arXiv:2608.20894v1 [math.OC] 21 Aug 2026

Maximum Covering Network Design on Graphs with Low Connectivity: Dynamic Programming and Block-Cut Trees

Felix Rauh, Jannik Matuschke, Hande Yaman

August 24, 2026

Abstract

Planning accessible public services such as health care, emergency response, and schools often requires not only choosing where to open facilities but also improving the network that connects people to them, for example upgrading flood-prone roads in vulnerable regions. Most location models, however, take the network as fixed and the budget as given. We study the Maximum Covering Network Design Problem, in which a single budget is shared between opening facilities and upgrading weak links to maximize the population within a target travel distance of an open facility. The problem is hard even on the simplest networks, and planners usually want to see how coverage grows with the budget, not a single plan.

We develop an exact dynamic-programming framework that exploits a property common to real road networks: their low connectivity, with many cut points whose removal disconnects the network. On trees, the recursion is self-contained: each state reduces to a few simple facility and upgrade choices that are fast to compute without a solver, giving predictable running times; for larger budgets and travel distances it outperforms solving the MILP formulation directly. On general low-connectivity networks, the framework decomposes the problem at the cut points and embeds a given MILP formulation to solve the resulting pieces, coordinating them through coverage conditions at the interfaces. This lets us compare a formulation on its own against the same formulation inside the framework: across 306 test cases the framework matches or outperforms direct solving on more than 80% of instances. Because it evaluates all budget levels in a single run, it also yields the full coverage-versus-budget curve at no extra cost, whereas direct solving must split its time across individual budgets.

Keywords: facility location, network design, dynamic programming, integer programming

1 Introduction

Infrastructure planning in health care, education, and emergency services involves fundamental trade-offs between facility investments and network improve-

ments. Opening new facilities increases service capacity, while improving connectivity can extend the reach of existing facilities. This trade-off is particularly relevant in regions with underdeveloped or vulnerable infrastructure, where accessibility may be limited by poor road conditions (such as flooded gravel roads) rather than a lack of facilities; investments may then include upgrading gravel roads or installing drainage systems (Cook et al. 2019, Rozenberg et al. 2019). Despite the practical relevance of combining network design with facility location, most of the location literature assumes the underlying network is fixed. A second limitation is that the budget is typically treated as a fixed input, whereas in practice it is often part of the planning question: decision-makers need to see how achievable coverage grows with the budget (the coverage-versus-budget curve) rather than the optimal plan for a single budget value.

We address both. We study a model (the MCNDP, defined below) that jointly optimizes facility locations and network upgrades, and develop exact methods that exploit the low connectivity of real-world networks through a decomposition approach. Because the decomposition evaluates many budget levels within a single solve, it yields the full budget curve at little additional cost.

The Maximum Covering Network Design Problem. We study the *Maximum Covering Network Design Problem* (MCNDP) on an undirected graph $G = (N, E)$, combining facility location and edge upgrade decisions under a joint budget constraint. Certain edges are "weak" and require investment to become usable; a coverage radius $\bar{d}$ specifies the maximum distance within which a demand node can be served by an open facility.

Definition 1 (Maximum Covering Network Design Problem (MCNDP))**.** *Given an undirected graph $G = (N, E)$ with demand weights $h_k \in \mathbb{Z}_{\geq 0}$ for $k \in N$, facility opening costs $c_j^F \in \mathbb{Z}_{\geq 0}$ for candidate sites $j \in F \subseteq N$, edge upgrade costs $c_e^E \in \mathbb{Z}_{\geq 0}$ and lengths $\ell_e \in \mathbb{Z}_{\geq 0}$ for edges $e \in E$, a coverage radius $\bar{d} \in \mathbb{Z}_{>0}$, and a budget $\bar{b} \in \mathbb{Z}_{>0}$, the MCNDP consists of choosing a set $Y \subseteq F$ of facilities to open and a set $X \subseteq E$ of edges to upgrade such that:*

1. *The total investment satisfies the budget: $\sum_{j \in Y} c_j^F + \sum_{e \in X} c_e^E \leq \bar{b}$.*
2. *The total weight of covered nodes is maximized, where a node k is covered if there exists $j \in Y$ with $d_{G[X]}(k, j) \leq \bar{d}$, where $G[X] = (N, X)$ is the subgraph with edge set X and $d_{G[X]}(k, j)$ is the length of a shortest k-j-path in $G[X]$.*

Without loss of generality, we assume $\ell_e > 0$ for all $e \in E$, since any zero-length edge can be contracted in preprocessing. Existing infrastructure is captured by zero costs: an already-usable edge has $c_e^E = 0$ and an already-open facility $c_j^F = 0$.

The MCNDP generalizes the classical Maximum Covering Location Problem (MCLP), which arises when all edges are already passable (zero upgrade cost). Moreover, the MCNDP is closely related to the network design problem

(NDP), or, more precisely, the multicommodity fixed-charge NDP (Magnanti and Wong 1984), which selects edges at minimum cost to route flow between origin–destination pairs. The MCNDP differs from it in objective (maximizing coverage under a budget constraint rather than minimizing cost) and can be viewed as its single-source variant, with all facilities connected to a dummy super-source and all nodes acting as potential targets.

Complexity. The MCNDP is NP-hard even on trees: on a star graph with an open facility at the center, selecting which leaf nodes to connect under a budget constraint corresponds to the 0-1 knapsack problem. Moreover, even connecting a single demand node to a facility is NP-hard in general, as finding a minimum-cost path of length at most $\bar{d}$ corresponds to a resource-constrained shortest path problem. Consequently, no constant-factor approximation exists: no polynomial-time algorithm can decide whether the optimal coverage is 0 or h_k for an arbitrarily large weight h_k (unless P = NP).

Focus on low-connectivity graphs. While the MCNDP is challenging on general graphs, many real-world networks exhibit low connectivity. Road networks in rural or developing areas often have few alternative routes due to geographical constraints such as valleys, rivers, or islands; Tian et al. (2017) show that articulation points, whose removal disconnects the graph, comprise almost 20% of nodes in three US state road networks. We therefore focus on trees and, more generally, graphs whose decomposition at articulation points yields subproblems of smaller size and enables efficient evaluation across many budget levels at once (exploited in Section 4.3 to produce full coverage-versus-budget curves).

Contributions and methodology. Our main contributions are as follows:

- For tree graphs, we present an assignment-based formulation and a pseudo-polynomial dynamic programming (DP) algorithm that exploits the unique path structure (Section 4.2).
- For connected graphs with *articulation points* (cut vertices whose removal disconnects the graph), we introduce a decomposition framework based on block-cut trees. The key insight is that subproblems on biconnected components can be solved with parameterized border conditions at articulation points, capturing both budget allocation and coverage interactions across blocks (Section 4.3).
- The framework is independent of how the block subproblems are solved: blocks are coordinated only through selector variables and signed border conditions, so each can be handled by any suitable block-level MILP formulation.
- In our experiments, we compare our approaches against solving the mixed-integer linear programming (MILP) formulation directly (Section 5). Since

the DP enumerates all budget levels inherently, it yields the complete coverage-versus-budget curve at no additional cost.

Beyond the MCNDP, the same framework could extend to other location and network design problems, such as uncapacitated facility location; we discuss possibilities for such extensions in Section 6.

Outline. The remainder of this paper is organized as follows. Section 2 reviews related literature on maximum covering location problems, network design, and decomposition techniques. Section 3 presents a MILP formulation for general graphs and one for tree graphs. Section 4 develops the DP decomposition framework, first applied to trees and then extended to block-cut tree decomposition. Section 5 reports computational results, and Section 6 concludes.

2 Literature review and related problems

This section reviews the literature on maximum covering location problems, network design, and their combination. We also discuss solution approaches relevant to our methodology, particularly dynamic programming on trees and decomposition techniques based on graph connectivity.

Maximum Covering Location Problems. The MCLP, as introduced by Church and ReVelle (1974), asks to open at most p facilities to maximize covered demand within a distance radius; see Laporte et al. (2019) for an overview. The problem is known to be well solvable with integer linear programming (Snyder 2011), and the state-of-the-art solver by Cordeau et al. (2019) handles instances with up to 15 million demand points via Benders decomposition. On trees, polynomial-time DPs are known: Megiddo et al. (1983) achieve $\mathcal{O}(pn^2)$ for the MCLP, and Tamir (1996) the same for the related p-median problem, using a binary-tree transformation that we also adopt. Our DP in Section 4.2 extends these works by replacing the p-facility constraint with a knapsack-type budget and adding edge upgrade decisions, yielding pseudo-polynomial rather than polynomial complexity.

Network Design Problems. The network design problem (NDP), formalized by Magnanti and Wong (1984) using flow-based formulations, selects edges to satisfy connectivity requirements at minimum cost; see Crainic et al. (2021) for an overview. In the multicommodity fixed-charge NDP closest to our setting, a separate commodity is routed for each origin–destination pair (the *multicommodity* aspect), and each selected edge incurs a fixed cost independent of the flow it carries (the *fixed-charge* aspect). Our flow formulation in Section 3.1 builds upon these classical flow-based formulations. Exact methods for this problem are typically limited to hundreds or a few thousands of nodes and edges (Zetina et al. 2019). Our framework is independent of the underlying

connectivity model and could equally use cut-based formulations (Arslan et al. 2020, 2019); we focus on the flow formulation throughout this paper.

Combined Facility Location and Network Design. An overview of combined facility location and network design problems is provided by Contreras and Fernández (2012). Melkote and Daskin (2001b,a) introduce the uncapacitated facility location/network design problem (UFLNDP) with a cost-minimization objective, studying the trade-off of improved network topology versus opening new facilities. Structurally close to our setting is the network design problem with relays (NDPR) studied by Leitner et al. (2019), in which signal regeneration devices are placed at nodes while additional links may be installed, so that the distance between consecutive devices along a path stays below a given threshold. As in the MCNDP, node installation and edge installation therefore interact through a distance bound, and the authors likewise develop exact mixed-integer programming approaches (branch-and-price and branch-price-and-cut) rather than heuristics. The NDPR differs from the MCNDP in that it minimizes total installation cost while serving all origin-destination pairs, whereas we maximize covered demand under a single budget shared between facility and upgrade decisions.

Most relevant to our work is Bucarey et al. (2022), who maximize covered origin-destination pairs subject to a budget constraint on network construction and a length bound on paths. The MCNDP is a special case of their problem (with a single super-source). In an earlier study, Murawski and Church (2009) study a similar coverage-maximizing network improvement model with fixed facility locations, solving a flow formulation on a case study in Ghana. van Veggel et al. (2026) address the same model at larger scale with a greedy heuristic. A related variant by Baldomero-Naranjo et al. (2022, 2024) considers shortening existing edges rather than binary upgrades, with a complexity analysis on trees similar in spirit to our DP in Section 4.2.

Decomposition via Block-Cut Trees. Several optimization problems have been solved by decomposing graphs at articulation points into biconnected components (blocks). Baïou and Barahona (2009) show that the integrality of facility location polytopes can be established block by block: if constraints describe an integral polyhedron for each block, so does the combined system with interaction constraints at articulation points. Block decomposition has also been applied to degree-dependent spanning tree problems (Landete et al. 2017), switch location (Landete et al. 2019), the d-Minimum Branch Vertices problem (Moreno et al. 2018), the generalized vertex cover problem (Correcher et al. 2025), and the generalized network dismantling problem (Feng et al. 2024).

A common property of these approaches is that subproblems on blocks are solved independently, with case distinctions or special constraints handling the interactions at articulation points. Our approach differs in that we must allocate a shared budget across blocks, and the covering interactions at articulation points involve both directions and distances. To the best of our knowledge, the

MCNDP has not been studied with such a decomposition approach.

3 MILP formulations for the MCNDP

We present two formulations: a flow-based formulation for general graphs (Section 3.1) and an assignment-based formulation for trees (Section 3.2). The flow formulation uses multicommodity flow constraints for connectivity in the tradition of Magnanti and Wong (1984), and is closest to the flow formulation of Bucarey et al. (2022).

High-level formulation. In all formulations, we use binary variables $z_k \in \{0,1\}$ indicating coverage of demand node $k \in N$ and $x_e \in \{0,1\}$ for the decision to upgrade edge e (edges that already exist have $c_e^E = 0$ and x_e can be fixed to 1). For the flow formulation, facility openings can be represented as edge decisions by introducing a source node s and the facility-edge set $E^F := \{\{s,j\} \mid j \in F\}$, yielding the extended edge set $\overline{E} := E \cup E^F$. These facility edges satisfy $\ell_{\{s,j\}} := 0$ and $c^E_{\{s,j\}} := c^F_j$. We use this construction whenever a single flow formulation MILP is solved, on the full graph or on an individual block, but not across the decomposition, as a global source node s would not preserve the tree and block-cut tree structure that the decomposition exploits. The tree formulation (Section 3.2) keeps facilities as nodes of G, using explicit facility variables $y_j = x_{\{s,j\}} \in \{0,1\}$. For a comprehensive summary of all notation, see Appendix A.

Using only the variables (x, z), both formulations presented below can be written in the unified form

$$\max\Big\{ \sum_{k\in N} h_k z_k \;\Big|\; (x,z) \in \mathcal{F},\;\; \sum_{e\in\overline{E}} c_e^E x_e \le \bar{b},\;\; x_e, z_k \in \{0,1\},\;\; e \in \overline{E},\;\; k \in N \Big\}, \tag{1}$$

where the feasibility set $\mathcal{F}$ encodes the connectivity and covering constraints. The formulations differ only in how $\mathcal{F}$ is represented: via flow variables (Section 3.1) or assignments along unique paths on trees (Section 3.2). In Section 4.3.3, we extend this to the block-cut tree setting, where $\mathcal{F}$ is parameterized by border conditions at the articulation points.

3.1 Flow formulation

The flow formulation uses the source-node construction introduced above. Let $\overline{N} := N \cup \{s\}$. With this construction, a node $i \in N$ is covered if and only if there exists an s-i-path of length at most $\bar{d}$ via edges in the solution.

To describe coverage as a directed flow from s to the demand nodes, let $\overline{A}$ denote the set of directed arcs containing both (u,v) and (v,u) for each $\{u,v\} \in \overline{E}$. We use one flow commodity per demand node $k \in N$, and denote the respective flow variables by $f_a^k \in \{0,1\}$ for $a \in \overline{A}$. For a node $v \in \overline{N}$, let $\delta^+(v)$ denote all outgoing arcs and $\delta^-(v)$ all incoming arcs of v in $\overline{G} = (\overline{N}, \overline{A})$.

Using the edge representation for facility openings as discussed above, we obtain:

$$\max \quad \sum_{k \in N} h_k z_k \tag{2}$$

$$\text{s.t.} \quad \sum_{e \in \overline{E}} c_e^E x_e \leq \bar{b} \tag{3}$$

$$\sum_{a \in \delta^+(v)} f_a^k - \sum_{a \in \delta^-(v)} f_a^k = \begin{cases} z_k & v = s, \\ -z_k & v = k, \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in \overline{N}, \forall k \in N \tag{4}$$

$$f_{(u,v)}^k + f_{(v,u)}^k \leq x_e \qquad \forall e = \{u, v\} \in \overline{E}, \forall k \in N \tag{5}$$

$$\sum_{a \in \overline{A}} \ell_a f_a^k \leq \bar{d}\, z_k \qquad \forall k \in N \tag{6}$$

$$x_e \in \{0,1\}\ \forall e \in \overline{E}, \quad z_k \in \{0,1\}\ \forall k \in N, \quad f_a^k \in \{0,1\}\ \forall k \in N,\ a \in \overline{A} \tag{7}$$

Objective (2) and budget constraint (3) are as in the high-level formulation. Constraint (4) enforces flow conservation with supply at s and demand at k; (5) links flow to edge decisions (including facility edges adjacent at s); and (6) bounds the path length for covered nodes by $\bar{d}$. Theoretically, the integrality of the flow variables f in (7) can be relaxed, since given integral x and z, an integral optimal flow exists by standard network flow arguments (Bucarey et al. 2022). In our experiments, however, relaxing it does not improve solver performance.

3.2 Assignment-based MILP formulation on trees

When G is a tree, paths are unique: a node k is covered by facility j if and only if j is open and all edges on the unique j-k-path of length at most $\bar{d}$ are built.

We use variables x_e, y_j, and z_k as introduced before, and add assignment variables $r_{jk} \in \{0,1\}$ indicating that demand node k is served by facility j. Let $P_{jk} \subseteq E$ denote the set of edges on the unique path between $k \in N$ and facility $j \in F$. For facilities j that exceed the length bound to node k, i.e. $d(k,j) > \bar{d}$, we can set $r_{jk} = 0$ and consider for each demand node k its set of reachable

facilities as $F_k \subseteq F$.

$$\max \quad \sum_{k \in N} h_k z_k \tag{8}$$

$$\text{s.t.} \quad \sum_{j \in F} c^F_j y_j + \sum_{e \in E} c^E_e x_e \leq \bar{b} \tag{9}$$

$$r_{jk} \leq y_j \qquad \forall j \in F_k, \forall k \in N \tag{10}$$

$$r_{jj} = y_j \qquad \forall j \in F \tag{11}$$

$$r_{jk} \leq x_e \qquad \forall e \in P_{jk}, \forall j \in F_k, \forall k \in N \tag{12}$$

$$z_k = \sum_{j \in F_k} r_{jk} \qquad \forall k \in N \tag{13}$$

$$r_{jk} \geq 0 \qquad \forall j \in F_k, \forall k \in N \tag{14}$$

$$x_e \in \{0,1\} \ \forall e \in E, \quad y_j \in \{0,1\} \ \forall j \in F, \quad z_k \in \{0,1\} \ \forall k \in N. \tag{15}$$

Objective, budget, and integrality constraints (8)–(9), (15) are as in the high-level formulation. Constraint (10) requires the facility to be open for an assignment, (11) ensures an open facility serves itself, (12) enforces that all path edges are built, and (13) links assignments to coverage. Note that, in (15), the integrality of r_{jk} has been relaxed to non-negativity by a $\pm\epsilon$-argument that shows that r_{jk} cannot be fractional at an extreme point.

4 Dynamic programming approach

We develop a dynamic programming (DP) decomposition framework for the MCNDP. The core idea is to decompose the graph along a tree structure and solve subproblems bottom-up, combining their solutions via budget allocation and coverage interactions at interface points. On trees (Section 4.2), each node constitutes a simple subproblem with a small number of binary decisions. On general graphs with limited connectivity (Section 4.3), the block-cut tree provides the decomposition, and each biconnected component becomes a subproblem solved via MILP. Unlike the classic p-median DP on trees, our budget constraint is of knapsack type, leading to pseudo-polynomial time complexity (Martello and Toth 1990). Table 1 summarizes the key differences between the two graph classes.

4.1 Border conditions and DP states

We describe the DP abstractly, on a rooted *decomposition tree* whose nodes are the subproblems. It has two instantiations: the rooted binary tree of Section 4.2, where each subproblem is a single graph node v, and the rooted block-cut tree of Section 4.3, where each subproblem is a block B.

A subproblem S is therefore a set of vertices—a single vertex $\{v\}$ on trees, or the vertices of a block B on BC-trees—equipped with a set of *child subproblems*

Component	DP on trees (Sec. 4.2)	DP on BC-trees (Sec. 4.3)
Subproblem	Single node v	Biconnected component B
Tree structure	Rooted binary tree	Rooted block-cut tree
Interface points	Every node	Articulation points
Budget allocation	Split between children v_1, v_2	Split between B and children
Relevant radii	$\mathcal{O}(\min(n, \bar{d}))$ values	$\mathcal{O}(\bar{d})$ values
Local decisions	Open facility? Upgrade edges?	Solve MCNDP on B (NP-hard)

Table 1: Comparison of the DP framework on trees and block-cut trees. On trees, each subproblem has a small number of binary decisions.

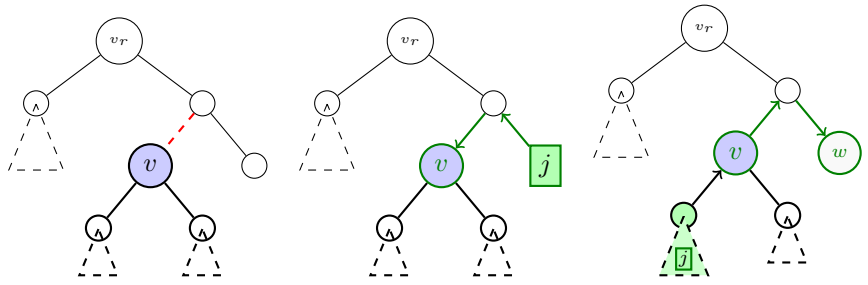
$\mathcal{C}_S$ and a single *interface point* connecting it to its parent: the parent edge on trees, or the parent articulation point on BC-trees. We write T_S for the *subtree rooted at* S, the subgraph induced by S together with all its descendants in the decomposition tree. On trees this is the usual subtree T_v; on BC-trees it is the subgraph formed by the union of all blocks in the subtree of the block-cut tree rooted at S.

The DP value $V(S, b, r)$ is the maximum coverage achievable within the *entire* subtree T_S, under budget $b \in \{0, \ldots, \bar{b}\}$ and a *signed covering condition* $r \in \{-\bar{d}, \ldots, -1, 0, 1, \ldots, \bar{d}\}$ at the interface point, whose sign indicates the direction of coverage flow and whose magnitude indicates its reach. Local decisions inside S (opening facilities, upgrading edges) must conform to the local budget and the covering condition, while the child subtrees enter recursively through the budget allocated to each child and the choice of child states. The three sign cases of r are:

- $r = 0$ (*BLANK state*): the two sides are decoupled at the interface; no coverage flows through it. At the root, this is the only relevant condition, and $V(\text{root}, \bar{b}, 0)$ yields the optimal value.
- $r > 0$ (*OUT state*): an external facility at the parent side provides coverage with remaining radius r at the interface point, i.e., nodes in S within distance r of the interface point are covered for free.
- $r < 0$ (*INT state*): the subproblem must provide coverage to the parent side; some facility in S must be within distance $\bar{d} - |r|$ of the interface point, or, equivalently, nodes on the parent side with maximum distance $|r|$ can be covered.

The three cases are illustrated in Figure 1 (drawn for the tree case; the block-cut tree case is analogous, with the node v replaced by a block B). Intuitively, large positive r makes abundant external coverage available to S, whereas large negative r imposes a tight outward-coverage obligation that may not always be feasible. This generalizes the two-state framework of Megiddo et al. (1983), which already distinguishes OUT and INT states (there, for the maximum covering problem with p facilities); here, the signed index unifies all three cases into a single function V that is monotone in r. In what follows, we use the sign

of r and the state names interchangeably, preferring the names when it aids readability (e.g., "the OUT case" for $r > 0$).



(a) $V(v, b, 0)$ **(b)** $V(v, b, r),\ r > 0$ **(c)** $V(v, b, r),\ r < 0$

Figure 1: The three cases of the signed-radius DP value $V(v, b, r)$ on trees. **(a)** $r = 0$: no covering interaction (BLANK). **(b)** $r > 0$: an external facility j (OUT). **(c)** $r < 0$: an internal facility in T_v (INT) can cover external nodes w.

Not every value $r \in \{-\bar{d}, \dots, \bar{d}\}$ needs to be considered: we denote by $\mathcal{R}_S$ the set of *relevant signed radii* at the interface of S. This set is obtained by pre-processing with the goal to only contain the distances at which coverage can actually be provided or received (for instance, r is not required in $\mathcal{R}_S$ if no path of length $|r|$ from the interface point to another node exists).

High-level recursion. For a subproblem S with child problems $\mathcal{C}_S$ and parent radius r, the DP optimizes over two types of decisions: x, encoding local decisions inside S, and for each child $C \in \mathcal{C}_S$ a selection $\lambda^C = (b_C, r_C) \in \Lambda_C$, determining the budget allocation and covering condition imposed on C. Here, $\Lambda_C \subseteq \{0, \dots, \bar{b}\} \times \mathcal{R}_C$ denotes the set of feasible child states. The recursion is:

$$V(S, b, r) \;=\; \max_{x,\,\lambda} \Bigg(\underbrace{h_S(x, \lambda)}_{\text{coverage inside } S} \;+\; \underbrace{\sum_{C \in \mathcal{C}_S} V(C, b_C, r_C)}_{\text{child contributions}} \Bigg), \tag{16}$$

where the child terms $V(C, b_C, r_C)$ are precomputed values that themselves represent the optimal coverage of their entire respective subtrees. Thus, $V(S, b, r)$ accounts for coverage across the full subtree rooted at S, even though only the decisions in S and the allocation to children are optimized at this level. Note also that the local contribution $h_S(x, \lambda)$ depends on the child radii as well, since a child with $r_C < 0$ can provide coverage to S whereas a child with $r_C > 0$ means that S needs to provide coverage.

Monotonicity. The DP values are monotonically non-decreasing in both arguments:

$$V(S, b, r) \;\le\; V(S, b', r') \qquad \text{whenever } b \le b' \text{ and } r \le r'.$$

Intuitively, a larger budget cannot hurt, and a larger signed radius either relaxes an obligation (when $r < 0$) or provides more free coverage (when $r > 0$). Monotonicity allows pruning the state set Λ_C: any state (b_C, r_C) that is dominated, i.e., another state $(b'_C, r'_C) \in \Lambda_C$ achieves at least the same DP value while consuming no more budget ($b'_C \leq b_C$) and imposing a weaker covering condition ($r'_C \leq r_C$, hence more favorable to the parent), can be removed without loss of optimality, since it would never be required in an optimal parent solution.

4.2 DP on trees

Rooted tree and binary tree transformation. We root the tree G at an arbitrary node v_r and process subtrees bottom-up. For simplicity of exposition, we assume that G is a binary tree (each node has at most two children). Any tree can be transformed into a binary tree by inserting at most n dummy nodes without facility and demand (i.e., $c_v^F = \infty$ and demand weight $h_v = 0$ for such a node v); this transformation preserves optimality (Tamir 1996).

DP states. For each node v, the subproblem is the subtree T_v, and the interface point is the parent edge e. More specifically, a positive $r > 0$ means the remaining coverage distance entering e from the parent side, while a negative $r < 0$ means the remaining coverage entering e from v. The set of considered signed radii at node v derives from the distances to reachable nodes outside T_v,

$$\mathcal{R}_v \;=\; \left\{ \pm(\bar{d} - d_G(v,u)) \;\middle|\; u \in N \setminus T_v,\; d_G(v,u) < \bar{d} \right\} \cup \{0\},$$

with $|\mathcal{R}_v| \in \mathcal{O}(\min(|N|, \bar{d}))$.

Base case (leaves). At a leaf v with a parent edge e with length ℓ_e, the only decision is whether to open a facility. If $r \geq \ell_e$, then v is already covered by the parent side for free. Infeasible states (no facility or insufficient budget) in the $r < 0$ case evaluate to $-\infty$.

$$V(v,b,r) \;=\; \begin{cases} h_v & \text{if } r \geq \ell_e, \\ \begin{cases} h_v & \text{if } v \in F \text{ and } b \geq c_v^F, \\ 0 & \text{otherwise,} \end{cases} & \text{if } 0 \leq r < \ell_e, \\ \begin{cases} h_v & \text{if } v \in F \text{ and } b \geq c_v^F, \\ -\infty & \text{otherwise,} \end{cases} & \text{if } r < 0. \end{cases} \tag{17}$$

Recursive case (inner nodes). Let v be an inner node with children v_1, v_2 and child edges $e_1 = \{v, v_1\}$, $e_2 = \{v, v_2\}$ with upgrade costs $c_{e_1}^E, c_{e_2}^E$. The local decisions are: open a facility at v, upgrade edge e_1, upgrade edge e_2.

We distinguish four coverage situations for v (not covered; covered by a facility at v or outside T_v; covered by the left subtree; covered by the right subtree), and capture each by a helper function that optimizes the budget split and edge-upgrade decisions (see Figure 2). Any helper or child state evaluated with negative remaining budget returns $-\infty$.

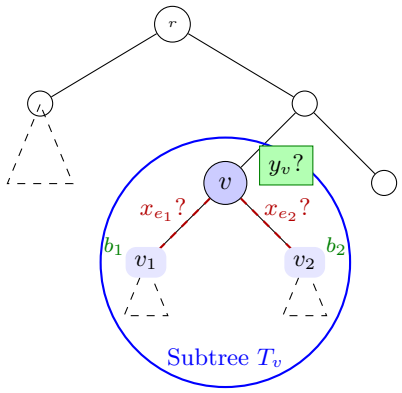


Figure 2: Decisions at an inner node v with children v_1 and v_2. The DP must decide: (i) whether to open a facility at v ($y_v \in \{0,1\}$), (ii) whether to upgrade edges e_1 and e_2 ($x_{e_1}, x_{e_2} \in \{0,1\}$), and (iii) how to allocate the remaining budget between subtrees ($b_1 + b_2 \leq b$). The blue circle indicates the subtree T_v rooted at v.

1. $\mathcal{A}(v,b)$: No coverage at v; no edges upgraded; full budget b distributed.

$$\mathcal{A}(v,b) := \max_{b_1+b_2\leq b} \Big[V(v_1,b_1,0) + V(v_2,b_2,0)\Big] \tag{18}$$

2. $\mathcal{B}(v,b,r)$: v is covered by a facility at v or through the parent edge and can pass a remaining coverage of r to the children.

$$\mathcal{B}(v,b,r) := \max_{\substack{b_1+b_2\leq b \\ x_{e_1},x_{e_2}\in\{0,1\}}} \Big[V(v_1,b_1-c^E_{e_1}x_{e_1},rx_{e_1}) + V(v_2,b_2-c^E_{e_2}x_{e_2},rx_{e_2})\Big] \tag{19}$$

Depending on the edge upgrade decisions, $\mathcal{B}$ calls V for the child trees with adjusted budget and with covering condition either r (if the edge is upgraded) or 0 (if not), since coverage can only flow through an upgraded edge.

3. $\mathcal{C}^1(v,b,r)$: Left subtree provides coverage of $\rho \geq r + \ell_{e_1}$ through e_1 into v such that a remaining coverage of at least r to the parent side or right subtree is available.

$$\mathcal{C}^1(v,b,r) := \max_{\substack{b_1+b_2\leq b-c^E_{e_1} \\ \rho\in\mathcal{R}_{v_1},\ \rho\geq r+\ell_{e_1} \\ x_{e_2}\in\{0,1\}}} \Big[V(v_1,b_1,-\rho) + V(v_2,b_2-c^E_{e_2}x_{e_2},(\rho-\ell_{e_1})x_{e_2})\Big] \tag{20}$$

The left edge is necessarily upgraded, so the maximum is taken only over the right edge upgrade decision as well as the budget and the child radius ρ that the left subtree can provide.

4. $\mathcal{C}^2(v,b,r)$: Symmetric to $\mathcal{C}^1$.

Resulting recursions. The DP values follow by maximizing over the coverage situations, each mirroring the structure of (16) with h_v as the local coverage and the helpers as the child contributions. Depending on whether the parent radius r is sufficient to cover v or instead imposes a covering obligation, a different subset of helpers is active:

$$V(v,b,r) \;=\; \begin{cases} \max\{\; h_v + \mathcal{B}(v, b - c_v^F, \bar{d}),\; h_v + \mathcal{B}(v,b,r-\ell_e), & \text{if } r \geq \ell_e, \\ \qquad h_v + \mathcal{C}^1(v,b,0),\; h_v + \mathcal{C}^2(v,b,0)\;\} & \\ \max\{\; \mathcal{A}(v,b),\; h_v + \mathcal{B}(v, b - c_v^F, \bar{d}), & \text{if } 0 \leq r < \ell_e, \\ \qquad h_v + \mathcal{C}^1(v,b,0),\; h_v + \mathcal{C}^2(v,b,0)\;\} & \\ \max\{\; h_v + \mathcal{B}(v, b - c_v^F, \bar{d}), & \text{if } r < 0. \\ \qquad h_v + \mathcal{C}^1(v,b,|r|),\; h_v + \mathcal{C}^2(v,b,|r|)\;\} & \end{cases} \tag{21}$$

All cases share the first term, which corresponds to opening a facility at v. Coverage from the children via $\mathcal{C}^1$ and $\mathcal{C}^2$ is available in all cases, but only for $r < 0$ must it reach the parent side. Not building a facility and not using child coverage is only possible for $r \geq 0$ and corresponds to the cases $\mathcal{B}(v, b, r - \ell_e)$ and $\mathcal{A}(v, b)$.

Complexity. Let $R := \max_{v \in N} |\mathcal{R}_v| \leq \min(n, \bar{d} + 1)$ denote the maximum number of relevant radii at any node. The DP table for V has $\mathcal{O}(n \cdot R \cdot \bar{b})$ entries, and each entry requires $\mathcal{O}(\bar{b} \cdot R)$ time for enumerating budget allocations and radii in the helper functions. The total time complexity is therefore

$$\mathcal{O}(n \cdot R \cdot \bar{b}) \times \mathcal{O}(\bar{b} \cdot R) = \mathcal{O}(n \cdot \bar{b}^2 \cdot R^2),$$

which is bounded by $\mathcal{O}(n^3 \cdot \bar{b}^2)$ and $\mathcal{O}(n \cdot \bar{b}^2 \cdot \bar{d}^2)$, i.e., pseudo-polynomial.

Implementation In practice, the DP can further be improved by memoizing helper function results and pruning infeasible budget/radius combinations early. A pseudocode summary of the tree DP is given in Appendix E.1. In Section 5, we report computational results of our DP implementation, and show that it is competitive with an MILP solver especially for larger budget and coverage radius values.

4.3 Block-cut tree decomposition

We now extend the DP framework to graphs that are connected but not biconnected.

4.3.1 From trees to graphs with articulation points

We now generalize the tree DP from Section 4.2 to graphs that contain *articulation points* (APs), also known as *cut vertices*: nodes whose removal disconnects the graph. Trees are the extreme case: every internal node is an articulation

point, and the per-block subproblems of this section reduce to the single-node subproblems of the tree DP in Section 4.2. The block-level solutions are combined via the general DP recursion (16) on the block-cut tree defined below.

Block-cut tree structure. A graph $G = (N, E)$ can be decomposed into its *biconnected components*, called *blocks*: maximal subgraphs that remain connected after removing any single node. The blocks are connected to each other only through articulation points, and this relationship forms a tree structure called the *block-cut tree* (BC-tree), which can be computed in linear time (Hopcroft and Tarjan 1973).

Formally, let $\mathcal{B}$ denote the set of blocks and $\mathcal{A}$ the set of articulation points. The BC-tree $\mathcal{T}$ has vertex set $\mathcal{B} \cup \mathcal{A}$, where block B and AP a are adjacent in $\mathcal{T}$ if and only if a belongs to B. We work directly on $\mathcal{T}$; however, DP subproblems are only necessary on the block nodes, since these include their incident articulation points.

We root $\mathcal{T}$ at an arbitrary block B_r, which induces parent-child relationships among blocks. For a non-root block B, we denote by a_B its *parent AP* (the unique AP connecting B to its parent block) and by $\mathcal{C}_B$ its set of *child blocks*. For a child block $C \in \mathcal{C}_B$, its parent AP a_C is also called a *child AP* of B, and $B \cap C = \{a_C\}$. For the demand accounting, each articulation point a_C is assigned to the parent block of C; that is, h_{a_C} is counted in the DP value of the parent (consistent with the sum over $N_B \setminus \{a_B\}$ in the block-level flow formulation).

4.3.2 Border conditions at articulation points

A facility in one block can cover nodes in another, but only through the connecting AP; the interaction therefore reduces to the coverage radius remaining at that AP. The signed-radius DP of Section 4.1 therefore instantiates on the BC-tree by taking blocks instead of single nodes as subproblems: the interface point of a block B is its parent AP a_B, and the signed radius r is exactly the coverage radius remaining at a_B. Consequently, $V(B, b, r)$ is the maximum coverage achievable in the subtree of $\mathcal{T}$ rooted at B.

A block B interacts with its neighbors through its parent AP a_B and its child APs $\{a_C\}_{C \in \mathcal{C}_B}$. Recall from Section 4.1 that the *parent radius* r is a fixed parameter of the DP state, while each *child selection* $\lambda^C = (b_C, r_C) \in \Lambda_C$ is a decision variable. At a_B, r is the covering condition imposed on B by its parent block. At each child AP a_C, λ^C specifies the budget b_C allocated to C together with r_C, the covering condition at a_C that simultaneously serves as C's parent radius (in $V(C, b_C, r_C)$) and constrains B's own coverage flow through a_C. These selections are chosen jointly with the local decisions inside B.

The sign of the radius encodes a duality in its effect on B: a positive parent radius ($r > 0$) and a negative child radius ($r_C < 0$) both represent free external coverage arriving at the respective AP (with remaining radius r or $|r_C|$, respectively), whereas a negative parent radius ($r < 0$) and a positive child radius

($r_C > 0$) both impose an obligation on B to provide coverage outward through that AP.

Auxiliary-node interpretation. It is useful to represent each signed-radius border condition as an auxiliary node at the articulation point, following the duality above: a condition providing free external coverage into B (parent $r > 0$, child $r_C < 0$) is represented by a *virtual facility* at distance $\bar{d} - r$ resp. $\bar{d} - |r_C|$ from the AP; a condition obliging B to cover outward (parent $r < 0$, child $r_C > 0$) by a *virtual demand node* at distance $|r|$ resp. r_C, which B must cover; the BLANK conditions ($r = 0$, $r_C = 0$) add no node, decoupling the blocks at the AP. In all cases, the node is placed so that the magnitude of the radius is exactly the coverage distance remaining at the AP. Since a child radius is signed from C's perspective (it is C's parent radius), the same sign yields opposite node types at the parent and child APs. This interpretation is illustrated in Figure 3. Adding the auxiliary nodes of the active border conditions to the block instance yields the *auxiliary instance* of B; Table 2 lists the auxiliary node for each of the four non-BLANK conditions. This construction directly guides the MILP extensions in Section 4.3.4.

Table 2: The auxiliary instance: one added node with its connecting arc per non-BLANK border condition (the BLANK cases $r = 0$, $r_C = 0$ add nothing). The parent nodes exist only in the DP states of matching sign, since the parent radius r is a fixed parameter; the child nodes are present for every candidate state and activated by the child selection $\lambda^C = (b_C, r_C)$. For the child OUT condition, a single node d_C per child suffices, its arc carrying the selected radius as its length (Section 4.3.4).

Border condition	Auxiliary node	Connecting arc	Length	Exists / active
parent OUT ($r > 0$)	virtual facility w_B	(w_B, a_B)	$\bar{d} - r$	parent states $r > 0$ only; free to use
child INT ($r_C < 0$)	virtual facility $w_C[r_C]$	$(w_C[r_C], a_C)$	$\bar{d} - \lvert r_C \rvert$	usable iff λ^C selects radius r_C
child OUT ($r_C > 0$)	virtual demand node d_C	(a_C, d_C)	selected radius r_C	covered iff λ^C selects $r_C > 0$
parent INT ($r < 0$)	virtual demand node d_B	(a_B, d_B)	$\lvert r \rvert$	parent states $r < 0$ only; must be covered

4.3.3 Computing block DP values via MILP

The DP recursion (16) requires, for each block B, optimizing over the local decisions inside B jointly with the choice of budget and covering condition for every child block. Since the number of child state combinations grows exponentially in the number of children, explicit enumeration and computation would be brute force and unlikely to perform well. Instead, we formulate the recursion (16) as a MILP that incorporates the unified formulation (1) for block B, extended with *selector variables* that pick a precomputed child state per child block and with a parameterized feasibility set $\mathcal{F}$ encoding the border conditions at the articulation points.

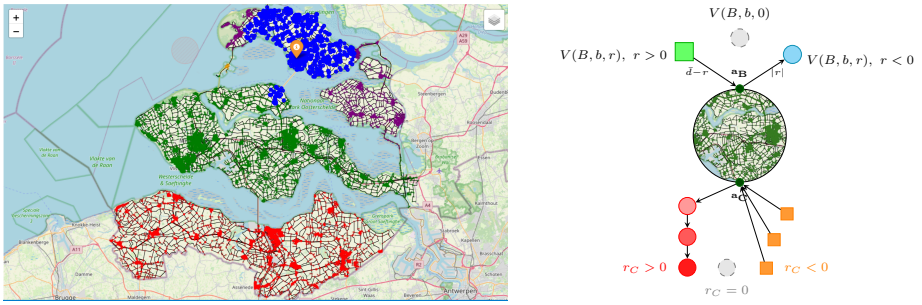


Figure 3: Border conditions as auxiliary nodes. *Left:* The Dutch province of Zeeland decomposed into three blocks; a facility in the northern block (orange pin, blue coverage area) reaches the central block through the AP, yielding a parent condition $r > 0$ there. *Right:* The auxiliary nodes at the parent AP a_B (top) and a child AP a_C (bottom), cf. Table 2: a virtual facility for coverage into the block ($r > 0$; $r_C < 0$), a virtual demand node for an outward obligation ($r < 0$; $r_C > 0$), and no node for BLANK ($r = 0$; $r_C = 0$). The parent radius r is fixed per DP state, whereas at each child AP the candidate radii $r_C \in \mathcal{R}_C$ appear as alternative nodes, of which the selection λ^C activates one. Map data from OpenStreetMap, available under the Open Database License; see https://www.openstreetmap.org/copyright.

Block-level MILP formulation. Let N_B, E_B, and $\overline{E}_B = E_B \cup \{\{s, j\} \mid j \in F_B\}$ denote the nodes, edges, and extended edges (including facility edges) of block B. For each child $C \in \mathcal{C}_B$, we introduce *selector variables* $\lambda^C_{b',r'} \in \{0,1\}$ indexed over the (pruned) state set $\Lambda_C \subseteq \{0,\dots,\bar{b}\} \times \mathcal{R}_C$ from Section 4.1. Setting $\lambda^C_{b',r'} = 1$ selects the precomputed DP value $V(C, b', r')$.

The block-level MILP for computing $V(B, b, r)$ is:

$$\max \quad \sum_{k \in N_B \setminus \{a_B\}} h_k z_k \;+\; \sum_{C \in \mathcal{C}_B} \sum_{(b',r') \in \Lambda_C} \lambda^C_{b',r'} \cdot V(C, b', r') \tag{22}$$

$$\text{s.t.} \quad \sum_{e \in \overline{E}_B} c^E_e x_e \;+\; \sum_{C \in \mathcal{C}_B} \sum_{(b',r') \in \Lambda_C} b' \cdot \lambda^C_{b',r'} \;\le\; b \tag{23}$$

$$\sum_{(b',r') \in \Lambda_C} \lambda^C_{b',r'} \;=\; 1 \qquad \forall\, C \in \mathcal{C}_B \tag{24}$$

$$(x, z) \;\in\; \mathcal{F}(r, \{\lambda^C\}_{C \in \mathcal{C}_B}) \tag{25}$$

$$x_e \;\in\; \{0,1\} \qquad \forall\, e \in \overline{E}_B, \tag{26}$$

$$z_k \;\in\; \{0,1\} \qquad \forall\, k \in N_B \tag{27}$$

$$\lambda^C_{b',r'} \;\in\; \{0,1\} \qquad \forall\, C \in \mathcal{C}_B,\ (b', r') \in \Lambda_C \tag{28}$$

The objective (22) maximizes coverage inside B (excluding the parent AP a_B, whose demand is accounted for in the parent block) plus the precomputed DP values of the selected child states. Constraint (23) ensures the total budget

(block plus children) does not exceed b. Constraint (24) ensures exactly one state is selected per child block. The feasibility set $\mathcal{F}(r, \{\lambda^C\})$ in (25) extends (1) by encoding the parent radius r and the selected child radii at the articulation points; we describe its implementation in Section 4.3.4.

Remark 1 (Selector subproblem is small and fast to solve)**.** *Fixing the local design to incumbent values $(\hat{x}, \hat{z})$ collapses the border-condition encoding into a feasibility filter on the child states, excluding those incompatible with the chosen design and typically leaving only a handful of candidates per child. The residual problem in λ is essentially*

$$\max_{\lambda} \Big(h_B(\hat{x}, \lambda) \; + \sum_{C \in \mathcal{C}_B} l(C, \lambda^C) \Big), \tag{29}$$

where $h_B(\hat{x}, \lambda)$ is the local coverage of the incumbent at child allocation λ and $l(C, \lambda^C)$ a lower bound on $V(C, \lambda^C)$ (Section 4.3.5). This is a multiple-choice knapsack problem *(MCKP) (Martello and Toth 1990) with at most $\bar{d} \cdot \bar{b}$ candidate states per child, hence small and fast to solve compared with the network-design part.*

Preliminary experiments support this: across a sample of block solves, re-solving the MCKP alone from the incumbent's $(\hat{x}, \hat{z})$ recovered that incumbent's λ in well under a second whenever the block MILP had stopped at its time limit. We adopt this as an assumption in our bound analysis (Section 4.3.5; details in Appendix C).

4.3.4 Extending the flow formulation

We realize the feasibility set $\mathcal{F}(r, \{\lambda^C\})$ in (25) by solving the block MILP on the auxiliary instance of Section 4.3.2 (Table 2). Since a node is covered if and only if a coverage path of length at most $\bar{d}$ connects it to an opened facility via built edges (Section 3.1), the auxiliary nodes extend exactly the two sides of this characterization: virtual facilities are further origins of coverage paths, virtual demand nodes further destinations that must be reached. The covering constraints therefore apply unchanged on the auxiliary instance; this holds for the flow formulation as for any other block-level formulation stated on the block instance.

What remains is to couple the auxiliary nodes to the DP states. The parent condition is fixed per block MILP, so its node is simply present or absent; only the child choice needs encoding. The MILP carries the auxiliary nodes of *all* candidate child states $(b', r') \in \Lambda_C$, and linking constraints tie them to the selectors $\lambda^C_{b',r'}$: a virtual facility is usable only if its state is selected, and a virtual demand node must be covered exactly if an OUT state of its child is selected. In the flow formulation, these linking constraints amount to a handful of extra flow variables and variable upper bounds.

Finally, the flow formulation spares us from constructing the auxiliary instance in its entirety. Conceptually, each candidate OUT radius of a child

contributes its own virtual demand node; since exactly one state is selected per child, they merge into the single node d_C, the length of its connecting arc becoming a linear expression in the selectors λ^C that evaluates to the selected radius. The block MILP therefore grows with the number of children rather than with the number of admissible child radii. Appendix B makes this construction precise and derives the resulting master MILP.

4.3.5 Propagating bounds

Under time limits, not every block MILP is solved to optimality. For each block B and state (b, r) we therefore maintain bounds $l(B, b, r) \leq V(B, b, r) \leq u(B, b, r)$ with gap $\Delta(B, b, r) := u(B, b, r) - l(B, b, r)$, and propagate them up the block-cut tree through the recurrence (16).

The absolute gap necessarily grows up the tree, since each block's bound aggregates those of its children and the subproblem itself grows with the subtree. The gap *relative* to the subproblem size, however, does not accumulate. Assuming the selector subproblem (29) is solved exactly (Remark 1), an inductive argument shows that if each block returns a primal design within factor $\alpha_B \in (0, 1]$ of its local optimum, the global approximation ratio is at least $\min_B \alpha_B$: the relative error is dictated by the weakest local solve, independent of tree depth. Formal statements (Propositions 1, 2, Corollary 1) and proofs are in Appendix C.

Bounds from neighboring states. Monotonicity (Section 4.1) gives a second, cheap source of bounds. When only the BLANK state $V(B, b, 0)$ has been solved, the OUT state at radius $r > 0$ exceeds the BLANK by at most the demand reachable from the parent AP a_B within distance r,

$$V(B, b, r) \;\leq\; V(B, b, 0) + w^B(a_B, r),$$

where $w^B(a_B, r)$ denotes that demand weight. The INT bound for $r < 0$ is analogous and given in Appendix C.

A pseudocode summary of the BC-tree DP, including the warm-starting, dominance pruning, and bound-propagation refinements introduced above, is given in Appendix E.2.

5 Computational experiments

We evaluate the proposed decomposition approach in two settings of increasing complexity. First, on tree graphs where the DP solves the MCNDP exactly in pseudo-polynomial time (Section 5.3), we compare against the assignment-based MILP to validate the approach in the simplest setting. Second, on general graphs with articulation points (Section 5.4), we investigate whether embedding a MILP formulation within the BC-tree DP framework improves performance compared to solving the MILP on the full graph. We focus on the flow formulation (Section 3.1), solved both directly and within the DP.

5.1 Experimental setup

The experiments were performed in a Linux computing environment (64-bit). The tree DP experiments ran on Intel Xeon Platinum 8360Y CPUs at 2.40 GHz, the BC-tree experiments on Intel Xeon Gold 6240 CPUs at 2.60 GHz. All runs are single-threaded (Gurobi threads=1) with a 16 GB solver memory limit and a 3600-second time limit. The algorithms are implemented in Python (3.11 for the tree DP, 3.13 for the BC-tree experiments) using NetworkX 3.5 for graph algorithms and Gurobi 12.0 as the MILP solver (Gurobi Optimization, LLC 2026). The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation Flanders (FWO) and the Flemish Government.

5.2 Instances

There are no standard benchmark instances for network design problems on graphs with low connectivity. We describe the instance classes for the tree DP and BC-tree DP experiments separately.

Tree instances. For the tree DP experiments, we sample trees on n nodes uniformly at random among all labeled trees with the `random_labeled_tree` function of NetworkX (Hagberg et al. 2008). Edge distances and upgrade costs are drawn independently from $\{1, \ldots, 10\}$, facility opening costs from $\{3, 6, \ldots, 30\}$, and demand weights from $\{1, \ldots, 10\}$.

BC-tree instances. The Sevilla network García-Archilla et al. (2013) (49 nodes, 119 edges) and Sioux Falls network (Leblanc 1975) (24 nodes, 76 edges) are standard transportation benchmarks but are biconnected, so the BC-tree decomposition yields a single block with no computational benefit. To obtain instances with controlled block-cut tree structure, we construct synthetic graphs by merging biconnected building blocks:

1. **Select base blocks:** full copies or connected subgraphs of Sioux Falls, Sevilla, and grid graphs (4×4, 5×5, 6×6).
2. **Sample a tree topology Θ on k blocks** (star, path, balanced tree, or random tree), where each node represents a block taken from base blocks and each edge indicates a shared articulation point.
3. **Merge via node identification:** for each edge in Θ, identify (i.e., shrink) one randomly chosen node from each adjacent block, creating an articulation point.

Table F.1 in Appendix F summarizes the resulting 34 instances (51–337 nodes, 76–833 edges, 2–34 blocks). Figure 4 illustrates a representative instance.

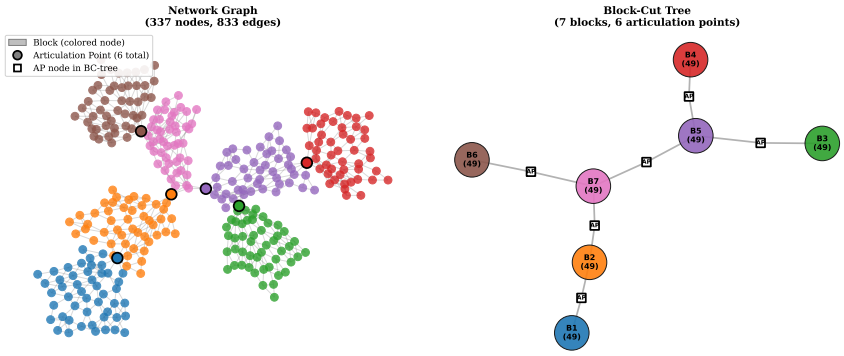


Figure 4: Example BC-tree instance with balanced tree topology: 7 blocks of 49 nodes each, connected in a path-like structure (Max AP° = 2). The network graph (left) and block-cut tree structure (right) are shown, with block nodes colored consistently.

Instance parameter generation. Edge distances are uniform (all equal to 1). Demand weights are drawn from $\{0, 1, 2, 3, 4\}$ with a fixed seed. Facility opening costs are drawn from $\{5, 10, 15\}$, and edge upgrade costs from $\{1, 2, 3, 4, 5\}$. The budget $\bar{b} = \beta \cdot b_{\text{ref}}$ is a fraction of a reference budget estimated to cover the entire graph, with budget factor $\beta \in \{0.3, 0.5, 0.7\}$. The coverage radius is

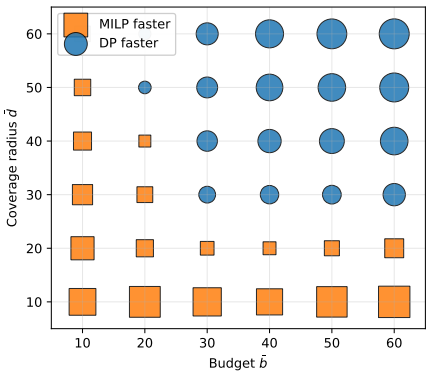
$$\bar{d} = \lfloor \gamma \cdot m \cdot \bar{\ell} \rfloor, \tag{30}$$

where m is the number of edges, $\bar{\ell}$ the average edge length, and $\gamma \in \{0.05, 0.10, 0.15\}$ the coverage radius factor. Intuitively, this is the fraction of the network's total edge length that a single facility-to-demand path may traverse. This yields coverage radii ranging from 3 to 124 across instances.

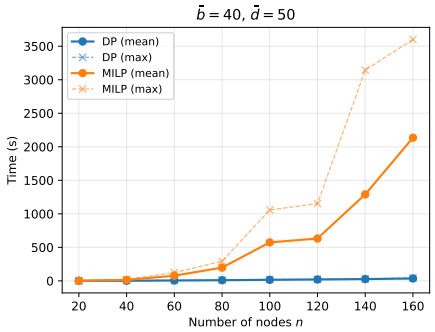
5.3 Results on trees

We evaluate the tree DP algorithm from Section 4.2 against solving the MILP formulation directly with Gurobi. For each $(\bar{b}, \bar{d}) \in \{10, 20, \ldots, 60\}^2$ we generate 9 random 20-node trees (324 test cases), and additionally study scaling at $n \in \{20, 40, \ldots, 160\}$ for three fixed parameter settings, with eight trees per size (192 further test cases). Both algorithms are exact, and on all 516 test cases they return the same objective value, which confirms the correctness of the DP implementation.

For moderate parameters the MILP is faster, though both algorithms finish quickly there: over the eighteen parameter combinations that the MILP wins, it averages 0.11 s against the DP's 0.21 s, and no solve exceeds 1.5 s. As $\bar{b}$ and $\bar{d}$ grow, the DP's pseudo-polynomial $\mathcal{O}(n\bar{b}^2\bar{d}^2)$ runtime becomes competitive: once $\bar{b} \geq 30$ and $\bar{d} \geq 40$, the DP is consistently faster (Figure 5a), by a median factor of 3.0. The margin widens with instance size (Figure 5b): at $\bar{b} = 40$, $\bar{d} = 50$ it



(a) Faster algorithm by $(\bar{b}, \bar{d})$ on 20-node trees; marker area scales with $|\log_2(t_{\text{MILP}}/t_{\text{DP}})|$.



(b) Scaling at $\bar{b} = 40, \bar{d} = 50$: at $n = 160$ the MILP averages over 2100 s, with one of the eight instances unsolved at the 3600 s limit, while the DP completes in about 36 s (max 46 s).

Figure 5: Tree DP vs. direct MILP. The DP dominates once both $\bar{b} \geq 30$ and $\bar{d} \geq 40$ (left), and the advantage widens with instance size (right).

reaches a median factor of 66 at $n = 160$, where one MILP run hit the 3600 s limit unsolved on a tree that the DP had settled in 46 s. The MILP also exhibits high variance across instances, with a coefficient of variation of 0.55 over the nine trees of a parameter combination against the DP's 0.21. This predictability is a practical advantage beyond raw speed. Detailed sensitivity at $n = 20$ and the two lower-parameter scaling settings ($\bar{b} = 20$, $\bar{d} = 40$ and $\bar{b} = 30$, $\bar{d} = 30$) appear in Appendix H. These findings motivate the extension to graphs with low connectivity via the block-cut tree decomposition (Section 4.3), where the DP is applied recursively to biconnected components.

5.4 Results on graphs with articulation points

We evaluate the BC-tree DP framework on graphs with articulation points, where it coordinates MILP solves on individual blocks, instead of solving a single MILP on the full graph.

5.4.1 Implementation choices

Several implementation choices affect the performance of the BC-tree DP framework. Compared with solving only the direct MILP, the decomposition offers more opportunities for performance improvements through careful design. We summarize the key decisions based on preliminary experiments and refer to Appendix D for details.

Warm-starting and state ordering. Each block MILP is initialized with a greedy heuristic solution. We further exploit monotonicity across neighboring

DP states: a solution for state $V(B, b, r)$ provides a valid warm start for $V(B, b+1, r)$ and $V(B, b, r+1)$. States are processed with budget increasing in the outer loop and radius in the inner loop.

Structural choices. The BC-tree is rooted at the largest block, which is solved only for the BLANK condition with full budget $\bar{b}$. Adjacent blocks with at most three vertices are merged to reduce DP overhead. We also filter clearly infeasible budget-distance combinations using combinatorial properties of edge costs and path lengths.

Time allocation. When a total time limit is imposed, time is distributed across blocks proportionally to edge count, then allocated adaptively across DP states within each block so that unused time from faster solves accumulates for harder states. States receiving insufficient time are skipped, with bounds constructed from dominating states and heuristics (Section 4.3.5).

5.4.2 Computational comparison

We evaluate whether the BC-tree decomposition yields computational benefits compared to solving a single MILP on the full graph. We use the 34 multi-block instances from Table F.1, each solved with three budget factors ($\beta \in \{0.3, 0.5, 0.7\}$) and three coverage radius factors ($\gamma \in \{0.05, 0.10, 0.15\}$), yielding $34 \times 3 \times 3 = 306$ test cases. All algorithms use Gurobi 12.0.3 with a one-hour time limit.

Algorithm selection via performance profiles. To select the MILP formulation presented here, we compared two MILP formulations for the MCNDP subproblems: the flow-based model (Section 3.1) and an alternative based on length-bounded cuts (Arslan et al. 2020).

We compare four configurations: the direct MILP with warm-starting using either formulation (FlowMIP, LBCutMIP), and the BC-tree DP with either formulation (DP-FlowMIP, DP-LBCutMIP). We use performance profiles (Dolan and Moré 2002) on solution quality to compare the configurations: for each algorithm, the profile reports the fraction of instances solved within a factor τ of the best objective found by any algorithm. Figure 6 shows the result.

FlowMIP and DP-FlowMIP clearly dominate and will be the focus in the following: at $\tau = 1.05$, both solve nearly all instances well (96.4% and 98.7%). The length-bounded cut formulation is clearly outperformed by the flow formulation; nonetheless, embedding it in the BC-tree DP (DP-LBCutMIP) still improves over the direct LBCutMIP, confirming the value of the decomposition.

Overall performance. Across all 306 test cases, the DP finds better solutions in 98 cases (32.0%), the MILP in 57 (18.6%), with 151 equivalent (49.3%). The mean objective difference is 0.43% in favor of the DP.

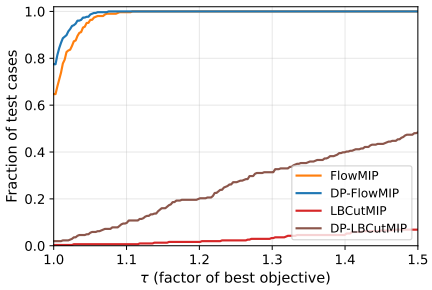


Figure 6: Solution quality performance profile. At $\tau = 1$: DP-FlowMIP achieves best in 75.2%, FlowMIP in 62.7%.

In solution quality, the DP shows robustness: when it is worse than FlowMIP, it remains within 2% in most instances, exceeding this threshold in only 26 cases compared to 51 for FlowMIP (see Figure 7(a)). FlowMIP solves 26.1% of instances to optimality versus 17.6% for DP-FlowMIP (see also the time-based performance profile in Appendix I). In the 54 cases where both are optimal, FlowMIP is faster on average (307 seconds versus 521 seconds for the DP). However, in the 252 instances not solved to optimality, the DP uses an average of 2983 seconds, leaving approximately 617 seconds of the 3600-second limit unused.

We compared performance across several metrics (BC-tree topology, largest block size, network size) but found no strong differentiating factors; the DP's solution quality advantage is broadly consistent, suggesting it stems from warm-starting and budget allocation rather than specific structural properties. A more nuanced picture emerges from the sensitivity to budget and coverage radius parameters, discussed next.

β	DP wins	FlowMIP wins	Equiv	Mean DP adv. (%)
0.3	26	28	48	−0.13
0.5	35	15	52	+0.82
0.7	37	14	51	+0.59
γ	DP wins	FlowMIP wins	Equiv	Mean DP adv. (%)
0.05	30	26	46	+0.03
0.10	36	16	50	+0.68
0.15	32	15	55	+0.58

Table 3: Algorithm comparison by budget factor β (top) and coverage radius factor γ (bottom). Larger values of both parameters favor the BC-tree DP.

Sensitivity to budget and coverage radius. At tight budget ($\beta = 0.3$), the MILP and DP are roughly tied, while at $\beta \geq 0.5$ the DP shows a consistent

advantage of ∼0.7% (Table 3). Similarly, higher coverage radius favors the DP (32 vs. 15 wins at $\gamma = 0.15$), consistent with the tree results from Section 5.3: longer radii mean more paths through articulation points, making border condition aggregation more effective. Figure 7(b) shows the same pattern as a heatmap over $\beta \times \gamma$.

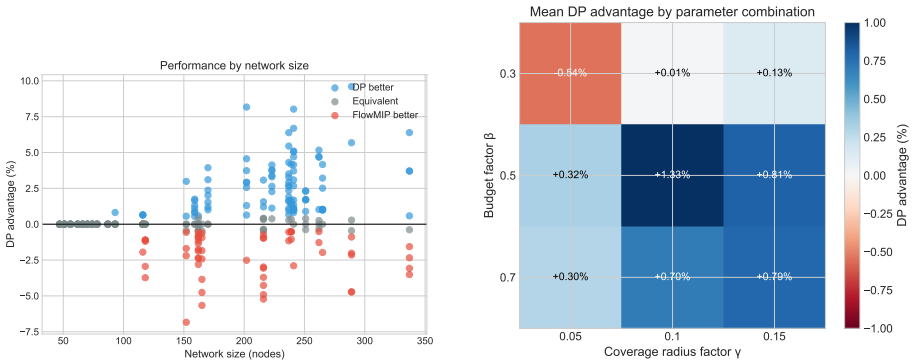


Figure 7: (a) DP advantage over FlowMIP by network size: red indicates DP advantage >0.5%, blue FlowMIP advantage, gray equivalence. (b) Mean DP advantage by budget factor and coverage radius factor.

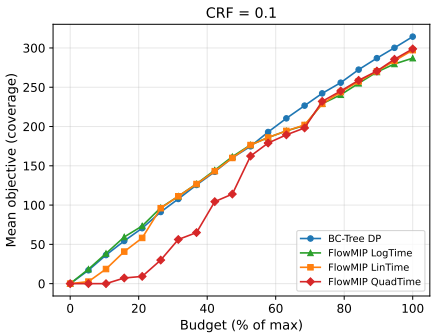
Practical solve scenarios. Two settings common in practice highlight the BC-tree DP's structural advantages. We evaluate both on the five largest instances that did not reach optimality within one hour, since these are the cases where solver choice actually matters; full experimental details are reported in Appendix G.

First, decision-makers often need a full coverage-versus-budget curve rather than a solution for a single budget value. Because the BC-tree DP enumerates budget states intrinsically, all curve points emerge from a single solve, so each budget value benefits from the full time budget. FlowMIP, by contrast, must be re-solved for each budget level, and the available time has to be split across points. We compare three time-allocation strategies that distribute time proportionally to the logarithm (*FlowMIP-LogTime*), linear value (*FlowMIP-LinTime*), or square (*FlowMIP-QuadTime*) of the budget. Figure 8(a) shows the resulting curves at $\gamma = 0.10$: the BC-tree DP dominates the FlowMIP variants at every budget value, with QuadTime degrading sharply at small budgets, where the time allotted per point is exhausted before a useful incumbent is found.

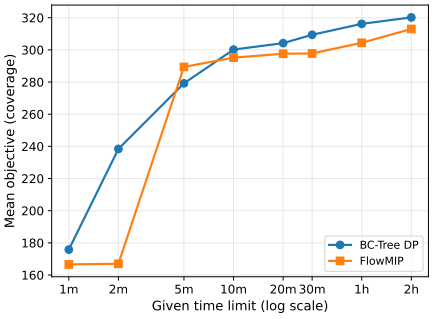
Due to its advantage on small budgets, FlowMIP-LogTime is the most competitive allocation strategy, yet is still outperformed by the DP by 1.9% on average across all budget levels. Restricting the analysis to the practically more relevant upper half of the budget range, a more consistent advantage emerges: the DP achieves mean relative gains of 4.5–6.7% against all three strategies

(6.7% vs. LogTime, 4.5% vs. LinTime, 5.4% vs. QuadTime).

Second, decision-making scenarios differ widely in how much computation time they allow, from minutes for time-sensitive choices to hours for long-term planning. We therefore evaluate solution quality across a range of total time budgets, rather than only at a single time limit. Figure 8(b) reports this *time-limit sensitivity* at the default budget. The BC-tree DP delivers better solutions than FlowMIP across nearly the entire range, with one narrow sweet spot at very small total runtimes: there, the MILP subproblems are small enough that FlowMIP finds a decent first incumbent before the DP has amortized its state-enumeration overhead. Beyond roughly ten minutes, the DP overtakes and the quality gap widens steadily.



(a) Coverage–budget curve at $\gamma = 0.10$.



(b) Time-limit sensitivity at the default budget.

Figure 8: Two practical solve scenarios on the five largest instances not solved to optimality within one hour. The BC-tree DP enumerates all budgets within a single solve, whereas the FlowMIP variants split the time limit across budget points by a log-, linear-, or quadratic-in-budget allocation.

6 Conclusions

We introduced a decomposition framework for the Maximum Covering Network Design Problem on graphs with low connectivity, centered on a block-cut tree dynamic program that exploits articulation points to decompose the problem into independent subproblems on biconnected components.

For tree graphs the resulting pseudo-polynomial DP solves the MCNDP exactly and outperforms direct MILP formulations on larger budgets and coverage radii, with the added advantage of consistent runtimes without outliers. For graphs with articulation points, the BC-tree DP coordinates MILP solves on the individual blocks and matches or outperforms direct MILP on over 80% of our test instances, with an advantage that grows with the budget and coverage radius factors; where it does fall behind, it typically stays within 2% of the direct MILP, exceeding that threshold in only 26 cases versus 51 for FlowMIP.

It is particularly well suited to two practical settings, producing full coverage–budget curves at essentially no extra cost over a single solve and dominating FlowMIP across a wide range of total time limits. Its key practical strength is reducing problem sizes through decomposition, and further improvements in bounding and state iteration could yield larger advantages.

Limitations. On 2-connected graphs without articulation points the BC-tree method reduces to a single-block solve, equivalent to direct MILP with neither advantage nor disadvantage. On less connected instances the per-instance gain may be modest, but the computational study shows that the BC-tree DP is preferable in the majority of cases.

Future directions. The framework extends along two axes. First, it is not specific to the MCNDP: the same selector-and-border-condition scheme applies to other location and network design problems in which the interaction across an articulation point reduces to a facility's remaining reach and a budget allocation, such as uncapacitated facility location and its network design variant (the UFLNDP; Section 2). Second, it extends to richer graph structure: SPQR-tree decompositions would lift it to 2-connected graphs, and standard graph partitioning could handle larger separators (Delling et al. 2011). Algorithmically, block subproblems are naturally multi-objective in budget, coverage, and border conditions, suggesting connections to Lagrangian relaxation (Larsson et al. 2024) or resource-directed decomposition (Bodur et al. 2022, Yıldız et al. 2022). Moreover, a Benders-like scheme could compute child states on demand, and sibling-block independence makes parallelization straightforward, yielding significant performance improvement potential on modern multi-core hardware. Finally, an instance-space analysis (Smith-Miles and Muñoz 2023) could characterize when BC-tree decomposition pays off in practice and guide method selection.

Code and data availability. The implementation, the instance generator and instance files, and the per-case result data referenced in the appendix are available in the authors' public GitHub repository.

Acknowledgments. This work was supported by the special research fund of KU Leuven (project C14/22/026). The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

References

Arslan O, Jabali O, Laporte G (2020) A flexible, natural formulation for the network design problem with vulnerability constraints. *INFORMS Journal on Computing* 32(1):120–134.

Arslan O, Karaşan OE, Mahjoub AR, Yaman H (2019) A Branch-and-Cut Algorithm for the Alternative Fuel Refueling Station Location Problem with Routing. *Transportation Science* 53(4):1107–1125.

Baïou M, Barahona F (2009) On the Integrality of Some Facility Location Polytopes. *SIAM Journal on Discrete Mathematics* 23(2):665–679.

Baldomero-Naranjo M, Kalcsics J, Marín A, Rodríguez-Chía AM (2022) Upgrading edges in the maximal covering location problem. *European Journal of Operational Research* 303(1):14–36.

Baldomero-Naranjo M, Kalcsics J, Rodríguez-Chía AM (2024) On the complexity of the upgrading version of the maximal covering location problem. *Networks* 83(4):627–641.

Bodur M, Ahmed S, Boland N, Nemhauser GL (2022) Decomposition of loosely coupled integer programs: A multiobjective perspective. *Mathematical Programming* 196(1-2):427–477.

Bucarey V, Fortz B, González-Blanco N, Labbé M, Mesa JA (2022) Benders decomposition for network design covering problems. *Computers & Operations Research* 137:105417.

Church R, ReVelle C (1974) The Maximal Covering Location Problem. *Papers in Regional Science* 32(1):101–118.

Contreras I, Fernández E (2012) General network design: A unified view of combined location and network design problems. *European Journal of Operational Research* 219(3):680–697.

Cook ADB, Suresh V, Nair T, Foo YN (2019) Integrating disaster governance in Timor-Leste: Opportunities and challenges. *International Journal of Disaster Risk Reduction* 35:101051.

Cordeau JF, Furini F, Ljubić I (2019) Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research* 275(3):882–896.

Correcher J, Landete M, Peiró J, Yaman H (2025) Decomposition techniques for the generalized vertex cover problem. Technical report.

Crainic TG, Gendreau M, Gendron B, eds. (2021) *Network Design with Applications to Transportation and Logistics* (Cham: Springer International Publishing).

Delling D, Goldberg AV, Razenshteyn I, Werneck RF (2011) Graph Partitioning with Natural Cuts. *2011 IEEE International Parallel & Distributed Processing Symposium*, 1135–1146.

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. *Mathematical Programming* 91(2):201–213.

Feng Z, Song H, Qi X (2024) A novel algorithm for the generalized network dismantling problem based on dynamic programming. *Chaos, Solitons & Fractals* 180:114585.

García-Archilla B, Lozano AJ, Mesa JA, Perea F (2013) GRASP algorithms for the robust railway network design problem. *Journal of Heuristics* 19(2):399–422.

Gurobi Optimization, LLC (2026) Gurobi Optimizer Reference Manual. URL https://www.gurobi.com.

Hagberg AA, Schult DA, Swart PJ (2008) Exploring network structure, dynamics, and function using NetworkX. Varoquaux G, Vaught T, Millman J, eds., *Proceedings of the 7th Python in Science Conference*, 11–15 (Pasadena, CA USA).

Hopcroft J, Tarjan R (1973) Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM* 16(6):372–378.

Landete M, Marín A, Sainz-Pardo JL (2017) Decomposition methods based on articulation vertices for degree-dependent spanning tree problems. *Computational Optimization and Applications* 68(3):749–773.

Landete M, Marín A, Sainz-Pardo JL (2019) Locating switches. *Expert Systems with Applications* 136:338–352.

Laporte G, Nickel S, Saldanha da Gama F, eds. (2019) *Location Science* (Cham: Springer International Publishing).

Larsson T, Quttineh NH, Åkerholm I (2024) A Lagrangian bounding and heuristic principle for bi-objective discrete optimization. *Operational Research* 24(2):14.

Leblanc LJ (1975) An Algorithm for the Discrete Network Design Problem. *Transportation Science* 9(3):183–199.

Leitner M, Ljubić I, Riedler M, Ruthmair M (2019) Exact Approaches for Network Design Problems with Relays. *INFORMS Journal on Computing* 31(1):171–192.

Magnanti TL, Wong RT (1984) Network Design and Transportation Planning: Models and Algorithms. *Transportation Science* 18(1):1–55.

Martello S, Toth P (1990) *Knapsack Problems: Algorithms and Computer Implementations*. Wiley-Interscience Series in Discrete Mathematics and Optimization (New York: Wiley & Sons).

Megiddo N, Zemel E, Hakimi SL (1983) The Maximum Coverage Location Problem. *SIAM Journal on Algebraic Discrete Methods* .

Melkote S, Daskin MS (2001a) Capacitated facility location/network design problems. *European Journal of Operational Research* 129(3):481–495.

Melkote S, Daskin MS (2001b) An integrated model of facility location and transportation network design. *Transportation Research Part A: Policy and Practice* 35(6):515–538.

Moreno J, Frota Y, Martins S (2018) An exact and heuristic approach for the d-minimum branch vertices problem. *Computational Optimization and Applications* 71(3):829–855.

Murawski L, Church RL (2009) Improving accessibility to rural health services: The maximal covering network improvement problem. *Socio-Economic Planning Sciences* 43(2):102–110.

Rozenberg J, Espinet Alegre X, Avner P, Fox C, Hallegatte S, Koks E, Rentschler J, Tariverdi M (2019) From A Rocky Road to Smooth Sailing: Building Transport Resilience to Natural Disasters. Technical report, World Bank, Washington, DC.

Smith-Miles K, Muñoz MA (2023) Instance Space Analysis for Algorithm Testing: Methodology and Software Tools. *ACM Comput. Surv.* 55(12):255:1–255:31.

Snyder LV (2011) Covering Problems. Eiselt HA, Marianov V, eds., *Foundations of Location Analysis*, 109–135 (New York, NY: Springer US).

Tamir A (1996) An O(pn2) algorithm for the p-median and related problems on tree graphs. *Operations Research Letters* 19(2):59–64.

Tian L, Bashan A, Shi DN, Liu YY (2017) Articulation points in complex networks. *Nature Communications* 8(1):14223.

van Veggel BWJR, den Hertog D, Aardal KI, Gromicho JAS (2026) A road network resilience optimization approach to improve healthcare accessibility, submitted to Healthcare Management Science.

Yıldız B, Boland N, Savelsbergh M (2022) Decomposition Branching for Mixed Integer Programming. *Operations Research* 70(3):1854–1872.

Zetina CA, Contreras I, Cordeau JF (2019) Exact algorithms based on Benders decomposition for multicommodity uncapacitated fixed-charge network design. *Computers & Operations Research* 111:311–324.

A Notation reference

Table A.1 provides a comprehensive overview of notation used in the paper.

Table A.1: Complete notation reference.

Symbol	Meaning
Graphs and sets	
$G = (N, E)$	Undirected road network.
N	Demand nodes and candidate sites.
F	Facility sites, $F \subseteq N$.
Parameters (nonneg. integers or $+\infty$)	
$\bar{b}$	Investment budget.
$\bar{d}$	Coverage radius / length bound.
h_k	Demand weight at node $k \in N$.
ℓ_e	Length of edge $e \in E$ (0 for facility edges).
c_e^E	Upgrade cost for $e \in E$ (0 for existing edges).
c_j^F	Facility opening cost for $j \in F$.
Decision variables	
z_k	Demand node k is covered.
x_e	Build/upgrade edge $e \in \overline{E}$.
y_j	Open facility at $j \in F$ ($\equiv x_{\{s,j\}}$).
f_a^k	Commodity-k flow on arc $a \in \overline{A}$.

Symbol	Meaning
Extended graph (super-source construction)	
$\overline{G}$	Extended graph with source s, facility edges.
$\overline{N}$	Extended node set, $\overline{N} = N \cup \{s\}$.
E^F	Facility edges, $E^F = \{\{s, j\} \mid j \in F\}$.
$\overline{E}$	Extended undirected edges, $\overline{E} = E \cup E^F$.
$\overline{A}$	Bidirected arcs of $\overline{E}$.
Block-cut tree notation	
$\mathcal{B}$	Set of blocks (biconnected components).
$\mathcal{A}$	Set of articulation points (APs).
N_B, E_B	Node set and edge set of block B.
a_C	Parent AP of child block C.
$\mathcal{C}_B$	Set of child blocks of block B.
T_B	Subtree rooted at block B.
$\mathcal{R}_B$	Set of relevant radii for block B.
DP states and selectors	
$V(B, b, r)$	DP value at budget b, signed parent radius r.
$\lambda^C_{b',r'}$	Select child state $(b', r') \in \Lambda_C$.

The signed radius $r \in \{-\bar{d}, \ldots, \bar{d}\}$ encodes the parent interaction: $r = 0$ no interaction (decoupled), $r > 0$ external coverage entering at remaining radius r, $r < 0$ an obligation to cover the parent side within distance $\bar{d} - |r|$.

B Block-level flow formulation with border conditions

This appendix provides the complete block-level flow formulation for computing the DP values $V(B, b, r)$ under the signed-radius border conditions of Section 4.1. We consider block B with node set N_B, edge set E_B, facility set F_B, parent AP a_B, and child blocks $\mathcal{C}_B$ with child APs a_C for $C \in \mathcal{C}_B$. Following Section 3.1, the source-node construction is applied to each block individually: the extended graph $\overline{G}_B$ has node set $\overline{N}_B := N_B \cup \{s\}$ with source node s, edge set $\overline{E}_B := E_B \cup \{\{s, j\} : j \in F_B\}$ with $\ell_{\{s,j\}} := 0$ and $c^E_{\{s,j\}} := c^F_j$, and arc

set $\overline{A}_B$ containing both directions $(u,v),(v,u)$ of each edge, each inheriting the length of its edge. Facility openings are the edge decisions $x_{\{s,j\}}$ rather than explicit y_j variables. For the child states, recall from Sections 4.1 and 4.3.3 the state sets $\Lambda_C \subseteq \{0,\dots,\bar{b}\} \times \mathcal{R}_C$ over the relevant child radii $\mathcal{R}_C$ and the selectors $\lambda^C_{b',r'}$. We write $\mathcal{R}^+_C := \{r' \in \mathcal{R}_C : r' > 0\}$ and $\mathcal{R}^-_C := \{r' \in \mathcal{R}_C : r' < 0\}$ for the positive and negative relevant child radii. As shorthand for the selected child states, the *activations* $\alpha^C_{\text{out}}[r']$ for $r' \in \mathcal{R}^+_C$ and $\alpha^C_{\text{int}}[r']$ for $r' \in \mathcal{R}^-_C$ indicate that the child state with radius r' is selected, and their aggregates α^C_{out} and α^C_{int} that some OUT resp. INT state of C is selected; they are plain sums of the selectors, defined by Constraints (34)–(36) of the master formulation.

The construction rests on the following observation: the border conditions largely preserve the flow model. They modify the *instance* on which it is solved, while the model itself remains unchanged and is only extended to enforce the border conditions and to couple the objective and budget to the precomputed child states. Recall from Section 3.1 that a node is covered if and only if there is an s-path of length at most $\bar{d}$ via edges in the solution. The border conditions extend exactly the two sides of this characterization, following the auxiliary-node interpretation of Section 4.3.2 and Figure 3: external coverage entering the block adds new origins for coverage paths (each virtual facility is added to the instance as a further facility), and coverage that B must provide to a neighboring block adds new destinations (each virtual demand node is added as a further demand node). On this auxiliary instance, the flow constraints of Section 3.1 apply unchanged; this coupling consists of the child-state selectors, which also add the DP value and child budget terms to the objective and budget, together with the linking constraints that tie the virtual elements to the selected states.

B.1 The auxiliary instance

Each of the four non-BLANK border conditions of Section 4.3.2 adds one vertex with its connecting arcs to the instance (the BLANK conditions $r = 0$, $r_C = 0$ add nothing; cf. Table 2 of the main paper):

- *Parent OUT ($r > 0$).* The virtual facility at distance $\bar{d} - r$ from a_B is added to the auxiliary instance as a vertex w_B, connected like any other facility: by the facility arc (s, w_B) of length 0, and by the connecting arc (w_B, a_B) of length $\bar{d} - r$ that leads into the block. Flow through w_B means that a coverage path originates at the parent side and enters B at a_B with remaining radius r. Since each parent state is solved as its own MILP, w_B is part of the parent-OUT MILPs ($r > 0$) only and open whenever present; using it remains optional. Unlike a regular facility, its opening is not a design decision and has no cost.

- *Child INT states ($r' \in \mathcal{R}^-_C$).* The virtual facility of a child INT state is added likewise, as a vertex $w_C[r']$ with the arcs $(s, w_C[r'])$ of length 0 and $(w_C[r'], a_C)$ of length $\bar{d} - |r'|$, one per candidate radius. Because the child

states are chosen inside the MILP, $w_C[r']$ is open only if the INT state with radius r' is selected, i.e., if $\alpha^C_{\text{int}}[r'] = 1$; even then it remains optional.

- *Child OUT states ($r' \in \mathcal{R}^+_C$).* The virtual demand node of a child OUT state is added as a vertex d_C, one per child, with the single connecting arc (a_C, d_C). Its length depends on the selected state, $\ell_{(a_C,d_C)} := \sum_{r' \in \mathcal{R}^+_C} r' \, \alpha^C_{\text{out}}[r']$, i.e., it is the selected radius r_C of the child selection λ^C if an OUT state of C is selected and 0 otherwise.[1] The node d_C must be covered exactly if an OUT state of C is selected; by the covering characterization above, covering it within $\bar{d}$ then means reaching a_C within $\bar{d} - r_C$.

- *Parent INT ($r < 0$).* The virtual demand node of the parent INT case is added likewise, as a vertex d_B with the connecting arc (a_B, d_B) of length $|r|$. It is part of the parent-INT MILPs ($r < 0$) only and must be covered there: an s-path of length at most $\bar{d}$ to d_B reaches a_B with remaining radius $|r|$, i.e., an opened facility covers a_B with radius $|r|$ to spare.

Virtual demand nodes carry no objective weight; the value of the coverage provided to a neighboring block enters through the DP terms $V(C, b', r')$ instead.

The auxiliary instance. Altogether, the auxiliary instance of block B and parent state (b, r) is the graph $\widetilde{G}_B$, obtained from $\overline{G}_B$ as follows:

- *Nodes.* The block nodes N_B and the source node s, extended by the virtual facilities $\mathcal{W}_B := \{w_B\} \cup \{w_C[r'] : C \in \mathcal{C}_B,\ r' \in \mathcal{R}^-_C\}$ and the virtual demand nodes $\mathcal{D}_B := \{d_C : C \in \mathcal{C}_B\} \cup \{d_B\}$.

- *Arcs.* The arc set $\widetilde{A}_B$ contains both directions of every edge in $\overline{E}_B$ (block edges and facility edges), extended by the virtual arcs listed above. Virtual arcs are directed and have no reverse arcs: flow can pass a virtual facility only from s into the block, and can enter a virtual demand node but not leave it. From now on, $\delta^{\pm}(v)$ are taken with respect to $\widetilde{A}_B$.

- *Demands.* The demand set is $\widetilde{N}_B := \big(N_B \setminus \{a_B\}\big) \cup \mathcal{D}_B$: every block node except the parent AP, whose demand is counted in the parent block, plus the virtual demand nodes. Exactly as in Section 3.1, demand k is covered if one unit of its commodity f^k runs from s to k within length $\bar{d}$: the coverage radius is the same for all demands, and the border distances are carried by the virtual arcs. Every demand carries a coverage indicator z_k, the right-hand side of its conservation constraint; the only distinction is that covering a real demand is a free decision, whereas the virtual demand nodes are forced to be covered exactly as required, by the explicit constraints $z_{d_C} = \alpha^C_{\text{out}}$ and $z_{d_B} = 1$ of Section B.2.

[1] Conceptually, each candidate radius $r' \in \mathcal{R}^+_C$ has its own virtual demand node at distance r' from a_C. Since exactly one state is selected, they merge into the single node d_C whose arc carries the selected radius as its length: as in Section 4.3.4, one radius-independent commodity per child suffices.

At most one of w_B and d_B is present, depending on the sign of r; Section B.2 states the convention under which a single formulation covers all three cases.

B.2 The block-level MILP

Rather than stating three near-identical MILPs, we present one *master formulation*, valid for every parent radius r, under the following convention:

> The parent virtual facility w_B exists only if $r > 0$; the parent virtual demand node d_B exists only if $r < 0$; for $r = 0$ neither exists. Elements that do not exist are omitted from all constraints together with their arcs and flow variables (equivalently, fixed to 0).

Instantiating the convention yields the three cases of Section 4.1: for OUT ($r > 0$) the virtual demand node d_B is absent together with its arc, its commodity, and Constraint (38); for INT ($r < 0$) the virtual facility w_B is absent with its arcs and flow variables; for BLANK ($r = 0$) both are absent and only the child-side virtual elements remain. Section D describes how the transitions between these cases are realized on a single MILP instance.

Objective, budget, and child-state selection. These rows realize the recursion (16) and extend the objective and budget of the base formulation by the child terms: the objective (31) maximizes the demand covered inside B plus the DP values of the selected child states (the parent AP a_B is excluded; its demand is counted in the parent block); (32) charges the local design cost plus the budgets of the selected child states against b; (33) selects exactly one state per child.

$$\max \quad \sum_{k \in N_B \setminus \{a_B\}} h_k z_k + \sum_{C \in \mathcal{C}_B} \sum_{(b',r') \in \Lambda_C} V(C, b', r') \lambda^C_{b',r'} \tag{31}$$

$$\text{s.t.} \quad \sum_{e \in \overline{E}_B} c^E_e x_e + \sum_{C \in \mathcal{C}_B} \sum_{(b',r') \in \Lambda_C} b' \lambda^C_{b',r'} \le b \tag{32}$$

$$\sum_{(b',r') \in \Lambda_C} \lambda^C_{b',r'} = 1 \qquad \forall C \in \mathcal{C}_B \tag{33}$$

Child-state activations. Constraints (34)–(36) define the activations $\alpha^C_{\text{out}}[r']$, $\alpha^C_{\text{int}}[r']$ and their aggregates used throughout Section B.1. They are sums of selectors and introduce no new decisions; they feed the opening of the child virtual facilities, the coverage indicators of the virtual demand nodes, and the length of the arcs (a_C, d_C). Together with (33) they satisfy $\alpha^C_0 + \alpha^C_{\text{out}} + \alpha^C_{\text{int}} = 1$, where $\alpha^C_0 := \sum_{b':(b',0) \in \Lambda_C} \lambda^C_{b',0}$ selects the BLANK child state.

$$\alpha^C_{\text{out}}[r'] = \sum_{b':\,(b',r') \in \Lambda_C} \lambda^C_{b',r'} \qquad \forall C \in \mathcal{C}_B,\ r' \in \mathcal{R}^+_C \tag{34}$$

$$\alpha^C_{\text{int}}[r'] = \sum_{b':\,(b',r') \in \Lambda_C} \lambda^C_{b',r'} \qquad \forall C \in \mathcal{C}_B,\ r' \in \mathcal{R}^-_C \tag{35}$$

$$\alpha^C_{\text{out}} = \sum_{r' \in \mathcal{R}^+_C} \alpha^C_{\text{out}}[r'], \quad \alpha^C_{\text{int}} = \sum_{r' \in \mathcal{R}^-_C} \alpha^C_{\text{int}}[r'] \qquad \forall C \in \mathcal{C}_B \tag{36}$$

Coverage of the virtual demand nodes. Covering a virtual demand node is not a decision but a consequence of the selected states, imposed by two explicit constraints: (37) forces d_C to be covered exactly if an OUT state of C is selected, and (38) makes the parent obligation unconditional. With these constraints, the coverage indicators z_k, $k \in \widetilde{N}_B$, enter the flow formulation below in the same way for real and for virtual demand nodes.

$$z_{d_C} = \alpha^C_{\text{out}} \qquad \forall C \in \mathcal{C}_B \tag{37}$$

$$z_{d_B} = 1 \tag{38}$$

The flow formulation on the auxiliary instance. Constraints (39)–(42) are the flow formulation of Section 3.1 on the auxiliary instance $\widetilde{G}_B$, with one commodity f^k per demand $k \in \widetilde{N}_B$. Flow conservation (39) carries over unchanged: commodity k has supply z_k at the source s and demand z_k at its node k, and the third case is standard flow conservation at every other node. The edge-availability constraints (40) likewise carry over: an edge can carry flow, in either direction, only if it is upgraded, and a facility edge only if its facility is opened. Virtual facilities have no design variables, so (41) plays this role for them, opening a child facility $w_C[r']$ exactly if its INT state is selected. The parent facility w_B is open whenever it exists and requires no such constraint, and neither do the connecting arcs, whose flows coincide with those on the facility arcs. The length bound (42) is the length bound of the base formulation with all virtual contributions written out: a path through a virtual facility consumes the offset length of its connecting arc, accounted on its facility-arc flow, and the connecting arc of a virtual demand node contributes the case term, its length times the coverage indicator, which simplifies to $\sum_{r' \in \mathcal{R}^+_C} r'\, \alpha^C_{\text{out}}[r']$ resp. $|r|\, z_{d_B}$ by (37) resp. (38); every term of (42) is therefore linear.

$$\sum_{a \in \delta^+(v)} f^k_a - \sum_{a \in \delta^-(v)} f^k_a = \begin{cases} z_k & \text{if } v = s \\ -z_k & \text{if } v = k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in \widetilde{N}_B,\ v \in \overline{N}_B \cup \mathcal{W}_B \cup \mathcal{D}_B \tag{39}$$

$$f^k_{(u,v)} + f^k_{(v,u)} \le x_e \qquad \forall k \in \widetilde{N}_B,\ e \in \overline{E}_B \tag{40}$$

$$f^k_{(s,w_C[r'])} \le \alpha^C_{\text{int}}[r'] \qquad \forall k \in \widetilde{N}_B,\ C \in \mathcal{C}_B,\ r' \in \mathcal{R}^-_C \tag{41}$$

$$\sum_{a \in \overline{A}_B} \ell_a f^k_a + (\bar{d} - r)\, f^k_{(s,w_B)} + \sum_{\substack{C \in \mathcal{C}_B \\ r' \in \mathcal{R}^-_C}} (\bar{d} - |r'|)\, f^k_{(s,w_C[r'])} + \begin{cases} \sum_{r' \in \mathcal{R}^+_C} r'\, \alpha^C_{\text{out}}[r'] & \text{if } k = d_C \\ |r|\, z_{d_B} & \text{if } k = d_B \\ 0 & \text{otherwise} \end{cases} \le \bar{d}\, z_k \qquad \forall k \in \widetilde{N}_B \tag{42}$$

Linking the child OUT commodities. One additional family of constraints completes the model: if no OUT state of C is selected, then $z_{d_C} = 0$, and the variable upper bounds (43) force the entire commodity of d_C to zero.

$$f^{d_C}_a \le \alpha^C_{\text{out}} \qquad \forall C \in \mathcal{C}_B,\ a \in \widetilde{A}_B \tag{43}$$

Reading these constraints per demand type shows how the auxiliary instance

realizes the border conditions. For a real demand nothing changes relative to Section 3.1, except that a coverage path may now originate at a virtual facility, entering the block at the respective AP with exactly the remaining radius that (42) accounts for; in particular, a demand node that is itself a child AP may be covered through its own child's virtual facility. For the child OUT obligation ($k = d_C$), the case term of (42) equals the selected radius r_C, so the facility supplying this coverage must reach a_C with radius r_C to spare; it is opened in B, or it is virtual, namely the parent side (w_B, if $r > 0$) or a sibling placed in an INT state.[2] For the parent INT obligation ($k = d_B$), the case term $|r|$ leaves at most $\bar{d} - |r|$ for the path to a_B, which is exactly the covering condition the parent side imposes.

Variable domains. All variables are binary; the virtual elements exist as stated by the convention above. The integrality of all flow variables can be relaxed to $[0, 1]$ without loss.

$$x_e \in \{0,1\}\ \forall e \in \overline{E}_B, \qquad z_k \in \{0,1\}\ \forall k \in \widetilde{N}_B,$$
$$\lambda^C_{b',r'} \in \{0,1\}\ \forall C \in \mathcal{C}_B,\ (b',r') \in \Lambda_C, \quad \alpha^C_{\text{out}}[r'],\ \alpha^C_{\text{int}}[r'],\ \alpha^C_{\text{out}},\ \alpha^C_{\text{int}} \in \{0,1\},$$
$$f^k_a \in \{0,1\}\ \forall k \in \widetilde{N}_B,\ a \in \widetilde{A}_B$$

The auxiliary instance is also how the formulation is built in practice; Section D describes how a single MILP instance per block carries every DP state.

C Bound propagation details

This appendix collects the formal assumption, statements, proofs, and bound derivations summarized in Section 4.3.5.

Decomposition assumption. The empirical observation from Section 4.3.3 (that the λ-allocation is recovered exactly from the incumbent's local design $(\hat{x}, \hat{z})$ via MCKP) motivates the following assumption used in the gap analysis below.

Assumption 1. *For the theoretical gap analysis we assume that the child allocation is solved exactly given the local decisions $(\hat{x}, \hat{z})$. That is, the primal bound $l(B, b, r)$ returned by a sub-optimal MILP solve is optimal with respect to λ given $(\hat{x}, \hat{z})$:*

$$l(B,b,r) \;=\; \max_{\lambda}\Big(h_B(\hat{x},\lambda) \;+\; \sum_{C\in\mathcal{C}_B} l(C,\lambda^C)\Big),$$

where $h_B(\hat{x}, \lambda)$ denotes the local coverage contribution of the primal solution.

[2] With a virtual facility as source, (42) reduces to path length $\le r - r_C$ resp. $\le |r_{C'}| - r_C$ for a sibling C' in an INT state: external coverage may continue through B and satisfy the OUT condition of C whenever the connecting path is short enough. The sibling facilities give one flow per ordered pair of children of B.

Supporting experiment. A preliminary experiment motivates the assumption: we re-solved the selector subproblem (29) alone from the incumbent $(\hat{x}, \hat{z})$ of 41 sampled block solves, 13 of which were sub-optimal and therefore in the scope of the assumption. In all 13, the MCKP recovered the incumbent's λ exactly and in at most 0.13 s, supporting the qualitative property the assumption abstracts, namely that the selector subproblem is easy relative to the network-design part. The sampling protocol and the per-solve outcomes (`results/lambda_experiment_combined.csv`) are in the online code repository.

Gap propagation setup. Consider a block B with children $\mathcal{C}_B$, where valid bounds $l(C, \lambda^C) \le V(C, \lambda^C) \le u(C, \lambda^C)$ with gap $\Delta(C, \lambda^C) := u(C, \lambda^C) - l(C, \lambda^C)$ are available for each child $C \in \mathcal{C}_B$ and each state $\lambda^C \in \Lambda_C$. When solving the MILP for $V(B, b, r)$, for lack of the optimal values $V(C, \lambda^C)$, we use child *lower* bounds $l(C, \lambda^C)$ for the child contributions in (16). For that problem, the solver returns a primal bound $l(B, b, r)$ and a dual bound $\tilde{u}(B, b, r)$:

$$l(B, b, r) \;\le\; \max_{x,\,\lambda}\Big(h_B(x, \lambda) \;+\; \sum_{C \in \mathcal{C}_B} l(C, \lambda^C)\Big) \;\le\; \tilde{u}(B, b, r).$$

Note that, since that computation did not use the child upper bounds $u(C, \lambda^C)$, $\tilde{u}(B, b, r)$ is not a valid upper bound on $V(B, b, r)$, however, we can make it one by adding the maximum gap of each child block.

Proposition 1 (Additive gap propagation)**.** *Define* $\Delta_{\max}(C) := \max_{(b', r') \in \Lambda_C} \Delta(C, b', r')$. *Then* $u(B, b, r) := \tilde{u}(B, b, r) + \sum_{C \in \mathcal{C}_B} \Delta_{\max}(C)$ *is a valid upper bound on* $V(B, b, r)$.

Proof. Replacing each child value $V(C, \lambda^C)$ by its upper bound $u(C, \lambda^C) = l(C, \lambda^C) + \Delta(C, \lambda^C)$, and bounding each $\Delta(C, \lambda^C) \le \Delta_{\max}(C)$, gives

$$\begin{aligned} V(B, b, r) \;&\le\; \max_{x,\,\lambda}\Big(h_B(x, \lambda) \;+\; \sum_{C \in \mathcal{C}_B} u(C, \lambda^C)\Big) \\ &\le \max_{x,\,\lambda}\Big(h_B(x, \lambda) \;+\; \sum_{C \in \mathcal{C}_B} l(C, \lambda^C)\Big) \;+\; \sum_{C \in \mathcal{C}_B} \Delta_{\max}(C) \\ &\le\; \tilde{u}(B, b, r) \;+\; \sum_{C \in \mathcal{C}_B} \Delta_{\max}(C). \end{aligned}$$

□

To avoid the conservatism of replacing state-dependent gaps by $\Delta_{\max}$, one can solve an explicit upper bounding problem $\bar{V}(B, b, r) := \max_{x,\lambda}\big(h_B(x, \lambda) + \sum_C u(C, \lambda^C)\big)$, yielding the tighter chain $l(B, b, r) \le V(B, b, r) \le \bar{V}(B, b, r) \le u(B, b, r)$.

Relative gap preservation. While additive gaps accumulate, relative gaps are preserved under Assumption 1. We assume $l(B, \lambda^B) > 0$ and $h_B(\hat{x}, \lambda) > 0$ for all states under consideration; states with zero local coverage can be pruned from the DP table.

Proposition 2 (Relative gap preservation)**.** *Under Assumption 1, suppose that for block B in state λ^B, the primal network design $\hat{x}$ satisfies*

$$h_B(\hat{x}, \lambda) \;\geq\; \alpha_B \cdot \max_x h_B(x, \lambda) \qquad \textit{for all child allocations } \lambda,$$

with $\alpha_B \in (0, 1]$. Suppose inductively that for each child $C \in \mathcal{C}_B$,

$$l(C, \lambda^C) \;\geq\; \alpha \cdot V(C, \lambda^C) \qquad \textit{for all } \lambda^C \in \Lambda_C,$$

for some common ratio $\alpha \in (0, 1]$. Then

$$l(B, \lambda^B) \;\geq\; \min(\alpha_B,\, \alpha) \cdot V(B, \lambda^B).$$

Proof. Let (x^*, λ^*) achieve $V(B, \lambda^B) = h_B(x^*, \lambda^*) + \sum_C V(C, \lambda^{C*})$, and write $H := h_B(x^*, \lambda^*)$, $S := \sum_C V(C, \lambda^{C*})$. By Assumption 1, $l(B, \lambda^B) = \max_\lambda \big(h_B(\hat{x}, \lambda) + \sum_C l(C, \lambda^C)\big)$. Evaluating at the feasible point λ^*:

$$\begin{aligned} l(B, \lambda^B) \;&\geq\; h_B(\hat{x}, \lambda^*) + \sum_C l(C, \lambda^{C*}) \;\geq\; \alpha_B \cdot H \;+\; \alpha \cdot S \\ &\geq\; \min(\alpha_B, \alpha) \cdot (H + S) \;=\; \min(\alpha_B, \alpha) \cdot V(B, \lambda^B). \end{aligned}$$

□

Corollary 1 (Global approximation ratio)**.** *Applying Proposition 2 inductively from the leaves to the root, the global ratio satisfies $l(\text{root}) \geq \alpha_{\text{global}} \cdot V(\text{root})$ with $\alpha_{\text{global}} = \min_{B \in \mathcal{T}} \alpha_B$, determined solely by the weakest local solve in the tree.*

Bounds from neighboring states. Under limited time it may be a choice to solve only a subset of all states; we illustrate how upper bounds are derived when the BLANK state $V(B, b, 0)$ is computed but the OUT and INT states are not. Let a_B be the parent articulation point of block B, let $N^B(a_B, \rho) := \{v \in T_B \mid d(a_B, v) \leq \rho\}$ with total weight $w^B(a_B, \rho) := \sum_{v \in N^B(a_B, \rho)} h_v$. For $r > 0$, external coverage adds at most the weight of newly reachable nodes, giving the OUT bound $V(B, b, r) \leq V(B, b, 0) + w^B(a_B, r)$ stated in Section 4.3.5. For $r < 0$, the block must open a facility within distance $\bar{d} - |r|$ of a_B, which covers nodes up to $\bar{d}$ beyond it, hence within $2\bar{d} - |r|$ of a_B. Letting $c_{\min}(\rho) := \min_{j \in N^B(a_B, \rho) \cap F_B} c_j^F$ and assuming feasibility (i.e., $N^B(a_B, \bar{d} - |r|) \cap F_B \neq \emptyset$ and $b \geq c_{\min}(\bar{d} - |r|)$):

$$V(B, b, r) \;\leq\; V(B, b - c_{\min}(\bar{d} - |r|), 0) + w^B(a_B, 2\bar{d} - |r|) \qquad \text{for } r < 0. \quad (44)$$

Removing this cheapest eligible facility from an optimal INT solution, but keeping its edges, frees exactly $c_{\min}(\bar{d} - |r|)$ of budget and loses only coverage within

$2\bar{d} - |r|$ of a_B; the remaining budget is bounded by the BLANK value. Combined with monotonicity (Section 4.1), these bounds allow us to tighten gaps across related states in negligible time in case lower bounds are computed by other heuristic methods.

D Implementation choices

The BC-tree DP framework involves several implementation choices that affect computational performance. We summarize the key decisions made after preliminary experiments.

Warm-starting and state ordering. We employ two complementary warm-starting strategies. First, each block MILP is initialized with a greedy Prim-like construction that grows a subgraph from the source node s, selecting edges by a profit function that trades coverage gain against upgrade cost and respects the budget; its profit function is documented with the code. Second, we exploit the monotonicity of DP values across neighboring states: given a solution for state $V(B, b, r)$, the solutions for $V(B, b+1, r)$ and $V(B, b, r+1)$ can only improve, so previous solutions provide valid lower bounds and warm starts. Of the possible iteration orders, preliminary experiments favored budgets increasing from 0 to $\bar{b}$ in the outer loop and, for each budget, the signed radius increasing from $-\bar{d}$ to $\bar{d}$ in the inner loop, in line with the monotonicity in r.

Incremental model construction. For each block B, the flow formulation of Section 3.1 restricted to B is built once as a persistent base model, and the border conditions are added as incremental modifications, a feature that modern MILP solvers support directly: the selectors and the virtual elements contribute a moderate number of new columns and rows, while all rows and columns of the base model remain unchanged. A single model instance per block therefore carries every DP state (b, r). Consecutive states differ only in the budget b and in a few bound and right-hand-side switches along the convention of Section B.2: for $r \leq 0$, the flows through the parent virtual facility are fixed to 0; for $r \geq 0$, the parent obligation is switched off by setting $z_{d_B} = 0$. The solver thus keeps the model and its warm-start information across all states of a block.

Root block selection and block merging. The BC-tree can be rooted at any block; we default to the largest block by vertex count, which can be expected to be the computationally most challenging one. Since the root block is solved only for the decoupled condition ($r = 0$) at budget $\bar{b}$, placing the root there is the least costly option. Graphs with many articulation points may decompose into numerous very small biconnected components such as single edges or triangles, for which the overhead of the DP framework may outweigh its benefit; we therefore merge adjacent blocks with at most three vertices, which reduces the number of DP subproblems while increasing the block sizes only marginally.

Relevant budget and distance values. A naive implementation iterates over all budgets $b \in \{0, \ldots, \bar{b}\}$ and distances $d \in \{0, \ldots, \bar{d}\}$, but only a subset of these values is achievable: the attainable budgets are the realizable sums of edge costs (a subset sum problem), and the feasible distances are the path lengths that exist in the graph. We use simple heuristics to identify a superset of relevant values, avoiding enumeration of clearly infeasible states. This filtering is purely combinatorial; among the remaining states, further states may be dominated and thus not Pareto-optimal, which is detected during the DP iteration, e.g., by obtaining the same solution despite having increased budget or radius.

Time allocation across DP states. When a total time limit T_{total} is imposed, blocks first receive time proportionally to their edge count, $T_B = T_{\text{total}} \cdot |E_B| / \sum_{B'} |E_{B'}|$, of which a small buffer is withheld. Within each block we then identify a small set of *promising* budgets from the LP relaxation of the BLANK state, using the ratio $\text{UB}_{\text{LP}}(b)/b$ as a measure of marginal coverage per unit budget, and give them a higher weight than the remaining budgets; each budget's weight is shared equally among its associated DP states (one per signed radius r). Time is finally distributed across the q states with a front-loaded weighted schedule: half of the block's time is reserved as a common deadline buffer and the remaining half is split proportionally to the weights w_i, so that the i-th state receives

$$T_i \;=\; \tfrac{T}{2} + \frac{\sum_{j \le i} w_j}{\sum_j w_j} \cdot \tfrac{T}{2} \;-\; t_{\text{elapsed}},$$

and unused time from faster solves automatically accumulates for later states. States that would receive too little time are skipped, and bounds are constructed from dominating states and heuristics (Section 4.3.5). Compared with uniform weighting, the LP-guided scheme concentrates effort on budgets that look most promising in the relaxation, which we found to improve solution quality on hard instances without sacrificing optimality on easy ones. The exact interval partition, weights, buffer fractions, and skipping threshold are documented with the code in the online repository.

E Pseudocode

This appendix collects pseudocode for the tree DP (Section 4.2) and the BC-tree DP (Section 4.3), in the unified signed-radius notation of Section 4.1 and including the practical refinements of Section D. The complete pseudocode, with the per-node recursion in full, the helper functions $\mathcal{A}, \mathcal{B}, \mathcal{C}^1, \mathcal{C}^2$, and the dominance-pruning and time-allocation procedures, is provided as `docs/PSEUDOCODE.pdf` in the online code repository, alongside a code-to-paper notation mapping that points from every algorithm step to the concrete file and function.

E.1 Tree DP for the MCNDP

Algorithm 1 implements the top-level tree DP from Section 4.2. It transforms the input tree into a binary tree, traverses nodes bottom-up, and at every node fills the table $V(v, b, r)$ for all $(b, r) \in \{0, \dots, \bar{b}\} \times \mathcal{R}_v$ from the recursions of Section 4.2, which combine the helpers $\mathcal{A}, \mathcal{B}, \mathcal{C}^1, \mathcal{C}^2$ defined there. The three radius regimes are distinguished as follows. At a leaf, $V(v, b, r) = h_v$ if $r \geq \ell_e$ (the parent covers v for free), and otherwise h_v if v can afford a facility, with the shortfall being 0 for $r \geq 0$ and $-\infty$ for $r < 0$, where the INT obligation would be violated. At an inner node, $V(v, b, r)$ is the maximum over opening a facility at v and passing $\bar{d}$ downward, forwarding the inherited radius $r - \ell_e$ (only if $r \geq \ell_e$), leaving v uncovered via $\mathcal{A}$ (only if $0 \leq r < \ell_e$), and letting either subtree cover v and pass a radius outward via $\mathcal{C}^1, \mathcal{C}^2$ (at 0 for $r \geq 0$ and at $|r|$ for $r < 0$). Negative budgets and radii outside $\mathcal{R}_v$ are treated as $-\infty$.

Algorithm 1: TREEDP$(G, \bar{b}, \bar{d})$: tree DP for the MCNDP.

Input : Tree $G = (N, E)$ with costs c^F, c^E, lengths ℓ, weights h; budget $\bar{b}$; radius $\bar{d}$.
Output: Optimal objective value, set $Y \subseteq F$ of facilities, set $X \subseteq E$ of upgraded edges.

Root G at an arbitrary v_r (we use the highest-degree node).;
Transform G into a binary tree by inserting dummy nodes ($h = 0$, $c^F = +\infty$, $c^E = 0$) so that every node has at most two children.;
Compute all-pairs distances $d_G(\cdot, \cdot)$.;
foreach *node* $v \in V$ **do**
    $\mathcal{R}_v \leftarrow \{ \pm(\bar{d} - d_G(v, u)) \mid u \notin T_v,\ d_G(v, u) < \bar{d} \} \cup \{0\}$;
foreach v *in post-order traversal* **do**
    foreach $(b, r) \in \{0, \dots, \bar{b}\} \times \mathcal{R}_v$ **do**
        Set $V(v, b, r)$ by the leaf resp. inner-node recursion of Section 4.2.;
opt $\leftarrow V(v_r, \bar{b}, 0)$;
$(Y, X) \leftarrow$ BACKTRACK$(v_r, \bar{b}, 0)$;
return (opt, Y, X)

The DP tables are stored as nested dictionaries indexed by b and the signed radius, and the helpers iterate over all budget splits $b_1 + b_2 \leq b$ and the binary edge-upgrade decisions, which is what the complexity bound $\mathcal{O}(n\bar{b}^2R^2)$ of Section 4.2 reflects.

E.2 BC-tree DP for the MCNDP

Algorithm 2 extends the framework of Section 4.3 to graphs with articulation points. For each block B and each state (b, r) it constructs the parameterized block-level MILP from Section 4.3.3 and Section B, applies a warm start, and solves within an allotted time $T_{b,r}$. The dominance-based pruning and the LP-guided time allocation it invokes are described in Section D.

Algorithm 2: BCTreeDP$(G, \bar{b}, \bar{d}, T_{\text{total}})$: BC-tree DP for the MC-NDP.

Input : Graph $G = (N, E)$ with costs and lengths as before; budget $\bar{b}$; radius $\bar{d}$; optional total time limit T_{total}.
Output: Best feasible value $l(\text{root})$ with global upper bound $u(\text{root})$; facilities Y; upgraded edges X.

Decompose G into blocks $\mathcal{B}$ and articulation points $\mathcal{A}$.;
(Optional) merge adjacent blocks of size ≤ 3 to reduce DP overhead.;
Build the block-cut tree $\mathcal{T}$ and root it at the largest block B_r.;
foreach *block* $B \in \mathcal{B}$ **do**
    Precompute $\mathcal{R}_B$, the set of relevant budgets, and AP distances.;
Allocate per-block time T_B proportional to $|E_B|$ (with cap on the root block).;
foreach B *in post-order traversal of the block nodes of* $\mathcal{T}$ **do**
    Compute $B.\Delta_{\max} := \sum_{C \in \mathcal{C}_B} \Delta_{\max}(C)$.; `// aggregate child gaps`
    foreach (b, r) *in the relevant states of* B*, budget outer, signed radius inner* **do**
        $T_{b,r} \leftarrow$ LP-guided share of T_B ; `// Section D`
        if $T_{b,r} < T_{\min}$ **then**
            $(l, u) \leftarrow$ SkipState(B, b, r) ; `// bounds from neighbor states, e.g.` (44)
        else
            Build the block MILP for (b, r) (Section B), reading the children's stored values $V(C, b', r')$ over the non-dominated states $(b', r') \in \Lambda_C$ through the child selectors.;
            Warm-start from the greedy heuristic or the incumbent of a neighboring state, and solve within $T_{b,r}$; record primal l and dual $\tilde{u}$.;
            $u \leftarrow \tilde{u} + B.\Delta_{\max}$; `// Prop. 1`
        Store l, u in the DP tables for $V(B, b, r)$.;
    Mark states (b', r') with $u(B, b', r') \leq l(B, b, r)$ for some $b \leq b'$, $r \leq r'$ as dominated, exclude them from Λ_B, and pass Λ_B with $\Delta_{\max}(B)$ to the parent.;
opt $\leftarrow l(B_r, \bar{b}, 0)$;
$(Y, X) \leftarrow$ BacktrackBCT$(B_r, \bar{b}, 0)$;
return (opt, $u(B_r, \bar{b}, 0)$, Y, X)

The block-level MILP is re-used across states of the same block (Section D), and the aggregate $B.\Delta_{\max}$ is constructed once per block, in line with the inductive bound argument in Section C. Skipped states fill the DP table with the conservative bounds of that section, so that the parent's selectors remain valid even when no full MILP solve has happened.

F Instance details

See Table F.1 for a summary of the BC-tree instances used in the experiments of Section 5.

G Practical solve scenarios: experimental details

This section documents the setup for the two practical solve scenarios discussed in the main text (Figure 8). Both experiments are run on the same five instances:

Table F.1: Summary of BC-tree instances. n: nodes; m: edges; Bl: biconnected components (before merging); APs: articulation points; AP°: maximum AP degree; Max/Min/Avg: block sizes.

ID	Instance	n	m	Bl	APs	AP°	Max	Min	Avg
1	mixed_2full_sevilla_sioux_4b	152	332	2	1	2	96	57	76.5
2	mixed_sevilla2_grid6x6_3_5b	202	418	5	4	2	49	36	41.2
3	mixed_sevilla3_grid5x5_2_grid4x4_2_7b	223	485	7	6	2	49	16	32.7
4	mixed_sevilla4_grid5x5_2_6b	241	556	6	5	2	49	25	41.0
5	mixed_sevilla5_grid5x5_1_6b	265	635	6	5	2	49	25	45.0
6	sevilla_5_5b	241	595	5	4	2	49	49	49.0
7	sevilla_6_6b	289	714	6	5	2	49	49	49.0
8	sioux_10copies_8b	216	374	6	5	2	57	24	36.8
9	sioux_11copies_2b	251	418	2	1	2	228	24	126.0
10	sioux_3subnets_5b	51	76	5	4	2	24	2	11.0
11	sioux_3subnets_6b	66	101	6	5	2	24	2	11.8
12	sioux_4subnets_18b	78	111	18	15	3	35	2	5.3
13	sioux_4subnets_20b	62	85	20	16	3	34	2	4.0
14	sioux_5copies_5b	116	190	5	4	2	24	24	24.0
15	sioux_5subnets_10b	74	108	10	9	2	60	2	8.3
16	sioux_5subnets_34b	87	111	34	30	3	16	2	3.5
17	sioux_7copies_2b	159	265	2	1	2	136	24	80.0
18	sioux_8b	47	76	4	3	2	43	2	12.5
19	sioux_8copies_4b	170	297	2	1	2	113	58	85.5
20	sioux_9b	56	87	9	7	3	25	2	7.1
21	sioux_9copies_17b	216	350	17	16	2	24	2	13.6
22	tree_balanced_4b_118n	118	233	4	1	4	49	24	30.2
23	tree_balanced_4b_93n	93	152	4	1	4	24	24	24.0
24	tree_balanced_7b_337n	337	833	7	6	2	49	49	49.0
25	tree_random_3b_70n	70	114	3	1	3	24	24	24.0
26	tree_random_7b_162n	162	266	7	6	2	24	24	24.0
27	tree_random_7b_165n	165	272	7	6	2	25	24	24.4
28	tree_star_5b_116n_v2	116	190	5	1	5	24	24	24.0
29	tree_star_7b_162n	162	266	7	5	3	24	24	24.0
30	tree_star_7b_162n_v2	162	266	7	1	7	24	24	24.0
31	tree_star_7b_237n	237	509	7	5	3	49	24	34.7
32	tree_star_7b_237n_v2	237	509	7	1	7	49	24	34.7
33	tree_star_7b_239n	239	513	7	1	7	49	24	35.0
34	tree_star_8b_262n	262	551	8	1	8	49	24	33.6

the largest networks in our benchmark that did not reach optimality within one hour for either solver, where the choice of approach has the highest practical impact. All runs use a single thread and the same Gurobi configuration as in the main computational study; the BC-tree DP uses the LP-guided state weighting described in Section D. Reported coverage values are in both cases the mean over the five instances.

Coverage–budget curve (Figure 8a). We split the interval $[0, \bar{b}]$ into 20 equally spaced budget values $b_i = \frac{i}{19}\bar{b}$ for $i = 0, \ldots, 19$, at coverage radius factor $\gamma = 0.10$. The BC-tree DP solves all 20 budget points within a single execution at a one-hour wall-clock limit. Each FlowMIP variant solves the 20 points sequentially under the same one-hour total budget; the difference between variants is solely the per-point time allocation, which is proportional to $\log(1 + b_i)$ for *FlowMIP-LogTime*, to b_i for *FlowMIP-LinTime*, and to b_i^2 for *FlowMIP-QuadTime*, with the weights normalized so that the per-point times sum to the total one-hour budget.

Time-limit sensitivity (Figure 8b). We solve each instance at its default budget for total time limits geometrically spaced between roughly one minute and one hour. For each time limit, both the BC-tree DP and FlowMIP are run independently under that single limit (no warm-starting across runs) and the best incumbent at the limit is recorded.

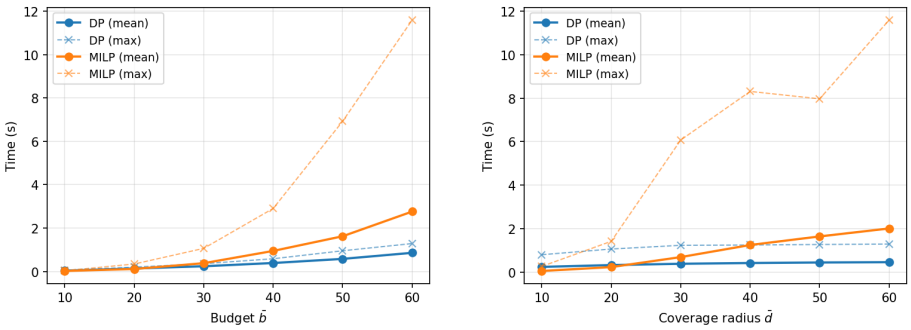
H Tree DP: additional results

Figure H.1 reports execution times on the 20-node tree experiment (324 test cases: $\bar{b}, \bar{d} \in \{10, 20, \ldots, 60\}$, nine random trees per combination) as a function of $\bar{b}$ and $\bar{d}$ separately. Both algorithms scale with increasing parameter values, but the MILP shows increasing variance and longer tails (up to 11.6 seconds), while the DP stays below 1.3 seconds. Table H.1 gives the underlying ratio of mean runtimes per parameter combination. The MILP is faster on average in 18 of the 36 cells, all of them at low $\bar{b}$ or low $\bar{d}$, but its advantage there is over solves that are already cheap: across those cells the MILP averages 0.11 seconds and the DP 0.21 seconds, and no single solve of either algorithm exceeds 1.5 seconds. The effect is most pronounced in the $\bar{d} = 10$ column, where the MILP wins all 54 test cases by a median factor of 6.5. In the remaining 18 cells the ordering reverses and the absolute times matter more: the MILP averages 1.84 seconds against the DP's 0.54 seconds.

Figure H.2 shows the scaling behavior at the two lower parameter settings, $\bar{b} = 20$, $\bar{d} = 40$ and $\bar{b} = 30$, $\bar{d} = 30$. Both complement Figure 5b in the main text, which shows the higher-parameter setting $\bar{b} = 40$, $\bar{d} = 50$ where the DP's margin is far larger. At $\bar{b} = 30$, $\bar{d} = 30$ the DP is ahead in mean runtime at every instance size, but only by a factor of 1.1 to 2.2, and it wins on 50 of the 64 individual trees. At $\bar{b} = 20$, $\bar{d} = 40$ the two algorithms are close: the mean runtime ratio ranges from 0.8 to 1.9, the MILP is ahead on average at

Table H.1: Ratio of mean MILP runtime to mean DP runtime on 20-node trees (nine trees per cell). Values above 1 indicate that the DP is faster.

	Coverage radius $\bar{d}$					
Budget $\bar{b}$	10	20	30	40	50	60
10	0.28	0.40	0.53	0.62	0.70	0.74
20	0.17	0.67	0.73	0.91	1.13	1.10
30	0.23	0.83	1.46	1.94	2.02	2.24
40	0.30	0.87	1.65	2.51	3.57	4.18
50	0.18	0.77	1.68	2.95	3.96	5.19
60	0.16	0.58	2.29	3.97	4.66	5.29

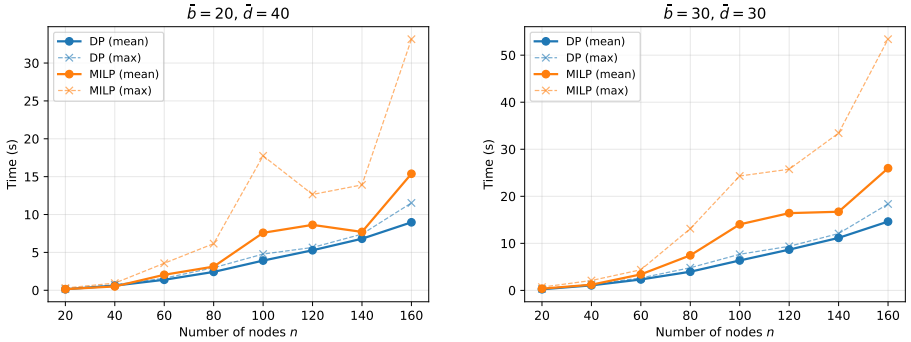


(a) Execution time vs. budget $\bar{b}$.

(b) Execution time vs. coverage radius $\bar{d}$.

Figure H.1: MILP and DP execution times on 20-node random trees across 36 parameter combinations.

$n = 40$, and the DP wins on 42 of the 64 individual trees. Across all three scaling settings the MILP's slowest tree of a given size takes 1.3 to 2.4 times its mean runtime, whereas the DP stays within 1.1 to 1.3 times its own mean, the same spread effect seen at $n = 20$.



(a) $\bar{b} = 20$, $\bar{d} = 40$.

(b) $\bar{b} = 30$, $\bar{d} = 30$.

Figure H.2: Scaling in n at the two lower parameter settings, eight trees per size. Solid lines are means over the trees of a given size, dashed lines the corresponding maxima.

I BC-tree DP: time performance profile

Figure I.1 reports the absolute time performance profile over the 306 BC-tree test cases, that is, the fraction of cases each method solves to proven optimality within a given time. It is the counterpart to the solution quality profile of Figure 6 in the main text: FlowMIP closes more instances outright, while DP-FlowMIP returns the better solution on the majority of the cases that neither method solves to optimality.

J Complete per-case results

Table J.1 below gives the per-test-case results for all 306 cases (34 instances $\times$ 9 budget/coverage combinations). The same data is also available as a CSV file in the code repository for programmatic access.

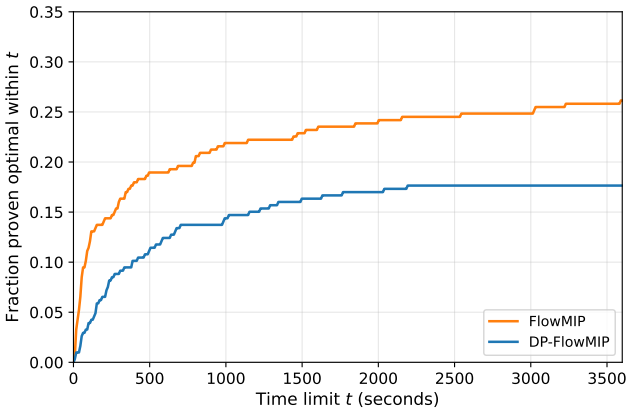


Figure I.1: Absolute time performance profile. The y-axis shows the fraction of the 306 test cases solved to optimality within time t seconds. FlowMIP reaches optimality on more instances (26.1% at $t = 3600$s) than DP-FlowMIP (17.6%). The lazy-cut variants rarely reach optimality within the time limit. This complements the solution quality profile in Figure 6: while FlowMIP proves optimality faster, DP-FlowMIP finds better solutions when the time limit is reached.

Table J.1: Complete results for all 306 test cases. ID = instance ID (Table F.1); LB = best solution found, UB = dual bound, Opt = optimality proven, Time = seconds. Budget factor $\beta \in \{0.3, 0.5, 0.7\}$; coverage radius factor $\gamma \in \{0.05, 0.1, 0.15\}$.

				BC-tree DP				FlowMIP			
ID	β	γ	$\lvert V\rvert$	LB	UB	Opt	Time	LB	UB	Opt	Time
1	0.3	0.05	152	109	227	No	3519	117	151	No	3606
1	0.3	0.1	152	116	230	No	3519	118	151	No	3607
1	0.3	0.15	152	116	231	No	3519	116	150	No	3606
1	0.5	0.05	152	176	292	No	3517	177	225	No	3607
1	0.5	0.1	152	178	296	No	3518	182	222	No	3607
1	0.5	0.15	152	179	299	No	3518	178	224	No	3608
1	0.7	0.05	152	234	351	No	3511	227	279	No	3608
1	0.7	0.1	152	227	352	No	3512	227	278	No	3608
1	0.7	0.15	152	228	356	No	3513	229	277	No	3608
2	0.3	0.05	202	155	546	No	3568	159	192	No	3611
2	0.3	0.1	202	157	596	No	3393	156	191	No	3611
2	0.3	0.15	202	156	595	No	3174	152	189	No	3611
2	0.5	0.05	202	239	785	No	3566	232	287	No	3612
2	0.5	0.1	202	241	810	No	3567	232	285	No	3612
2	0.5	0.15	202	241	817	No	3568	234	283	No	3612
2	0.7	0.05	202	307	912	No	3565	303	347	No	3613
2	0.7	0.1	202	306	937	No	3567	281	346	No	3614
2	0.7	0.15	202	307	949	No	3568	293	345	No	3614
3	0.3	0.05	223	178	622	No	3569	176	219	No	3615
3	0.3	0.1	223	177	646	No	3570	171	216	No	3615
3	0.3	0.15	223	176	653	No	3571	172	217	No	3615

Continued on next page

Table J.1 continued

				BC-tree DP				FlowMIP			
ID	β	γ	$\|V\|$	LB	UB	Opt	Time	LB	UB	Opt	Time
3	0.5	0.05	223	264	762	No	3566	263	320	No	3616
3	0.5	0.1	223	265	786	No	3570	255	317	No	3617
3	0.5	0.15	223	266	794	No	3572	256	317	No	3617
3	0.7	0.05	223	339	840	No	3567	325	390	No	3619
3	0.7	0.1	223	340	870	No	3571	331	388	No	3619
3	0.7	0.15	223	333	867	No	3572	322	389	No	3619
4	0.3	0.05	241	201	682	No	3418	207	245	No	3618
4	0.3	0.1	241	206	715	No	3574	204	247	No	3619
4	0.3	0.15	241	203	719	No	3304	201	246	No	3619
4	0.5	0.05	241	298	870	No	3573	293	351	No	3622
4	0.5	0.1	241	299	880	No	3575	279	349	No	3622
4	0.5	0.15	241	298	891	No	3575	294	351	No	3623
4	0.7	0.05	241	370	949	No	3572	367	429	No	3625
4	0.7	0.1	241	371	967	No	3575	367	425	No	3626
4	0.7	0.15	241	374	977	No	3576	364	425	No	3626
5	0.3	0.05	265	199	739	No	3556	203	266	No	3625
5	0.3	0.1	265	199	774	No	3558	194	264	No	3626
5	0.3	0.15	265	199	773	No	3558	199	266	No	3626
5	0.5	0.05	265	310	935	No	3557	307	388	No	3629
5	0.5	0.1	265	311	955	No	3559	301	389	No	3630
5	0.5	0.15	265	312	978	No	3559	299	385	No	3630
5	0.7	0.05	265	390	1021	No	3558	386	477	No	3632
5	0.7	0.1	265	388	1049	No	3559	384	469	No	3633
5	0.7	0.15	265	388	1064	No	3560	384	470	No	3635
6	0.3	0.05	241	193	663	No	3247	185	247	No	3620
6	0.3	0.1	241	194	659	No	3067	188	252	No	3620
6	0.3	0.15	241	196	679	No	3484	186	246	No	3620
6	0.5	0.05	241	297	839	No	3555	282	364	No	3624
6	0.5	0.1	241	299	867	No	3557	275	357	No	3625
6	0.5	0.15	241	298	867	No	3556	284	356	No	3625
6	0.7	0.05	241	367	911	No	3556	367	429	No	3626
6	0.7	0.1	241	365	940	No	3556	361	430	No	3626
6	0.7	0.15	241	377	961	No	3557	363	429	No	3627
7	0.3	0.05	289	222	1134	No	3563	233	292	No	3634
7	0.3	0.1	289	228	1163	No	3562	233	302	No	3634
7	0.3	0.15	289	222	1170	No	3563	233	296	No	3634
7	0.5	0.05	289	351	1500	No	3562	350	441	No	3639
7	0.5	0.1	289	354	1547	No	3563	320	448	No	3641
7	0.5	0.15	289	352	1547	No	3563	332	447	No	3640
7	0.7	0.05	289	444	1719	No	3563	448	544	No	3644
7	0.7	0.1	289	436	1745	No	3563	445	543	No	3645
7	0.7	0.15	289	441	1740	No	3563	443	541	No	3647
8	0.3	0.05	216	155	669	No	3469	163	197	No	3611
8	0.3	0.1	216	157	729	No	3434	164	197	No	3612
8	0.3	0.15	216	158	743	No	3350	163	195	No	3612
8	0.5	0.05	216	249	1011	No	3229	248	299	No	3613
8	0.5	0.1	216	248	1066	No	3473	245	297	No	3613
8	0.5	0.15	216	249	1093	No	3256	248	298	No	3613
8	0.7	0.05	216	325	1203	No	3187	315	377	No	3615
8	0.7	0.1	216	319	1244	No	3566	314	373	No	3615
8	0.7	0.15	216	313	1258	No	3568	312	375	No	3615
9	0.3	0.05	251	176	242	No	3450	173	214	No	3614
9	0.3	0.1	251	175	242	No	3450	172	215	No	3615
9	0.3	0.15	251	176	234	No	3450	173	214	No	3615
9	0.5	0.05	251	262	347	No	3447	263	324	No	3618
9	0.5	0.1	251	261	338	No	3449	255	321	No	3617
9	0.5	0.15	251	257	338	No	3451	256	320	No	3618
9	0.7	0.05	251	344	431	No	3447	336	404	No	3618
9	0.7	0.1	251	336	429	No	3449	333	407	No	3620

Continued on next page

Table J.1 continued

				BC-tree DP				FlowMIP			
ID	β	γ	$\|V\|$	LB	UB	Opt	Time	LB	UB	Opt	Time
9	0.7	0.15	251	329	428	No	3450	329	401	No	3620
10	0.3	0.05	51	31	31	Yes	39	31	31	Yes	2
10	0.3	0.1	51	31	31	Yes	98	31	31	Yes	11
10	0.3	0.15	51	28	28	Yes	133	28	28	Yes	15
10	0.5	0.05	51	48	48	Yes	181	48	48	Yes	52
10	0.5	0.1	51	48	48	Yes	327	48	48	Yes	57
10	0.5	0.15	51	48	48	Yes	496	48	48	Yes	114
10	0.7	0.05	51	62	62	Yes	227	62	62	Yes	76
10	0.7	0.1	51	64	64	Yes	386	64	64	Yes	986
10	0.7	0.15	51	64	64	Yes	583	64	64	Yes	683
11	0.3	0.05	66	43	43	Yes	670	43	43	Yes	45
11	0.3	0.1	66	43	43	Yes	1152	43	43	Yes	38
11	0.3	0.15	66	43	43	Yes	1766	43	43	Yes	47
11	0.5	0.05	66	67	84	No	1648	67	67	Yes	192
11	0.5	0.1	66	66	83	No	1775	66	66	Yes	491
11	0.5	0.15	66	66	90	No	1883	66	66	Yes	421
11	0.7	0.05	66	86	106	No	1483	86	86	Yes	1467
11	0.7	0.1	66	85	107	No	1607	85	90	No	3601
11	0.7	0.15	66	85	106	No	1721	85	86	No	3601
12	0.3	0.05	78	51	51	Yes	270	51	51	Yes	30
12	0.3	0.1	78	50	50	No	488	50	50	Yes	53
12	0.3	0.15	78	50	50	Yes	696	50	50	Yes	51
12	0.5	0.05	78	79	79	Yes	467	79	79	Yes	304
12	0.5	0.1	78	79	79	No	887	79	79	Yes	471
12	0.5	0.15	78	79	79	Yes	1219	79	79	Yes	292
12	0.7	0.05	78	100	100	Yes	502	100	100	Yes	2540
12	0.7	0.1	78	101	101	No	983	101	101	Yes	3026
12	0.7	0.15	78	101	101	Yes	1288	101	101	Yes	2000
13	0.3	0.05	62	33	33	Yes	48	33	33	Yes	12
13	0.3	0.1	62	33	33	Yes	114	33	33	Yes	15
13	0.3	0.15	62	32	32	Yes	149	32	32	Yes	14
13	0.5	0.05	62	54	54	Yes	144	54	54	Yes	39
13	0.5	0.1	62	53	53	Yes	388	53	53	Yes	113
13	0.5	0.15	62	52	52	Yes	540	52	52	Yes	101
13	0.7	0.05	62	72	72	Yes	230	72	72	Yes	341
13	0.7	0.1	62	71	71	Yes	667	71	71	Yes	263
13	0.7	0.15	62	71	71	Yes	979	71	71	Yes	206
14	0.3	0.05	116	76	150	No	2623	76	87	No	3603
14	0.3	0.1	116	76	185	No	2638	76	87	No	3602
14	0.3	0.15	116	76	199	No	2634	76	86	No	3603
14	0.5	0.05	116	117	245	No	2495	117	137	No	3603
14	0.5	0.1	116	117	282	No	2498	117	136	No	3603
14	0.5	0.15	116	117	296	No	2496	117	136	No	3603
14	0.7	0.05	116	151	280	No	2388	154	174	No	3603
14	0.7	0.1	116	154	346	No	2402	154	174	No	3603
14	0.7	0.15	116	154	354	No	2409	153	172	No	3603
15	0.3	0.05	74	44	44	Yes	43	44	44	Yes	53
15	0.3	0.1	74	42	42	Yes	51	42	42	Yes	89
15	0.3	0.15	74	42	42	Yes	61	42	42	Yes	79
15	0.5	0.05	74	66	66	Yes	300	66	66	Yes	372
15	0.5	0.1	74	65	65	Yes	1015	65	65	Yes	1601
15	0.5	0.15	74	64	64	Yes	1626	64	64	Yes	901
15	0.7	0.05	74	85	85	Yes	2031	85	85	Yes	1442
15	0.7	0.1	74	84	87	No	3420	84	88	No	3601
15	0.7	0.15	74	84	87	No	3421	84	88	No	3601
16	0.3	0.05	87	48	48	Yes	226	48	48	Yes	19
16	0.3	0.1	87	47	47	No	598	47	47	Yes	34
16	0.3	0.15	87	47	47	Yes	986	47	47	Yes	38
16	0.5	0.05	87	74	74	Yes	634	74	74	Yes	115

Continued on next page

Table J.1 continued

				BC-tree DP				FlowMIP			
ID	β	γ	$\|V\|$	LB	UB	Opt	Time	LB	UB	Opt	Time
16	0.5	0.1	87	73	110	No	1660	73	73	Yes	293
16	0.5	0.15	87	72	121	No	2013	72	72	Yes	338
16	0.7	0.05	87	97	97	Yes	1335	97	97	Yes	798
16	0.7	0.1	87	96	157	No	1988	96	96	Yes	3018
16	0.7	0.15	87	95	164	No	2132	95	95	Yes	3226
17	0.3	0.05	159	123	136	No	3460	122	138	No	3605
17	0.3	0.1	159	123	166	No	3467	123	138	No	3604
17	0.3	0.15	159	123	143	No	3467	123	138	No	3605
17	0.5	0.05	159	182	205	No	3461	180	207	No	3605
17	0.5	0.1	159	183	229	No	3462	180	204	No	3605
17	0.5	0.15	159	182	230	No	3462	181	205	No	3605
17	0.7	0.05	159	231	279	No	3458	229	255	No	3606
17	0.7	0.1	159	231	279	No	3459	228	255	No	3606
17	0.7	0.15	159	229	282	No	3461	225	253	No	3606
18	0.3	0.05	47	38	38	Yes	4	38	38	Yes	4
18	0.3	0.1	47	37	37	No	13	37	37	Yes	13
18	0.3	0.15	47	37	37	Yes	9	37	37	Yes	13
18	0.5	0.05	47	57	57	Yes	14	57	57	Yes	37
18	0.5	0.1	47	56	56	Yes	214	56	56	Yes	142
18	0.5	0.15	47	56	56	Yes	143	56	56	Yes	153
18	0.7	0.05	47	71	71	Yes	51	71	71	Yes	58
18	0.7	0.1	47	71	71	Yes	424	71	71	Yes	392
18	0.7	0.15	47	70	70	Yes	574	70	70	Yes	627
19	0.3	0.05	170	130	221	No	3526	128	152	No	3607
19	0.3	0.1	170	129	229	No	3527	129	151	No	3607
19	0.3	0.15	170	129	228	No	3527	125	152	No	3607
19	0.5	0.05	170	199	296	No	3524	197	232	No	3608
19	0.5	0.1	170	198	298	No	3525	198	230	No	3608
19	0.5	0.15	170	199	301	No	3524	195	227	No	3608
19	0.7	0.05	170	254	349	No	3516	244	290	No	3608
19	0.7	0.1	170	253	356	No	3519	247	290	No	3608
19	0.7	0.15	170	250	355	No	3520	247	289	No	3608
20	0.3	0.05	56	44	44	No	33	44	44	Yes	15
20	0.3	0.1	56	44	44	Yes	75	44	44	Yes	23
20	0.3	0.15	56	42	42	Yes	95	42	42	Yes	26
20	0.5	0.05	56	69	69	No	68	69	69	Yes	29
20	0.5	0.1	56	67	67	Yes	146	67	67	Yes	84
20	0.5	0.15	56	67	67	Yes	210	67	67	Yes	90
20	0.7	0.05	56	86	86	No	80	86	86	Yes	60
20	0.7	0.1	56	86	86	Yes	165	86	86	Yes	274
20	0.7	0.15	56	86	86	Yes	249	86	86	Yes	360
21	0.3	0.05	216	130	592	No	2865	134	160	No	3610
21	0.3	0.1	216	130	661	No	2868	135	162	No	3610
21	0.3	0.15	216	127	698	No	2888	134	157	No	3610
21	0.5	0.05	216	207	851	No	2811	209	250	No	3612
21	0.5	0.1	216	207	892	No	2852	208	249	No	3612
21	0.5	0.15	216	203	947	No	2882	205	246	No	3612
21	0.7	0.05	216	272	988	No	2817	274	321	No	3613
21	0.7	0.1	216	273	1056	No	2853	274	326	No	3613
21	0.7	0.15	216	271	1086	No	2909	272	325	No	3613
22	0.3	0.05	118	90	187	No	2804	91	96	No	3603
22	0.3	0.1	118	90	200	No	2799	91	95	No	3603
22	0.3	0.15	118	91	191	No	2799	91	96	No	3603
22	0.5	0.05	118	129	258	No	2629	134	143	No	3603
22	0.5	0.1	118	132	258	No	2646	132	142	No	3603
22	0.5	0.15	118	132	255	No	2648	132	142	No	3603
22	0.7	0.05	118	165	290	No	2492	170	182	No	3604
22	0.7	0.1	118	166	292	No	2533	168	180	No	3604
22	0.7	0.15	118	166	285	No	2481	166	180	No	3604

Continued on next page

Table J.1 continued

				BC-tree DP				FlowMIP			
ID	β	γ	$\|V\|$	LB	UB	Opt	Time	LB	UB	Opt	Time
23	0.3	0.05	93	66	114	No	3045	66	66	Yes	781
23	0.3	0.1	93	64	122	No	3032	64	64	Yes	800
23	0.3	0.15	93	64	111	No	3033	64	64	Yes	942
23	0.5	0.05	93	98	170	No	2639	98	103	No	3602
23	0.5	0.1	93	97	176	No	2655	97	102	No	3602
23	0.5	0.15	93	96	173	No	2664	96	101	No	3602
23	0.7	0.05	93	122	204	No	2369	121	131	No	3602
23	0.7	0.1	93	121	203	No	2387	121	129	No	3602
23	0.7	0.15	93	121	200	No	2391	121	129	No	3602
24	0.3	0.05	337	257	1287	No	3568	258	343	No	3646
24	0.3	0.1	337	247	1301	No	3568	256	347	No	3647
24	0.3	0.15	337	252	1297	No	3568	260	343	No	3648
24	0.5	0.05	337	405	1700	No	3568	390	516	No	3657
24	0.5	0.1	337	407	1735	No	3568	381	516	No	3659
24	0.5	0.15	337	404	1720	No	3569	389	512	No	3657
24	0.7	0.05	337	500	1950	No	3568	512	638	No	3666
24	0.7	0.1	337	514	162665	No	3568	511	637	No	3666
24	0.7	0.15	337	503	163539	No	3569	511	637	No	3667
25	0.3	0.05	70	41	41	Yes	1490	41	41	Yes	92
25	0.3	0.1	70	40	40	Yes	2185	40	40	Yes	246
25	0.3	0.15	70	40	54	No	2885	40	40	Yes	99
25	0.5	0.05	70	64	113	No	2829	64	64	Yes	1139
25	0.5	0.1	70	65	121	No	2819	65	65	Yes	1517
25	0.5	0.15	70	64	118	No	2830	64	64	Yes	827
25	0.7	0.05	70	83	118	No	2492	83	88	No	3601
25	0.7	0.1	70	85	124	No	2507	85	89	No	3601
25	0.7	0.15	70	85	138	No	2498	85	88	No	3601
26	0.3	0.05	162	106	295	No	2768	107	122	No	3605
26	0.3	0.1	162	105	405	No	2761	107	121	No	3605
26	0.3	0.15	162	104	453	No	2762	107	121	No	3605
26	0.5	0.05	162	167	535	No	2678	170	193	No	3606
26	0.5	0.1	162	166	616	No	2672	170	192	No	3606
26	0.5	0.15	162	164	656	No	2693	168	191	No	3606
26	0.7	0.05	162	220	638	No	2636	223	250	No	3606
26	0.7	0.1	162	221	807	No	2657	222	246	No	3606
26	0.7	0.15	162	220	812	No	2677	220	246	No	3606
27	0.3	0.05	165	100	341	No	2769	106	123	No	3605
27	0.3	0.1	165	100	385	No	2762	105	120	No	3606
27	0.3	0.15	165	100	392	No	2771	104	120	No	3606
27	0.5	0.05	165	161	504	No	2683	165	192	No	3606
27	0.5	0.1	165	161	581	No	2701	162	190	No	3606
27	0.5	0.15	165	160	587	No	2686	163	190	No	3606
27	0.7	0.05	165	216	610	No	2640	216	248	No	3606
27	0.7	0.1	165	212	680	No	2687	214	248	No	3607
27	0.7	0.15	165	211	693	No	2689	212	246	No	3607
28	0.3	0.05	116	85	162	No	2638	85	85	Yes	3589
28	0.3	0.1	116	85	172	No	2639	85	85	Yes	1843
28	0.3	0.15	116	84	177	No	2604	84	84	Yes	2155
28	0.5	0.05	116	124	228	No	2253	124	132	No	3603
28	0.5	0.1	116	123	214	No	2281	123	131	No	3603
28	0.5	0.15	116	122	227	No	2243	122	129	No	3603
28	0.7	0.05	116	155	258	No	2049	154	168	No	3603
28	0.7	0.1	116	155	256	No	2074	154	166	No	3603
28	0.7	0.15	116	154	259	No	2050	154	164	No	3603
29	0.3	0.05	162	111	313	No	2652	111	126	No	3605
29	0.3	0.1	162	111	340	No	2655	111	128	No	3605
29	0.3	0.15	162	111	341	No	2601	112	126	No	3605
29	0.5	0.05	162	173	448	No	2359	172	196	No	3606
29	0.5	0.1	162	171	468	No	2392	172	194	No	3606

Continued on next page

Table J.1 continued

				BC-tree DP				FlowMIP			
ID	β	γ	$\|V\|$	LB	UB	Opt	Time	LB	UB	Opt	Time
29	0.5	0.15	162	171	474	No	2351	173	197	No	3606
29	0.7	0.05	162	226	546	No	2237	226	252	No	3607
29	0.7	0.1	162	224	561	No	2282	224	251	No	3607
29	0.7	0.15	162	222	562	No	2258	223	253	No	3607
30	0.3	0.05	162	135	285	No	2553	135	139	No	3605
30	0.3	0.1	162	133	273	No	2528	133	137	No	3605
30	0.3	0.15	162	133	292	No	2519	133	137	No	3605
30	0.5	0.05	162	194	354	No	2229	194	203	No	3606
30	0.5	0.1	162	193	353	No	2236	193	203	No	3606
30	0.5	0.15	162	193	365	No	2212	193	202	No	3606
30	0.7	0.05	162	241	404	No	2104	243	253	No	3607
30	0.7	0.1	162	240	400	No	2121	241	250	No	3607
30	0.7	0.15	162	241	417	No	2114	240	251	No	3607
31	0.3	0.05	237	188	479	No	3563	185	225	No	3617
31	0.3	0.1	237	187	475	No	3565	182	224	No	3617
31	0.3	0.15	237	188	473	No	3564	176	223	No	3617
31	0.5	0.05	237	273	581	No	3560	270	321	No	3619
31	0.5	0.1	237	273	583	No	3563	265	322	No	3621
31	0.5	0.15	237	275	587	No	3563	266	321	No	3620
31	0.7	0.05	237	341	646	No	3560	345	400	No	3623
31	0.7	0.1	237	342	661	No	3566	334	398	No	3623
31	0.7	0.15	237	342	654	No	3565	330	396	No	3624
32	0.3	0.05	237	209	459	No	3563	207	230	No	3617
32	0.3	0.1	237	209	477	No	3564	200	229	No	3617
32	0.3	0.15	237	209	468	No	3564	205	230	No	3617
32	0.5	0.05	237	303	551	No	3560	298	327	No	3620
32	0.5	0.1	237	301	572	No	3562	299	327	No	3620
32	0.5	0.15	237	302	576	No	3560	294	326	No	3619
32	0.7	0.05	237	367	628	No	3559	369	401	No	3622
32	0.7	0.1	237	367	637	No	3564	366	396	No	3623
32	0.7	0.15	237	367	635	No	3564	347	396	No	3622
33	0.3	0.05	239	189	503	No	3018	190	217	No	3617
33	0.3	0.1	239	188	503	No	3004	189	215	No	3618
33	0.3	0.15	239	188	499	No	3004	189	216	No	3618
33	0.5	0.05	239	284	616	No	2991	283	318	No	3620
33	0.5	0.1	239	283	617	No	2976	279	319	No	3620
33	0.5	0.15	239	282	594	No	2986	285	316	No	3620
33	0.7	0.05	239	357	693	No	3043	352	399	No	3622
33	0.7	0.1	239	359	696	No	3009	344	396	No	3623
33	0.7	0.15	239	359	694	No	3003	353	398	No	3623
34	0.3	0.05	262	213	554	No	2968	202	243	No	3622
34	0.3	0.1	262	214	568	No	2977	204	243	No	3622
34	0.3	0.15	262	212	559	No	2979	202	241	No	3622
34	0.5	0.05	262	310	665	No	2919	311	347	No	3625
34	0.5	0.1	262	311	659	No	2929	311	347	No	3625
34	0.5	0.15	262	308	682	No	2924	312	346	No	3625
34	0.7	0.05	262	386	742	No	2966	388	428	No	3628
34	0.7	0.1	262	390	744	No	2977	386	427	No	3628
34	0.7	0.15	262	388	754	No	2980	387	424	No	3629