On a geometric graph-covering problem related to optimal safety-landing-site location

Claudia D'ambrosio, Marcia Fampa, Jon Lee, Felipe Sinnecker

On a geometric graph-covering problem related to optimal safety-landing-site location

Claudia D'Ambrosio[1], Marcia Fampa[2], Jon Lee[3], and Felipe Sinnecker[2]

[1] LIX CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau
[2] Federal University of Rio de Janeiro
[3] University of Michigan, Ann Arbor

Abstract. We develop a set-cover based integer-programming approach to an optimal safety-landing-site location arising in the design of urban air-transportation networks. We link our minimum-weight set-cover problems to efficiently-solvable cases of minimum-weight set covering that have been studied. We were able to solve large random instances to optimality using our modeling approach. We carried out *strong fixing*, a technique that generalizes reduced-cost fixing, and which we found to be very effective in reducing the size of our integer-programming instances.

Keywords: set covering · 0/1 linear programming · reduced-cost fixing · strong fixing · safety landing site · urban air mobility

Introduction

In the last few years, different actors all around the world have been pushing for the development of Urban Air Mobility (UAM). The idea is to integrate into the current transportation system, new ways to move people and merchandise. In particular, drones are already a reality, and they have the potential to be highly exploited for last-mile deliveries (for example, Amazon, UPS, DHL, and FedEx, just to mention a few involved operators). Concerning passenger transportation, several companies are competing to produce the first commercial electric Vertical Take-Off and Landing (eVTOLs) vehicles, which will be used to move passengers between vertiports of sprawling cities. Several aspects have to be taken into account for this kind of service, the most important one being safety. Air-traffic management (ATM) provides and adapts flight planning to guarantee a proper separation of the trajectories of the flights; see e.g., [14,4,13]. In the case of UAM, some infrastructure has to be built to provide safe landing locations in case of failure or damage of drones/eVTOLs. These locations are called "Safety Landing Sites" (SLSs) and should be organized to cover the trajectory of eVTOLs for emergency landings at any position along flight paths.

In what follows, we study the optimal placement of SLSs in the air-transportation network. We aim at installing the minimum-cost set of SLSs, such that all the drones/eVTOLs trajectories are covered. We show an example in Figure 1. It represents an aerial 2D view of a part of a city, where the rectangles are the roofs of existing buildings. The black crosses represent potential sites for SLSs.

The two red segments are the trajectory of the flights in this portion of the space. The trajectory is fully covered thanks to the installation of 3 SLS over 5 potential sites, namely the ones corresponding to the center of the green circles. The latter represents the points in space that are at a distance that is smaller than the safety distance for an emergency landing. Note that every point along the trajectory is inside at least one circle, thus the trajectory is fully covered.

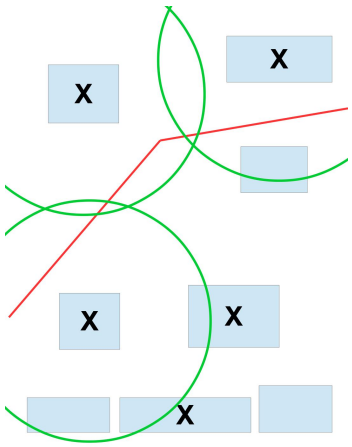

Fig. 1. Example of full covering provided by installing 3 SLSs

The problem of finding an optimal placement of SLSs has not received a lot of attention. In fact, there appears to be no published work on the variant that we are considering. In the literature, we can find the masters thesis of Xu [19], where he studied the problem of SLS placement coupled with the routing problem of drones/eVTOLs for each origin-destination pair. The potential SLSs are assumed to fully cover a subset of the arcs of the considered network, i.e., a partial covering is not allowed. In [15], Pelegín and Xu consider a variant of the problem, which can be formulated as a continuous covering problem. In particular, they interpret the problem as a set-covering/location problem where both candidate locations and demand points are continuous on a network. In this work, we consider the demand points continuous on the network, but the candidate-location set is finite and composed of points not restricted to be in the network; see e.g., Figure 1.

More attention has been accorded to the optimal placement of vertiports; see e.g., [18,17]. However, their location depends on the estimated service demand and, based on the decisions made on the vertiport location, the UAM network is identified. In contrast, we suppose that these decisions were already made. In fact, despite the scarcity of literature on the topic, the main actors in the UAM field assert that pre-identified emergency landing sites are necessary to guarantee safety in UAM; see e.g., [12,8].

Organization and contributions. In §1, we describe our new mathematical model for the optimal safety-landing-site location problem, a generally NP-hard minimum-weight set covering problem. In §2, we see what kinds of set-covering matrices can arise from our setting, linking to the literature on ideal matrices. In §3, we identify three classes of efficiently-solvable cases for our setting. In §4, toward practical optimal solution of instances, we carry out "strong fixing", which enhances the classical technique of reduced-cost fixing. In §5, we present results of computational experiments, demonstrating the value of strong fixing for reducing model size and as a useful tool for solving difficult instances to optimality. In §6, we identify some potential next steps.

Notation. We denote the set of real numbers by $\mathbb{R}$ and the set of positive real numbers by $\mathbb{R}_{++}$. We denote an all-ones vector by $\mathbf{e}$. For a vector $x \in \mathbb{R}^n$, we denote its 2-norm by $\|x\| := \sqrt{\sum_{i=1}^{n} x_i^2}$. For a matrix A, we denote the transpose of A by $A^\intercal$ and column j of A by $A_{\cdot j}$.

1 Covering edges with a subset of a finite set of balls

We begin with a formally defined geometric optimization problem. Let G be a straight-line embedding of a graph in $\mathbb{R}^d$, $d \geq 1$ (although our main interest is $d = 2$, with $d = 3$ possibly also having some applied interest), where we denote the vertex set of G by $\mathcal{V}(G)$, and the edge set of G by $\mathcal{I}(G)$, which is a finite set of intervals, which we regard as *closed*, thus containing its end vertices. Note that an interval can be a single point (even though this might not be useful for our motivating application). We are further given a finite set N of n points in $\mathbb{R}^d$, a weight function $w : N \to \mathbb{R}_{++}$, and covering radii $r : N \to \mathbb{R}_{++}$ (we emphasize that points in N may have differing covering radii). A point $x \in N$ $r(x)$-covers all points in the $r(x)$-ball $B(x, r(x)) := \{y \in \mathbb{R}^d \ : \ \|x - y\|_2 \leq r(x)\}$. A subset $S \subset N$ r-covers G if every point y in every edge $I \in \mathcal{I}(G)$ is $r(x)$-covered by some point $x \in S$. We may as well assume, for feasibility, that N r-covers G. Our goal is to find a minimum w-weight r-covering of G.

Connecting this geometric problem with our motivating application, we observe that any realistic road network can be approximated to arbitrary precision by a straight-line embedded graph, using extra vertices, in addition to road junctions; this is just the standard technique of piecewise-linear approximation of curves. The point set N corresponds to the set of potential SLSs. In our application, a constant radius for each SLS is rather natural, but our methodology does not require this. We also allow for cost to depend on SLSs, which can be natural if sites are rented, for example.

Associated with each ball $B(x, r(x))$ is its boundary, the sphere $\bar{B}(x, r(x))$. Each such sphere $\bar{B}(x, r(x))$ intersects each interval $I \in \mathcal{I}(G)$ at most twice. Collecting all of these at most $2n$ subdivision points, as we let x vary over N, the interval I is subdivided into at most $1+2n$ closed subintervals, the collection of which we denote as $\mathcal{C}(I)$. It is clear that all points in a given one of these subintervals are covered by the same set of balls.

With all of this notation, we can re-cast the problem of finding a minimum w-weight r-covering of G as the 0/1-linear optimization problem

$$\begin{aligned} \min \ & \sum_{x \in N} w(x) z(x) \\ & \sum_{\substack{x \in N \ : \\ J \subset B(x, r(x))}} z(x) \geq 1, \ \forall \ J \in \mathcal{C}(I), \ I \in \mathcal{I}(G); \\ & z(x) \in \{0,1\}, \ \forall \ x \in N, \end{aligned} \tag{CP}$$

where each $z(x)$ is a binary indicator variable associated with selecting the point $x \in N$ (equivalently, the ball $B(x, r(x))$). Because $|\mathcal{C}(I)| \leq 1 + 2n$, for each edge $I \in \mathcal{I}(G)$, the number of covering constraints, which we will denote by m, is at most $(1 + 2n)|\mathcal{I}(G)|$. Of course we can view this formulation in matrix format as

$$\min\{w^{\intercal} z \ : \ Az \geq \mathbf{e}, \ z \in \{0,1\}^n\}, \tag{SCP}$$

for an appropriate 0/1-valued $m \times n$ matrix A, and it is this view that we mainly work with in what follows.

The problem is already NP-Hard for $d > 1$, when all balls have identical radius, the weights are all unity, $N := \mathcal{V}(G)$, and the edges of G are simply the points N; see [9, Theorem 4]. Of course, this type of graph (with only degenerate edges) is not directly relevant to our motivating application, and anyway we are aiming at exact algorithms for practical instances of moderate size.

2 What kind of constraint matrices can we get?

There is a big theory on when set covering LPs have integer optima (for all objectives). It is the theory of *ideal* matrices; see [5]. A 0/1 matrix is *balanced* if it does not contain a square submatrix of odd order with two ones per row and per column. In fact, the 0/1 TU (totally unimodular; see [16], for example) matrices are a proper subclass of the balanced matrices.

Berge [3] showed that, if A is balanced, then both the packing and covering systems associated with A have integer vertices. Fulkerson, Hoffman, and Oppenheim [10] showed that, if A is balanced, then the covering system is TDI (totally dual integral; see [16], for example). So balanced 0/1 matrices are a subclass of ideal 0/1 matrices.

We can observe that the matrices that can arise for us are not generally balanced, already for a simple example, depicted in Figure 2; the drawing is for $n = 5$, but it could have been for any odd $n \geq 3$. The edges of the graph are indicated with red. The points of N are at the midpoints of the black lines (which are not themselves edges). The (green) circle for each point of N goes through the center. It is easy to see that edges are not subdivided by circles, and each circle covers a pair of edges in a cyclic fashion. The constraint matrix of the covering problem is an odd-order (5 in this case) 0/1 matrix violating the

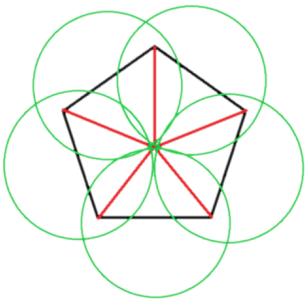

Fig. 2. Does not have an ideal covering matrix

definition of balanced. In fact, the matrix is not even ideal as the covering LP has an all-$\frac{1}{2}$ extreme point.

We can also get a counterexample to idealness for the covering matrix with respect to unit-grid graphs. In particular, it is well known that the "circulant matrix" of Figure 3 is non-ideal, see [6]. Now, we can realize this matrix from

$$\mathcal{C}_8^3 := \begin{pmatrix} 1&1&1&0&0&0&0&0 \\ 0&1&1&1&0&0&0&0 \\ 0&0&1&1&1&0&0&0 \\ 0&0&0&1&1&1&0&0 \\ 0&0&0&0&1&1&1&0 \\ 0&0&0&0&0&1&1&1 \\ 1&0&0&0&0&0&1&1 \\ 1&1&0&0&0&0&0&1 \end{pmatrix}$$

Fig. 3. Non-ideal circulant matrix

an 8-edge "unit-grid graph" (or the reader may prefer to see it as a 4-edge "unit-grid graph"), see Figure 4, and eight well-designed covering disks, each covering an "L", namely $\{a, b, c\}$, $\{b, c, d\}$, $\{c, d, e\}$, $\{d, e, f\}$, $\{e, f, g\}$, $\{f, g, h\}$, $\{g, h, a\}$, $\{h, a, b\}$. As a sanity check, referring to Figure 5, we see that $\mathcal{C}_8^3$ is not balanced.

3 Efficiently solvable cases

In this section, we present three situations for which CP/SCP is efficiently solvable.

3.1 When G is a unit-grid graph in $\mathbb{R}^d$, $d \geq 2$, $N \subset \mathcal{V}(G)$, and $1 \leq r(x) < \sqrt[d]{5/4}$, for all $x \in N$

Briefly, a *unit-grid graph* is a finite subgraph of the standard integer lattice graph. For this case, we will observe that our minimum-weight r-covering problem on

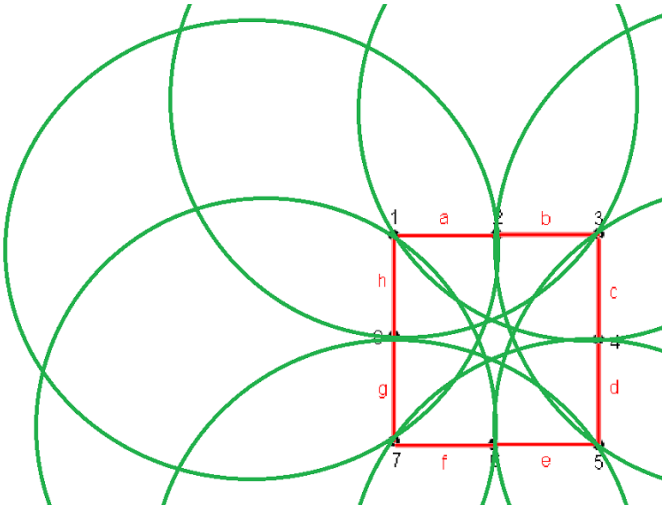

Fig. 4. Yields the non-ideal covering matrix $\mathcal{C}_8^3$.

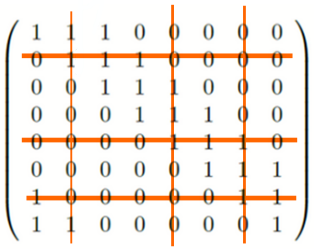

Fig. 5. $\mathcal{C}_8^3$ is not balanced.

G is equivalently an ordinary minimum-weight vertex covering problem on G, with vertices in $\mathcal{V}(G) \setminus N$ disallowed. Choosing a vertex x as a covering vertex fully r-covers all of its incident edges (because $r(x) \geq 1$), but it will not r-cover the midpoint of any other edge (because $r(x) < \sqrt[d]{5/4}$, which is the minimum distance between x and the midpoint of an edge that is not incident to x). The only way to r-cover a midpoint of an edge is to choose one of its endpoints, in which case the entire edge is r-covered (because $r(x) \geq 1$).

The efficient solvability easily follows, because unit-grid graphs are bipartite, and the ordinary formulation of minimum-weight vertex covering (with variables corresponding to vertices in $\mathcal{V}(G) \setminus N$ set to 0) has a totally-unimodular constraint matrix; so the SCP is efficiently solved by linear optimization.

3.2 When G is a path intersecting each ball on a subpath

When G is a *straight* path (in any dimension $d \geq 1$), ordering the subintervals naturally, as we traverse the path, we can see that in this case the constraint matrix for SCP is a (column-wise) consecutive-ones matrix, and hence is totally unimodular (and so SCP is polynomially solvable in such cases). In fact, this is true as long as the path (not necessarily straight) intersects $B(x, r(x))$ on a subpath, for each $x \in N$. For example, if G is monotone in each coordinate, then G enters and leaves each ball at most once each.

3.3 A fork-free set of subtrees

Given a tree T, a pair of subtrees T_1 and T_2 has a *fork* if there is a path P_1 with end-vertices in T_1 but not T_2 , and a path P_2 with end-vertices in T_2 but not T_1 , such that P_1 and P_2 have a vertex in common. We consider the problem of finding a minimum-weight covering of a tree by a given set of subtrees; see [2, called problem "C_0"]. This problem admits a polynomial-time algorithm when the family of subtrees is fork free, using some problem transformations and then a recursive algorithm. This problem and algorithm is relevant to our situation when: G is a tree, each ball intersects G on a subtree (easily checked), and the set of these subtrees is fork free (easily checked). A simple special case is the situation considered in §3.1. Much more broadly, if the degrees of a tree are ≤ 3, then any family of subtrees is fork free.

Now, it is easy to make a simple example arising from our situation (which is even a unit-grid graph), where a fork arises; see Figure 6 (The tree is the red graph, and the pair of green covering disks define the two subtrees). So

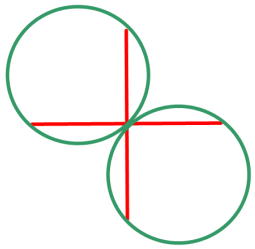

Fig. 6. A forking configuration

the algorithm from [2] does not apply to this example. On the other hand, the constraint matrix of this instance of SCP is balanced, so this is an easy instance for linear programming.

Referring back to Figure 2, we also find forks. Also see [2, Sec. 5] which raises the interesting question on the relationship between *totally*-balanced matrices[4]

[4] a 0/1 matrix is *totally balanced* if it has no square submatrix *(of any order)* with two ones per row and per column, thus a subclass of balanced 0/1 matrices; see [11].

and covering matrices of fork-free families. These notions are incomparable; [2] has a very simple example that is fork free but not *totally* balanced. But we can even get fork free coming from our context and not balanced, returning again to our example of Figure 4 (it is not a tree, but we can break an edge).

4 Strong fixing

We consider the linear relaxation of SCP, that is $\min\{w^\mathsf{T} z \ : \ Az \geq \mathbf{e},\ z \geq 0\}$, and the associated dual problem

$$\max\{u^\mathsf{T}\mathbf{e} \ : \ u^\mathsf{T} A \leq w^\mathsf{T},\ u \geq 0\}. \tag{D}$$

An optimal solution of D is commonly used in the application of *reduced-cost fixing*, see, e.g. [7], a classical technique in integer programming that uses upper bounds on the optimal solution values of minimization problems for inferring variables whose values can be fixed while preserving the optimal solutions. The well-known technique is based on Theorem 1 (a general theorem, which we state specifically for our situation).

Theorem 1. *Let* UB *be the objective-function value of a feasible solution for SCP, and let* $\hat{u}$ *be a feasible solution for D. Then, for every optimal solution* $z^\star$ *for SCP, we have:*

$$z_j^\star \leq \left\lfloor \frac{\mathrm{UB} - \hat{u}^\mathsf{T}\mathbf{e}}{w_j - \hat{u}^\mathsf{T} A_{\cdot j}} \right\rfloor, \quad \forall\, j \in \{1,\ldots,n\} \text{ such that } w_j - \hat{u}^\mathsf{T} A_{\cdot j} > 0. \tag{1}$$

For a given $j \in \{1,\ldots,n\}$, we should have the right-hand side in (1) equal to zero to be able to fix the variable z_j at zero in SCP. Equivalently, we should have $\hat{u}^\mathsf{T}(\mathbf{e} - A_{\cdot j}) > \mathrm{UB} - w_j$.

We observe that any feasible solution $\hat{u}$ can be used in (1). Then, for all $j \in \{1,\ldots,n\}$, we propose the solution of

$$\mathfrak{z}_j := \max\{u^\mathsf{T}(\mathbf{e} - A_{\cdot j}) \ : \ u^\mathsf{T} A \leq w^\mathsf{T},\ u \geq 0\}. \tag{F$_j$}$$

Note that, for each $j \in \{1,\ldots,n\}$, if there is a feasible solution $\hat{u}$ to D that can be used in (1) to fix z_j at zero, then the optimal solution of F$_j$ has objective value greater than $\mathrm{UB} - w_j$ and can be used as well.

We call *strong fixing*, the procedure that, for a given upper bound UB on the optimal value of SCP, fixes the maximum number of variables in SCP at zero, by solving problems F$_j$ and applying Theorem 1, for all $j \in \{1,\ldots,n\}$. In fact, [1] (and probably many others) considered this approach (they call a model "relaxed consistent" after no variable fixing of this type is possible) but discarded the idea as probably being computationally prohibitive.

5 Computational experiments

We have implemented a framework to generate 25 random instances of CP and formulate them as the set covering problem SCP. We solve the instances applying the following procedures in the given order.

(a) Reduce the number of constraints in SCP by eliminating dominated rows of the associated constraint matrix A.
(b) Fix variables in the reduced SCP by applying *reduced-cost fixing*, i.e., using Theorem 1, taking $\hat{u}$ as an optimal solution of D. If it was possible to fix variables, reapply (a) to reduce the number of constraints further.
(c) Apply *strong fixing* (see §4). We note that the only difference between problems F_j, for $j = 1\ldots, n$, is the objective function, so we warm start the solution of each problem (except the first one solved) with the optimal solution of the previous one solved. After solving F_j for a given j we fix all possible variables using its optimal solution. Then, we select the next problem to solve, $F_{\hat{j}}$, among all the possible options associated to the unfixed variables. $F_{\hat{j}}$ is such that $A_{\cdot\hat{j}}$ is the closest column to $A_{\cdot j}$, according to the 'Jaccard similarity' (i.e. for a pair of columns, the cardinality of the intersection of the supports divided by the cardinality of the union of the supports). In case it was possible to fix variables, reapply procedure (a) to reduce the number of constraints in the remaining problem.
(d) Solve the last problem obtained with `Gurobi`.

Our implementation is in `Python`, using `Gurobi` v. 10.0.2. We ran `Gurobi` with one thread per core, default parameter settings (with the presolve option on). We ran on an 8-core machine (under Ubuntu): Intel i9-9900K CPU running at 3.60GHz, with 32 GB of memory.

In Algorithm 1, we show how we construct random instances for CP, for a given number of points n, and a given interval $[R_{\min}, R_{\max}]$ in which the covering radii for the points must be. We construct a graph $G = (V, E)$ with node-set V given by $\nu := 0.03n$ points randomly generated in the unit square in the plane, and the edges in E initially given by the edges of the minimum spanning tree (MST) of V, which guarantees connectivity of G. Finally, we compute the Delaunay triangulation of the ν points in V, and add to E the edges from the triangulation that do not belong to its convex hull. All other details of the instance generation can be seen in Algorithm 1. In Figure 7, we show an example of the MST of V and of the graph G.

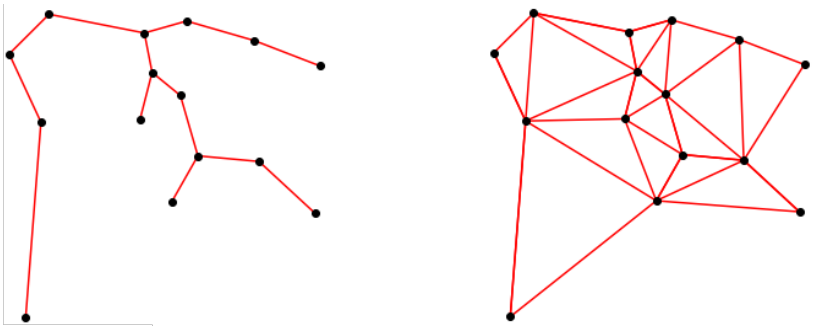

Fig. 7. Constructing an instance of CP - MST of V (left) and graph G (right)

Algorithm 1: Instance generator

Input: n, $R_{\min}$, $R_{\max}$ $(0.1 < R_{\min} < R_{\max} < 0.2)$.
Output: instance of CP/SCP.

randomly generate a set V of $\nu := 0.03n$ points in $Q := \{[0,1] \times [0,1]\}$;
use the Python package `networkx` to compute the edge set E_{ST} of an MST of V, letting the weight for edge (i,j) as the Euclidean distance between i and j;
compute the Delaunay triangulation of V, and denote the subset of its edges that are not in the convex hull by E_{DT} ;
let $G = (V, E)$, where $E := E_{\text{ST}} \cup E_{\text{DT}}$;
randomly generate a set N of n points x^k in Q, $k = 1, \ldots, n$;
randomly generate r_k in $[R_{\min}, R_{\max}]$, $k = 1, \ldots, n$;
randomly generate w_k in $[0.5r_k^2, 1.5r_k^2]$, $k = 1, \ldots, n$;
let c_k be the circle centered at x^k with radius r_k , for $k = 1, \ldots, n$;
for each $e \in E$, compute the intersections (0, 1 or 2) of c_k and e, for $k = 1, \ldots, n$;
compute all the subintervals defined on each edge by the intersections, and let m be the total number of subintervals for all edges;

We note that the instance constructed by Algorithm 1 may be infeasible, if any part of an edge of G is not covered by any point. In this case, we iteratively increase all radii r_k , for $k = 1, \ldots, n$, by 10%, until the instance is feasible. For feasible instances, we check if there are circles that do not intercept any edges of the graph. If so, we iteratively increase the radius of each of those circles by 10%, until they intercept an edge. By this last procedure, we avoid zero columns in the constraint matrices A associated to our instances of SCP. In Figure 8, we represent the data for an instance of CP and its optimal solution. In the optimal solution we see nine points/circles selected.

In Table 1 we present numerical results aiming at observing the impact of strong fixing in reducing the size of SCP, after having already applied standard reduced-cost fixing and eliminating redundant constraints. Because one of our main goals is to see the full power of strong fixing, in applying Theorem 1, we always set UB to the optimal objective value of SCP. We display the number of rows (m) and columns (n) of the constraint matrix A after applying each procedure described in (a-d) and their elapsed time (seconds). We have under '`Gurobi`' the size of the original A and the time to solve with `Gurobi`, and under '`Gurobi` presolve', the size of the matrix A after `Gurobi`'s presolve is applied and the time to apply it. Finally, we have in the next three columns the size of the matrix A after applying the procedures described in (a-c) and the times to apply them. Finally, in the last column, we have the time to solve the last problem obtained with `Gurobi`, as described in (d). We see that our own presolve, corresponding to procedures (a-c), is effective in reducing the size of the problem, and does not lead in general to problems bigger than the ones obtained with `Gurobi`'s presolve (even though, from the increase in the number of variables when compared to the original problem, we see that `Gurobi`'s presolve implements different procedures, such as sparsification on the equation system

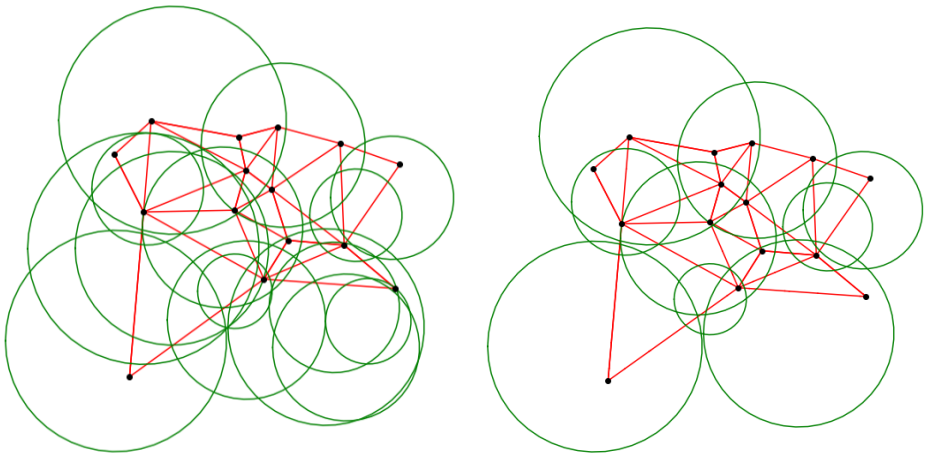

Fig. 8. Instance data (left) and its optimal solution (right)

after adding slack variables). We also see that *strong fixing* is very effective in reducing the size of the problem. Compared to the problems to which it is applied, we have an average decrease of 49% in m and 44% in n. Of course, our presolve is very time consuming compared to `Gurobi`'s. Nevertheless, for 22 out of the 25 instances tested, there is an improvement in the final `Gurobi` time to solve the problem, and we note that all the steps of reduction and fixing can still be further improved. This shows that *strong fixing* is a promising tool to be adopted in the solution of difficult problems, as is the case of the well-known *strong-branching* procedure.

We can also see that with our set-covering modeling methodology, even without strong fixing, a modern integer-programming solver is capable of solving to optimality large instances of CP in the plane. Specifically, our largest five instances all have 75 vertices and $\approx$ 273 edges, and we choose from a set of $n = 75/.03 = 2500$ points/circles.

6 Outlook

Extensions. Our integer-programming approach can be extended to other geometric settings. We can use other metrics, or replace balls $B(x, r(x))$ with arbitrary convex sets $B(x)$ for which we can compute the intersection of with each edge $I \in \mathcal{I}(G)$. In this way, N is just an index set, and we only need a pair of line searches, to determine the endpoints of the intersection of $B(x)$ with each $I \in \mathcal{I}(G)$. We still have $|\mathcal{C}(I)| \leq 1+2n$, for each $I \in \mathcal{I}(G)$, and so the number of covering constraints in CP is at most $(1+2n)|\mathcal{I}(G)|$. Finally, we could take G to be geodesically embedded on a sphere and the balls replaced by geodesic balls.

	Gurobi			Gurobi presolve			matrix reduction (a)			reduced-cost fixing (b)			strong fixing (c)			Gurobi reduced
I	m	n	time	m	n	time	m	n	time	m	n	time	m	n	time	time (d)
1	7164	500	0.28	372	257	0.17	531	500	24.49	214	167	0.50	12	19	5.39	0.003
2	4078	500	0.37	427	327	0.10	574	500	12.08	344	267	0.94	49	63	16.20	0.01
3	3865	500	0.34	399	331	0.10	537	500	10.22	342	291	0.83	46	71	17.90	0.007
4	4399	500	0.27	412	300	0.10	559	500	13.15	310	236	0.79	6	14	11.35	0.007
5	4098	500	0.22	446	359	0.13	546	500	11.92	146	132	0.35	16	16	2.33	0.008
6	13254	1000	2.90	1279	1119	0.73	1728	1000	136.00	1331	705	15.92	244	202	96.34	0.035
7	9709	1000	3.91	1263	1125	0.77	1581	1000	81.86	1443	837	13.33	909	562	140.72	2.29
8	10847	1000	2.53	1307	1140	0.80	1672	1000	94.13	1351	712	15.22	263	196	94.20	0.06
9	9720	1000	2.81	1335	1149	0.69	1700	1000	84.27	1247	667	12.14	209	193	89.13	0.03
10	9377	1000	3.20	1300	1193	0.67	1621	1000	75.76	1329	764	12.17	322	246	103.94	0.08
11	17970	1500	29.46	2720	2222	2.75	3175	1500	278.33	2862	1304	62.73	1061	609	486.60	2.31
12	22022	1500	65.65	2838	2208	3.01	3316	1500	425.03	3115	1336	68.69	1725	844	595.06	9.24
13	18716	1500	42.05	2944	2219	2.86	3466	1500	315.96	3311	1379	82.08	2184	983	619.66	24.44
14	18464	1500	14.30	2545	1988	2.44	3114	1500	306.49	2856	1270	59.32	1548	762	441.53	6.17
15	19053	1500	33.95	2908	2207	2.82	3376	1500	327.13	3162	1361	72.56	1745	851	581.55	7.57
16	29227	2000	170.78	4194	3037	6.59	4988	2000	832.81	4691	1793	177.51	2959	1221	1493.63	83.46
17	29206	2000	212.07	4365	3186	7.04	4993	2000	823.83	4793	1828	188.71	3182	1318	1547.75	102.57
18	28927	2000	762.26	4120	2971	7.22	5016	2000	799.22	4889	1862	192.64	4060	1557	1717.11	497.09
19	30746	2000	6970.90	4943	3645	7.22	5243	2000	842.61	5243	1998	229.23	5109	1944	2879.75	7421.27
20	31606	2000	1914.49	5158	3711	8.20	5378	2000	885.00	5374	1997	241.15	5088	1897	2924.22	1034.66
21	42137	2500	30111.51	6928	4656	14.99	7606	2500	1785.99	7580	2470	517.05	7308	2356	7588.82	28318.53
22	41229	2500	594.63	6511	4454	15.07	7454	2500	1689.45	7127	2344	432.95	3979	1450	4391.32	142.18
23	43100	2500	614.62	6236	4209	15.16	7381	2500	1856.51	7139	2313	462.52	5477	1828	4212.32	494.37
24	42053	2500	51978.94	6654	4776	16.84	7031	2500	1685.95	7031	2500	433.37	6958	2467	7493.67	56012.21
25	42641	2500	3594.00	6639	4723	16.43	7038	2500	1615.20	7032	2497	412.83	6604	2360	6135.45	6540.81

Table 1. Numerical results

Solvable cases. We may seek to generalize the algorithm mentioned in §3.3 to arbitrary families of subtrees of unit-grid-graph trees (i.e., get rid of the fork-free condition but restrict to unit-grid graphs — which can have degree-4 vertices).

Extending our computational work. An important direction is to reduce the time for strong fixing, so as to get a large number of variables fixed without solving all of the F_j. Additionally, the strong-fixing methodology is very general and could work well for other specific classes of mixed-integer optimization problems.

Acknowledgments. This work was supported by NSF grant DMS-1929284 while C. D'Ambrosio, M. Fampa and J. Lee were in residence at ICERM (the Institute for Computational and Experimental Research in Mathematics; Providence, RI) during the Discrete Optimization program, 2023. C. D'Ambrosio was supported by the Chair "Integrated Urban Mobility", backed by L'X - École Polytechnique and La Fondation de l'École Polytechnique (The Partners of the Chair accept no liability related to this publication, for which the chair holder is solely liable). M. Fampa was supported by CNPq grant 307167/2022-4. J. Lee was supported by the Gaspard Monge Visiting Professor Program, École Polytechnique, and from ONR grant N00014-21-1-2135. F. Sinnecker was supported on a masters scholarship from CNPq. J. Lee and M. Fampa acknowledge (i) interesting conversations at ICERM with Zhongzhu Chen on variable fixing, and (ii) some helpful information from Tobias Achterberg on `Gurobi` presolve.

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Bajgiran, O.S., Cire, A.A., Rousseau, L.M.: A first look at picking dual variables for maximizing reduced cost fixing. In: Salvagnin, D., Lombardi, M. (eds.) Integration of AI and OR Techniques in Constraint Programming. pp. 221–228. Springer International Publishing (2017), https://doi.org/10.1007/978-3-319-59776-8_18
2. Bárány, I., Edmonds, J., Wolsey, L.A.: Packing and covering a tree by subtrees. Combinatorica **6**(3), 221–233 (1986), https://doi.org/10.1007/BF02579383
3. Berge, C.: Balanced matrices. Mathematical Programming **2**(1), 19–31 (1972), https://doi.org/10.1007/BF01584535
4. Cerulli, M., D'Ambrosio, C., Liberti, L., Pelegrín, M.: Detecting and solving aircraft conflicts using bilevel programming. Journal of Global Optimization **81**(2), 529–557 (2021), https://doi.org/10.1007/s10898-021-00997-1
5. Cornuéjols, G.: Combinatorial Optimization: Packing and Covering. SIAM (2001), https://doi.org/10.1137/1.9780898717105
6. Cornuéjols, G., Novick, B.: Ideal 0,1 matrices. Journal of Combinatorial Theory, Series B **60**(1), 145–157 (1994), https://doi.org/10.1006/jctb.1994.1009
7. Crowder, H., Johnson, E.L., Padberg, M.: Solving large-scale zero-one linear programming problems. Operations Research **31**(5), 803–834 (1983), https://doi.org/10.1287/opre.31.5.803
8. FAA: Urban Air Mobility (UAM) Concept of Operations, version 2.0 (2023), https://www.faa.gov/air-taxis/uam_blueprint
9. Fowler, R.J., Paterson, M.S., Tanimoto, S.L.: Optimal packing and covering in the plane are NP-complete. Information Processing Letters **12**(3), 133–137 (1981), https://doi.org/10.1016/0020-0190(81)90111-3
10. Fulkerson, D.R., Hoffman, A.J., Oppenheim, R.: On balanced matrices. In: Pivoting and Extensions, M.L. Balinski, ed., *Mathematical Programming Studies,* 1:120–132 (1974), https://doi.org/10.1007/BFb0121244
11. Hoffman, A.J., Kolen, A.W., Sakarovitch, M.: Totally-balanced and greedy matrices. SIAM Journal on Algebraic and Discrete Methods **6**(4), 721–730 (1985), https://doi.org/10.1137/0606070
12. NASA: UAM Vision Concept of Operations (ConOps) UAM Maturity Level (UML) 4 (2021), https://ntrs.nasa.gov/citations/20205011091
13. Pelegrín, M., D'Ambrosio, C.: Aircraft deconfliction via mathematical programming: Review and insights. Transportation Science **56**(1), 118–140 (2022), https://doi.org/10.1287/trsc.2021.1056
14. Pelegrín, M., D'Ambrosio, C., Delmas, R., Hamadi, Y.: Urban air mobility: from complex tactical conflict resolution to network design and fairness insights. Optimization Methods and Software **38**(6), 1311–1343 (2023), https://doi.org/10.1080/10556788.2023.2241148
15. Pelegrín, M., Xu, L.: Continuous covering on networks: Improved mixed integer programming formulations. Omega **117**, 102835 (2023), https://doi.org/10.1016/j.omega.2023.102835
16. Schrijver, A.: Theory of Linear and Integer Programming. Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, Ltd., Chichester (1986), ISBN: 0-471-90854-1
17. Serhal, K., Macias, J.J.E., Angeloudis, P.: Demand modelling and optimal vertiport placement for airport-purposed eVTOL services (2021), https://optimization-online.org/?p=18217

18. Villa, A.: Hub Location Problem For Optimal Vertiport Selection In Urban Air Mobility — Chicago Case Study (2020), Masters Thesis, Politecnico di Milano, https://doi.org/10.25417/uic.22226494.v1
19. Xu, L.: Optimal location of safety landing sites. Research Report, LIX, École Polytechnique (2021), https://hal.science/hal-03286640