Drone Air Traffic Control: Tracking a Set of Moving Objects with Minimal Power*

Sándor P. Fekete[1,2], Malte Hoffmann[1], Chek-Manh Loi[1], and Michael Perk[1]

1 **Department of Computer Science, TU Braunschweig**
s.fekete@tu-bs.de, {mhoffman,loi,perk}@ibr.cs.tu-bs.de
2 **L3S, Germany**

Abstract

A common sensing problem is to use a set of stationary tracking locations to monitor a collection of moving devices: Given n objects that need to be tracked, each following its own trajectory, and m stationary traffic control stations, each with a sensing region of adjustable range; how should we adjust the individual sensor ranges in order to optimize energy consumption? We present an algorithm based on geometric insights that is able to find optimal solutions for the min max variant of the problem, which aims at minimizing peak power consumption. Instances with 500 moving objects and 25 stations can be solved in the order of seconds for scenarios that take minutes to play out in the real world, demonstrating real-time capability of our methods.

Related Version *Full version and code* [14–16]

1 Introduction

Keeping track of a collection of moving objects is a fundamental problem with a long history in Computational Geometry. In air traffic control, stationary centers monitor planes with powerful tracking devices to coordinate overall motion. With the growing ubiquity of drones, air traffic control needs further development [4,17], such as the use of less powerful tracking stations: How should we adjust sensing radii over time to minimize power consumption?

Formally, this is the *Kinetic Disk Covering Problem* (KDC), see Figure 1 for an illustration: Given a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of n points that move at constant speeds along linear trajectories $p_i(t) : [0,1] \to \mathbb{R}^2$ over the time interval $[0,1]$ as well as a set $\mathcal{Y} \subset \mathbb{R}^2$ of m centers, the KDC problem asks for an assignment of radii $r_i(t)$ to each center $y_i \in \mathcal{Y}$ at each time t such that all points are covered by at least one disk and the maximum total area of all disks at any time is minimized, i.e., $\min \max_{t\in[0,1]} \sum_{i=1}^{m} \pi r_i(t)^2$. The problem of finding an area-optimal assignment for a (stationary) set of points is known as the *Disk Covering Problem* [1] (DC) and known to be NP-hard. In an optimal solution for the KDC, the radius of each disk is determined by the farthest point assigned to it. We call these *supporting points*. A solution to the KDC is thus a list of k assignments $\{A_{t_0}, A_{t_1}, \ldots, A_{t_k}\}$ with $0 = t_0 \le t_1 \le \ldots \le t_k = 1$. Each assignment $A_t : (\mathcal{Y} \to \mathcal{P})$ assigns supporting points to centers for a specific time window. The radius of the disk centered at y_i at time $t \in [t_j, t_{j+1})$ is simply the distance from y_i to its supporting point, i.e., $r_i(t) = \left\| y_i - A_{t_j}(y_i)(t) \right\|$.

In this work, we present exact and heuristic methods for the KDC. Our approaches start from a stationary solution at time $t = 0$ and use geometric insights to extend it over time (thus producing a feasible solution for all $t \in [0,1]$). We can then locate the time t' at which

* Supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) as part of project *Computational Geometry: Solving Hard Optimization Problems* (CG:SHOP) - 444569951.

42nd European Workshop on Computational Geometry, Hagen, Germany, March 25–27, 2026.
This is an extended abstract of a presentation given at EuroCG'26. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

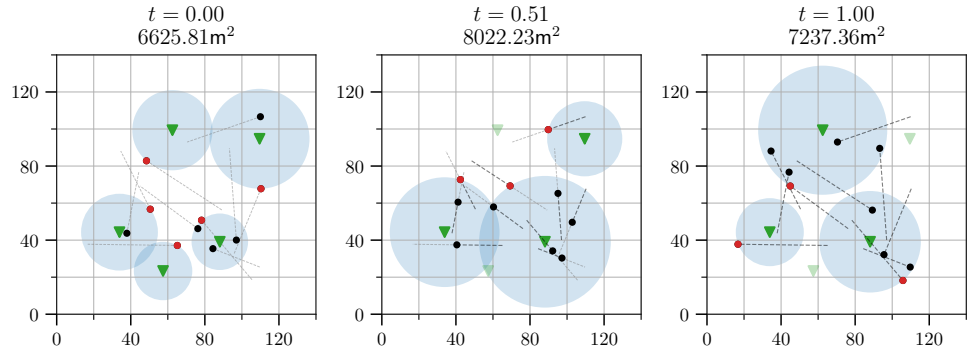


Figure 1 A solution for an instance of the Kinetic Disk Covering Problem for $n = 10$ and $m = 5$. Stations are shown as green triangles, points move along the dotted lines. We start with an IP solution at time $t = 0$. Our algorithm can find a solution over time, that uses at most 8022.23 m^2 of area over the whole time interval. This solution is matched by a lower-bound computed by solving the static DC problem at this time step.

the peak total area is attained and attempt to compute an alternative feasible assignment at t' that, when combined with the current solution, produces an improved overall solution with a smaller maximum area. Furthermore, by formulating the DC as a integer program (IP) we obtain provable lower bounds for the KDC and can certify optimality for many instances; see our empirical evaluation in Section 3.

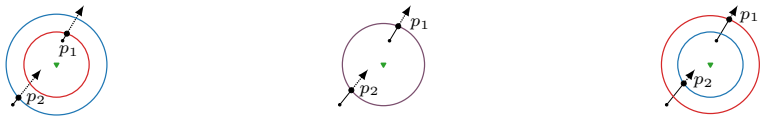
Related Work The DC [1] and its generalization the non-uniform minimum-cost multi-cover (MCMC) problem [12, 13] are known to be NP-hard. Recently, there has been some work on optimally multi-covering fixed points with disks of varying sizes [10]. Other work examines multi-covering points with the minimum number of unit disks, with centers at arbitrary locations [9]. Research on fundamental concepts for kinetic data structures [2] enabled algorithms for k-center and k-means problem for both kinetic stations and objects [3].

2 Algorithmic Approaches

In this section, we discuss how to extend a solution to the (stationary) DC at time t to a solution for the KDC. Further geometric insights allow us to improve extended solutions and design an iterative algorithm to solve the KDC problem. The proofs of the lemmas presented in this section are omitted due to space constraints and can be found in the full version [15].

Stationary Disk Cover We implement two strategies to solve the stationary Disk Cover problem (DC); (1) an IP formulation that can solve the DC problem to provable optimality, and (2) a nearest-neighbor heuristic.

Ensuring Feasibility Every solution to the (stationary) DC problem at time t can be extended to a solution for the KDC problem over time by maintaining the same assignment of objects to stations and altering the supporting point of each station as needed to ensure coverage. This yields a feasible solution for the KDC problem. Figure 2 shows, how initially the supporting point of the station y is p_2. At time $t \approx 0.41$ the disks defined by p_1 and p_2 have the same radius. After this point in time, p_1 becomes the supporting point of the disk.



(a) $t = 0.2$ **(b)** $t \approx 0.41$ **(c)** $t = 0.6$

Figure 2 The supporting point of station y changes from p_1 to p_2 to ensure coverage.

▶ **Lemma 1.** *In $\mathcal{O}(mn^2)$ time, all $\mathcal{O}(mn^2)$ possible supporting point changes when extending a stationary solution to $t \in [0, 1]$ can be computed.*

Our algorithm also handles degenerate cases where the supporting point might not change or multiple objects might have the same distance to a station, see Figure 3.

Figure 3 Degenerate cases after which the supporting point might change.

Improving Extended Solutions It is easy to see that only ensuring feasibility can lead to suboptimal solutions. To reduce the overall cost, we consider a second type of event that changes the assignment of objects to stations, see Figure 4a. Intuitively, we want to identify points in time in which it is cheaper to handover a point from one station to another.

(a) **(b)**

Figure 4 (a) Station y_2 takes over a point p_2. The objective values of the red and blue disks are equal at this point in time. We change the supporting point of both y_1 and y_2 at the same time. (b) Generalization of the case depicted on the left to more than two stations.

▶ **Lemma 2.** *In $\mathcal{O}(m^2n^3)$ time all $\mathcal{O}(m^2n^3)$ possible handovers of objects between two stations (when extending a stationary solution to $t \in [0, 1]$) can be computed.*

The concept of handing over objects between stations can be generalized to three or more stations, see Figure 4b. Furthermore, a chain of stations can successively take over objects from each other as time progresses. Figure 5 shows how initially the optimal solution may consist of blue disks covering all objects. At $t = 0.5$ the optimal solution switches to red disks, and the previously used blue disks are no longer part of the solution. This process can be extended to arbitrarily long chains of objects and stations. In the worst case, detecting

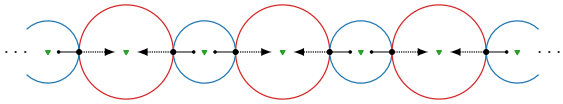


Figure 5 Multiple stations and objects in the same event point. Either the blue or the red disk cover the objects, but not both.

these types of events requires comparing every subset of stations with every other subset, resulting in $\Omega(2^m)$ possible events.

Iterative Algorithm We present an algorithm that uses the beforementioned geometric observations to compute heuristic and optimal solutions for the KDC. It starts by computing a stationary solution at $t = 0$ using any of the methods presented above. In the case of the IP formulation, the algorithm also provides a valid lower bound on the optimal solution which can be used to determine the quality of the solution during the solving process.

We then extend the stationary solution over time by applying the adjustments from Lemma 1, thereby obtaining a feasible KDC solution for the interval $[0, 1]$. To speed up the computation, we cache the next supporting point change for each center y and update the cache when assignments change. The algorithm then proceeds by identifying the point $t' \in [0, 1]$ where the current solution has maximum total area. This point must lie at the intersection of two intervals in which the assignment of objects to stations changes, since the objective value within each interval is an upward-opening parabola. At this point in time we compute a new stationary solution. If the stationary solution has a smaller objective value than the current solution, we can improve our current solution at time t' and potentially over the whole interval $[0, 1]$ or any set of subintervals. This is done by again extending the solution at time t' both in the positive and negative time direction. Afterwards we merge two solutions into a single valid solution, by taking the lower envelope of both functions. If the stationary solution has equal or higher objective value than the current solution, we terminate the procedure. Additionally, if the integer programming formulation is used, we can also terminate if the solution is within a desired optimality gap of the lower bound.

Improvements The first improvement is to the extension of solutions by considering handovers of objects between two stations, see Lemma 2. Again an efficient caching scheme can be used to speed up the computation. The second improvement targets solutions that are obtained while extending the solution. If the supporting point of a center y_i is also covered by another disk y_j, we remove it from y_i, reducing the radius of y_i and therefore the total area of the disks. We can repeat this process until no further improvement is possible. Finally, we reduce the time spent for extending and combining solutions. This can be done by simultaneously extending and combining the stationary solution (which is better than the current solution) only until it intersects the current solution, see Figure 6.

3 Evaluation

Experiments were carried out on desktop workstations equipped with AMD Ryzen 9 7900 (12×3.7 GHz) CPUs and 96 GB of RAM. Code and data are available online [14]. We use Gurobi [11] (version 12) solver for the integer program, with a default optimality gap of 0.01% and a time limit of 600 s. To detect numerical issues for degenerate cases, we added checks to ensure correctness of our implementation. For the instance generation, we used

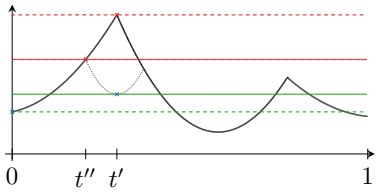


Figure 6 Iterative approach to improve the min max power consumption over time. Green and red lines denote lower and upper bounds on the optimal solution.

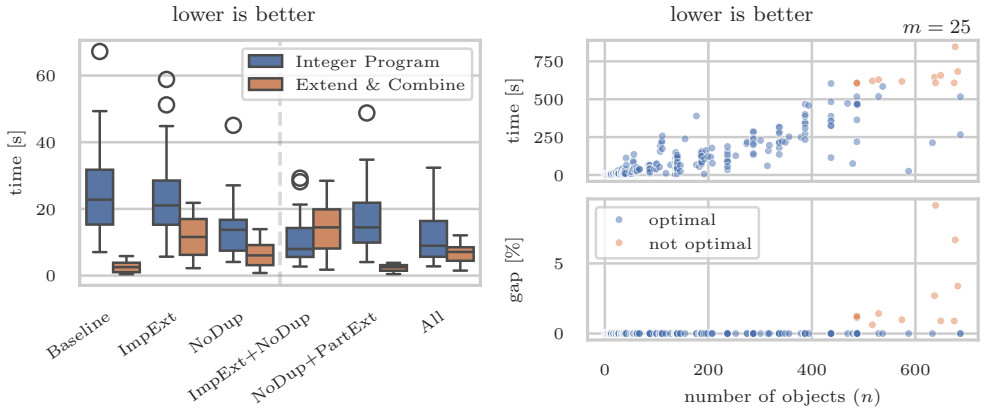
line segments with lengths uniformly distributed in $[25, 50]$ in a 100×100 grid. In real-world applications where objects are drones have velocities up to $100\,\mathrm{km\,h^{-1}}$, this resembles scenarios that take between 15 and 30 minutes to play out on a $100\,\mathrm{km}^2$ canvas. In total, we have the following instances sets.

`fix` We generate 25 instances with $m = 25$ and $n = 500$.

`fix_n` Fix number of objects to $n = 500$. For $m \in \{5, 10, \ldots, 50\}$ we generate 10 instances.

`fix_m` Fix number of sensors to $m = 25$. For $n \in \{50, 100, \ldots, 500\}$ we generate 10 instances.

`pub` Instances from well-known publicly available benchmarks used in [8], such as point sets from the CG:SHOP challenges [5, 6] and [7, 18, 19]. We scaled point sets to a fixed size and sampled $m = 25$ centers and point pairs (for trajectories) at random.



(a) Improvement Strategies **(b)** Public Instances

Figure 7 **(a)** Evaluation of different improvement strategies on the `fix` benchmark set. The results show that enabling all three solution strategies yields the best performance. **(b)** Performance of the exact version of `IP` on the `pub` dataset. Despite the variety and degeneracies, almost all instances can be solved to provable optimality.

Improvement Strategies We evaluate the influence of three proposed improvement strategies to the algorithm from Section 2. We denote the removal of duplicates as `NoDup`, the improved extension in Lemma 2 as `ImpExt` and partial extension as `PartExt`. Our baseline is the exact algorithm that uses the IP solver. We compare the time to compute the stationary IP solutions and the time to extend and combine solutions, see Figure 7a.

Our experiments show that both `ImpExt` and `NoDup` have a positive effect on the time spent in the IP solver and a negative effect on the time to extend and combine solutions. The use of `PartExt` however, reduces the time spent during extension and combination

significantly, which mitigates the negative effect resulting in the best configuration that achieves an overall runtime reduction of about 39 % compared to the baseline. We use the best configuration that enables all three improvement strategies for the following evaluations.

Influence of m and n We evaluate the influence of the number of objects n and the number of stations m on the performance of the algorithm. We execute the algorithm from Section 2 with both an IP solver and the nearest neighbor heuristic to solve the DC problem in each iteration. We denote the IP-based algorithm as `IP` and the nearest neighbor based algorithm as `NN`. Additionally, we implemented another heuristic that divides the time into $k = 10$ same sized intervals and computes the nearest neighbor solution at each border. The algorithm `FixedNN` then extends all solutions and reports the lower envelope. Figure 8 shows the performance of the different algorithms. We note that one instance of `IP` on the `fix_n` benchmark was restarted with a different seed due to numerical instability in the non-exact extension step that caused our checks to fail.

Overall, `IP` is able to solve all instances to optimality in a matter of seconds, demonstrating real-time capability for scenarios that take minutes to play out in the real world.

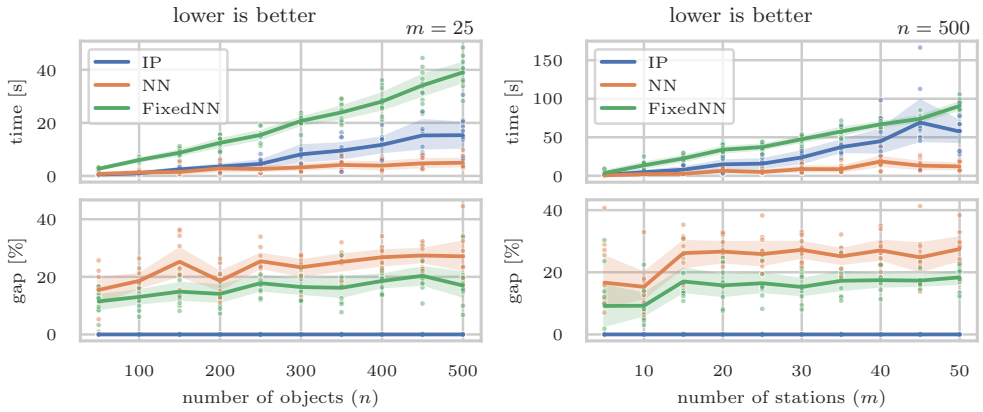


Figure 8 Performance of `IP`, `NN` and `FixedNN` on instances from the `fix_m` and `fix_n` datasets. While `IP` is significantly slower than `NN`, it produces optimal solutions for all instances. `FixedNN` is slightly better in terms of solution quality than `NN` but significantly slower.

Instances from Literature We further assess the algorithm's performance on the `pub` instance set, as shown in Figure 7b and illustrated by Figure 9. The presence of degeneracies, originating from real-world data, led to various numerical challenges in our implementation. To address these issues, we developed an alternative version of `IP` utilizing exact number types for computations. Our results show that 290 out of 302 instances with up to 700 objects could be solved to provable optimality within 600 s of computation time.

4 Conclusions

We provide novel insights into tracking large sets of moving objects from a set of fixed observation stations with optimal energy. While this problem is hard in theory, our practical methods can compute provably optimal coverage in a matter of seconds, demonstrating perspectives for real-world capability. There are many promising directions for future work;

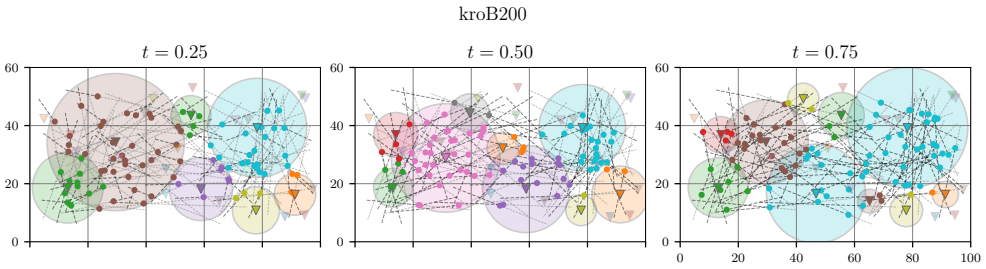


Figure 9 A solution of an instance from `pub` instance set based on *kroB200* TSPLIB [18].

deployment requires addressing practical issues, such as communication delays, online trajectory updates, and three-dimensional coverage. If the stations can be repositioned, the problem becomes even more complex, but also opens up new possibilities for optimization that may further reduce energy consumption. Uncertainty in trajectories and alternative coverage models, such as probabilistic sensing, also present interesting avenues for further research that could enhance the robustness and applicability of our methods in real-world scenarios.

References

1 Helmut Alt, Esther M. Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P. Fekete, Christian Knauer, Jonathan Lenchner, Joseph S. B. Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Symposium on Computational Geometry (SoCG)*, pages 449–458, 2006. doi:10.1145/1137856.1137922.

2 Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999. doi:10.1006/JAGM.1998.0988.

3 Sergei Bespamyatnikh, Binay K. Bhattacharya, David G. Kirkpatrick, and Michael Segal. Mobile facility location. In *International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 46–53, 2000. doi:10.1145/345848.345858.

4 Adam P. Cohen, Susan A. Shaheen, and Emily M. Farrar. Urban air mobility: History, ecosystem, market potential, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):6074–6087, 2021. doi:10.1109/TITS.2021.3082767.

5 Erik D Demaine, Sándor P Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing convex partitions for point sets in the plane: The CG:SHOP Challenge 2020. *arXiv preprint arXiv:2004.04207*, 2020.

6 Erik D Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Area-optimal simple polygonalizations: The CG Challenge 2019. *Journal of Experimental Algorithmics (JEA)*, 27(2):1–12, 2022.

7 Günther Eder, Martin Held, Steinþór Jasonarson, Philipp Mayer, and Peter Palfrader. Salzburg database of polygonal data: Polygons and their generators. *Data in Brief*, 31:105984, 2020. doi:10.1016/j.dib.2020.105984.

8 Sándor P. Fekete, Phillip Keldenich, and Michael Perk. Exact Algorithms for Minimum Dilation Triangulation. In *Symposium on Computational Geometry (SoCG)*, pages 48:1–48:18, 2025. doi:10.4230/LIPIcs.SoCG.2025.48.

9 Xuening Gao, Longkun Guo, and Kewen Liao. Fast approximation algorithms for multiple coverage with unit disks. In *Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 185–193, 2022.

10 Mariem Guitouni, Chek-Manh Loi, Sándor P. Fekete, Michael Perk, and Aaron T. Becker. Multi-covering a point set by m disks with minimum total area. In *ICRA*, pages 3000–3006. IEEE, 2025. doi:10.1109/ICRA55743.2025.11127835.

11 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2025. URL: https://www.gurobi.com.

12 Ziyun Huang, Qilong Feng, Jianxin Wang, and Jinhui Xu. PTAS for minimum cost multi-covering with disks. In *Symposium on Discrete Algorithms (SODA)*, pages 840–859, 2021.

13 Ziyun Huang, Qilong Feng, Jianxin Wang, and Jinhui Xu. PTAS for minimum cost multi-covering with disks. *SIAM Journal on Computing*, 53(4):1181–1215, 2024.

14 Chek-Manh Loi and Michael Perk. Drone air traffic control: Tracking a set of moving objects with minimal power, 2025. doi:10.5281/zenodo.17119781.

15 Chek-Manh Loi, Michael Perk, Malte Hoffmann, and Sándor P. Fekete. Drone air traffic control: Tracking a set of moving objects with minimal power. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. To appear.

16 Chek-Manh Loi, Michael Perk, Malte Hoffmann, and Sándor Fekete. Drone air traffic control: Tracking a set of moving objects with minimal power, 2026. doi:10.48550/arXiv.2603.05286.

17 Franco Mazzenga, Romeo Giuliano, and Alessandro Vizzarri. 5G-based synchronous network for air traffic monitoring in urban air mobility. *IEEE Access*, 12:188542–188559, 2024. doi:10.1109/ACCESS.2024.3513212.

18 G. Reinelt. TSPLIB–A Traveling Salesman Problem Library. *ORSA Journal of Computing*, 3(4):376–384, 1991.

19 Andre Rohe. VLSI data set. URL: https://www.math.uwaterloo.ca/tsp/vlsi/index.html.