

Call Control with k Rejections^{*}

R. Sai Anand^{**}, Thomas Erlebach, Alexander Hall^{**}, and Stamatis Stefanakos

Computer Engineering and Networks Laboratory (TIK),
ETH Zürich, CH-8092 Zürich, Switzerland
{anand|erlebach|hall|stefanak}@tik.ee.ethz.ch

Abstract. Given a set of connection requests (calls) in a communication network, the call control problem is to accept a subset of the requests and route them along paths in the network such that no edge capacity is violated, with the goal of rejecting as few requests as possible. We investigate the complexity of parameterized versions of this problem, where the number of rejected requests is taken as the parameter. For the variant with pre-determined paths, the problem is fixed-parameter tractable in arbitrary graphs if the link capacities are bounded by a constant, but W[2]-hard if the link capacities are arbitrary. If the paths are not pre-determined, no FPT algorithm can exist even in series-parallel graphs unless $\mathcal{P} = \mathcal{NP}$. Our main results are new FPT algorithms for call control in tree networks with arbitrary edge capacities and in trees of rings with unit edge capacities in the case that the paths are not pre-determined.

1 Introduction

Given a set of connection requests in a communication network, call admission control is the problem of determining which of the requests can be accepted without exceeding the capacity of the network. The goal is to maximize the number of accepted requests or, equivalently, to minimize the number of rejected requests. In [2], Blum et al. argue that, when considering approximation algorithms, it is meaningful to consider the number of rejected requests as optimization criterion, because the number of rejected requests is expected to be small in practice due to overprovisioning of network resources.

In this paper, we consider call admission control from the viewpoint of parameterized complexity [7]. Roughly speaking, the approach of parameterized complexity is to capture a part of the input or output of an \mathcal{NP} -hard problem by a *parameter*, usually denoted by k . Then one tries to find a *fixed-parameter tractable* (FPT) algorithm for the problem, i.e., an algorithm with running-time $O(f(k) \cdot \text{poly}(n))$, where $f(k)$ is an arbitrary function of the parameter and $\text{poly}(n)$ is a polynomial function of the input size n . If such an algorithm exists, this means that there is a good hope of solving large instances of the problem efficiently provided that the value of the parameter k is small. Thus, the so-called *combinatorial explosion* that is typical for \mathcal{NP} -hard problems can be restricted to the parameter.

^{*} Research partially supported by the Swiss National Science Foundation under Contract No. 21-63563.00 (Project AAPCN).

^{**} Supported by the joint Berlin/Zurich graduate program Combinatorics, Geometry, and Computation (CGC), financed by ETH Zurich and the German Science Foundation (DFG).

Motivated by the arguments given above, we consider a parameterized version of the call admission control problem and take the number of rejected requests as the parameter. We investigate different variants of the problem with respect to the network topology, the link capacities, and the existence of pre-determined routes for the requests.

Preliminaries. In the undirected version of the problem, the communication network is modeled as an undirected graph $G = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{N}$. Connection requests are represented by pairs of nodes. Accepting a request (u, v) requires reserving one unit of bandwidth on all edges along a path from u to v . A set of paths is *feasible* if no edge $e \in E$ is contained in more than $c(e)$ paths. For a given set P of paths in a graph with edge capacities, an edge e is called *violated* (by P) if the number of paths in P that contain e is greater than $c(e)$. We say that a path p in P *contains a maximal set of violated edges* if there is no other path q in P such that the set of violated edges in p is a strict subset of the set of violated edges in q .

The basic call admission control problem can now be defined as follows.

CALLCONTROL: *Given a set R of requests in an undirected graph $G = (V, E)$ and a capacity function $c : E \rightarrow \mathbb{N}$, compute a subset $A \subseteq R$ and assign a path to each request in A such that the resulting set of paths is feasible. The goal is to minimize the number $|R \setminus A|$ of rejected paths.*

CALLCONTROL is \mathcal{NP} -hard even in undirected trees with edge capacities 1 or 2 [10] and in trees of rings with unit edge capacities [8]. Note that in the case of unit edge capacities, all accepted requests must be routed along edge-disjoint paths.

We introduce the following parameterized version of the problem:

CALLCONTROL- k : *Given a set R of requests in an undirected graph $G = (V, E)$, a capacity function $c : E \rightarrow \mathbb{N}$, and an integer $k \geq 0$, compute a subset $A \subseteq R$ and assign a path to each request in A such that the resulting set of paths is feasible and at most k requests are rejected, or decide that no such subset exists.*

There are several interesting variations of the call control problem. First, it is sometimes the case that the assignment of paths to requests cannot be determined by the algorithm, but is pre-determined by other constraints (such as routing protocols). In this case, an instance of the problem includes, for every request $r = (u, v)$, an undirected path p_r from u to v . If r is accepted, it must be routed along p_r . We denote this variant of CALLCONTROL by CALLCONTROLPRE and its parameterized version by CALLCONTROLPRE- k .

Sometimes it is meaningful to model the communication network as a *directed* graph $G = (V, E)$. In this case, we assume that a request (u, v) asks for a *directed* path from u to v in G . A special case of directed graphs are the *bidirected* graphs, i.e., graphs that can be obtained from an undirected graph by replacing each undirected edge with two directed edges of opposite directions. Therefore, we use terms like “CALLCONTROL in bidirected trees of rings” to specify a concrete variant of the problem.

We always use $|I|$ to denote the size of a given instance of a call control problem. An algorithm for a parameterized call control problem is an *FPT algorithm* if its running-time is bounded by $O(f(k) \cdot \text{poly}(|I|))$ for some function f .

Specific network topologies that we consider are chains (graphs consisting of a single path), rings (graphs consisting of a single cycle with at least three nodes), trees (connected, acyclic graphs) and trees of rings, as well as their bidirected versions. An

undirected graph is a *tree of rings* if it can be obtained by starting with a ring and then, repeatedly, adding a disjoint ring to the graph and then identifying one node of the new ring with an arbitrary node of the existing graph.

A standard technique in the tool-box of parameterized complexity is the method of *bounded search trees* [7]. This technique tackles a problem by searching for a solution in a search tree whose size (number of nodes) is bounded by a function of the parameter only, for example, 2^k . If the work at each node of the search tree is polynomial in the size of the instance, an FPT algorithm is obtained.

This technique lends itself nicely to the parameterized call control problem: If we try to find a solution that rejects at most k requests, we identify a small set R_{rej} of *rejection candidates*. This set must have the property that, if a solution rejecting at most k requests exists at all, then there exists a solution rejecting at most k requests that rejects at least one request in R_{rej} . Thus we can branch for each request $r \in R_{\text{rej}}$ and check recursively whether the set $R \setminus \{r\}$ admits a solution rejecting at most $k - 1$ requests. If the resulting search tree is explored up to depth k and no solution is found, we know that no solution rejecting at most k requests can exist.

We will apply the technique of bounded search trees in order to obtain FPT algorithms for parameterized call control problems. The interesting part of each FPT algorithm will be how a set R_{rej} of rejection candidates can be identified and how its size can be proved to be bounded by a constant or by a function of k .

Our Results. In Sect. 2, we give FPT results and hardness results of parameterized call control problems that follow easily from existing results: $\text{CALLCONTROLPRE-}k$ is FPT in general graphs provided that the edge capacities are bounded by a constant. If the edge capacities can be arbitrary, $\text{CALLCONTROLPRE-}k$ contains HITTINGSET as a special case and is thus $\text{W}[2]$ -hard. $\text{CALLCONTROL-}k$ is \mathcal{NP} -hard even for $k = 0$ and unit edge capacities, implying that there cannot be an FPT algorithm unless $\mathcal{P} = \mathcal{NP}$. These results apply to undirected and bidirected graphs. Our main results are given in Sections 3 and 4. In Sect. 3, we devise an FPT algorithm for $\text{CALLCONTROL-}k$ in trees with arbitrary edge capacities. In Sect. 4, we present an FPT algorithm for $\text{CALLCONTROL-}k$ in trees of rings with unit edge capacities. Finally, we draw our conclusions in Sect. 5.

Previous Work on Call Control Algorithms. We are not aware of any previous work on parameterized versions of call control problems. Previous work on call admission control has focused on on-line algorithms (that receive the connection requests over time and must accept or reject each request without knowledge about the future) and approximation algorithms. In most of this work, the number of accepted requests has been used as the objective function. A survey of known results on on-line call control algorithms can be found in the book by Borodin and El-Yaniv [4, Chapter 13].

However, call control problems are not only interesting in the on-line scenario. A number of researchers have studied off-line approximation algorithms for the maximum edge-disjoint paths problem (MEDP), i.e., CALLCONTROL with unit edge capacities and with the number of accepted requests as the objective function. The problem is polynomial for chains, rings, and undirected trees. Constant-factor approximation algorithms have been found for bidirected trees [9], trees of rings [8], and a class of graphs including two-dimensional meshes [13]. For general directed graphs with m edges, it is known that no approximation algorithm can achieve ratio $O(m^{\frac{1}{2}-\varepsilon})$ for any $\varepsilon > 0$

unless $\mathcal{P} = \mathcal{NP}$ [11]. Approximation algorithms with ratio $O(\sqrt{m})$ have been found for MEDP [12] and also for its generalization to arbitrary edge capacities, bandwidth requirements and profit values associated with the requests, the unsplittable flow problem [1] (the ratio increases by a logarithmic factor if the largest bandwidth requirement can exceed the smallest edge capacity). We point out that, unlike the unsplittable flow problem, all requests have the same bandwidth requirement in our definition of the call control problem.

On-line algorithms and approximation algorithms for CALLCONTROLPRE with the number of rejected requests as objective function were studied by Blum et al. [2]. They observed that, in the case of unit edge capacities, this problem is a special case of VERTEXCOVER. Concerning on-line algorithms, they gave a 2-competitive algorithm for chains with arbitrary capacities, a $(c + 1)$ -competitive algorithm for arbitrary networks with capacities bounded by c , and an $O(\sqrt{m})$ -competitive algorithm for arbitrary networks with m edges and arbitrary edge capacities. Their algorithms are allowed to preempt (reject) requests that have been accepted earlier. Furthermore, they presented an off-line $O(\log m)$ -approximation algorithm for arbitrary graphs and arbitrary edge capacities.

2 General Networks

First, let us consider the problem CALLCONTROLPRE- k for networks with unit edge capacities, i.e., $c(e) = 1$ for all $e \in E$. A set of accepted paths is feasible if and only if the paths are pairwise edge-disjoint. The *conflict graph* of a given set P of paths is the graph $H = (P, E')$ with a vertex for each path in P and an edge between two vertices if the corresponding paths share an edge. Then, any feasible subset $A \subseteq P$ is an independent set in H , and its complement $P \setminus A$ is a vertex cover in H , i.e., a subset of the vertices such that each edge has at least one endpoint in the subset.

Therefore, checking whether there exists a feasible solution that rejects at most k paths is equivalent to determining whether H contains a vertex cover of size at most k . The vertex cover problem is well known to be FPT [7]; the best known algorithm so far has running time $O(kn + 1.271^k k^2)$ for deciding whether a graph on n nodes has a vertex cover of size at most k [6]. Thus, CALLCONTROLPRE- k is FPT for arbitrary graphs with unit edge capacities.

Now assume that the edge capacities are bounded by a constant c . If a set of paths violates some edge e , then we can obtain a set P_{rej} of rejection candidates by taking an arbitrary set of $c(e) + 1$ paths containing e . Thus we obtain a search tree of depth k and branching degree at most $c + 1$. The size of this tree is $O((c + 1)^k)$. The task to be carried out at each node of the search tree is determining whether there exists a violated edge e and, if so, picking $c(e) + 1$ paths through e to obtain the set P_{rej} . This can easily be done in polynomial time. Hence, there is an algorithm with running-time $O((c + 1)^k \cdot \text{poly}(|I|))$ for CALLCONTROLPRE- k in arbitrary graphs provided that all edge capacities are bounded by the constant c . This discussion leads to the following proposition.

Proposition 1. CALLCONTROLPRE- k is fixed-parameter tractable for arbitrary directed or undirected graphs if the edge capacities are bounded by a constant.

Proposition 1 leaves open the cases where the edge capacities can be arbitrarily large and/or the paths are to be determined by the algorithm. We show that these cases are unlikely to be FPT for arbitrary graphs. Our hardness results apply even to series-parallel graphs, a very restricted subclass of planar graphs with treewidth at most two [3].

Proposition 2. *If the edge capacities can be arbitrarily large, CALLCONTROLPRE- k is $W[2]$ -hard even for series-parallel graphs.*

Hardness for $W[t]$ for some $t \geq 1$ is considered strong evidence that a problem is not FPT [7]. The proof of Proposition 2, which is omitted due to lack of space, shows that HITTINGSET- k is a special case of CALLCONTROLPRE- k . An instance of HITTINGSET- k consists of a family $\mathcal{S} = \{S_1, \dots, S_m\}$ of subsets of a ground set $U = \{1, 2, \dots, n\}$ and a parameter k . The problem is to determine whether there is a subset $T \subseteq U$, $|T| \leq k$, such that T “hits” all sets in \mathcal{S} , i.e., $T \cap S_i \neq \emptyset$ for all $1 \leq i \leq m$. HITTINGSET- k is $W[2]$ -complete [7].

If a problem is FPT, this implies that the problem is polynomial for each fixed value of the parameter k . Therefore, the following proposition shows that CALLCONTROL- k is very unlikely to be FPT even for series-parallel graphs.

Proposition 3. *CALLCONTROL- k is \mathcal{NP} -hard for $k = 0$ in series-parallel graphs with unit edge capacities.*

Proof. For $k = 0$, the problem CALLCONTROL- k with unit edge capacities reduces to checking whether *all* requests can be accepted and routed along edge-disjoint paths. This edge-disjoint paths problem has been proved to be \mathcal{NP} -hard for series-parallel graphs by Nishizeki, Vygen and Zhou [15]. \square

In order to get FPT results not covered by Proposition 1, we must allow arbitrary capacities or arbitrary (not pre-determined) paths. In view of Propositions 2 and 3, however, we have to restrict the class of graphs that we allow as network topologies. Therefore, we consider CALLCONTROL- k in tree networks with arbitrary edge capacities and CALLCONTROL- k in trees of rings with unit edge capacities.

3 Trees with Arbitrary Capacities

In this section, we develop FPT algorithms for trees with arbitrary edge capacities. First, we discuss the algorithm for undirected trees in detail. Then we explain how the result can be adapted to bidirected trees. Note that CALLCONTROL and CALLCONTROLPRE are equivalent in trees, because paths are uniquely determined by their endpoints.

We remark that CALLCONTROL can be solved optimally in polynomial time for chain networks (using techniques of [5]) and for undirected trees with unit edge capacities, but is \mathcal{NP} -hard for trees already if edge capacities in $\{1, 2\}$ are allowed [10].

The Undirected Case. Let an instance of CALLCONTROL- k in trees be given by an undirected tree $T = (V, E)$ with edge capacities $c : E \rightarrow \mathbb{N}$, a set P of paths in the tree, and an integer parameter $k \geq 0$. Consider the tree to be rooted at an arbitrary node. If $k = 0$, the problem reduces to checking whether the set P is feasible, which can be done

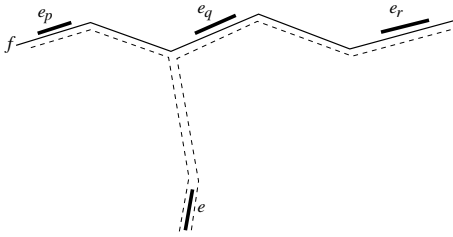


Fig. 1. Proof of Lemma 1.

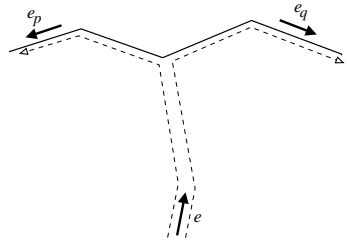


Fig. 2. The bidirected case.

efficiently. So we assume from now on that $k > 0$. If there is no violated edge, answer YES and output P as a solution. Otherwise, let e be a violated edge such that there is no other violated edge in the subtree below e . In what follows, we refer to such an edge as a *bottom-most violated edge*. The algorithm determines a set P_e of paths containing edge e such that $|P_e|$ is small enough (more precisely, $|P_e| \leq 2k$) to be taken as the set of rejection candidates for the bounded search tree technique. Then the algorithm can branch for each $p \in P_e$ and check recursively whether there exists a solution for $P \setminus \{p\}$ that rejects at most $k - 1$ paths. At depth k of the search tree, k paths have been rejected, and we will have either found a feasible solution or no solution with k or less rejections exists.

Now we show how the algorithm determines a set P_e of rejection candidates satisfying $|P_e| \leq 2k$. Since e is a violated edge, at least one of the paths in P that contain e must be rejected. Moreover, it is easy to see that there exists a feasible solution that rejects a path through e which contains a maximal set of violated edges; given any feasible solution, we can construct one of the same cardinality and with the desired property by replacing a non-maximal path with a maximal one. Therefore, the algorithm needs to consider only paths that contain a maximal set of violated edges as rejection candidates.

Since we consider only one rejection at a time, we can restrict the set of rejection candidates even further by considering only one representative from each set of paths that contain the same set of violated edges. So let P_e be a set of paths that contain e satisfying the properties defined above (i.e., each path containing a maximal set of violated edges, and no two paths containing the same set of violated edges).

For each path $p \in P_e$, let e_p be the violated edge on p that is farthest from e , and let $E_e = \{e_p \mid p \in P_e\}$ be the set of all such edges e_p . (If e is the only violated edge on a path $p \in P_e$, we let $e_p = e$. This can happen only if $|P_e| = 1$.) Note that $|E_e| = |P_e|$.

Lemma 1. *No path in P can contain three edges in E_e .*

Proof. Assume to the contrary that there is a path f in P that contains three edges in E_e , say, e_p , e_q , and e_r . Without loss of generality, assume that e_q is between e_p and e_r on f , as shown in Fig. 1. Note that f cannot contain e , because otherwise the path q would not contain a maximal set of violated edges.

All three paths p , q , and r must meet f at the same node, for otherwise we would have a cycle. Assume that they meet f at a node between e_p and e_q . Then e_q is contained in r . Since e_q is the farthest violated edge in q and e is a bottom-most violated edge,

path q does not contain a maximal set of violated edges, a contradiction to the choice of P_e . The cases where the three paths p , q , and r meet f in a different node lead to a contradiction as well. \square

Lemma 2. *If there exists a solution to the given instance of CALLCONTROL- k that rejects at most k paths, then there can be at most $2k$ paths in P_e .*

Proof. For every edge in E_e , any feasible solution must reject at least one path containing that edge. Since each path in P can contain at most two edges in E_e by Lemma 1, a feasible solution must reject at least $|E_e|/2 = |P_e|/2$ paths. Therefore, for a feasible solution with at most k rejections to exist, there can be at most $2k$ paths in P_e . \square

The depth of the search tree is at most k . In a node of the search tree where i paths are already rejected, the algorithm needs to consider at most $2(k - i)$ branches; by Lemma 2, if $|P_e| > 2(k - i)$, there cannot be a feasible solution in the subtree below the current node. Thus the size of the search tree is bounded by $O(2^k k!)$. Finding a bottom-most violated edge e and determining the set of paths P_e can obviously be done in polynomial time. Thus, the algorithm solves the problem CALLCONTROL- k in time $O(2^k \cdot k! \cdot \text{poly}(|I|))$, and we obtain the following theorem.

Theorem 1. *There is an FPT algorithm for CALLCONTROL- k in undirected tree networks with arbitrary edge capacities.*

The Bidirected Case. The FPT algorithm for call control in undirected trees can easily be adapted to bidirected trees, yielding even a better running-time. The algorithm proceeds as in the undirected case by picking a bottom-most violated edge e (which is now a directed edge) and determining a set P_e of rejection candidates that contain e and a maximal set of violated edges. The set E_e is defined as before. Since the paths and edges are now directed, it is easy to see that no path in P can contain two edges in E_e (see Fig. 2). Therefore, if after rejecting i paths, there are more than $k - i$ paths in P_e , there does not exist a feasible solution in the subtree of the current node of the search tree. Thus the size of the search tree can now be bounded by $O(k!)$ and the total running time is $O(k! \cdot \text{poly}(|I|))$.

Theorem 2. *There is an FPT algorithm for CALLCONTROL- k in bidirected tree networks with arbitrary edge capacities.*

4 Trees of Rings with Unit Capacities

In this section, we present the first FPT results for an \mathcal{NP} -hard call control problem where the paths for the requests must be determined by the algorithm. We restrict the network topology to be a tree of rings, and we require that all edges have capacity 1. We obtain FPT algorithms for undirected trees of rings and bidirected trees of rings.

The Undirected Case. Let an instance of CALLCONTROL- k in trees of rings with unit edge capacities be given by an undirected tree of rings $T = (V, E)$, a set S of connection requests, and an integer parameter $k \geq 0$. The algorithm must determine if there exists

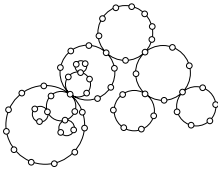


Fig. 3. A tree of rings.

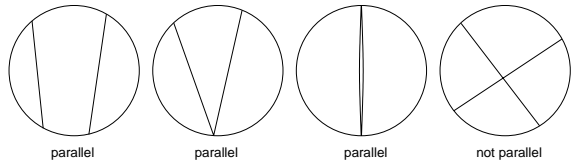


Fig. 4. Chords that are parallel or not parallel.

a subset $S' \subseteq S$ such that $|S \setminus S'| \leq k$ and the requests in S' can be routed along edge-disjoint paths in T . Again, we employ the technique of bounded search trees.

First, let us mention some simple facts about paths in trees of rings (see Fig. 3 for an example of a tree of rings). For any request (u, v) , all undirected paths from u to v contain edges of the same rings. For each ring that a path from u to v passes through (i.e., contains an edge of that ring), the node at which the path enters the ring (or begins) and the node at which the path leaves the ring (or terminates) is uniquely determined by the endpoints of the path. Thus, a request (u, v) in a tree of rings can be viewed as a combination of *subrequests* in all rings that a path from u to v passes through. So, a set of requests can be routed along edge-disjoint paths if and only if all subrequests of the requests can be routed along edge-disjoint paths in the individual rings of the tree of rings. Hence, before we tackle the problem in trees of rings, we need to investigate conditions for a set of requests in a ring being routable along edge-disjoint paths.

Let R be a ring. Imagine R drawn as a circle in the plane, with its nodes distributed at equal distance along the circle. A (sub)request (u, v) between two nodes in R can be represented as a straight line segment joining u and v . We call these line segments *chords* and use the terms *chord* and *request* interchangeably if no confusion can arise. Two chords are said to be *parallel* if (1) they do not intersect, or (2) they intersect at a node in R , or (3) they are identical (see Fig. 4). If two chords are parallel then we can assign edge-disjoint paths to the corresponding requests and these paths are uniquely determined.

A *cut* in a ring R is a pair of edges in the ring. A request *crosses* a cut if each of the two possible paths connecting the endpoints of the request contains exactly one of the edges in the cut.

Lemma 3. *Given a ring R and a set S of requests in R , the requests in S can be routed along edge-disjoint paths if and only if (i) the chords of the requests in S are pairwise parallel and (ii) no cut is crossed by three requests.*

Proof (sketched). If $|S| = 1$, the lemma is trivially true. Assume $|S| \geq 2$. The “only if” part is obvious. To prove the “if” part, assume that S satisfies (i) and (ii). Consider any two requests (u, v) and (w, x) in S . By (i), they are parallel and can thus be assigned edge-disjoint paths p_1 and p_2 in a unique way. If there is a third request (y, z) in S , one can show that y and z must lie in the same chain of $R \setminus (p_1 \cup p_2)$ and so there is a path p_3 from y to z that is disjoint from p_1 and p_2 . This argument can be repeated for all remaining requests, giving a construction of disjoint paths for all requests in S . \square

With the result of Lemma 3 at our disposal, we are ready to obtain the FPT algorithm for CALLCONTROL- k in trees of rings with unit edge capacities. If $k = 0$, the algorithm checks if the conditions of Lemma 3 hold for each ring of the tree of rings. If this is the case, an edge-disjoint routing can be computed efficiently (the proof of Lemma 3 is constructive) and the algorithm answers YES. Otherwise, there is no edge-disjoint routing and the algorithm answers NO.

Now assume that $k > 0$. If the condition of Lemma 3 holds for the subrequests in each ring of the tree of rings, the answer is YES and an edge-disjoint routing for all requests is obtained. Otherwise, there are either two subrequests in a ring that are not parallel, so that at least one of the two must be rejected, or there are three subrequests crossing a cut in some ring, so that at least one of the three must be rejected. In the former case, we get a set of two rejection candidates, in the latter case, we have three rejection candidates. For each request r in the set of rejection candidates, we check recursively whether there exists a solution rejecting at most $k - 1$ requests from $S \setminus \{r\}$. The degree of any node in the search tree is at most 3. Since the depth of the search tree is bounded by k , the size of the search tree is $O(3^k)$. As the conditions of Lemma 3 can be checked easily in polynomial time at each node of the search tree, we obtain an FPT algorithm with running-time $O(3^k \cdot \text{poly}(|I|))$.

Theorem 3. *There is an FPT algorithm for CALLCONTROL- k in undirected trees of rings with unit edge capacities.*

The above discussion shows also that CALLCONTROL- k in undirected trees of rings with unit edge capacities can be seen as an instance of the problem HITTINGSET- k in which each set has cardinality at most 3, i.e., of the 3-HITTINGSET- k problem: the ground set U consists of the requests S , and the family \mathcal{S} of subsets of U consists of all sets of two requests whose subrequests in some ring are not parallel and all sets of three requests whose subrequests cross a cut in some ring. For 3-HITTINGSET- k , an FPT algorithm with running-time $O(2.311^k + n)$, where n is the size of the input, is given in [14]. Thus, by transforming a given instance of CALLCONTROL- k into an instance of 3-HITTINGSET- k , we obtain an FPT algorithm for CALLCONTROL- k in undirected trees of rings with unit edge capacities that runs in time $O(2.311^k + \text{poly}(|I|))$.

The Bidirected Case. We turn to bidirected trees of rings. Each accepted request (u, v) must now be assigned a *directed* path from u to v . As in the undirected case, a set of requests is feasible if and only if all subrequests are feasible in the individual rings. Consider a set S of (sub)requests in a ring. We proceed by doing a case analysis. In each case, we either find that all requests can be routed along edge-disjoint paths, or we are able to identify (in polynomial time) a set S_{rej} of requests such that at least one request in S_{rej} must be rejected in any feasible solution. The cardinality of S_{rej} is at most 5. The details of the case analysis are omitted due to lack of space. We can perform this case analysis for the subrequests of all requests in each individual ring of the tree of rings. Thus, we can apply the bounded search tree technique and get an FPT algorithm for CALLCONTROL- k in bidirected trees of rings with unit edge capacities. The size of the search tree can be bounded by $O(5^k)$, so the running-time is $O(5^k \cdot \text{poly}(|I|))$.

Theorem 4. *There is an FPT algorithm for CALLCONTROL- k in bidirected trees of rings with unit edge capacities.*

5 Conclusion

We have considered parameterized versions of call admission control problems. Since the number of rejected requests can often be expected to be small in practice, we have taken this number as the parameter. For arbitrary networks, the problem was shown to be fixed-parameter tractable if the paths are pre-determined and the edge capacities are bounded by a constant. If either of these restrictions is removed, it was shown that even for series-parallel graphs, the existence of FPT algorithms is unlikely. Hence, we studied trees and trees of rings as network topologies. We gave FPT algorithms for trees with arbitrary capacities and for trees of rings with unit edge capacities if the paths are not pre-determined. Both algorithms can be adapted to the bidirected case.

References

1. Y. Azar and O. Regev. Strongly polynomial algorithms for the unsplittable flow problem. In *Proceedings of the 8th Integer Programming and Combinatorial Optimization Conference (IPCO)*, LNCS 2081, pages 15–29, 2001.
2. A. Blum, A. Kalai, and J. Kleinberg. Admission control to minimize rejections. In *Proceedings of the 7th International Workshop on Algorithms and Data Structures (WADS 2001)*, LNCS 2125, pages 155–164, 2001.
3. H. L. Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.
4. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
5. M. C. Carlisle and E. L. Lloyd. On the k -coloring of intervals. *Discrete Applied Mathematics*, 59:225–235, 1995.
6. J. Chen, I. A. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. In *Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'99)*, LNCS 1665, pages 313–324, 1999.
7. R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, New York, 1997.
8. T. Erlebach. Approximation algorithms and complexity results for path problems in trees of rings. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS 2001)*, LNCS 2136, pages 351–362, 2001.
9. T. Erlebach and K. Jansen. The maximum edge-disjoint paths problem in bidirected trees. *SIAM Journal on Discrete Mathematics*, 14(3):326–355, 2001.
10. N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.
11. V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 19–28, 1999.
12. J. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.
13. J. Kleinberg and É. Tardos. Disjoint paths in densely embedded graphs. In *Proc. of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, pages 52–61, 1995.
14. R. Niedermeier and P. Rossmanith. An efficient fixed parameter algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 2(1), 2001.
15. T. Nishizeki, J. Vygen, and X. Zhou. The edge-disjoint paths problem is NP-complete for series-parallel graphs. *Discrete Applied Mathematics*, 115:177–186, 2001.