Mathematical programming algorithms for spatial cloaking

Alberto Ceselli, Maria Luisa Damiani, Giovanni Righini, Diego Valorsi
University of Milan, Dept. of Computer Science

June 23, 2015

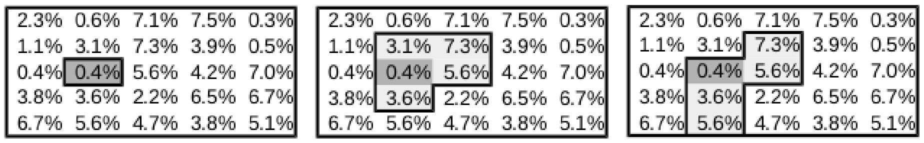
Abstract

We consider a combinatorial optimization problem for spatial information cloaking. The problem requires to compute one or several disjoint arborescences on a graph from a predetermined root or subset of candidate roots, so that the number of vertices in the arborescences is minimized but a given threshold on the overall weight associated with the vertices in each arborescence is reached. For a single arborescence case, we solve the problem to optimality by designing a branch-and-cut exact algorithm. Then we use the same algorithm for the purpose of pricing out columns in an exact branch-and-price algorithm for the multi-arborescence version. We also propose a branch-and-price-based heuristic algorithm, where branching and pricing respectively act as diversification and intensification mechanisms. The heuristic consistently finds optimal or near-optimal solutions within a computing time which can be three to four orders of magnitude smaller than that required for exact optimization. From an application point of view, our computational results are useful to calibrate the values of relevant parameters, determining the obfuscation level that is achieved.

Keywords: Steiner trees, branch-and-price, branch-and-cut.

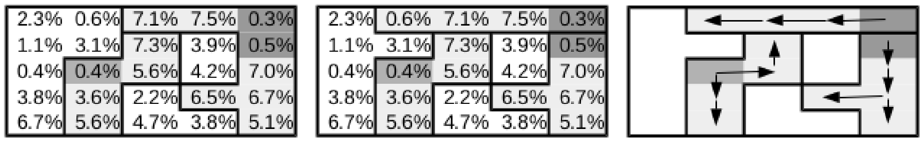
1 Problem definition

The diffusion of geo-social networking applications for mobile devices (e.g. Google Latitude, Geo-Twitter) originates relevant problems concerning users' privacy protection. One possible way of dealing with the issue is *cloaking*: the user selects some locations as *sensible* and when he is close to a sensible location, the information about his position is purposely made imprecise. Spatial cloaking is used, for instance, in the PROBE system [2] [3]: the area under study is represented by a grid, and every cell of the grid has an associated probability of the user being in it. The user is allowed to select some non-overlapping and connected subsets of cells, called *base subsets*, as sensible; when the user is located close to a base cell, the information about its position is cloaked, i.e. only the information about a region around it is made available. The region includes



(a) (b) (c)

Figure 1: Sample single base cell cloaking instance.



(a) Solution using 2 regions. (b) Solution using 3 regions. (c) Graph representation.

Figure 2: Sample multi base cell cloaking instance.

the base cell and the cell where the user is located but also other cells. The obfuscation level obtained in this way is measured by a sensitivity measure of the regions, as formally defined below, that estimates the probability of detecting the true position of the user.

In its simplest version, cloaking is applied to each single base cell independently, and the goal is to compute a region of limited size around it, whose sensitivity does not exceed a given threshold. In the most general version, instead, many base cells are given at once, adjacent ones forming base subsets. Each base subset must be included into a properly selected region, avoiding overlaps between regions.

In Figure 1a a sample 5x5 area is depicted, representing a single base cell cloaking instance. The numbers on each cell represent the probability of a user being in it; a sensible location is marked in dark gray. In Figures 1b and 1c two possible cloaking solutions are reported: when the user approaches the sensible location, its position is obfuscated be returning only the region delimited with black borders. Intuitively, the solution in Figure 1c offers more protection, as the probability of a user being in the sensible location, knowing he is in the region, is smaller. In Figures 2a and 2b, instead, a multi-base cell instance is depicted. Being adjacent, the two base cells at top right form a single base subset. Possible cloaking solutions are represented by the regions identified by black borders: two and three regions are depicted in 2a and 2b, respectively.

1.1 Modeling

Let $G = (N, \mathcal{E})$ be a 4-connected graph, representing the area grid. For each cell $i \in N$, a probability p_i is given for a user being in it. Without loss of generality we assume $\sum_{i \in N} p_i = 1$.

Sensitivity of the regions. We define the *sensitivity* $\sigma(R)$ of any region $R \subseteq N$ as the conditional probability of a user being in a base cell, given he is in the region; that is:

$$\sigma(R) = \begin{cases} \dfrac{\sum_{i\in R} s_i p_i}{\sum_{i\in R} p_i} & \text{if } \sum_{i\in R} p_i \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where the binary datum s_i indicates whether cell $i \in N$ is a base cell or not. The sensitivity of a region R represents the cloaking level that obfuscates the base cells in R. Smaller values of $\sigma(R)$ correspond to higher obfuscation levels. A region with no base cells has null sensitivity.

We denote as $\tau \in (0, 1]$ the sensibility threshold. For each region we formulate a corresponding sensitivity constraint as follows:

$$\sigma(R) \leq \tau$$

$$\frac{\sum_{i\in R} s_i p_i}{\sum_{i\in R} p_i} \leq \tau$$

which is always well-defined because $\sum_{i\in R} p_i$ is always strictly positive for each region containing at least one base cell. Now we can rewrite the constraint as

$$\sum_{i\in R} s_i p_i \leq \tau \sum_{i\in R} p_i$$

and

$$\sum_{i\in R} (s_i - \tau) p_i \leq 0.$$

We associate a binary variable x_i with each cell $i \in N$, indicating whether cell i belongs to a selected region or not, and we rewrite the sensitivity constraint as

$$\sum_{i\in N} (s_i - \tau) p_i x_i \leq 0.$$

To further simplify the notation we introduce a coefficient $w_i = (\tau - s_i)p_i$ for each cell $i \in N$. The sensitivity constraint is now

$$\sum_{i\in N} w_i x_i \geq 0$$

for each region. We note that since $s_i \in \{0, 1\}$ and $\tau \in (0, 1]$, $w_i \leq 0$ for each base cell, and $w_i > 0$ for any other cell. In the remainder we denote as "favorable" and "unfavorable" the cells with positive and negative weight respectively, because including them in $\mathcal{R}$ helps satisfying the sensitivity constraint or makes it more difficult.

In a feasible solution sensitivity constraints must be respected for each base cell. For instance, by fixing $\tau = 1.99\%$, the solution in Figure 1b is infeasible, while that of 1c is feasible, having respectively $\sigma(R) = 2.00\%$ and $\sigma(R) = 1.78\%$.

Topology and cost of the regions. To guarantee that the selected regions are connected, we represent them as arborescences in $G = (N, \mathcal{E})$, rooted at base cells. Regions must not overlap, and therefore arborescences must be disjoint. Furthermore, cells in the same base subsets can be split in different regions, but no base cells of different base subsets can belong to the same region, and therefore no arborescence is allowed to connect vertices of different base sets. We also make the assumption that portions of the same base subset belonging to the same region must be adjacent. Our aim is to design regions which are as small as possible, and therefore we define as cost of each arborescence the number of its vertices, that is the number of cells in the corresponding region. A solution is therefore a collection of arborescences; appealing solutions optimize the tradeoff between number of regions and their sum of costs. For instance, in Figure 2c, the arborescences corresponding to the solution of Figure 2b are depicted.

Related literature. Related problems deal with the computation of optimal arborescences in graphs. The Steiner Tree/Arborescence Problem (SP) is a well-known NP-hard optimization problem, for which several exact (see [4, 5]) and heuristic algorithms have been devised: for a recent review we refer to [6].

A useful starting point for the design of an algorithm for the single-base-cell problem is the paper by Ljubic et al. [7], that addresses the problem of computing a Steiner tree on a graph with costs on the edges and prizes on the vertices, where the objective function is to minimize the sum of the overall cost of the edges belonging to the tree and the overall prize of the vertices not belonging to the tree. This problem is called Prize-Collecting Steiner Tree Problem (PC-STP). Our single-base-cell problem turns out to be a PCSTP with some special characteristics: (i) all edges have unit cost (and this induces the existence of a lot of equivalent solutions); (ii) all vertices of the graph are Steiner vertices; (iii) we do not have a mixed objective function, but we have an additional constraint on the sensitivity of the region (arborescence): edge costs are in the objective, while vertex prizes are in the sensitivity constraint; (iv) vertex prizes can be positive or negative, according to the contribution of the vertices to the sensitivity constraint: base cells are unfavorable, i.e. their inclusion increases the sensitivity of the region; (v) the region is represented by an arborescence rooted at a base cell.

In this paper we face the optimal cloaking problem with a mathematical programming approach. We first consider the single base cell problem (Section 2), presenting an Integer Linear Programming (ILP) formulation with an exponential number of constraints, and providing an exact branch-and-cut algorithm. Then, we tackle the general multi base cell version, introducing an extended ILP formulation, and designing a branch-and-price algorithm exploiting an adaptation of the single base cell branch-and-cut for pricing (Section 3). Computational results are presented and discussed independently for the two versions of the problem, and conclusions are drawn in Section 4.

2 The single base-cell problem

In the single-base-cell problem a base cell corresponding to a single node $b \in N$ is given and the goal is therefore to compute a minimum cost arborescence rooted at b, complying with the sensitivity constraint.

2.1 Formulation: a 0-1 ILP model

To obtain a provably optimal solution we use a mathematical programming model based on the following notation. We consider the general setting in which $\mathcal{G} = (N, \mathcal{E})$ is a graph, not necessarily a grid, with vertex set N and edge set $\mathcal{E}$. We transform the graph into an equivalent digraph by replacing each edge $[u, v] \in \mathcal{E}$ with a pair of arcs (u, v) and (v, u) and we indicate by $\mathcal{A}$ the resulting arc set. We indicate by $\delta^-(S)$ the set of arcs entering any vertex subset $S \subset \mathcal{V}$, i.e. $\delta^-(S) = \{(u, v) \in \mathcal{A} : u \in \bar{S}, v \in S\}$, where $\bar{S} = N \backslash S$. We indicate the vertex corresponding to the base-cell by r and the weight of each vertex $v \in N$ by w_v. We use binary variables x_v associated with each vertex $v \in N$ to indicate whether the vertex belongs to the region or not; we also use binary variables y_{uv} associated with each arc $(u, v) \in \mathcal{A}$ to indicate whether u is the predecessor of v along the path from r to v.

$$\text{minimize } z = \sum_{(u,v)\in\mathcal{A}} y_{uv} \tag{1}$$

$$\text{subject to } \sum_{u\in N:(u,v)\in\mathcal{A}} y_{uv} = x_v \qquad \forall v \in N\backslash\{r\} \tag{2}$$

$$\sum_{v\in N} w_v x_v \geq 0 \tag{3}$$

$$\sum_{(u,v)\in\delta^-(S)} y_{uv} \geq x_s \qquad \forall S \subset N\backslash\{r\},\ \forall s \in S \tag{4}$$

$$x_r = 1 \tag{5}$$

$$y_{uv} \in \{0, 1\} \qquad \forall (u, v) \in \mathcal{A} \tag{6}$$

$$x_v \in \{0, 1\} \qquad \forall v \in N. \tag{7}$$

The objective function (1) requires to minimize the cardinality of the region; constraints (2) state the relationship between variables x and variables y: a vertex v belongs to the region (i.e. $x_v = 1$) if and only if it is the head of a selected arc (i.e. $y_{uv} = 1$ for some vertex u); constraints (3) impose that the overall weight of the region complies with the sensitivity constraint; constraints (4) impose that the region is connected; constraints (5) impose that the region includes the base-cell; finally constraints (6) and (7) require the integrality of the variables.

This model resembles the one proposed by Ljubic et al. [7] for the prize-collecting Steiner tree problem; our adaptations cope with the five main differences outlined in the previous Section. The algorithm we have designed and tested is also based on a relaxation adapted from [7]: we initially relax constraints (4) and integrality restrictions (6) and (7), we solve the resulting linear programming (LP) problem and we detect violated constraints (4) by a separation algorithm; some violated constraints are then inserted into the LP model and the procedure is iterated until the optimal LP solution satisfies all constraints (4).

2.2 Relaxation: an LP model

The model of the linear relaxation we iteratively solve in our branch-and-cut algorithm is the following:

$$\text{minimize } z = \sum_{(u,v)\in\mathcal{A}} y_{uv} \tag{8}$$

$$\text{subject to } \sum_{u\in N:(u,v)\in\mathcal{A}} y_{uv} = x_v \qquad \forall v \in \mathcal{V}\setminus\{r\} \tag{9}$$

$$\sum_{v\in N} w_v x_v \geq 0 \tag{10}$$

$$\sum_{(u,v)\in\delta^-(S)} y_{uv} \geq x_s \qquad \forall S \in \mathcal{S} \forall s \in S \tag{11}$$

$$x_r = 1 \tag{12}$$

$$x_u \geq y_{uv} \qquad \forall (u,v) \in \mathcal{A} \tag{13}$$

$$\sum_{(r,v)\in\mathcal{A}} y_{rv} \geq 1 \tag{14}$$

$$2y_{uv} + 2y_{vu} \leq x_u + x_v \qquad \forall (u,v) \in \mathcal{A} \tag{15}$$

$$0 \leq y_{uv} \leq 1 \qquad \forall (u,v) \in \mathcal{A} \tag{16}$$

$$0 \leq x_v \leq 1 \qquad \forall v \in N. \tag{17}$$

The set $\mathcal{S}$ of connectivity constraints is initially empty. This model is obtained from the linear relaxation of model (1)-(7) by relaxing connectivity constraints (4), which are exponential in number, and by inserting constraints (13), (14) and (15) to reinforce the formulation. Constraints (13) state that an arc can be used only if the node corresponding to its tail belongs to the region. Constraints (14) state that at least one arc must leave the root. Constraints (15) forbid cycles of cardinality 2.

Since connectivity constraints (4) have been relaxed, it is possible that the optimal solution of model (8)-(17) is disconnected and contains cycles. Violated connectivity constraints (11) are dynamically generated by an exact separation algorithm illustrated in the remainder. According to our notation, they are inserted in $\mathcal{S}$. However, owing to the relaxation of the integrality requirements (6) and (7), the optimal solution of the linear program with connectivity constraints is not guaranteed to be integer and acyclic. For this reason we finally resort to branching, when necessary.

2.3 Preprocessing

Before starting the branch-and-cut algorithm we run a pre-processing routine, whose goal is to identify and discard some vertices of the graph that cannot be part of any optimal solution, thus reducing its size. This goal is achieved by visiting the graph with a breadth-first-search algorithm, starting from the root vertex r. Every vertex u is labeled with the maximum weight of a path from r to u; the weight of a path is the sum of the weights of the vertices along the path. The weight of r is negative, because r corresponds to a base-cell; as soon as the weight of a path P emanating from r becomes non-negative, the algorithm is stopped: a feasible solution has been detected and the number of arcs along P is a valid upper bound. Therefore all vertices whose distance from r is larger than P cannot belong to any optimal solution and are discarded.

2.4 Separation

The separation problem is a min-cut problem, which can be solved to optimality in polynomial-time. We indicate with x^* and y^* the values of the x and y variables in the current optimal solution of model (8)-(17). For each vertex $u \in N$ with $x^*_u > 0$ we search for a minimum cut separating r from u in the support digraph where each arc $(u, v) \in \mathcal{A}$ has capacity equal to y^*_{uv}. We indicate by C_S the capacity of a (r, u)-cut separating $S \subset N$ from its complement $\overline{S} \subset N$, with $u \in S$ and $r \in \overline{S}$. Any cut S with $C(S) < x^*_u$ corresponds to a violated connectivity constraint; with a little abuse of terminology, we call it *violated cut* in the remainder.

To compute minimum cuts we used the efficient algorithm by Cherkassky and Golberg [1]. However it is important to remark that there are several possible strategies that can be used in the cut generation process. It is common experience that adding more than one constraint per iteration usually speeds up cutting planes algorithms. Therefore, for each destination vertex $u \in N$, it is convenient to find multiple violated connectivity constraints, i.e. multiple violated (r, u)-cuts. For this purpose we also generated *nested cuts* and *back cuts* [7] and we made computational tests to suitably tune the separation strategy.

Nested cuts are obtained as follows. Assume a violate cut has been found; then all arcs belonging to the cut are given capacity equal to 1, so that they cannot belong to a bottleneck cut anymore; then the max-flow-min-cut algorithm is run again and another violated cut is possibly found. The procedure is iterated and it generates a sequence of violated cuts, initially close to r and then closer and closer to u. When the same technique is executed sending the flow from u to r, *back (nested) cuts* are generated.

We compared four separation strategies, all generating nested cuts. In strategies A and B the choice between forward and backward nested cuts is made at random with the same probability at each iteration and for each (r, u) pair; on the contrary, in strategies C and D nested cuts are always generated both forward and backward. In strategies A and C the generation of cuts stops as soon as the connected component of the target vertex is reached, while in strategies B and D the cut generation procedure stops only when the target vertex is reached (by target vertex we mean u for forward cuts and r for backward cuts). The goal is to reach a (possibly fractional) optimal solution in which all connectivity constraints are satisfied as early as possible. Besides tightening the lower bound, this also allows to run heuristics at each node of the branch-and-bound tree and in turn this allows to early compute good feasible solutions and to effectively prune the search.

To fine tune our algorithm, we tested all strategies on a set of instances corresponding to 25×25 grids in this way: a node u is selected at random; cuts are generated with one of the four strategies and inserted in the relaxed model; the linear program is re-optimized; the procedure is repeated until a time-out of 200 seconds expires. A typical outcome is represented in Figures 3 and 4. Strategy C turns out to be the most effective in tightening the lower bound.

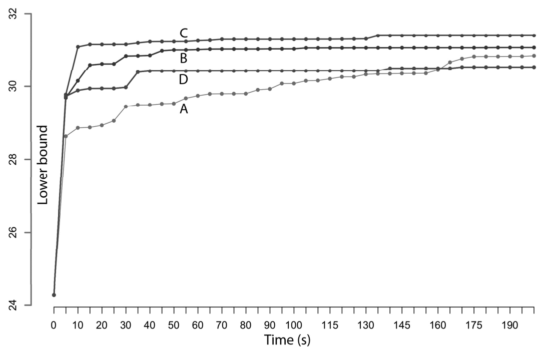


Figure 3: Comparison between the four separation strategies: lower bound as a function of computing time.

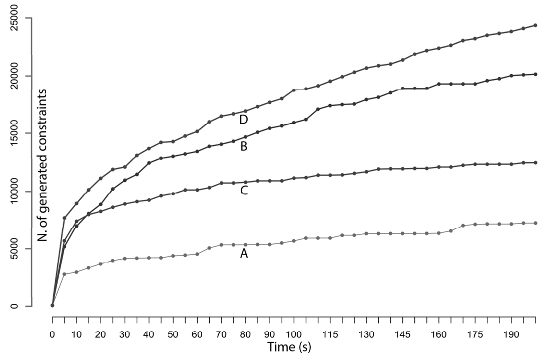


Figure 4: Comparison between the four separation strategies: number of generated constraints as a function of computing time.

The same results are shown in Table 1.

	A	B	C	D
Lower bound	30.86	30.95	31.57	30.82
N. cuts	7188	20073	12364	24256

Table 1: Final lower bound and number of cuts with different separation strategies.

From computational experiments on larger instances we could observe that the best trade-off between the number of LP iterations and the number of constraints generated depends on the instances. When instances are easier (i.e. smaller), strategies B and D which generate a larger number of cuts, are faster. When the instance is more difficult, it is convenient to generate less constraints and to optimize the LP more often, with strategy C. Computational results reported in the remainder have been obtained with strategy C.

2.5 Heuristics

At every node of the branch-and-bound tree we run a fast heuristic algorithm, when the cut generation procedure is over. The algorithm works on the subgraph $(N^*, \mathcal{A}^*)$ defined by vertices $u \in N$ with $x^*_u > 0$ and the arcs $(u, v) \in \mathcal{A}$ with $y^*_{uv} > 0$. This support graph is guaranteed to be connected but it may not correspond to an arborescence. If any node in N^* turns out to be the head of more than one arc in $\mathcal{A}^*$, then we arbitrarily choose a path from r to u in the support graph and we delete from $\mathcal{A}^*$ all arcs entering u and not belonging to the path. After this preprocessing every non-root vertex u has a unique predecessor $\pi(u)$ and the support graph is an r-arborescence. Since the sensitivity constraint is satisfied, we are guaranteed that $\sum_{u \in N} w_u x^*_u >= 0$ and hence $\sum_{u \in N^*} w_u >= 0$. We formulate the problem of selecting a minimum cardinality r-arborescence in $(N^*, \mathcal{A}^*)$ still complying with the sensitivity constraint. The corresponding ILP model requires a binary variable t_{uv} for each arc in $\mathcal{A}^*$ and reads as follows.

$$\begin{aligned} \text{minimize} \quad & \sum_{(u,v) \in \mathcal{A}^*} t_{uv} \\ \text{subject to} \quad & w_r + \sum_{(u,v) \in \mathcal{A}^*} w_v t_{uv} \geq 0 \\ & t_{uv} \leq t_{\pi(u)u} && \forall (u,v) \in \mathcal{A}^* : u \neq r \\ & t_{uv} \in \{0, 1\} && \forall (u,v) \in \mathcal{A}^*. \end{aligned}$$

Preliminary experiments verified that the solution of this model with state-of-the-art general purpose solvers takes negligible computing time.

2.6 Policies for branching and searching

We included a binary branching policy: we consider the last optimal solution of the linear program obtained when the cut generation procedure is over and we select the x variable whose current optimal value is closest to 1/2. Then we create two subproblems by fixing the variable respectively to 0 and to 1.

We compared four tree exploration policies: best-first, depth-first, breadth-first and a mixed strategy. This latter one is motivated by the observation that fixing a variable to 1 is more effective than fixing it to 0. Therefore sub-problems generated by fixings to 1 (1-nodes) are given precedence with respect to those originated from fixings to 0 (0-nodes): every time a branch occurs the 1-node is inserted in the first position in the queue of open nodes while the 0-node is appended in the last position. From our preliminary tests, we observed that best-first tends to be the most effective strategy and this was selected for the final extensive set of computational tests.

2.7 Computational results

We defined two data-sets, identified by A and B in the remainder, reproducing areas with a single sensible location or multiple sensible locations respectively. Data were generated at random as follows.

- The grid is a square with size ranging from 15×15 to 25×25; its cells are tentatively populated with the probability values p given by a a bivariate Gaussian distribution, centered in a cell chosen at random with uniform probability distribution in the grid.

- Cells are iteratively selected at random and their corresponding tentative values of probability p are confirmed. When 70% of the cells is associated with a confirmed positive value of the probability, the other cells receive zero probability. The sum of confirmed values is normalized to 1.

- In data-set A the base cell is chosen at random among the cells with zero probability. In data-set B we generate at random a rectangular *base subset* with size 2×2 (data-set B1) or 4×4 (data-set B2) containing only cells with positive probability. The root base cell is then fixed by choosing uniformly at random among the base cells.

- Each instance in data-set A is tested with three values for τ, namely 0.3, 0.5 and 0.7. Each instance in data-sets B is tested with four values for τ, i.e. 0.05, 0.1, 0.2 and 0.4.

As a result, data-set A is characterized by vertices with non-negative weights $w_i = p_i$ only, while vertices in the base subsets of data-set B can have negative weights, being $w_i = (\tau - s_i)p_i$ with $s_i = 1$ for the cells of the base subset, 0 otherwise. The sensitivity constraint for data-set A requires $\sum_{i \in N} p_i x_i \geq \tau$, while that for data-sets B requires $\sum_{i \in N} w_i x_i \geq 0$,

Remarkably many instances were solved at the root node. Instances in data-sets B are harder to solve to optimality; this can be seen for instance from the values in the "Gap" columns. The gap also tends to be larger when the base subset is larger (data-set B2). The reason is that the fractional optimal solution of the relaxed model tends to combine a lot of variables; besides enlarging the

Instance	Root node				B&C tree			
	UB	LB	Time	Cuts	UB	Gap	Time	Nodes
A-01-1-15-0.3	21	19.96	0.06	2238	21	4.76	0.39	4
A-02-1-15-0.5	36	35.94	0.05	4305	36	0.00	0.36	0
A-03-1-15-0.7	60	58.95	0.38	2759	60	1.67	1.00	2
A-04-2-15-0.3	17	15.05	1.41	3869	17	5.88	7.52	20
A-05-2-15-0.5	27	26.58	0.91	3696	27	0.00	1.22	0
A-06-2-15-0.7	42	41.17	0.68	7486	42	0.00	1.21	0
A-07-3-15-0.3	24	23.12	0.45	3155	24	0.00	1.03	0
A-08-3-15-0.5	31	29.30	0.87	5367	30	0.00	2.45	0
A-09-3-15-0.7	43	42.43	1.04	6948	43	0.00	2.11	0
A-10-1-20-0.3	25	21.07	3.72	6055	23	4.35	49.49	18
A-11-1-20-0.5	36	35.50	3.57	13579	36	0.00	5.10	0
A-12-1-20-0.7	57	56.75	15.51	34825	57	0.00	19.66	0
A-13-2-20-0.3	34	31.49	42.32	26213	33	3.03	622.57	28
A-14-2-20-0.5	55	54.50	18.14	37523	55	0.00	35.95	0
A-15-2-20-0.7	95	94.33	0.36	30778	95	0.00	12.39	0
A-16-3-20-0.3	19	18.08	0.03	1945	19	0.00	0.65	0
A-17-3-20-0.5	29	27.89	0.78	3449	28	0.00	3.45	0
A-18-3-20-0.7	67	65.43	2.45	12311	68	2.94	13.34	4
A-19-1-25-0.3	33	31.52	17.94	26423	33	3.03	835.72	30
A-20-1-25-0.5	65	64.20	4.13	42997	65	0.00	34.70	0
A-21-1-25-0.7	113	113.00	15.32	71861	113	0.00	66.35	0
A-22-2-25-0.3	37	35.65	33.69	33440	37	2.70	103.57	4
A-23-2-25-0.5	68	67.14	3.10	25822	68	0.00	21.40	0
A-24-2-25-0.7	109	109.00	0.49	69610	109	0.00	48.99	0
A-25-3-25-0.3	37	36.13	3.03	14349	37	0.00	12.67	0
A-26-3-25-0.5	51	50.17	5.44	17349	51	0.00	29.45	0
A-27-3-25-0.7	68	66.78	9.78	24511	68	1.47	67.31	8

Table 2: Results with data-set A.

primal-dual gap this also requires to generate more cutting planes. This is particularly evident when τ is small (i.e. $\tau = 0.05$) and the probability of the root vertex is relatively large, because in this case it necessary to include many vertices in the arborescence to satisfy the sensitivity constraint.

Instance	Root node				B&C tree			
	UB	LB	Time	Cuts	UB	Gap	Time	Nodes
B1-01-1-15-0.05	9	7.72	0.01	521	9	11.11	0.18	4
B1-02-1-15-0.1	7	4.67	0.03	818	7	28.57	2.36	42
B1-03-1-15-0.2	5	2.50	0.03	1015	5	40.00	53.97	84
B1-04-1-15-0.4	4	1.47	0.04	926	4	50.00	319.78	148
B1-05-2-15-0.05	16	9.48	8.86	8152	16	37.50	86.77	8
B1-06-2-15-0.1	5	4.00	0.08	1945	5	20.00	0.65	8
B1-07-2-15-0.2	4	2.43	0.13	1881	4	25.00	2.51	28
B1-08-2-15-0.4	3	1.28	0.12	1920	2	0.00	0.79	0
B1-09-1-20-0.05	12	10.63	44.22	19720	12	8.33	205.43	4
B1-10-1-20-0.1	8	5.54	35.98	12950	6	0.00	471.54	0
B1-11-1-20-0.2	2	2.00	0.01	0	2	0.00	0.02	0
B1-12-1-20-0.4	1	1.00	0.01	0	1	0.00	0.01	0
B1-13-2-20-0.05	8	7.19	0.71	7119	8	0.00	3.88	0
B1-14-2-20-0.1	4	3.15	0.14	1555	4	0.00	0.92	0
B1-15-2-20-0.2	3	2.25	0.36	1133	3	0.00	0.62	0
B1-16-2-20-0.4	2	1.31	0.27	2002	2	0.00	1.19	0
B1-17-1-25-0.05	38	32.50	5.92	19432	38	13.16	4670.56	304
B1-18-1-25-0.1	12	11.61	3.02	7175	12	0.00	6.98	0
B1-19-1-25-0.2	4	3.57	0.01	0	4	0.00	0.03	0
B1-20-1-25-0.4	1	1.00	0.02	0	1	0.00	0.03	0
B1-21-2-25-0.05	12	6.75	13.62	15319	7	0.00	355.66	0
B1-22-2-25-0.1	3	3.00	2.01	5427	3	0.00	4.45	0
B1-23-2-25-0.2	2	1.80	0.98	7372	2	0.00	5.56	0
B1-24-2-25-0.4	2	1.10	1.01	6467	2	0.00	5.12	0

Table 3: Results with data-set B1.

3 The multi-base-cell problem

As discussed in the introduction, in the most general version of cloaking many base cells can be defined in the area, forming base subsets; appealing solutions cover all base cells with many regions of limited size. However, maximizing the number of regions and minimizing their overall size are two conflicting objectives. Therefore, we tackle this generalization with a bi-objective optimization approach. For each suitable number of regions k, we search for the minimum number of cells $\mathcal{P}_k$ to be included in the regions to cover all base cells and respect sensitivity constraints. Once all $\mathcal{P}_k$ values are computed, different synthetic tradeoff measures can be selected to choose a particular solution, like the average region size $\mathcal{P}_k/k$.

The number of solution values $\mathcal{P}_k$ to compute can be in principle large, but can be substantially reduced by exploiting the following two observations.

Observation 3.1. *There is no feasible solution using less regions than the number of base subsets ℓ,*

Instance	Root node				B&C tree			
	UB	LB	Time	Cuts	UB	Gap	Time	Nodes
B2-01-1-15-0.05	12	9.13	0.12	1023	12	16.67	32.21	36
B2-02-1-15-0.1	8	5.03	0.08	956	8	25.00	9.34	22
B2-03-1-15-0.2	7	2.76	0.04	1145	6	50.00	65.45	48
B2-04-1-15-0.4	5	2.12	0.03	998	5	40.00	176.45	92
B2-05-2-15-0.05	23	13.55	15.78	12333	21	33.33	895.98	58
B2-06-2-15-0.1	9	5.34	2.23	4389	8	25.00	23.13	8
B2-07-2-15-0.2	6	3.12	0.39	2221	6	33.33	57.43	94
B2-08-2-15-0.4	4	2.62	0.23	2134	4	25.00	7.65	16
B2-09-1-20-0.05	11	9.34	13.78	13455	11	9.09	89.45	4
B2-10-1-20-0.1	7	4.12	14.31	9423	6	16.67	238.42	22
B2-11-1-20-0.2	4	3.11	0.31	764	4	0.00	12.56	0
B2-12-1-20-0.4	4	2.13	12.00	453	3	0.00	7.89	0
B2-13-2-20-0.05	8	7.56	0.56	5674	8	0.00	5.64	0
B2-14-2-20-0.1	4	3.14	0.11	1342	4	0.00	1.98	0
B2-15-2-20-0.2	3	2.02	0.09	674	3	0.00	1.54	0
B2-16-2-20-0.4	2	1.91	0.04	19	2	0.00	0.09	0
B2-17-1-25-0.05	48	39.98	7.98	20134	47	14.89	534.98	66
B2-18-1-25-0.1	27	21.32	5.98	9842	27	18.52	499.66	128
B2-19-1-25-0.2	12	8.34	2.28	2898	12	25.00	231.98	84
B2-20-1-25-0.4	4	3.24	0.89	958	4	0.00	12.23	0
B2-21-2-25-0.05	14	9.34	21.12	21656	13	23.08	756.23	28
B2-22-2-25-0.1	8	5.45	9.32	7853	7	14.29	161.79	12
B2-23-2-25-0.2	5	4.12	3.21	7664	5	0.00	98.34	0
B2-24-2-25-0.4	4	3.67	0.43	5234	4	0.00	3.72	0

Table 4: Results with data-set B2.

as no region can cover base cells of different subsets.

Observation 3.2. *When an instance is feasible, an optimal solution always exists in which the root of each arborescence lies in the border of a base subset.*

In fact, suppose to have an optimal solution in which an arborescence is rooted in an inner cell of a base subset. In order to satisfy the sensitivity constraint, the root must be connected to non-base outer cells; therefore at least one directed path exists from the root to outer cells, thereby crossing the border of the base subset in a particular base cell. Hence, an equivalent optimal solution can be obtained by arbitrarily selecting one of those paths, considering the subpath from the root to the base cell on the border, and reversing its arcs.

Observation 3.3. *For each suitable number of regions $k > \ell$, $\mathcal{P}_{k-1} \leq \mathcal{P}_k$.*

In fact, one can always obtain a solution in which $k-1$ regions are defined by a solution with k regions, by simply merging two regions with adjacent base cells. Since sensitivity constraints hold for each of them independently, they hold also after merging, being actually replaced by a single surrogate constraint. That solution might, of course, be suboptimal.

Let $\bar{\ell}$ be the number of base cells on the border of base subsets. Summarizing, we compute the Pareto-optimal frontier of the bi-objective problem by searching for all optimal $\mathcal{P}_k$ values with a bisection process: starting with $k' = \ell$ and $k'' = \bar{\ell}$, we compute $\mathcal{P}_{k'}$ and $\mathcal{P}_{k''}$; if $\mathcal{P}_{k'} = \mathcal{P}_{k''}$ then all values $\mathcal{P}_k$ with $k' \leq k \leq k''$ are equal to $\mathcal{P}'_k$, and therefore we stop; otherwise we select $k = \lfloor (k''+k')/2 \rfloor$ and we proceed recursively on the ranges $[k', k]$ and $[k, k'']$, stopping when $k = k'$.

A key issue remains on how to minimize $\mathcal{P}_k$ for a fixed k. In the following we detail how we solve it by a branch-and-price algorithm, where the pricing algorithm that generates minimum cardinality arborescences complying with the sensitivity constraint is adapted from the branch-and-cut algorithm we have presented in the previous section for the single-base-cell problem.

3.1 Mathematical formulation

As in the previous section we indicate as $G = (N, \mathcal{A})$ the graph corresponding to the grid, and as w_i the weight associated with each node $i \in \mathcal{N}$. The set of base subsets is indicated by $\mathcal{R}$; base subsets are disjoint. For each base subset $r \in \mathcal{R}$ we indicate by $\mathcal{B}_r \subseteq N$ the corresponding subset of base cells; the set of all the base cells is indicated by $\mathcal{B} = \cup_{r \in \mathcal{R}} \mathcal{B}_r$. For each base cell $b \in \mathcal{B}$ we indicate by Θ_b the set of all feasible arborescences rooted at b, that are those satisfying the sensitivity constraints without including cells of other base sets.

The union of all the subsets of columns is indicated by Θ. The required number of arborescences is indicated by k.

A binary variable λ_l is associated with each column (arborescence) $l \in \Theta$. A binary coefficient x_{ibl} indicates whether vertex $i \in N$ belongs to the arborescence rooted at vertex $b \in B$ in column $l \in \Theta_b$. With this notation the master problem reads as follows:

$$\text{minimize } z = \sum_{r \in \mathcal{R}} \sum_{b \in \mathcal{B}_r} \sum_{l \in \Theta_b} \sum_{i \in N} x_{ibl} \lambda_l \tag{18}$$

$$\text{s.t.} \quad \sum_{r \in \mathcal{R}} \sum_{b \in \mathcal{B}_r} \sum_{l \in \Theta_b} x_{ibl} \lambda_l \leq 1 \qquad \forall i \in N \tag{19}$$

$$\sum_{r \in \mathcal{R}} \sum_{b \in \mathcal{B}_r} \sum_{l \in \Theta_b} x_{ibl} \lambda_l \geq 1 \qquad \forall i \in \mathcal{B} \tag{20}$$

$$\sum_{r \in \mathcal{R}} \sum_{b \in \mathcal{B}_r} \sum_{l \in \Theta_b} \lambda_l = k \tag{21}$$

$$\lambda_l \text{ binary} \qquad \forall r \in R \ \forall b \in B_r \ \forall l \in \Theta_b \tag{22}$$

We consider the linear relaxation of this master problem, where the integrality conditions (22) are replaced by constraints $0 \leq \lambda_l \leq 1 \ \ \forall r \in R \ \forall b \in B_r \ \forall l \in \Theta_b$. Because of constraints 20 it is redundant to impose $\lambda_l \leq 1$. Hence constraints (22) are simply replaced by non-negativity conditions.

We indicate the (non-positive) dual variables of constraints (19) with μ_i, the (non-negative) dual variables of constraints (20) with γ_i and the (free) dual variable of constraint (21) with ξ. Hence the reduced cost of a column $l \in \Theta_b$ for any $b \in \mathcal{B}$ is:

$$\overline{c}_{bl} = \sum_{i \in N} x_{ibl} - \sum_{i \in N} \mu_i x_{ibl} - \sum_{i \in \mathcal{B}} \gamma_i x_{ibl} - \xi$$

that is

$$\overline{c}_{bl} = \sum_{i \in N} x_{ibl}(1 - \mu_i - \gamma_i) - \xi$$

with $\gamma_i = 0$ for all non-base cells.

Further, since the number of arborescences is exponential in the size of the graph, only a subset of them is actually inserted in a restricted master problem. In order to guarantee that the linear relaxation of the restricted master problem is always feasible, we insert a dummy column with a cost equal to $|N| + 1$ and such that it satisfies all constraints of the master problem, i.e. it covers all base cells in constraints (20) and it counts as k arborescences in constraint (21). To accelerate the convergence of the column generation algorithm we also insert columns corresponding to "empty arborescences" made by a single base cell. Since they are infeasible (a root alone cannot satisfy the sensitivity constraint), they also have a very large cost like the dummy column.

3.2 Pricing

The linear relaxation of the master problem is solved via column generation. When a fractional optimal solution is reached we resort to branching. The pricing sub-problem is solved with the branch-and-cut algorithm illustrated for the single-base-cell problem. The only difference is that in the pricing sub-problem of the multi-base-cell problem there are also penalties on the vertices: they are the current values of the dual variables μ and γ. The pricing algorithm is run for each base cell and the corresponding optimal solutions are inserted in the restricted master problem if their reduced cost is negative.

3.3 Symmetry breaking

The pre-processing procedure outlined for the single-base-cell problem is no longer applicable in the multi-base-cell case, because several arborescences may compete for the same favorable cells. Hence a feasible solution with disjoint arborescences is not guaranteed not to use paths longer than the minimum path found by the breadth-first-search algorithm.

However we pre-process the instances in a different way, to reduce the symmetries due to the different base cells in the same base subset. First, Observation 3.2 allows us to directly avoid pricing for cells in the interior of a base set. Furthermore, since we also made the assumption that no disjoint subsets of base cells belong to the same arborescence, it is sufficient to keep the arcs of a cycle along the borders of each base subset, deleting those in the opposite direction.

3.4 Greedy pricing

Before running the branch-and-cut algorithm for exact pricing, we run a greedy heuristic pricing routine: starting from the base cell the arborescence is iteratively extended with the reachable cell of minimum weight until the sensitivity constraint is satisfied. After that, the arborescence is possibly further extended with negative reduced cost cells. To speed up the search for the minimum weight cell among those that can be appended to the arborescence, we use a heap data-structure. The effectiveness of the greedy pricing algorithm in producing good solutions is well documented by the computational results of the branch-and-price-based heuristics described in subsection 3.6.

3.5 Branching

When the column generation algorithm is over and the optimal solution of the linear relaxation of the restricted master problem is fractional, we resort to branching in three different ways.

Branching 1: roots. We consider all the candidate roots, i.e. the cells along the borders of the base subsets. For each candidate root b we compute $\sum_{l \in \Theta_b} \lambda_l$ and we select the value that is closest to $1/2$. Then we generate two sub-problems with a binary branching operation. In a sub-problem we constrain b to be a root: this is achieved by deleting all arcs entering b from other base cells of the same base subset. In this way b is forbidden to be included in any arborescence rooted elsewhere and therefore it is constrained to be a root. In the other sub-problem we forbid b to be a root: all columns in Θ_b are deleted from the master problem and the pricing algorithm is no longer executed from cell b in that branch.

Branching 2: base cells. When all values of $\sum_{l \in \Theta_b} \lambda_l$ are integer, then we resort to a second branching strategy. We consider all pairs of base cells i and b in the same base subset $\mathcal{B}_r$ and we compute $\sum_{l \in \Theta_b} x_{ibl}\lambda_l$. We select the value that is closest to $1/2$ and we generate two sub-problems with a binary branching operation. In a sub-problem we forbid i to belong to an arborescence rooted at b by fixing $x_{ib} = 0$ in the pricing sub-problem for root b. In the other sub-problem we constrain i to belong to an arborescence rooted at b: this is achieved by fixing $x_{ib'} = 0$ in the pricing sub-problems corresponding to all roots $b' \neq b$.

Branching 3: non-base cells. When the values of $\sum_{l \in \Theta_b} x_{ibl}\lambda_l$ are integer for all base cells i, then we resort to a third branching strategy, which is identical to the second one but considers non-base cells. For each non-base cell i and each base cell b we compute $\sum_{l \in \Theta_b} x_{ibl}\lambda_l$. We select the value that is closest to $1/2$ and we generate two sub-problems with a binary branching operation. In a sub-problem we forbid i to belong to an arborescence rooted at b by fixing $x_{ib} = 0$ in the pricing sub-problem for root b. In the other sub-problem we forbid i to belong to any arborescence not rooted at b: this is achieved by fixing

$x_{ib'} = 0$ in the pricing sub-problems corresponding to all roots $b' \neq b$. This third branching strategy does not partition the solution space of the current sub-problem: solutions in which i is not part of any arborescence remain feasible in both branches.

3.6 A branch-and-price-based heuristics

To cope with the combinatorial explosion and the exponential increase in computing time, we also devised a heuristic algorithm to compute good solutions quickly, optimizing also instances that are out-of-reach for the exact optimization branch-and-price. The heuristic algorithm relies upon the structure of the branch-and-price algorithm where the pricing sub-problem is solved heuristically. The price is of course to lose any optimality guarantee, including that the final solution of the linear restricted master problem is a valid lower bound. Nevertheless the algorithm produces a well-diversified set of "good" columns and often achieves the optimal solution. Branching acts as a diversification strategy, forcing the search to explore all the solution space, while heuristic pricing acts as an intensification mechanism, producing feasible solutions of small cardinality. We tested two versions of this branch-and-price-based heuristics: in one case we only use the greedy pricing algorithm described in subsection 3.4; in the other case we also use the branch-and-cut algorithm described in subsection 2 truncated at the root node. When the root node is solved to optimality, the heuristic procedure described in 2.5 produces a feasible arborescence.

3.7 Computational results

The instances for testing the branch-and-price algorithm were generated with the same technique described in subsection 2.7 for data-sets B. The size of the grid was set to 15×15 when searching for optimal solutions. We used one or two base subsets of size 2×2 or one base subset of size 3×3. To offer a clear view on how selecting the number of regions affects the computing time of branch-and-price, we run a test for all suitable values of the number of arborescences k, ranging from the number of base subsets (1 or 2) to the number of base cells on the borders of the base subsets (4 or 8). We also used different values of the sensitivity threshold τ.

Tests on larger grids were also carried out with the aim of producing heuristic solutions by the branch-and-price-based heuristic algorithm. In these cases we used grid of size up to 30×30, up to 5 base subsets of size 2×2 or 3×3, resulting into a number of arborescences up to 24.

We present aggregate results in Table 5 and detailed results in the tables collected in appendix A. The first column in Table 5 defines how data have been aggregated. The we report the average computing time to solve the root node, the average computing time to obtain a heuristic solution at the root node, the average cardinality of the optimal solution (overall number of cells in the arborescences) and the average cardinality of the heuristic solution.

Aggregation criterion	Time (root)	Time (B&C)	N.cells (opt.)	N.cells (heur.)
$\tau = 0.05$	3444.35	61.43	27.14	27.29
$\tau = 0.10$	2596.50	38.36	19.57	19.64
$\tau = 0.20$	1637.79	14.64	15.36	15.36
$\tau = 0.40$	754.07	12.31	12.71	12.71
One base subset 2×2	426.75	13.25	12.31	12.38
Two base subsets 2×2	2517.72	21.05	24.94	24.94
One base subset 3×3	1746.44	68.31	19.75	19.88
$k = \|\mathcal{R}\|$	2233.50	13.63	14.13	14.13
$k \approx \|\mathcal{R}\|$	2269.78	60.11	14.58	14.58
$\|\overline{\mathcal{B}}\| < k < \|\mathcal{R}\|$	2939.94	63.13	20.56	21.43
$k \approx \|\overline{\mathcal{B}}\|$	2212.65	58.31	21.50	21.61
$k = \|\overline{\mathcal{B}}\|$	1361.08	22.50	23.08	23.17

Table 5: Aggregated results.

The first section of Table 5 shows the same effect already observed with the single-base-cell problem: as the threshold τ becomes large, the average size of the solutions tends to decrease, because few non-base cells are sufficient to meet the sensitivity requirement. As a consequence it is less likely that arborescence "compete" for using the same favorable cells and the instances become easier to solve; for $\tau = 0.20$ or larger, heuristic solutions are often optimal.

The second section of Table 5 shows the effect of the number of base subsets $|\mathcal{R}|$. When $|\mathcal{R}|$ increases there are more base cells in $\overline{\mathcal{B}}$ and, as expected, this increases the number of runs of the pricing algorithm and the computing time required by the branch-and-price algorithm. The same effect is produced by different sizes of the base subset.

The third section of Table 5 shows the effect of the number of arborescences k. We indicate by $|\mathcal{R}|$ the cardinality of the set of base subsets; this is a lower bound for k because at least one arborescence is needed for each base subset. We indicate by $|\overline{\mathcal{B}}|$ the cardinality of subset $\overline{\mathcal{B}}$ that includes the base cells on the borders of their regions; this is an upper bound for k, because w.l.o.g. roots are not allowed to be in the interior of the base subsets. It is apparent that extreme values of k, i.e. $k = |\mathcal{R}|$ and $k = \overline{\mathcal{B}}$ correspond to easier instances. This is because the number of feasible choices for the subset of roots is larger when k is far from the extreme values. This yields a remarkable increase in the number of negative reduced cost columns to be considered, a larger number of pricing iterations and a larger number of fractional values in the optimal solution of the linear restricted master problem, which in turn requires a larger number of branching levels in the branch-and-price tree.

4 Conclusions

The exact optimization algorithm we have devised for the single-base-cell and the multi-base-cell problems allowed to study the trade-off between the obfuscation level, represented by the value of the parameter τ, and the computational hardness of the resulting combinatorial optimization instance. Our tests confirmed that the value of the threshold τ strongly affects the cardinality of the resulting obfuscated regions and the computing time, as expected.

In the multi-arborescence problem the user can suitably calibrate the number of required arborescences which results in fine-grained or coarse-grained representation of the sensible subsets of cells to be obfuscated. We observed a significant relationship between the number of arborescences k and the computing time. When k is close to the maximum or minimum values, instances turn out to be easier to solve to optimality. However the effect of the combinatorial explosion is quite evident, as expected.

To cope with large instances we have devised a branch-and-price-based heuristic algorithm which is able to consistently find optimal or near-optimal solutions in a computing time which is reduced by three to four orders of magnitude with respect to that required for exact optimization.

References

[1] B.V. Cherkassky, A.V. Goldberg, *On implementing push-relabel method for the maximum flow problem*, Algorithmica 19 (1997) 390-410.

[2] M.L. Damiani, E. Bertino, C. Silvestri, *The PROBE Framework for the Personalized Cloaking of Private Locations*, Transactions on Data Privacy 3(2) (2010) 123-148.

[3] M.L. Damiani, C. Silvestri, E. Bertino, *Fine-Grained Cloaking of Sensitive Positions in Location-Sharing Applications*, IEEE Pervasive Computing 10(4) (2011) 64-72.

[4] S. Chopra, E. Gorres, M.R. Rao, *Solving a Steiner tree problem on a graph using branch and cut*, ORSA Journal on Computing 4 (1992) 320-335.

[5] T. Koch, A. Martin, *Solving Steiner tree problems in graphs to optimality*, Networks 32 (1998) 207-232.

[6] J.W. van Laarhoven, *Exact and heuristic algorithms for the Euclidean Steiner tree problem*, Thesis, University of Iowa, 2010.

[7] I. Ljubic, R. Weiskircher, U. Pferschy, G.W. Klau, P. Mutzel, M. Fischetti, *An Algorithmic Framework for the Exact Solution of the Prize-Collecting Steiner Tree Problem* Mathematical Programming 105 (2006) 427-449

5 Appendix A: Detailed results

In Tables 6-13 we present the detailed results of our tests. All tables have the following columns:

- Instance: the ID is followed by the grid size and the value of τ;
- TS: type of solution. In this field "O" stands for optimal solution (the pricing problem is solved to optimality); "B&CT" indicates that the pricing problem is only solved with truncated branch-and-cut and greedy; "G" indicates that only the greedy pricer is used.
- k: the number of arborescences to compute;
- rTime: this is the computing time spent at the root node by the branch-and-price exact algorithm or the branch-and-price-based heuristics;
- rLB: this is the final optimal value of the linear restricted master problem at the root node; it is a valid lower bound only when pricing is solved to optimality;
- rUB: this is the value of the upper bound computed at the root node by the branch-and-price algorithm of the branch-and-price-based heuristics;
- UB: this is the best upper bound found. It is guaranteed to be optimal only when TS is "O";
- Obj: this is the average cardinality of the arborescences ($\frac{UB}{k}$);
- pTime: this is the overall computing time required by the pricing algorithms;
- Time: this is the overall computing time taken by the branch-and-price algorithm.

Instance	TS	k	rTime	rLB	rUB	UB	Obj	pTime	Time
1-15x15-0.05	O	1	812.14	15.00	15.00	15.00	15.00	799.32	812.14
1-15x15-0.05	B&CT	1	10.78	15.00	15.00	15.00	15.00	10.57	10.78
1-15x15-0.05	G	1	0.02	17.00	17.00	17.00	17.00	0.01	0.02
2-15x15-0.05	O	2	735.19	17.00	17.00	17.00	8.50	723.12	735.19
2-15x15-0.05	B&CT	2	9.34	17.00	23.00	17.00	8.50	43.99	44.79
2-15x15-0.05	G	2	0.02	23.00	23.00	23.00	11.50	0.01	0.02
3-15x15-0.05	O	3	912.56	20.50	-	21.00	7.00	2679.33	2812.54
3-15x15-0.05	B&CT	3	8.34	21.00	-	21.00	7.00	33.82	34.49
3-15x15-0.05	G	3	0.03	-	-	-	-	0.02	0.03
4-15x15-0.05	O	4	981.12	23.00	23.00	23.00	5.75	967.69	981.12
4-15x15-0.05	B&CT	4	57.50	24.00	24.00	24.00	6.00	56.36	57.50
4-15x15-0.05	G	4	0.03	-	-	-	-	0.02	0.03
5-15x15-0.10	O	1	234.89	10.00	10.00	10.00	10.00	227.19	234.89
5-15x15-0.10	B&CT	1	4.75	10.00	10.00	10.00	10.00	4.65	4.75
5-15x15-0.10	G	1	0.02	10.00	10.00	10.00	10.00	0.01	0.02
6-15x15-0.10	O	2	287.19	12.00	12.00	12.00	6.00	280.72	287.19
6-15x15-0.10	B&CT	2	34.53	12.00	12.00	12.00	6.00	33.85	34.53
6-15x15-0.10	G	2	0.02	13.00	13.00	13.00	6.50	0.01	0.02
7-15x15-0.10	O	3	837.15	14.00	-	14.00	4.67	828.19	837.15
7-15x15-0.10	B&CT	3	12.64	14.00	-	14.00	4.67	35.01	35.69
7-15x15-0.10	G	3	0.01	-	-	-	-	0.01	0.01
8-15x15-0.10	O	4	201.38	17.00	17.00	17.00	4.25	198.27	201.38
8-15x15-0.10	B&CT	4	20.79	17.00	17.00	17.00	4.25	20.39	20.79
8-15x15-0.10	G	4	0.02	-	-	-	-	0.01	0.02

Table 6: One base subset of size 2×2 and $\tau \in \{0.05, 0.10\}$.

Instance	TS	k	rTime	rLB	rUB	UB	Obj	pTime	Time
9-15x15-0.20	O	1	198.12	8.00	8	8	8.00	195.74	198.12
9-15x15-0.20	B&CT	1	8.62	8.00	8	8	8.00	8.45	8.62
9-15x15-0.20	G	1	0.02	8.00	8	8	8.00	0.01	0.02
10-15x15-0.20	O	2	389.32	8.00	8	8	4.00	380.37	389.32
10-15x15-0.20	B&CT	2	10.05	8.00	8	8	4.00	9.83	10.05
10-15x15-0.20	G	2	0.02	8.00	8	8	4.00	0.01	0.02
11-15x15-0.20	O	3	107.49	10.00	10	10	3.33	104.48	107.49
11-15x15-0.20	B&CT	3	11.89	10.00	10	10	3.33	11.64	11.89
11-15x15-0.20	G	3	0.04	10.00	10	10	3.33	0.02	0.04
12-15x15-0.20	O	4	85.28	12.00	12	12	3.00	81.41	85.28
12-15x15-0.20	B&CT	4	4.74	12.00	12	12	3.00	4.63	4.74
12-15x15-0.20	G	4	0.01	12.00	12	12	3.00	0.01	0.01
13-15x15-0.40	O	1	389.37	6.00	6	6	6.00	385.54	389.37
13-15x15-0.40	B&CT	1	9.43	6.00	6	6	6.00	9.24	9.43
13-15x15-0.40	G	1	0.02	6.00	6	6	6.00	0.01	0.02
14-15x15-0.40	O	2	292.43	6.00	6	6	3.00	287.74	292.43
14-15x15-0.40	B&CT	2	5.32	6.00	6	6	3.00	5.13	5.32
14-15x15-0.40	G	2	0.04	6.00	6	6	3.00	0.02	0.04
15-15x15-0.40	O	3	298.65	8.00	8	8	2.67	291.57	298.65
15-15x15-0.40	B&CT	3	4.29	8.00	8	8	2.67	4.01	4.29
15-15x15-0.40	G	3	0.02	8.00	8	8	2.67	0.01	0.02
16-15x15-0.40	O	4	108.18	10.00	10	10	2.50	103.84	108.18
16-15x15-0.40	B&CT	4	2.63	10.00	10	10	2.50	2.41	2.63
16-15x15-0.40	G	4	0.02	10.00	10	10	2.50	0.01	0.02

Table 7: One base subset of size 2×2 and $\tau \in \{0.20, 0.40\}$.

Instance	TS	k	rTime	rLB	rUB	UB	Obj	pTime	Time
1-15x15-0.05	O	3	5173.77	28.00	28	28	9.33	5074.75	5173.77
1-15x15-0.05	B&CT	3	53.01	28.00	28	28	9.33	52.17	53.01
1-15x15-0.05	G	3	0.03	-	-	-	-	0.02	0.03
2-15x15-0.05	O	4	5333.87	29.00	29	29	7.25	5202.68	5333.87
2-15x15-0.05	B&CT	4	43.32	29.00	29	29	7.25	42.59	43.32
2-15x15-0.05	G	4	0.02	-	-	-	-	0.01	0.02
3-15x15-0.05	O	5	5521.37	31.00	31	31	6.20	5391.49	5521.37
3-15x15-0.05	B&CT	5	43.80	31.00	31	31	6.20	43.10	43.80
3-15x15-0.05	G	5	0.02	-	-	-	-	0.01	0.02
4-15x15-0.05	O	6	7662.36	33.00	33	33	5.50	7512.12	7662.36
4-15x15-0.05	B&CT	6	47.21	33.00	33	33	5.50	46.46	47.21
4-15x15-0.05	G	6	0.02	-	-	-	-	0.02	0.03
5-15x15-0.05	O	7	5327.78	37.00	-	37	5.29	16329.91	16593.45
5-15x15-0.05	B&CT	7	37.38	37.00	-	37	5.29	78.67	80.01
5-15x15-0.05	G	7	0.02	-	-	-	-	0.01	0.02
6-15x15-0.05	O	8	5997.45	40.00	40	40	5.00	5802.98	5997.45
6-15x15-0.05	B&CT	8	58.17	40.00	40	40	5.00	57.21	58.17
6-15x15-0.05	G	8	0.02	-	-	-	-	0.01	0.02

Table 8: Two base subsets of size 2×2 and $\tau = 0.05$.

Instance	TS	k	rTime	rLB	rUB	UB	Obj	pTime	Time
7-15x15-0.10	O	3	5005.09	19.00	19	19	6.33	4897.25	5005.09
7-15x15-0.10	B&CT	3	11.72	19.00	19	19	6.33	11.48	11.72
7-15x15-0.10	G	3	0.03	-	-	-	-	0.02	0.03
8-15x15-0.10	O	4	4639.56	20.50	-	21	5.25	25848.57	26409.40
8-15x15-0.10	B&CT	4	10.96	20.50	-	21	5.25	22.19	22.61
8-15x15-0.10	G	4	0.03	-	-	-	-	0.02	0.03
9-15x15-0.10	O	5	4532.21	22.50	24	23	4.60	26398.56	27946.30
9-15x15-0.10	B&CT	5	11.45	23.00	24	23	4.60	22.57	22.99
9-15x15-0.10	G	5	0.02	-	-	-	-	0.01	0.02
10-15x15-0.10	O	6	3989.76	24.50	25	25	4.17	3811.32	3989.76
10-15x15-0.10	B&CT	6	10.98	24.50	25	25	4.17	28.90	29.41
10-15x15-0.10	G	6	0.02	-	-	-	-	0.01	0.02
11-15x15-0.10	O	7	3655.21	27.00	28	27	3.86	7945.73	8139.23
11-15x15-0.10	B&CT	7	13.67	27.00	28	27	3.86	34.30	34.93
11-15x15-0.10	G	7	0.02	-	-	-	-	0.01	0.02
12-15x15-0.10	O	8	2987.38	30.00	30	30	3.75	2871.91	2987.38
12-15x15-0.10	B&CT	8	36.75	30.00	30	30	3.75	36.16	36.75
12-15x15-0.10	G	8	0.02	-	-	-	-	0.01	0.02

Table 9: Two base subsets of size 2×2 and $\tau = 0.10$.

Instance	TS	k	rTime	rLB	rUB	UB	Obj	pTime	Time
13-15x15-0.20	O	3	9312.33	15.00	15	15	5.00	9158.34	9312.33
13-15x15-0.20	B&CT	3	4.45	15.00	15	15	5.00	4.34	4.45
13-15x15-0.20	G	3	0.04	15.00	15	15	5.00	0.02	0.03
14-15x15-0.20	O	4	2257.11	15.00	15	15	3.75	2227.92	2257.11
14-15x15-0.20	B&CT	4	4.20	15.00	15	15	3.75	4.10	4.20
14-15x15-0.20	G	4	0.04	15.00	15	15	3.75	0.02	0.04
15-15x15-0.20	O	5	3245.32	16.00	16	16	3.20	3201.92	3245.32
15-15x15-0.20	B&CT	5	9.48	16.00	16	16	3.20	9.25	9.48
15-15x15-0.20	G	5	0.04	16.00	16	16	3.20	0.02	0.04
16-15x15-0.20	O	6	1837.87	18.00	-	18	3.00	4521.88	4643.14
16-15x15-0.20	B&CT	6	5.45	18.00	-	18	3.00	12.62	12.94
16-15x15-0.20	G	6	0.07	18.00	-	18	3.00	0.04	0.07
17-15x15-0.20	O	7	589.12	20.00	20	20	2.86	577.63	589.12
17-15x15-0.20	B&CT	7	9.92	20.00	20	20	2.86	9.73	9.92
17-15x15-0.20	G	7	0.03	20.00	20	20	2.86	0.02	0.03
18-15x15-0.20	O	8	481.32	22.00	22	22	2.75	476.54	481.32
18-15x15-0.20	B&CT	8	6.70	22.00	22	22	2.75	6.55	6.70
18-15x15-0.20	G	8	0.03	-	-	-	-	0.02	0.03

Table 10: Two base subsets of size 2×2 and $\tau = 0.20$.

Instance	TS	k	rTime	rLB	rUB	UB	Obj	pTime	Time
19-15x15-0.40	O	3	781.14	12.00	12	12	4.00	777.02	781.14
19-15x15-0.40	B&CT	3	9.67	12.00	12	12	4.00	9.46	9.67
19-15x15-0.40	G	3	0.03	12.00	12	12	4.00	0.02	0.03
20-15x15-0.40	O	4	1353.89	12.00	12	12	3.00	1339.20	1353.89
20-15x15-0.40	B&CT	4	16.00	12.00	12	12	3.00	15.68	16.00
20-15x15-0.40	G	4	0.03	13.00	13	13	3.25	0.02	0.03
21-15x15-0.40	O	5	1023.87	13.00	13	13	2.60	1009.18	1023.87
21-15x15-0.40	B&CT	5	14.30	13.00	13	13	2.60	13.99	14.30
21-15x15-0.40	G	5	0.03	14.00	14	14	2.80	0.02	0.03
22-15x15-0.40	O	6	1078.24	14.00	14	14	2.33	1065.38	1078.24
22-15x15-0.40	B&CT	6	13.30	14.00	14	14	2.33	12.95	13.30
22-15x15-0.40	G	6	0.03	15.00	15	15	2.50	0.05	0.06
23-15x15-0.40	O	7	819.01	16.00	16	16	2.29	814.73	819.01
23-15x15-0.40	B&CT	7	4.38	16.00	16	16	2.29	4.25	4.38
23-15x15-0.40	G	7	0.03	16.00	16	16	2.29	0.02	0.03
24-15x15-0.40	O	8	691.09	18.00	18	18	2.25	686.93	691.09
24-15x15-0.40	B&CT	8	2.93	18.00	18	18	2.25	2.86	2.93
24-15x15-0.40	G	8	0.03	18.00	18	18	2.25	0.02	0.03

Table 11: Two base subsets of size 2×2 and $\tau = 0.40$.

Instance	TS	k	rTime	rLB	rUB	UB	Obj	pTime	Time
1-15x15-0.05	O	2	2389.32	18.00	18	18	9.00	2351.33	2389.32
1-15x15-0.05	B&CT	2	58.34	18.00	18	18	9.00	57.41	58.34
1-15x15-0.05	G	2	0.06	22.00	22	22	11.00	0.05	0.06
2-15x15-0.05	O	4	4191.18	23.50	-	24	6.00	10494.23	10663.78
2-15x15-0.05	B&CT	4	287.13	23.50	-	24	6.00	369.31	375.28
2-15x15-0.05	G	4	0.02	-	-	-	-	0.02	0.02
3-15x15-0.05	O	6	1743.69	29.00	29	29	4.83	1715.97	1743.69
3-15x15-0.05	B&CT	6	101.45	30.00	30	30	5.00	99.84	101.45
3-15x15-0.05	G	6	0.02	-	-	-	-	0.02	0.02
4-15x15-0.05	O	8	1481.78	35.00	35	35	4.38	1458.22	1481.78
4-15x15-0.05	B&CT	8	49.32	35.00	35	35	4.38	48.54	49.32
4-15x15-0.05	G	8	0.02	-	-	-	-	0.02	0.02
5-15x15-0.10	O	2	1881.19	12.00	12	12	6.00	1851.28	1881.19
5-15x15-0.10	B&CT	2	38.87	12.00	12	12	6.00	38.25	38.87
5-15x15-0.10	G	2	0.04	14.00	14	14	7.00	0.03	0.04
6-15x15-0.10	O	4	5919.32	16.75	18	17	4.25	9666.27	9822.45
6-15x15-0.10	B&CT	4	212.34	17.00	19	18	4.50	577.76	587.09
6-15x15-0.10	G	4	0.02	-	-	-	-	0.02	0.02
7-15x15-0.10	O	6	1366.25	19.50	20	20	3.33	3387.39	3442.12
7-15x15-0.10	B&CT	6	94.41	20.50	21	20	3.33	186.20	189.21
7-15x15-0.10	G	6	0.02	-	-	-	-	0.02	0.02
8-15x15-0.10	O	8	819.92	27.00	27	27	3.38	806.88	819.92
8-15x15-0.10	B&CT	8	31.44	27.00	27	27	3.38	30.94	31.44
8-15x15-0.10	G	8	0.03	-	-	-	-	0.02	0.03

Table 12: One base subset of size 3×3 and $\tau \in \{0.05, 0.10\}$.

Instance	TS	k	rTime	rLB	rUB	UB	Obj	pTime	Time
9-15x15-0.20	O	2	818.67	13.00	13	13	6.50	805.65	818.67
9-15x15-0.20	B&CT	2	21.54	13.00	13	13	6.50	21.20	21.54
9-15x15-0.20	G	2	0.03	14.50	15	15	7.50	0.09	0.11
10-15x15-0.20	O	4	2019.34	15.00	15	15	3.75	1987.23	2019.34
10-15x15-0.20	B&CT	4	51.32	15.00	15	15	3.75	50.50	51.32
10-15x15-0.20	G	4	0.04	16.00	16	16	4.00	0.03	0.04
11-15x15-0.20	O	6	691.32	20.00	20	20	3.33	680.33	691.32
11-15x15-0.20	B&CT	6	48.14	20.00	20	20	3.33	47.37	48.14
11-15x15-0.20	G	6	0.04	-	-	-	-	0.03	0.04
12-15x15-0.20	O	8	901.32	23.00	23	23	2.88	886.99	901.32
12-15x15-0.20	B&CT	8	14.76	23.00	23	23	2.88	14.53	14.76
12-15x15-0.20	G	8	0.04	-	-	-	-	0.03	0.04
13-15x15-0.40	O	2	829.34	12.00	12	12	6.00	816.15	829.34
13-15x15-0.40	B&CT	2	17.75	12.00	12	12	6.00	17.47	17.75
13-15x15-0.40	G	2	0.05	12.00	12	12	6.00	0.04	0.05
14-15x15-0.40	O	4	1759.71	14.00	14	14	3.50	1731.73	1759.71
14-15x15-0.40	B&CT	4	48.87	14.00	14	14	3.50	48.09	48.87
14-15x15-0.40	G	4	0.05	15.00	15	15	3.75	0.04	0.05
15-15x15-0.40	O	6	536.72	17.00	17	17	2.83	528.19	536.72
15-15x15-0.40	B&CT	6	15.47	17.00	17	17	2.83	15.22	15.47
15-15x15-0.40	G	6	0.04	19.00	19	19	3.17	0.03	0.04
16-15x15-0.40	O	8	601.89	20.00	20	20	2.50	592.32	601.89
16-15x15-0.40	B&CT	8	9.89	20.00	20	20	2.50	9.73	9.89
16-15x15-0.40	G	8	0.04	-	-	-	-	0.03	0.04

Table 13: One base subset of size 3×3 and $\tau \in \{0.20, 0.40\}$.

In Table 14 we report the results obtained with the heuristic branch-and-price, where all pricing problems were solved with the greedy algorithm and the branch-and-cut algorithm truncated at teh root node. A time limit of one hour was imposed to these tests. The columns of Table 14 are the following:

- "Instance" includes the size of the base subsets and the value of τ;
- $|\mathcal{R}|$ indicates the number of base subsets;
- "Size" indicates the size of the base subsets;
- k indicates the number of arborescences;
- "N. cells" indicates the number of cells in the best solution found;
- "Obj" is the average cardinality of the arborescences in the best solution found;
- "Time" is the computing time.

Instance	$\|\mathcal{R}\|$	Size	k	N.cells	Obj	Time
1-EUR-20x20-0.05	3	2×2	12	59	4.92	238.29
1-EUR-20x20-0.10	3	2×2	12	49	4.08	231.78
1-EUR-20x20-0.20	3	2×2	12	38	3.17	187.39
1-EUR-20x20-0.40	3	2×2	12	28	2.33	90.49
2-EUR-20x20-0.05	5	2×2	20	-	-	1739.69
2-EUR-20x20-0.05	5	2×2	19	-	-	1928.48
2-EUR-20x20-0.05	5	2×2	18	93	5.17	2491.38
2-EUR-20x20-0.10	5	2×2	20	-	-	1439.81
2-EUR-20x20-0.10	5	2×2	19	79	4.16	2001.64
2-EUR-20x20-0.20	5	2×2	20	-	-	1249.53
2-EUR-20x20-0.20	5	2×2	19	61	3.21	1191.83
2-EUR-20x20-0.40	5	2×2	20	45	2.25	963.18
3-EUR-20x20-0.05	3	3×3	24	-	-	3600.00
3-EUR-20x20-0.10	3	3×3	24	-	-	3600.00
3-EUR-20x20-0.20	3	3×3	24	78	3.25	2301.15
3-EUR-20x20-0.40	3	3×3	24	54	2.25	3036.48
4-EUR-30x30-0.05	3	2×2	12	65	5.42	2996.19
4-EUR-30x30-0.10	3	2×2	12	52	4.33	2693.45
4-EUR-30x30-0.20	3	2×2	12	40	3.33	2227.31
4-EUR-30x30-0.40	3	2×2	12	2	2.25	1801.75
5-EUR-30x30-0.05	3	3×3	24	-	-	3600.00
5-EUR-30x30-0.10	3	3×3	24	-	-	3600.00
5-EUR-30x30-0.20	3	3×3	24	-	-	3600.00
5-EUR-30x30-0.40	3	3×3	24	52	2.17	3048.79

Table 14: Heuristic results on large instances.