technische universität
dortmund

2-InterConnected Facility Location: Specifications, Complexity Results, and Exact Solutions

Markus Chimani
Maria Kandyba
Maren Martens

Algorithm Engineering Report
TR09-1-008
Dec. 2009
ISSN 1864-4503

Faculty of Computer Science
Algorithm Engineering (Ls11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

2-InterConnected Facility Location: Specifications, Complexity Results, and Exact Solutions

Markus Chimani[1], Maria Kandyba[1⋆], and Maren Martens[2⋆⋆]

[1] Faculty of Computer Science, TU Dortmund, Otto-Hahn-Str. 14, 44227 Dortmund, Germany,
{markus.chimani,maria.kandyba}@tu-dortmund.de
[2] Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, martens@zib.de

Technical Report: TR09-1-008
Chair XI Algorithm Engineering, TU Dortmund
December 2009

Abstract. Connected facility location problems combine cost-efficient facility placement (including cheap client-to-facility-connection) with the requirement to connect the facilities among each other. Such network design problems arise, e.g., in telecommunication applications where networks consist of a central core and local clients that have to be connected to it. In practice, reliability of the core network is a central issue, and we may hence require the core network to be (at least) 2-interconnected, i.e., there are two disjoint paths within the core network between every pair of core nodes.
We establish the problem class of 2-interConnected Facility Location (2-iCFL), categorize its central variants, and prove that they are hard to approximate. However, as our computational results show, cut-based ILP formulations, also presented herein, allow us to effectively solve such problems to optimality for hundreds of nodes. On the way, we establish simple but exhaustive characterizations for problem instances admitting feasible solutions for 2-iCFL. These characterizations are constructive and can be used for algorithmic feasibility checks, preprocessing steps, and heuristics.

1 Introduction

Combined facility location and connectivity problems have gained a lot of attention in the past decade. In this paper, we consider problems where we have to open facilities (chosing from a given set of locations) and connect every other location (a client) to one of them. Our motivation for considering such problems stems from various projects with industrial cooperation where the goal is the cost-efficient design of optical telecommunication networks. A typical real-world network contains a highly secured core and clients that are directly connected to it. For reliability, it is common to require that the core network is 2-connected whereas a client

⋆ Supported by Deutsche Telekom Stiftung
⋆⋆ Supported by the Federal Ministry of Education and Research (BMBF), Germany, as part of the project "EUREKA 100GET Technologies"

needs only a simple connection to some node in the core (a facility). Opening facilities and establishing connections cause costs that we want to minimize. As this problem is of high relevance for topology planning of modern glass fiber networks, it can also be found in theory oriented books, such as [10] where various problems, which are important in the context of communication network design, are introduced.

Problem Definition and Notation. Given an undirected connected graph $G = (V, E)$, a set $P \subseteq V$ of possible facilities (usually $|P| \geq 3$), nonnegative opening costs $f : P \to \mathbb{R}_0^+$ and nonnegative edge costs $c : E \to \mathbb{R}_0^+$. A *2-interconnected facility location problem* (2-iCFL) is to choose a subset $\mathcal{F} \subseteq P$ where we open facilities and to connect each node $u \notin \mathcal{F}$ to some node $v \in \mathcal{F}$ directly via an edge $\{u, v\} \in E$. Furthermore, we require a survivable connection between each pair of facilities: For each pair of nodes $u, v \in \mathcal{F}$ we require two disjoint paths, only visiting facilities. In the resulting network $N = (V, E')$ with $E' \subseteq E$ being the established connections, we minimize the objective function $\sum_{v \in \mathcal{F}} f_v + \sum_{e \in E' : |e \cap \mathcal{F}| = 1} c_e + M \cdot \sum_{e \in E' : |e \cap \mathcal{F}| = 2} c_e$, where $M \geq 1$ reflects possibly higher costs for core connections. We call the network $N[\mathcal{F}]$ spanned purely by the facilities, the *facility network* of the solution. In the following, we use $n = |V|$ and $m = |E|$ to denote the number of nodes and edges in G, respectively. We use $E(G')$ and $V(G')$ to denote the edges and nodes in a subgraph G' of G. Furthermore, we write $N(v)$ for the set of nodes being adjacent to $v \in V$ in G. (To denote the neighborhood of v in a graph other than G, we index $N(v)$ with the respective graph.)

Depending on the exact definition of the disjointness of paths between facilities, we can define the following problems: We are dealing with a *2-node-interconnected facility location problem* (2N-iCFL) if we require two node-disjoint paths between each pair of facilities, i.e., the facility network has to be 2-node-connected (and not consist of only a single edge). Relaxing the node-disjointness to edge-disjointness we obtain the *2-edge-interconnected facility location problem* (2E-iCFL). In the following, we write only 2-iCFL when our results hold for both versions of connectivity.

In real-world applications of 2-iCFL the input may already contain a fixed facility node r. We refer to such 2-iCFL problems as *rooted 2-interconnected facility location* (r-2-iCFL), and can analogously distinguish between r-2N-iCFL and r-2E-iCFL. We say a problem is *degree-bounded* if we allow only solutions where every node has degree at most Δ. This comes from the practical background that, e.g., a router in a telecommunication network can only be attached to some maximum number of other devices, due to its number of physical slots, plugs, or sockets.

Generally, since the edge costs are nonnegative, it suffices to search for solutions with the following structural properties (depending on whether we consider node- or edge-connectivity, respectively):

(S/1) All nodes $v \in \mathcal{F}$ belong to the same 2-node(/edge)-connected component $N[\mathcal{F}]$.
(S/2) $N[\mathcal{F}]$ is the only non-trivial 2-node(/edge)-connected component in N.
(S/3) All nodes $v \notin \mathcal{F}$ have degree 1.

Related Results from the Literature. To the best of our knowledge, 2-iCFL has only been considered in [10] where it is studied in the context of complete facility networks and for $M = 1$. For this problem, an undirected ILP formulation is used to solve instances of up to 25 nodes with a fixed number of open facilities. *Connected facility location* (CFL), however, and variants thereof have been widely studied. In contrast to the definition in this paper, for CFL it

is usually sufficient that a Steiner tree connects the open facilities with each other; the network spanned only by the open facilities needs not to be connected. Furthermore, general CFL allows to connect clients to facilities via paths. In optical telecommunication applications, however, it is common to ask for a network where open facilities are (2-)connected amongst themselves and clients are directly connected to the core. Hence we consider the 2-iCFL problem, and do not call it 2-connected facility location, as we would reserve that name for the case considering Steiner nodes. We are not aware of any research specifically conducted on this latter problem class. Note that the abbreviation should not be confused with *k-CFL* [3, 15], where it is used as a CFL problem where at most k facilities are allowed to be opened.

Problems that implicitly require 2-connectivity amongst open facilities have, e.g., been studied as *Ring Star* or *tour-CFL* problems; see, e.g., [3, 5, 6]. In both problem types, the open facilities have to lie on a common cycle (aka. ring). The former problem, like ours, requires that the clients are connected via a single edge to this ring; furthermore it contains a root node that has to lie on the ring. In this context one could also call r-2-iCFL a *Block Star* problem, for a *block* being a maximal 2-node-connected component. Note that in complete Euclidean graphs, a minimal block will in fact always form a simple cycle [9].

Recently, Eisenbrand et al. [3] gave the currently best approximation algorithms for CFL, k-CFL, and tour-CFL (amongst others), obtaining expected ratios of 4, 6.85, and 4.12, respectively. See that paper for an overview on further related results. For an overview on the Ring Star problem, as well as ILP formulations and polyhedral results see, e.g., [5, 6]. For ILP approaches—and practical hybridizations with metaheuristics—in the context of CFL see, e.g., [8, 16]; efficient heuristics can, e.g., be found in [10]. As we will discuss when establishing our ILP formulations, 2-iCFL is also closely related to the problem of finding optimal $\{0, 1, 2\}$-survivable networks [2, 11, 14]. Recent approximation results for such problems can, e.g., be found in [4, 7].

Contribution of this Paper. In this paper, we establish the problem class of 2-iCFL and categorize its central variants. While it is straight-forward that all such variants are NP-hard, we show that even strong approximations are NP-hard (Section 2). For degree-bounded 2-iCFL, already checking if an instance allows any feasible solution is NP-hard. On the other hand, we show that feasibility of 2-iCFL variants without degree bounds can be tested in linear time (Section 3). Our feasibility check also gives rise to preprocessing routines and heuristics. In Section 4, we show how 2-iCFL can be solved to optimality using strong ILP formulations and we conduct a set of experiments in Section 5. All our results can be generalized to the case with two different cost functions for client to core connections and intercore connections.

2 Complexity

For the following results on the complexity of 2-iCFL, we restrict our considerations to the case where $M = 1$. Note that this can be done without loss of generality.

Theorem 1. *It is NP-hard to approximate 2-iCFL within a factor $c \log |V|$, for some constant $c > 0$. This is true even in complete graphs.*

Proof. We use a reduction of MINIMUM DOMINATING SET. Recall that in MINIMUM DOMINATING SET we are given a graph $H = (V_H, E_H)$ and we are looking for a subset $D \subseteq V_H$

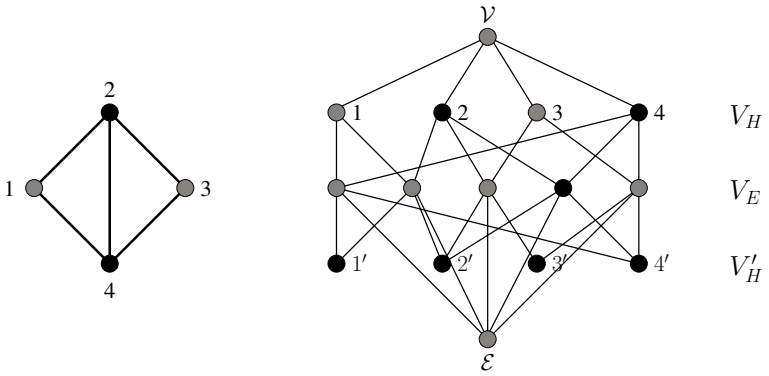

Fig. 1. Reduction of MINIMUM DOMINATING SET to 2-iCFL with H on the left and G on the right. A minimum dominating set and the corresponding open facilities are drawn gray.

of minimum cardinality such that for all $u \in V_H \setminus D$ there is a $v \in D$ for which $(u, v) \in E_H$. MINIMUM DOMINATING SET is known not to be approximable within $c \log |V_H|$, for some $c > 0$, unless P=NP [12]. W.l.o.g., we consider only connected instances of MINIMUM DOMINATING SET where $|E_H| \geq 2$ and every dominating set has cardinality at least 2.

We consider a 2-iCFL instance on the graph $G = (V, E)$ where V is the union of V_H, a copy V'_H of V_H, a node set V_E containing a node v_e for every edge $e \in E_H$, and two nodes $\mathcal{V}$, $\mathcal{E}$. For a node $v \in V_H$, we denote its copy in V'_H as v'. We specify the set of edges in G as $E := \bigcup_{v \in V_H} \{(v, \mathcal{V})\} \cup \bigcup_{e=(v,w) \in E_H} \{(v_e, \mathcal{E}), (v_e, v), (v_e, w), (v_e, v'), (v_e, w')\}$ (cf. Figure 1). All edge costs are set to 0. The set of potential facilities is $P := V \setminus V'_H$, where the opening costs are 1 on potential facilities from V_H, and 0 elsewhere.

We show that every feasible 2-iCFL solution on G can efficiently be transformed into a dominating set for H and vice versa: Consider the set $\mathcal{F}$ of open facilities in any feasible 2-iCFL solution. Since $N(\mathcal{V}) = V_H$, either $\mathcal{V} \in \mathcal{F}$ or $V_H \cap \mathcal{F} \neq \emptyset$. Set $D := V_H \cap \mathcal{F}$. For any $u \in V_H \setminus D$ its copy $u' \in V'_H$ must be connected to some $v_e \in V_E \cap \mathcal{F}$ with $(u', v_e) \in E$. By the definition of feasible 2-iCFL solutions, there are at least two disjoint paths within the graph induced by $\mathcal{F}$ that connect v_e to any other open facility (in particular to any in $(V_H \cup \{\mathcal{V}\}) \cap \mathcal{F} \neq \emptyset$) and at most one such path may traverse $\mathcal{E}$. Thus, v_e must be connected to some open $v \in V_H$ and therefore $v \in D$ with $(u, v) \in E_H$.

Now consider any dominating set $D \subseteq V_H$ in H. We set $\mathcal{F} := D \cup \{\mathcal{E}, \mathcal{V}\} \cup \{v_e \in V_E \mid N(v_e) \cap D \neq \emptyset\}$ and open all edges in E that connect two open facilities. Trivially, this connects all nodes in V_H and V_E to some open facility, namely $\mathcal{V}$ or $\mathcal{E}$ respectively. For each $u' \in V'_H$, we show $N(u') \cap \mathcal{F} \neq \emptyset$, what allows us to also connect u' to $\mathcal{F}$, thus constructing a feasible 2-iCFL solution: If u''s original $u \in V_H$ is in D, we opened some node $v_e \in V_E$ with $v_e \in N(u) \cap N(u')$. If, on the other hand, $u \notin D$, there is some $v \in D$ with $(u, v) \in E_H$ and so we have $v_{(u,v)} \in \mathcal{F}$ with $v_{(u,v)} \in N(u')$. It remains to show that for any two nodes $u, v \in \mathcal{F}$ there are two disjoint paths connecting them. By our assumption that H is connected and $|E_H|, |D| \geq 2$, there are $\bar{u}, \bar{v} \in D$ with $\bar{u} \neq \bar{v}$ and $|(N(\bar{u}) \cup N(\bar{v})) \cap V_E \cap \mathcal{F}| \geq 2$. Therefore, there exist two disjoint paths from $\mathcal{V}$ to $\mathcal{E}$, or, in other words, a simple cycle containing $\mathcal{V}$ and $\mathcal{E}$. A simple case distinction gives the existence of similar cycles for the other open facilities.

Since the opening costs for a facility are 1 iff the facility is in V_H, the cost of the set of open facilities is the same as the cardinality of the corresponding dominating set. Furthermore $|V| = 2|V_H| + |E_H| + 2$, and so a $c \log |V|$-approximation for 2-iCFL would lead to a $c' \log |V_H|$-approximation for MINIMUM DOMINATING SET, for $c, c' > 0$. This proves the first part of the theorem. The results in this proof still hold when we make G a complete graph by introducing all missing edges with (appropriately) high cost. A $c \log |V|$-approximation to the 2-iCFL problem would never use any of those heavy edges as it is always feasible to open all possible facilities and use all 0 cost edges. □

When we add a degree bound, 2-iCFL becomes much harder, also in complete graphs.

Theorem 2. *For any constant degree bound $\Delta \geq 2$, it is NP-hard to approximate degree-bounded 2-iCFL within a factor $c^{p(n)}$, where $c > 1$ is a constant and $p(n)$ is a polynomial in n. This is already true in complete graphs.*

Proof. Part A. In a first step we use a reduction of HAMILTONIAN CYCLE to prove the claim only for $\Delta = 2$. Recall that in HAMILTONIAN CYCLE we are given an undirected graph $H = (V_H, E_H)$ and we have to decide whether H allows a cycle that visits every node exactly once (w.l.o.g., $|V_H| \geq 4$.) We consider an instance of degree-bounded 2-iCFL in the complete graph $G = (V, E)$ with $V := V_H$. We define $K := (c^{p(n)} - 1)n + 2$ and choose the edge costs as $c_e = 1$, for all $e \in E_H$, and $c_e = K$, for all $e \in E \setminus E_H$. All opening costs for facilities are 0. (The coding length of this instance of 2-iCFL is polynomial in the coding length of H.)

Further note that $|V| \geq 4$ together with $\Delta = 2$ requires that every degree-bounded 2-iCFL solution in G contains at least two facilities. (Otherwise the solution would have to be a star with a central node of degree ≥ 3.) Closer examination yields that in fact every solution is a hamiltonian cycle: The degree bound restricts the set of feasible solutions to unions of simple cycles and paths. Since a solution has to be connected, we have exactly one simple path or cycle visiting every node. The 2-connectivity among facilities requires a cycle so that indeed every solution is a hamiltonian cycle. In particular, every degree-bounded 2-iCFL solution contains exactly n edges.

We assume by contradiction that there is a $c^{p(n)}$-approximation algorithm $\mathcal{A}$ for degree-bounded 2-iCFL and argue that $\mathcal{A}$ would solve HAMILTONIAN CYCLE in polynomial time: We show that a hamiltonian cycle exists in H iff $\mathcal{A}$ finds a solution for the instance of degree-bounded 2-iCFL of cost at most $nc^{p(n)}$. First assume that we have a hamiltonian cycle C in H. Note that C is a solution for degree-bounded 2-iCFL in G and so we have that the minimum cost of a solution is n. Since $\mathcal{A}$ is a $c^{p(n)}$-approximation algorithm, it finds a solution of cost at most $nc^{p(n)}$.

For the backward implication, assume that $\mathcal{A}$ finds a solution for the instance of degree-bounded 2-iCFL of cost at most $nc^{p(n)}$. By the discussion above, this solution has to form a hamiltonian cycle. Moreover, this cycle may not use any edges of cost K as otherwise its total cost would sum up to at least $K + (n-1) = nc^{p(n)} + 1$. Therefore, all edges have cost 1 and the cycle only uses edges in E_H.

Part B. It might look like it is essential for the proof of Theorem 2 to choose $\Delta = 2$. In the following we prove that we can achieve the same result with any constant degree bound $\Delta \geq 2$; cf. Figure 2.

Let $v_1, \ldots, v_h$ be the nodes in V_H. Then we create $(\Delta - 2)h$ new nodes denoted by v_{ij}, for $1 \leq i \leq h$ and $1 \leq j \leq (\Delta - 2)$, and let V be the union of V_H and the new nodes. We consider

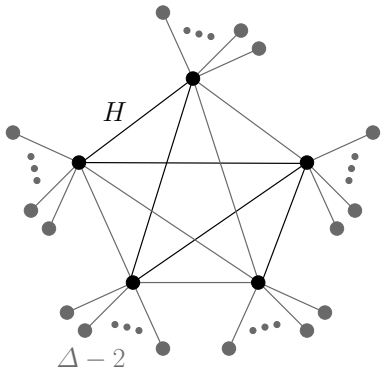

Fig. 2. Reduction of HAMILTONIAN CYCLE to degree-bounded 2-iCFL. We omit $\{v_i, v_{kj}\}$, for $i \neq k$, and $\{v_{ij}, v_{kl}\}$ for clearity.

a complete graph on V with the following cost function: $c_e = 1$, for all $e \in E_H$; $c_{\{v_i,v_{ij}\}} = 0$, for all $1 \leq i \leq h$ and $1 \leq j \leq (\Delta - 2)$; and cost K for all other edges.

The set of possible facilities in this new instance of degree-bounded 2-iCFL is V_H. Again we have all opening costs being 0. The degree bound is Δ. It turns out that every solution of cost at most $K + n - 2 = nc^{p(n)}$ is in fact a hamiltonian cycle in H plus arcs $\{v_i, v_{ij}\}$. This is because in such a solution we have to choose every node v_i as a facility and connect all nodes v_{ij} to it. (Otherwise we would have connection costs of at least K for the nodes v_{ij} plus at least cost $n - 1$ to get a connected graph among the nodes in V_H.) But then we have only a degree of 2 left for every node in V_H to 2-connect it to the other facilities. Therefore, we are back to the situation in Part A. □

If, in the preceding proof, we leave out every edge of cost K, it follows that there is a feasible solution to 2-iCFL iff H contains a Hamiltonian cycle, and hence:

Corollary 1. *It is NP-hard to decide whether an instance of degree-bounded 2-iCFL admits a feasible solution.*

3 Feasibility Check, Preprocessing, and Heuristic

In contrast to the degree-bounded case, we can test feasibility of input instances without degree bound in linear time. Furthermore, the testing algorithm can be used as a preprocessing routine to fix and delete edges and fix nodes as clients or facilities. Finally, the algorithm can also be used as a heuristic to obtain upper bounds for the problem.

3.1 Checking Feasibility for 2N-iCFL

Algorithm 1 shows how to decide if an input instance allows a feasible solution.

Let H be the subgraph of G induced by the potential facilities P. The 2-interconnected facility network in a solution, if any exists, has to be a subgraph of H. Therefore, H has to

be connected, otherwise we could not find a 2-interconnected facility network within it and attach all other nodes as clients to it. We compute the BC-tree $\mathcal{B}$ of H, i.e., $\mathcal{B}$ has a *block node* for each maximal 2-node-connected component (block) in H. Whenever some blocks share a vertex, there exists a *cut node* in $\mathcal{B}$ which is adjacent to the corresponding block nodes. We observe that a 2-node-interconnected facility network can only reside within a single block, say B^*, of $\mathcal{B}$. If B^* consists of only one single edge, we obtain the trivial subcase that exactly one of its nodes becomes a facility and all other nodes have to be connected to it as clients. Assume $|E(B^*)| > 1$ in the following. Any node not in this block has to become a client node and needs to get attached to some facility within B^*. Hence the cut nodes of B^* have to become facilities and all nodes in P not within B^* will be attached to their nearest cut node in B^*. Clearly, we can only find a feasible solution if every other block shares a cut node with B^*. This induces that B^* is the only block node which may have a degree greater than 1. Furthermore, we require that *any* node not in B^*—i.e., also considering the nodes $V \setminus P$—is directly incident to some node in B^*.

Unless $\mathcal{B}$ contains exactly two block nodes, the choice for B^* is simply any block node with degree at least 2; in case of a feasible instance this will be unique. If $\mathcal{B}$ has exactly two blocks, one can distinguish simple subcases. However, for brevity it shall be enough to state that we can simply test feasibility for both possible choices of B^* without sacrificing our runtime guarantee.

If the above requirements hold, we can select all nodes in B^* as facilities $\mathcal{F}$. Then, $G[\mathcal{F}] = B^*$ is a feasible 2-interconnected facility network. As we know for every other node that it is incident to some node in $\mathcal{F}$, we can choose any of the corresponding edges and obtain a feasible 2N-iCFL solution.

Theorem 3. *We can check whether a given instance allows a feasible solution for 2N-iCFL in linear time.*

Proof. As described above, Algorithm 1 returns false iff no feasible solution exists. It remains to prove the linear runtime. Observe that testing connectivity and 2-node-connectivity can be done in linear time using augmented DFS traversals. Computing a BC-tree, and checking if there is at most one tree node with degree larger than 1, is linear as well. If the maximum degree is 1, there are exactly two blocks in H, and the subsequent code is (independently) run twice. The for-each loop is executed a linear number of times and hence it remains to check the neighborhood of a node in constant time. In the beginning of the algorithm, we can count for each node how many of its neighbors belong to P. Whenever a node of P is fixed as a client, we reduce this count by one for all its neighbors. The check within the loop then requires only constant time by inspecting v's counter. Since this counting scheme touches each edge at most four times (one increase and one decrease per each adjacent node), realizing the scheme takes only linear time, and we obtain the theorem. □

3.2 Checking Feasibility for 2E-iCFL

Reusing the ideas from checking feasibility for 2N-iCFL, we obtain a similar algorithm for 2E-iCFL. The central difference between the two problem variants is that 2E-iCFL allows the 2-edge-interconnected facility network to span over multiple connected blocks (maximal 2-node-connected components); clearly, two blocks are *connected* if they share a common cut

Algorithm 1 Checking feasibility of 2N-iCFL input instances.

Require: Input instance $G = (V, E)$ for 2N-iCFL with potential facilities $P \subseteq V$
Ensure: Returns ***true*** if the instance allows a feasible solution, ***false*** otherwise. In the first case, F_c and F_f give the nodes that are required to be clients and facilities, respectively.

1: $F_c = V \setminus P$, $F_f = \emptyset$, $H = G[P]$ ▷ initialization
2: **if** H is not connected **then return** ***false***
3: Compute the BC-tree $\mathcal{B}$ of H.
4: **if** more than one block node of $\mathcal{B}$ has degree > 1 **then return** ***false***
5: **if** $\mathcal{B}$ contains exactly two blocks **then**
6: Run Lines 10–19 of this code independently twice for both possible choices of B^*.
7: **if** any of the runs returns true **then**
8: Set F_f and F_c to the respective sets computed in that run. **return** ***true***
9: **else return** ***false***
10: Let B^* be a block with highest degree in $\mathcal{B}$. Let C be the cut nodes in B^*.
11: **if** $|E(B^*)| = 1$ **then**
12: **if** $\exists v \in V(B^*)$ with $N(v) = V \setminus \{v\}$ **then**
13: $F_f := \{v\}, F_c := V \setminus \{v\}$. **return** ***true*** ▷ $|P| \geq 3$ induces that v is unique
14: **else return** ***false***
15: $F_f := C$ ▷ fix the cut nodes as facilities
16: $F_c := F_c \cup (P \setminus V(B^*))$ ▷ fix the nodes in other blocks as clients
17: **for each** $v \in F_c$ **do**
18: **if** $N(v) \setminus F_c = \emptyset$ **then return** ***false***
19: **return** ***true*** ▷ there exists a feasible solution for the input instance

vertex. Therefore we cannot concentrate on a single block B^* in $H = G[P]$, but rather have to consider a collection of blocks.

We say a block in $\mathcal{B}$ is *trivial* if it contains only a single edge. We can observe that a feasible 2-edge-interconnected facility network may span over multiple non-trivial blocks, but it cannot contain a trivial block: the removal of this edge would disconnect the network. We call a set of non-trivial blocks a *block cluster*, if the union of these blocks forms a connected graph. A *maximal* block cluster (MBC) is a block cluster to which we cannot add any non-trivial block without losing this connectivity. We observe that all MBCs in a graph are pairwise disjoint. If there is no non-trivial block, we define an arbitrary node of highest degree to form a unique MBC.

In order for an input instance to allow a feasible solution, H has to contain exactly one unique MBC: If there are more than one, they are pairwise seperated by at least one trivial block, i.e., an edge. The final facility network can only reside in one of the MBCs, say B°, but then there is no edge to directly connect a node of some other MBC (as a client) to any node in B°. Overall, we can reuse Algorithm 1 to check feasibility of 2E-iCFL instances by replacing B^* by B° and substituting Lines 4–14 by

1: **if** there are multiple MBCs **then return** ***false***
2: Let B° be the unique MBC in $\mathcal{B}$. Let C be the nodes of B° adjacent to $H \setminus B^\circ$.

Since identifying MBCs in $\mathcal{B}$ requires only linear time, we obtain:

Theorem 4. *We can check whether a given instance allows a feasible solution for 2E-iCFL in linear time.*

3.3 Preprocessing and Heuristic

Assume in the following that we have a unique non-trivial B^* or B°, respectively. The otherwise arising special cases can be handled analogously to below.

Preprocessing. Based on the feasibility test, we can fix or remove certain substructures in our graph to obtain a smaller problem instance that remains to be solved. The feasibility testing algorithms already give sets F_f and F_c containing nodes that are required to become facilities or clients, respectively. In Line 18 of Algorithm 1, a check is performed on the size of $W := N(v) \setminus F_c$, for each $v \in F_c$. In a more in-depth implementation, we can check if W contains only a single element w. If so, we can add w to F_f (if it is not already in that set), and fix the edge (v, w) to be chosen in the solution. Furthermore, if there exists a $w \in (W \cap F_f)$ with $c(v, w) = \min\{c(v, w') \mid w' \in W\}$, we can also fix to choose the edge (v, w) without losing optimality. Finally, we can remove all clients with a fixed incident edge (we already know how to attach them to the final solution), and all edges between two clients (they will never be used).

Greedy Heuristic. Although the testing algorithms give a feasible solution as a witness, this solution will in general be far from optimal. Let $\mathcal{F}'$ and N' be the facilities and the facility network obtained after the testing procedure. We attach each client to a facility neighbor with smallest edge cost. We can now incrementally try to improve this solution via steps of two types, in decreasing order of their cost improvement (if positive): (a) remove an edge e from N'; (b) change a node v of $\mathcal{F}'$ into a client. After each step, we test whether the resulting network is still feasible; if not, we undo the step and continue with the next one. One could further extend this algorithm by simultaneously transforming two adjacent facilities into clients.

4 Exact Algorithms

We can formulate the variants of 2-iCFL as integer linear programs using directed cuts. The core of the formulation is similar to the currently strongest ones known for $\{0, 1, 2\}$-survivable network design [2]. Consequently, our directed cut formulations can be shown to be stronger than undirected cut formulations, and are equally strong as, but practically preferable over, formulations based on multi-commodity flow. Proofs are similar to those in [2].

Orientations. We start with considering r-2-iCFL problems, and recall the structural properties S/1–S/3 in Section 1. Based on the orientation properties of 2-node/edge-connected graphs [2, 13], we can recognize feasible networks (including an optimal one) as follows.

Lemma 1. *A network N with facilities $\mathcal{F}$ is r-2E-iCFL feasible if there exists an orientation $\hat{N}$ of N with the following properties:*
(P1) Each node $v \notin \mathcal{F}$ has in-degree 1 and out-degree 0.
(P2) For each node $v \in \mathcal{F} \setminus \{r\}$, $\hat{N}$ contains directed paths $(r \to v)$ and $(v \to r)$.

Lemma 2. *A network N with facilities $\mathcal{F}$ is r-2N-iCFL feasible if there exists an orientation $\hat{N}$ of N that satisfies property (P1) from above and:*

(P3) For each node $v \in \mathcal{F} \setminus \{r\}$, $\hat{N}$ contains directed paths $(r \to v)$ and $(v \to r)$ that are internally node-disjoint.
(P4) The in-degree of r is 1.

Formulation for r-2-iCFL. Instead of $G = (V, E)$ we will consider its bidirected counterpart $G' = (V, A)$ which contains the two arcs (v, u) and (u, v) for each edge $\{v, u\} \in E$. As discussed in Section 3.3, we can assume that $\{v, u\} \cap P \neq \emptyset$ for all edges $\{v, u\} \in E$. We use binary indicator variables x_a for each $a \in \mathcal{X} = A \cap (P \times P)$, z_a for each $a \in \mathcal{Z} = A \cap (P \times V)$, and y_v for each $v \in P$. We set $x_a = 1$ if a is in the solution's core network, $z_{(v,u)} = 1$ if the client u is attached to the facility v, and $y_v = 1$ if v is chosen as a facility.

Let $S \subset V$, then $\delta^+(S) := \{(s,t) \in A \mid s \in S, t \in V \setminus S\}$ and $\delta^-(S) := \{(s,t) \in A \mid s \in V \setminus S, t \in S\}$ denote the arcs leaving and entering S, respectively. We may use one or more nodes as a subscript to denote that any arcs incident to any of these nodes shall be ignored. Furthermore, we use the shorthand $x(B) := \sum_{e \in B} x_e$ (where x_e exists) for $B \subseteq A$.

We can then write the following ILP for r-2N-iCFL. Every node is either attached to a facility via a z variable, or belongs to the core network; used arcs emanate only from facilities ((3) and (2)). If a node is a facility, there exists a path (uniquely oriented via x variables) from it to the root r and vice versa; these two paths are internally node-disjoint and all orientations are unique ((4), (5), (6), and (7)). Constraint (8) guarantees a single block containing all facilities. By omitting the latter two constraints (7) and (8) we obtain the formulation for r-2E-iCFL. Due to the cost function in (1) we may skip the constraints (a) $x_{vw} \leq y_w$, for $(v, w) \in \mathcal{X}$, (b) $\sum_{v \in P:(v,u) \in \mathcal{Z}} z_{vu} \leq 1 - y_u$, for $u \in P$, and (c) $z_{vu} + z_{uv} \leq 1$, for $(u, v), (v, u) \in \mathcal{Z}$.

$$\min \quad \sum_{i \in P} f_i y_i + \sum_{e \in \mathcal{Z}} c_e z_e + M \cdot \sum_{e \in \mathcal{X}} c_e x_e \tag{1}$$

$$z(\delta^-(\{v\})) + x(\delta^-(\{v\})) \geq 1 \qquad \forall v \in V \tag{2}$$

$$x_{vw} \leq y_v, \quad z_{vu} \leq y_v \qquad \forall (v,w) \in \mathcal{X}, (v,u) \in \mathcal{Z} \tag{3}$$

$$x_{vw} + x_{wv} \leq 1 \qquad \forall (v,w), (w,v) \in \mathcal{X} \tag{4}$$

$$x(\delta^-(S)) \geq y_v \qquad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap P \tag{5}$$

$$x(\delta^+(S)) \geq y_v \qquad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap P \tag{6}$$

$$x(\delta^+_w(S_1)) + x(\delta^-_w(S_2)) \geq y_v \qquad \begin{array}{l} \forall w \in V \setminus \{r\}, \forall S_1, S_2 \subseteq V \setminus \{r, w\}, \\ \forall v \in S_1 \cap S_2 \cap P \end{array} \tag{7}$$

$$x(\delta^-(\{r\})) = 1 \tag{8}$$

$$x_e, z_f, y_u \in \{0, 1\} \qquad \forall e \in \mathcal{X}, f \in \mathcal{Z}, u \in P \tag{9}$$

Formulation for 2-iCFL. Having a formulation for the rooted case, we can extend the formulation for the unrooted problem family. We extend G via an artificial root node $r \in P$ and introduce zero-cost edges between r and all nodes in $P \setminus \{r\}$ together with their x-variables, but without z's. We then restrict the solution space via (10) to networks where r has not only a single in-going edge, but also a unique out-going one. Finally, it remains to guarantee 2-connectivity even after the removal of r. To guarantee feasible 2E-iCFL solutions, it suffices

to require (11); for 2N-iCFL we instead require (12):

$$x(\delta^+(\{r\})) = 1 \tag{10}$$

$$x(\delta_r^+(S)) \geq y_u + y_v - 1 \qquad \begin{array}{l}\forall u \in P \setminus \{r\}, S \subseteq V \setminus \{r,u\},\\ \forall v \in S \cap P\end{array} \tag{11}$$

$$x(\delta_{w,r}^+(S_1)) + x(\delta_{w,r}^-(S_2)) \geq y_u + y_v - 1 \qquad \begin{array}{l}\forall w \in V \setminus \{r\}, \forall u \in P \setminus \{r,w\},\\ \forall S_1, S_2 \subseteq V \setminus \{r,w,u\},\\ \forall v \in S_1 \cap S_2 \cap P\end{array} \tag{12}$$

Formulation for degree-bounded problems. The maximum degree restriction Δ can be added to either of the above formulations straight-forwardly by requiring that, for each node v, the sum over all x and z variables corresponding to arcs incident to v is at most Δ.

$$\sum_{i:\{v,i\}\in E} x_{vi} + x_{iv} \leq \Delta \quad \forall v \in P. \tag{13}$$

Formulations for other cost functions. Using a more general cost function for the core network can be simply accomplished by modifying (1). In the special case $M = 1$, our formulation can be simplified by unifying the x and z variables, such that x_e ($e \in A$) simply indicates if e is included in the solution network. We can remove (2), and rewrite (5) as $x(\delta^-(S)) \geq 1$ for all $S \subseteq V \setminus \{r\}$.

Theorem 5. *The above formulations for all problem variants of 2-iCFL (i.e., any combination of edge- vs. node-connectivity; rooted vs. unrooted; degree-bounded vs. unrestricted) give optimal solutions for the respective problems, if any exist.*

Note that the above formulations, although requiring an exponential number of constraints, can be solved straight-forwardly using Branch-and-Cut techniques: Since all the constraints in question are of the well-known cut type, the separation problem can be solved in polynomial time via max flow (see, e.g., [2]).

5 Experiments

As a proof-of-concept, we implemented a Branch-and-Cut algorithm using the above formulations for r-2-iCFL. The main aim of this study is to evaluate the general practicability of the approach and to estimate the dependance of the ILP's solvability on various instance properties. We used Cplex 9 as a Branch-and-Bound framework and implemented the necessary separation routines in C++ facilitating the efficient max-flow code of [1]. Our code does not use any heuristics for upper bounds, in order to test pure ILP performance. The experiments were conducted on an Intel Xeon 2.33Ghz CPU with 2GB of RAM per process and we set a time limit of 1 hour per problem instance.

Test instances. We tested our exact algorithms on a generated test suit, mimicking real-world data. Using such instances allows us to observe the algorithm's dependency on different input parameters. In the following, n, λ, ϱ, and δ will be these parameters: We distribute n points

Table 1. Overview on the experimental results for $M = 1$: For each parameter setting we give the number of solved instances (out of 5) in bold, and the average running time (in seconds) over these solved instances. The letters E and N denote whether we consider r-2E-iCFL or r-2N-iCFL, respectively.

			$\varrho = 10\%$						$\varrho = 25\%$						$\varrho = 50\%$					
n	λ		$\delta = 30\%$		$\delta = 70\%$		$\delta = 100\%$		$\delta = 30\%$		$\delta = 70\%$		$\delta = 100\%$		$\delta = 30\%$		$\delta = 70\%$		$\delta = 100\%$	
50	0.0	E	**5**	0.0	**5**	0.0	**5**	0.0	**5**	0.1	**5**	0.3	**5**	0.2	**5**	4.9	**5**	2.6	**5**	2.2
	0.0	N	**5**	0.0	**5**	0.0	**5**	0.0	**5**	0.1	**5**	0.3	**5**	0.2	**5**	5.6	**5**	2.3	**5**	2.6
	0.1	E	**5**	0.0	**5**	0.0	**5**	0.1	**5**	0.1	**5**	0.2	**5**	0.3	**5**	2.0	**5**	4.1	**5**	3.2
	0.1	N	**5**	0.0	**5**	0.0	**5**	0.1	**5**	0.1	**5**	0.3	**5**	0.3	**5**	1.6	**5**	4.0	**5**	3.1
	0.3	E	**5**	0.0	**5**	0.0	**5**	0.1	**5**	0.1	**5**	0.3	**5**	0.3	**5**	3.1	**5**	5.6	**5**	11.3
	0.3	N	**5**	0.0	**5**	0.0	**5**	0.1	**5**	0.1	**5**	0.4	**5**	0.4	**5**	3.8	**5**	5.9	**5**	9.9
100	0.0	E	**5**	0.3	**5**	0.6	**5**	1.2	**5**	6.1	**5**	6.2	**5**	7.1	**4**	526.1	**5**	611.9	**5**	123.3
	0.0	N	**5**	0.3	**5**	0.6	**5**	0.9	**5**	9.1	**5**	5.3	**5**	6.0	**4**	362.3	**5**	346.6	**5**	119.6
	0.1	E	**5**	0.3	**5**	0.8	**5**	1.1	**5**	4.6	**5**	13.9	**5**	5.7	**3**	841.1	**4**	411.2	**5**	327.1
	0.1	N	**5**	0.5	**5**	0.8	**5**	1.1	**5**	6.0	**5**	11.3	**5**	5.2	**3**	766.1	**4**	508.2	**5**	313.2
	0.3	E	**5**	0.3	**5**	1.2	**5**	1.7	**5**	4.0	**5**	25.8	**5**	28.9	**2**	1274.1	**2**	1733.8	**3**	1663.4
	0.3	N	**5**	0.4	**5**	1.2	**5**	1.8	**5**	7.0	**5**	23.4	**5**	25.4	**2**	866.2	**2**	1535.6	**3**	1194.6
200	0.0	E	**5**	11.8	**5**	35.3	**5**	52.1	**4**	693.2	**4**	712.8	**5**	845.5	**0**	—	**0**	—	**0**	—
	0.0	N	**5**	9.0	**5**	33.3	**5**	56.3	**4**	674.5	**4**	676.5	**5**	792.6	**0**	—	**0**	—	**0**	—
	0.1	E	**5**	11.5	**5**	41.7	**5**	52.9	**3**	2767.4	**4**	1813.8	**5**	490.5	**0**	—	**0**	—	**0**	—
	0.1	N	**5**	13.6	**5**	52.6	**5**	52.9	**2**	2116.7	**3**	1512.0	**5**	456.5	**0**	—	**0**	—	**0**	—
	0.3	E	**5**	12.2	**5**	42.7	**5**	88.3	**0**	—	**3**	1593.7	**0**	—	**0**	—	**0**	—	**0**	—
	0.3	N	**5**	15.6	**5**	67.5	**5**	101.9	**1**	3574.5	**3**	2109.6	**0**	—	**0**	—	**0**	—	**0**	—
400	0.0	E	**5**	817.9	**3**	2380.5	**5**	2503.5	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—
	0.0	N	**3**	936.9	**4**	2331.3	**5**	1894.0	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—
	0.1	E	**5**	584.5	**4**	1780.8	**1**	2779.2	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—
	0.1	N	**5**	644.4	**4**	2029.7	**2**	2995.8	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—
	0.3	E	**5**	660.4	**3**	2667.4	**1**	1648.6	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—
	0.3	N	**4**	879.1	**1**	2714.6	**1**	2294.1	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—	**0**	—

uniformly at random on a 1000×1000 grid (taking the first chosen node as the root), and compute the Euclidean distances $d_{\{u,v\}}$ for each pair of nodes $\{u, v\}$. Based thereon, we generate (random) edge costs via a Gauss distribution, using $d_{\{u,v\}}$ as the mean and $\lambda \cdot d_{\{u,v\}}$ as the variance. Clearly, $\lambda = 0$ gives Euclidean instances. Using a uniform distribution, we select $\varrho\%$ of the nodes into the set P of potential facilities with uniformly distributed costs from the interval $[250, 750]$[3]. Overall, we generate $\delta\%$ of all possible edges. To guarantee feasible instances, we thereby start with a random cycle through the nodes P; for every node $V \setminus P$ we introduce an edge connecting it to some (randomly chosen) node in P. The remaining edges are generated randomly over all remaining node pairs. We report on the instances generated over all possible combinations of the following parameters: $n \in \{50, 100, 200, 400\}$, $\lambda \in \{0, 0.1, 0.3\}$, $\varrho \in \{10\%, 25\%, 50\%\}$, $\delta = \{30\%, 70\%, 100\%\}$. For each combination we generated 5 instances and considered $M \in \{1, 10, 20\}$.

Evaluation. Due to space restrictions, we only give a brief overview, outlining the applicability of the exact approach from a user's perspective. We postpone more in-depth technical-oriented evaluations (LP gaps, number of branching nodes, etc.) to an extended version of this paper. Table 1 shows the central findings for $M = 1$, giving the number of solved instances and average computation time. Although the ILP formulation for $M = 1$ looks simpler than for general M, we obtain virtually the same running times for $M = 10, 20$. This seems to be

[3] This interval constitutes the sweetspot with respect to the expected edge lengths to avoid degenerate cases where either all or only a single potential facility would be opened.

due to the fact that higher core costs result in smaller and therefore more simply computable facility networks.

The table shows that small graph sizes and few routers are easily solvable; the regions shaded in gray show where the algorithmic behaviour starts to degrade with respect to these two parameters. Surprisingly, while on first sight we may think that the number of graph vertices is a central parameter, we can actually observe that the size of P is much more important: the shaded cells all correspond to instances with 40–50 potential routers. It is potentially fortunate that in most practical settings, this parameter seems to be rather small. Also somewhat surprisingly, the graph's density does not play such a crucial role. While 2E-iCFL and 2N-iCFL are computationally comparable, the Gauss variance is of moderate importance: Generally, Euclidean instances are the simplest.

Conclusions. Our experiments show that even though the problem's complexity may seem daunting at first sight, the exact approaches are promising and can be used for some real-world problems with not too many potential routers, even in their current simple form, i.e., without requiring strong primal heuristics or algorithmic tricks.

References

1. B. V. Cherkassky and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.
2. M. Chimani, M. Kandyba, I. Ljubić, and P. Mutzel. Orientation-based models for $\{0,1,2\}$-survivable network design: Theory and practice. *Mathematical Programming B*, 2009. Accepted. Preliminary versions appeared as two successive parts at ESA'07 and COCOA'08.
3. F. Eisenbrand, F. Grandoni, T. Rothvoß, and G. Schäfer. Approximating connected facility location problems via random facility sampling and core detouring. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1174–1183, 2008.
4. K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
5. S. Kedad-Sidhoum and V. H. Nguyen. An exact algorithm for solving the ring star problem. http://www.optimization-online.org/DB_HTML/2008/03/1942.html, March 2008.
6. M. Labbe, G. Laporte, I. Rodriguez Martin, and J. J. Salazar Gonzalez. The ring star problem: Polyhedral analysis and exact algorithm. *Networks*, 43(3):177–189, 2004.
7. L.C. Lau and M. Singh. Additive approximation for bounded degree survivable network design. In *Proceedings of the 40th ACM Symposium on Theory of Computing*, pages 759–768, 2008.
8. I. Ljubić. A hybrid VNS for connected facility location. In *Hybrid Metaheuristics, Volume 4771 of Springer Lecture Notes in Computer Science*, pages 157–169, 2007.
9. C. L. Monma, B. S. Munson, and W. R. Pulleyblank. Minimum-weight two-connected spanning networks. *Mathematical Programming*, 46(2):153–171, 1990.
10. M. Pioro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.
11. S. Raghavan. *Formulations and Algorithms for the Network Design Problems with Connectivity Requirements*. PhD thesis, MIT, Cambridge, MA, 1995.
12. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and sub-constant error-probability pcp characterization of np. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484, 1997.
13. H.E. Robbins. A theorem on graphs with an application to a problem of traffic control. *American Mathematical Monthly*, 46:281–283, 1939.

14. M. Stoer. *Design of Survivable Networks*. Springer, 1992. Volume 1531 of LNM.
15. C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.
16. A. Tomazic and I. Ljubić. A GRASP algorithm for connected facility location. In *Proceedings of the 2008 Int'l Symposium on Applications and the Internet*, pages 257–260, 2008.