Multi-Covering a Point Set with m Disks of Minimum Total Area in the Presence of Obstacles

Chek-Manh Loi[1], Mariem Guitouni[2], Michael Perk[1], Aaron T. Becker[2], Sándor P. Fekete[1]

***Abstract*—A central challenge in robotic sensing is placing sensors to monitor a set of assets with redundancy guarantees while minimizing the overall sensor footprint. The General Multi-Coverage (GMC) addresses this by covering each asset by a specified number of sensors, such that the total energy consumption (i.e., the sum of all individual sensor areas) is minimized. While prior work has analyzed heuristics and exact methods for the unconstrained GMC, practical drone sensing deployments face additional challenges: no-fly zones ("Do Not Enter") restrict where drones can be positioned, while prohibited areas ("Do Not Cover") limit the extent of sensing regions. In this work, we study the two corresponding obstacle-constrained extensions of the GMC. Both turn out to be quite difficult: We formally prove that both the *Do Not Enter* GMC (DNE-GMC) and the *Do Not Cover* GMC (DNC-GMC) are NP-hard.**

For both problem variants, we present heuristics for generating feasible placements and develop exact methods for computing provably optimal solutions. For the DNE-GMC, our heuristics yield high-quality solutions that scale effectively, while our exact methods solve instances with up to 300 points to provable optimality. For the DNC-GMC, we show that even *finding* feasible solutions (regardless of their objective value) is computationally challenging: heuristic approaches often fail to produce solutions reliably within reasonable time. In contrast, our exact methods scale well, achieving optimality for instances with up to 9000 points in mere seconds.

***Index Terms*—Computational Geometry, Aerial Systems: Applications, Optimization, and Optimal Control.**

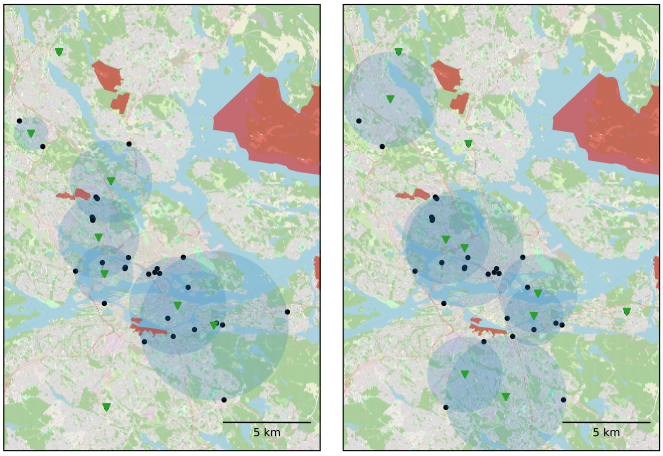


Fig. 1: Map of Stockholm, a simplified instance of a benchmark described in Section VII-B, illustrating both the Do Not Enter GMC (left) and the Do Not Cover GMC (right). Black points represent 30 ATM assets, with a random coverage requirement of 1 or 2 (omitted for readability). These are to be covered by $m = 10$ drones (depicted as green triangles), with their respective coverage regions shown as blue disks. Red regions indicate 14 exclusion zones. (Left) A solution for the DNE-GMC, which achieves a cost of approximately $129.95\,\mathrm{km}^2$; (right) A solution for the DNC-GMC, with a cost of approximately $158.63\,\mathrm{km}^2$. Each figure shows an optimal solution, computed with the help of an integer program.

I. Introduction

Autonomous aerial systems such as drones are increasingly used for surveillance, delivery, inspection, and disaster response. For most of these applications, reliable coverage is critical, motivating multi-coverage, in which each asset is covered with a redundancy rate that corresponds to its importance. At the same time, it is desirable to minimize the overall power consumption, i.e., the sum of all sensor areas. These goals are formalized by the GMC problem [1]: Given a set $P = \{p_1, \ldots, p_n\}$ of n assets that need to be covered, each with multiplicity κ_i, place m disks (representing drone sensing regions), such that each asset p_i is covered κ_i times, while the area of the union of covering disks is minimized.

We study two realistic extensions of the GMC problem, providing heuristic and exact methods for both variants. The first, called *Do Not Enter GMC* (DNE-GMC), imposes exclusion zones for the disk centers. The second, called *Do Not Cover GMC* (DNC-GMC), prohibits disks from reaching into forbidden regions. Figure 1 illustrates a drone-based multi-coverage instance over Stockholm, highlighting how the constraints impact the placement of drones and the cost of the solution. Figure 2 shows optimal solutions for the GMC problem and its two extensions for $n = 8$ assets, $m = 4$ sensors, and $\kappa \in \{1, 2\}$.

In practice, the output of our solver—a set of disk centers and radii—can be interpreted as high-level sensing setpoints for a robot team. For aerial robots, a disk corresponds to the ground footprint of a downward-looking sensor at a chosen altitude or transmit power; DNE-GMC obstacles correspond to regions where waypoint setpoints are disallowed (regulatory no-fly or safety zones [2]), and DNC-GMC obstacles correspond to regions where the sensing footprint must not overlap (privacy zones, interference, or restricted sensing). A standard deployment pipeline is to (i) compute a static plan using our method from a map layer, (ii) track these setpoints with a low-level controller and onboard obstacle avoidance, and (iii) replan periodically (receding horizon) as assets/obstacles or state estimates change.

In the following, we present heuristic and exact methods that scale well for the DNE-GMC, solving instances with up to 300 points to provable optimality, which is comparable to prior work on the GMC [1]. For the DNC-GMC, we show that even *finding* feasible solutions is NP-hard and practically

[1]{loi,perk}@ibr.cs.tu-bs.de, s.fekete@tu-bs.de; partially supported by DFG grant FE407/21-2.

[2]mguitoun@CougarNet.uh.edu, atbecker@uh.edu; partially supported by NSF IIS-2130793, ARL W911NF-23-2-0014, and DAF AFX23D-TCSO1.

challenging. To address these difficulties, we develop an exact method based on integer programming that outperforms heuristic approaches. Remarkably, this allows us to handle instances with up to 9000 points in seconds.

The remainder of this paper is organized as follows. Section II reviews related work, Section III shows our hardness results, while Section IV presents a discretization method that reduces the search space to a finite set. Using this discretization, we define two heuristics for generating feasible solutions in Section V, and describe exact methods based on Integer Programming (IP) in Section VI. We evaluate our methods on benchmark instances in Section VII.

Preliminaries

In the GMC problem, we are given a set P of n points, a number of sensors m, and a coverage function $\kappa : P \to \mathbb{N}$, the goal is to assign a center y_i and radius r_i for each disk d_i with $i \in \{1, \ldots, m\}$ so that each $p \in P$ is covered by at least $\kappa(p)$ disks while minimizing the total area of the disks, i.e., $\pi \cdot \sum_i r_i^2$. We define the disk $d_i \subset \mathbb{R}^2$ with center y_i and radius r_i and write $p_j \in d_i$ if the point p_j is covered by the disk d_i, i.e., if $\|p_j - y_i\|_2 \leq r_i$. Additionally, we are given a set of obstacles $\mathcal{O}$. An obstacle $o_i \in \mathcal{O}$ can be a disk or a (possibly non-simple) polygon. W.l.o.g. we assume that the obstacles are disjoint.

The DNE-GMC asks for an area-optimal disk cover such that disk centers do not intersect any obstacles, i.e., $\forall i \in \{1, \ldots, m\}, \forall o_j \in \mathcal{O} : y_i \notin o_j$. The DNC-GMC requires an area-optimal disk cover such that no disk intersects any obstacle, i.e. $\forall i \in \{1, \ldots, m\}, \forall o_j \in \mathcal{O} : d_i \cap o_j = \emptyset$. As every instance that has a point $p_i \in P$ inside an obstacle is automatically infeasible, we assume that all points are outside of all obstacles for the DNC-GMC.

Uncertainty-aware safety margins: Real systems face uncertainty in robot state (localization), asset location, and sensing/detection. Deterministic margins incorporate such uncertainty without changing the combinatorial structure of our methods. If robot position error is bounded by δ_y and asset position error is bounded by δ_p, then enforcing coverage with an expanded radius $r_i' = r_i + \delta_y + \delta_p$ yields a conservative guarantee. Similarly, obstacle avoidance can be made robust by inflating obstacles by a margin δ_o (taking the Minkowski sum with a disk of radius δ_o) before solving: for DNE-GMC this captures waypoint safety under localization error, while for DNC-GMC it ensures the *entire* footprint remains outside restricted regions despite pose uncertainty. These deterministic margins can be viewed as a conservative surrogate for chance constraints when uncertainty is approximately Gaussian.

II. Related Work

Table I compares our work to solutions in the literature. Most closely related are the static and mobile drone location problems of Zorbas et al. [3], [4], who discretize the search space to a finite set of candidate disk centers and solve the resulting model, yielding almost optimal solutions for instances with up to 10 points. Later, Zorbas and Douligeris [5] use a geometric discretization to reduce the search space to a finite set and solve the static problem optimally for up to 50 points. Using the same observations with modern integer program solvers [6], Guitouni et al. [1] solve the General Multi-Coverage (GMC) problem with multiplicity constraints to provable optimality for up to 300 points.

These optimal static solutions can serve as baselines for heuristics that scale to larger instances. In mobile variants, the time horizon is often discretized into time steps [7]–[9]; thus, one can use our optimal initial solution and, at each time step, compute an optimal static solution as a provable lower bound, as implemented in [10]. Such solutions can also serve as lower bounds for more complex problems such as wireless power transfer with additional assignment constraints [11].

The DNE-GMC has been theoretically considered by Hamacher and Schöbel [12], who study optimal displacement of disks in the presence of obstacles. There is also theoretical work on obstacles that block sensing regions [13]. To the best of our knowledge, we are the first to consider the DNC-GMC problem and the first to provide practical algorithms for both variants. Therefore, we can only compare ourselves to methods for the unconstrained GMC [1].

Our discretization abstracts from complex real-world cost functions and applies whenever the objective is monotone in the disk area; in practice, increasing coverage can be achieved by increasing altitude or transmission power. More realistic cost models for UAV-mounted base stations (UAV-BSs), such as those derived by Mozaffari et al. [14], integrate seamlessly with our scheme. Their results also show an optimal altitude maximizing coverage area, which we can enforce by removing candidate disks that exceed this altitude; similarly, a minimum required radius can be enforced by inflating small disks.

The same discretization enables polynomial-time optimal solutions for single UAV-BS placement [15] and allows minimizing the number of UAV-BSs needed to cover a set of points given a maximum bound on the radius [16].

A more restricted variant of the problem focuses on assigning radii to given set of sensor locations [17]. Recently, Huang et al. [18] propose a PTAS for a variant of this problem including multi coverage constraints.

Recent works study environmental constraints in target monitoring. Saeed et al. [20] introduce Argus, a visual coverage system for camera-equipped drones that addresses occlusions due to the environment and other targets. Penin et al. [21] present an online trajectory algorithm for dynamic target tracking that incorporates vision constraints, obstacle avoidance, and occlusion management. Other research investigates region (area) coverage with obstacles using one [22], [23] or multiple [24] UAVs, as opposed to discrete point coverage. Another line of work examines dangers arising from tracked targets: Li et al. [19] propose reactive replanning and recovery from failures induced by sensing and communication danger zones.

Our work extend the static GMC by introducing obstacle constraints that limit both drone placement (Do Not Enter) and sensing coverage (Do Not Cover) while maintaining multi-coverage requirements with provable optimality through Integer Programming formulations.

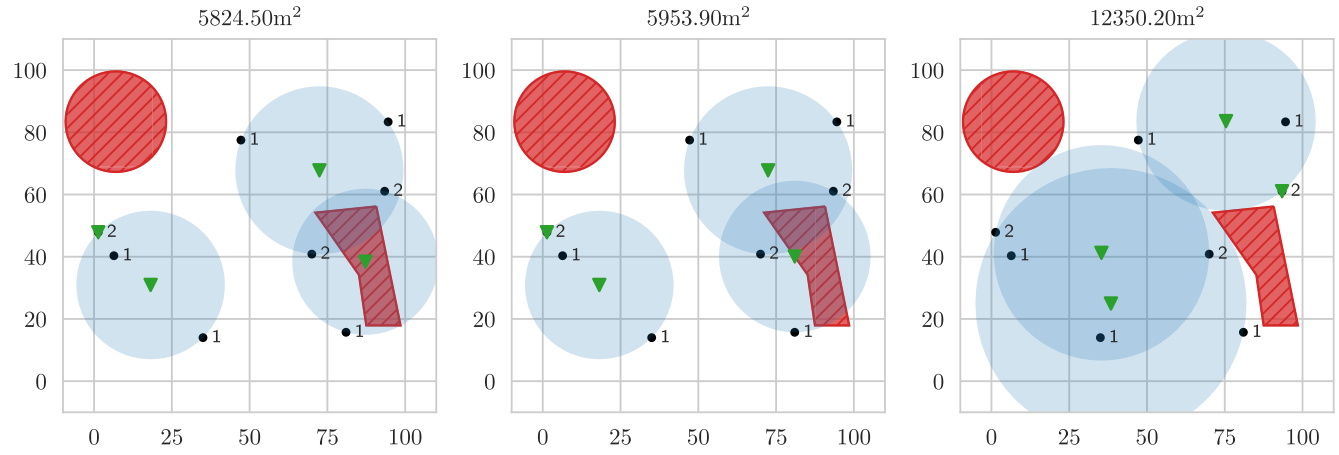


Fig. 2: Three optimal solutions for the (left) GMC, (center) DNE-GMC, and (right) DNC-GMC. The disks are centered at the green triangles, the assets are annotated by their coverage requirement $\kappa(p)$ and the obstacles are shown in red. The cost of the solutions increases from left to right.

TABLE I: Summary of multi-coverage / drone scheduling approaches and reported problem sizes.

Ref	Approach Name	Type	Obstacles	Soln	Max Tgt	Max Drn
Current	DNC-GMC IP (Do Not Cover GMC IP)	Static	Yes	Optimal	9000	3000
Current	DNC-GMC Heuristic (Genetic Algorithm)	Static	Yes	Heuristic	9000	3000
Current	DNE-GMC IP (Do Not Enter GMC IP)	Static	Yes	Optimal	300	100
Current	DNE-GMC Heuristic (Greedy)	Static	Yes	Heuristic	300	100
[1]	DGMC IP (Dispersive Multi-Coverage with separation IP)	Static	No	Optimal	300	30
[1]	GMC IP (General Multi-Coverage Integer Programming)	Static	No	Optimal	250	100
[1]	GMC Heuristic (Iterative k-means based)	Static	No	Heuristic	250	100
[3]	CAS (Centralized Altitude Scheduler)	Dynamic	No	Heuristic	175	14
[3]	LAS (Localized Altitude Scheduler)	Dynamic	No	Heuristic	175	11
[3]	LAS-sec (LAS with Smallest Enclosing Circle)	Dynamic	No	Heuristic	175	11
[10]	Centralized Algorithm	Dynamic	No	Optimal	90	24
[10]	Decentralized Algorithm	Dynamic	No	Heuristic	90	24
[7]	AFW-DHOA (Adaptive Fitness Wind angle-based Deer Hunting)	Static	No	Heuristic	50	250
[4]	LAS (Localized Altitude Scheduler)	Dynamic	No	Heuristic	50	17
[4]	L-MDLP (Localized Mobile Drone Location Problem)	Dynamic	No	Heuristic	50	16
[4]	C-MDLP (Centralized Mobile Drone Location Problem)	Dynamic	No	Heuristic	50	15
[5]	DPH (Drone Positioning Heuristic)	Static	No	Heuristic	50	14
[5]	ILP (Integer Linear Programming)	Static	No	Optimal	50	12
[11]	MILP (Mixed Integer Linear Program)	Static	No	Optimal	50	10
[11]	Greedy Heuristic	Static	No	Heuristic	50	10
[8]	M1 (Minimize #UAVs – identical replacements)	Dynamic	No	Optimal	30	48
[8]	M2 (Minimize #UAVs – different types)	Dynamic	No	Optimal	30	30
[8]	Mc (Minimize activation cost)	Dynamic	No	Optimal	30	28
[4]	Optimal Mobile Drone Location Problem	Dynamic	No	Optimal	10	6
[19]	Resilient Coordination Framework	Dynamic	Yes	Heuristic	4	2

III. NP-Hardness

In this section, we prove that the optimization versions of both problem variants, DNE-GMC and DNC-GMC, are NP-hard. For a proof of the former, we can build on work by Alt et al. [17], who proved that it is NP-hard to optimally cover a set of points with disks centers chosen from a given set of candidate locations. We can simulate this restriction in the DNE-GMC by placing obstacles that prevent disks from being placed anywhere else, e.g., by placing a single obstacle with holes at the candidate locations, obtaining the following result.

Corollary 1. *For any fixed $\alpha > 1$, let the cost of a disk of radius r be $f(r) = r^\alpha$. Then it is NP-hard to decide for a given set P of n points, a number $m \in \mathbb{N}$, a cost bound $c \in \mathbb{R}$, and a single non-simple obstacle o, whether there exists m disks centered outside of o such that every point $p \in P$ is covered by at least one disk and the total cost is at most c.*

Then DNE-GMC is a special case of Corollary 1 with $\kappa : P \to \{1\}$ and $\alpha = 2$.

Corollary 2. *Given $c \in \mathbb{R}$, deciding whether there exists a DNE-GMC solution of cost at most c is NP-hard.*

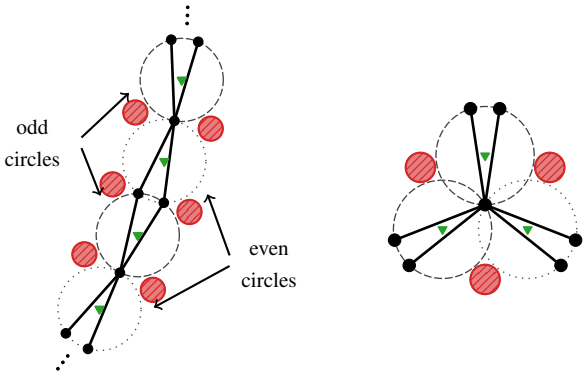
Although the *optimization* version is challenging for the DNE-GMC, *feasibility* is trivial: any instance admits a feasible placement with $m \geq \kappa_{max}$ disks by locating the disks in non-constrained regions and choosing radii large enough to cover

all points, with $\kappa_{max} := \max_{p \in P} \kappa(p)$.

In contrast, for the DNC-GMC, even *finding* a feasible solution is challenging. In particular, the presence of obstacles introduces non-convex constraints on the feasible region for disk placement, fundamentally altering the structure of the search space and preventing the direct application of standard covering techniques. In fact, we prove that even deciding whether there exists a feasible placement for the DNC-GMC is NP-hard.

Theorem 1. *Given a set P of n points, a number $m \in \mathbb{N}$ and a set of obstacles $\mathcal{O}$, it is NP-hard to decide whether there exists a set of m disks, such that every point $p \in P$ is covered by at least one disk and no disk intersects any obstacle.*

Proof. We reduce from Planar 3SAT [25]. Let I be an instance and let G_I be its planar variable-clause incidence graph. We use the variable gadget of [17] to represent the two truth assignments of each variable, see Figure 3a. This allows two different ways of covering the points (using the "odd" or the "even" circles), representing the two truth assignments of the variable. The obstacles are positioned so that disks can only be placed at these intended locations. For each edge in G_I, we attach a similar chain of points that connects the variable gadget to the clause gadget. The parity of the variable loop assigns a parity to all incident chains.



(a) Variable gadget [17]. (b) Clause gadget.

Fig. 3: NP-hardness proof gadgets with obstacles.

Figure 3b depicts the clause gadget. It connects three incident variable chains at a common center point. The obstacle layout ensures that the clause center can be covered if and only if at least one of the three terminal disks corresponding to the literals of the clause is selected. We place only a polynomial number of obstacles to delimit the feasible regions tightly enough to forbid any unintended large disks and to enforce the above behavior. We follow the argument from [17] to conclude that there exists a cover, if and only if the truth assignment corresponding to the cover yields a satisfying assignment for the original instance. □

IV. Discretization Scheme

A key component of any algorithm that tackles both the DNE-GMC and DNC-GMC is a discretization scheme, as it allows reducing the continuous geometric question to one with only a finite candidate of possible solutions. In the case without obstacles, the answer is straightforward: the smallest enclosing disk of the points suffices. However, in the presence of obstacles, additional considerations are required to ensure feasibility and optimality. In this section, we discuss how to compute a discrete candidate set of disks for both variants.

A. Do Not Enter GMC

Consider a disk d that covers a subset of points $P' \subseteq P$. If the center of d is not intersecting any obstacle, then it is also optimal for the constrained version. If it is intersecting an obstacle, Hamacher and Schöbel [12] showed how to compute the smallest feasible disk.

Figure 4 shows an example of how to identify the smallest feasible disk for a given set of points. The dashed circle centered at c is the original disk that intersects the polygonal obstacle. The blue disks around the points are of equal size.

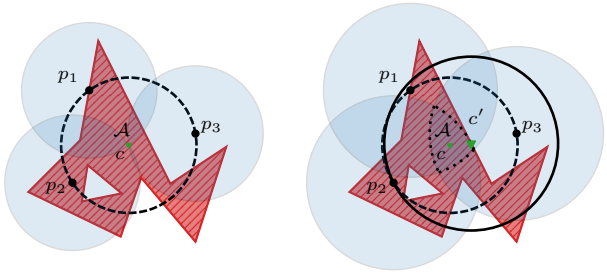


Fig. 4: Polygonal obstacle with holes shown in red. $\mathcal{A}$ is the intersection of the same size blue disks. The smallest disk covering the points is centered at c. (left) $\mathcal{A}$ is a single point. (right) The smallest feasible disk is obtained by growing the same size disks until $\mathcal{A}$ touches the polygon boundary. The projection of p_2 onto the intersected edge gives the new feasible center c'.

Geometrically, we increase the radius of the blue disks until the intersection $\mathcal{A}$ touches the polygon boundary. This yields the minimum translation of the original center c to a new feasible center c'.

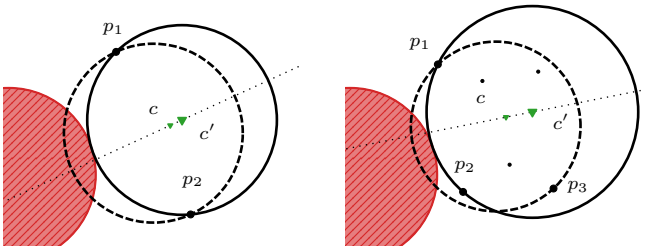
For a polygonal obstacle, it is easy to see that an optimal center disk is always a projection of a point on an edge, a concave vertex, or the intersection of a perpendicular bisector between two points and the polygon boundary. It is straightforward to generalize this approach to circular obstacles.

B. Do Not Cover GMC

Let $P' \subseteq P$ be a set of points that we want to cover optimally using a disk. For $P' = \{p_1\}$, a disk centered at p_1 with radius 0 is still optimal.

Next, let $P' = \{p_1, p_2\}$, see Figure 5a. If the smallest disk d centered at c covering P' does not intersect any obstacle, we are done. Otherwise, d intersects an obstacle o. We distinguish two cases: (i) If the line segment $\overline{p_1, p_2}$ intersects an obstacle, there is no disk that covers both points without intersecting the obstacle, see Figure 6a. (ii) Otherwise, due to the geometry of circles, the smallest disk that covers both points must have both points on the boundary and a third point on the boundary of the obstacle o. We move c along the perpendicular bisector of $\{p_1, p_2\}$ until the disk is tangential to o. We refer the reader

to Appendix A for details on the construction and the extension to polygonal obstacles.



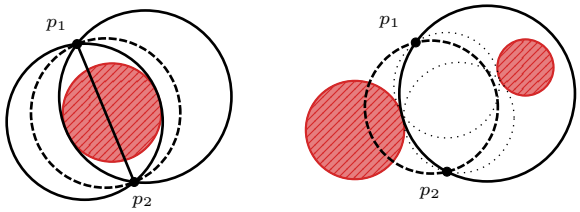
(a) $|P| = 2$. (b) $|P| \geq 3$.

Fig. 5: Obstacle shown in red. (a) The smallest circle (dashed) covering p_1 and p_2 centered at c intersects the obstacle. The smallest feasible disk (solid) is obtained by moving the center along the bisector of $\{p_1, p_2\}$, until the disk no longer intersects the obstacle. (b) extending the approach to $|P| \geq 3$.

Next, we consider the case $|P'| \geq 3$. If the smallest circle does not intersect any obstacle, we are done. Otherwise, we again know that the smallest disk that covers all points has one point on the boundary of the obstacle and two points of P' on the boundary of the circle, see Figure 5b. We can apply the procedure above for each pair of points to find the smallest feasible disk.

Finally, we have to deal with infeasibility, see Figure 6. As mentioned above, if the line segment $\overline{p_1, p_2}$ intersects an obstacle, by convexity of circles, there is no feasible disk for any set of points P with $\{p_1, p_2\} \subseteq P$. Let d' be the smallest feasible disk and p_o be the intersection point of d' and o. W.l.o.g, assume that (p_1, p_2, p_o) is a right turn, see Figure 6b. If d' is intersecting any other obstacle o' in a point $p_{o'}$ and $(p_1, p_2, p_{o'})$ is a right-turn, we can push the center of d' further along the bisector until it no longer intersects o'.

If $(p_1, p_2, p_{o'})$ is a left turn, there exists no disk that covers P' without intersecting either o or o'. Intuitively, there is a bottleneck between the two obstacles, such that we cannot place a disk that covers both points without intersecting at least one of the obstacles.



(a) Line segment $\overline{p_1, p_2}$ intersects an obstacle. (b) Circle is constrained by two obstacles on opposite sides.

Fig. 6: Points $\{p_1, p_2\}$ cannot be covered by the same disk. The dashed circle d is the smallest that covers both points. (a) Infeasible by convexity of circles. (b) The circle d intersects the left obstacle. Adjusting the disk to avoid the obstacle yields the solid circle, which then intersects the right obstacle. The dotted circles are feasible, but neither covers both points.

V. Heuristic Solutions

We now present heuristic approaches for both the DNE-GMC and DNC-GMC. Building on the discretization scheme introduced in Section IV, we leverage the set of possible disks to design efficient solution strategies tailored to the unique constraints of each version. For the DNE-GMC, we adapt and extend a GMC heuristic to handle obstacles and ensure feasibility. For the DNC-GMC, where the constraint precludes simple expansion, we develop a genetic algorithm that explores the search space of feasible disk placements.

A. Do Not Enter GMC Heuristic

We extend the GMC heuristic to handle obstacles. A point p_j is *over-covered* if it is covered by more than $\kappa(p_j)$ disks. Each disk is determined by at most three boundary points, which we call *supporting points*. Starting from an obstacle-agnostic GMC solution [1], we first check whether all disk centers lie outside obstacles. If not, each violating disk is adjusted using the optimal strategy in Section IV-A, producing a feasible solution that may still over-cover some points.

Let $\mathcal{V}$ denote the set of over-covered supporting points. We iteratively select the largest disk containing a supporting point $p_j \in \mathcal{V}$, remove p_j from the disk, and recompute the smallest feasible disk. After updating $\mathcal{V}$, the process repeats until no over-covered points remain. The final solution is a local optimum: no supporting point can be removed without violating coverage requirements. Termination is guaranteed, as each step either reduces the solution cost or the size of $\mathcal{V}$.

B. Do Not Cover GMC Heuristic

A greedy heuristic like the one in Section V-A is unsuitable for the DNC-GMC because disks cannot be arbitrarily expanded to generate a feasible solution. Instead, we explore the search space with a *genetic algorithm*, with any chromosome corresponding to an arrangement of m disks from the candidate set as generated in Section IV-B. To initialize the population, we use a heuristic approach: 10% of the chromosomes are high-quality seeds constructed by greedily selecting disks that maximize coverage of currently insufficiently covered points, while the remaining 90% are generated by random selection to promote diversity. The quality of each solution is evaluated using

$$f(Y, r) = \sum_{j=1}^{n} \mathbb{I}\left(\sum_{i=1}^{m} \mathbb{I}(\|p_j - y_i\|_2 \leq r_i + \epsilon) \geq \kappa_{p_j} \right)$$

as a fitness function, with $Y = \{y_1, \ldots, y_m\}$ denoting the disk centers, $r = \{r_1, \ldots, r_m\}$ the radii, p_j the j-th point, κ_{p_j} the coverage requirement for point j; ϵ is a tolerance parameter, and $\mathbb{I}(\cdot)$ is the indicator function.

For the genetic operators, we perform the crossover by creating a gene pool with disks from both parents. At each selection round we choose the highest scoring disk, with the score given by

$$s_i = \sum_{j \in C_i} \max(0, k_j - c_j) \cdot \lambda + \frac{1}{\lambda}|C_i| + \xi,$$

where C_i is the set of points covered by disk i, c_j is the current coverage count for point j, $\xi \sim U(0, 0.5)$ is a random tie-breaking component, and $\lambda = 10$ is a scaling factor. For mutation, 95% of offspring undergo a procedure in

which one of their disks is replaced by a candidate disk that maximizes coverage of currently insufficiently covered points. The replacement disk is selected by maximizing

$$s_{\text{mut}}(d) = \sum_{j \in I} \mathbb{I}(j \in C_d) + \eta,$$

where $I = \{j : c_j < k_j\}$ is the set of insufficiently covered points, C_d is the set of points covered by candidate disk d, and $\eta \sim U(0, 0.1)$ is a small random component. The disk to be replaced is the one covering the fewest insufficient points, i.e.,

$$i_{\text{remove}} = \arg \min_{i=1,\dots,m} \sum_{j \in I} \mathbb{I}(j \in C_i).$$

To accelerate coverage queries, we partition the 2D space into a uniform grid of cell size $\Delta = \bar{r}$ (the average candidate disk radius), and index each disk in all grid cells its bounding box intersects.

VI. Exact Solutions

An effective method for finding optimal solutions to NP-hard problems is Integer Programming (IP). Although solving an IP can take exponential time in the worst-case scenario, using meticulously designed mathematical models, specialized algorithm engineering, and existing IP solvers, allows IP to solve considerably large instances to provable optimality.

Given a discrete candidate set $\mathcal{C}$, we can formulate an integer program as follows. For every disk d_i in the candidate set $\mathcal{C}$, we define integer variables x_i that encode the number of times each disk is used in the solution. The number m_i is the maximum coverage redundancy of any point covered by the disk d_i, i.e., $m_i \coloneqq \max_{p_j \in d_i} \kappa(p_j)$. The objective is to minimize the total area of the disks used in the solution. The constraints ensure that at most m disks are placed and every point $p_j \in P$ is covered by at least $\kappa(p_j)$ disks.

$$\begin{aligned}
\text{minimize} \quad & \pi \cdot \sum_{d_i \in C} r_i^2 x_i \\
\text{subject to} \quad & \sum_{d_i \in C} x_i \le m \\
& \sum_{\substack{d_i \in C \\ p_j \in d_i}} x_i \ge \kappa(p_j), \qquad \forall p_j \in P \\
& x_i \in \{0, \dots, m_i\}, \quad \forall d_i \in C
\end{aligned}$$

Recall that for the DNE-GMC, we can choose an arbitrary $m \ge \kappa_{max}$ to ensure a feasible solution. For the DNC-GMC, we proved that even finding a feasible assignment is NP-hard (see Theorem 1). Therefore, we cannot choose an arbitrary m to ensure a feasible solution. Instead, we can combine the Integer Programming formulation with a binary search to find the smallest m that yields a feasible solution.

To compute optimal solutions for the DNE-GMC and DNC-GMC, we enumerate a discrete set of candidate disks $\mathcal{C}$. For the GMC, there are three ways in which a set of points $P' \subseteq P$ can be covered optimally (i.e., with minimum-area) by a disk [1].

a) For $P' = \{p_1\}$, a disk centered at p_1 with radius 0 is optimal.
b) For $P' = \{p_1, p_2\}$, there is a unique disk with radius $\frac{\|p_i - p_j\|}{2}$ centered at the midpoint of $\overline{p_1, p_2}$ that is the minimum-area disk that contains both points.
c) For $|P'| \ge 3$, there is a minimum-area disk that has either (i) two points $p_1, p_2 \in P'$ spanning a diameter (which resembles case b)) or (ii) at least three points $p_1, p_2, p_3 \in P'$ on its boundary. This is because every other disk can be shrunk and translated until either (i) or (ii) is satisfied.

We adapt the enumeration scheme to account for the additional obstacles and constraints of our variants as follows.

Candidate set for the DNE-GMC: Given the optimal displacement strategy from Section IV-A, we can compute a candidate set $\mathcal{C}$ of disks for the DNE-GMC as follows.

1) Compute the GMC candidate set $\mathcal{C}'$ (see Section VI).
2) For each candidate disk $d_i \in \mathcal{C}'$,
   - if d_i does not intersect any obstacle, add it to $\mathcal{C}$.
   - otherwise, compute the smallest feasible disk d_i' and add it to $\mathcal{C}$. See Section IV-A for details.

To compute the optimal displacements we use the algorithm from Woeginger [26], who applied techniques from computational geometry to reduce the number of bisectors and projections that need to be considered. The method starts by computing the *farthest point Voronoi diagram* [27] of P'. Now, only the projection of points on the convex hull and the Voronoi ridges as bisectors have to be considered in the algorithm. We further optimize the intersection tests by putting the disk centers of $\mathcal{C}'$ into a KD-tree. For each obstacle o_i, we identify potentially intersecting disks by querying the minimum enclosing circle of o_i. A point-in-polygon test can be easily implemented using geometric primitives, see Section VII for implementation details.

Candidate set for the DNC-GMC: We propose a simple enumeration scheme to compute the candidate set $\mathcal{C}$ based on the observation from Section IV-B.

1) For each point $p_i \in P$, include the trivial disk of radius zero centered at p_i in $\mathcal{C}$.
2) For each pair $\{p_1, p_2\} \subseteq P$, try to compute the smallest feasible disk as described in Section IV-B and include it in $\mathcal{C}$ if it exists.
3) For each triplet of points not containing any infeasible pair, compute the smallest obstacle-agnostic disk and include it in $\mathcal{C}$ if it is feasible. Otherwise, the smallest feasible disk was already computed while considering all pairs of points.

Intersections between disks and polygonal obstacles are efficiently checked using incircles and smallest enclosing circles. All obstacles are indexed in a quadtree, allowing for fast queries for each disk. The implementation of both schemes was parallelized to leverage multiple CPU cores.

VII. Empirical Evaluation

In this section, we empirically evaluate the performance of the presented algorithms. In particular, we design experiments to answer the following research questions (RQs).

RQ1 How does the cost of a GMC solution compare to the cost of the DNE-GMC?

RQ2 How does the DNE-GMC heuristic compare to the exact solver and how do n and m influence their performance?
RQ3 How does the number of obstacles influence the candidate set $\mathcal{C}$ for the DNC-GMC?
RQ4 How does the heuristic perform when trying to find feasible solutions for the DNC-GMC?
RQ5 What is the performance of both exact solvers on real-world problems?

A. Experimental Setup

Experiments were carried out on Linux desktop workstations with AMD Ryzen 9 7900 ($12\times3.7\,\mathrm{GHz}$) CPUs and $100\,\mathrm{GB}$ of RAM running Ubuntu 24.04.3 LTS. Code and data with benchmark problems and results are available [28]. We use the Gurobi [6] solver for the Integer Program (IP). Unless explicitly stated, we use the default optimality gap of 0.01%. The implementation of computationally heavy parts, such as the candidate set enumeration, was implemented in C++ while model building was implemented in Python. We used *Computational Geometry Algorithms Library* (CGAL) [29] for geometric predicates, e.g., for point-in-polygon tests.

B. Instances

We generate instances based on multiple classes of point sets. The instances used to answer each research question can be found in their respective sections.

`rand` Instances generated uniformly at random.
`GMC` Instances from a previous paper on the GMC [1].
`public` Instances from well-known publicly available benchmark instance sets used in [30]. These include point sets previously used in the CG:SHOP challenges [31], [32], TSPLIB instances [33], instances from a VLSI dataset [34], and point sets from the Salzburg Database of Polygonal Inputs [35]. To avoid numerical issues, we scale the Salzburg Database instances from a $[0,1]^2$ bounding box to a $[0,100]^2$ bounding box.
`OSM` Instances derived from *Open Street Map* (OSM) data [36]. Our dataset covers 315 major cities with diverse environments. Assets correspond to ATMs, and obstacles represent restricted areas such as airports, military zones, government buildings, and nature reserves.

C. RQ1: Comparison to the GMC

To answer RQ1, we compare the performance of our solvers for the DNE-GMC on instances from a previous work on the GMC [1]. We modify the point sets to obtain instances for the DNE-GMC problem as follows.

Polygons were generated with CGAL's random polygon generator and values for κ were sampled uniformly from $\{1,2,3\}$, unless otherwise stated. For the DNE-GMC, we add $\lceil n \cdot l \rceil$ polygonal obstacles to the point sets. The size of the obstacle is bounded by the distance to the k-th closest point.

We choose $l = 0.1$ and $k = 20$. This yields the following instance sets:

- `fix_m` and `fix_n`: We modify *uni_sm* and *uni_fix_n* [1] by sampling random points of the instances as obstacle centers. This yields 5 instances for each number of assets $n = 20, 30, \ldots, 200$ with a fixed value of $m = 20$ and 5 instances for each number of drones $m = 5, 10, \ldots, 100$ with a fixed value of $n = 250$, respectively.

Figure 7 shows the performance of the IP with the candidate set enumeration and compares it to runtimes from GMC. We observe that the performance is comparable, but the computation of C is more costly due to the necessary disk displacements. The fourth row in Figure 7 shows the relative size of the candidate set between the DNE-GMC and the GMC problem. Making the disks feasible generates duplicate disks, which decreases the size of the candidate set by $20-30\%$. This reduces the size of the IP and could explain the runtime advantages when finding an optimal IP solution.

Similar to the GMC (see [1]), the IP performance seems to be better for small and large m. The reason may be a smaller set of feasible solutions and better bounds due to enforced large (for small m) or small disks (for large m), respectively.

Figure 7 also shows that the objective values of the optimal GMC and DNE-GMC differ most when m is large relative to n. This is because for small disks, the displacement leads to a relatively larger proportional change in the total area.

D. RQ2: Performance of the DNE-GMC heuristic

We again use instances from the `fix_m` and `fix_n` benchmark sets. Figure 7 shows the performance of the DNE-GMC heuristic. It can be observed that the heuristic produces solutions that are $20-40\%$ optimal even for instances with large m or n. For small m in `fix_m`, the heuristic took less than one second for all instances, while for `fix_n` the runtime increases with m up to $13\,\mathrm{s}$.

E. RQ3: Candidate Set $\mathcal{C}$ for the DNC-GMC

The DNC-GMC generalizes the GMC: if there are no obstacles, the DNC-GMC reduces to the GMC problem which can be reliably solved for instances with up to 300 points, see Section VII-G. Conversely, we can add a large number of obstacles, such that there does not exist a disk that covers any pair of points, making only the trivial solution of placing $\sum_{p\in P}\kappa(p)$ degenerate disks with radius 0 feasible.

To investigate how the size of the candidate set varies with the number of obstacles, we generate instances following the approach described in Section VII-C. Specifically, we add $\lceil n \cdot l \rceil$ polygonal obstacles to each point set. To ensure feasibility, we have to restrict the size of each obstacle to ensure that it does not contain any other points. To achieve this, we bound the size of each polygon by the distance to its nearest neighbor, i.e., we set $k = 1$.

We generate the `rand` instance set, with 5 uniformly random instances for each $n = 250, 500, \ldots, 5000$, with incrementally added obstacles for different values of l.

Figure 8 shows the candidate set size for different percentages of obstacles. For 5% obstacles, the candidate set already

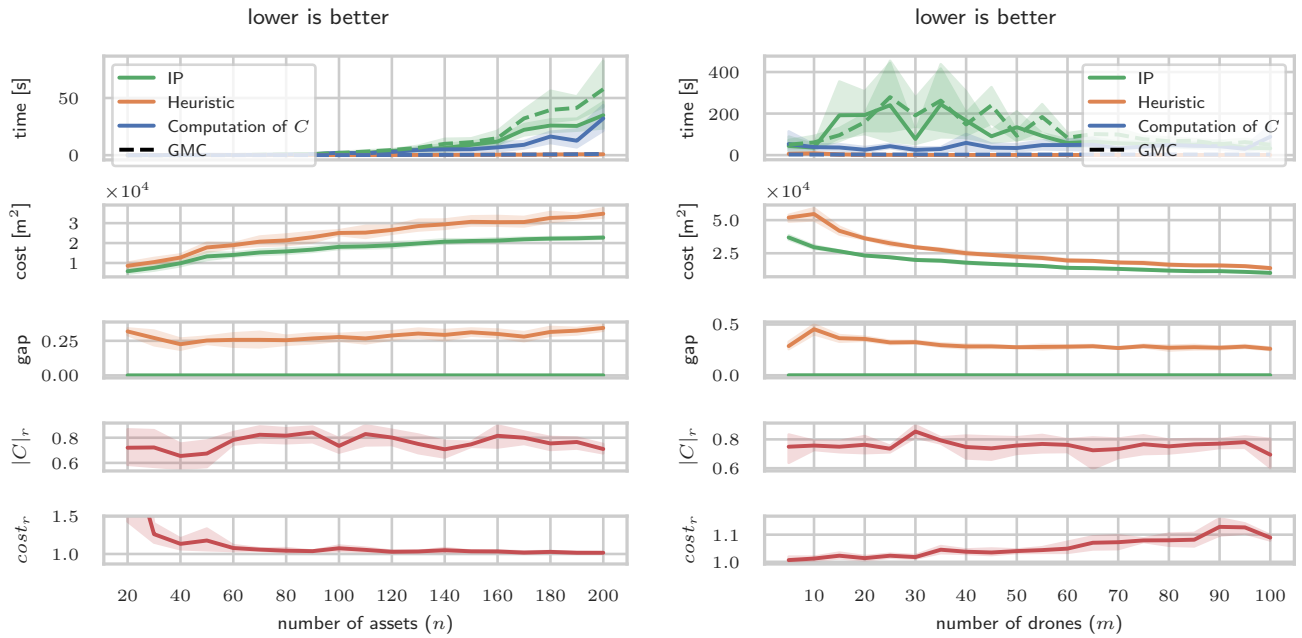


Fig. 7: Comparison of runtime, cost (total area), and optimality gap between the heuristic and IP solver for the DNE-GMC problem on `GMC` instances. The top row compares also compares the performance of the solver if the GMC problem (ignoring obstacles) is solved instead of the DNE-GMC problem. The bottom rows show the relative size of the candidate set C and the relative cost of the solutions, between the DNE-GMC problem and the GMC. (Left) `fix_m`: fixed $m = 20$ and varying n. (Right) `fix_n`: fixed $n = 250$ and varying m.

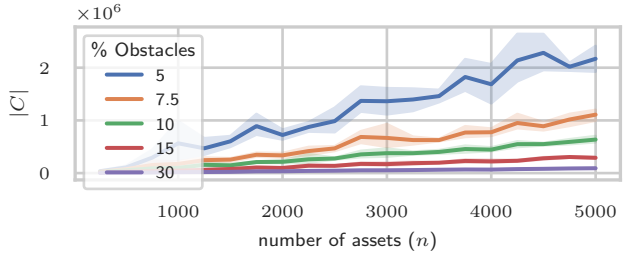


Fig. 8: Size of the candidate set C for the DNC-GMC problem on `rand`. The number of obstacles is a percentage of the number of assets n.

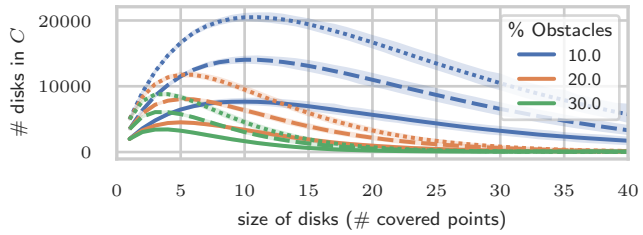


Fig. 9: Number of disks in the candidate set C of a specific size for the DNC-GMC problem on `rand`. The line styles indicate the different instances sizes of 2000 (solid), 3500 (dashed), and 5000 (dotted) assets. The number of obstacles is a percentage of the number of assets n.

contains about one million disks for 2500 assets. Interestingly, the candidate set size grows linearly with the number of assets. Figure 9 shows how when there are many obstacles, each asset can only generate valid disks in combination with nearby assets that are not separated by obstacles. Because we choose the number of obstacles as a percentage of the number of assets, the number of potential candidate disks per asset remains roughly the same with increased number of assets, leading to a linear growth of the candidate set size.

For the remaining experiments, we choose $l = 0.1$ obstacles, which gives a good compromise between the size of the candidate set and the number of obstacles.

F. RQ4: Heuristic of the DNC-GMC

In Section III, we showed that the decision version of the DNC-GMC problem is NP-hard. Thus, unless P=NP, there is no hope of finding a polynomial-time heuristic that produces feasible solutions for all inputs. We evaluate the performance of our genetic algorithm from Section V-B in finding feasible solutions for the DNC-GMC.

For this experiment, we again use the `rand` instance set from Section VII-E with instances with up to 5000 vertices and 10% obstacles. As not all values of m yield feasible solutions, we investigated suitable values of m and found that $m = n/3$ works well for all instances, see Appendix B for details.

We evaluated the performance of the DNC-GMC heuristic and tuned its parameters for robust convergence across all instances. During this process, we determined that a population size of 10, mutation probability of 0.95, and parent selection ratio of 0.7 provided a reliable balance between solution quality and computational efficiency. The heuristic was then compared to the IP solver with an optimality gap of 1%. However, the heuristic's objective was solely to find a feasible solution, without regard to solution quality, whereas the IP solver provides solutions with explicit performance guarantees (e.g., within 1% optimality).

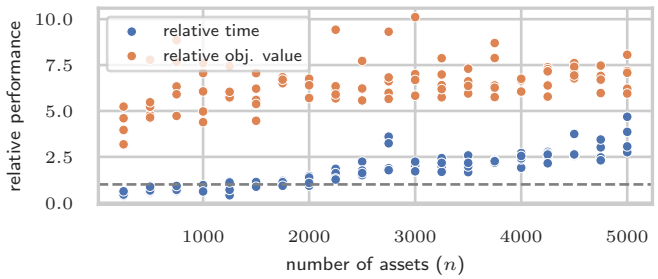


Fig. 10: Relative runtime and objective value between the genetic algorithm and the IP solver for the DNC-GMC on pub_c. The IP was solved to 1% optimality.

Figure 10 shows the result of the experiment. The heuristic can find feasible solutions faster than the IP solver only for small instances with $n \leq 2000$. However, the IP solver was able to provide solutions in less than 18 s for all instances with up to 2000 points. A relative speedup in this regime is thus not relevant in practice. For larger instances, the runtime of the heuristic exceeds that of the IP solver up to a factor of 5. The solution quality is generally poor with an average optimality gap of more than 500%. Finding feasible solutions for the DNC-GMC is challenging for the proposed heuristic. Other attempts to reliably find feasible solutions, such as greedy approaches, also failed. Therefore, the IP solver is the preferred option when tackling this problem. In some cases, it may be necessary to set a higher optimality gap to obtain solutions within reasonable time.

G. RQ5: Real-world instances

We now evaluate the performance of both GMC variants on real-world instances. For that we use instances from Open Street Map data and public benchmark sets, see Figure 15 for examples of both variants.

Do Not Enter GMC: We generated instances from the public points sets and OSM data and set $m = n/5$.

- pub_e: This set includes 184 public benchmark instances, each with fewer than 300 assets. Obstacles are generated by sampling points from each instance as obstacle centers.
- osm_e: This set comprises 266 OSM-derived instances with fewer than 300 assets.

The overall performance of both the IP solver and the heuristic is similar to the performance on random instances from Section VII-C. Figure 11 shows that there is more variation in the quality of heuristic solutions with the average optimality gap for instances with more than 100 assets being close to 27% and 42% for pub_e and osm_e, respectively. We observe that the IP can solve all instances within 4 min to provable optimality. Also, some outliers appear in osm_e during the construction of covering sets, due to the presence of complex obstacles.

Do Not Cover GMC: The DNC-GMC allows us to evaluate the performance of our approach on larger instances with more assets, see Figure 16 for examples. We generated the following benchmark set.

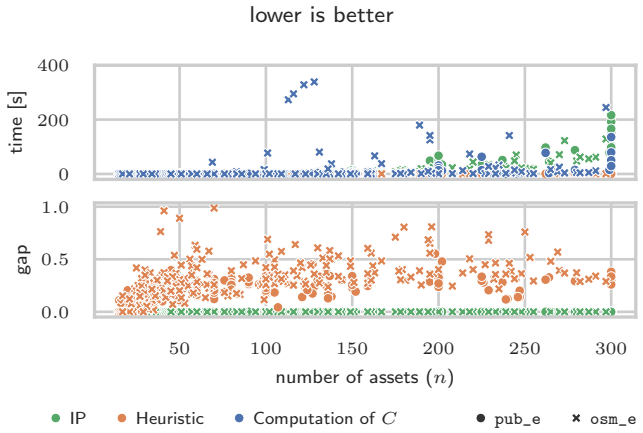


Fig. 11: Comparison of runtime and optimality gap between the heuristic and IP solver for the DNE-GMC problem on pub_e and osm_e. The figure excludes two outliers from osm_e that required 14 and 42 min to compute the covering sets.

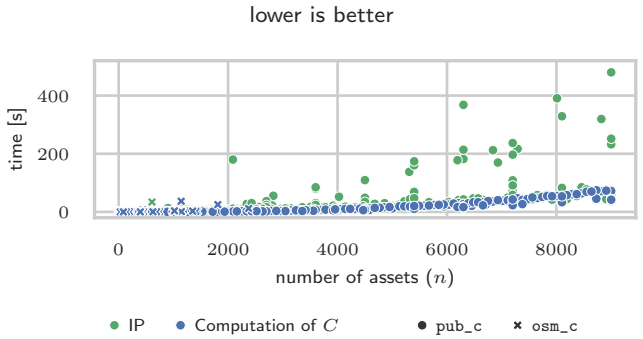


Fig. 12: Runtime of the IP and covering set construction for the DNC-GMC on pub_c and osm_c. The osm_c instances are shown as crosses. Four instances from pub_c required more than 500 s, with the two longest instances taking as long as 25 min and 45 min to solve to optimality.

- pub_c: This set comprises all public point sets, totaling 555 instances with up to 10 000 assets. To preserve the internal structure of each instance, obstacles are generated by sampling points from the instance as obstacle centers; sampled points are removed to avoid overlap. Following the results in Section VII-E, 10% of the points are replaced by obstacles. As for rand, we found that $m = n/3$ works well for all instances, see Appendix B for details.
- osm_c: This set contains all 315 OSM instances with up to 2375 assets. Given the heterogeneous distribution of obstacles, we set m to be 10% higher than the minimum number of required disks, see Appendix B for details.

As our experiments from Section VII-F show, the genetic algorithm is not able to find feasible solutions for most instances in a reasonable time and also produces solutions that are far from optimal. Therefore, we focus on the performance of the IP solver.

Figure 12 shows the runtime of the candidate set enumeration and the IP that solve the instance set to 0.01% optimality. Overall, the runtime for enumerating the candidate disks is generally comparable to that of solving the IP. However, there are some outlier instances where solving the IP requires

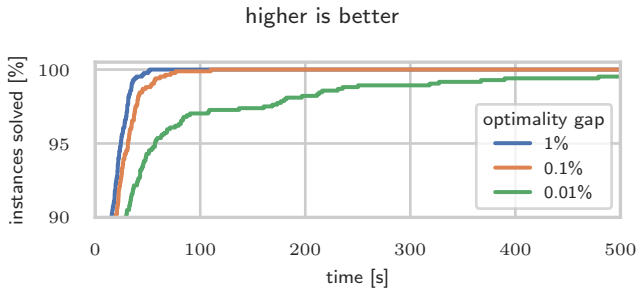


Fig. 13: Percentage of instances solved to specific optimality gap for the DNC-GMC problem. All instances from pub_c and osm_c were solved to within 1% and 0.1% optimality within 52 and 110 s, respectively. See caption of Figure 12 for further details.

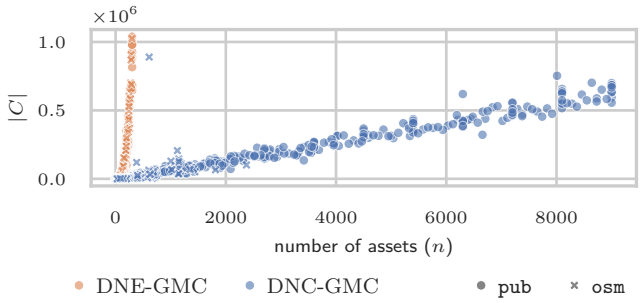


Fig. 14: Size of candidate set $\mathcal{C}$ for the DNE-GMC and DNC-GMC problems on the benchmark instances pub_e, osm_e and pub_c, osm_c, respectively.

significantly more time than the candidate set enumeration.

We conducted a second experiment where the IP was instead solved to 0.1% or 1% optimality. The results can be seen in Figure 13. They show that proving 0.01% optimality is significantly more expensive than proving 0.1% optimality, which in turn is only slightly more expensive than proving 1% optimality. If the practical application can afford an optimality gap of less than 0.1% or 1%, all instances of the benchmark set would be solvable within 110 s or 52 s, respectively.

Limiting Factors: As can be seen in Figures 11 and 12, the performance of the solvers drastically differs between DNE-GMC and DNC-GMC. The primary limiting factor in our IP is the high memory consumption resulting from the size of the candidate set. Figure 14 illustrates the growth of the candidate set for both problems. For the DNE-GMC, the number of candidate disks reaches one million when covering just 300 points. In contrast, for the DNC-GMC, sampling only 10% of the obstacles yields a candidate set of a similar order of magnitude, even when considering as many as 9000 assets. Notably, there is an outlier in the osm_c benchmark with relatively few obstacles, which results in a much larger candidate set compared to other instances.

VIII. Conclusions

We presented several algorithmic methods for solving the *Do Not Enter* and the *Do Not Cover* GMC. Our methods are able to solve instances with several thousand assets to provable optimality. Remarkably, our results demonstrate that the DNC-GMC (which appears harder in theory) allows computing provably optimal solutions for instances of considerably larger size. This appears to be mainly because the set of feasible solutions for the DNE-GMC is less constrained, resulting in a considerably larger number of candidates, which are then harder to handle in practice.

In real deployments, obstacles and restricted regions come from GIS layers and onboard perception; both can be updated online. Our solvers could support this by replanning whenever obstacle layers or asset estimates change, using the previous solution to warm-start the IP. If the true footprint is anisotropic (camera FOV, wind, antenna patterns), one can conservatively approximate it by an enclosing disk (safe) or replace disks with convex regions; our discretization could be adapted using obstacle-free convex region computation.

There are many directions for future work. For example, exploring more heuristics for reducing the candidate set for the DNE-GMC, thus allowing provably optimal solution for larger instances. Covering assets in 3D is natural for many robot domains such as flying robots, space applications, or undersea sensor networks. A practical example is ground targets arranged on terrain with topology (mountains and valleys). Computing candidate centers is still possible, but more complicated [37]. Without obstacles, we can also project the high elevations of points in 3D space to disks at the lowest elevation and discretize the search space using techniques from Xu et al. [38] who study the problem of computing the smallest enclosing circle for a given set of circles.

Further variants of the problems that have additional distance constraints or covering moving targets are of independent interest to a variety of applications [1].

Deits and Tedrake [39] study the problem of computing large convex regions that are not allowed to intersect with obstacles. This could be use to extend our discretization scheme to non-circular sensing regions or to maximize the sensing area [40].

Acknowledgments

Work in Braunschweig was partially supported by the German Research Foundation, project CG:SHOP, FE 407/21-2. Work in Houston was partially supported by the Army Research Laboratory under Cooperative Agreement Number W911NF-23-2-0014. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] M. Guitouni, C. Loi, S. P. Fekete, M. Perk, and A. T. Becker, "Multi-covering a point set by m disks with minimum total area," in *ICRA*. IEEE, 2025, pp. 3000–3006.

[2] United States Government, "B4UFLY," 2025. [Online]. Available: https://www.faa.gov/uas/getting_started/b4ufly

[3] D. Zorbas, T. Razafindralambo, F. Guerriero *et al.*, "Energy efficient mobile target tracking using flying drones," *Procedia Computer Science*, vol. 19, pp. 80–87, 2013.

[4] D. Zorbas, L. D. P. Pugliese, T. Razafindralambo, and F. Guerriero, "Optimal drone placement and cost-efficient target coverage," *Journal of Network and Computer Applications*, vol. 75, pp. 16–31, 2016.

[5] D. Zorbas and C. Douligeris, "Computing optimal drone positions to wirelessly recharge iot devices," in *INFOCOM WKSHPS*. IEEE, 2018, pp. 628–633.

[6] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2025. [Online]. Available: https://www.gurobi.com

[7] S. Bandari and L. N. Devi, "An optimal uav height localization for maximum target coverage using improved deer hunting optimization algorithm," *International Journal of Intelligent Robotics and Applications*, vol. 6, no. 4, pp. 773–790, 2022.

[8] L. Di Puglia Pugliese, F. Guerriero, D. Zorbas, and T. Razafindralambo, "Modelling the mobile target covering problem using flying drones," *Optimization Letters*, vol. 10, no. 5, pp. 1021–1052, 2016.

[9] F. Al-Turjman, H. Zahmatkesh, I. Al-Oqily, and R. Daboul, "Optimized unmanned aerial vehicles deployment for static and mobile targets' monitoring," *Computer Communications*, vol. 149, pp. 27–35, 2020.

[10] M. Shahsavar, S. Rajasekaran, R. Kabin, M. Yannuzzi, and A. T. Becker, "Tracking multiple moving assets with a smaller group of drones," in *CASE*. IEEE, 2025, pp. 171–178.

[11] C. Caillouet, T. Razafindralambo, and D. Zorbas, "Optimal placement of drones for fast sensor energy replenishment using wireless power transfer," in *WD*. IEEE, 2019, pp. 1–6.

[12] H. W. Hamacher and A. Schöbel, "A note on center problems with forbidden polyhedra," *Operations Research Letters*, vol. 21, no. 1, pp. 55–60, 1997.

[13] A. Efrat, S. Har-Peled, and J. S. Mitchell, "Approximation algorithms for two optimal location problems in sensor networks," in *2nd International Conference on Broadband Networks*. IEEE, 2005, pp. 714–723.

[14] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Drone small cells in the clouds: Design, deployment and performance analysis," in *GLOBECOM*. IEEE, 2015, pp. 1–6.

[15] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, "3-d placement of an unmanned aerial vehicle base station (uav-bs) for energy-efficient maximal coverage," *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 434–437, 2017.

[16] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, "Placement optimization of uav-mounted mobile base stations," *IEEE Communications Letters*, vol. 21, no. 3, pp. 604–607, 2016.

[17] H. Alt, E. M. Arkin, H. Brönnimann, J. Erickson, S. P. Fekete, C. Knauer, J. Lenchner, J. S. B. Mitchell, and K. Whittlesey, "Minimum-cost coverage of point sets by disks," in *SoCG*. ACM, 2006, pp. 449–458.

[18] Z. Huang, Q. Feng, J. Wang, and J. Xu, "Ptas for minimum cost multicovering with disks," *SIAM Journal on Computing*, vol. 53, no. 4, pp. 1181–1215, 2024. [Online]. Available: https://doi.org/10.1137/22M1523352

[19] P. Li, Y. Wu, J. Liu, G. S. Sukhatme, V. Kumar, and L. Zhou, "Resilient multi-robot target tracking with sensing and communication danger zones," 2025, submitted 17 Sept. 2024, revised 30 Jul 2025.

[20] A. Saeed, A. Abdelkader, M. Khan, A. Neishaboori, K. A. Harras, and A. Mohamed, "Argus: realistic target coverage by drones," in *IPSN*, 2017, pp. 155–166.

[21] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE RA-L*, vol. 3, no. 4, pp. 3725–3732, 2018.

[22] Shahid and M. A. Shah Md, "UAV coverage path planning of complex regions containing exclusion zones," *arXiv preprint arXiv:2411.07053*, 2024.

[23] S. P. Fekete, L. Kleist, F. Kollhoff, C.-M. Loi, and M. Perk, "Approximation algorithms for lawn mowing with obstacles," in *EuroCG*, 2025.

[24] N. Karapetyan, K. Benson, C. McKinney, P. Taslakian, and I. Rekleitis, "Efficient multi-robot coverage of a known environment," *arXiv preprint arXiv:1808.02541*, 2018.

[25] D. Lichtenstein, "Planar formulae and their uses," *SIAM Journal on Computing*, vol. 11, no. 2, pp. 329–343, 1982. [Online]. Available: https://doi.org/10.1137/0211025

[26] G. J. Woeginger, "A comment on a minmax location problem," *Operations Research Letters*, vol. 23, no. 1–2, pp. 41–43, 1998.

[27] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*. Springer Science & Business Media, 1987, vol. 10.

[28] Anonymous Authors, "Obstacle-constrained multi-coverage of a point set by m disks with minimum total area," Oct. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.17338580

[29] The CGAL Project, *CGAL User and Reference Manual*, 6.0.1 ed. CGAL Editorial Board, 2024. [Online]. Available: https://doc.cgal.org/6.0.1/Manual/packages.html

[30] S. P. Fekete, P. Keldenich, and M. Perk, "Exact Algorithms for Minimum Dilation Triangulation," in *SoCG*, ser. LIPIcs, vol. 332. Schloss Dagstuhl, 2025, pp. 48:1–48:18.

[31] E. D. Demaine, S. P. Fekete, P. Keldenich, D. Krupke, and J. S. B. Mitchell, "Area-optimal simple polygonalizations: The CG Challenge 2019," *JEA*, vol. 27, no. 2, pp. 1–12, 2022.

[32] ——, "Computing convex partitions for point sets in the plane: The CG:SHOP Challenge 2020," *arXiv preprint arXiv:2004.04207*, 2020.

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

[34] "Vlsi data set." [Online]. Available: https://www.math.uwaterloo.ca/tsp/vlsi/index.html

[35] G. Eder, M. Held, S. Jasonarson, P. Mayer, and P. Palfrader, "Salzburg database of polygonal data: Polygons and their generators," *Data in Brief*, vol. 31, p. 105984, 2020.

[36] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org," https://www.openstreetmap.org, 2017.

[37] B. Gärtner, "Fast and robust smallest enclosing balls," in *European symposium on algorithms*. Springer, 1999, pp. 325–338.

[38] S. Xu, R. M. Freund, and J. Sun, "Solution methodologies for the smallest enclosing circle problem," *Computational Optimization and Applications*, vol. 25, no. 1, pp. 283–292, 2003.

[39] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 109–124.

[40] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1647–1650, 2016.

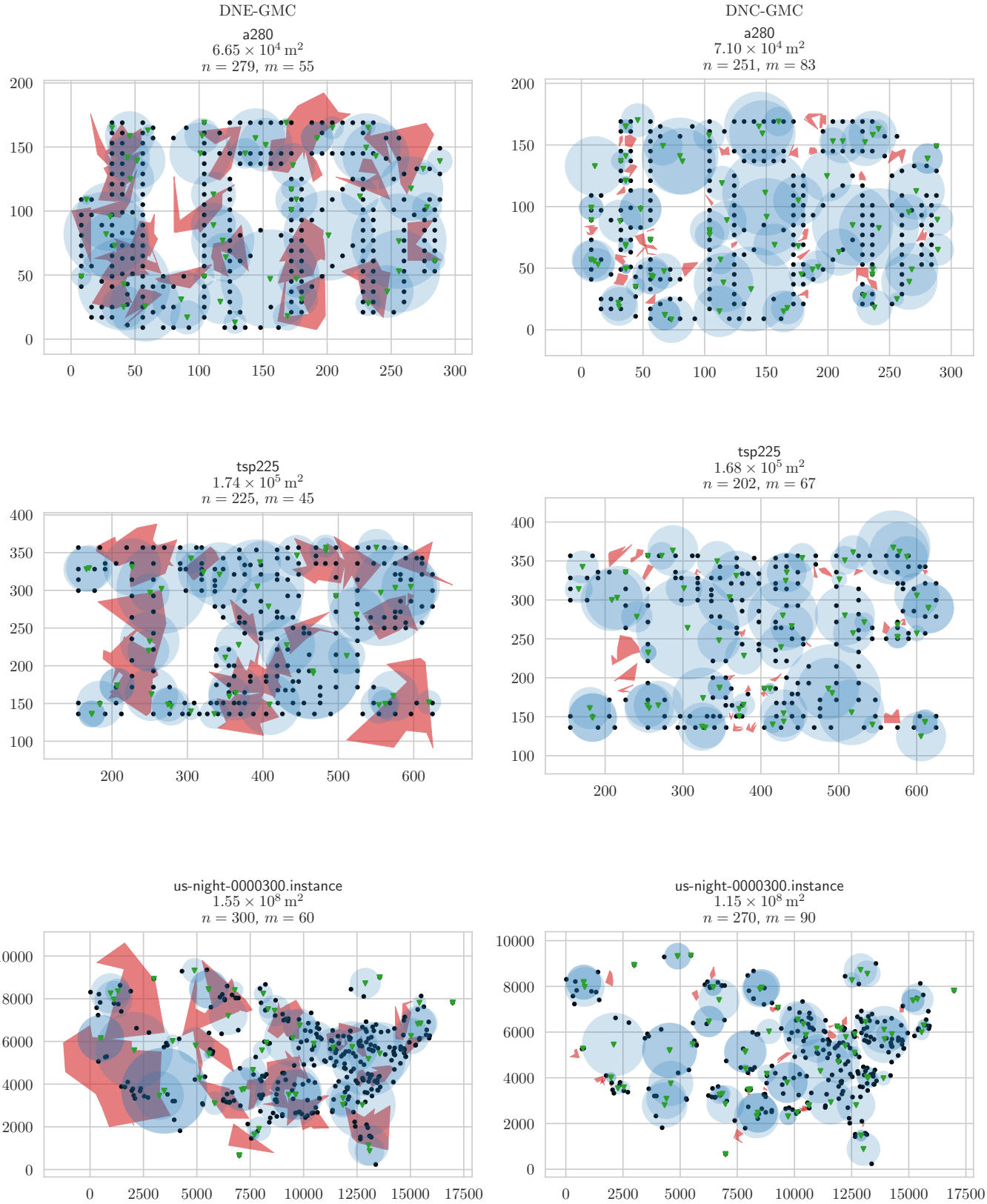


Fig. 15: Optimal solutions for the DNE-GMC and DNC-GMC in comparison. The obstacles differ due to the constraints of each variant. κ is omitted for readability, $\kappa \in \{1, 2, 3\}$.

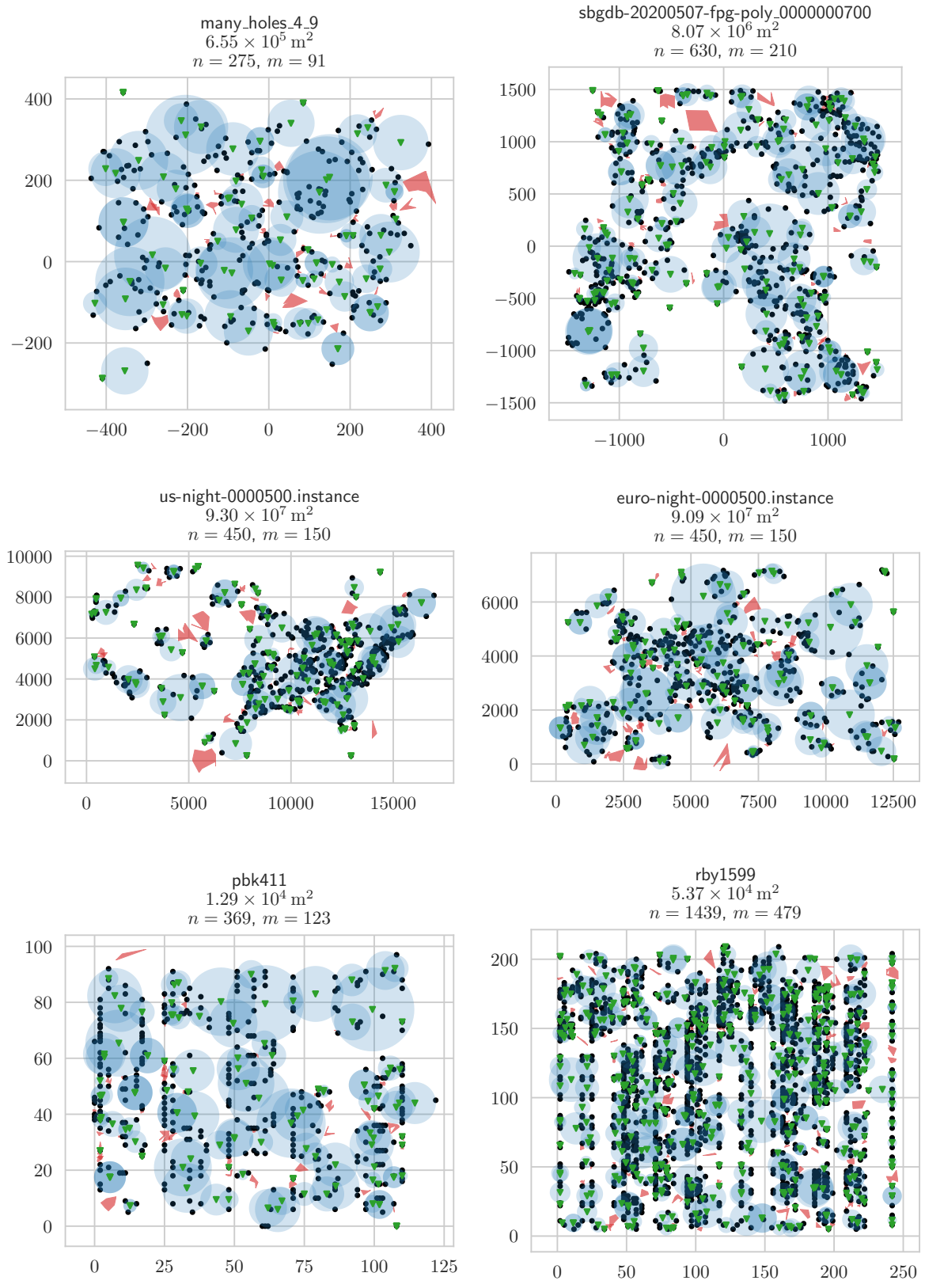


Fig. 16: Optimal solutions for the DNC-GMC on larger instances that could not be solved by the DNE-GMC integer program. κ is omitted for readability, $\kappa \in \{1, 2, 3\}$.

APPENDIX A
DO NOT COVER GMC FORMULAS

In this section, we derive analytical formulas describing optimal circles for the DNC-GMC.

A. Circular Obstacles

Given two points p_1, p_2 and a circular obstacle o centered at e with radius r, such that the smallest disk covering both points centered at c intersects the obstacle, see Figure 5a.

To make the disk feasible, move c along the perpendicular bisector of $\{p_1, p_2\}$ until the disk no longer intersects o.

The function f_n characterizes all points on the perpendicular bisector, i.e., $f_n : \mathbb{R} \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$, $f_n(t, A, B) \coloneqq \frac{A+B}{2} + t \cdot \frac{B_y - A_x}{A_x - B_y}$.

We solve the following equation for t,

$$\|f_n(t, p_1, p_2), e\|_2 = \|f_n(t, p_1, p_2), p_1\|_2 + r$$

In words, we find t such that the distance of the new center and the obstacle center is equal to the radius of the new disk plus the radius of the obstacle. This is the exact point where the disk no longer intersects the obstacle.

Without loss of generality, assume that the obstacle is centered at the origin and (p_1, e, p_2) is a left turn, then the solution is given by the following formula.

$$t(r, p_1, p_2) = \Bigg(r\sqrt{\left(p_{1x}^2 + p_{1y}^2 - r^2\right)\left(p_{2x}^2 + p_{2y}^2 - r^2\right)} \sqrt{\left((p_{1x} - p_{2x})^2 + (p_{1y} - p_{2y})^2\right)} + (p_{1y}p_{2x} - p_{1x}p_{2y})\left(p_{1x}p_{2x} + p_{1y}p_{2y} - r^2\right)\Big) / \left(2(p_{1y}p_{2x} - p_{1x}p_{2y})^2 - 2r^2\left((p_{1x} - p_{2x})^2 + (p_{1y} - p_{2y})^2\right)\right).$$

B. Polygonal Obstacles

Given two points p_1, p_2 and a straight edge $e = \{v, w\}$, such that the smallest disk covering both points centered at c intersects the edge.

Again, to make the disk feasible, we move c along the perpendicular bisector of $\{p_1, p_2\}$ until the disk no longer intersects the edge. Now, we project the center onto the edge e, using $f_p : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$, $f_p(e, p) \coloneqq$

$$e^s + \frac{(p_x - e_x^s)(e_x^t - e_x^s) + (p_y - e_y^s)(e_y^t - e_y^s)}{(e_x^s - e_x^t)^2 + (e_y^s - e_y^t)^2} \cdot (e^t - e^s),$$

where e^s and e^t denote the first and second vertex of the edge e, respectively.

Now we solve for t such that

$$\|c' - f_p(e, c')\|_2 = \|c' - p_1\|_2,$$

where $c' = f_n(t, p_1, p_2)$. In words, we find the point on the perpendicular bisector such that the distance to the edge is equal to the radius of the new disk.

Without loss of generality, we assume that the edge is $\{(0,0), (1,0)\}$ and $(p_1, (0,0), p_2)$ is a left turn, then the solution is given by the following formula.

$$t(p_1, p_2) = \Big((p_{1x} - p_{2x})(p_{1y} + p_{2y}) + 2\sqrt{p_{1y}p_{2y}\left((p_{1x} - p_{2x})^2 + (p_{1y} - p_{2y})^2\right)}\Big) / 2(p_{1y} - p_{2y})^2.$$

For the implementation, we simplify the general formula, keeping the edge variable. For the degenerate case in which the edge is parallel to $\overline{p_1, p_2}$, we compute the circle defined by $(p_1, p_2, f_p(e, (p_1 + p_2)/2))$. This is the circle defined by the two points and the projection of their midpoint onto e.

The newly computed center is tangential to the line $\overline{e^s, e^t}$ but not necessarily to the edge e. If that is the case, the smallest feasible circle is defined by either (p_1, p_2, e^s) or (p_1, p_2, e^t). For a polygonal obstacle, we iterate over every edge.

APPENDIX B
FINDING GOOD VALUES FOR m

Not every instance of the DNC-GMC admits a feasible solution for arbitrary m. In this section, we investigate how to find good values for m that yield feasible solutions in practice, i.e., how to determine the minimum number of disks required to cover all points according to their coverage requirements.

To test for feasibility, we ran our integer programming solver and set the `SolutionLimit` parameter to 1 to stop the execution once a feasible solution is found. The feasibility check was then used to perform a binary search on the minimum value of m for each instance. Naturally, $m = \sum_{p \in P} \kappa(p)$ always allows a trivial feasible solution with a radius of 0 for all m disks. We ran the solver on the following instance sets.

`rand`: Figure 17 shows the minimum value for m found by the binary search procedure. The growth of the minimum m appears to be linear in n and is comfortably below the function $n/3$ for all tested instances.

`pub_c`: Figure 18 shows the minimum value for m found by the IP solver. We observe that $m = n/3$, yields feasible solutions for almost all instances; there are only 17 instances with less than 50 points that do not yield a feasible solution.

`osm_c`: Figure 19 shows the minimum value for m found by the IP solver. Due to the heterogeneous nature of the instances, we do not observe a clear trend.

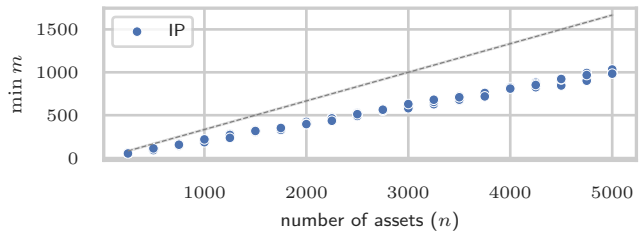


Fig. 17: Minimum number of disks m to ensure a feasible solution for the DNC-GMC problem for instances from `rand` with 10% percent obstacles. The dotted lines represents the function $n/3$.

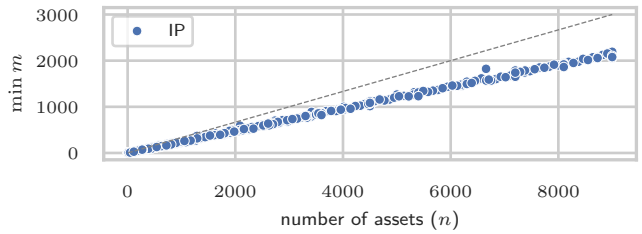


Fig. 18: Minimum number of disks m to ensure a feasible solution for the DNC-GMC problem for instances from `pub_c`. The dotted lines represents the function $n/3$.

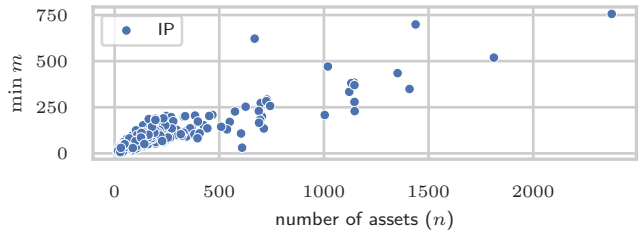


Fig. 19: Minimum number of disks m to ensure a feasible solution for the DNC-GMC problem for instances from `osm_c`.