Discrete Applied Mathematics 159 (2011) 695–705

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

An s–t connection problem with adaptability

David Adjiashvili *, Rico Zenklusen [1]

Institute for Operations Research, ETH Zurich, Switzerland

ARTICLE INFO

Article history:
Received 28 February 2010
Received in revised form 16 November 2010
Accepted 22 December 2010
Available online 20 January 2011

Keywords:
Robust optimization
s–t connectivity
Shortest path
Two-stage optimization
Adaptability

ABSTRACT

We study a new two-stage version of an s–t path problem, which we call *adaptable robust connection path (ARCP)*. Given an undirected graph $G = (V, E)$, two vertices $s, t \in V$ and two integers $f, r \in \mathbb{Z}_+$, ARCP asks to find a set $S \subset E$ of minimum cardinality which connects s and t, such that for any 'failure set' $F \subset E$ with $|F| \leq f$, the set of edges $S \setminus F$ can be completed to a set which connects s and t by adding at most r edges from $E \setminus F$. We show the problem is NP-hard, and there is no polynomial-time α-approximation algorithm for the problem for $\alpha < 2$ unless P = NP. For $f = r = 1$ we provide an exact polynomial algorithm, and for $f = 1$ and arbitrary r we provide a polynomial 2-approximation algorithm. A characterization of the feasible set is provided for $f = 1$ and several links are established between ARCP and other combinatorial optimization problems, including a new combinatorial optimization problem, the *minimum reduced cost cycle problem* whose complexity is open.

1. Introduction

Optimization under data uncertainty is a topic that was extensively studied in the last decade. Two main approaches were thoroughly investigated, *stochastic optimization* and *robust optimization*. In stochastic optimization the data is given as a probability distribution over some set of inputs. The optimization problem is to find a solution that minimizes the expected cost of the solution. In robust optimization one tries to find a solution that minimizes the cost of the worst case realization of the input data. Both approaches suffer from drawbacks. In stochastic optimization a distribution over the different scenarios is needed. This kind of stochastic information is often not readily available. Furthermore, optimization of the expected cost is only relevant when an optimization problem is solved many times. The robust optimization approach tends to produce overly conservative solutions, ignoring the fact that in reality a small adaptation of the solution is often possible after the scenario is revealed.

In this paper we adopt a *two-stage* approach, which received significant attention in recent years. In the two-stage approach a solution of the optimization problem is composed of a *first stage solution*, which needs to be computed before a particular scenario is revealed, and a *second stage solution*, which can be computed after the scenario is revealed. Both solutions together should comprise a feasible solution for the revealed scenario. Second stage solutions are often referred to as *recovery actions* or *adaptations* of the first stage solution. The first stage solution and the second stage solution are subject to *first stage costs* and *second stage costs* respectively. Normally, first stage costs are not scenario dependent, whereas second stage costs are (often scenarios represent different cost functions on the resources). An alternative approach to two-stage optimization is to optimize only the first stage costs while limiting allowed recovery action in the second stage using a *recovery budget*. In this setting the goal is to find the smallest possible first stage solution which can be adapted in the second stage to a feasible solution *while respecting the recovery budget*. We denote this class of problems by *two-stage feasibility problems*. In this paper we study the s–t path problem in the two-stage feasibility problem setting.

* Corresponding author. Tel.: +41 774016276; fax: +41 446321025.
E-mail addresses: david.adjiashvili@ifor.math.ethz.ch, d.adjiashvili@gmail.com (D. Adjiashvili), ricoz@math.mit.edu (R. Zenklusen).
[1] Current address: MIT, Cambridge, USA.

0166-218X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2010.12.018

The classical shortest path problem is defined in the following way. Given a graph $G = (V, E)$ and two distinct vertices $s, t \in V$, find a shortest path from s to t in G. The two-stage feasibility counterpart of the shortest path problem which we denote by *adaptable robust connection problem (ARCP)* is defined as follows. Given a graph $G = (V, E)$, two vertices $s, t \in V$ and two integers $r, f \in \mathbb{Z}_+$, find a minimum subset $S \subset E$ of edges that connects s and t and satisfies the following requirement. For every subset $F \subset E$ of at most f edges that are removed from the graph, there exists a subset $R \subset E \setminus F$ of at most r *adaptation edges* such that s and t are connected in $(S \setminus F) \cup R$.

The ARCP is motivated by settings in which a robust solution is needed, due to an underlying networked system whose edges are not perfectly reliably, but where a limited recourse action after the failure event is possible. As a possible application, one can consider the problem of connecting two nodes s and t in a communication network, where lines are prone to failure. Assume that in a first stage, s and t can be connected by buying a set of connection lines of a given underlying network. In case of line failures, additional lines can be rented to bypass the broken lines temporarily. There is a setup time incurred by every new line that has to be included in the system, limiting the number of lines that can be used during the recourse action to some given number r. A natural question is to ask about the minimum number of lines that have to be bought in the first stage to guarantee connectedness of s and t for any failure scenario of at most up to f lines. This corresponds exactly to the ARCP. One can as well think of a setting where a possible outage of underling lines is due to a malicious attack that can remove up to f lines. In this setting it is very natural to consider a worst case analysis, i.e., that any removal of up to f edges can be recovered.

We call solutions to the ARCP, *adaptable robust solutions*. Note that in the above setting, allowing no recourse action would require the solution to contain $f + 1$ disjoint paths between the two designated nodes, while the adaptable robust solution may contain significantly fewer links. An important property of our problem is that the second stage decision (the recovery action) corresponds to a computationally tractable optimization problem, namely a Shortest Path computation (see details in the following discussion about deciding feasibility of a solution).

The following is a formal statement of the problem.

Adaptable robust connection problem (ARCP):

Input: A graph $G = (V, E)$, two vertices $s, t \in V$ and $f, r \in \mathbb{Z}_+$.
Problem: Find $S \subset E$ of minimum cardinality which connects s and t, such that for every $F \subset E$ with $|F| \leq f$ there exists $R \subset E \setminus F$ with $|R| \leq r$ such that s and t are connected in $(S \setminus F) \cup R$.

We stress that the optimal solution to ARCP need not contain a shortest s–t path in the graph. Note that in the ARCP, the scenarios are given implicitly, where a scenario corresponds to a set $F \subset E$ with $|F| \leq f$.

We will consider more restricted variants of the problem in which either one or both of the parameters f and r are fixed. In particular we will focus on the case $f = 1$ and study the problem for fixed and variable r.

Note that the cases $f = 0$ and $r = 0$ are solvable in polynomial time. If $f = 0$ the problem is the classical shortest path problem from s to t. When $r = 0$ any solution must contains $f + 1$ edge-disjoint paths from s to t in any feasible solution. Finding such a set with a minimum number of edges can easily be reduced to a network flow problem and solved in polynomial time (see e.g. [18]).

Consider next the problem of deciding feasibility of a solution S to an ARCP instance. In the case of variable f and r we prove in Section 4 that the problem is NP-hard. In the case of fixed f the problem can be solved in polynomial time by enumerating all possible failure scenarios $F \subset E$ and checking the value of the shortest s–t path in the graph $G = (V, E \setminus F)$ with a length function l which assigns the value 0 to edges in $S \setminus F$ and the value 1 to all other edges in $E \setminus F$. S is a feasible solution if and only if the value of the shortest path is at most r for every F. The complexity of the case where r is fixed and f is variable is not known to the authors.

Notice that if the given graph $G = (V, E)$ does not contain $f + 1$ edge-disjoint paths between s and t, then the ARCP does not admit a solution (for any r), since in this case s and t can be disconnected in G by removing a minimum cardinality s–t cut. Hence, throughout this paper, we assume that the minimum cardinality s–t cut in G contains at least $f + 1$ edges when dealing with an ARCP where f edges can be removed.

Main contribution. This paper presents a new version of the s–t connectivity problem in the two-stage robust optimization framework. We establish a graph-theoretic characterization of the feasible set in the case $f = 1$ and we provide an exact algorithm for the case $f = r = 1$. In the latter we show that the optimal solutions to the problem correspond to two-edge-connected subgraphs containing s and t with a minimum number of vertices. In the case $f = 1$ and variable r we show that the problem admits a polynomial 2-approximation algorithm. For this approximation algorithm we define a new combinatorial optimization problem, namely the problem of finding the *minimum reduced-cost cycle* containing two specified vertices in the graph. (The reduced cost of a cycle is the length of the shorter path between the two vertices on the cycle plus the length of the longer path reduced by at most a given number r of edges.) The complexity of this problem is open for the unweighted case to the extent of the authors' knowledge. For the case of variable f and variable r we prove that it is NP-hard to approximate the problem within any factor $\alpha < 2$ by a transformation from a shortest path interdiction problem.

The rest of the paper is organized as follows. Section 2 provides related work. The analysis of the case of a single edge failure ($f = 1$) is provided in Section 3. In Section 4 we prove a hardness result for the ARCP. Section 5 summarizes and lists challenges and open problems.

2. Related work

Significant progress has been made in the field of robust optimization in recent years. Complexity results and algorithms for robust versions of many classical discrete optimization problems are provided in [14]. Bertsimas and Sim [5] proposed a framework for discrete robust optimization. They show that robust counterparts of mixed integer programs with uncertainty in the matrix coefficients and cost functions can be solved using solutions to moderately larger mixed integer programs with no uncertainty.

A rather general notion of robustness in a two-stage discrete optimization setting, called *recoverable robustness*, was proposed by Liebchen et al. [15] as a tractable framework to introduce robustness in the realm of railway optimization. This work proposed a general recipe for designing two-stage robust problems from any combinatorial optimization problem. In [3] Ben-Tal et al. proposed a method to 'delay' some decisions in an uncertain convex optimization problem to the stage at which the real data is revealed. Their approach does not force the decision variables X to be fixed at the first stage decision. Instead, some variables are allowed to be associated in the first stage decision with affine functions $X(\zeta)$ of the uncertain input vector ζ. The resulting formulation turns out to be theoretically and practically tractable in many cases. Bertsimas and Caramanis [4] provided a framework that can deal with discrete changes in the second stage decision in the case of linear optimization.

Dhamdhere et al. [8] introduced an approach to two-stage robustness called *demand-robust optimization*. In this setting a scenario corresponds to a subset of the initially given constraints that need to be satisfied. In the setting of Shortest Path this means that the source node and the target node are not fixed but will rather be revealed in the second stage. In [8] the authors provide approximation algorithms for some problems such as Steiner Tree, Multi-Cut, facility Location etc. in the demand-robust setting. Golovin et al. [10] later improved some of those results. In particular a constant factor approximation algorithm was provided for the robust shortest path problem. Feige et al. [9] and Khandekar et al. [12] studied combinatorial optimization problems in the demand-robust setting with exponentially many scenarios.

The s–t connectivity problem was studied in numerous robust settings ([19,20,8,10,1,17,6]). Yu and Yang [19] studied the shortest path problem with a discrete scenario set in the minimax and regret models and showed that the problem is NP-hard even for 2 scenarios and layered graphs. Aissi et al. [1] later extended those results by showing that both version of the problem admit a FPTAS if the number of scenarios is bounded, and that they are not approximable within $2 - \epsilon$ for any $\epsilon > 0$ with an unbounded number of scenarios unless $P = \text{NP}$. Dhamdhere et al. [8] showed that the shortest path problem in the demand robust setting admits a constant factor approximation. Golovin et al. [10] later improved the approximation constant. Puhl [17] studied the robust shortest path problem in the two-stage optimization setting. Hardness results were proved for discrete scenario sets, interval scenarios and Γ-scenarios in two settings. In one setting the second stage solution can only differ from the first stage solution by at most k edges. In the second setting two parameters are introduced, $\alpha \in (0, 1)$ and $\beta > 0$. An edge e in the first stage solution costs $(1 - \alpha)c^S(e)$ for 'renting' this edge. If this edge is used in the second stage solution, an 'implementation' cost $\alpha c^S(e)$ is added to the total cost. An edge e that is used in the second stage solution which was not rented in the first stage costs $(\alpha + \beta)c^S(e)$. In the second setting an approximation algorithm was provided. Büsing [6] later addressed the problem of finding the smallest number of edges in the graph which contains a shortest path according to every scenario. An approximation algorithm and inapproximability results were provided.

In the field of survivable network design and communication networks the problem of establishing a reliable link between two nodes in the network is a fundamental and well-studied problem. There are two main approaches to survivability often referred to as *protection* and *restoration*. The protection paradigm tries to immunize the network from failures by allowing redundant solutions. The restoration paradigm focuses on ways of recovering from failures by recalculating large parts of the solution. The former approach allows for fast recovery and is more suitable in applications where long disruptions cannot be tolerated, and often leads to costly solutions. The latter approach may not be realistic in some real world applications, in which long disruptions in communication cannot be tolerated. See [11,16] for surveys on these topics. The main difference between previous work in these fields and our problem is the way in which the network is protected from failures. While the approach in most previous work is either to obtain a solution which remains feasible after a failure (and hence does not exploit the possibility of recovery), or provide precomputed alternatives for replacing failed links. Our approach attempts to optimally exploit a given capacity for recourse in case of failure. Furthermore, the optimal recourse action is given by solving a canonical Shortest Path problem. Another fundamental difference is that our approach allows for modeling the possibility of exponentially many failure scenarios (with respect to the size of the graph), which are given explicitly by the parameter f. While the robust approach will provide a solution also for large values of f, the quality of the solution may be significantly worse than the adaptable robust solution. Approaches which use precomputed alternatives to protect against different failure scenarios are not applicable when f is large.

3. Single edge failure

3.1. The feasible set

We establish some notation to simplify the discussion to follow. Let $G = (V, E)$ be an undirected graph. For some $A \subset E$ let $V(A) \subset V$ denote the vertices touched by A. The subgraph induced by a subset of vertices $U \subset V$ is the graph containing

the vertices in U and all edges in E that connect two vertices in U, and is denoted by $G[U]$. Furthermore, a path is always node-disjoint, i.e., it does not contain cycles.

We start by characterizing the feasible set in the case $f = 1$. The characterization we provide will lead to a polynomial algorithm in the case $r = 1$ and to a polynomial 2-approximation algorithm when $r \in \mathbb{N}$ is part of the input.

We start by formally defining the feasible set in the general case. In all following discussions we fix a graph $G = (V, E)$ and two vertices $s, t \in V$. For $k \in \mathbb{N}$, let $[k] := \{1, \ldots, k\}$.

Definition 1. A set of edges $S \subset E$ is said to be *connecting* if it contains a path from s to t. A connecting set S is said to be (f, r)-*recoverable* if for every $F \subset E$ with $|F| \leq f$ there exists $R \subset E \setminus F$ with $|R| \leq r$ such that s and t are connected in $(S \setminus F) \cup R$.

Hence, the feasible set in the case $f = 1$ is the family of all $(1, r)$-recoverable subsets of E, and hence in this case ARCP asks to find such a set of minimal cardinality. To establish the aforementioned characterization of the feasible set we define the notion of r-*cyclic* sets.

Definition 2. A set $S \subset E$ of edges is said to be r-*cyclic* if for every $e \in S$ there exists a cycle C (represented as a set of edges) in G containing e and satisfying

$$|C \setminus S| \leq r.$$

We say that a set S is *minimal* with respect to any property if the property holds for S, and for every nonempty subset A of S the property does not hold for $S \setminus A$. Note that if a set S is (f, r)-recoverable, then it is minimal if and only if $S \setminus \{e\}$ is not (f, r)-recoverable, for every $e \in S$. This is due to the monotonicity property of (f, r)-recoverable sets which states that any super-set of a (f, r)-recoverable set is a (f, r)-recoverable set. In contrast, r-cyclic sets do not satisfy the monotonicity property. Furthermore, the notion of minimality is not interesting in case of r-cyclic sets, since the only minimal r-cyclic set is the empty set. Therefore, in the following we will only consider minimal r-cyclic sets which contain an s–t path, namely r-cyclic sets which are connecting. We proceed by proving the following simple lemma.

Lemma 3. *Let S be r-cyclic and connecting. Then S is $(1, r)$-recoverable.*

Proof. S is connecting by definition. It remains to show that for every failure set $F = \{e\}$ there exists a recovery set R of at most r edges different from e. Since F is r-cyclic there exists a cycle C in G containing e which satisfies $|C \setminus S| \leq r$. We choose $R = C \setminus S$. Clearly s and t are connected in $(S \setminus F) \cup R$. If e was not on an s–t path we are done. Otherwise, adding R to S closes a cycle with e and thus there is a detour for e in $(S \setminus F) \cup R$, hence s and t are connected. □

We can now provide an exact characterization of minimal $(1, r)$-recoverable sets.

Theorem 4. *A set $S \subset E$ is minimal $(1, r)$-recoverable if and only if it is a minimal set with the properties to be r-cyclic and connecting.*

Proof. Assume first that S is minimal $(1, r)$-recoverable. Then S is connecting by definition. Next we note that S is acyclic. Otherwise one could remove any edge from any cycle in S and the result would be a smaller $(1, r)$-recoverable set, contradicting the minimality of S. Consequently, there is a unique path, denoted by p, from s to t in S. To prove that S is r-cyclic we choose some $e \in S$. Consider first the case that e is in p. Let R be a recovery set for $F = \{e\}$ and let $\hat{p}$ be some s–t path in $(S \setminus \{e\}) \cup R$. It is clear that $|\hat{p} \setminus S| \leq r$, hence setting $C = p \cup \hat{p}$ provides a cycle containing e with $|C \setminus S| \leq r$. Consider next the case that e is not in p. In this case there exists an edge $\bar{e} \in p$, such that every recovery set R for $F = \{\bar{e}\}$ contains e. Such $\bar{e}$ must exist, otherwise $S \setminus \{e\}$ would be $(1, r)$-recoverable, contradicting the minimality of S. Now let R be any recovery set for $F = \{\bar{e}\}$. Let $\hat{p}$ be some s–t path in $(S \setminus F) \cup R$. Again we are guaranteed that $|\hat{p} \setminus S| \leq r$, hence setting $C = p \cup \hat{p}$ provides a cycle containing e with $|C \setminus S| \leq r$. To conclude this direction of the proof we show that S is minimal. Assume toward contradiction that there exists a nonempty set A such that $S \setminus A$ is r-cyclic and connecting. Hence, Lemma 3 suggests that $S \setminus A$ is a smaller $(1, r)$-recoverable set, contradicting the minimality of S.

For the other direction we assume that S is minimal set with the properties of being r-cyclic and connecting. By Lemma 3 we know that S is $(1, r)$-recoverable. It remains to show that S is minimal. Assume S is not minimal. Let $\hat{S} \subset S$ be a minimal $(1, r)$-recoverable subset of S. By the first direction of the proof we know that $\hat{S}$ is r-cyclic and hence by Lemma 3 also $(1, r)$-recoverable, contradicting the minimality of S. □

Corollary 5. *$S \subset E$ is minimum $(1, r)$-recoverable if and only if S is an r-cyclic and connecting set of minimum cardinality.*

3.2. An exact efficient algorithm for $r = 1$

In this section we provide and exact polynomial algorithm for the case $f = r = 1$. Corollary 5 suggest that the minimum $(1, 1)$-recoverable subsets of E are exactly the minimum 1-cyclic subsets of E containing an s–t path. We now establish a connection between minimal connecting 1-cyclic sets and 2-edge connected subgraphs of G.

There has been significant work on finding 2-edge connected graphs in the field of survivable network design. We refer the reader to [13] for a survey. Despite some similarities between the $(1, 1)$-ARCP and the Minimum 2-Edge Connected Spanning Subgraph Problem (e.g. every 2-edge connected spanning subgraph of G is a feasible solution to $(1, 1)$-ARCP), the problems are inherently different. In particular, $(1, 1)$-ARCP is only concerned with connecting two specified vertices, rather than all pairs of vertices. A more related problem of finding the shortest cycle containing two designated vertices is a well-understood problem. The similarities of $(1, 1)$-ARCP to the latter problem are explained and exploited in the remaining of this section.

We start by stating and proving a simple lemma on the connectivity of the minimal connecting 1-cyclic sets.

Lemma 6. *Let S be a minimal connecting 1-cyclic set. Then S is connected, i.e., its edges form one connected component.*

Proof. Assume by contradiction that S is composed of at least 2 connected components $A, B \subset S$. Either A or B contain no edge from the s–t path in S. Assume that it is A. We set $\hat{S} = S \setminus A$. By the choice of A, $\hat{S}$ is connecting. We claim that $\hat{S}$ is 1-cyclic. Let $e \in \hat{S}$ and let C be a cycle in S such that $|C \setminus S| = 1$. It is clear that C contains no edges from A. If C did contain edges from A it would imply that A is connected with 2 edges to the connected component of e after introducing a single edge into S. We conclude that $C \subset \hat{S}$, hence $\hat{S}$ is a smaller connecting 1-cyclic set, contradicting the minimality of S. □

We established in the proof of Theorem 4 that minimal connecting r-cyclic sets are acyclic. Combining this insight with Lemma 6 provides the observation that every minimal 1-cyclic set is a spanning tree of some induced subgraph H of G. In the next lemma we prove that H is 2-edge connected.

Lemma 7. *Let S be connecting 1-cyclic and let $V(S) \subset V$ denote the set of vertices adjacent to at least one edge in S. Then the induced subgraph $G[V(S)]$ is 2-edge connected, and S is a spanning tree in $G[V(S)]$.*

Proof. Let $H = G[V(S)]$. The fact that S is a spanning tree in H follows by the discussion preceding the lemma. Assume toward contradiction that H contains a bridge e. e must be contained in S, or S would be disconnected contradicting Lemma 6. Since S is 1-cyclic, there is a cycle C in G containing e and satisfying $|C \setminus S| = 1$. Since only one extra edge is required to close a cycle in S with e we conclude that both endpoints of this edge lie in $V(S)$, which means that they lie in H, and hence the edge lies in H as well. Hence $C \subset H$, and therefore no edge of C can be a bridge in H, contradicting the assumption that $e \in C$ is a bridge in H. □

To complete the discussion we provide the following simple observation.

Lemma 8. *Any spanning tree of any 2-edge connected induced subgraph of G containing s and t is a connecting 1-cyclic set.*

Proof. Let $S \subset E$ be any spanning tree of a 2-edge connected induced subgraph H of G containing s and t. Clearly S is connecting. Let $e \in S$ and let A and B be a partition of $V(S)$ according to the side of e on which they lie in the spanning tree S. Clearly there is some edge between some $u \in A$ and some $v \in B$ in H, otherwise e would be a bridge, contradicting 2-edge connectivity of H. Consequently, the edge $\{u, v\}$ closes a cycle C with e satisfying $|C \setminus S| = 1$. □

In light of the discussion above we can easily conclude that the connecting 1-cyclic sets in the graph are exactly the spanning trees of 2-edge connected subgraphs containing s and t. Consequently to obtain such a set of minimum cardinality one needs to find a two edge connecting subgraph in G connecting s and t with *a minimum number of vertices* (and take a spanning tree of it). Hence, the following high-level description of an algorithm returns an optimal $(1, 1)$-recoverable set.

Algorithm $(1, 1)$*-ARCP*:

1. Find a 2-edge connected subgraph H, containing s and t with a minimum number of vertices.
2. Return a spanning tree S of H.

In the rest of this section we show that one can efficiently determine a 2-edge connected subgraph H of G with a minimum number of vertices connecting s and t. Notice that we can restrict ourselves to subgraphs that consists of two edge-disjoint paths from s to t: on the one hand the graph H contains two edge-disjoint paths from s to t by the max-flow min-cut theorem, on the other hand a subgraph consisting of two edge-disjoint paths from s to t is a 2-edge connected graph containing s and t. Hence the problem of finding H can be reduced to finding two edge-disjoint paths between s and t that touch a minimum number of vertices. In the following we show how to construct two edge-disjoint paths between s and t, whose union is a desired subgraph.

The following structural lemma provides the key property for our method.

Lemma 9. *Let U be a smallest set of edges comprising a two edge connected subgraph in G connecting s and t with a minimum number of vertices. Let p and q be two edge-disjoint paths from s to t in U and let the vertices on those paths according to their order from s to t be*

$$p = (s = u_1 \to u_2 \to \cdots \to u_k = t),$$
$$q = (s = v_1 \to v_2 \to \cdots \to v_l = t).$$

Then if two vertices v_{i_1}, v_{i_2} with $i_1 < i_2$ are also vertices of p, namely $v_{i_1} = u_{j_1}$ and $v_{i_2} = u_{j_2}$ for some $1 \leq j_1, j_2 \leq k$, then $j_1 < j_2$.

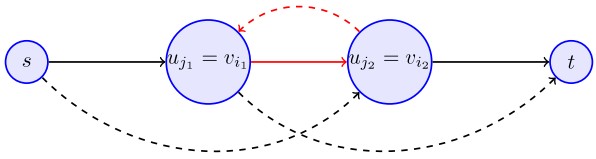

Fig. 1. The thick subpaths and the dashed subpaths correspond to p and q respectively. The red cycle can be removed without breaking 2-edge connectivity.

Proof. We proceed by contradiction, assuming that $j_1 > j_2$. The cases $u_{j_1} = s$ or $u_{j_2} = t$ are not possible since then p would not be a path because it would not be node-disjoint. Fig. 1 illustrates the situation with $j_1 > j_2$. It is easy to see that two shorter edge-disjoint paths p', q' can be constructed in this case from s to t by taking p' and q' to be

$$p' = (s = u_1 \to \cdots \to u_{j_2} \to v_{i_2+1} \to \cdots \to v_l = t)$$
$$q' = (s = v_1 \to \cdots \to v_{i_1} \to u_{j_1+1} \to \cdots \to u_k = t).$$

We obtained a contradiction to the fact that H was chosen to be with the minimum number of edges among all 2-edge connected subgraphs connecting s and t with a minimum number of vertices. □

Lemma 9 suggests that one can look for 2-edge connected subgraphs in which the two paths from s to t induce a linear order on the vertices in the subgraph (or in other words the order of the vertices which is defined by one path is maintained by the order defined by the other path). The previous lemma implies the following results which can then easily be used to derive an efficient algorithm.

Lemma 10. *Let U be a smallest set of edges comprising a 2-edge connected subgraph in G connecting s and t with a minimum number of vertices. Let p and q be the two edge-disjoint paths from s to t in U. Let $s = u_1, u_2, \ldots, u_l = t$ be the vertices which appear on both paths sorted according to their order on p from s to t. Then*

1. *The vertices $u_1, u_2, \ldots, u_l$ appear in the same order when traversing q from s to t.*
2. *For every $i \in [l-1]$ the union of the parts of p and q between u_i and u_{i+1} is a shortest edge cycle in G containing u_i and u_{i+1}.*

Proof. (1) Follows immediately from Lemma 9. For (2) notice that the subpaths of p and q between u_i and u_{i+1} form together a vertex-disjoint cycle in U with a minimum number of edges. Vertex-disjointness follows by definition of u_i and u_{i+1}. Assume toward contradiction that a cycle $\bar{C}$ containing u_i and u_{i+1} with fewer edges existed in G. We can replace the cycle formed by the subpaths of p and q between u_i and u_{i+1} by the cycle $\bar{C}$ to obtain a new edge set U'. Notice that we replaced a vertex-disjoint cycle with another cycle (not necessarily vertex-disjoint) with a smaller number of edges, hence U' touches no more vertices than U does. Furthermore, U' has fewer edges than U and it is still clearly a 2-edge connected subgraph containing s and t. This contradicts the minimality of U. □

Hence, by Lemma 10 it suffices to find the family of cycles formed by the subpaths of p and q between u_i and u_{i+i} for $i \in \{1, \ldots, l\}$. Therefore the problem of finding a smallest set U of edge comprising a 2-edge connected subgraph in G containing s and t and with a minimum number of vertices, reduces to finding a sequence of vertices $s = u_1, \ldots, u_l = t$ and for every pair u_i, u_{i+1} with $i \in \{1, \ldots, l-1\}$ a cycle C_i of minimum length containing u_i and u_{i+1} such that $l + \sum_{i=1}^{l-1}(|C_i| - 2) = 1 + \sum_{i=1}^{l-1}(|C_i| - 1)$ is minimal.

This justifies the following procedure for finding a desired set of edges U.

Procedure FindSubgraph:

1. For every pair of vertices $u, v \in V$ find a shortest (edge) cycle $C_{u,v}$ in G containing u and v and define $l_{u,v} = |C_{u,v}| - 1$ (the number of edges in $C_{u,v}$ minus 1).
2. Find a shortest path p^* in the complete graph on the vertex set V from s to t with l as the associated length function.
3. Set $U = \cup_{\{u,v\} \in p^*} C_{u,v}$.

We note that finding a shortest cycle containing two vertices is equivalent to finding two edge-disjoint paths with minimum total combined length. The latter problem can be solved using network flow. In particular, one can assign capacities and costs 1 to all edges in the graph and set the source of the flow to be s and the sink to be t. The edges used in a minimal cost 2-flow correspond to the edges which comprise a minimum pair of paths. Hence, the procedure FindSubgraph has a polynomial running time. More precisely the algorithm's running time is dominated by the computation of the length function l. The length function can be computed by invoking a minimum-cost flow algorithm $O(|V|^2)$ times (once for each pair of vertices). Let $F(G)$ denote the running time of one such flow problem on G. Then the total running time is $O(|V|^2 F(G))$.

Theorem 11. (1, 1)*-ARCP can be solved in polynomial time.*

3.3. A 2-approximation algorithm for variable r

In this section we consider the ARCP with $f = 1$ and variable r and present a 2-approximation for this problem. The algorithm extends the idea presented in the previous section for the case $r = 1$. Again, we want to find a sequence of vertices $s = u_1, \ldots, u_l = t$ and for every $i \in [l-1]$ we construct an r-cyclic set connecting u_i to u_{i+1}. In the previous section, the r-cyclic sets used to connect u_i to u_{i+1} were obtained by choosing a shortest cycle between u_i and u_{i+1} and removing a maximum number of edges from the cycle. In the current setting with variable r, we use additionally a second kind of r-cyclic sets since there is the following additional difficulty in this setting compared to the case $r = 1$. Let C_i be a cycle with a minimum number of edges that contains the vertices u_i and u_{i+1}. In the setting $r = 1$ we eventually removed one edge of C_i to obtain the final 1-cyclic set connecting u_i and u_{i+1}. Hence, the cycle C_i contributed with $|C_i| - 1$ edges to the final solution. However when $r > 1$ edges can be recovered, it is not always the case that the best cycle to connect u_i and u_{i+1} is a cycle with a minimum number of edges. For example assume to simplify the exposition that r is divisible by four, and let C and C' be two cycles containing two vertices u_i and u_{i+1}. Let $|C| = r/2$ and $|C'| = r + 1$, and let p, q and p', q', respectively, be the two paths in C and C', respectively, that connect u_i and u_{i+1}. Assume $|p| = |q| = r/4$ and $|p'| = r$, $|q'| = 1$. Therefore, a subset of the edges of C of minimum cardinality that connects u_i and u_{i+1} and can be recovered if one edge fails is given by the path p, and thus has cardinality $r/4$. However, for the cycle C' such a subset is given by q' and has only cardinality 1. A key observation is that it is typically good to have a cycle C_i between u_i and u_{i+1} such that the two disjoint paths in the cycle connecting u_i and u_{i+1} are quite unbalanced, i.e., one is much longer than the other. The following definition of *cycle recovery length* formalizes this, where $\mathcal{C}(u, v)$ denotes the set of all cycles in G containing the vertices u and v.

Definition 12. Let u and v be two vertices in G. The *cycle recovery length with respect to* r, denoted by $l_r(u, v)$, is defined as

$$l_r(u, v) = \min_{C \in \mathcal{C}(u,v)} (|p_C| + \max\{0, |q_C| - r\}),$$

where p_C denotes the set of edges on the shorter path on C from u to v, and q_C denotes the set of edges on the longer path on C from u to v. If both paths have equal length, one is chosen arbitrarily to be the shortest and the value of $l_r(u, v)$ is not affected by this choice.

Hence, for a given cycle C, it is possible to choose a subset of $l_r(u, v)$ edges of C such that u and v are connected and any failure of one edge can be recovered by r recovery edges.

We do not know how to efficiently find a cycle between two vertices u, v with minimum recovery length. However, we can overcome this problem by using an additional type of r-cyclic set to connect u and v. For every pair of vertices u and v we define an r-cyclic set $A_{u,v} \subseteq E$ as follows. Let C be a cycle with a minimum number of edges that contains u and v, and let p_C and q_C be defined as in Definition 12. We distinguish two cases, $|q_C| > r$ and $|q_C| \leq r$. If $|q_C| > r$, then we define $A_{u,v}$ to be the set $C \setminus B$, where B is an arbitrarily chosen set of r edges on q_C. If $|q_C| \leq r$, we choose $A_{u,v}$ to be a shortest path between u and v.

Lemma 13. *For every pair of edges u, $v \in V$, $A_{u,v}$ is an r-cyclic set satisfying $|A_{u,v}| \leq l_{u,v}$.*

Proof. We first consider the case $|q_C| > r$. Notice, that $A_{u,v}$ is clearly recoverable in this case, since any failure of an edge on p_C can be recovered by the r edges in B. Let $\widetilde{C}$ be the cycle connecting u and v with minimum recovery length. We clearly have $l_r(u, v) \geq |\widetilde{C}| - r$. As C is a shortest cycle containing u and v, $|C| \leq |\widetilde{C}|$. Hence $|A_{u,v}| = |C| - r \leq |\widetilde{C}| - r = l_{u,v}$.

Now assume $|q_C| \leq r$. Notice that there cannot be any r-cyclic set with a smaller cardinality than $A_{u,v}$ that contains u and v, since any such set must contain a path from u to v. Furthermore, for every failure edge $e \in A_{u,v}$, either $e \notin q_C$ or $e \notin p_C$, since p_C and q_C are disjoint. Hence, if e fails we can recover by choosing as recovery edges one of the paths p_C, q_C that does not contain e. Since $|q_C| \leq r$ and p_C is by definition not longer than q_C, both of these paths contain at most r edges. □

Our algorithm then works as the one in the previous section. In a first step we determine for every pair of vertices u, $v \in V$, a set $A_{u,v}$ as described above. Let $\bar{l}_r(u, v) = |A_{u,v}|$. Then, a shortest path p is determined in the complete graph on V with the edge lengths given by $\bar{l}_r$. The algorithm finally returns the set $A = \cup_{e \in p} A_e$. We clearly have that A is an r-cyclic set containing u and v. In the following we show that A is a 2-approximation to the $(1, r)$-ARCP.

Let $\bar{d}_r(u, v)$ be the shortest distance between u and v in the complete graph on V with edge length given by $\bar{l}_r$. Hence, $|A| = \bar{d}_r(s, t)$. Similarly, we denote by $d_r(u, v)$ the shortest distance between u and v in the complete graph on V with edge lengths given by l_r. Since, $\bar{l}_r(u, v) \leq l_r(u, v)$, we have $\bar{d}_r(u, v) \leq d_r(u, v)$. We will prove the following theorem which immediately implies that A is a 2-approximation, since by the above discussion $|A| = \bar{d}_r(s, t) \leq d_r(s, t)$.

Theorem 14. *$d_r(s, t) \leq 2\mathrm{OPT}_r(G)$, where $\mathrm{OPT}_r(G)$ is the cardinality of an optimal solution to the given $(1, r)$-ARCP.*

To prove Theorem 14, we give some auxiliary results which are insightful in their own right and allow to deduce Theorem 14 rather easily. We denote by $\mathcal{C}$ the set of all cycles in G.

Definition 15. Let S be a $(1, r)$-recoverable set. A family of cycles $\{C_1, \ldots, C_m\} \subset \mathcal{C}$ is a *recoverable cycle family for S* if $|C_i \setminus S| \leq r$ for $i \in [m]$ and $S \subset \cup_{i=1}^m C_i$. The family is called a *minimum recoverable cycle family for S* if furthermore $\sum_{i=1}^m |C_i \setminus S|$ is minimum among all recoverable cycle families for S.

Lemma 16. *Let S^* be a $(1, r)$-recoverable set of minimum cardinality and let P^* be the unique s–t path in S^*. Let $\mathcal{F} = \{C_1, \ldots, C_m\} \subset \mathcal{C}$ satisfying $|C_i \setminus S^*| \leq r$ for $i \in [m]$ and $P^* \subset \cup_{C\in\mathcal{F}} C$. Then $C_1, \ldots, C_m$ is a recoverable cycle family for S^*.*

Proof. The only missing condition to check for $\mathcal{F}$ to be a recoverable cycle family for S^*, is $S^* \subset \cup_{C\in\mathcal{F}} C$. Let $S = S^* \cap (\cup_{C\in\mathcal{F}} C)$. S is a $(1, r)$-recoverable set since the edges of S can be recovered by the cycles $C_1, \ldots, C_m$. Since S^* has minimal cardinality and $S \subset S^*$, we obtain $S = S^*$ and thus $S^* \subset \cup_{C\in\mathcal{F}} C$. □

Lemma 17. *Let S^* be a $(1, r)$-recoverable set of minimum cardinality and let P^* be the unique s–t path in S^*. Let $\mathcal{F} = \{C_1, \ldots, C_m\} \subset \mathcal{C}$ be a minimum recoverable cycle family for S^*. Then $C_i \cap P^* \neq \emptyset$ for $i \in [m]$.*

Proof. Assume by contradiction that $C_j \cap P^* = \emptyset$ for some $j \in [m]$. Hence, $P^* \subset \cup_{i\in[m]\setminus\{j\}} C_i$, which implies by Lemma 16 that $\{C_i \mid i \in [m] \setminus \{j\}\}$ is a recoverable cycle family for S^*. This contradicts that $\mathcal{F}$ was minimum because $C_j \setminus S^* \neq \emptyset$ since by minimality of S^*, S^* does not contain cycles. □

Lemma 18. *Let S^* be a $(1, r)$-recoverable set of minimum cardinality and let P^* be the unique s–t path in S^*. We denote by $s = v_1, \ldots, v_l = t$ the vertices on P^* when traversing the path from s to t. Let $\mathcal{F} = \{C_1, \ldots, C_m\} \subset \mathcal{C}$ be a minimum recoverable cycle family for S^*. Then for $i \in [m]$, $C_i \cap P^*$ is a subpath of P^* between v_{a_i} and v_{b_i}, where $a_i = \min\{h \in [l] \mid v_h \text{ lies on } C_i\}$ and $b_i = \max\{h \in [l] \mid v_h \text{ lies on } C_i\}$.*

Proof. Let $e_i = \{v_i, v_{i+1}\}$ for $i \in [l-1]$. With every cycle C_i for $i \in [m]$, we associate a family of cycles, as follows. $C_i \setminus P^*$ consists of a union of node-disjoint paths, each of which starting and ending at a vertex on P^*. Each of these paths can be completed to a cycle by adding a subpath of P^* to it. We denote by $D^i_1, \ldots, D^i_{m_i}$, all cycles obtained in this way by completing the different maximal paths of $C_i \setminus P^*$ to cycles. We call the cycle family $\{D^i_1, \ldots, D^1_{m_i}\}$ the *corresponding flattened cycle family* to C_i. We clearly have $\sum_{k=1}^{m_i} |D^i_k \setminus S^*| = |C_i \setminus S^*|$ for $i \in [m]$. This implies that $|D^i_k \setminus S^*| \leq r$ for $i \in [m]$ since $|C_i \setminus S^*| \leq r$, and furthermore that by replacing a subset of the cycles of the family $\mathcal{F}$ by the cycles in their corresponding flattened cycle family, we again get a minimum recoverable cycle family for S^*.

Assume by contradiction that C_1 is a cycle not satisfying the claim of the lemma. Notice that this is equivalent to saying that the flattened cycle family which corresponds to C_1 consists of more than one cycle. Observe that the cycle family $\{D^1_1, \ldots, D^1_{m_1}\}$ covers the subpath of P^* between v_{a_1} and v_{b_1}. Since $C_1 \cap P^*$ is not the subpath of P^* between a_1 and b_1, there is another cycle, assume C_2, such that $C_2 \cap P^*$ has a non-empty intersection with the set $\{e_{a_1}, \ldots, e_{b_1-1}\} \setminus C_1$. Let $\mathcal{F}' = \{C_3, \ldots, C_m\} \cup \{D^1_1, \ldots, D^1_{m_1}\} \cup \{D^2_1, \ldots, D^2_{m_2}\}$. By the discussions in the previous paragraph, $\mathcal{F}'$ is a minimum recoverable cycle family for S^*. In the following we derive a contradiction by showing that there is a cycle $D \in \{D^1_1, \ldots, D^1_{m_1}\}$ that can be removed from $\mathcal{F}'$ such that the resulting set is still a recoverable cycle family, which contradicts minimality of $\mathcal{F}'$. To guarantee that D is removable, D will be chosen such that $D \cap P^* \subset \bigcup_{C\in\mathcal{F}'\setminus\{D\}} C$. Lemma 16 then guarantees that $\mathcal{F}' \setminus \{D\}$ is a recoverable cycle family for S^*.

The fact of $\mathcal{F}'$ being a minimum recoverable cycle family for S^*, implies that none of the two intervals $[a_1, b_1]$, $[a_2, b_2]$ contains the other, since if for example $[a_2, b_2] \subset [a_1, b_1]$, then $\{C_3, \ldots, C_m\} \cup \{D^1_1, \ldots, D^1_{m_1}\}$ would be a recoverable cycle family for S^*, contradicting the minimality of $\mathcal{F}$. Assume without loss of generality that $a_1 < a_2 < b_1 < b_2$. Observe that $C_1 \cap P^*$ contains a non-trivial path P', i.e., with strictly positive length, that has v_{b_1} as one of its endpoints: if this was not the case then there would be two cycles D' and D'' in $\{D^1_1, \ldots, D^1_{m_1}\}$ that have v_{b_1} as one of their endpoints, and thus one of the sets $D' \cap P^*$, $D'' \cap P^*$ is included in the other, implying that one of the cycles can be removed from $\mathcal{F}$. We denote by q the other endpoint of P'. Notice that $a_2 < q$ since by choice of C_2, $C_2 \cap P^*$ has a non-empty intersection with $\{e_{a_1}, \ldots, e_{b_1-1}\} \setminus C_1$ and P' contains the edges between v_q and v_{b_1}. The flattened cycle family that corresponds to C_1 contains a cycle D^1_j such that $D^1_j \setminus P^*$ has q as one of its endpoints. Hence, $D^1_j \cap P^*$ is a subpath of P^* where the endpoint closer to t is v_q, and we denote by v_{c_j}, with $c_j < q$, the other endpoint. Similarly, the flattened cycle family that corresponds to C_1 contains a cycle D^1_k that has v_{b_1} as one of its endpoints. Again, $D^1_k \cap P^*$ is a subpath of P^*, where v_{b_1} is the endpoint closer to t and we denote by v_{c_k} the other endpoint of the subpath (see Fig. 2). Notice that $c_k < a_2$, since otherwise $D^1_k \cap P^* \subset \cup_{i=1}^{m_2} D^2_i$, and hence D^1_k can be removed from $\mathcal{F}'$. Furthermore $c_j < c_k$, since otherwise $D^1_j \subset D^1_k$ and hence D^1_j can be removed from $\mathcal{F}'$. However, under the current assumptions (see Fig. 2 to recall the current order of the mentioned vertices on P^*) D^1_k can be removed from $\mathcal{F}'$ since every edge $e \in D^1_k \cap P^*$ is either contained in $\cup_{i=1}^{m_2} D^2_i$, which covers all edges on P^* between v_{a_2} and v_{b_2}, or $e \in D^1_j$. □

Lemma 19. *Let S^* be a $(1, r)$-recoverable set of minimum cardinality and let P^* be the unique s–t path in S^*. Let $s = v_1, \ldots, v_l = t$ be the vertices on P^* numbered consecutively when traversing the path from s to t. Let $\mathcal{F} = \{C_1, \ldots, C_m\} \subset \mathcal{C}$ be a minimum recoverable cycle family for S^*. Furthermore, we assume that the cycles are numbered such that if $i < j$, then $\max\{h \in [l] \mid v_h \text{ lies on } C_i\} \leq \max\{h \in [l] \mid v_h \text{ lies on } C_j\}$. Then*

1. *C_i and C_j do not have a common vertex if $|i - j| > 1$.*

If furthermore, $\mathcal{F}$ is such that $\sum_{C\in\mathcal{F}} |P^* \cap C|$ is minimum among all minimum recoverable cycle families for S^*, then the following holds.

2. *For $i \in [m-1]$, either $C_i \cap C_{i+1} \cap P^* = \emptyset$, or all common vertices of C_i and C_{i+1} lie on P^*.*

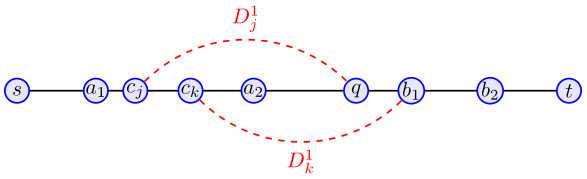

Fig. 2. The black line represents the path P^*. The nodes between s and t are labeled with respect to their index.

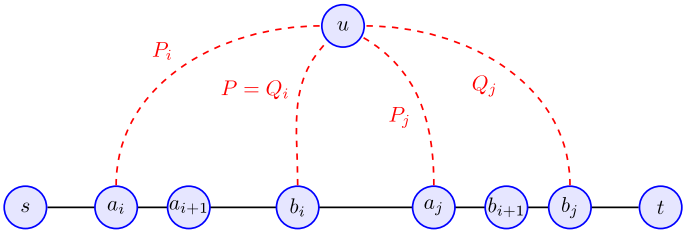

Fig. 3. The black line represents the path P^*. The nodes between s and t are labeled with respect to their index.

Proof. (1) For $k \in [m]$ we define $a_k = \min\{h \in [l] \mid v_h \text{ lies on } C_k\}$, $b_k = \max\{h \in [l] \mid v_h \text{ lies on } C_k\}$. By Lemma 18, $C_k \cap P^*$ for $k \in [m]$, consists of the subpath of P^* containing all edges between a_k and b_k. Notice that $a_i < a_j$ for $i < j$, since otherwise $C_i \cap P^* \subset C_j \cap P^*$ which in turn implies by Lemma 16 that $\mathcal{F}$ was not a minimum recoverable cycle family for S^* since C_i can be removed from $\mathcal{F}$.

Assume by contradiction that there is $i, j \in [m]$, $i + 1 < j$ such that C_i and C_j have a common vertex u. Let P_i and Q_i, be the subpaths of C_i between a_i and u, and between b_i and u, respectively. Similarly, let P_j and Q_j, be the subpaths of C_j between a_j and u, and b_j and u, respectively. Let $P \in \{P_i, Q_i, P_j, Q_j\}$ be the path such that $|P \setminus S^*|$ is minimal among the paths in $\{P_i, Q_i, P_j, Q_j\}$, breaking ties arbitrarily. Assume $P = Q_i$, the other cases are similar. Let $D \in \mathcal{C}$ be the cycle consisting of the edges in $Q_i \cup Q_j$ and the edges between v_{b_i} and v_{b_j} on P^* (see Fig. 3). By choice of D, the cycle family $\mathcal{F}' = (\mathcal{F} \setminus \{C_j\}) \cup \{D\}$ is again a recoverable cycle family for S^*. Furthermore $\sum_{C \in \mathcal{F}'} |C \setminus S^*| \leq \sum_{C \in \mathcal{F}} |C \setminus S^*|$, since $|D \setminus S^*| \leq |C_j \setminus S^*|$. Hence, $\mathcal{F}'$ is a minimum recoverable cycle family for S^*. However, $C_{i+1} \cap P^* \subset C_i \cup D$ since $C_{i+1} \cap P^*$ is a subpath of P^* between $a_{i+1} > a_i$ and $b_{i+1} < b_j$, and all edges on P^* between a_i and b_j are covered by $C_i \cup D$. Thus by Lemma 16, $\mathcal{F}' \setminus \{C_{i+1}\}$ is again a recoverable cycle family for S^*, which contradicts the minimality of the cycle family $\mathcal{F}'$.

(2) Assume by contradiction that there is some $i \in [m]$ such that $C_i \cap C_{i+1} \cap P^* \neq \emptyset$ and such that there is a vertex $u \in V$ that lies not on P^* but is on both cycles C_i and C_{i+1}. Again, for $k \in [m]$, let $a_k = \min\{j \in [l] \mid C_k \text{ passes trough } v_j\}$ and $b_k = \max\{j \in [l] \mid C_k \text{ passes trough } v_j\}$. Since $C_i \cap C_{i+1} \cap P^* \neq \emptyset$, we have $a_{i+1} < b_i$. For $k \in \{i, i+1\}$, let P_k and Q_k be the subpaths of C_k between a_k and u, and b_k and u, respectively. Let $P \in \{P_i, Q_i, P_{i+1}, Q_{i+1}\}$ be the path such that $|P \setminus S^*|$ is minimal among the paths in $\{P_i, Q_i, P_{i+1}, Q_{i+1}\}$, breaking ties arbitrarily. Assume $P = Q_i$, the other cases are similar. Let $D \in \mathcal{C}$ be the cycle consisting of the edges in $Q_i \cup Q_{i+1}$ and the edges of P^* between v_{b_i} and $v_{b_{i+1}}$. Let $\mathcal{F}' = (\mathcal{F} \setminus C_{i+1}) \cup \{D\}$. By choice of D we have $|D \setminus S^*| \leq |C_{i+1} \setminus S^*|$, which implies $|D \setminus S^*| \leq r$. Since furthermore $(C_i \cup C_{i+1}) \cap P^* \subset (C_i \cup D) \cap P^*$, we have by Lemma 16 that $\mathcal{F}'$ is again a minimum recoverable cycle family for S^*. As $|D \cap P^*| < |C_{i+1} \cap P^*|$, we have $\sum_{C \in \mathcal{F}'} |C \cap P^*| < \sum_{C \in \mathcal{F}} |C \cap P^*|$, which contradicts the assumption of the lemma. □

Proof of Theorem 14. Let S^* be a minimum $(1, r)$-recoverable set, let P^* be the unique s–t path in S^* and let $\mathcal{F} = \{C_1, \ldots, C_m\}$ be a minimum recoverable cycle family for S^*. Let $s = v_1, \ldots, v_l = t$ be the vertices on P^* numbered consecutively when traversing the path from s to t. For $i \in [m]$, let $b_i = \max\{h \in [l] \mid v_h \text{ lies on } C_i\}$. We assume that the cycles in $\mathcal{F}$ are numbered as in Lemma 19, namely if $i < j$, then $b_i \leq b_j$. Let $w_0 = s$ and for $i \in [m]$ let $w_i = v_{b_i}$. Clearly w_0 is on C_1, $t = w_m$ is on C_m and for every $i \in [m-1]$, w_i is on both C_i and C_{i+1}. Furthermore, for $i \in \{0, \ldots, m-1\}$, $l_r(w_i, w_{i+1}) \leq |C_i \cap S|$ because the cycle C_i covers the path between w_i and w_{i+1} on P^* and has a recovery length bounded by $|C_i \cap S|$, as $C_i \setminus S \leq r$. Since by Lemma 19(1) every edge of G can be shared by at most 2 cycles in $\mathcal{F}$ we obtain

$$d_r(s, t) \leq \sum_{i=0}^{m-1} l_r(w_i, w_{i+1}) \leq \sum_{i=1}^{m} |C_i \cap S| \leq 2|S|. \quad \square$$

Finally, we note that the running time of the algorithm is dominated by the computation of the sets $A_{u,v}$ for all pairs of vertices. The sets $A_{u,v}$ can be computed using a minimum-cost flow algorithm, hence the running time is $O(|V|^2 F(G))$, where $F(G)$ is the running time of the minimum-cost flow algorithm on G.

3.4. The existence of connected optimal $(1, r)$-recoverable sets

In this section we prove an additional structural result on optimal $(1, r)$-recoverable sets, namely that the problem can be restricted to connected sets. This additional property is not needed for the suggested algorithm, however, it provides a natural additional structure that might be of importance in some applications.

Lemma 20. *There is a $(1, r)$-recoverable set S^* of minimum cardinality that is connected, i.e., the subgraph consisting of the edges in S^* and the vertices adjacent to these edges is connected.*

Proof. Assume towards contradiction that the statement is false. Let S^* be a minimum $(1, r)$-recoverable set with minimum number of edges which are not contained in the connected component of its unique s–t path P^*. Let $\mathcal{F} = \{C_1, \ldots, C_m\}$ be a minimum recoverable cycle family for S^* with minimum $\sum_{C\in\mathcal{F}} |P^* \cap C|$ (as in Lemma 19(2)). We denote by $A \subset S^*$ the connected component of S^* that contains P^*. Let $e \in S^* \setminus A$. Consider all cycles in F which contain e. Lemma 19 implies that there are either one or two cycles containing e.

Assume first there is a unique cycle $C_i \in F$ containing e. Let $f \in C_i \setminus S^*$ be an edge touching a vertex in the connected component defined by A. Consider the set $S = S^* \setminus \{e\} \cup \{f\}$. It is easy to check that $\mathcal{F}$ is a recoverable cycle family for S as well, hence S is $(1, r)$-recoverable. However, the connected component in S that contains P^* has more edges than the corresponding connected component in S^*, contradicting the choice of S^*.

Assume next that there are two cycles containing e, namely $C_i, C_j \in \mathcal{F}$. Lemma 19 guarantees that these cycles are adjacent in the ordering defined in Lemma 19 (hence, say $j = i + 1$). Furthermore, part 2 of Lemma 19 suggests that $C_i \cap C_{i+1} \cap P^* = \emptyset$ (since $e \in (C_i \cap C_{i+1}) \setminus P^*$, C_i and C_{i+1} have a common vertex not lying on P^*), and Lemma 18 suggests that $C_i \cap P^*$ and $C_{i+1} \cap P^*$ are subpaths of P^*. We conclude that $C_i \cap P^*$ and $C_{i+1} \cap P^*$ are adjacent subpaths of P^* touching each other at a common vertex v on P^*. Let u be one of the endpoints of e. Denote by p_i (p_{i+1}, respectively) the path on $C_i \setminus P^*$ ($C_{i+1} \setminus P^*$, respectively) that connects v with u. Assume $|p_i \setminus S^*| \leq |p_{i+1} \setminus S^*|$ (the other case is analogous). Then we can replace the cycle C_{i+1} in the cycle family $\mathcal{F}$ with the cycle $C'_{i+1} = (C_{i+1} \setminus p_{i+1}) \cup p_i$ (in case $(C_{i+1} \setminus p_{i+1}) \cup p_i$ is not internally vertex-disjoint, we choose C'_{i+1} to be the cycle in $(C_{i+1} \setminus p_{i+1}) \cup p_i$ which covers $C_{i+1} \cap P^*$). If C'_{i+1} does not contain e, then we are in the previous case, i.e., only one cycle contains e, namely C_i. Therefore, we can assume that C_i and C_{i+1} both contain e, and $p_i = p_{i+1}$. Since e is not in A, there is an edge $f \in p \setminus A$ that is adjacent to an edge in A. Consider the set $S = (S^* \setminus \{e\}) \cup \{f\}$. In the remainder we show that $\mathcal{F}$ is also a recoverable cycle family for S. This implies that S is $(1, r)$-recoverable and leads to a contradiction since the connected component in S that contains P^* has more edges than the corresponding connected component in S^*.

We clearly have $S \subseteq \cup_{C\in\mathcal{F}} C$. For $\mathcal{F}$ to be a recoverable cycle family for S it remains to check whether $|C \setminus S| \leq r$ for all $C \in \mathcal{F}$. The only sets of type $C \setminus S$ to which an element was added compared to $C \setminus S^*$, are $C_i \setminus S$ and $C_{i+1} \setminus S$ which additionally contain the edge e. However, from both sets, the element f was removed. Hence $|C_i \setminus S| = |C_i \setminus S^*| \leq r$ and $|C_{i+1} \setminus S| = |C_{i+1} \setminus S^*| \leq r$. □

4. Complexity of ARCP

We show a reduction from the Most Vital Arcs problem (MVAP), which was introduced by Corley and Sha [7]. The task is to increase the length of a shortest path between two given vertices in a graph by removing some fixed number of edges. More precisely, we are given an undirected graph $G = (V, E)$ with two distinguished vertices $s, t \in V$ and an integer $k \in \mathbb{N}$. For a set $U \subseteq E$ we defined by $d_U(s, t)$ the cardinality of a shortest path between s and t in $(V, E \setminus U)$. In case s and t are disconnected in $(V, E \setminus U)$, we set $d_U(s, t) = \infty$. The MVAP asks to find a set $U \subseteq E$ with $|U| \leq k$ that maximizes $d_U(s, t)$. In [2] it was shown that the following natural decision version of MVAP is NP-hard: decide for some given integers $f, D \in \mathbb{N}$ whether there is a set $U \subseteq E$, $|U| \leq f$ such that $d_U(s, t) > D$.

Theorem 21. *The ARCP is NP-hard if f and r are part of the input, and no polynomial α-approximation algorithm exists for $\alpha < 2$ unless $P =$ NP. Furthermore, deciding whether a solution to ARCP is feasible is NP-hard.*

Proof. Consider an instance of the decision version of MVAP. Hence, we are given a graph $G = (V, E)$ with two distinguished vertices $s, t \in V$, and two integers $k, D \in \mathbb{N}$. Let $G' = (V, E')$ be the graph defined by $E' = E \cup \{e\}$, where e is an edge between s and t. Consider the ARCP on G' between s and t with parameters $f = k + 1$ and $r = D$. We will show that the thus defined ARCP has a solution of cardinality one if and only if the decision problem of MVAP evaluates to false.

Consider first the case where MVAP evaluates to false. In this case $\{e\}$ is a solution to the ARCP because for any failure set $F \subseteq E'$ with $|F| \leq f$ and $e \in F$, there is a path in $(V, E \setminus F)$ of length at most $D = r$. This path can be used for recovery. Conversely, assume that MVAP evaluates to true. Hence, there is a set $U \subseteq E$ with $|U| \leq k$ such that there is no path in $(V, E \setminus U)$ of length $\leq D = r$. Let $F = U \cup \{e\}$. Consequently, an (f, r)-recoverable set $S \subseteq E$ for the ARCP must contain at least one edge from $E \setminus F$. Since there is no edge in $E \setminus F$ between s and t (all paths between s and t in $E \setminus F$ have length strictly greater than $r \geq 1$), S must contain at least two edges since it has to be connecting. Hence, in this case the ARCP has no solution of cardinality one.

Note that the optimal solution value to the transformed ARCP is 1 if and only if MVAP evaluates to false, and in any other case the optimal solution value is at least 2. As a result, obtaining an α-approximation with $\alpha < 2$ for ARCP allows to decide MVAP. Hence the inapproximability result.

Finally note that deciding the feasibility of the set $\{e\}$ for ARCP is equivalent to deciding MVAP, hence deciding feasibility of a solution to ARCP is NP-hard. □

5. Conclusions

This paper studies the adaptable robust connection problem which exhibits a rich combinatorial structure. The graph-theoretic structure of the problem in the case $f = 1$ is provided. In the case $f = r = 1$ an exact polynomial algorithm is developed. In the case $f = 1$ and variable r a 2-approximation algorithm is provided and a connection to a new combinatorial optimization problem is established. Hardness of approximation within any factor $\alpha < 2$ is proven for ARCP with variable f and r.

We conclude by listing some of the open problems and possible research directions that arise from our study.

- The complexity of the problem in the cases of fixed f and fixed r remains open. In the case $f = 1$ the problem is equivalent to finding a minimum size r-cyclic set connecting s and t.
- The complexity of deciding whether a set S of edges is (f, r)-recoverable in the case that r is fixed remains open.
- The complexity of finding the cycle containing two specified vertices and minimum reduced cost remains open.
- An interesting variation of the ARCP might be to drop the requirement that s and t need to be connected in the first-stage solution.

Acknowledgements

We are grateful to the referees whose comments helped to improve the presentation of the results.
The second author is supported by the Swiss National Science Foundation, grant number: PBEZP2-129524.

References

[1] H. Aissi, C. Bazgan, D. Vanderpooten, Approximation complexity of min–max (regret) versions of shortest path, spanning tree, and knapsack, in: ESA, 2005, pp. 862–873.
[2] A. Bar-Noy, S. Khuller, B. Schieber, The complexity of finding most vital arcs and nodes, Technical Report, Univ. of Maryland Institute for Advanced Computer Studies Report No. UMIACS-TR-95-96, College Park, MD, USA, 1995.
[3] A. Ben-Tal, S. Boyd, A. Nemirovski, Extending scope of robust optimization: comprehensive robust counterparts of uncertain problems, Mathematical Programming 107 (1) (2006) 63–89.
[4] D. Bertsimas, C. Caramanis, Finite adaptability in multistage linear optimization, 2007.
[5] D. Bertsimas, M. Sim, Robust discrete optimization and network flows, Mathematical Programming. Series B 98 (2003).
[6] C. Büsing, The exact subgraph recoverable robust shortest path problem, Robust and Online Large-Scale Optimization (2009) 231–248.
[7] H.W. Corley, D.Y. Sha, Most vital links and nodes in weighted networks, Operations Research Letters 1 (4) (1982) 157–160.
[8] K. Dhamdhere, V. Goyal, R. Ravi, M. Singh, How to pay, come what may: approximation algorithms for demand-robust covering problems, in: FOCS'05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, USA, 2005, pp. 367–378.
[9] U. Feige, K. Jain, M. Mahdian, V. Mirrokni, Robust combinatorial optimization with exponential scenarios, in: IPCO'07: Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 439–453.
[10] D. Golovin, V. Goyal, R. Ravi, Pay today for a rainy day: improved approximation algorithms for demand-robust min-cut and shortest path problems, in: B. Durand, W. Thomas (Eds.), Proceedings of the 23rd Symposium on Theoretical Aspects of Computer Science, STACS 2006, in: Lecture Notes in Computer Science, vol. 3884, Springer-Verlag, 2006, pp. 206–217.
[11] W.D. Grover, Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.
[12] R. Khandekar, G. Kortsarz, V. Mirrokni, M.R. Salavatipour, Two-stage robust network design with exponential scenarios, in: ESA'08: Proceedings of the 16th Annual European Symposium on Algorithms, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 589–600.
[13] S. Khuller, Approximation algorithms for finding highly connected subgraphs, in: Approximation Algorithms for NP-Hard Problems, PWS Publishing Co., Boston, MA, USA, 1997, pp. 236–265.
[14] P. Kouvelis, G. Yu, Robust Discrete Optimization and its Applications, Kluwer Academic Publishers, Boston, 1997.
[15] C. Liebchen, M. Lübbecke, R.H. Möhring, S. Stiller, Recoverable robustness, Technical Report, ARRIVAL-Project, August 2007.
[16] M. Pióro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
[17] C. Puhl, Recoverable robust shortest path problems. Preprint 034-2008, Intitute of Mathematics, Technische Universität Berlin, 2008.
[18] A. Schrijver, Combinatorial Optimization—Polyhedra and Efficiency, Springer, 2003.
[19] G. Yu, J. Yang, On the robust shortest path problem, Computers & Operations Research 25 (6) (1998) 457–468.
[20] P. Zielinski, The computational complexity of the relative robust shortest path problem with interval data, European Journal of Operational Research 158 (3) (2004) 570–576.