Int. J. Appl. Math. Comput. Sci., 2025, Vol. 35, No. 2, 293–309
DOI: 10.61822/amcs-2025-0021

EXACT ALGORITHMS FOR THE SATELLITE IMAGE SELECTION PROBLEM

SYLWESTER SWAT [a], MACIEJ ANTCZAK [a,b], TOMASZ ZOK [a], JACEK BLAZEWICZ [a,b], JEDRZEJ MUSIAL [a,*]

[a] Institute of Computing Science
Poznan University of Technology, ul. Piotrowo 2, 60-965 Poznan, Poland
e-mail: Jedrzej.Musial@cs.put.poznan.pl

[b] Institute of Bioorganic Chemistry
Polish Academy of Sciences, ul. Noskowskiego 12/14, 61-704 Poznan, Poland

Space development is more relevant than ever with the increasing number of satellite launches for various applications. The amount of space data collected daily is growing exponentially and many customers are interested in continuously monitoring different regions of the Earth. It often requires stitching together many images from other providers to cover an Area of Interest (AOI), resulting in a mosaic. Each satellite image has various parameters, such as cost, download time, cloud coverage, and resolution. The main question is how to optimally select the subset of available images to fully cover the AOI while minimizing total cost and cloud coverage. The problem is known as satellite image mosaic selection (SIMS). Manual selection of promising images is often impossible, especially when dealing with large AOIs or many photos. To solve the problem, we propose several new exact algorithms using different techniques, such as branch-and-bound or mixed-integer linear programming. These algorithms show quality and efficiency compared with existing approaches and are expected to benefit various industrial applications.

Keywords: satellite image mosaic selection, set cover, hitting set, plane division.

1. Introduction

Satellite images have been used for over 50 years to monitor Earth's ecosystems, including species diversity (Hall *et al*., 2012), populations (Sánchez-Azofeifa *et al*., 2011), carbon stocks and emissions (Asner *et al*., 2010), and changes in land cover (Hansen *et al*., 2008). However, their practical application depends on specific factors such as price, security regulations, cloud cover, spatial resolution, and geographic/temporal coverage (Goetz, 2007). Furthermore, high-resolution data are not available for all areas of the world.

In recent years, there has been a continuous increase in satellite launches dedicated to earth observation (Rovetto, 2017). The commercial satellite image market collects petabytes of data annually.

Satellite image mosaicking is essential for various applications, such as environmental monitoring and surveying (Verpoorter *et al*., 2014; Flood *et al*., 2019), urban development analysis (Henderson *et al*., 2012; Jean *et al*., 2016; Wang *et al*., 2020), socioeconomic dynamics monitoring (Bennett and Smith, 2017; Yeh *et al*., 2020), promoting sustainable development (Burke *et al*., 2021), automated construction of digital elevation models (Shean *et al*., 2016), and crop classification (Tian *et al*., 2020; Felegari *et al*., 2021) or conversely, plant invasion monitoring (Müllerová *et al*., 2017).

In this work, we address the satellite image mosaic selection (SIMS) problem (Combarro Simón *et al*., 2023). The primary objective is to create a mosaic using selected images that meet specific criteria. First, we introduce key concepts to describe the model and algorithms under consideration accurately. Then, we explore commonly used objectives and cost functions in the problem. We convert the representation of the problem from geometric to graph-based, a technique widely used in the design of highly efficient optimization algorithms (Lopez-Loces *et al*., 2016; Zok *et al*., 2020). We propose six novel algorithms that differ in the optimization techniques applied: (a) three algorithms employing enumeration, (b) two approaches integrating the branch-and-bound

*Corresponding author

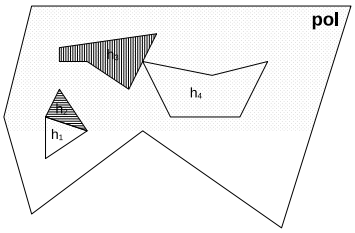

Fig. 1. An exemplary *p-polygon pol* with four holes h_1, h_2, h_3, h_4. Note that holes can have common points but not an intersection with a positive area.

concept, and (c) one method uses mixed integer linear programming. We perform comprehensive computational experiments to assess the performance of algorithms compared with state-of-the-art methods, using datasets obtained from (Combarro Simón *et al.*, 2023). We discuss the advantages and limitations of the proposed approaches to enable users to select the most suitable one for their requirements. Finally, we briefly outline the roadmap for the further development of the proposed solutions.

2. Background

2.1. Basic notions and definitions. Unless stated otherwise, all notions will be considered (when applicable) in a two-dimensional space $\mathbb{R}^2$.

A *p-polygon* (see Fig. 1) is defined as a simple polygon T with holes, where each hole (h_i) is also a simple polygon, and the interiors of the hole polygons do not intersect. In other words, a *p-polygon* can be represented as a pair $pol = (T, H)$, where T is a simple polygon and $H = \{h_1, \ldots, h_t\}$ is a set of interior-disjoint simple polygons contained within polygon T, that is

$$\underset{\substack{h_i, h_j \in H \\ i \neq j}}{\forall} area(h_i \cap h_j) = 0,$$

where $area(X)$ denotes the area of polygon X. We can use standard operations such as union, intersection, and a set difference on *p-polygons*. For example, the area of a *p-polygon* $pol = (T, H)$ is defined as

$$area(pol) = area(T) - \sum_{h_i \in H} area(h_i)$$

We also denote by $union(Q)$ the union of all *p-polygons* (pol_i) in a given set $Q = \{pol_1, \ldots, pol_t\} = \{(T_1, H_1), \ldots, (T_t, H_t)\}$ as

$$union(Q) = \bigcup_{pol_i \in Q} \left(pol_i \setminus \bigcup_{h \in H_i} h \right).$$

We will also use the standard notation 2^Q for all subsets of a given set Q.

2.2. Definition of the satellite image mosaic selection problem. Let A be a *p-polygon* referred to as the *area of interest (AOI)* which is divided into m unique *p-polygons*. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of *p-polygons* referred to as *photos* or *images*. Each photo p_i is associated with a cost denoted as c_i. The objective is to identify a subset $S \subseteq P$ that collectively covers the AOI while minimizing the total cost.

To formulate this problem, we introduce the binary decision variable x_i, where $x_i = 1$ if photo p_i is selected to cover the AOI, and $x_i = 0$ otherwise. Additionally, we define the binary parameter a_{ij}, which equals 1 if photo p_i covers the j-th subpart of polygon A, and 0 otherwise. This parameter allows us to define the coverage constraint, ensuring that each part of polygon A is covered at least once: Minimize

$$\sum_{i=1}^{n} c_i x_i, \tag{1}$$

subject to

$$x_i \in \{0, 1\}, \qquad i = 1, \ldots, n,$$
$$\sum_{i=1}^{n} a_{ij} x_i \geq 1, \qquad j = 1, \ldots, m.$$

This is a general formulation of the problem. In practical applications, variations often arise. For instance, a satellite photo provider might offer discounts on bulk purchases, or the end user might be satisfied with achieving, say, 90% coverage of the AOI at a significantly lower total cost. Hence, we introduce a generalized formulation involving a scoring function $score(A, S)$, which assigns a score to any given subset S, and a cost function $cost(S)$, which computes the total price for the selected photos. The objective of this reformulated problem is to find the subset S that minimizes $score(A, S)$.

In the subsequent section, we will discuss various approaches to defining both $score(A, S)$ and $cost(S)$ for real-world scenarios.

2.3. Example scoring and cost functions. Let us consider the following examples of score functions:

1. To cover the entire AOI using images while minimizing the total cost of selected images, the scoring function may look like this:

$$score(A, S) = \begin{cases} C(S) & \text{if } A \subset union(S), \\ \infty & \text{otherwise.} \end{cases} \tag{2}$$

Here, $C : 2^P \to \mathbb{R}$ is a cost function, for example, with $C(S) = \sum_{p_i \in S} c(p_i)$, and $c : P \to \mathbb{R}$ being

a function with non-negative values. This setup could apply, for example, when purchasing images, where buying the image p_i costs $c(p_i)$. The cost function C could be more complex, such as when purchasing images from different sellers with potential discounts for buying multiple photos from the same seller.

2. In a scenario where the requirement is to cover a fraction of the AOI, rather than the entire area, we may define the score function as

$$score(A,S) = \begin{cases} C(S) & \text{if } area(A \cap union(S)) \geq \\ & \alpha \cdot area(A), \\ \infty & \text{otherwise.} \end{cases}$$

Here, $\alpha \in [0,1]$ is a real-valued parameter. Any subsets of images that do not cover a fraction α of the area of A will be considered invalid, with the score value set to ∞.

3. The third approach combines elements of the previous two. To allow a solution S of an arbitrary coverage ratio $\frac{area(A \cap union(S))}{area(A)}$ while favoring higher coverage ratios, an additional function $g : [0,1] \to [0,1]$ can be introduced to penalize low-coverage solutions. This function, for example, could be piecewise linear:

$$g(x) = \begin{cases} \epsilon & \text{if } x < \alpha, \\ \frac{\beta-1}{\alpha-1} \cdot x + \frac{\alpha-\beta}{\alpha-1} & \text{otherwise.} \end{cases}$$

Here, $\alpha, \beta \in [0,1]$ are fixed parameters, and ϵ is a small constant (such that the value ϵ^{-1} can be treated in practice in the same way as ∞). The parameter α dictates the minimum value of x (representing the coverage of the Area of Interest) that is considered meaningful. Specifically, for $x < \alpha$, the function $g(x)$ is set to ϵ. Conversely, the parameter β determines the value of $g(x)$ for $x \in [\alpha, 1]$ such that $g(\alpha) = \beta$. Thereafter, the function increases linearly, reaching $g(1) = 1$. The score function then takes the form

$$score(A,S) = \frac{C(S)}{g\left(\frac{area(A \cap union(S))}{area(A)}\right)}.$$

In this setup, solutions with low coverage will have their score multiplicatively penalized by the value of $g(x)^{-1}$, where $x = \frac{area(A \cap union(S))}{area(A)}$.

In the proposed scoring functions, we use the cost function $C : 2^P \to \mathbb{R}$. This cost function significantly influences the optimization algorithms required to produce a valid and high-quality solution. Let us now consider a few examples of possible cost functions.

1. $C(S) = \sum_{p_i \in S} c(p_i)$

The cost function $C(S)$ is given by the sum of costs of individual images in subset S. When all costs $c(p_i)$ are set to 1 for all images $p_i \in P$, the SIMS optimization problem becomes equivalent to the set cover problem. This involves finding the smallest subset $S \subseteq F$, for a given universe U and a family of sets $F = \{P_1, \ldots, P_n : P_i \subseteq U\}$ such that each element in the given universe U is covered by at least one set in $P_i \in S$.

When non-uniform image costs are considered, the weighted variant of the set cover problem arises, where the goal is to minimize the sum of costs, rather than the size of the subset. One common scenario is when costs are proportional to the area of polygons, i.e., $c(p_i) = \text{area}(p_i)$.

2. Additionally, the cost function can be designed to penalize solutions in which regions in A are covered by a high number of images from set P. For example, a penalizing function can be expressed as

$$C(S) = \sum_{p_i \in S} c(p_i) \cdot \left(1 + \frac{area(Z)}{area(A)}\right).$$

Here, Z represents points in A covered by at least two elements in S. A more rigorous penalizing function can be defined as:

$$C(S) = \sum_{p_i \in S} c(p_i) \cdot \left(1 + \sum_{\substack{p_i, p_j \in S \\ i \neq j}} area(p_i \cap p_j)\right).$$

3. In real life, sellers offer discounts depending on the items purchased. For example, a seller can provide a 30% discount for the least expensive product from the purchased items. To consider such situations in considered models, the cost function C must also consider sellers from which items are bought.

When solving optimization problems, selecting the appropriate objective function scoring and cost functions C and c are essential.

2.4. Problem transformation. In the SIMS problem, we are given an AOI A and a set of images P. To apply optimization techniques and find a subset $S \subseteq 2^P$ for which the value $score(A,S)$ is as tiny as possible, we need to transform the geometric representation of the problem into a more manageable form. A natural approach is to convert the geometric representation to a graph-theoretic approach and then apply optimization algorithms that operate purely on the graph, eliminating the cumbersome geometry. In this context, we propose

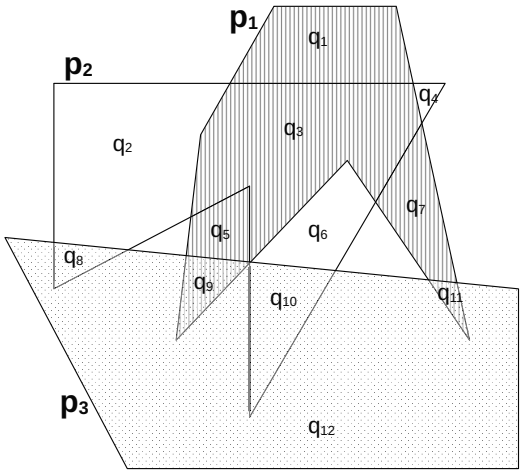

Fig. 2. An example of a plane division created for a set T and an empty set $A = \emptyset$. Set T contains three polygons p_1, p_2, p_3 without holes (polygons marked by different hatching styles). Set $div_A(T)$ contains 12 polygons $q_i, i \in [1, 12]$.

transforming the SIMS problem into a variation of the hitting-set problem, which is natural since the set cover and hitting-set problems are dual.

Let us introduce a few notions. A plane division concerning AOI A and set $T = \{t_1, \ldots, t_k\}$ of *p-polygons* is a minimum-size set of *p-polygons*

$$div_A(T) = \{q_1, \ldots, q_m : area(q_i \cap q_j) = 0 \text{ for } i \neq j\}$$

such that each *p-polygon* in $T \cup A$ has a unique representation as a union of elements from $div_A(T)$ (see Fig. 2).

For a set $div_A(T)$, we create an undirected bipartite graph $G = (A_G, B_G, E_G)$, called the *hitting-set graph*, where

$$A_G = \{a_1, a_2, \ldots, a_{|T|}\},$$
$$B_G = \{b_1, b_2, \ldots, b_{|div_A(T)|}\},$$
$$E_G = \{\{a_i, b_j\} : a_i \in A, b_j \in B, q_j \subset t_i\}.$$

In other words, there is an edge between node a, representing a *p-polygon* from T, and node b, representing a polygon from plane division $div_A(T)$ if and only if the second one is contained in the first one.

Figure 3 shows a bipartite graph structure created for a plane division from Fig. 2.

For the SIMS problem (A, P), we can create a bipartite graph G for the plane division $div_A(P)$. Since we are interested in covering only the nodes representing the AOI, we prune the graph of vertices in B_G that do not represent *p-polygons* from the AOI. The problem can

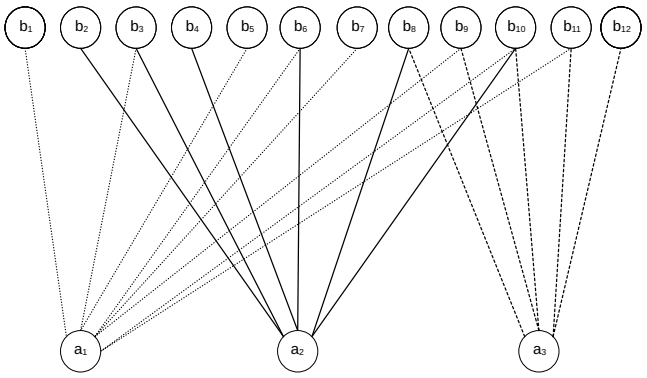

Fig. 3. Structure of the bipartite graph created for plane division from Fig. 2. Nodes a_i represent the photos under consideration, while nodes b_j represent unique fragments of the area of interest (AOI). The objective is to identify a subset of nodes a_i that ensures connectivity to all nodes b_j while minimizing the associated cost of the selected subset.

then be restated as a graph-theoretic concept: find a subset $S \subset A_G$ that minimizes a given objective $score(A, S)$. In this context, the scoring functions are formulated for the graph nodes, and the cost functions C and c are associated with the weights assigned to the nodes from A_G. For example, using the objective (2) with a constant cost function $c(p_i) = 1$, the problem becomes finding the smallest subset $S \subseteq A$ that covers all elements from B_G. This problem is known as the *hitting-set problem*. It can be shown that with such $score$ and c functions and without additional constraints on the shape of polygons, the SIMS and hitting set problems are equivalent.

It is well known that the hitting set problem is NP-complete and $W[2]$-hard when parameterized by the solution size (Cygan *et al.*, 2015). This suggests that it is unlikely to devise a theoretically efficient algorithm to solve the SIMS problem, even though the size of optimal solutions is relatively tiny in real-world data (see Section 5.2).

2.5. Some obstacles and how to overcome them: sampling points approach. When solving hitting-set instances using algorithms such as MILP formulations, a challenge arises in creating the bipartite graph G. The primary issue relies on the efficient creation of the plane division $div_A(T)$. There are several problems, one of the most significant being the potential size of $div_A(T)$, which can be of the order of $O(N^2)$, where N is the total number of sides of polygons (including holes) in the set $A \cup T$. Furthermore, each *p-polygon* representing an image may contain $O(N^2)$ elements of $div_A(T)$, leading to a potential number of edges in the graph G of order $O(n \cdot N^2)$. This presents a formidable time and memory

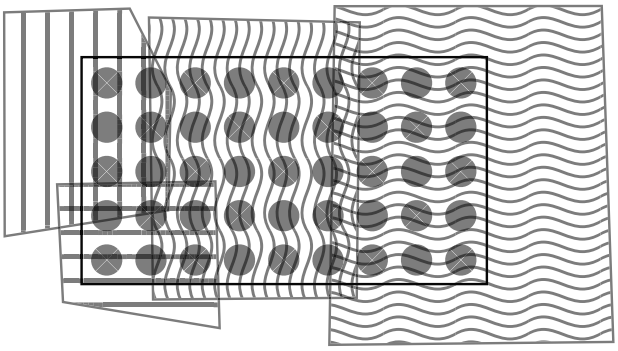

Fig. 4. Example featuring four images displaying various patterns (lines or waves) and a rectangular AOI with sampling points arranged in a 9×5 grid.

complexity barrier when using algorithms, especially for large problem instances where optimal solutions are sought.

Fortunately, we do not need to find optimal solutions. Therefore, we will now consider an alternative way of creating a bipartite graph for the given SIMS problem. We call this approach a *sampling points approach*. This approach is much simpler than the plane division concept. Still, it does not guarantee that the optimal solution to the hitting-set problem will correspond to the optimal solution of the considered SIMS problem.

By B_A we denote the bounding rectangle $\{(x,y),(x+d_x,y),(x+d_x,y+d_y),(x,y+d_y)\}$ of *p-polygon* A. Let us now consider points spaced equidistantly within *p-polygon* B_A. These points form a grid $X \times Y$:

$$\left\{(x+\frac{i}{X-1}\cdot d_x, y+\frac{j}{Y-1}\cdot d_y) : i \in \mathbb{N}_X, j \in \mathbb{N}_Y\right\}$$

where $\mathbb{N}_n$ denotes the set of natural numbers less than n.

In the following, we will refer to the points on the grid as "sampling points." Let us define D as the set of points on the grid that are also in A, written as $D = \{d \in Grid : d \in A\} = \{d_1, d_2, \ldots, d_{|D|}\}$.

Consider the SIMS problem (A, P), where $P = \{p_1, \ldots, p_n\}$. For each *p-polygon* $p \in P$ and each point $d \in Grid$, we check if d belongs to p. Then, we construct a hitting-set graph $G = (A_G, B_G, E_G)$, similarly to the approach used for plane division:

$$\begin{aligned} A_G &= \{a_1, a_2, \ldots, a_n\}, \\ B_G &= \{b_1, b_2, \ldots, b_D\}, \\ E_G &= \{\{(a_i, b_j\} : a_i \in A, b_j \in B, d_j \in p_i\}. \end{aligned}$$

It is essential to sample points in the set D that correspond to plane points belonging to *p-polygons* of the set $div_A(T)$. The denser the points are within A, the more accurate the model we should obtain. Figure 4 illustrates an example instance with sampling points. However, this is not always true for the "small" values of X and Y, as shown in Section 5.

To put it more formally, what we mean is that for any $\epsilon > 0$, there exist values X and Y such that for any two *p-polygons* p_i and $p_j \in div_A(T)$, we obtain the inequality

$$\left|\frac{area(p_i)}{area(p_j)} - \frac{r_{p_i}}{r_{p_j}}\right| < \epsilon,$$

where r_{p_i} and r_{p_j} denote the number of points from D (created for given values X and Y) that belong to polygons p_i and p_j, respectively.

The sampling points approach offers several advantages. One key benefit is that there is no need to create plane divisions. All we have to do is check if a point d is in the set P for each *p-polygon* $p_i \in P$ and each point $d \in D$. Therefore, the algorithm is relatively simple to implement.

The process of checking for a *p-polygon* $p_i \in P$ with a total of n_i points (including holes) can be easily accomplished in $O(n_i)$ time using the ray-shooting algorithm. First, we check whether d belongs to any holes in p_i. If it does, then d does not belong to p_i. Otherwise, we check whether d belongs to a simple polygon that forms the boundary of p_i. As a result, we obtain a complexity of $O(n_i \cdot |D|)$ for point $p_i \in P$, and the complexity of creating the bipartite graph using the sampling points approach is $O(N \cdot |D|)$, where $N = \sum_{p_i \in P} n_i$. This can also be expressed as $O(n \cdot n_{\max} \cdot |D|)$, where $n_{\max} = \max_{p_i \in P}\{n_i\}$.

Additionally, the size of the created graph is at most of the order of $O(n \cdot |D|)$.

Creating a bipartite graph using the sampling-points approach can be approached differently. We can use a faster method to check if a point $d \in D$ belongs to the *p-polygon* p_i. For example, a point location algorithm could answer such point-belonging queries in $O(\log n_i)$ time but requires $O(n_i \cdot \log n_i)$ preprocessing time. Therefore, we can develop an algorithm to create G that runs in total time $O(N \cdot \log n_{\max} + n \cdot |D| \cdot \log n_{\max})$ or a simpler $O(N \log N + n \cdot |D| \cdot \log N)$. It is important to note that the point location algorithm is much faster, but only if the considered polygons are large enough. In our application, where most polygons represent images and have a small number of edges (usually less than 10), those algorithms are slower as they incur extra constant-factor overhead to the running time.

Table 1 summarizes the advantages and disadvantages of the plane division and sampling point approaches.

At the end, it is also worth noting that the sampling points approach might be considered a variation of the patch cover problem, where the goal is to cover pixels of a plane or an image with pixels originating from other

images. Operating on a pixel scale, however, would result in a significant increase in the problem size and seriously affect the performance of the algorithms used to find solutions. The sampling points approach allows us to easily control the problem size and is invulnerable to issues related to the properties of the images used (e.g., images differing in format or resolution).

3. Algorithms

Specifying the objective function when designing algorithms to solve the SIMS problem is necessary. We will focus on solving the weighted hitting set problem from this point forward. This problem involves finding a subset S of vertices from a bipartite graph $G = (A, B, E)$, where A and B are two disjoint sets of vertices and E represents the edges between them. Additionally, there is a function c, which assigns a non-negative actual number to each vertex in the set A. The goal is to find a subset $S \subseteq A$ with the minimum possible cost $C(S)$ while satisfying the condition that every vertex in the set B is covered by the vertices in S. The cost function $C(S)$ is calculated by adding the costs of the vertices in S, $C(S) = \sum_{v \in S} c(v)$ (see Section 2).

We consider two different functions: $c(v) = 1$ and $c(v) = deg(v)/|B|$. Minimizing the objective $C(S)$ using the first function helps us find a hitting set with the minimum number of vertices, while the second function operates on the relative area of each image represented by a vertex v and enables us to find a subset of images with a minimal coverage overhead, provided that a sufficient number of sampling points were used to create the graph. The coverage overhead is calculated by subtracting the cost of the subset of vertices which correspond to sampling points covered by the area of interest (represented by the polygon A_{AOI}) from the sum of the costs of the vertices corresponding to sampling points covered by the images in the set P. By the definition of c, this translates to the difference of the areas of all the images in the set P and the area of the AOI:

$$\sum_{p_i \in P} area(p_i) - area(A_{\text{AOI}}).$$

We have designed and implemented several algorithms to solve this problem. Now, let us briefly describe these algorithms. First, for convenience, set $n = |A|$ and $M = |B|$.

brute This algorithm is based on enumerating all possible subsets $S \subseteq 2^A$. For each subset S, it needs to be checked if $N(S) = B$. A straightforward approach is to check each subset S independently, resulting in time complexity of $O(n \cdot M \cdot 2^n)$. Since the number of subsets to check is usually very large (exponential in $|A|$), it is essential to reduce the time required to process each set as much as possible. To achieve this, we enumerate all subsets $S \subseteq 2^A$ using Gray codes so that two consecutively visited subsets differ by exactly one element. This allows us to iterate over neighbors $u \in N(v)$ for only one node $v \in S$ for each set S. By keeping cumulative values of the number of nodes covering elements in B, the complexity improves from $O(n \cdot M \cdot 2^n)$ to $O(M \cdot 2^n)$. In our implementation, we applied an additional trick to improve performance further. By noting that the i-th element from set A will be considered 2^{n-i} times as the changed element during transitions between subsets of 2^A, we can reduce the number of performed operations by remapping nodes $a_1, \ldots, a_n \in A$, so that their degrees $deg(a_i)$ form a non-decreasing sequence. Although this improvement significantly decreases the number of performed operations in practice, the theoretical complexity of the algorithm remains unchanged.

minimal_subsets_one_sided **(MS1)** The main idea behind this algorithm is to enumerate only subsets of A that are inclusion-minimal solutions to the SIMS problem. The enumeration of those subsets is done recursively, and most non-minimal subsets are not visited at all. We start with an empty set S and iteratively add elements to S. An element $u \in A$ is a candidate node and can only be added to S if $N(u) \setminus N(S) \neq \emptyset$. If an element can be added, we consider two cases: adding or not adding the node to S, and recursively find solutions for both cases. By keeping track of the number of nodes in A that cover elements of B, it is enough to check only nodes $u \in N(v)$ for the added node v to determine if the current set S is a valid solution. Hence, the algorithm's complexity is $O(R \cdot M)$, where R is the number of visited elements of 2^A, and the required additional memory is of order $O(n + M)$. Please note that even though we can expand set S by a node for which the set $N(u) \setminus N(S)$ is not empty, this does not guarantee that all subsets visited by the algorithm will be minimal. Indeed, in some future recursive call, there might be an added node u to S for which there will exist a node $v \in S$ such that $N(v) \subseteq N((S \cup \{u\}) \setminus v)$.

minimal_subsets_two_sided **(MS2)** This algorithm is an extension of the previous one, aiming to visit only minimal subsets. To achieve this, we ensure that set S will never be expanded by a node u if there exists a node $v \in S$ such that $N(v) \subseteq N((S \cup \{u\}) \setminus v)$. To efficiently enforce this, we need to keep track of the number of elements in B that are exclusively reached by each node $a \in A$. This is done by iterating over sets $N(N(v))$ when considering whether to add node v and dynamically adjusting temporary values

Table 1. Summary of the advantages and disadvantages of creating graph G using the plane division approach and the sampling points approach.

	Plane division	Sampling points
Time complexity	at least $O(n \cdot N^2)$	$O(N \cdot \lvert D\rvert)$ using ray-shooting algorithm $O(N \log N + n \cdot \lvert D\rvert \cdot \log N)$ using point-location algorithm
Maximum graph size	$O(n \cdot N^2)$	$O(n \cdot \lvert D\rvert)$
Control of graph size	NO	YES $\lvert D\rvert = X \cdot Y$ values X, Y can be chosen arbitrarily
Exact results guarantee	YES	NO
Implementation complexity	complex	relatively easy

as needed. The algorithm's complexity is therefore $O(R_{min} \cdot M \cdot n)$, where R_{min} is the number of visited subsets. In this case, we can guarantee that all visited subsets are minimal, so the set S under consideration will never be a superset of a valid solution. However, this comes with the additional cost of iterating over $N(N(v))$ instead of just $N(v)$ when considering node v as a potential candidate to add to S.

branch_and_bound **(b&b)** This algorithm is based on the branching idea applied in the *minimal_subsets_one_sided* algorithm. In addition, we apply a simple lower-bound estimate of the solution cost to trim the search space. The lower bound is calculated as follows. Let $B_S = B \setminus N(S)$ and let $r_a = \frac{c(a)}{|N(a) \setminus N(S)|}$ for $a \in A$. The lower bound is then given by $|B_S| \cdot \min_{a \in A \setminus S'}\{r_a\}$, where S' is a set of all nodes considered earlier as candidates, regardless of whether they were added to S or not. It can be observed that this lower bound is valid, as it represents the minimal cost required to cover a single unhit element multiplied by the number of such elements. Selecting a candidate as the branching node, we choose the vertex $v \in A \setminus S'$ with the smallest value of $\frac{c(v)}{|N(v) \setminus N(S)|}$. To calculate the values of r_a, we need to iterate over the neighborhoods of all nodes in the set $N(A \setminus S')$. This gives us a complexity of $O(R \cdot E)$. It is worth noting that in our implementation we do not check the lower bound at each branching step but rather every third step. This causes the number of visited states to increase slightly but significantly reduces the number of operations needed to find the bound. We have observed that this enhancement allows for a roughly $\frac{1}{3}$-factor speedup, although the exact speedup depends on the characteristics of the input.

branch_and_bound_trace **(b&b-T)** This is a modified version of the previous algorithm. Here, we keep track of values that are necessary to quickly determine the lower bound value and the best candidate node. This is done without the need to recompute them from scratch by iterating over nodes in $N(A \setminus S')$. When a node v is added to set S, we iterate over all nodes $a \in N(N(v) \setminus N(S))$ and modify the corresponding values $r_a = \frac{c(a)}{|N(a) \setminus N(S)|}$, which are stored in an auxiliary array. This allows us to find the lower bound and the best candidate by simply iterating over nodes in $A \setminus S'$, resulting in a time complexity of $O(n)$ time. Compared with the *branch_and_bound* algorithm, this approach enables us to find the lower bound much faster. However, it comes at the cost of increased time needed to keep the necessary values updated, as we need to iterate over $N(N(v) \setminus N(S))$ instead of just $N(v)$. Although the worst-case complexity is not better than that of the *branch_and_bound* algorithm, it is reasonable to expect this algorithm to be much faster in practice. The time complexity of this algorithm can be expressed as $O(R \cdot n \cdot \Delta)$, where Δ is the maximum degree of a node in A. Additionally, if the node degrees are distributed "uniformly," it can be shown that the complexity can be expressed as $O(R \cdot n + R \cdot n \cdot \frac{\Delta^2}{M})$, which simplifies to $O(R \cdot n)$ for graphs with $\Delta \ll M^{\frac{1}{2}}$.

milp This algorithm is based on the integer linear programming approach. The problem can be formulated as follows: minimize the value $\sum_{a \in A} c(a) \cdot x_a$ subject to M given constraints (created for elements $b \in B$) of the form: $\sum_{a \in N(b)} x_a \geq 1$, where x_a are Boolean variables (or, alternatively, integers from the set $\{0, 1\}$). This is a standard integer linear program formulation of the hitting set problem, and it can be solved using state-of-the-art solvers that apply a variety of optimization techniques. In our experiments, we used the HiGHS solver (Huangfu and Hall, 2018),

which is implemented in the SciPy software (Virtanen *et al*., 2020).

In Table 2, we compare the running times and memory usage of the algorithms we created.

4. Experiments

4.1. Setup. All experiments were carried out on a computer equipped with an Intel Core i9-9880H processor, 64 GB of DDR4 RAM (2667 MT/s), and the Linux Mint 21 "Vera" operating system. To ensure that processes do not compete for cache memory resources, each experiment was run sequentially using a single core and a single thread. However, the *milp* algorithm was an exception and was run in its default mode, utilizing the 8 available cores and 16 threads. In the results section, we demonstrate why this does not invalidate the comparison of *milp* with other algorithms. Additionally, CPU scaling was disabled during the experiments to minimize any noise in the measured running times.

4.2. Datasets. To replicate the projection of various real-world tasks, we selected five areas (AOI) from different parts of the world. The choices were Mexico City (Mexico), Rio de Janeiro (Brazil), Paris (France), Lagos (Nigeria), and Tokyo Bay (Japan). For each of the designated areas, we reviewed the available images using constellations such as Pléiades (Airbus, 2023a), Pléiades Neo (Airbus, 2023b), and finally SPOT (Airbus, 2023c). These constellations were chosen because they provide a comprehensive set of data and metadata that are crucial to our analysis. The choice of just these three data sources was not accidental. The indicated constellations contain all the metadata used in our computational experiment, so they represent real market situations in the best possible way. Other constellations known to us did not include several necessary parameters, often key ones, such as the angle of incidence. We carefully selected a specific time frame to conduct a focused analysis. The analysis was based on images taken between January 1, 2021, and January 1, 2023. This period was chosen to ensure the relevance and accuracy of the data. For the dates and constellations indicated, we downloaded the available images.

We aim to provide a comprehensive analysis by generating seven instances for all defined locations except Lagos. Each of these instances differs from each other in the number of images given to cover the AOIs. The number of images for the instances was 30, 50, 100, 150, 200, 250, and "500", where "500" denotes that all the available images were used for a given region and that there were not necessarily 500 images. The number of images in the "500" instances is 347 for Mexico City, 349 for Rio de Janeiro, 339 for Paris, and 493 for Tokyo Bay. For Lagos, the total number of images available for the specified date range was 145, so the number of images for Lagos instances was 30, 50, 100, and 145.

In our experiment, we did not consider clouds that might be present in images and cover some areas. The cloud detection problem is parallel to ours and does not affect our algorithms (as they operate on the hitting-set graph's structure and abstract from any other context), though taking into account clouds might increase the time needed to construct the graph and decrease the size of the graphs (if built using sampling points approach). It should be stressed that, though the graph size might decrease if we consider clouds and use the sampling-points approach, it need not mean that the instance will be easier to solve. How it affects the performance of algorithms in practice remains to be verified experimentally and is a plan for future research.

To compare implemented algorithms, we created three instance classes. The first class comprises images of the instances described earlier. The second class, called *incremental*, was created using a greedy heuristic. From the set of all available images for a given area, we created a sequence of those images in the following way: we iteratively selected an image that covered a sampling point that was already covered by the least number of images already considered. If there were multiple such sampling points, then we selected an image that covered the greatest number of those points (and if there was still a tie, any of the tied images were selected). For the found order of images, created instances contain an increasing number of initial images of the order. In this way, we guarantee that consecutive instances (with increasing size) are supersets of all smaller instances. This enables us, e.g., to track how the solution size or the running time changes with an increasing instance size. It also creates an opportunity to analyze the "behavior" of a greedy heuristic used to create the order of images. The third class, called *random subsets*, was created by randomly selecting subsets of all images available for a given instance. These test cases were created only for the Lagos area.

5. Results

5.1. Building the hitting-set graph. In this section, we report the results and analyze the influence of instance size (number of images) and the number of sampling points on the running time of the algorithm used to build the hitting-set graph. For each considered region and instance size, we build the hitting-set graph using 4096, 8192, 16384, 32768, and 65536 sampling points, respectively. We conducted the experiment 10 times for each set of parameters and reported the mean value. Figure 5 displays the running times for different instance sizes and the numbers of sampling points for the Rio de Janeiro instance. We do not provide the results for other

Table 2. Comparison of algorithms, their running time complexity, and additional memory usage (in addition to storing the hitting-set graph structure). The values R and $R_{\min}$ are highly dependent on the input characteristics. The value of R can be significantly reduced by applied optimization techniques, such as data reduction rules, node remapping, or the application of different lower-bound methods.

Algorithm	Time complexity	Additional memory overhead
brute	$\frac{O(M \cdot 2^n)}{\sum_{i=1}^{n} 2^{n-i} \cdot deg(a_i)}$	$O(M)$
MS1	$\frac{O(R \cdot M)}{\text{set } R \text{ of checked solutions is a superset of all minimal solutions}}$	$O(n+M)$
MS2	$\frac{O(R_{\min} \cdot M \cdot n)}{R_{\min} \text{ – number of all minimal solutions}}$	$O(n+M)$
b&b	$\frac{O(R \cdot E)}{R \text{ – number of checked solutions}}$	$O(n+M)$
b&b-T	$\frac{O(R \cdot E)}{O(R \cdot n + R \cdot n \cdot \frac{\Delta^2}{M})}$ for graphs with "uniform" degree distribution and degrees bounded by Δ	$O(n+M)$
milp	depending on the ILP solver	$\frac{\text{depending on the ILP solver}}{\text{usually an } O(n \cdot M) \text{ overhead to store constraints}}$

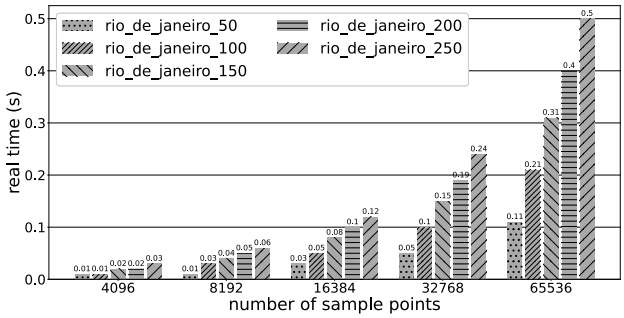

Fig. 5. Mean running times for different instance sizes (number of images) and sampling points used to create the hitting-set graph for the Rio de Janeiro area.

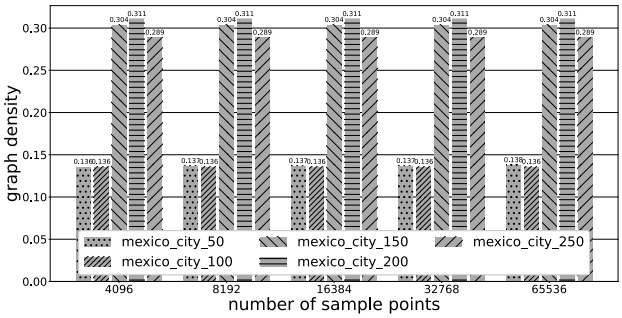

Fig. 6. Graph density for different values of instance size and sampling points for the Mexico City area.

instances as they are similar.

As observed, the running time increases in a linear fashion as the test size or the number of sampling points increase. This aligns with our expectations, as the ray-shooting algorithm has a linear-time complexity with respect to these parameters (when treated independently).

In Fig. 6, we illustrate how the density of the created hitting-set graph changes with an increase in the instance size and the number of sampling points. It can be noticed that the density of the graph does not depend (almost) on the number of sampling points used to create the hitting-set graph. However, it does vary for different instance sizes. It is important to note that the density is not dependent on the number of images in the instance, but rather on the sizes of the images. For example, in the case of Mexico City, instances with at least 200 images contain many images with much larger areas compared with smaller instances, resulting in a significantly increased graph density.

5.2. Size of optimal result. Let us now examine how the number of sampling points used influences the optimal result of the problem. It can be observed that for a fixed instance (e.g., Paris 100, see Fig. 7), the obtained result only slightly depends on the number of sampling points used. Intuitively, one might expect the result values to form a nondecreasing sequence as the number of sampling points increases and becomes denser in the AOI area. However, in our experiment, this is not the case, as the number of sampling points used (ranging from 625 to 40000) is below the point where it can be proved that the optimal result values for n and n_0 are equal. (There exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ the values of optimal results obtained for n and n_0 are equal. This is easily seen if we consider images as a set of pixels: by taking a sufficiently dense set of sampling points, each pixel of every image will cover at least one sampling point, hence covering all vertices representing sampling points will be equivalent to covering all pixels of the AOI.) What is most interesting and worth noting is that the value of the optimal solution found for 15625 and 30625 sampling points is smaller than that for 10000 and 22500, respectively. This implies that the distribution of sampling points affects the optimal result for the problem and is more important than just the density of those points within the area of interest.

Another important point to emphasize is the size of the optimal solution obtained for specific instances. In the case of Paris, the number of images in the optimal solution varies from 6 (Paris 50) to 9 (Paris 100). Similarly, for other locations, the optimal result sizes are also relatively small, not exceeding 10.

5.3. Comparison of algorithms and instance classes. In this subsection, we compare the algorithms implemented. For each set of parameters being compared, we run ten independent instances. The only exception is the *random subsets* class, where the algorithms were run on 25 randomly selected subsets of available images. This was done to provide more accurate results by increasing the sample size. Unless otherwise specified, the reported values are the means of those 10 (or 25) runs. Hitting-set graphs created for test cases in the *random subsets* class were built using 900 sampling points (a grid 30×30). It is also important to note that all algorithms running in the *random subsets* class had their subsets of images selected independently from each other. This means that each algorithm operated on a different set of 25 test cases for a given size.

In Table 3 we compare the running time of algorithms obtained for different sample sizes and different numbers of sampling points used to create a hitting-set graph. The comparison is based on the Rio de Janeiro area and the incremental class instances. What can be observed is that the algorithms behave similarly and exhibit almost linear scaling with an increasing number of sampling points. This aligns with expectations, as the theoretical complexity of nearly all designed algorithms depends linearly on the number of sampling points used. Running times obtained for other areas exhibit the same linear-scaling characteristics.

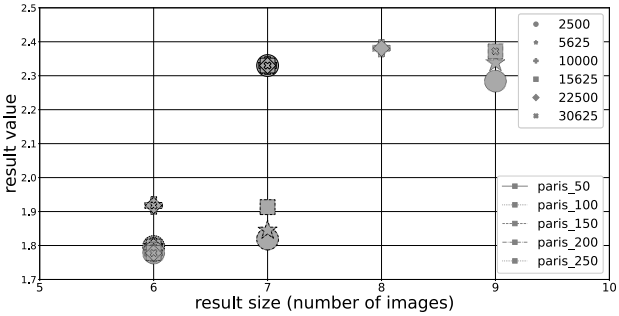

Fig. 7. The result size and result value of an optimal solution obtained for different sizes and number of sampling points used in the Paris area. Line styles of plotted markers and their transparency represent the instance size, shapes of markers indicate the number of sampling points used.

Comparing running times, the $brute$ algorithm is the slowest. This is expected because the algorithm needs to enumerate all subsets 2^n. The $MS1$ and $MS2$ algorithms are faster than $brute$ but slower than the proper branch-and-bound algorithms. It is interesting to note the difference between the $MS1$ and $MS2$ algorithms. Although the latter checks fewer solutions than the former, the time overhead caused by the need to keep track of additional data to make updates makes it considerably slower. Subsequently, the algorithms $b\&b$ and $b\&bT$ follow, with the latter being faster. The speedup obtained by eliminating the need to iterate over the neighbors of the nodes in $A \setminus S'$ is not negated by the necessity of iterating over $N(N(v))$ when v is added to S, and this fact is clearly visible. An interesting sudden fall in the number of solutions visited for the incremental class can be noticed when the instance size increases from 86 to 90. This is caused by considering new images that affect the value of the optimal result found. Consequently, due to the use of power bounding, a decreased total number of checked solutions is produced.

The $milp$ algorithm is the only one that was able to efficiently find optimal solutions for all the instances used, regardless of the size or class of the instance. In our experiment, it also does not exhibit exponential growth in running time but seems to scale linearly with the hitting-set graph size, suggesting that the algorithm does hardly any branching and most of the work is done by optimization techniques, such as data reduction rules. The $milp$ algorithm is not the fastest for instances with fewer images, especially when considering that it was run in

Table 3. Integers in the first column denote the number of images, while the three real values in other table cells represent the time in seconds needed to find an optimal answer for an instance, where the hitting set graph was built for a grid consisting of 30×30, 40×40 and 50×50 points, respectively. A dash indicates that an algorithm has not been run for the given number of images. Data was obtained for the Rio de Janeiro area.

	MS1			MS2			b&b			b&b-T			brute			milp		
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.3	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.4	0.5	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.7	1.1	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	1.3	2.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	1.6	2.4	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.8	3.2	4.9	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
38	0.0	0.0	0.1	0.1	0.3	0.4	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
42	0.0	0.1	0.1	0.4	0.9	1.0	0.0	0.0	0.0	0.0	0.0	0.0		–		0.0	0.0	0.0
46	0.2	0.3	0.5	1.3	2.7	3.2	0.0	0.1	0.1	0.0	0.1	0.1		–		0.0	0.0	0.0
50	0.4	0.9	1.4	2.8	5.4	6.6	0.1	0.1	0.1	0.0	0.1	0.1		–		0.0	0.0	0.0
54	1.5	3.9	5.9	14.3	31.2	31.7	0.2	0.3	0.5	0.1	0.2	0.3		–		0.0	0.0	0.0
58	4.5	12.4	23.0	50.5	90.0	112.8	0.2	0.3	0.5	0.1	0.1	0.2		–		0.0	0.0	0.0
62		–			–		0.3	0.6	1.0	0.1	0.2	0.3		–		0.0	0.0	0.0
66		–			–		1.1	1.8	3.1	0.2	0.3	0.5		–		0.0	0.0	0.0
70		–			–		2.2	3.7	6.7	0.4	0.7	1.1		–		0.0	0.0	0.0
74		–			–		3.2	5.6	9.8	0.6	1.1	1.6		–		0.0	0.0	0.0
78		–			–		6.4	11.3	18.2	0.8	1.5	2.0		–		0.0	0.0	0.0
82		–			–		15.4	23.0	38.5	2.2	3.9	5.7		–		0.0	0.0	0.0
86		–			–			–		2.9	4.9	7.2		–		0.0	0.0	0.0
90		–			–			–		1.4	2.1	3.4		–		0.0	0.0	0.0
94		–			–			–		1.9	2.9	4.4		–		0.0	0.0	0.0
98		–			–			–		2.4	4.0	5.9		–		0.0	0.0	0.0

the default configuration on 16 threads, not on a single thread. However, the running times for these instances are negligible for most application use cases.

We now proceed to the analysis of experiments performed for the incremental and random subsets classes. In Table 4, we present the results of analyzing the impact of the used instance class. Based on the Lagos area, we compare the performance of the algorithms.

Firstly, we observe that test cases in the random subsets class are generally much more difficult to solve optimally (as measured by the time required) compared to those in the incremental class. The exceptions to this are the $brute$ and $milp$ algorithms. The $brute$ algorithm considers all possible subsets of images, and its running time depends on the sizes of the images (degrees of the nodes in set A in the hitting-set graph). However, the images in the incremental class tend to be larger than those in the random subsets class, as they are chosen based on a heuristic that prioritizes larger images at the beginning of the image order. On the other hand, the $milp$ algorithm does not show this tendency and appears to scale linearly with the size of the graph (see Table 5).

We conducted one-sided Wilcoxon signed-rank tests to compare the performance of various algorithms on the Lagos dataset, specifically focusing on the incremental class (Table 4). This statistical test evaluates whether the running times of one algorithm are significantly shorter than those of another. The resulting p-values are illustrated in Fig. 8

Based on a significance threshold of $p < 0.05$, the following conclusions can be drawn:

Table 4. Integers in the first column denote number of images, while the six real values in other table cells represent the time in seconds required for the algorithms $MS1$, $MS2$, $b\&b$, $b\&b$-T, $brute$ and $milp$, respectively. A dash indicates that algorithms have not been run for the given number of images. Data obtained for the Lagos area.

	incremental						random_subsets					
15	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00
16	0.00	0.00	0.00	0.00	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00
17	0.00	0.00	0.00	0.00	0.03	0.01	0.01	0.02	0.00	0.00	0.01	0.00
18	0.00	0.00	0.00	0.00	0.05	0.01	0.02	0.03	0.00	0.01	0.01	0.00
19	0.00	0.00	0.00	0.00	0.08	0.01	0.03	0.05	0.00	0.01	0.02	0.00
20	0.00	0.00	0.00	0.00	0.16	0.01	0.06	0.08	0.00	0.01	0.03	0.00
21	0.00	0.00	0.00	0.00	0.20	0.01	0.07	0.10	0.00	0.03	0.07	0.00
22	0.00	0.00	0.00	0.00	0.26	0.01	0.11	0.13	0.01	0.02	0.13	0.01
23	0.00	0.01	0.00	0.00	0.52	0.01	0.16	0.17	0.00	0.03	0.25	0.01
24	0.00	0.02	0.00	0.00	1.02	0.01	0.36	0.22	0.01	0.06	0.52	0.01
25	0.00	0.03	0.00	0.00	2.03	0.01	0.55	0.30	0.01	0.06	0.92	0.01
26	0.00	0.03	0.00	0.00	–	0.01	0.74	0.61	0.01	0.07	–	0.01
27	0.01	0.08	0.00	0.00	–	0.01	1.48	0.94	0.02	0.13	–	0.01
28	0.01	0.11	0.00	0.00	–	0.01	1.53	0.92	0.03	0.19	–	0.01
29	0.02	0.15	0.00	0.01	–	0.01	2.58	1.06	0.04	0.16	–	0.01
30	0.02	0.20	0.00	0.01	–	0.01	4.40	2.31	0.04	0.22	–	0.01
32	0.03	0.25	0.00	0.01	–	0.01	6.85	2.91	0.10	0.43	–	0.01
34	0.10	0.82	0.01	0.02	–	0.01	17.73	6.63	0.23	0.53	–	0.01
36	0.16	1.20	0.01	0.02	–	0.01	33.53	14.12	0.23	0.67	–	0.01
38	0.23	1.64	0.02	0.05	–	0.01	–	–	0.40	1.26	–	0.01
40	0.53	3.42	0.02	0.06	–	0.01	–	–	0.59	2.31	–	0.01
42	0.66	4.14	0.04	0.09	–	0.01	–	–	1.23	4.25	–	0.01
44	1.26	7.71	0.05	0.12	–	0.01	–	–	2.82	8.00	–	0.01
46	1.84	11.25	0.09	0.18	–	0.01	–	–	3.37	10.59	–	0.01
48	2.42	14.06	0.14	0.27	–	0.01	–	–	5.49	9.30	–	0.01
50	5.02	27.16	0.17	0.34	–	0.01	–	–	6.50	15.88	–	0.01
52	7.28	38.32	0.23	0.47	–	0.02	–	–	–	–	–	–
54	12.74	56.96	0.29	0.61	–	0.02	–	–	–	–	–	–
56	16.49	70.30	0.47	0.83	–	0.02	–	–	–	–	–	–
58	18.45	77.77	0.59	1.08	–	0.02	–	–	–	–	–	–
60	35.48	127.57	0.75	1.57	–	0.02	–	–	–	–	–	–
62	–	–	1.10	1.81	–	0.02	–	–	–	–	–	–
64	–	–	1.63	2.67	–	0.02	–	–	–	–	–	–
66	–	–	1.82	3.34	–	0.02	–	–	–	–	–	–
68	–	–	7.11	7.81	–	0.02	–	–	–	–	–	–
70	–	–	2.16	0.35	–	0.02	–	–	–	–	–	–
72	–	–	2.44	0.43	–	0.02	–	–	–	–	–	–
74	–	–	4.45	0.85	–	0.02	–	–	–	–	–	–
76	–	–	8.21	1.50	–	0.02	–	–	–	–	–	–
78	–	–	9.04	1.68	–	0.02	–	–	–	–	–	–
80	–	–	15.35	2.47	–	0.02	–	–	–	–	–	–
82	–	–	–	2.97	–	0.02	–	–	–	–	–	–
84	–	–	–	4.96	–	0.02	–	–	–	–	–	–
86	–	–	–	6.17	–	0.02	–	–	–	–	–	–
88	–	–	–	9.17	–	0.02	–	–	–	–	–	–
90	–	–	–	11.14	–	0.02	–	–	–	–	–	–

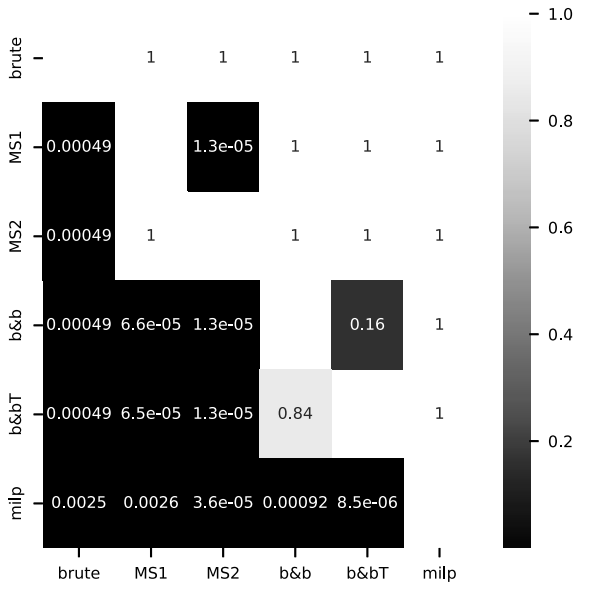

Fig. 8. P-values from one-sided Wilcoxon signed-rank tests are presented in a heatmap format. Each cell contains the p-value resulting from testing whether the algorithm in the corresponding row performs significantly faster than the algorithm in the respective column. P-values less than 0.05, indicating statistical significance, are highlighted with a black background.

(i) The $milp$ algorithm demonstrates significantly superior performance compared with all other algorithms.

(ii) The $brute$ algorithm performs significantly worse than all other algorithms.

(iii) Algorithm $MS1$ outperforms $MS2$.

(iv) Both branch and bound algorithms are superior to $MS1$, $MS2$, and $brute$.

However, the comparative performance between the basic branch and bound algorithm $b\&b$ and its variant $b\&b$-T is inconclusive. For instance sizes up to 68, the simpler $b\&b$ algorithm is faster, whereas for instance sizes of 70 and above, the $b\&b$-T variant outperforms the basic $b\&b$. Consequently, our data does not allow us to definitively conclude whether $b\&b$ is better than $b\&b$-T or vice versa. Similar observations can also be done for results obtained for other areas.

Instances from incremental classes are easier than random subsets for several reasons, one of which is the size of the optimal result. Since the images are arranged in a specific order, the optimal result for smaller-to-medium instances of the incremental class typically consists of a very small number of images (see Fig. 9). This means that there are large images in the datasets that can alone cover a significant portion, if not all, of the AOI.

Table 5. Running times (in seconds) of the $milp$ algorithm. Integers in the first column denote the number of images, while the four values in other cells are obtained, respectively, for the Mexico City, Paris, Rio de Janeiro, and Tokyo Bay areas.

	incremental				random_subsets			
10	0.05	0.04	0.04	0.05	0.02	0.02	0.02	0.02
20	0.09	0.07	0.06	0.08	0.05	0.03	0.03	0.04
30	0.13	0.09	0.07	0.10	0.07	0.05	0.05	0.06
40	0.15	0.11	0.09	0.13	0.08	0.06	0.06	0.07
50	0.18	0.13	0.10	0.15	0.10	0.07	0.07	0.09
60	0.21	0.15	0.11	0.17	0.11	0.08	0.08	0.10
70	0.23	0.16	0.12	0.20	0.13	0.10	0.10	0.12
80	0.25	0.17	0.13	0.22	0.15	0.11	0.12	0.14
90	0.27	0.18	0.14	0.25	0.16	0.13	0.13	0.15
100	0.29	0.19	0.15	0.27	0.19	0.14	0.15	0.16
110	0.32	0.21	0.17	0.30	0.20	0.15	0.16	0.18
120	0.33	0.22	0.18	0.32	0.23	0.17	0.17	0.20
130	0.36	0.23	0.19	0.34	0.24	0.18	0.19	0.22
140	0.38	0.24	0.21	0.35	0.28	0.21	0.20	0.23
150	0.40	0.26	0.22	0.38	0.29	0.22	0.22	0.26
160	0.42	0.27	0.24	0.39	0.32	0.24	0.24	0.28
170	0.43	0.28	0.26	0.41	0.33	0.27	0.25	0.30
180	0.45	0.30	0.27	0.43	0.36	0.28	0.27	0.32
190	0.49	0.31	0.30	0.44	0.38	0.30	0.29	0.34
200	0.50	0.33	0.32	0.46	0.41	0.32	0.31	0.37
210	0.52	0.35	0.34	0.49	0.44	0.33	0.33	0.39
220	0.57	0.36	0.35	0.51	0.46	0.37	0.35	0.40
230	0.57	0.38	0.37	0.53	0.49	0.37	0.37	0.42
240	0.58	0.41	0.39	0.55	0.51	0.41	0.39	0.44
250	0.59	0.42	0.41	0.57	0.54	0.42	0.40	0.46
260	0.60	0.43	0.43	0.56	0.56	0.45	0.42	0.49
270	0.61	0.44	0.44	0.58	0.59	0.47	0.44	0.51
280	0.63	0.45	0.46	0.60	0.61	0.49	0.46	0.53
290	0.66	0.46	0.48	0.61	0.63	0.47	0.47	0.55
300	0.67	0.48	0.50	0.63	0.65	0.51	0.49	0.58

We now juxtapose the quality of results obtained for the incremental and random subsets classes for a varied number of images. Measurements were obtained for instance sizes ranging from 10 to 300 with a step of 10. Let us now consider Fig. 10. The dark solid line with triangular markers represents the optimal result value for the incremental class. The dotted line represents the mean result value for 25 independent random subsets; triangular markers represent the median of those runs; the range of vertical lines represents the standard deviation, and the filled area ranges from the minimum to the maximum of the results obtained. It can be noticed for the incremental class that the results form a non-increasing sequence, as larger instances are supersets of smaller ones. The optimal result value diminishes as new images are considered. For the Paris area, we can see three such turning points at sizes 20, 70, and 160. The last value is worth attention because it means that selecting the 150 best images with

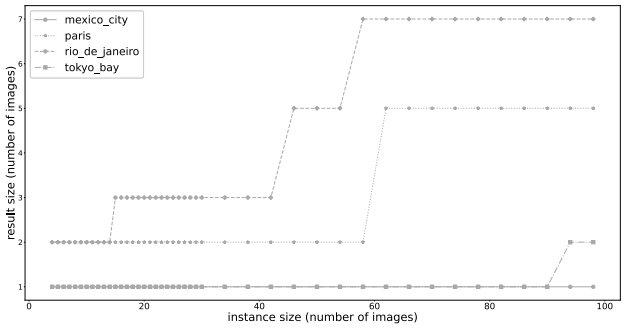

Fig. 9. The sizes of the optimal results found for instances in the incremental class. The hitting-set graph was constructed using 1600 sampling points. The number of images in the optimal solution increases as the instance size grows.

a greedy heuristic is not enough to guarantee that the optimal solution will be formed by some subset of those images. This shows, in some sense, how far the simple greedy heuristic can be from optimum.

On the other hand, the optimum result for the subset of 20 best images selected by the heuristic was no worse by no more than 50% than the optimum for the whole set. An interesting fact is that the best (minimum value) results for random subsets class on small ($\leq$ 30 images) instances are equally good or even smaller than for subsets of images chosen by the greedy heuristic. A sizeable gap can be noticed for the Tokyo Bay area (Fig. 10), where the best of the solutions found for randomly selected subsets (or even the mean, starting from the size of 60) is much better than in the incremental case for sizes not exceeding 150. This is not the case for the Paris area, where the results for the incremental class are comparable to the best results obtained for random subsets of the same size. The distribution of results obtained for the random subsets class has been additionally depicted in more detail on Fig. 11. The result characteristics for other areas are comparable to those in either of the two.

6. Discussion

The use of satellite imagery has gained popularity for monitoring and responding to changes in different regions of the Earth. However, constructing a mosaic of high-resolution images to cover the flexible Area of Interest often requires retrieving images from multiple providers with varying costs and cloud coverage. The main objective of this study is to achieve the optimal multi-objective selection for the SIMS problem.

The SIMS problem exhibits parallels with the challenges encountered in aerial photo analysis, particularly regarding the optimal coverage of an Area of Interest (AOI) using a subset of images. However, several distinct nuances differentiate these two contexts.

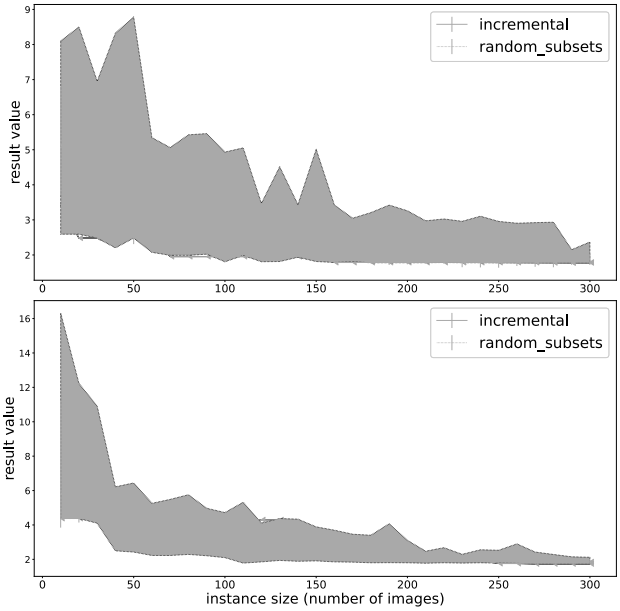

Fig. 10. Optimal results for incremental and random subsets class for the Paris area (top) and Tokyo Bay area (bottom).

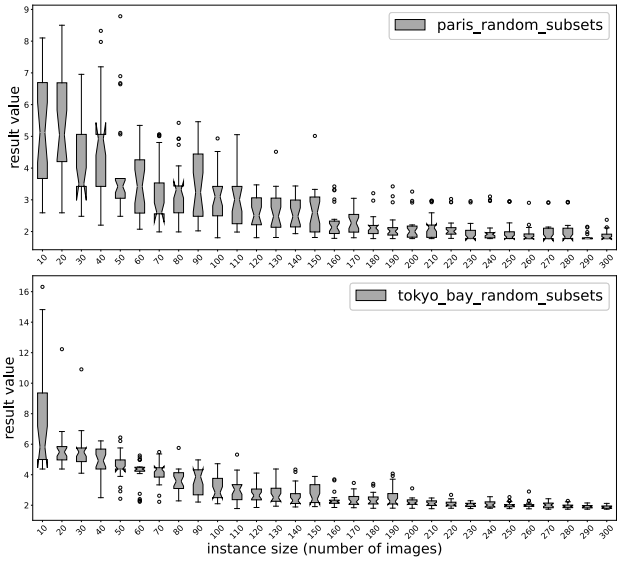

Fig. 11. Distribution of results for random subsets class for the Paris area (top) and Tokyo Bay area (bottom).

In aerial photography, images are typically captured using Unmanned Aerial Vehicles (UAVs) following optimized flight paths to ensure optimal overlap (Chen *et al.*, 2024). Conversely, in the SIMS context, the overlap is predetermined and cannot be optimized. Additionally, aerial photos are taken from lower altitudes, necessitating specific corrections for tilt angles, which are negligible in satellite imaging (James and Robson, 2014). Despite these differences, both satellite and aerial imaging

technologies share fundamental similarities but are tailored to different applications, each presenting unique challenges.

At first glance, the Internet Shopping Optimization Problem (Błazewicz *et al*., 2010), in which one must make an optimal selection of products from online stores, has some similarities (at a higher level of abstraction) with the challenging problem described in the present paper. However, digging deeper into the topic reveals too many differences to make simple model or algorithm transformations. Therefore, we developed new models and implemented innovative algorithms to address this problem. We then performed a comprehensive evaluation to confirm their high quality and competitive efficiency compared to existing state-of-the-art methods.

By using the sampling-point-based method, we were able to easily and efficiently create the necessary data structure for building the hitting-set graph. Our experiments also indicate that a relatively small number of sampling points is sufficient to obtain an optimal or near-optimal result for the corresponding graph, compared to a more complex approach involving plane division.

All designed and implemented algorithms produce optimal results to the problem (and the same one, if it is unique). From a practical point of view, selection of the best algorithm can be based purely on computational time and resource requirements. The algorithm $milp$, based on the ILP formulation, outperforms others by a significant margin in terms of efficiency. However, it should be noted that the other approaches can be easily adapted to a more general and challenging problem model, where not all elements of the set B need to be covered, but rather a fixed fraction of them. In such cases, the number of variables used in the ILP formulation increases significantly (from $O(n)$ to $O(n^2)$ in general). It would be interesting to see how the $milp$ approach handles this challenge and how the presented algorithms compare with each other. This is a plan for future work, along with conducting additional experiments to compare the sampling point method with the plane-divide approach. Finally, we intend to examine the impact of cloud coverage on the performance of the designed algorithms and focus on heuristic approaches, which will likely be necessary to find in acceptable time solutions to the more complex variations of the SIMS problem. We believe that these proposed algorithms will drive further development in the space data market.

Acknowledgment

Research is partially supported by the Luxembourg National Research Fund (FNR) and the National Centre for Research and Development (NCBR) under the SERENITY Project (ref. C22/IS/17395419; POLLUX-XI/15/Serenity/2023).

References

Airbus (2023a). Pléiades, https://www.intelligence-airbusds.com/en/8692-pleiades.

Airbus (2023b). Pléiades neo, https://www.airbus.com/en/products-services/space/earth-observation/earth-observation-portfolio/pleiades-neo.

Airbus (2023c). SPOT, https://www.intelligence-airbusds.com/en/8693-spot-67.

Asner, G.P., Powell, G.V.N., Mascaro, J., Knapp, D.E., Clark, J.K., Jacobson, J., Kennedy-Bownoten, T., Balaji, A., Paez-Acosta, G., Victoria, E., Secada, L., Valqui, M. and Hughes, R.F. (2010). High-resolution forest carbon stocks and emissions in the Amazon, *Proceedings of the National Academy of Sciences* **107**(38): 16738–16742, DOI: 10.1073/pnas.1004875107.

Bennett, M.M. and Smith, L.C. (2017). Advances in using multitemporal night-time lights satellite imagery to detect, estimate, and monitor socioeconomic dynamics, *Remote Sensing of Environment* **192**: 176–197, DOI: 10.1016/j.rse.2017.01.005.

Błazewicz, J., Kovalyov, M., Musiał, J., Urbański, A. and Wojciechowski, A. (2010). Internet shopping optimization problem, *International Journal of Applied Mathematics and Computer Science* **20**(2): 385–390, DOI: 10.2478/v10006-010-0028-0.

Burke, M., Driscoll, A., Lobell, D.B. and Ermon, S. (2021). Using satellite imagery to understand and promote sustainable development, *Science* **371**(6535): eabe8628, DOI: 10.1126/science.abe8628.

Chen, K., Luo, W., Lin, X., Song, Z. and Chang, Y. (2024). Evolutionary Biparty Multiobjective UAV Path Planning: Problems and Empirical Comparisons, *IEEE Transactions on Emerging Topics in Computational Intelligence* **8**(3): 2433–2445, DOI: 10.1109/TETCI.2024.3361755.

Combarro Simón, M., Talbot, P., Danoy, G., Musial, J., Alswaitti, M. and Bouvry, P. (2023). Constraint Model for the Satellite Image Mosaic Selection Problem, *LIPIcs, Volume 280, CP 2023* **280**: 44:1–44:15, DOI: 10.4230/LIPICS.CP.2023.44.

Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M. and Saurabh, S. (2015). *Parameterized Algorithms*, Springer, Cham, DOI: 10.1007/978-3-319-21275-3.

Felegari, S., Sharifi, A., Moravej, K., Amin, M., Golchin, A., Muzirafuti, A., Tariq, A. and Zhao, N. (2021). Integration of Sentinel 1 and Sentinel 2 Satellite Images for Crop Mapping, *Applied Sciences* **11**(21): 10104, DOI: 10.3390/app112110104.

Flood, N., Watson, F. and Collett, L. (2019). Using a U-net convolutional neural network to map woody vegetation extent from high resolution satellite imagery across Queensland, Australia, *International Journal of Applied Earth Observation and Geoinformation* **82**: 101897, DOI: 10.1016/j.jag.2019.101897.

Goetz, S. (2007). Crisis in Earth Observation, *Science* **315**(5820): 1767–1767, DOI: 10.1126/science.1142466.

Hall, K., Reitalu, T., Sykes, M.T. and Prentice, H.C. (2012). Spectral heterogeneity of QuickBird satellite data is related to fine-scale plant species spatial turnover in semi-natural grasslands, *Applied Vegetation Science* **15**(1): 145–157, DOI: 10.1111/j.1654-109X.2011.01143.x.

Hansen, M.C., Stehman, S.V., Potapov, P.V., Loveland, T.R., Townshend, J. R.G., DeFries, R.S., Pittman, K.W., Arunarwati, B., Stolle, F., Steininger, M.K., Carroll, M. and DiMiceli, C. (2008). Humid tropical forest clearing from 2000 to 2005 quantified by using multitemporal and multiresolution remotely sensed data, *Proceedings of the National Academy of Sciences* **105**(27): 9439–9444, DOI: 10.1073/pnas.0804042105.

Henderson, J.V., Storeygard, A. and Weil, D.N. (2012). Measuring Economic Growth from Outer Space, *American Economic Review* **102**(2): 994–1028, DOI: 10.1257/aer.102.2.994.

Huangfu, Q. and Hall, J.A.J. (2018). Parallelizing the dual revised simplex method, *Mathematical Programming Computation* **10**(1): 119–142, DOI: 10.1007/s12532-017-0130-5.

James, M. R. and Robson, S. (2014). Mitigating systematic error in topographic models derived from UAV and ground-based image networks, *Earth Surface Processes and Landforms* **39**(10): 1413–1420, DOI: 10.1002/esp.3609.

Jean, N., Burke, M., Xie, M., Davis, W.M., Lobell, D.B. and Ermon, S. (2016). Combining satellite imagery and machine learning to predict poverty, *Science* **353**(6301): 790–794, DOI: 10.1126/science.aaf7894.

Lopez-Loces, M.C., Musial, J., Pecero, J.E., Fraire-Huacuja, H.J., Blazewicz, J. and Bouvry, P. (2016). Exact and heuristic approaches to solve the Internet shopping optimization problem with delivery costs, *International Journal of Applied Mathematics and Computer Science* **26**(2): 391–406, DOI: 10.1515/amcs-2016-0028.

Müllerová, J., Brůna, J., Bartaloš, T., Dvořák, P., Vítková, M. and Pyšek, P. (2017). Timing Is Important: Unmanned Aircraft vs. Satellite Imagery in Plant Invasion Monitoring, *Frontiers in Plant Science* **8**: 887, DOI: 10.3389/fpls.2017.00887.

Rovetto, R.J. (2017). An ontology for satellite databases, *Earth Science Informatics* **10**(4): 417–427, DOI: 10.1007/s12145-017-0290-x.

Sánchez-Azofeifa, A., Rivard, B., Wright, J., Feng, J.-L., Li, P., Chong, M.M. and Bohlman, S.A. (2011). Estimation of the Distribution of Tabebuia guayacan (Bignoniaceae) Using High-Resolution Remote Sensing Imagery, *Sensors* **11**(4): 3831–3851, DOI: 10.3390/s110403831.

Shean, D.E., Alexandrov, O., Moratto, Z.M., Smith, B.E., Joughin, I.R., Porter, C. and Morin, P. (2016). An automated, open-source pipeline for mass production of digital elevation models (DEMs) from very-high-resolution commercial stereo satellite imagery, *ISPRS Journal of Photogrammetry and Remote Sensing* **116**: 101–117, DOI: 10.1016/j.isprsjprs.2016.03.012.

Tian, H., Pei, J., Huang, J., Li, X., Wang, J., Zhou, B., Qin, Y. and Wang, L. (2020). Garlic and Winter Wheat Identification Based on Active and Passive Satellite Imagery and the Google Earth Engine in Northern China, *Remote Sensing* **12**(21): 3539, DOI: 10.3390/rs12213539.

Verpoorter, C., Kutser, T., Seekell, D.A. and Tranvik, L.J. (2014). A global inventory of lakes based on high-resolution satellite imagery, *Geophysical Research Letters* **41**(18): 6396–6402, DOI: 10.1002/2014GL060641.

Virtanen, P. *et al.* (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python, *Nature Methods* **17**(3): 261–272, DOI: 10.1038/s41592-019-0686-2.

Wang, Y., Zhang, D. and Dai, G. (2020). Classification of high resolution satellite images using improved U–Net, *International Journal of Applied Mathematics and Computer Science* **30**(3): 399–413, DOI: 10.34768/amcs-2020-0030.

Yeh, C., Perez, A., Driscoll, A., Azzari, G., Tang, Z., Lobell, D., Ermon, S. and Burke, M. (2020). Using publicly available satellite imagery and deep learning to understand economic well-being in Africa, *Nature Communications* **11**(1): 2583, DOI: 10.1038/s41467-020-16185-w.

Zok, T., Badura, J., Swat, S., Figurski, K., Popenda, M. and Antczak, M. (2020). New models and algorithms for RNA pseudoknot order assignment, *International Journal of Applied Mathematics and Computer Science* **30**(2): 315–324, DOI: 10.34768/amcs-2020-0024.

Sylwester Swat is a researcher at the Poznan University of Technology, where he also did PhD studies. His research interests lie in the field of combinatorial optimization, algorithm design and analysis, HPC and software engineering. In his work he puts particular stress on the design of heuristic approaches to complex real-world problems, usually related to big data analysis. He is a winner of the START 2022 stipend for young scientists and the Barbara Skarga Research Stipend.

Tomasz Zok is an associate professor at the Institute of Computing Science, Poznan University of Technology. His research expertise encompasses computational methods for analyzing biomolecular structures and dynamics, operations research, as well as high-throughput and high-performance computing. He has received numerous accolades in national contests and holds PhD (2018) and DSc (2022) degrees in computing science.

Maciej Antczak is an associate professor at the Poznan University of Technology and IBCh PAS. His research interests include RNA structure, structural bioinformatics, algorithmics, combinatorial optimization, and artificial intelligence. He is the author of highly-cited papers in leading scientific journals on computing and life sciences as well as widely used tools/web servers. He holds PhD (2013) and DSc (2019) degrees in computing science.

Jacek Blazewicz is a full professor at the Poznan University of Technology. His research interests include algorithm design, computational complexity, scheduling, combinatorial optimization, bioinformatics. He has PhD and DSc degrees in computer science. His publication record includes over 430 papers in outstanding journals. He also co-authored over ten monographs. His Web of Science citation index is over 8100, and his h-index is 41. He is an IEEE fellow.

Jedrzej Musial is an associate professor at the Poznan University of Technology. His research interests include e-commerce, Internet shopping, cloud brokering, algorithms, applications of combinatorial optimization, and operations research. He has a PhD and habilitation (DSc) in computer science from the Poznan University of Technology and a PhD from the University of Luxembourg. He has published 41 papers and 2 monographs.

Received: 26 June 2024
Revised: 2 October 2024
Accepted: 15 November 2024