relaxation model. The total number of inequalities increases with σ_3, and the augmented relaxation model is correspondingly harder to solve with more valid inequalities. A cut generation procedure is executed to add the valid inequalities iteratively, which can speed up the procedure compared with adding all the inequalities to model $[M3]$ in one batch: First, we solve model $[M3]$ and draw an initial solution. As none of the valid inequalities is imposed so far, the initial solution may violate some of the inequalities (30) and (31). The violated inequalities will be added back to model $[M3]$ defining an augmented relaxation model. This model is then solved to yield a new solution (a tighter upper bound), then other violated inequalities may be found and added back to the previous augmented relaxation model. With iteratively added inequalities, the corresponding augmented relaxation model is resolved at each iteration. In the end, we can obtain the upper bound $Area^{(U)}$ for the CCOP.

5. Second stage of the two-stage exact algorithm

The second stage of the two-stage exact algorithm iteratively considers TSPs, in order to check the travel time feasibility condition for each task $\boldsymbol{y} \in Y'$. The TSP is defined over graph $G(V^+(\boldsymbol{y}), A(\boldsymbol{y}))$, where the vertex set $V^+(\boldsymbol{y})$ includes vertices of sampling sites of task $\boldsymbol{y}$ and a vertex of depot. (In the TSP, the depot is represented by one vertex rather than being split into two vertices with the same location as we did in the CCOP.) An arc $(i,j) \in A(\boldsymbol{y})$ is between any two vertices for $\forall i,\ j \in V^+(\boldsymbol{y})$, $i \neq j$, with the travelling time t_{ij} satisfying the triangle inequality. Here we give the formal formulation for the TSP of task $\boldsymbol{y} \in Y'$ as model $[M4]$:

$$[M4] \quad \text{minimize} \sum_{i \in V^+(\boldsymbol{y})} \sum_{j \in V^+(\boldsymbol{y})} t_{ij} x_{ij} \tag{32}$$

subject to

$$\sum_{i \in V^+(\boldsymbol{y})} x_{ik} + \sum_{j \in V^+(\boldsymbol{y})} x_{kj} = 2 \quad k \in V^+(\boldsymbol{y}) \tag{33}$$

$$\sum_{i \in \Theta} \sum_{j \in \Theta} x_{ij} \leq |\Theta| - 1 \quad \Theta \subset V^+(\boldsymbol{y}),\ |\Theta| \geq 2 \tag{34}$$

$$x_{ij} \in \{0,1\} \quad i,\ j \in V^+(\boldsymbol{y}),\ i \neq j. \tag{35}$$

Equation (33) defines the degree of each vertex, constraints (34) eliminate the subtours, and constraints (35) are the integrality constraints.

In Section 5.1, 5.2 and 5.3, we will introduce the TSP heuristic procedure **(d)**, the TSP relaxation procedure **(e)**, and procedure **(f)** to solve the TSP to optimality, respectively. Once the feasibility or infeasibily is validated by anyone(s) of these three procedures, the others can be skipped, then the sampling task deleter **(g)** or **(h)** is triggered to remove inferior or infeasible tasks (described in Section 5.4). We iteratively execute procedures **(d)**–**(h)** until completing all the feasibility checks and removing all the inferior and infeasible tasks, then the task(s) left in Y' should be the optimal feasible task(s) $\boldsymbol{y}^*$ of the CCOP.

5.1. TSP heuristics to accelerate the feasibility check

The heuristics introduced in this section aim at fast producing an upper bound $\overline{TSP(\boldsymbol{y})}$ for the TSP of task $\boldsymbol{y}$ for its feasibility check. An initial TSP tour κ_0 is first constructed by a greedy construction heuristic, i.e., the nearest neighborhood heuristic. It starts from the depot, builds a tour of task $\boldsymbol{y}$ by adding the closest vertex to the previous vertex added to the tour, and returns to the depot.

The 2-opt heuristic is then executed to reorder the initial tour κ_0 so as to reduce its total travel time (tighten the TSP upper bound). Two non-adjacent arcs are randomly selected and removed in iteration ω, such that the cycle tour κ_ω is broken into two subpaths. The two subpaths are then

reconnected to create a new cycle κ' different from the previous one. If the total travel time of the TSP is reduced, set $\kappa_{\omega+1} = \kappa'$, otherwise, set $\kappa_{\omega+1} = \kappa_\omega$. We run several iterations in this fashion until it encounters a local minimum.

Each time a local minimum is encountered, we rerun the 2-opt heuristic described above, attempting to reach other local minima. Although the same initial tour κ_0 is employed, it may still jump to another local minimum as long as the two arcs are randomly selected and broken. The upper bound is updated by comparing the newly generated local minimum with the previous ones. These local minima are compared to obtain the best-known TSP upper bound $\overline{TSP(\boldsymbol{y})}$.

In our experiments, the best-known upper bound has a high probability of reaching the TSP optimality. Nevertheless, the heuristics can stop whenever the updated upper bound for the TSP is lower than the travel time limit $T_{\max}$, where task $\boldsymbol{y}$ can be confirmed to be feasible.

If $\overline{TSP(\boldsymbol{y})} \leq T_{\max}$, task $\boldsymbol{y}$ is feasible, and the TSP relaxation (Section 5.2) and exact TSP solving procedures (Section 5.3) can be skipped. Otherwise, the algorithm proceeds to the other bounding procedure described in Section 5.2.

5.2. TSP relaxation to accelerate the feasibility check

This procedure solves the TSP relaxation problem to obtain the TSP lower bound $\underline{TSP(\boldsymbol{y})}$ for the feasibility check. Constraints (34) and (35) in model $[M4]$ are relaxed to constraints (37) and (38) in the TSP relaxation model $[M5]$:

$$[M5] \quad \text{minimize} \sum_{i \in V^+(\boldsymbol{y})} \sum_{j \in V^+(\boldsymbol{y})} t_{ij} x_{ij} \tag{36}$$

subject to (31), and

$$\sum_{i \in \Theta} \sum_{j \in \Theta} x_{ij} \leq |\Theta| - 1 \quad \Theta \subset V^+(\boldsymbol{y}),\ 2 \leq |\Theta| \leq \sigma_4 \tag{37}$$

$$0 \leq x_{ij} \leq 1 \quad i,\ j \in V^+(\boldsymbol{y}),\ i \neq j, \tag{38}$$

where σ_4 is a control parameter, and $\sigma_4 \leq |V^+(\boldsymbol{y})| - 1$. A smaller value of σ_4 speeds up the algorithm but may affect the quality of the TSP lower bound. Model $[M5]$ can be solved efficiently to obtain the TSP lower bound because the number of vertices of $\boldsymbol{y}$ is limited.

If $\underline{TSP(\boldsymbol{y})} > T_{\max}$, task $\boldsymbol{y}$ is confirmed to be infeasible. If the feasibility or infeasibility is still undetermined after applying the two bounding procedures described in Sections 5.1 and 5.2, we execute the exact TSP solving procedure in Section 5.3.

5.3. Exact TSP solving procedure

Here we describe an exact DP procedure for the TSP in order to finally complete the feasibility check for task $\boldsymbol{y}$. Recall that the TSP is based on the complete graph of $G(V^+(\boldsymbol{y}), A(\boldsymbol{y}))$. Each

state $(\boldsymbol{\eta}, T, i)$ of this DP represents a path from the depot to a vertex i; the vector $\boldsymbol{\eta}$ has $|V(\boldsymbol{y})|$ elements corresponding to the $|V(\boldsymbol{y})|$ selected sampling sites for task $\boldsymbol{y}$; T is the time at which the USV reaches vertex i. The initial state is defined as $(0^{|V(\boldsymbol{y})|}, 0, \text{depot})$ when the USV departs from the depot at time 0 and has yet to visit any sampling site. Once a site is visited, the corresponding element in $\boldsymbol{\eta}$ is updated from 0 to 1, and the travel time T is updated accordingly. A state $(\boldsymbol{\eta}^1, T^1, i)$ dominates a state $(\boldsymbol{\eta}^2, T^2, i)$, if (a) $\boldsymbol{\eta}^1 \geq \boldsymbol{\eta}^2$, (b) $T^1 \leq T^2$, and at least one of the inequalities is strict.

The optimal solution value $TSP^*(\boldsymbol{y})$ is the minimum travel time of a state $(1^{|V(\boldsymbol{y})|}, T, \text{depot})$ associated with the USV located at the depot after having visited all the sampling sites of $V(\boldsymbol{y})$. If $TSP^*(\boldsymbol{y}) \leq T_{\max}$, then task $\boldsymbol{y}$ is feasible; if $TSP^*(\boldsymbol{y}) > T_{\max}$, then task $\boldsymbol{y}$ is infeasible. To avoid unnecessary label extensions, we can also impose the travel time limit by ensuring $T \in [0, T_{\max}]$ in this DP. If no feasible solution is obtained in the end, we can directly confirm that task $\boldsymbol{y}$ is infeasible; otherwise, task $\boldsymbol{y}$ is feasible.

5.4. Sampling task deleter

After validating the feasibility or infeasibility for a task $\boldsymbol{y}$, we can directly determine some other infeasible or inferior tasks and then remove them by invoking the corresponding sampling task deleter as procedure **(g)** or **(h)**.

If task $\boldsymbol{y}$ is proven to be feasible, its coverage area size $Area(\boldsymbol{y})$ can be updated as the best-known lower bound for the CCOP. Then, inferior tasks $\boldsymbol{y}' \in Y'$ with $Area(\boldsymbol{y}') < Area(\boldsymbol{y})$ can be removed from Y' by procedure **(g)**, whereas tasks $\boldsymbol{y}''$ with $Area(\boldsymbol{y}'') \geq Area(\boldsymbol{y})$ are waiting for the feasibility check. Thereafter, whenever we obtain a feasible task, i.e., still denoted as $\boldsymbol{y}''$, its coverage area size $Area(\boldsymbol{y}'')$ is directly updated as the best-known CCOP solution, considering that $Area(\boldsymbol{y}'')$ must be at least equal to the previously known CCOP lower bound. This sampling task deleter applies whenever a feasible task is found, not necessarily only after solving the exact TSP, but may also after using the TSP heuristics, as long as the feasibility is proven.[2]

If task $\boldsymbol{y}$ is proven to be infeasible, all tasks $\boldsymbol{y}' \in Y'$ that satisfy $\boldsymbol{y}' > \boldsymbol{y}$ are also infeasible and can be removed from Y' by procedure **(h)**, because of the triangular inequality and $V(\boldsymbol{y}) \subset V(\boldsymbol{y}')$. Following this principle, in the second stage of our algorithm, a task $\boldsymbol{y}$ with a smaller number of the selected sites $|V(\boldsymbol{y})|$ is checked first, so that more infeasible tasks can be directly removed without executing any TSP procedures if task $\boldsymbol{y}$ is infeasible, reducing the overall computation time. This sampling task deleter applies whenever an infeasible task is detected, and the exact TSP procedure is not required if the TSP relaxation has already proven the infeasibility.

[2] Procedure **(g)** can be replaced by a preliminary check at each iteration (for task $\boldsymbol{y}''$) of the second stage: as $\boldsymbol{y}$ is the best incumbent (feasible) sampling task found so far, if $Area(\boldsymbol{y}'') \geq Area(\boldsymbol{y})$, task $\boldsymbol{y}''$ is going to be evaluated; otherwise, task $\boldsymbol{y}''$ is removed. It can avoid double checks involved in procedure **(g)** and thus speed up the algorithm.

6. Computational experiments

The proposed two-stage exact algorithm for the CCOP is tested in this section. Performance tests under small-scale and large-scale instances are conducted, and the algorithm is compared with CPLEX by solving model $[M2]$, model $[M2']$, model $[M2'']$ directly. Then, sensitivity analyses of time and capacity budget of the CCOP are conducted, from which some insights can be drawn. All tests are performed using a PC with Intel® Core™ i7-3770 CPU (3.40 GHz) and 8 GB RAM. For small-scale instances, we solve all tests to optimality by all the three model benchmarks $[M2]$, $[M2']$, and $[M2'']$ and the two-stage exact algorithm. For large-scale instances, the two-stage exact algorithm and benchmarks are tested with a CPU time limit of two hours. In our experiments, to implement the CCOP heuristic procedure (Section 4.2), we set the speed control parameter $\sigma_1 = 0.8$ for small-scale instances, $\sigma_1 = 0.6$ for large-scale instances, and the relaxation control parameter $\sigma_2 = 1$ for all instances. For the CCOP cut generation procedure (Section 4.3), we set the subtour elimination parameter $\sigma_3 = 6$. For the TSP relaxation (Section 5.2), we also set the subtour elimination parameter $\sigma_4 = 6$ for all tests.

6.1. Computational performances on small-scale and large-scale instances

The two-stage exact algorithm is first executed on small-scale instances. Tests of instances with around 30 samplings sites are conducted, for which instance details are shown in Table 1 and computational results are reported in Table 2. For each instance (Inst.), Table 1 provides the total number of vertices, the number of sampling site clusters, the number of candidate sampling sites of each cluster and the minimum selected number of each cluster, e.g., $|V^+|$, $|R|$, and e.g., $|V_1|$ and $S_1^{\min}$ of cluster 1. In Table 2, three combinations of capacity budget ($C_{\max}$) and time budget ($T_{\max}$) are set for each instance in Table 1 to test the performances of all solution methods. We present here the time budget $T_{\max}$ as a distance because we assume a constant speed of the USV.

Table 1 Characteristics of small-scale instances.

Inst.	$\lvert V^+\rvert$	$\lvert R\rvert$	$\lvert V_1\rvert(S_1^{\min})$	$\lvert V_2\rvert(S_2^{\min})$	$\lvert V_3\rvert(S_3^{\min})$	$\lvert V_4\rvert(S_4^{\min})$	$\lvert V_5\rvert(S_5^{\min})$	$\lvert V_6\rvert(S_6^{\min})$
I–1	25	5	4(1)	7(4)	4(4)	4(1)	4(1)	–
I–2	27	5	4(2)	7(1)	3(2)	4(4)	7(1)	–
I–3	29	6	5(3)	6(3)	4(2)	4(1)	4(2)	4(1)
I–4	31	5	6(1)	7(5)	3(3)	5(1)	8(3)	–
I–5	33	6	4(2)	6(3)	5(3)	5(3)	6(2)	5(3)

For all instances, we consider that the number of regions and candidate sampling sites increases with the size of the reservoir under consideration. For instance I–1, the reservoir captured by the coordinate range of $[0, 1200] \times [0, 800]$ has five regions. The region sizes can be set differently, e.g.,

400×400 and 400×800, so that coordinate ranges of the five regions can be set as $[0, 400] \times [0, 400]$, $[0, 400] \times [400, 800]$, $[400, 800] \times [0, 400]$, $[400, 800] \times [400, 800]$, and $[800, 1200] \times [0, 800]$. In each region, candidate sampling site coordinates are generated randomly, whereas the diameter of the coverage area of each sampling site is generated within $[80, 120]$ by also considering that the coverage area cannot exceed its region boundary. To make the instances more general, the USV depot can be located anywhere in the reservoir. The distance t_{ij} is computed via the Euclidean norm after generating coordinates of candidate sampling sites and the USV depot.

Table 2 Computational results comparing CPLEX in model $[M2]$, model $[M2']$, model $[M2'']$ and the two-stage exact algorithm for the CCOP on small-scale instances.

			Model $[M2]$		Model $[M2']$		Model $[M2'']$		Two-Stage Exact Algorithm					
Inst.	$C_{\max}$	$T_{\max}$	Sol.	CPU	Sol.	CPU	Sol.	CPU	Sol.	CPU	$Area^{(L)}$	$Area^{(U)}$	$\|Y\|$	$\|Y'\|$
I–1	15	3000	107044	5416	107044	10	107044	10	107044	4	100540	107044	135760	35719
	15	3400	121876	4195	121876	5	121876	11	121876	2	117292	121876	135760	5726
	16	3200	123524	7238	123524	9	123524	10	123524	3	120168	123524	175612	10565
I–2	15	3000	130212	3466	130212	5	130212	7	130212	3	121092	130212	309680	61646
	17	3000	136344	1622	136344	9	136344	9	136344	6	124444	136344	565446	190881
	19	3200	156836	4926	156836	7	156836	11	156836	4	145068	156836	685538	54357
I–3	13	2500	103472	7428	103472	17	103472	85	103472	8	96712	104348	758400	271462
	14	2500	110776	6771	110776	27	110776	46	110776	13	104824	110776	2511680	569523
	14	2600	113408	5020	113408	40	113408	32	113408	12	104824	113816	2511680	626989
I–4	13	2700	103936	13090	103936	50	103936	63	103936	2	103936	103936	35280	0
	13	2800	109956	8336	109956	93	109956	182	109956	2	103936	109956	35280	9216
	14	3000	123696	2785	123696	47	123696	22	123696	3	115712	123696	249900	19628
I–5	16	2800	134740	2116	134740	47	134740	41	134740	20	122992	134740	1800000	734761
	17	2900	145600	4833	145600	34	145600	33	145600	22	135496	145600	9450000	1277054
	17	3000	148312	3189	148312	31	148312	34	148312	18	135496	148312	9450000	1311126
Avg.				**5362**		**29**		**40**		**8**				

Table 2 reports the solution (Sol.) and the computational time (CPU, in seconds) of all solution methods, in all instances. The first observation is that the direct implementation of model $[M2]$ is not efficient even for small-scale instances. In 10 of these 15 instances, model $[M2]$ takes more than one hour to derive the optimal solution. However, models $[M2']$ and $[M2'']$ can derive the optimal solution in less than four minutes in all these 15 instances, for which model $[M2]$ runtimes span 1622–13090 seconds (i.e., 27 minutes to 3.6 hours). As model $[M2']$ and model $[M2'']$ are much more efficient than model $[M2]$, it shows that constraints (14)–(17) are more effective than constraints (7)–(8). Constraints (14)–(17) are also applied in the CCOP bounding procedure in the two-stage exact algorithm. As indicated by the column $Area^{(U)}$ in Table 2, the upper bounds are very close to the corresponding optimal solutions to the problem, which speeds up the algorithm.

We now compare the outputs of the two-stage exact algorithm to the results of benchmarks: our algorithm dominates all the CPLEX-based implementations, returning the optimal solutions in shorter computational times in all tests; on average, it reduces by 72.4% the computational time of model $[M2']$ and reduces by 80% the computational time of model $[M2'']$. Besides, for each test of the two-stage exact algorithm, the lower and upper bounds for the CCOP are listed, i.e., $Area^{(L)}$ and $Area^{(U)}$, and the sizes of sampling task set and the remaining set after two bounding procedures, i.e., $|Y|$ and $|Y'|$, are also listed. The largely decreased size by comparing $|Y|$ and $|Y'|$ indicates that the bounding procedures can efficiently remove a large number of sampling tasks that cannot be optimal solutions.

Table 3 Characteristics of large-scale instances.

Inst.	$\|V^+\|$	$\|R\|$	$\|V_1\|(S_1^{\min})$	$\|V_2\|(S_2^{\min})$	$\|V_3\|(S_3^{\min})$	$\|V_4\|(S_4^{\min})$	$\|V_5\|(S_5^{\min})$	$\|V_6\|(S_6^{\min})$	$\|V_7\|(S_7^{\min})$
I–6	40	6	6(4)	6(4)	6(6)	6(2)	7(1)	7(1)	–
I–7	45	6	7(4)	7(4)	7(7)	8(2)	7(1)	7(1)	–
I–8	50	6	8(5)	8(5)	8(8)	9(2)	8(1)	7(1)	–
I–9	55	7	10(8)	6(1)	7(1)	7(6)	7(1)	6(1)	10(1)
I–10	60	7	7(5)	7(6)	10(1)	6(5)	11(1)	7(6)	10(1)

The two-stage exact algorithm is then tested on large-scale instances, including a larger number of candidate sampling sites and site clusters, as well as a large-capacity budget, i.e., $|V^+|$, $|R|$, and $C_{\max}$. The characteristics of large-scale instances are listed in Table 3, and computational results are reported in Table 4. Considering results for small-scale instances have already shown that model $[M2]$ is not efficient, Table 4 only reports the performance for CPLEX implementations of models $[M2']$ and $[M2'']$ as benchmarks.

We first compare the performance of models $[M2']$ and $[M2'']$: In three instances, both models cannot find a feasible solution (i.e., "Null"). For the eight instances in which both models find the same solution: model $[M2']$ is faster than model $[M2'']$ in six instances and slower than model $[M2'']$ in one instance; while in the other instance, both models $[M2']$ and $[M2'']$ actually reach the optimal solution but have not proven the optimality i.e., they do not terminate within 7200 seconds. Moreover, model $[M2']$ finds a better solution in three instances but a worse solution in one instance compared with model $[M2'']$.

We now analyze the performance of the two-stage exact algorithm: our algorithm provides Pareto improvements over all the CPLEX implementations in all tests, resulting in superior solutions and shorter computational times. In six of the 15 instances, our algorithm provides the best overall solution, for which the CPLEX implementation may not derive a feasible solution. Our algorithm

Table 4 Computational results comparing CPLEX in model $[M2']$, model $[M2'']$ and the two-stage exact algorithm for the CCOP on large-scale instances.

			Model $[M2']$		Model $[M2'']$		Two-Stage Exact Algorithm					
Inst.	$C_{\max}$	$T_{\max}$	Sol.	CPU	Sol.	CPU	Sol.	CPU	$Area^{(L)}$	$Area^{(U)}$	$\|Y\|$	$\|Y'\|$
I–6	19	3100	154530	>7200	152720	>7200	154832	219	154832	162448	3344250	1060404
	20	3200	165468	>7200	161872	>7200	165764	745	162568	174200	14987385	4825033
	20	3300	171100	765	171100	1351	171100	303	162568	176204	14987385	4894259
I–7	19	3300	170268	>7200	170268	>7200	170268	35	155420	170268	1680700	888359
	20	3300	Null	>7200	178132	>7200	178132	108	165088	181208	17143140	6449713
	21	3200	154532	>7200	Null	>7200	176792	3783	171700	183000	90858642	35993225
I–8	22	3200	Null	>7200	Null	>7200	178424	94	157304	182252	2709504	2552676
	23	3300	Null	>7200	Null	>7200	187420	897	167016	192236	30256128	25067395
	23	3400	191100	6353	191100	2139	191100	858	167016	193632	30256128	25079299
I–9	19	3700	144944	1093	144944	2757	144944	208	124084	145056	5556600	4723383
	20	3500	Null	>7200	Null	>7200	141528	6989	137916	155096	93712500	41392922
	20	3600	149628	4538	149628	>7200	149628	3328	140356	156048	93712500	26874519
I–10	25	4200	187080	2187	187080	5715	187080	1243	182636	194336	6791400	2516060
	25	4300	197012	756	197012	2416	197012	387	182636	198160	6791400	2611900
	26	4300	207764	3374	207764	>7200	207764	2611	191600	209608	107207100	35466246
Avg.				**5111**		**5759**		**1454**				

returns the optimal solutions in all of these 15 instances within two hours, and our algorithm terminates much faster than both models $[M2']$ and $[M2'']$. For instance, we derive the optimal solution in less than 600 seconds (i.e., ten minutes) in seven instances, for which the corresponding runtimes of the CPLEX implementation at least span 756–7200 seconds. In four other instances, we derive the optimal solution in less than 1500 seconds (i.e., 25 minutes), while the corresponding computational time of CPLEX implementation at least span 2139–7200 seconds (i.e., 35 minutes to over two hours). On average, for the large-scale instances, our algorithm reduces by 71.6% the computational time of model $[M2']$ and reduces by 74.8% the computational time of model $[M2'']$. In both small-scale and large-scale instances, the two-stage exact algorithm significantly outperforms the CPLEX implementations.

6.2. Sensitivity analyses for the CCOP

To test the effects of increasing battery endurance and capacity load of a USV, we conduct sensitivity analyses with regard to time and capacity budgets based on Instance I–4 of Table 1 (sensitivities of other instances are similar to Instance I–4). Optimal results are obtained for 66 combinations of different time and capacity budgets by conducting the proposed two-stage exact algorithm of the CCOP, as illustrated in Figures 4 and 5, respectively.

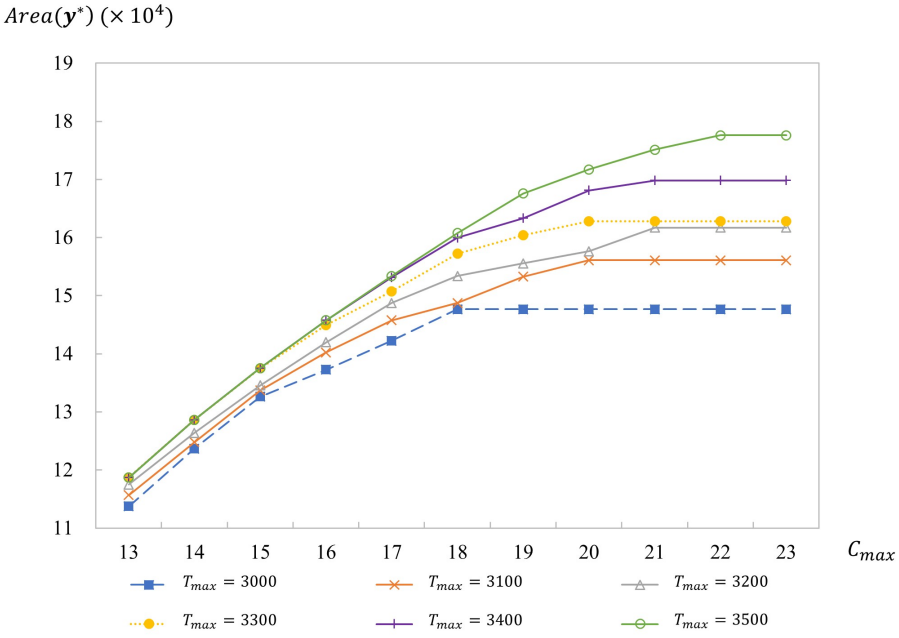

Figure 4 **Sensitivity analysis for the capacity budget of the CCOP.**

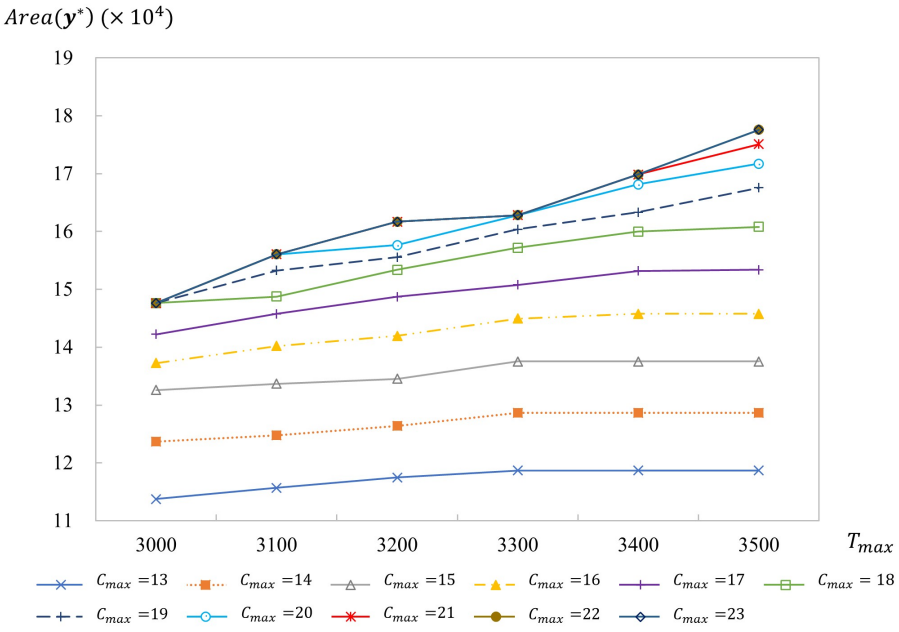

Figure 5 **Sensitivity analysis for the time budget of the CCOP.**

Figure 4 shows a series of polylines that are approximately concave, i.e., the additional benefit of increasing the capacity generally becomes marginal when $C_{\max}$ is large. With increasing $C_{\max}$, the optimal result $Area(\boldsymbol{y}^*)$ is further limited by the time budget $T_{\max}$. In real-life applications of water sampling by a USV, an operator can decide which of the budgets to increase. As shown in

Figure 4, if the cost of increasing 100 units of battery endurance is the same as that of increasing by one unit of USV load capacity, then increasing the USV load capacity is more effective when the current capacity is under 18. If the current load capacity is sufficient, e.g., more than 20, then the operator should invest in increasing USV battery endurance to increase benefits. The previous insight can also be explained by interpreting Figure 5. The six polylines with $C_{\max}$ under 18 of the lower part of Figure 5 remain distributed separately when $T_{\max}$ increases from 3000 to 3500 because USV load capacity is the dominant budget. The other five lines of the upper part of Figure 5 may intersect with each other with increasing $T_{\max}$, whereas the intersecting nodes indicate the USV battery endurance limits.

7. Conclusion

The clustered coverage orienteering problem (CCOP) is in a generalization of the orienteering problem (OP). It was described in the context of performing water sampling tasks by unmanned surface vehicles. Two special characteristics distinguish the CCOP from the traditional OP, which are "cluster" and "coverage" of the CCOP. In order to monitor the whole reservoir considering the special correlation and differentiation, candidate sampling sites are grouped into clusters. For each cluster, the minimum sampling sites should be selected to assess the water quality of different special areas. The water sample collected from each sampling site can represent a certain area of the water, called the "coverage area", whereas water samples collected from one cluster may share common information indicated by the overlapping coverage area of different sampling sites. The aim of the CCOP is to collect samples to maximize the coverage area of the selected sampling sites. The total coverage area is the area of the union of the coverage areas of the selected sites but not their sum. To deal with this non-linear objective function, we linearized the model so that it can be solved by off-the-shelf solvers for small-scale instances.

The two-stage exact algorithm for the CCOP was then proposed to deal with large instances that cannot be solved by the off-the-shelf solver. Our two-stage exact algorithm generally provides Pareto improvements, resulting in superior solutions and shorter computational times compared with CPLEX implementations: it solves instances with up to 60 vertices and reduces the computational time by over 70% on average. Hence, the two-stage exact algorithm has the advantages of being applicable in practice, especially for emergent water sampling tasks, e.g., when facing pollution of the reservoir. The efficiency and effectiveness of the two-stage exact algorithm can be attributed to the exploration of the specific problem structure of the CCOP. In the first stage of the two-stage exact algorithm, the capacity limit of the CCOP and the minimum number of selected sites of each cluster are both guaranteed. Meanwhile, the second stage only deals with the time limit, which is a TSP. We believe our work may inspire the construction of future algorithms for the OP, which is a special case of the CCOP.

Acknowledgments

This study is supported by the National Natural Science Foundation of China (Grant Nos. 71701178 and 71831008). Gilbert Laporte was supported by the Canadian Natural Sciences and Engineering Research Council [grant number 2015-06189]. The authors would like to thank the Editor, the Associate Editor and the anonymous referees for their constructive comments and suggestions.

References

Alamdari S, Fata E, Smith SL (2014) Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. *The International Journal of Robotics Research* 33(1):138–154.

Angelelli E, Archetti C, Vindigni M (2014) The clustered orienteering problem. *European Journal of Operational Research* 238(2):404–414.

Choset H (2000) Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots* 9(3):247–253.

Choset H (2001) Coverage for robotics—A survey of recent results. *Annals of Mathematics and Artificial Intelligence* 31(1-4):113–126.

Dunbabin M, Grinham A, Udy J (2009) An autonomous surface vehicle for water quality monitoring. *Australasian Conference on Robotics and Automation*, 2–4.

Faigl J (2010) Approximate solution of the multiple watchman routes problem with restricted visibility range. *IEEE Transactions on Neural Networks* 21(10):1668–1679.

Fazli P, Davoodi A, Mackworth AK (2013) Multi-robot repeated area coverage. *Autonomous Robots* 34(4):251–276.

Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. *Transportation Science* 39(2):188–205.

Fischetti M, Salazar González JJ, Toth P (1998) Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing* 10(2):133–148.

Franco C, Stipanović DM, López-Nicolás G, Sagüés C, Llorente S (2015) Persistent coverage control for a team of agents with collision avoidance. *European Journal of Control* 22:30–45.

Gendreau M, Laporte G, Semet F (1998) A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks* 32(4):263–273.

Girard AR, Howell AS, Hedrick JK (2004) Border patrol and surveillance missions using multiple unmanned air vehicles. *The 43rd IEEE Conference on Decision and Control*, volume 1, 620–625.

Golden BL, Levy L, Vohra R (1987) The orienteering problem. *Naval Research Logistics* 34(3):307–318.

Grocholsky B, Keller J, Kumar RV, Pappas GJ (2006) Cooperative air and ground surveillance. *IEEE Robotics and Automation Magazine* 13(3):16–25.

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255(2):315–332.

Hokayem PF, Stipanović DM, Spong MW (2007) On persistent coverage control. *The 46th IEEE Conference on Decision and Control*, 6130–6135.

Hollinger GA, Sukhatme GS (2014) Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research* 33(9):1271–1287.

Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* 30(7):846–894.

Laporte G, Martello S (1990) The selective travelling salesman problem. *Discrete Applied Mathematics* 26(2-3):193–207.

Maffioli F, Sciomachen A (1997) A mixed-integer model for solving ordering problems with side constraints. *Annals of Operations Research* 69:277–297.

Michael N, Stump E, Mohta K (2011) Persistent surveillance with a team of mavs. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2708–2714.

Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery* 7(4):326–329.

Nigam N, Kroo I (2008) Persistent surveillance using multiple unmanned air vehicles. *2008 IEEE Conference on Aerospace*, 1–14.

Smith RN, Schwager M, Smith SL, Jones BH, Rus D, Sukhatme GS (2011) Persistent ocean monitoring with underwater gliders: Adapting sampling resolution. *Journal of Field Robotics* 28(5):714–741.

Steimle ET, Hall ML (2006) Unmanned surface vehicles as environmental monitoring and assessment tools. *OCEANS 2006*, 1–5.

Tsiligirides T (1984) Heuristic methods applied to orienteering. *Journal of the Operational Research Society* 35(9):797–809.

Vansteenwegen P, Souffriau W, Van Oudheusden D (2011) The orienteering problem: A survey. *European Journal of Operational Research* 209(1):1–10.

Vidal T, Maculan N, Ochi LS, Vaz Penna PH (2015) Large neighborhoods with implicit customer selection for vehicle routing problems with profits. *Transportation Science* 50(2):720–734.

Water Supplies Department (2018) Trial use of unmanned surface vessel (USV) system for water quality monitoring and sampling at impounding reservoirs. Https://www.wsd.gov.hk/en/core-businesses/water-quality/my-drinking-water-quality/unmanned-surface-vessel-system/index.html. Accessed 27 December 2018.

Wivou J, Udawatta L, Alshehhi A, Alzaabi E, Albeloshi A, Alfalasi S (2016) Air quality monitoring for sustainable systems via drone based technology. *2016 IEEE International Conference on Information and Automation for Sustainability*, 1–5.

Xia J, Wang K, Wang S (2019) Drone scheduling to monitor vessels in emission control areas. *Transportation Research Part B: Methodological* 119:174–196.

Yu J, Schwager M, Rus D (2014) Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 342–349.

Yu J, Schwager M, Rus D (2016) Correlated orienteering problem and its application to persistent monitoring tasks. *IEEE Transactions on Robotics* 32(5):1106–1118.

Zhang Y, Zheng B, Fu G, Liu H (2006) On the assessment methodology and standards for nutrition status in channel type reservoirs based on zoning of eutrophication sensitivity. *Acta Scientiae Circumstantiae* 26(6):1016–1021.

Appendix A: Two-stage exact algorithm for the CCOP.

Stage 1: A sampling task generator (procedure **(a)**) produces a set of sampling tasks $Y = \left\{ y \in \{0,1\}^{|V|} \,|\, \sum_{i \in V} y_i \leq C_{\max},\ \sum_{i \in V_r} y_i \geq S_r^{\min},\ r \in R \right\}$. The coverage area size of $\boldsymbol{y} \in Y$ is denoted by $Area\,(\boldsymbol{y})$. A CCOP heuristic (procedure **(b)**) obtains the initial lower bound for the CCOP, that is, $Area^{(L)}$. A cut generation (procedure **(c)**) solves the CCOP relaxation problem to obtain the upper bound for the CCOP, that is, $Area^{(U)}$. We obtain a remaining task set $Y'(Y' \subset Y)$, in which all the tasks $\boldsymbol{y} \in Y'$ satisfy $Area^{(L)} \leq Area\,(\boldsymbol{y}) \leq Area^{(U)}$.

Stage 2: For each $\boldsymbol{y} \in Y'$, a TSP heuristic (procedure **(d)**) obtains the TSP upper bound, $\overline{TSP(\boldsymbol{y})}$. If $\overline{TSP(\boldsymbol{y})} \leq T_{\max}$, remove all inferior tasks $\boldsymbol{y}' \in Y'$ that satisfy $Area\,(\boldsymbol{y}') < Area\,(\boldsymbol{y})$ from Y' (procedure **(g)**). If $\overline{TSP(\boldsymbol{y})} > T_{\max}$, a TSP relaxation problem (procedure **(e)**) obtains the TSP lower bound, $\underline{TSP(\boldsymbol{y})}$. If $\underline{TSP(\boldsymbol{y})} > T_{\max}$, remove all infeasible tasks $\boldsymbol{y}' \in Y'$ that satisfy $\boldsymbol{y}' \geq \boldsymbol{y}$ from Y' (procedure **(h)**). If $\underline{TSP(\boldsymbol{y})} \leq T_{\max}$, an exact TSP solving procedure (procedure **(f)**) solves the TSP to optimality, that is, $TSP^*(\boldsymbol{y})$. If $TSP^*\,(\boldsymbol{y}) \leq T_{\max}$, remove all inferior tasks $\boldsymbol{y}' \in Y'$ that satisfy $Area\,(\boldsymbol{y}') < Area\,(\boldsymbol{y})$ from Y' (procedure **(g)**). If $TSP^*\,(\boldsymbol{y}) > T_{\max}$, remove all infeasible tasks $\boldsymbol{y}' \in Y'$ with $\boldsymbol{y}' \geq \boldsymbol{y}$ from Y' (procedure **(h)**). The final elements in Y' are the optimal solutions.