DISS. ETH NR. 20327

Structural Robustness in Combinatorial Optimization

DISSERTATION

Submitted to

ETH ZURICH

for the degree of

DOCTOR OF SCIENCES

presented by

David Adjiashvili
M. Sc. In Computer Science, Weizmann Inst. of Science

born March 4, 1984
citizen of Israel

accepted on the recommnedation of

Prof. Dr. Komei Fukuda, examiner
Dr. Rico Zenklusen, co-examiner
Prof. Dr. Emo Welzl, co-examiner

2012

Acknowledgements

My stay at IFOR was a remarkably enjoyable and fruitful experience, from the professional and from the personal perspectives alike. I would like to start by collectively thanking everybody, who was involved with IFOR in the last four years, for making it such an enjoyable environment.

I wish to express my deepest gratitude to my adviser, Prof. Komei Fukuda for trusting me along the way, giving me the freedom to pursue my research interests and creating a warm and productive working environment. I owe him a lot of what I learned in the field of optimization in many discussions and collaborations.

I owe many thanks to Prof. Hans-Jakob Lüthi, head of IFOR, for letting me work in his institute. His welcoming attitude and support made it particularly easy for me to blend into the group.

I am grateful to Dr. Rico Zenklusen for all the time he put in to help kick-start the research that is summarized in this thesis, and consequently for his collaboration on the topics of this thesis. This outcome would not have been possible without his friendly support.

Many thanks to Prof. Emo welzl, who agreed to referee this thesis.

I owe a lot to Kaspar Schüpbach and Martin Fuchsberger for being good friends and helping me settle into IFOR and Switzerland when I arrived. I would like to thank all my other friends and colleagues at IFOR and ETH for helping making Switzerland a second home.

Finally, I am grateful to my family and friends back home, and those scattered around the world for their continuous love and support.

Abstract

The need to model the presence of uncertainty in systematic decision making situations has sprung the development of two main fields in optimization theory, *stochastic optimization* and *robust optimization*. Stochastic optimization models the uncertainty underlying the input data using probability distributions, and asks to optimize a certain stochastic quantity, such as the expected value. The main shortcomings of stochastic optimization is the need to assume certain stochastic knowledge about the possible realizations of input data, as well as the implicit assumption that the optimization needs to be performed many times, in order for the optimized stochastic quantity to be meaningful. On the other hand, robust optimization assumes the presence of an adversary that controls the materialized input data, and chooses it so as to minimize the decision makers' utility. This approach does away with the aforementioned shortcomings, but it has one of its own. In particular, robust optimization often gives overly conservative solutions. In many situations, however, this level of conservatism is appropriate, while in other situations it can be regulated.

In a nutshell, this thesis studies robust combinatorial optimization problems. In its heart is a study of several new *robust counterparts* for combinatorial problems. Unlike many existing such models, which model uncertainty in the form of unknown costs of resources, our models try to capture the uncertainty present in the underlying combinatorial structure. In other words, the present thesis studies the effect of *structural uncertainty* on classical combinatorial optimization problems. Concretely, we assume that an adversary is able to remove a certain subset of resources, such as edges in a graph, after the *nominal solution* was chosen. The nominal solution is feasible if for any materialization of an admissible *failure scenario*, the required structural property of the solution (e.g. containing an s-t path, or comprising a connected graph) is maintained.

Along these lines we define several new models of robust combinatorial optimization. The common feature of all these models is the assumption that the underlying combinatorial structure is an upper-ideal. This property ensures that all resulting robust counterparts addressed by this thesis are covering problems. On the one hand, this property is realistic from the point of view

of many applications. On the other hand, it helps to control the quality of the robust solutions, thus making it less conservative than other robust models.

The focus of this thesis is put on the development of efficient exact and approximate algorithms for robust counterparts of classical combinatorial optimization problems. These results are complemented by corresponding complexity-theoretic results. A survey of the robust combinatorial optimization literature is also presented.

Zusammenfassung

Das Bedürfnis für die Modellierung von Unsicherheit in der systematischen Entscheidungsfindung hat zur Entwicklung von zwei Hauptgebieten in der Optimierungstheorie geführt, der *stochastischen Optimierung* und der *robusten Optimierung*. Die stochastische Optimierung modelliert die den Inputdaten zugrunde liegende Unsicherheit mit Hilfe von Wahrscheinlichkeitsverteilungen und hat das Ziel, eine bestimmte stochastische Grösse, wie z.B. den Erwartungswert, zu optimieren. Die grössten Einschränkungen von stochastischer Optimierung sind die Voraussetzungen, dass gewisse stochastische Informationen über die möglichen Realisierungen der Inputdaten bekannt sein müssen und die implizite Annahme, dass das Optimierungsproblem wiederholt gelöst wird, so dass die optimierte stochastische Grösse eine Bedeutung erhält. Robuste Optimierung auf der anderen Seite nimmt an, dass ein Gegenspieler existiert, der die Inputdaten so steuert, dass der Nutzen für den Entscheidungsträger minimiert wird. Dieser Ansatz vermeidet die vorher erwähnten Nachteile, führt aber oft zu übermässig konservativen Lösungen. In manchen Situationen ist dieser Grad an Konservativismus berechtigt, in anderen kann er reguliert werden.

Kurz gefasst untersucht diese Dissertation robuste kombinatorische Optimierungsprobleme. Im Zentrum stehen mehrere neue robuste Versionen für kombinatorische Probleme. Im Gegensatz zu vielen existierenden Modellen, die Unsicherheit in der Form von ungewissen Ressourcenkosten modellieren, erfassen unsere Modelle die Unsicherheit in der zugrunde liegenden kombinatorischen Struktur. Anders gesagt untersucht die vorliegende Dissertation den Effekt von *struktureller Unsicherheit* auf klassische kombinatorische Probleme. Konkret nehmen wir an, dass ein Gegenspieler gewisse Untermengen der Ressourcen entfernen kann, wie z.B. Kanten in einem Graphen, nachdem die *nominale Lösung* bestimmt wurde. Die nominelle Lösung ist zulässig falls die geforderte strukturelle Eigenschaft (z.B. das Beinhalten eines kürzesten *s-t* Pfades, oder eines zusammenhängenden Graphen) für jedes mögliche *Störungsszenario* erhalten bleibt.

In diesem Sinne definieren wir mehrere neue robuste kombinatorische Optimierungsprobleme. Die gemeinsame Eigenschaft von all diesen Modellen ist die Annahme, dass die zugrunde liegende kombinatorische Struktur aufwärts

monoton ist. Diese Eigenschaft bewirkt, dass die resultierenden robusten Probleme, die in dieser Dissertation untersucht werden, Überdeckungsprobleme sind. Auf der einen Seite ist diese Eigenschaft realistisch für viele Anwendungen. Auf der anderen Seite hilft es, die Qualität der robusten Lösung zu beeinflussen, so dass diese weniger konservativ ausfallen als für andere robuste Modelle.

Der Fokus der Dissertation liegt in der Entwicklung von effizienten Algorithmen zur exakten oder approximativen Lösung von robusten Versionen klassischer kombinatorischer Probleme. Diese Resultate werden ergänzt mit entsprechenden komplexitätstheoretischen Resultaten. Zudem ist ein Überblick über die Literatur in robuster kombinatorischer Optimierung enthalten.

Contents

Chapter 1

Introduction

1.1 Decision making under uncertainty

Uncertainty is in the heart of many decision making situations. Rapidly growing volumes of data, distributed representation of resources and adversarial attacks are some of many examples for causes of uncertainty. To cope with this inherent aspect of decision making, the optimization community devised many models, and corresponding solution methods, which incorporate uncertainty in the input data into systematic decision making procedures. To date, a huge mass of literature is available, which is devoted to optimization problems with uncertainty. The vast number of different models can be crudely divided into *stochastic models* and *robust models*. Stochastic models incorporate probabilities to represent the uncertainty underlying the input data. In most stochastic models a distribution over a certain set of possible realizations of the input data is given. The decision maker needs to find a solution, which optimizes a certain probabilistic function of the solution, such as the expected cost, or profit, the variance etc. Robust models assume that the worst-case scenario always happens after the solution was chosen. In other words, a certain *adversary* observes the chosen solution and picks a *scenario* from a certain set of admissible data realizations, which results in the worst possible outcome for the decision maker. The decision maker needs to obtain a solution, based only on the admissible set of scenarios. We will aggregatively refer to the classes of optimization problems with stochastic models and robust models as *stochastic optimization* and *robust optimization*, the latter being the focus of this thesis.

The notion of robust optimization[1] subsumes a large body of literature in itself. In the following exposition we attempt to give a bird's-eye view of the field of robust optimization. Our focus will mostly, but not exclusively be on

[1] synonymously, *fault-tolerant optimization*

progress made in the last two decades in the realm of *robust combinatorial optimization*. One can further restrict the contribution of this thesis to this field.

The main object of interest in our exposition will be a general *linear combinatorial optimization problem*. We will henceforth drop the adjectives "linear", and "optimization", which can be sweepingly assumed throughout the rest of this thesis. In other words, this thesis deals exclusively with linear optimization problems, mostly of combinatorial nature. Despite the fact that we will stick to one blueprint of problems in the following discussion, analogous definition and formulations can be derived for much more general classes of problems.

An instance of our general combinatorial problem is given via a *combinatorial structure* C represented as a *set system* $C = (A, \mathcal{S})$. A set system consists of a finite set A of *elements* and a collection $\mathcal{S}$ of subsets of A, called *feasible sets*. A combinatorial problem $\mathcal{P}$ is a collection of pairs $P = (C, w)$, where C is a combinatorial structure taken from some domain of structures and $w : A \to \mathbb{Q}$ is a *cost function*. Given an instance P of $\mathcal{P}$ the goal is to solve the following minimization problem.

$\mathcal{P}$: Given an instance $P = (C = (A, S), w)$ of $\mathcal{P}$ solve

$$\min_{X \in \mathcal{S}} w(X), \tag{1.1}$$

with $w(X) = \sum_{a \in X} w(a)$.

A *solution* of an instance P is any feasible set X attaining the minimum in Equation (1.1). It is evident that our definition of a combinatorial problem is indeed general, in that it encompasses essentially every finite combinatorial optimization problem. For example, consider the *shortest path* problem in graphs, which asks to find a simple path of minimal length connecting two nodes s and t in a graph G. We can cast the shortest path problem into our setting by letting A correspond to the set of edges in G, and $\mathcal{S}$ to the set of simple s-t paths in G. In the following sections we review some of the existing models for robust combinatorial optimization by deriving the appropriate *robust counterparts* from our general combinatorial problem. We discuss strengths and weaknesses of each model and give some applications. To this end we point out a distinction corresponding to the source of uncertainty, which is in the core of this thesis. The different robust models can be

coarsely divided into *cost robust* models and *structural robust* models. Cost robust models incorporate the uncertainty underlying the input data into the cost function. In other words, the cost of the solution chosen by the decision maker can vary from one scenario to another. At the same time, cost robust models assume that the feasible set is completely certain. In contrast, in the structural robust models the source of uncertainty is the feasible set. The decision maker needs to choose a certain "robust solution" that is either feasible in every scenario, or one which can be made feasible after the scenario is revealed, with a small further intervention called a *recourse action*. This thesis deals with models of the latter type.

The sections which follow give a broad picture of the field of robust cominatorial optimization. The topics treated by this thesis are introduced in Section 1.2.3. Our exposition takes a historical perspective, trying to follow the chronological order in which the field was developed.

We conclude this section with a remark on the representation of the input data P. Although the question of representation of P is crucial in any discussion about efficiency of algorithms and complexity, for the general discussion that follows we omit such details. The appropriate details are filled in later on, when concrete computational problems are presented.

1.2 Robust combinatorial optimization

1.2.1 Cost robust combinatorial optimization

Cost robust combinatorial optimization assumes that the underlying set of resources is not subject to uncertainty. Instead, the *cost structure* is subject to uncertainty. We start by considering the most general case of a robust counterpart of problem (1.1) in this setting. Recall that an instance of our general combinatorial problem $\mathcal{P}$ is given by the pair $P = (C, w)$, where $C = (A, \mathcal{S})$ is a set system and w is a linear function on the ground set A. In this robust model we can omit the cost function w, as the cost structure of the resources is given in the scenario set. Let $\mathbf{W}$ denote the set of all rational functions on the set A, namely

$$\mathbf{W} = \mathbb{Q}_\infty^A, \tag{1.2}$$

where $\mathbb{Q}_\infty = \mathbb{Q} \cup \{\infty, -\infty\}$, is the extension of the rational numbers that contains the infinite values. Then the *cost robust counterpart (CRC)* of P is the following problem.

CRC($\mathcal{P}$): Given an instance P and a family of linear maps $\mathcal{W} \subset \mathbf{W}$, find $X^* \in \mathcal{S}$ attaining

$$\min_{X \in \mathcal{S}} \max_{w \in \mathcal{W}} w(X).$$

For an instance P of $\mathcal{P}$ we write CRC(P) to denote the collection of instance of CRC($\mathcal{P}$) with P as the instance of the underlying static problem (and some $\mathcal{W}$). Note that the nominal cost function w_0 does not play a role in the definition of CRC($\mathcal{P}$). For a demonstration of this definition we return to the shortest path problem (SP).

Example 1.2.1. *Let $G = (V, E)$ be a graph and let $w_1, w_2 : E \to \mathbb{Q}$ be two weight functions on the set of edges. We set $\mathcal{W} = \{w_1, w_2\}$ and consider the corresponding instance of CRC(SP). The corresponding instance P of SP has C as the set system with ground set $A = E$, and $\mathcal{S}$ being the set of all s-t paths in G, for some fixed $s, t \in V$. CRC(P) is hence to find a path $X \in \mathcal{S}$, which minimizes*

$$\max\{w_1(X), w_2(X)\}. \tag{1.3}$$

Figure 1.1 provides a concrete example. The paths $s-a-b-t$ and $s-c-d-t$ have lengths 4 and 5 according to w_1 and w_2, respectively, and are shortest for their respective weight functions. The optimal solution to CRC(SP) is the path $s-a-d-t$, with a worst case cost of 6.

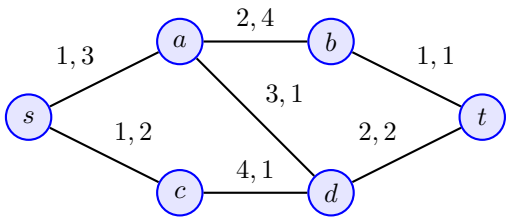

Figure 1.1: An instance of CRC(SP). The label of every edge e should be read as $w_1(e), w_2(e)$.

To this end the question of data representation arises again. In particular, the way that $\mathcal{W}$ is represented in the input is crucial from the computational point of view. In Example 1.2.1, $\mathcal{W}$ consisted of two costs functions only. It is

hence reasonable to assume an explicit representation of each cost function, e.g. in the form of lists specifying the output of the functions for every edge. This brings us to one of the first robust models studied in the field of combinatorial optimization. We denote this model by *explicit cost robust counterpart (ECRC)*, formally stated as follows.

ECRC($\mathcal{P}$): Given an instance P and a finite family of k linear maps $\{w_1, \cdots, w_k\} \subset \mathbf{W}$, explicitly listed in the input, find $X^* \in \mathcal{S}$ attaining

$$\min_{X \in \mathcal{S}} \max_{i=1}^{k} w_i(X).$$

The instance in Example 1.2.1 can hence be seen as an instance of ECRC(SP). The study of ECRC(SP) dates back to the paper of Yu and Yang [67] and is one of the first robust combinatorial problems studied. The authors show that ECRC(SP) is weakly NP-hard even for layered graphs and only 2 scenarios. This immediate increase in complexity from the tractable shortest path problem to a very restricted class of ECRC(SP) is a feature of many robust counterparts of combinatorial problems, as we will see in several other examples. In fact, when the number of scenarios is not bounded by a fixed constant, ECRC(SP) is strongly NP-hard. On the positive side, Yu and Yang give two pseudo-polynomial algorithms for the case of a bounded number of scenarios. For the case of two scenarios this result is also implied by the fully polynomial time approximation schemes (FPTAS) available for *bi objective shortest path (BOSP)* (see Warburton[66] and Hassin [45]). The book of Kouvelis and Yu [54] and the recent survey of Aissi, Bazgen and Vanderpooten [8] provide many more results on ECRCs of combinatorial problems. We conclude our discussion about ECRC with a brief review of the known results on ECRC variants of classical combinatorial problems. Kouvelis and Yu [54] show that every instance of ECRC(P) of a polynomial problem P can be approximated within a factor k by taking the optimal solution to the nominal problem P with the *median* cost function $w' = \frac{1}{k}\sum_{i=1}^{k} w_i$. For the case of a bounded number of scenarios NP-hardness was proved for the spanning tree, assignment and knapsack problems by Kouvelis and Yu [54], and for the min s-t cut problem by Aissi, Bazgan and Vanderpooten [5]. Armon and Zwick [10] showed that the robust counterpart of the minimum cut problem is solvable in polynomial time for a bounded number of scenarios. All aforementioned problems become strongly NP-hard when the number of scenarios is not bounded. Aissi, Bazgan and Vanderpooten [7] show that SP, the spanning tree problem, and the knapsack problems admit FPTASs for the case of a bounded number

of scenarios. For further references on concrete problems we refer to Aissi, Bazgen and Vanderpooten [8] and references therein. This paper also states some open problems related to explicit cost robust counterparts.

We turn our attention to a different model of a robust counterpart, which differs from ECRC in the way that the scenario set $\mathcal{W}$ is represented. Unlike the explicit representation of the scenarios in ECRC, the new model specifies possible cost realizations of single elements by providing upper and lower bounds. In other words, every element $a \in A$ is associated with two numbers w_a^- and w_a^+ with $w_a^- \leq w_a^+$, specifying an interval. Let I denote the collection of these intervals. The set of scenarios is given by

$$\mathcal{W}_I = \{w \in \mathbf{W} : \forall a \in A \quad w_a^- \leq w(a) \leq w_a^+\}. \tag{1.4}$$

Put differently, any cost function, which respects the bounds given in the input for every element is a scenario in $\mathcal{W}_I$. This data representation model stems from numerous applications, in which nominal values for resource costs are available, and a certain variability from the nominal values is possible, but it is bounded by upper and lower bounds. This situation is suitable for modeling stock prices fluctuations, inaccuracies in experimental models etc. The simplest way to incorporate the interval data model into the cost robust framework would be to define the following problem denoted *interval cost robust counterpart (ICRC).*

ICRC($\mathcal{P}$): Given an instance P and intervals $w_a^- \leq w_a^+$ for every element $a \in A$, find $X^* \in \mathcal{S}$ attaining

$$\min_{X \in \mathcal{S}} \max_{w \in \mathcal{W}_I} w(X),$$

where $\mathcal{W}_I$ is given by Equation (1.4).

It is however obvious that for any solution $X \in \mathcal{S}$, the worst case cost realization is given by setting the costs of every element to its upper limit. It follows that solving ICRC is equivalent to solving the original problem with the latter cost function, which makes ICRC(P) computationally equivalent to the original problem P. The interval uncertainty model does reveal new challenges when the min-max objective function is replaced with the *min-max regret* objective function. We define next the notion of *regret.* For a cost function w, let $X_w^* = argmin_{X \in \mathcal{S}} w(X)$, and $OPT_w = w(X_w^*)$ be the corresponding optimal solution and the optimal cost, respectively. For a solution

$X \in \mathcal{S}$, the regret incurred in scenario w is

$$Reg(X, w) = w(X) - OPT_w, \tag{1.5}$$

namely the deviation of the solution value in scenario w from the optimal solution for this scenario. The *min-max regret interval cost robust counterpart (RICRC)* is defined as follows.

RICRC($\mathcal{P}$): Given an instance P and intervals $w_a^- \leq w_a^+$ for every element $a \in A$, find $X^* \in \mathcal{S}$ attaining

$$\min_{X \in \mathcal{S}} \max_{w \in \mathcal{W}_I} Reg(X, w),$$

where $\mathcal{W}_I$ is given by Equation (1.4).

Informally, the RICRC tries to find a solution, which minimizes the worst possible *deviation* form the scenario optimal outcome. Regret models situations, in which the decision maker's performance is evaluated in accordance to the best possible performance in the materialized scenario. For further motivating examples and applications see Aissi, Bazgan and Vanderpooten [8]. Although the number of scenarios in a typical instance of the class RICRC is infinite, a closer examination reveals that the number of scenarios can always be reduced to at most 2^m, where $m = |A|$. This finite set of scenarios consists of all cost functions w, which assign extremal values for each element a in its corresponding interval, namely the set

$$\{w \in \mathbf{W} : \forall a \in A \quad w(a) \in \{w_a^-, w_a^+\}\}. \tag{1.6}$$

Scenarios corresponding to the latter set are often called *extreme scenarios*. In fact, Aissi, Bazgan and Vanderpooten [8] show that for every solution $X \in \mathcal{S}$, the worst case regret is always attained with at least one of the cost functions from the set in Equation (1.6). The following example illustrates the RICRC model with the shortest path problem.

Example 1.2.2. *Consider the graph in Figure 1.2. The only three edges for which the cost is uncertain are ab, ad and cd. This gives rise to 8 extreme scenarios, corresponding to all combinations of extreme values for the costs of the uncertain edges. The path $p_1 = a - b - c - d$ is a solution with maximal regret 2. This regret is attained for the scenario, in which $w(ab) = 4, w(ad) = 1$ and $w(cd) = 2$. The corresponding scenario-optimal solution is $p_2 = s -$*

$a - d - t$ with cost 4, while the cost of p_1 in this scenario is 6. p_2 is the optimal solution to this instance of RICRC(SP). The worst case regret for p_2 is attained in the scenario with $w(ab) = 3, w(ad) = 3$ and $w(cd) = 2$. The minimal maximal regret is 1, as the cost of p_2 in this worst-case scenario is 6, while the scenario-optimal solution, p_1, has cost 5.

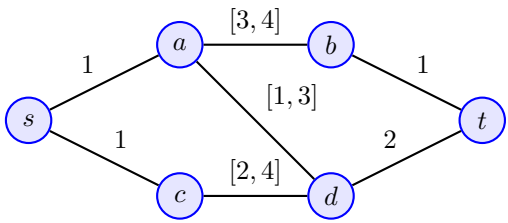

Figure 1.2: An instance of RICRC(SP). The label of every edge e shows the interval $[w_e^-, w_e^+]$ in case that the edge's cost is subject to uncertainty, or its nominal cost, otherwise.

A brief review of results on the RICRC model is as follows. Strong NP-hardness was proved for the shortest path and spanning tree problems by Averbakh and Lebedev [13], and for the assignment and min s-t cut problems by Aissi, Bazgan and Vanderpooten [6, 5]. Independently of the result in [13], Zieliński [68] showed that the robust shortest path problem is NP-hard. We note that the two papers prove that the result still holds for different restrictions of the problem to special graph classes. In [5] the authors show that the min cut problem admits a polynomial time algorithm.

We conclude our exposition of regret models by presenting the *min-max regret explicit cost robust counterpart (RECRC)* defined as follows.

RECRC($\mathcal{P}$): Given an instance P and k cost functions $w_1, \cdots, w_k$, find $X^* \in \mathcal{S}$ attaining

$$\min_{X \in \mathcal{S}} \max_{i=1}^{k} Reg(X, w_i).$$

We provide a brief review of the known results for RECRC. Consider first the case of bounded k. NP-hardness was proved for the spanning tree, the assignment and the knapsack problems by Kouvelis and Yu [54], for the shortest

path problem by Yu and Yang [67], and for the min s-t cut problem by Aissi Bazgan and Vanderpooten [5]. Aissi Bazgan and Vanderpooten [7] show that the shortest path and the spanning tree problems admit a FPTAS. The same paper shows that the knapsack problem is not approximable in this robustness model.

Consider next the case of unbounded k. All combinatorial problems discussed so far are strongly NP-hard in this robustness model (see Kouvelis and Yu [54] for the shortest path problem, Aissi, Bazgan and Vanderpooten [7] for the spanning tree and knapsack problems, Aissi, Bazgan and Vanderpooten [6] for the assignment problem and Aissi, Bazgan and Vanderpooten [5] for the min cut and min s-t cut problems). Results on hardness of approximation were given for the shortest path and spanning tree problems by Kasperski and Zielinski [49].

This concludes our discussion on cost robust models for combinatorial optimization. As we mentioned before, the focus of this thesis is on uncertainty underlying the feasible set of solutions, rather than the cost structure. We note that some of the models discussed in the sequel, and addressed by this thesis can be cast in one of the four models discussed here. For example, setting the cost of some element $a \in A$ to infinity in a scenario, effectively means that this element cannot be used by any solution for this scenario (otherwise we obtain the objective function value ∞). In other words, we can model *failed* resources using, for example, the model ECRC. This implies that some of the algorithmic results available for the models described in this section can be directly applied for the other models as well. However, in doing so we can rarely obtain the best possible algorithmic results. We will elaborate on this remark later on.

1.2.2 Demand robust combinatorial optimization

The robust model presented here is the first of several models mentioned in this thesis, which fall into the category of *two-stage optimization*. In two-stage optimization the decision making process is divided into two *stages*. In the first stage, a solution is constructed based only on the set of possible realizations of the input data, namely on the set scenarios. The second stage solution is obtained *after* the materialized scenario was observed by the decision maker. In other words, the second stage solution is computed without the presence of uncertainty. The second stage solution is called *recourse action*.

Two-stage optimization is often useful to avoid the over-conservativeness of the single-stage robust approach. In fact, in many applications it is reasonable

to assume that the decision maker is allowed to act, albeit in a limited way, after observing the final state of the system. This further intervention in the first stage solution often comes at a higher price per-resource, or it is limited by a certain budget. In this section we describe a model which allows unlimited recourse.

Dhamdhere, Goyal, Ravi and Singh [32] introduced a new model for robust combinatorial optimization, called *demand robustness*. Demand robustness models a situation, in which the decision maker is uncertain about the operational objective (or *demand*) that his solution should satisfy in the phase of solution implementation. We demonstrate this concept with the shortest path problem. Consider a situation, in which we need to connect two uncertain nodes in a graph $G = (V, E)$. The input data specifies two nonnegative cost functions $w_0, w_r \in \mathbf{W}$, with $w_0(e) \leq w_r(e)$ for every $e \in E$. In the first stage, we are required to specify a certain collection of edges $X^0 \subset E$. At the second stage a scenario materializes. The scenario corresponds to a pair of vertices $s, t \in V$, which need to be connected. If s and t are in different connected components of the graph $G^0 = (V, X^0)$, we need to find an additional set of edges $X^r \subset E$, such that s and t are connected in $G^r = (V, X^0 \cup X^r)$. The cost of the solution $X = X^0, X^r$ is calculated as $w(X) = w_0(X^0) + w_r(X^r)$. The goal is to minimize the worst case solution cost in this model, namely to solve

$$\min_{X^0} \max_{\substack{s,t \in V \times V \\ s \neq t}} \min_{\substack{X^r: \\ X^0 \cup X^r \text{connects } s \text{ and } t}} \left(w_0(X^0) + w_r(X^r) \right). \tag{1.7}$$

Typically, an algorithm for a demand robust counterpart is required to specify both the first stage solution X^0 and the recourse sets X^r for every scenario. In contrast, Problem (1.7) is an optimization problem over the first stage solution only. The recourse sets are assumed to be chosen optimally. We remark on the difference between the two variants later.

Note that the input data can also specify a certain subset of admissible scenarios $\Omega \subset V \times V$. In this case the maximization in Problem (1.7) is only over pairs in Ω. To define the general demand robust model we need the following definition.

Definition 1.2.3. A combinatorial problem $\mathcal{P}$ is *monotonic* if for every instance P of $\mathcal{P}$, its feasible set $\mathcal{S}$ satisfies that $X \in \mathcal{S}$ and $Y \supset X$ imply $Y \in \mathcal{S}$ for every $X, Y \subset A$.

Monotonic combinatorial problems are often called *covering problems*. These problems posses useful structural properties, which make certain solution techniques applicable. In fact, most computational problems addressed by

this thesis are robust counterparts of monotonic problems. The full extent of the gain we get from dealing with monotonic problems is revealed throughout this thesis.

The general demand robust model is denoted *explicit demand robust counterpart (EDRC)* and defined in the following way. Let $\Omega = \{s_1, \cdots, s_k\}$ be a set of scenarios. Each scenario s_i is associated with an instance P_i of a monotonic combinatorial problem $\mathcal{P}$, and a second stage cost function w_r^i, which satisfies $w_r^i(a) \geq w_0(a)$ for every $a \in A$. The ground set of elements of all problems P_i is the same set A. EDRC($\mathcal{P}$) is given by

EDRC($\mathcal{P}$): Given a set of scenarios $\Omega = \{s_1, \cdots, s_k\}$, each specifying an instance P_i of $\mathcal{P}$ and a second stage cost function $w_r^i \in \mathbf{W}$, and first stage cost function $w_0 \in \mathbf{W}$, such that $w_r^i(a) \geq w_0(a)$ for every $i \in [k]$ and $a \in A$, find $X^0 \subset A$ and $X_i^r \subset A$ for every $s_i \in \Omega$ attaining

$$\min_{\substack{X^0,\, X_1^r, \cdots, X_k^r: \\ \forall i \in [k]\ X^0 \cup X_i^r \in S_i}} \max_{s_i \in \Omega} \left(w_0(X^0) + w_r^i(X_i^r) \right),$$

where $\mathcal{S}_i$ is the feasible set of the instance P_i.

We interpret the latter definition in the context of the problem EDRC(SP), defined beforehand. The monotonic combinatorial problem that corresponds to SP has as feasible set $\mathcal{S}$ all subsets of edges, which connect some given pair of vertices s and t. The inclusion-wise minimal such sets are exactly the s-t paths, which in turn constituted the entire feasible set $\mathcal{S}$ for SP in our previous discussions. The new extended feasible set is clearly monotonic, as adding edges to a set, which connects s and t cannot disconnect s and t. Note that the original feasible set, which contained only the s-t paths is not monotonic. This form of extension of a combinatorial problem to obtain a monotonic combinatorial problem is always possible and well-defined. Also note that EDRC is defined with scenario-specific second stage costs w_r^i, while in EDRC(SP) defined before, the second stage cost w_r in the same for all scenarios.

As remarked before, one distinction between the problem in Equation (1.7) and the corresponding problem EDRC(SP), derived from our general definition, is that in the former the optimization is performed only over the first

stage solution X^0. The reason for this is that we can simply assume that the second stage solution is always computed optimally in the case of the shortest path problem. Indeed, given a first stage solution X^0 and a scenario $\{s,t\}$, finding the optimal recourse set $X^r_{s,t}$ amounts to solving an ordinary shortest path problem. In general, the computational problem which comprises the computation of the recourse set can be an NP-hard problem. In the latter case, the variant, which requires optimizing over both X^0 and the recourse sets X^r_i should be used.

We illustrate the class EDRC further by the following numerical example, corresponding to the EDRC(SP) problem.

Example 1.2.4. *Consider the graph in Figure 1.3. The set of admissible scenarios Ω contains the two pairs $\{s,t\}$ and $\{b,c\}$. The cost function w_0 assigns $w_0(e) = 1$ to all edges in the graph. The values of w_r are given in the figure (in this example, the second stage costs are the same for both scenarios). Note that we have a simple upper bound of 4 on the worst case cost, since one can choose the first stage solution $X^0 = \{sa, sc, ab, bt\}$, which contains both a s-t path and a b-c path. A better solution is obtained by choosing $X^0 = \{ad\}$. The corresponding recourse sets are $X^r_{s,t} = \{sa, dt\}$ and $X^r_{b,c} = \{ad, cd\}$. The worst case cost of this solution is $w_0(X^0) + w_r(X^r_{s,t}) = 1 + 2.6 = 3.6$.*

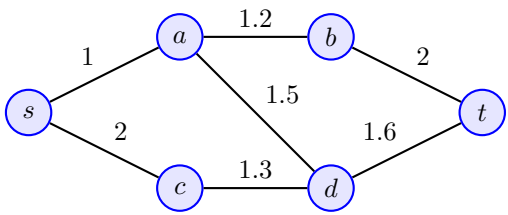

Figure 1.3: An instance of EDRC(SP). The label of every edge e shows the second stage cost $w^r(e)$.

To this end we would like to emphasize the nature of uncertainty in this problem. Unlike cost robust models, the costs of the resources in the demand robust model are certain. The uncertain element is the feasible set. For example, in EDRC(SP), a certain subset X of edges is feasible in exactly those scenarios, which correspond to pairs of vertices that lie in the same component of the graph (V, X). Put differently, X is a feasible solution for some scenarios, and an infeasible solution for other scenarios. This makes it the first *structural robust* model described so far. We conclude by reviewing

the work related to the demand robust model. To avoid confusion we provide next a list of combinatorial problems treated in the literature with a short description of the corresponding demand robust problems.

- *Min cut (MC).* A single terminal $r \in V$ is fixed. Each scenario s_i is associated with another terminal t_i, and the goal is to disconnect r and t_i.
- *Min multi-cut (MMC).* Each scenario s_i is associated with d terminal pairs (r_i^l, t_i^l), for $l \in [d]$, and the goal is to disconnect all pairs of terminals.
- *Steiner tree (STT).* Each scenario s_i is associated with a subset $T_i \subset V$ of the terminals, which need to be connected in the solution.
- *Steiner forest (STF).* Each scenario s_i is associated with a collection of terminal pairs (r_i^l, t_i^l), for $l \in [d]$, which need to be connected in the solution.
- *Set cover (SC).* Each scenario s_i is associated with a subset S_i of the ground set elements, which need to be covered. The *Vertex cover (VC)* and *Edge cover (EC)* counterparts are defined analogously.
- *Uncapacitated facility location (UFL).* The set of facilities F is fixed. Each scenario s_i is associated with a set of clients $S_i \subset V \setminus F$, which need to be served.

In all works on demand robust counterparts so far the second stage costs w_r^i is defined with respect to the first stage cost w_0 by letting $w_r^i = \lambda_i w_0$. Here $\lambda_i \geq 1$ is called an *inflation factor*, and it corresponds to the factor by which every resource becomes more expensive in the second stage in scenario i (in our definition of EDRC, the second stage costs can be arbitrary).

In the first paper on demand robust combinatorial optimization Dhamdhere, Goyal, Ravi and Singh [32] developed approximation algorithms for several problems. Concretely, they gave polynomial algorithms for EDRC(MC), EDRC(MMC), EDRC(ST), EDRC(VC) and EDRC(UFC) with approximation guarantees $O(\log k)$, $O(\log dk \log\log dk)$, 30, 4 and 5, respectively. Their results for EDRC(MC) was improved to a $(1+\sqrt{2})$-approximation by Golovin, Goyal and Ravi [41]. The same paper also gives an algorithm with an approximation guarantee of 7.1 for EDRC(SP), which improves on a result implied for this problem in [32].

Feige, Jain, Mahdian and Mirrokni [36] later extended the model to allow exponential sets of scenarios, defined implicitly with an upper bound on the number of active clients. Let us denote this variant by *implicit demand robust counterpart (IDRC)*. In this setting the second stage solutions are not computed explicitly, but rather obtained in the second stage using an approximation algorithm. The authors develop a framework for a class of problems. In particular, they give an $O(\log n \log m)$-approximation algorithm for IDRC(SC) and constant factor approximation algorithms for IDRC(VC) and IDRC(EC). A simple algorithm with an improved approximation guarantee is presented for the case that $w_0(a) = 1$ for all resources $a \in A$ and a uniform inflation factor λ across all scenarios. Khandekar, Kortsarz, Mirrokni and Salavatipour [50] studied the problems IDRC(ST), IDRC(SF) and IDRC(UFL), providing constant factor approximation algorithms for IDRC(ST) and IDRC(UFL), and a logarithmic factor approximation algorithm for the special case of IDRC(SF) with a uniform inflation factor. The authors also show hardness-of-approximation results for IDRC(SF) and IDRC(MC).

This concludes our discussion on demand robust combinatorial optimization. We remark that while robust combinatorial optimization is a relatively new field, demand robust combinatorial optimization is newer still, hence a lot of interesting questions remain open. For a thorough treatment of demand robust combinatorial optimization in the explicit scenario model and its extensions we refer to the dissertation of Goyal [42].

1.2.3 Structural robust combinatorial optimization

This section provides the primer for the topics directly addressed by this thesis. Put differently, this thesis deals with structural robust counterparts of combinatorial optimization problems. Most models in this section are new, but at the same time, they can be viewed as a natural extension of existing models. We start by listing a number of key characteristics of our structural robust models. Those characteristics are in the heart of the analysis of our algorithms.

- Structural robust counterparts problems are *covering problems*. In other other words, they correspond to monotonic combinatorial problems (see Definition 1.2.3).
- Structural robust counterparts deal with *failure of resources*.

- Structural robust counterparts have a *certain cost structure.*

We define both single-stage and two-stage structural robust counterparts. Let us start with the most basic single-stage model. Consider a monotonic combinatorial problem $\mathcal{P}$. The input to the robust problem specifies a nominal instance $P = (C, w)$ of $\mathcal{P}$ and a set of scenarios $\Omega = \{F_1, \cdots, F_k\}$. The i'th scenario corresponds to a set of resources $F_i \subset A$, which *fail*, and cannot be used in this scenario. The goal is to find a set of elements, which will comprise a feasible set to the nominal problem, *after removing the failed elements* F_i, for every $i \in [k]$. The resulting class of problems is denoted the *explicit robust covering counterpart of* $\mathcal{P}$ *(ERCC(*$\mathcal{P}$*))* and is formally defined as follows.

ERCC($\mathcal{P}$): Given an instance $P = (C = (A, S), w)$ and a set of explicit scenarios $\Omega = \{F_1, \cdots, F_k\}$, with $F_i \subset A$ for every $i \in [k]$, find X^* attaining

$$\min_{\substack{X \subset A: \\ \forall i \in [k]\ X \setminus F_i \in \mathcal{S}}} w(X).$$

We make some simple observations about ERCCs. A first simple observation relates to the complexity of deciding feasibility of an instance of ERCC($\mathcal{P}$). Note that monotonicity of $\mathcal{P}$ guarantees that an instance (P, Ω) for ERCC($\mathcal{P}$) is feasible, if and only if $X = A$ is a feasible solution. Indeed, if a feasible solution $X' \subset A$ exists, then adding all the elements in $A \setminus X'$ to it maintains the feasibility of the solution. It follows that the complexity of deciding feasibility of ERCC($\mathcal{P}$) is essentially the same as deciding feasibility for $\mathcal{P}$: to decide if the instance (P, Ω) is feasible, simply use an algorithm for deciding feasibility for $\mathcal{P}$ $k = |\Omega|$ times - once for every set of the $A \setminus F_i$. Decide that the instance to ERCC($\mathcal{P}$) is feasible if and only if $A \setminus F_i \in \mathcal{S}$ for *every* $i \in [k]$.

Another observation is related to the approximability of ERCC($\mathcal{P}$). Consider any monotonic combinatorial problem $\mathcal{P}$, which admits a polynomial β-approximation. Then it is easy to obtain a polynomial $k\beta$-approximation algorithm for ERCC($\mathcal{P}$). For an instance $(P = (C, w), \Omega)$ of ERCC($\mathcal{P}$), the algorithm defines k instances of $\mathcal{P}$ by letting the feasible set of the i'th problem be

$$C_i = (A \setminus F_i, \{X \subset A \setminus F_i : X \in \mathcal{S}\}), \tag{1.8}$$

and obtains β-approximate solutions Y_i^* for each instance using the algorithm for $\mathcal{P}$. Finally, the algorithm returns the solution $X = \cup_{i=1}^k Y_i^*$ as the solution to ERCC($\mathcal{P}$). X is clearly a feasible solution, since for any scenario $F_i \in \Omega$, we have $Y_i^* \subset X \setminus F_i$, and $Y_i^* \in \mathcal{S}$. The approximation guarantee follows from the fact that optimal solution value of the robust instance is an upper bound of every one of the k problems defined by the algorithm.

The latter two properties are a simple consequence of monotonicity of the combinatorial problem $\mathcal{P}$. We will see later that this property of $\mathcal{P}$ lets us achieve much better algorithmic results. In fact, under reasonable complexity assumptions, we will be able to obtain optimal approximation algorithms for some problems. To obtain a better feeling of ERCCs we demonstrate this class with our usual example, the shortest path problem.

Example 1.2.5. *Consider the following instance of ERCC(SP). The graph $G = (V, E)$ and the two terminal s and t are given in Figure 1.4. The scenario set Ω contains two failure sets $F_1 = \{cd, bt\}$ and $F_2 = \{sa, ad, bd\}$. We illustrate the two scenarios in the figure by displaying the graphs $G_i = (V, E \setminus F_i)$ for $i = 1, 2$. The inclusion-wise minimal feasible solutions to this instance are the subgraphs $X_1, \cdots, X_6$ in the figure. The optimal solution for the cost function $w \equiv 1$ are X_1 and X_2 with optimal cost 5. If we change the cost of ad to $w(ad) = 3$, these solutions become suboptimal with cost 7. The new optimal solutions are X_3, X_4 and X_5 with optimal cost 6.*

In this thesis we study robust ERCCs of several classical combinatorial problems including SP, bipartite matching (BM), ST, matroid linear optimization (MO) and sparsest k-spanner (SkS). In some cases we are able to give tight upper and lower bounds for the approximability of these problems under standard complexity-theoretic assumptions. In some cases we will study fixed-parameter versions of this problem, which are defined as follows. For a combinatorial problem $\mathcal{P}$, the robust counterpart BERCC($\mathcal{P}$) is defined as the problem ERCC($\mathcal{P}$), restricted to instances (P, Ω), satisfying $|F| \leq B$ for every $F \in \Omega$. In other words the class BERCC is the same as ERCC, but restricted to instances in which each failure scenario F contains at most B elements. For example, the instance in Example 1.2.5 is an instance of BERCC(SP) with $B = 3$.

We move on to the next class of problems, which is closely related to ERCC. The only difference between the two classes is the representation of the feasible set. While in ERCC the scenarios are listed explicitly in the input, in the new class the set of scenarios is given by an upper bound on the number of possible failures in the structure. Formally, the input specifies an instance P

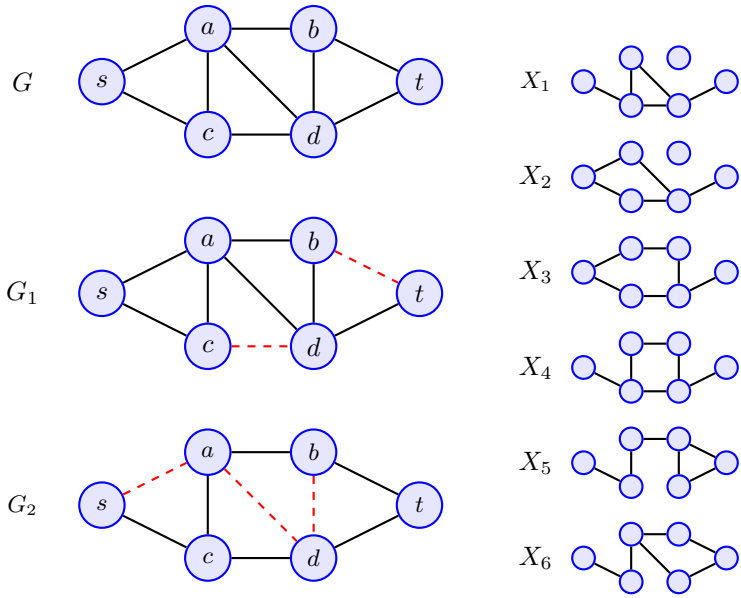

Figure 1.4: An instance of ERCC(SP). The graphs G_1 and G_2 illustrate the two scenarios. The inclusion-wise non-dominated solutions are $X_1, \cdots, X_6$.

of a monotonic combinatorial problem $\mathcal{P}$, as well as an integer $1 \leq k \leq |A|$, called *the adversarial budget*. The set of scenarios $\Omega(P, k)$ is defined as

$$\Omega(P, k) = \{F \subset A : |F| \leq k\}. \tag{1.9}$$

The goal is to solve the *implicit robust covering counterpart of* $\mathcal{P}$ *(IRCC(*$\mathcal{P}$*))* defined as follows.

IRCC($\mathcal{P}$): Given an instance $P = (C, w)$ and an integer $1 \leq k \leq m$, find X^* attaining

$$\min_{\substack{X \subset A: \\ \forall F \in \Omega(P,k)\ X \setminus F \in \mathcal{S}}} w(X),$$

where $\Omega(P, k)$ is given in Equation (1.9).

The scenario set $\Omega(P, k)$ obviously contains $\binom{m}{k}$ scenarios, were $m = |A|$, which is exponential in the encoding length of the input.

Among all robust models defined in this section, IRCC is the most studied one to date. Perhaps the two most notable examples of IRCCs are the *minimum k edge-disjoint paths (k-EDP)* problem and the *minimum k-edge-connected spanning subgraph (k-ECSS)* problem. Recall that the input in k-EDP is a weighted graph and two terminal vertices s, t, and the goal is to find k edge-disjoint paths from s to t with minimal total cost. In k-ECSS the input is a weighted graph and the goal is to find a set of edges admitting no cut of $r < k$ edges of minimal total cost.

It follows from the definitions that k-EDP is IRCC(SP), and k-ECSS is IRCC(ST), both with adversarial budget $k - 1$. While k-EDP is well- known to be polynomially solvable using network-flow techniques (see e.g. Schrijver [62]), k-ECSS is a strongly NP-hard problem (and in fact APX-hard). Gabow, Goemans, Tardos and Williamson [37] developed a polynomial time $(1+\frac{c}{k})$-approximation algorithm for the latter problem, for a fixed constant c. The authors also show that for some constant $c' < c$, the existence of a polynomial time $(1 + \frac{c'}{k})$-approximation algorithm implies P=NP. An intriguing property of k-ECSS is that the problem becomes easier to approximate when k grows. Concretely, while for every fixed k, k-ECSS is NP-hard to approximate within some factor $\alpha_k > 1$, the latter result asserts that there is function $\beta(k)$ tending to one as k tends to infinity such that k-ECSS is approximable within a factor $\beta(k)$. This phenomenon was already discovered by Cheriyan and Thurimella [26], who gave algorithms with a weaker approximation guarantee. The more general *generalized steiner network (GSN)* problem admits a polynomial 2-approximation algorithm due to Jain [46].

A seemingly small, but significant modification of the class IRCC is presented next. Instead of allowing every subset of k elements of A to fail, in the new model we restrict the failures to a smaller subset $U \subset A$ of *faulty elements*. The set U, which is supplied as an input parameter, is allowed to be any subset of the ground set of elements A. The motivation for this modification is that in many applications, the set of resources is divided into vulnerable and invulnerable resources. Consider for example a large facility that need to be operating even in case of sabotage by an enemy. While some parts of the facility have guards placed to guarantee the integrity of certain important resources, other parts of the facility might be left unguarded (e.g. due to shortage in personnel). It is hence reasonable to assume that only the subset of the resources in the facility, that is left unguarded, is prone to failure, while the guarded resources are guaranteed to operate in every scenario.

The resulting model is called *subset implicit recoverable robust counterpart of $\mathcal{P}$ (SIRCC($\mathcal{P}$))* and is formally defined as follows.

SIRCC($\mathcal{P}$): Given an instance $P = (C, w)$, an integer $1 \leq k \leq m$ and subset $U \subset A$ of the ground set, find X^* attaining

$$\min_{\substack{X \subset A: \\ \forall F \in \Omega(U,k)\ X \setminus F \in \mathcal{S}}} w(X),$$

where $\Omega(U, k) = \{F \subset U : |F| \leq k\}$.

The impact of this modification turns our to be significant. For example, we show in Chapter 3 that the SIRCC(SP) is NP-hard and in fact admits no polynomial time constant factor approximation algorithms, unless P=NP. This fact is in great contrast to the polynomiality of the problem IRCC(SP). Although this model seems to be a natural and simple extension of the extensively studied model IRCC, not much is known about it. In this thesis we study the SP problem in this robust setting.

The last robust model we present in this section is a two-stage model with implicitly given scenarios. In this model the set of possible failures is given as $\Omega(P, k)$, defined in Equation (1.9). The allowed recourse action is given by a *recovery budget* $r \in \mathbb{Z}_+$. The goal is to find a *feasible* first stage solution $X \in \mathcal{S}$ of minimal cost, such that for every scenario $F \in \Omega$, there exists a *recovery set* $R \subset A \setminus F$ such that $(X \setminus F) \cup R$ is a feasible solution. This model is called *adaptable robust covering counterpart of $\mathcal{P}$ (ARCC($\mathcal{P}$))*, and it is formally defined as follows.

ARCC($\mathcal{P}$): Given an instance $P = (C, w)$, and two integers $1 \leq k \leq m$ and $1 \leq r \leq m$, find X^* attaining

$$\min_{\substack{X \subset A: X \in \mathcal{S} \\ \forall F \in \Omega(P,k)\ \exists R \subset A \setminus F,\ |R| \leq |r|\ (X \setminus F) \cup R \in \mathcal{S}}} w(X),$$

where $\Omega(P, k)$ is given by Equation (1.9).

Note that unlike the unlimited recourse demand robust models presented in Section 1.2.2, the recourse model in ARCC allows only *limited recourse*. In other words, it is not true, that any feasible solution $X \in \mathcal{S}$ chosen in the first stage can be recovered in any failure scenario. Instead, a feasible solution X to an instance of ARCC($\mathcal{P}$) is guaranteed to be 'close to feasible', even after any k elements were removed from it. Another important difference is that recovery action does not come at an extra cost, namely the optimization is only over the cost of the first stage solution.

ARCC has potential applications in certain situations. Consider, for example, a network design problem, in which the continuous operation of certain structure in the network needs to be guaranteed. Failures of network resources happen rarely, hence taking a robust approach such as the IRCC model is over-conservative and prohibitively expensive. In the event of a failure, it is possible to immediately deploy at most $r \in \mathbb{Z}_+$ new network resources to recover the operational capabilities of the network structure. The budget r is due to limited stand-by personnel available to quickly act at the event of a failure. A concrete example of an application comes from the field of electricity grids operation. Guarisco, Friedrich, Balderer, Laumanns and Zdrallek [43] propose a model that addresses the need to balance between the quality of service, representing the continuous availability of power in the grid, and the cost of maintaining qualified personnel, that is available at the grid operator's disposal in the event of failures.

We emphasize that failure sets can contain elements, which are not contained in the first stage solution X^*. This assumption is justified in most applications, and in fact, in many situations the optimal strategy for the adversary is to fail both resources acquired in the first stage solution in order to break its feasibility, and to fail resources not acquired in the first stage solution, but which correspond to a cheap recourse action.

In this thesis we study the ARCC(SP) and ARCC(ST) problems. We conclude this section with an example of a concrete instance of ARCC(SP).

Example 1.2.6. *Consider the graph $G = (V, E)$ in Figure 1.5. The goal is to find a robust path connecting s and t. The s-t connectivity of G is 2, hence for $k > 1$ and any r the robust instance is infeasible. The graph is unweighted ($w \equiv 1$). Consider first the instance with $k = 1$ and $r = 2$. The optimal solution $X^* = \{sc, ce, eh, ht\}$ is displayed in the figure. Also displayed are the recovery sets $\{sa, ac\}$, $\{cf, fh\}$ and $\{hg, gt\}$, which correspond to the failure scenarios sc, ce or eh and ht, respectively. Indeed, for every failure scenario, the corresponding recourse set contains at most 2 edges, and the resulting graph contains an s-t path. It is a coincidence that X^* is also the shortest*

s-t path in G. Consider next the instance with $k = 1$ and $r = 1$. The optimal solution in this case is $X^* = \{sc, bc, ce, eh, fh, hg, ht\}$. the recovery edges sb, cf and gt correspond to the failure scenarios sc, ce or eh and ht, respectively. For both instances, all failure scenarios not listed above do not disconnect the s-t path that is contained in X^*, hence no recovery action is needed for them.

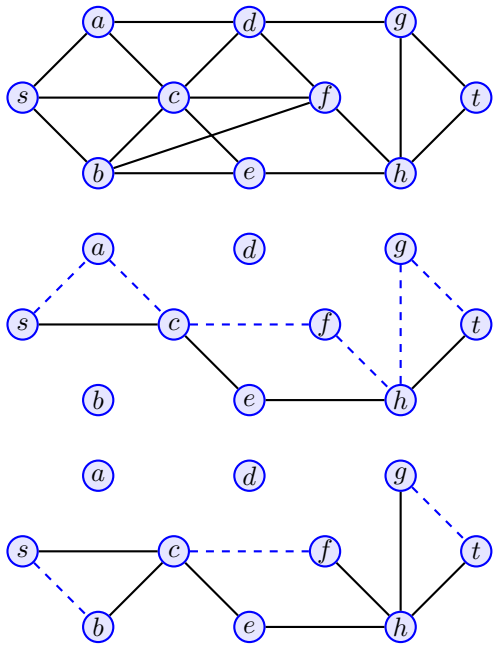

Figure 1.5: Two instances of ARCC(SP) with the same graph and terminals. Top: The input graph. Middle: The optimal solution and recourse sets of the instance with $k = 1$ and $r = 2$. Bottom: The optimal solution and recourse sets of the instance with $k = r = 1$.

1.2.4 Other robust combinatorial models

We conclude our introductory excursion through the realm of robust combinatorial optimization with a brief review of the work, which does not fall into any of the categories discussed so far.

Γ-scenarios. Bertsimas and Sim [20] developed a model of robust optimization for mixed-integer linear programming (MIP). The robustness model they

consider is as follows. Consider a MIP given by a $m \times n$ matrix B, a cost vector c, a RHS vector b and upper and lower bounds u, l.

$$\min\{c'x \,:\, Bx \leq b, \quad l \leq x \leq u, \quad x_i \in \mathbb{Z}\ \forall i \in [k]\} \tag{1.10}$$

The first k out of n variables are integral. Each entry b_{ij} in the input matrix B is modeled as a symmetric random variable with support $[b_{ij} - \bar{b}_{ij}, b_{ij} + \bar{b}_{ij}]$. Each coefficient c_i in the cost vector is allowed a variability in an interval $[c_i, c_i + d_i]$, defined by another set of parameters d_i. The parameters $\bar{b}_{ij}$ and d_i are nonnegative. The input to the robust problem also supplies m robustness parameters $\Gamma_1, \cdots, \Gamma_m$, one for each of the m rows in the matrix B. The parameter Γ_i controls the number of coefficients in the i'th row of M, which might deviate from their nominal values. The values Γ_i are not restricted to be integral. If Γ_i is fractional, then it is assumed that $\lfloor \Gamma_i \rfloor$ coefficients change in an arbitrary fashion within the corresponding intervals, and one more coefficient a_{it} changes by at most $(\Gamma_i - \lfloor \Gamma_i \rfloor)\bar{b}_{it}$. Finally, an integer Γ_0 is also supplied to control the variability in the cost vector. Analogously to the way the matrix coefficients change, the number of cost coefficients c_j, which are allowed to take values different from the nominal ones is Γ_0. The goal is to find an optimal solution, which will be *deterministically* feasible in every scenario, which complies with the input parameters, and to optimize the worst case cost.

The authors show that it is possible to transform the robust problem into a moderately larger MIP. They also proved that if more than the postulated number Γ_i of coefficients change in any row of B, the solution still remains feasible with high probability. For the case of combinatorial optimization, the authors consider the case that $\Gamma_i = 0$ for all rows of the constraint matrix and the uncertainty only lies in the objective function. It is shown that the robust counterpart of any polynomial combinatorial problem can be solved by computing the optimal solution of only $m + 1$ nominal problems, where $m = |A|$ is the cardinality of the ground set. This implies that the robust counterpart of a polynomial problem is again a polynomial problem. A similar result was provided for problems admitting approximation algorithms. For the case of the minimum cost flow (MCF) problem, the authors show that the problem can be reduced to a significantly smaller number of flow problems, given that a small deviation ϵ from the optimal robust value is tolerated.

Note that it is possible to associate the framework of Bertsimas and Sim with cost robust models for combinatorial optimization. At the same time it must be noted that their work also applies for a larger class, including MIP, where not only the cost is allowed to change, but also the constraint matrix

coefficients. It is for this reason that we chose to exclude it from the discussion about cost robust optimization.

Robust spanners. A *r-spanner* in a graph $G = (V, E)$ is a subgraph $H = (V, E')$ (with $E' \subset E$), such that for every $u, v \in V$ the shortest path distance between u and v in H is at most r times the shortest path distance between u and v in G. The parameter r is called the *stretch* of the spanner. A r-spanner is *k-robust* if for every subset U of k vertices, the graph $H - U$ is a r-spanner of $G - U$ (here $G - U$ is the subgraph of G induced on the set of vertices $V \setminus U$). The goal is to find spanners with low *stretch* α, which are as sparse as possible. There exists a vast body of literature on efficient methods to construct sparse spanners (see e.g. Elkin and Peleg [34]). More recently, Chechik, Langberg, Peleg and Roditty [24] developed an algorithm for constructing k-robust spanners, which are moderately larger than the best known constructions for ordinary spanners. The authors also address the case of edge failures. A simpler construction was later found by Dinitz and Krauthgamer [33]. For certain geometric settings, results for robust spanners were developed earlier by Levcopoulos, Narasimhan and Smid [56] and Czumaj and Zhao [29].

Fault-tolerant uncapacitated facility location. The ordinary uncapacitated facility location (UFL) problem is defined as follows. The input specifies n facilities $f_1, \cdots, f_n$ and m clients $c_1, \cdots, c_m$. Each facility f_i is associated with an *opening cost* d_i, and each facility-client pair f_i, c_j is associated with a *service cost* w_{ij}. The goal is to open facilities and assign them to clients, so as to minimize the total cost (opening cost plus service cost), under the constraint that each client is assigned to at least one facility. In the *fault-tolerant uncapacitated facility location (FTUFL)* problem, the input is augmented with another parameter r_j for each client c_j, which corresponds to the number of facilities that need to serve c_j. UFL is hence FTUFL with $r_j = 1$ for every client c_j. FTUFL models the situation in which some facilities may fail and it is hence desirable to obtain a redundant solution, which assigns a number of facilities to each client. FTUFL was first studied by Jain and Vazirani [47], who gave a $\log(r_{max})$-approximation algorithm, with $r_{max} = \max_{i=1}^{m} r_i$. This result was subsequently improved in several papers. We refer to the paper of of Swamy and Shmoys [64] for references on this problem. This paper also gives a 2.076-approximation algorithm for FTUFL, which was the best know bound until the recent paper of Byrka, Srinivasan and Swamy [23], which gives a 1.725-approximation algorithm for the problem. Most works on FTUFL use LP-rounding techniques.

A different robust setup was defined by Chechik and Peleg [25]. Instead of

assigning the clients to a certain number r_j of facilities and paying for all corresponding service costs in advance, the authors consider concrete facility failures. In their model a certain parameter $\alpha > 0$ is provided, which controls the number of possible failures. When a facility fails, the clients that were served by this facility in the solution are served instead by the nearest open facility that did not fail. The goal is to minimize the total worst-case cost in this model. The authors develop combinatorial approximation algorithms with guarantees 6.5 and $1.5+7.5\alpha$ for the cases $\alpha = 1$ and $\alpha > 1$, respectively.

Robust shortest paths. Puhl [61] 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 $(1 - \alpha)w^S(e)$ for 'renting' this edge. If this edge is used in the second stage solution, an 'implementation' cost $\alpha w^S(e)$ 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 $(\alpha + \beta)w^S(e)$. In the second setting an approximation algorithm was provided. Büsing [22] 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.

Robust network flow and network design. Atamtürk and Zhang [12] study a network flow problem, in which the demand d_u of each vertex u in the graph is uncertain. The input specifies some uncertainty set $\mathcal{U}$, which comprises all possible realizations of demand vectors $\mathbf{d}$. A flow f in the network is feasible is for every demand realization $\mathbf{d} \in \mathcal{U}$, the net flow in every vertex u is at least $\mathbf{d}_u$. The arc set E of the graph is partitioned into a first stage decision set E_1 and second stage decision set E_2. A first stage decision, which corresponds to an assignment of flow values to the arcs in E_1, is feasible if for every demand realization $\mathbf{d} \in \mathcal{U}$, there exists an assignment of flow values to the second stage arcs E_2, such that the resulting flow is feasible. The authors characterize the set of feasible first stage solutions and show that the separation problem over it is an NP-hard problem in general. For some special cases they show that the separation problem can be solved in polynomial time. The authors also consider the network design problem, which treats the arc capacities as decision variables as well.

Recoverable robustness. A rather general notion of robustness in a two-stage discrete optimization setting, called *recoverable robustness*, was pro-

posed by Liebchen, Lubbecke, Mohring and Stiller [57] as a tractable framework to introduce robustness in the realm of railway optimization. This work proposed a general receipt for designing two-stage robust problems from any combinatorial optimization problem.

1.3 Other models of robust optimization

In this section we give a short account of other models of robust optimization. The study of linear optimization with set-based uncertainty dates back to the work on *generalized linear programming (GLP)* by Dantzig and Wolfe [30]. GLP is an extension of ordinary LP, which incorporates the columns of the constraint matrix into the set of decisions. Each column is associated with a polyhedral set, which comprises the feasible domain of realizations for this column. A solution to a GLP specifies not only the optimal control variables x^*, but also the constrain matrix B^*, such that each column of B^* is an element in the corresponding polyhedral uncertainty set, and x^* is a feasible solution for set of constraints defined by B^*. The authors show that GLPs can be transformed to LPs using a simple algorithm. For a thorough account of this technique we refer the reader to the book of Dantzig [31].

In contrast to Dantzig and Wolfe's model, which allows the decision maker more freedom than a standard LP, a model proposed by Soyster [63] poses stronger restrictions on the decision maker. Analogously to the model in [30], Soyster's model, denoted *inexact linear programming (IXLP)*, defines independent polyhedral uncertainty sets for columns of the constraint matrix of a LP. The goal is to find a solution x^* maximizing a certain objective function, which is feasible *for all* possible realizations of the constraint matrix. Soyster shows that an IXLP can be transformed into an ordinary LP with a constraint matrix, whose components can be easily computed from the smallest axis-parallel bounding boxes of the uncertainty sets.

Thuente [65] developed a duality theory for the models developed by Dantzig and Wolfe and Soyster, showing that GLP and IXLP are dual problems.

Robust optimization has a natural application in linear optimization, as was shown by Ben-Tal and Nemirovski [17]. The authors show that only mild deviations of the coefficients in the nominal data of a LP can render an optimal nominal solution both highly infeasible and highly suboptimal, making it effectively useless. The authors define a notion of *reliability*, which accounts for the likelihood of such undesirable behavior to occur in a LP. Two methods are proposed to immunize a solution for an 'unreliable' LP. The first is to use

Soyster's approach and convert the LP into a suitable IXLP. The second approach models the coefficients in the LP as random variables and asks for a solution, which satisfies all constraints with high probability. The paper displays a case study using problems from the NETLIB library.

Bertsimas and Sim [21] initially developed the robust model of Γ-scenarios for LPs. Their result shows that the robust counterpart of a LP can be transformed to an ordinary LP of a moderately larger size. The framework naturally extends to discrete optimization problems as well.

Ben-Tal, Boyd and Nemirovski [15] 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 with affine functions $X(\zeta)$ of the uncertain input vector ζ. The resulting formulation turns out to be both theoretically and practically tractable in many cases. This work succeeded several earlier works on robust convex optimization. We refer the reader to the book of Ben-Tal, El Ghaoui and Nemirovski [16] for a thorough treatment of this topic.

Finally, we refer the reader the recent survey on robust optimization and its applications by Bertsimas, Brown and Caramanis [19]. This paper provides a broad overview of the field with concrete motivating examples.

1.4 Contribution

This section summarizes the main contribution of this thesis. The content of this thesis is a compilation of results from three papers coauthored with Rico Zenklusen [2, 4, 3] and a paper coauthored with Sebastian Stiller and Rico Zenklusen [1].

The goal of the thesis is to study structural robust combinatorial optimization problems. The main focus is on the robust models ERCC and ARCC. Some results are also presented for the SIRCC model. The emphasis is put on developing efficient exact and approximate algorithms for robust counterparts of classical combinatorial problems. Our treatment of the different models is divided into models which assume *uniform* and *nonuniform* scenario sets. A structural robust model is uniform if the failure sets of the corresponding couterparts are given by a single integer k, defining an upper bound on the number of resources allowed in a single scenario. IRCC and ARCC are uniform robust models, while ERCC and SIRCC are nonuniform robust mod-

els. This presentation is motivated by similarities in the different methods underlying our results.

We present results for the ERCC model in Chapter 3. Our exposition in this chapter is organized around the solution methods, rather than the concrete problem at hand. A uniform hardness-of-approximation result is also provided for the ERCC of essentially any combinatorial problem (Theorem 3.1.4). This result motivates most of the techniques used for deriving approximation algorithms. In particular, we show that some simple algorithms obtain close to optimal approximation guarantees for several combinatorial problems, under standard complexity assumptions. The following is a concrete list of results presented for the ERCC model. To this end recall that k denotes the number of scenarios in the input to a ERCC of a combinatorial problem $\mathcal{P}$. We use n and m to denote the number of vertices and edges of a graph.

- We prove that there are no polynomial algorithms for ERCC($\mathcal{P}$) with approximation factor better than $\log k$ for essentially any combinatorial problem $\mathcal{P}$, assuming that NP $\not\subset$ DTIME($n^{\log \log n}$). This is done via a generic reduction from the set cover problem.

- We show that randomly rounding an appropriate LP relaxation of the robust counterpart of the ST problem gives an approximation algorithm with guarantee $O(\log n + \log k)$ (Theorem 3.3.5).

- We develop an algorithm based directly on the greedy algorithm for SC. We use this algorithm to obtain an essentially optimal approximation guarantee $\log n + \log k$ for robust S2S (Theorem 3.4.9), with respect to both parameters n and k. The algorithm extends to the SkS problem as well. We also show that the robust ST admits a very simple algorithm in this framework, albeit with an inferior approximation guarantee (Theorem 3.4.2).

- For the robust matroid optimization (MO) problem (which generalizes the ST problem) we show an algorithm with an approximation guarantee $\log r + \log k$ (Theorem 3.5.2), where r is the rank of the matroid. This algorithm has an optimal dependence of the parameter k, subject to the hardness result for this problem. This algorithm invokes an approximation algorithm for *submodular function maximization (SFM)*. The algorithm is combinatorial, except for the subroutine which runs the SFM algorithm, the implementation of which is combinatorial in the unweighted case, or LP-based in the weighted case.

- We study the robust SP problem with several restrictions of the ERCC model. When the number of edges which can be failed across all scenarios is bounded, we show that the problem is solvable in polynomial time (Theorem 3.6.2). When the number of scenarios is bounded we prove that the problem remains NP-hard for four or more scenarios in arbitrary graphs (Theorem 3.6.5), while for directed acyclic graphs the problem is solvable in polynomial time (Theorem 3.6.4). Finally, when the cardinality of each failure set is bounded by a constant B we show constant factor approximation algorithms for arbitrary graphs and $B = 2$ (Theorem 3.6.10), as well as series-parallel graphs and arbitrary B (Theorem 3.6.16). At the same time we show that the problem is APX-hard even for $B = 2$ (Theorem 3.6.21). This result also holds on series-parallel graphs and $B > 2$ (Theorem 3.6.23). The algorithms in this section employ different methods and ideas, which can be adapted to solve other robust counterparts.

In Chapter 3 we also show some results for SIRCC(SP). In particular, we show that

- SIRCC(SP) on directed graphs admits no polynomial algorithms with approximation guarantee $\log k$ assuming that NP $\not\subset$ DTIME($n^{\log \log n}$), even when restricted to directed acyclic graphs (Theorem 3.7.4). In contrast, when k is bounded and the graphs are acyclic, or when the graph is series-parallel, we show that the problem admits a polynomial algorithm (Theorem 3.7.8). In addition, we show that the case $k = 1$ is solvable in polynomial time with a simple combinatorial algorithm (Theorem 3.7.3). We also study the fractional version of this problem. We prove that the integrality gap of this problem is $k + 1$ (Theorem 3.7.6), and obtain an approximation algorithm with this factor, as a result (Theorem 3.7.7).

Chapter 4 deals with the model ARCC. Here we focus on the SP and ST problems. We study the unweighted case of both problems. Concretely, we obtain the following results.

- We show that ARCC(SP) is NP-hard to approximate within any factor $\delta < 2$. Moreover, the deciding if a solution to ARCC(SP) is feasible is NP-hard (Theorem 4.2.21). Our study of the algorithmic aspects of ARCC(SP) focuses on the unweighted case. For the fixed-parameter

case $k = r = 1$ we obtain a simple exact polynomial algorithm (Theorem 4.2.11). For the case in which $k = 1$ and r is allowed to vary we obtain a combinatorial 2-approximation algorithm (Theorem 4.2.14). We also provide some structural results (Theorems 4.2.4, Lemma 4.2.20).

- We observe that ARCC(SP) is APX-hard. At the same time, we obtain a simple combinatorial 2-approximation algorithm (Theorem 4.3.2). For the case of fixed r we are able to obtain an approximation algorithm with a better approximation guarantee (Theorem 4.3.11). This approximation factor tends to 1 as k grows. We also give a better analysis for the combinatorial algorithm for some parameters k and r (Corollary 4.3.8), and examine the relation to the minimum m-edge connected spanning subgraph problem.

Chapter 2

Preliminaries

In this chapter we establish notation and mention some basic facts that will be used throughout this thesis. We refer the reader to the book of Schrijver [62] for a thorough treatment of most combinatorial notions defined hereafter. For an introduction to the field of computational complexity theory, we refer reader to the book of Garey and Johnson [39].

2.1 Graphs and matroids

For an integer $n \in \mathbb{Z}_+$ we denote $[n] = \{1, 2, \cdots, n\}$. All graphs considered in this thesis are finite. Let $G = (V, E)$ be a graph. For an edge e connecting two vertices u and v we write $e = uv$. If the graph is directed we use the latter notation to indicate that u is the tail of the edge and v is its head. We say that e is incident to u and to v. For a subset of the vertices $V' \subset V$ we denote by $E[V']$ all edges in E with both endpoints in V'. For two subsets $U, W \subset V$ we denote by $E(W, U)$ the set of edges with one endpoint in U and another in W. If G is directed the latter set contains all edges with the tail in U and the head in W. For a subset of edges $E' \subset E$ we denote by $V[E']$ all vertices, which are incident to at least one edge in E'.

A graph $H = (V', E')$ is a subgraph of G if $V' \subset V$ and $E' \subset E$. In this case we write $H \subset G$. For a set of vertices $V' \subset V$ the subgraph of G induced by V' is $G[V'] = (V', E[V'])$.

Paths in the graph are always vertex-disjoint, unless stated otherwise. Paths are represented as sets of edges. A cut in G with shores S and $V \setminus S$ is denoted by $(S, V \setminus S)$. The set of edges crossing the cut is defined as

$$\Delta(S) := \{e \in E : |e \cap S| = 1\}.$$

An r-cut is a partition of V into r nonempty subsets $V_1, \cdots, V_r$. The r-cut is

denoted by $(V_1, \cdots, V_r)$. The set of edges crossing the r-cut is defined as

$$E(V_1, \cdots, V_r) := \{e \in E : |e \cap V_i| \leq 1 \quad \forall i \in [r]\}.$$

For a set of vertices $V' \subset V$ and a vertex $v \in V$ we denote by $G - V'$ and $G - v$ the graphs, which are obtained by removing the corresponding vertices as well as all edges incident to them from G. For a set of edges $E' \subset E$ and an edge $e \in E$ we denote by $G - E'$ and $G - e$ the graphs, which are obtained by removing the corresponding edges from G.

For an edge $uv = e \in E$, the contraction of G by e is the graph obtained from G by removing e, replacing the two vertices incident to e with a single new vertex w, and replacing every other edge of the form $e' = uu'$ or $e'' = vv'$ by wu' and wv', respectively. For a subset of edges $E' \subset E$, the contraction of G by E' is the graph obtained by performing $|E'|$ consecutive contraction operations on G, once for every edge in E'.

A weighted graph is a graph, accompanied with a cost function $w : E \to \mathbb{Q}$. The weight of any subset of edges $E' \subset E$ is computed via

$$w(E') := \sum_{e \in E'} w(e). \tag{2.1}$$

We will sometimes use the vector notation w_e to denote $w(e)$.

A capacitated graph is a graph accompanied by a nonnegative capacity function $c : E \to \mathbb{Q}_+$. The capacity of a cut $(S, V \setminus S)$ in G is defined as the quantity

$$c(E(S, V \setminus S)) = \sum_{e \in E(S, V \setminus S)} c(e).$$

If G is directed, the previous sum ranges over all edges in $E(S, V \setminus S)$ with a tail in S. We say that $(S, V \setminus S)$ is a s-t cut for some distinct $s, t \in V$ if S contains exactly one of the vertices s, t.

For a vector indexed by the vertices $d \in \mathbb{Q}^V$ of a capacitated graph G, a d-flow is a function $f : E \to \mathbb{Q}_+$, which satisfies $f(e) \leq c(e)$ for every $e \in E$ as well as

$$\sum_{ux \in E} f(e) - \sum_{xu \in E} f(e) = d_u,$$

for every $u \in V$. If f is a d-flow for a vector d of the form

$$d_u = \begin{cases} -R & \text{if } u = s \\ R & \text{if } u = t \\ 0 & \text{otherwise.} \end{cases}$$

for some distinct vertices $s, t \in V$ we say that f is an s-t flow of value R. Menger's max-flow min-cut theorem states that if G is a capacitated graph, $s, t \in V$ are two distinct vertices, $(S, V \setminus S)$ is an s-t cut with minimum capacity C^*, and f is an s-t flow with maximum value R^*, then $C^* = R^*$. If G is a weighted graph, then the cost of the flow f is defined as

$$w(f) := \sum_{e \in E} w(e) f(e).$$

Given a capacitated, weighted graph G and a vector $d \in \mathbb{Q}^V$ it is possible to find a maximum s-t flow, a minimum s-t cut, a minimum cost d-flow, a shortest s-t path (a path connecting s and t with minimum cost) and a minimum spanning tree (a connected subgraph of $T \subset G$ with minimum cost) in polynomial time. For a detailed description of the methods underlying the latter algorithms see the book of Schrijver [62].

A matroid $\mathcal{M} = (A, \mathcal{I})$ is a pair with a finite ground set A and a collection of subsets $\mathcal{I}$ of A satisfying the following properties.

- $\emptyset \in \mathcal{I}$.
- If $Y \subset X \in \mathcal{I}$ then $Y \in \mathcal{I}$.
- If $Y, X \in \mathcal{I}$ with $|Y| < |X|$ then there exists $a \in X \setminus Y$ such that $Y \cup \{a\} \in \mathcal{I}$.

Any set system satisfying the first two constraints is called an independence system. The third property is called the augmentation property. Any subset $X \in \mathcal{I}$ of maximal cardinality is called a basis of the matroid. All bases of a matroid have the same cardinality, due to the exchange property. A matroid is weighted if it is accompanied by a cost function $w : A \to \mathbb{Q}_+$. The weight of a subset of elements $X \subset A$ is defined as in Equation (2.1). The problem of finding the minimum (maximum) cost basis of a matroid is solvable in polynomial time. for details we refer again to the book of Schrijver [62].

2.2 Linear and integer programming

A linear program (LP) is an optimization problem of the form

$$\min\{c'x : Ax \leq b,\ x \in \mathbb{R}^n\},$$

where x is a vector of variables in n-dimensional space, $Ax \leq b$ is a system of linear inequalities and c is an n-dimensional vector. The feasible set of an LP is the polyhedron $\{x \in \mathbb{R}^n : Ax \leq b\}$.

An linear integer program (IP) is an optimization problem of the form

$$\min\{c'x : Ax \leq b,\ x \in \mathbb{Z}^n\}.$$

While LPs can be solved in polynomial time, IP is an NP-complete problem. We refer the reader to the book of Schrijver [62] for an introduction to LP and IP theory.

2.3 Algorithms and complexity

We are assuming that in all computational problems addressed by this thesis, the encoding of the input is done in the ordinary compact way. For details see the book of Garey and Johnson [39]. We say that a decision problem is polynomial if it belongs to the class P. We say that an optimization problem is polynomial if there is an algorithm whose running time is bounded by a polynomial function of the input length, which reports the optimal solution on every legal input. We refer to the book of Garey and Johnson [39] for a treatment of the notions of NP-hardness, NP-completeness and reducibility between combinatorial problems.

A β-approximation algorithm ($\beta \geq 1$) for a minimization problem is an algorithm that given an input with an optimal value OPT, finds a feasible solution with value at most βOPT. Note that β can depend on the input size. An approximation algorithm is a constant factor approximation if it is a β-approximation algorithm for some constant $\beta \geq 1$ (independent of the input size). The class of optimization problems admitting constant factor approximation algorithms is called APX. For notions of completeness in APX and related topics we refer the reader to the paper of Papadimitriou and Yannakakis [60].

Chapter 3

Non-uniform Faults

3.1 Introduction

This chapter studies robust models of combinatorial optimization with nonuniform failures. The notion of uniformity accounts for the extent to which, all the resources in the system are equally vulnerable to adversarial attacks. More precisely, in a uniform failure model all subsets of resources of a particular size comprise a potential failure scenario. Consequently, a model is nonuniform, if the sets of resources that the adversary is able to fail are not given by some upper bound on the cardinality of such a set. In the extreme case, the adversary is only able to fail one of k subsets, explicitly given in the input. This situation corresponds to the ERCC model. The SIRCC model is an extension of the uniform IRCC model. While the set of scenarios is any subset of k resources in the IRCC model, the SIRCC model assumes every subset of k elements of a given subset U to comprise a scenario. As we will see later, this difference is significant, both from the algorithmic, and from the complexity theoretic point of view.

This chapter deals with the models ERCC and SIRCC. We briefly recall the former model, the study of which represents the majority of our efforts. The input in ERCC($\mathcal{P}$) constitutes an instance $P = (C, w)$ of the underlying monotonic combinatorial problem $\mathcal{P}$, as well as k *failure sets* $\Omega = \{F_1, \cdots, F_k\}$, where $F_i \subset A$ corresponds to a set of resources not available in the i'th scenario. The goal is to find a solution $X \subset A$ with $X \setminus F_i$ feasible (namely $X \setminus F_i \subset \mathcal{S}$), so as to minimize $w(X)$. We will assume throughout this chapter that the instance is feasible. As we remarked in the introduction, feasibility can be verified by checking that $A \setminus F_i \in \mathcal{S}$ holds in every scenario $F_i \in \Omega$.

To this end we elaborate on the interpretation of combinatorial problems as monotonic problems. It was mentioned in the introduction that any combinatorial problem $\mathcal{P}$ admits a canonical extension to a monotonic property $\mathcal{P}'$

by means of the following transformation. For every instance P of $\mathcal{P}$ with a feasible set $\mathcal{S}$, $\mathcal{P}'$ contains an instance with the following *monotonic extension* of $\mathcal{S}$

$$Mon(\mathcal{S}) = \{X \subset A : \exists X' \in \mathcal{S},\ X' \subset X\}. \tag{3.1}$$

In other words, $Mon(\mathcal{S})$ contains all subsets $X' \subset A$, which contain some feasible set $X \in \mathcal{S}$. It is straightforward to verify that $\mathcal{P}'$ is indeed monotonic and that $\mathcal{S} \subset Mon(\mathcal{S})$ always holds. Consider for example the ST problem. The normal variant of ST has as feasible sets all spanning trees of a given graph. In contrast, the monotonic version of ST has as feasible sets all subsets of the edges, which are *connected* (namely, which contain a spanning tree of the graph). Throughout this thesis, whenever we refer to a concrete combinatorial problem, and unless otherwise stated, we mean the monotonic extension of this problem.

Our first result for the class ERCC is a general hardness-of-approximation result. In simple words, this result states that ERCC($\mathcal{P}$) contains the *Set Cover (SC)* problem as a special case for (almost) every *graph problem* $\mathcal{P}$. The notion of a graph problem is defined next.

Definition 3.1.1. A combinatorial problem $\mathcal{P}$ is a *graph problem* if the ground set of every instance of $\mathcal{P}$ is the edge set of a graph [1], and every graph $G = (V, A)$ is associated with exactly one instance of $\mathcal{P}$.

For example, ST, SP, BM and SkS are all graph problems. In fact, all concrete problems addressed by this thesis are graph problems, or generalizations of graph problems. To prove our hardness result we will need two more definitions.

Definition 3.1.2. A combinatorial problem $\mathcal{P}$ is *simple* if for *every* instance P of $\mathcal{P}$, the feasible set $\mathcal{S}$ is either 2^A (namely all subsets of the ground set), or the empty set.

Definition 3.1.3. A graph problem $\mathcal{P}$ is *structural* if for every instance P with an associated graph $G = (V, A)$, a feasible set X, an infeasible set Y, two parallel edges $e_1, e_2 \in A$ and a copy $e' \notin A$ of e_1 and e_2 the following conditions hold.

- If $e_1 \in X$ and $e_2 \notin X$, then the set $X' = X \setminus \{e_1\} \cup \{e_2\}$ is also feasible.
- X is feasible for the instance of $\mathcal{P}$ associated with the graph $G' = (V, A')$, with $A' = A \cup \{e'\}$.

[1] We allow parallel edges and self-loops in graphs.

- Y is infeasible for the instance of $\mathcal{P}$ associated with the graph $G' = (V, A')$, with $A' = A \cup \{e'\}$.

Obviously, none of the graph problems discussed so far is simple. This definition is only needed to eliminate some possible pathology in the proof of Theorem 3.1.4. Note that by 'simple' we do not mean 'easy to solve'. Structural problems are all problems which do not distinguish between several copies of parallel edges. In other words, any edge in a feasible solution can always be replaced by any parallel copy of it, without breaking feasibility. Another requirement is that while adding parallel edges to a graph might introduce new feasible solutions, it cannot render a feasible solution in the original graph infeasible. It is also easy to verify that all combinatorial problems discussed so far are structural problems. We demonstrate this on the ST problem. Consider any feasible solution X to the ST problem with input graph $G = (V, E)$. Feasibility of X simply means that the graph (V, X) is connected. It is obvious that replacing any edge $e \in X$ with a parallel copy $e' \notin X$ results in another connected graph. Furthermore, if we add any number of parallel copies of edges to the edge set E, (V, X) will remain a connected subgraph of the resulting graph. Hence, ST is a structural problem.

We are ready to state our general hardness result.

Theorem 3.1.4. *Let $\mathcal{P}$ be a non-simple structural graph problem. Assuming that NP $\not\subset$ DTIME($n^{\log\log n}$), there is no polynomial $c \log k$-approximation algorithm for ERCC($\mathcal{P}$) for any $c < 1$.*

We prove this theorem in the following section. From the previous discussion we obtain the following corollary.

Corollary 3.1.5. *Assuming that NP $\not\subset$ DTIME($n^{\log\log n}$), there is no polynomial $c \log k$-approximation algorithm for ERCC(SP), ERCC(ST), ERCC(StS) and ERCC(BM), for every $c < 1$.*

Our goal in the following sections is to develop approximation algorithms for the problems mentioned in Corollary 3.1.5. In fact, we will be able to match the latter lower bound for some of these problems with various algorithm. To obtain approximation algorithms which perform even better we also study restrictions of the ERCC model.

Let us briefly outline the organization of this chapter. Section 3.2 is devoted to the proof of Theorem 3.1.4. Section 3.3 describes algorithms for various ERCCs using techniques of LP rounding. Section 3.4 outlines a combinatorial algorithmic framework for ERCCs, and presents algorithms for the

ERCC(ST) and ERCC(S2S) problems. The matroid optimization problem is considered in Section 3.5. Various restrictions of the ERCC model are studied in Section 3.6. Section 3.7 focuses of the SIRCC model, and more specifically, on the SIRCC(SP) problem.

3.2 A generic hardness of approximation result

In this section we prove Theorem 3.1.4. In fact we show that the theorem holds in the unweighted case $w \equiv 1$.

Proof of Theorem 3.1.4.

We call an instance P simple if $\mathcal{S}$ is either 2^A or the empty set. Since $\mathcal{P}$ is not simple, there exists a non-simple instance $\bar{P}$ of $\mathcal{P}$. Let $G = (V, E)$ be the associated graph and let $\bar{E} \subset E$ be a inclusions-wise minimal feasible set. From the fact $\bar{P}$ is not simple, and from monotonicity of $\mathcal{P}$ we have $\bar{E} \neq \emptyset$. Let $e \in \bar{E}$ be any edge.

To prove that ERCC($\mathcal{P}$) is hard to approximate we reduce the SC problem to it. Recall that SC is defined in the following way. Given a ground set $S = \{a_1, \cdots, a_n\}$ of n elements, a family $\mathcal{F} = \{A_1, \cdots, A_m\}$ of m subsets of S and a positive integer $1 \leq D \leq m$, does there exist a set $\mathcal{B} \subset \mathcal{F}$ of at most D sets satisfying $S = \cup_{A \in \mathcal{B}} A$? The optimization version of the problem asks to find a cover of minimal cardinality, and is known to admit no polynomial approximation algorithms with factor $c \log n$ for any $c < 1$, unless NP $\subset$ DTIME($n^{\log \log n}$) (see Feige [35]).

Let $I = (S, \mathcal{F})$ be an instance of SC and let $k = |S|$ and $m = |\mathcal{F}|$. We will construct an instance of ERCC($\mathcal{P}$) using I and $\bar{P}$. We emphasize that $\bar{P}$, G, $\bar{E}$ and e are fixed for the reduction, hence they are the same for any I. Consider the instance P' corresponding to the graph G', obtained from G by removing all edges in $E \setminus \bar{E}$ and e, and adding m edges $e_1, \cdots, e_m$ parallel to e, corresponding to the sets $A_1, \cdots, A_m \in \mathcal{F}$. The edge set of this graph is hence $\bar{E} \setminus \{e\} \cup \{e_1, \cdots, e_m\}$. Next we specify a set of k failure scenarios $\Omega = \{F_1, \cdots, F_k\}$ where F_i contains all edges of G', except for $e_1, \cdots, e_m$, plus those edges $e_1, \cdots, e_m$, which correspond to all sets A_j with $a_i \notin A_j$. Note that from the fact that $\mathcal{P}$ is structural, it follows that $\bar{E} \setminus \{e\} \cup \{e_i\}$ is a feasible set for P' for every $i \in [m]$, while $\bar{E} \setminus \{e\}$ is not a feasible set.

We claim that the optimal solution to ERCC($\mathcal{P}$) on P' contains $|\bar{E}| - 1 + D$ edges if and only if the minimum set cover in instance I has cardinality D.

Since $|\bar{E}| - 1$ is a constant, this implies that an approximation algorithm for ERCC($\mathcal{P}$) with approximation guarantee $c \log k = c \log |\Omega|$ for $c < 1$ can be used to simulate an approximation algorithm with the guarantee $c' \log k$ for any other $c < c' < 1$ for SC, which implies the hardness result.

To prove the claim observe that if I admits a set cover with D sets, we can construct a solution to ERCC($\mathcal{P}$) with cardinality $|\bar{E}| - 1 + D$ by simply taking all edges in $\bar{E}$ except for e, and also all edges from $e_1, \cdots, e_m$, which correspond to the set cover of size D. By the way the failure sets were defined, it is clear that the aforementioned set is feasible for P' (since for some $i \in [m]$, this set contains all edges in $\bar{E} \cup \{e_i\}$).

If, on the other hand, ERCC($\mathcal{P}$) admits a feasible solution X with at most $|\bar{E}| - 1 + D$ edges then we claim that the optimal set cover for the instance I contains at most D elements. To see this observe that the sets in $\mathcal{F}$ which correspond to the edges in $X \cap \{e_1, \cdots, e_m\}$ are a feasible solution to the SC problem on instance I. If this was not the case, let a_i be some element which is not covered. This implies that none of the edges $e_1, \cdots, e_m$ are contained in the graph $(V, X \setminus F_i)$, which means that $X \setminus H_i$ is not a feasible solution for instance $\mathcal{P}$, due to minimality of $\bar{E}$. We reached a contradiction to the assumption that X is a feasible solution to ERCC($\mathcal{P}$). Since, on the other hand, $\bar{E} \setminus \{e\} \subset X$ must hold, due to the fact that $\mathcal{P}$ is structural, we conclude that $|X \cap \{e_1, \cdots, e_m\}| \leq D$, and hence the optimal solution to set cover contains at most D sets. □

Note that it is possible to replace the complexity assumption in Theorem 3.1.4 with P≠NP, and the lower bound by $c \log k$ for some $c > 0$. In any case, Theorem 3.1.4 classifies the class of problems ERCC among NP-hard problems, which admit no polynomial constant-factor approximation algorithms. Therefore, our objective henceforth is twofold, and can be briefly summarized as follows.

- Obtain polynomial approximation algorithms for ERCC with a polylogarithmic approximation guarantee.
- Find restrictions of ERCC which can be solved efficiently, or which admit constant factor approximation algorithms.

The following sections mainly deal with the former goal. The obtained results achieve this goal for some important combinatorial problems such as ST, SP, StS and some generalizations. We conclude this section with some remarks related to the latter objective.

One approach is to restrict the class of graphs in the input. It is however easy to see that for the aforementioned combinatorial problems, the ERCC variant is hard as in Theorem 3.1.4 already for the class of series-parallel graphs. On the other hand, restricting the structure of the failure set can result in significant simplifications. This is demonstrated later on, when we study restricted versions of ERCC(SP).

3.3 LP-based algorithms

We start our algorithmic study of the class ERCC with a natural LP-based approach. Consider a combinatorial problem $\mathcal{P}$, which admits a representation as a *covering integer program (CIP)*. A CIP is any integer program in the form

$$\min\{w'x : Mx \geq b, \quad 0 \leq x \leq \mathbf{1}, \quad x \in \mathbb{Z}^m\}, \tag{3.2}$$

where $M \in \{0,1\}^{n\times m}$, is a binary matrix, b is a RHS and $\mathbf{1}$ is an n-dimensional all-ones vector. We call a combinatorial problem $\mathcal{P}$ *well representable* if its feasible sets are the feasible solutions of a CIP. More formally, for every instance P, there is a CIP formulation with decision variables $x \in \{0,1\}^m$, where $m = |A|$ is the cardinality of the feasible set, such that the set of feasible solutions of this IP is exactly the set characteristic vectors of feasible solutions $X \in \mathcal{S}$. It is well known that the class of CIPs admits $O(\log n)$-approximation algorithms. One way to obtain this approximation guarantee is to use LP relaxations and randomized rounding. The following proposition shows that the ERCC of well representable problems is also well representable.

Proposition 3.3.1. *Let $\mathcal{P}$ be a well representable combinatorial problem. Then ERCC($\mathcal{P}$) is also well representable.*

Proof. Let (P, Ω) be an instance of ERCC($\mathcal{P}$), with $\Omega = \{F_1, \cdots, F_k\}$, and let

$$\{x \in \{0,1\}^m : Mx \geq b, \quad 0 \leq x \leq \mathbf{1}, \quad x \in \mathbb{Z}^m\}$$

be the representation of the feasible set $\mathcal{S}$ of P. Let M_i be a matrix obtained from M by replacing all columns j, corresponding to elements $j \in [F_i]$, with the zero column. The following CIP models ERCC($\mathcal{P}$).

$$\{x \in \{0,1\}^m : \forall i \in [k] \quad M_i x \geq b, \quad 0 \leq x \leq \mathbf{1}, \}.$$

□

Observe that the number of constraints in the CIP representation of P is $n(P)$, then Proposition 3.3.1 immediately guarantees the existence of a $O(\log n(P) + \log k)$-approximation algorithm for ERCC($\mathcal{P}$). The most natural example of a problem, which falls into this category is SC. The natural CIP formulation of SC has n constraints, one for each element in the ground set, hence the ERCC(SC) formulation arising from Proposition 3.3.1 has nk constraints. We obtain the following easy corollary.

Corollary 3.3.2. *ERCC(SC) admits a $O(\log n + \log k)$-approximation algorithm.*

The situation with other combinatorial problems such as ST is not so simple. While CIP formulations of ST do exist, they typically have exponential size. Consider the following CIP formulation of ST.

$$(IP_{ST}) \qquad \min \sum_{e \in E} w_e x_e \tag{3.3}$$

$$\text{subject to} \sum_{x_e \in \Delta(S)} x_e \geq 1 \quad \forall S \subset V \tag{3.4}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E. \tag{3.5}$$

Unfortunately, Proposition 3.3.1 cannot be directly applied to this CIP for two reasons. Firstly, this IP has exponential size in the length of the input. Secondly, even if an algorithm for CIP can be applied to this problem, the approximation guarantee will be logarithmic in the number of constraints, which is exponential. As a result we will obtain a linear approximation factor. The way to overcome these difficulties is to use LP relaxation and randomized rounding. Consider the LP relaxation of IP_{ST}, which is obtained by replacing constraint (3.5) with $0 \leq x_e \leq 1 \, \forall e \in E$. We denote this LP by LP_{ST}. To solve the exponential size LP_{ST} one needs to apply the Ellipsoid algorithm with a separation oracle. The appropriate separation oracle for LP_{ST} can be implemented using a min cut (MC) algorithm. Indeed, if a violated inequality (3.4) exists for some (fractional) solution x, it corresponds to a cut in G with edge capacities defined by x and total capacity $c < 1$. An algorithm for MC is guaranteed to find the most violated such inequality.

To solve ERCC(ST) we need to obtain first a fractional solution to the LP relaxation of the IP obtained as in Proposition 3.3.1. For clarity we write this IP explicitly.

$$(IP_{ERCC(ST)}) \qquad \min \sum_{e \in E} w_e x_e \tag{3.6}$$

$$\text{subject to} \sum_{x_e \in \Delta(S) \setminus F_i} x_e \geq 1 \quad \forall i \in [k], \forall S \subset V \tag{3.7}$$

$$x_e \in \{0,1\} \qquad \forall e \in E. \tag{3.8}$$

We denote the LP relaxation of $IP_{ERCC(ST)}$ by $LP_{ERCC(ST)}$. This LP can clearly be solved in polynomial time as well using the same separation oracle. The only difference is that in order to find a violated inequality of $LP_{ERCC(ST)}$, one needs to invoke the MC algorithm at most k times: once for each set of constraints corresponding to a failure set F_i.

Consider an optimal fractional solution x^* of $LP_{ERCC(ST)}$. We demonstrate next how x^* can be rounded to obtain an integral solution, which is at most $c(\log n + \log k)$ times more expensive for some constant c, and feasible with overwhelming probability. The rounding method we use is randomized rounding. The idea in randomized rounding is to round up each component of the fractional solution corresponding to a single element e independently, with a probability which is proportional to its value x_e^* in the optimal fractional solution. In other words, a random rounding of a vector $x \in [0,1]^E$ is a random integral vector $y \in \{0,1\}^E$ with $Pr[y_e = 1] = x_e$, independently for each component. From linearity of expectation we have that for every cost vector $w \in \mathbf{W}$ the equality

$$\mathbb{E}[w(y)] = w(y) \tag{3.9}$$

holds. This indeed guarantees that in expectation, randomized rounding does not increase the cost of the solution. The remaining issue is to ensure feasibility of the solution. The standard way to perform this analysis is to compute the probability that a single constrain is violated by the rounded solution and then use union-bound to bound the probability that *all* constraints are satisfied simultaneously. This approach works directly for CIPs with a polynomial number of constraints. Unfortunately $IP_{ERCC(ST)}$ contains exponentially many constraints. To overcome this difficulty we use a technique due to Asadpour, Goemans, Madry, Gharan and Saberi [11]. The main ingredient of the analysis is the following lemma due to Karger [48].

Lemma 3.3.3 ([48]). *Let G be a graph and let γ be the size of the minimal capacity cut in G. Then for every half-integral $c \geq 1$, the number of cuts in G, which have size at most $c\gamma$ is at most n^{2c}.*

A special case of Lemma 3.3.3 asserts that the number of minimum cuts in a graph with n vertices is at most n^2. Note that the result holds for the general case of weighted (capacitated) graphs. We will use Lemmas 3.3.3 to our advantage in the following way. First we bound the probability that some cut $\Delta(S)$ is violated. This bound will be exponential in the value $x^*(\Delta_S)$. Then we bound the probability that some cut, whose capacity is the interval $[c\gamma, (c+\frac{1}{2})\gamma]$ is violated for every half-integer c using the union bound. The sum of the latter expressions, when c ranges over all half-integers from 1 to ∞ is a bound on the required probability. To obtain the first bound we need some form of the Chernoff bound. The following theorem is sufficient for our needs. Recall that a Bernoulli random variable with parameter p takes the value 1 with probability p and 0 with probability $1-p$.

Theorem 3.3.4. *Let $X_1, \cdots, X_k$ be k independent Bernoulli random variables with $\mathbb{E}(X_i) = p_i$ for $i \in [k]$. Then the random variable $X = \sum_{i=1}^k X_i$ satisfies for every $\delta \in (0,1]$*

$$Pr[X \leq (1-\delta)\mu] \leq e^{-\mu\delta^2/2},$$

and for every $\theta > 0$

$$Pr[X \leq (1+\theta)\mu] \leq e^{-\mu\theta^2/3},$$

where $\mu = \sum_{i=1}^k p_i$ is the expectation of X.

To successfully perform the aforementioned analysis we need to scale the fractional solution x^* by a sufficiently large number. Specifically, we define the e'th component of the scaled fractional solution x' to be

$$x'_e = \min\{1, Rx^*_e\}, \tag{3.10}$$

where $R = \log n + \log k$. The solution, which we will randomly round is x'. Note that if in some cut $\Delta(S)$, there is an edges e such that $x^*_e \geq \frac{1}{R}$, then the constraint corresponding to this cut is deterministically satisfied by the rounded solution y. In any cut $\Delta(S)$, which does not satisfy the previous condition, we have

$$x'(\Delta(S)) = Rx^*(\Delta(S)) \geq R, \tag{3.11}$$

since for all edges $e \in \Delta(S)$ we have $x^*_e < \frac{1}{R}$, while $x^*(\Delta(S)) \geq 1$ is due to feasibility of x^*. We henceforth assume that $\Delta(S)$ is of the latter type. This does not compromise generality since, as we commented before, all other cuts will satisfy $y(\Delta(S)) \geq 1$ deterministically. Consider any such cut $\Delta(S)$ of

capacity $x^*(\Delta(S)) = \alpha$ with $\alpha \in [c\gamma, (c+\frac{1}{2})\gamma]$. From Equation (3.11) we have that $x'(\Delta(S)) = R\alpha$, and by Theorem 3.3.4 we can bound the probability that the constraint of this cut is violated in the solution y by

$$Pr[y(\Delta(S)) = 0] = Pr[y[\Delta(S)] < (1 - \frac{R\alpha - 1}{R\alpha})R\alpha] \leq e^{-\frac{3R}{8}(\log n + \log k)\alpha}. \tag{3.12}$$

In the last inequality we used $1 - \frac{R\alpha-1}{R\alpha} > \sqrt{\frac{3}{4}}$. Assuming $R \geq 11$, the latter expression is bounded by

$$\frac{1}{k^{4\alpha}}\frac{1}{n^{4\alpha}} \leq \frac{1}{k}\frac{1}{n^{4c\gamma}} \leq \frac{1}{k}\frac{1}{n^{4c}}. \tag{3.13}$$

The last inequality follows from $\gamma \geq 1$, which is true since γ is the capacity of some cut with respect to the feasible solution x^*. This establishes the required bound for a single cut. Consider next *all* cuts with respect to x^*, which have capacity in the range $[c\gamma, (c+\frac{1}{2})\gamma]$. Next, we use the union bound over all such constraints and all scenarios F_i, $i \in [k]$. From Lemma 3.3.3 there are at most $n^{2c+1} \leq n^{3c}$ such cuts for every scenario. Consequently, there are at most kn^{3c} constraints in $LP_{ERCC(ST)}$, which correspond to such cuts. We conclude that the probability that some constraint corresponding to a cut with capacity in the interval $[c\gamma, (c+\frac{1}{2})\gamma]$ is violated is at most

$$kn^{3c}\frac{1}{k}\frac{1}{n^{4c}} = \frac{1}{n^c}.$$

To finally bound the probability that some constraint in some scenario is violated by y we sum over all possible choices of c.

$$Pr[y \text{ is infeasible}] \leq \sum_{c \geq 1,\ c \text{ half-integral}} \frac{1}{n^c} \leq \frac{1}{n-1}. \tag{3.14}$$

To complete the analysis of the algorithm is remains to show that randomized rounding does not increase the cost of the solution by too much. To do this we invoke Theorem 3.3.4 again. Let OPT denote the value of the optimal solution. For a vector $z \in \mathbb{R}^E$ let $w(z) = \sum_{e \in E} w_e z_e$. We have $w(x^*) \leq OPT$ and $\mathbb{E}[w(y)] = w(x') \leq ROPT$. Consider the probability

$$Pr[w(y) \geq (1+\delta)ROPT] = Pr[w(y) \geq (1+\delta)E[w(y)] \leq e^{(-ROPT\delta^2)/3}. \tag{3.15}$$

Assuming $OPT \geq 1$ and choosing $\delta \geq \sqrt{3}$ we get the bound

$$\frac{1}{kn}.$$

We can now use union bound to claim that with probability $\frac{2}{n-1} > \frac{1}{n-1} + \frac{1}{kn}$, the algorithm outputs a feasible solution y of $IP_{ERCC(ST)}$, with cost at most $(1+\sqrt{3})ROPT$. We conclude the description of this algorithm with the following theorem.

Theorem 3.3.5. *There is a probabilistic $O(\log n + \log k)-$approximation algorithm for ERCC(ST).*

We describe next a simple improvement of the previous algorithm. The algorithmic approach depicted above relied on the prohibitively expensive Ellipsoid algorithm for LP. In what follows we show that it is possible to use other IP formulations of all treated combinatorial problems, which are polynomial in the size of the input. The main ingredient is the following simple proposition.

Proposition 3.3.6. *Let $\mathcal{P}$ be a combinatorial problem such that the set of feasible solutions of every instance P of $\mathcal{P}$ can be represented as the feasible set of an IP*

$$\min\{w'x : M^P x \leq b\}$$

with $m = |A|$ variables $x \in \{0,1\}^A$ and n constrains. Then every instance of ERCC($\mathcal{P}$) can be modeled as an IP with $(k+1)m$ variables and $k(n+m)$ constraints.

Proof. Consider an instance (P, Ω) of ERCC($\mathcal{P}$). Let M^i be the matrix obtained from M^P by replacing the columns corresponding to each element $a \in F_i$ with the zero column. Consider the following IP with $y^i \in \{0,1\}^A$ for every $i \in [k]$ and $x \in \{0,1\}^A$

$$\min\{w'x : M^i y^i \leq b, \quad y^i \leq x \ \forall i \in [k]\}. \tag{3.16}$$

This IP models ERCC($\mathcal{P}$): The variables y^i ensure that for every scenario F_i, the obtained solution contains a feasible solution not using the elements of F_i. The constraints $y^i \leq x$ ensure that every element used in the solution of some scenario F_i are also used in the total solution x. □

Proposition 3.3.6 gives a simple way to obtain an IP representation of the robust counterpart from the IP formulation of the nominal problem. Consider, for example, the ST problem. One polynomial size IP formulation of ST uses flow variables. Concretely, the IP uses variables z_e to indicate the chosen set

of edges, and a set of variables $y_e^{s,t}$ for every distinct $s, t \in V$, which represent a 1-flow in G, capacitated by z. The IP reads

$$
\begin{aligned}
(IP2_{ST}) \qquad & \min \sum_{e \in E} w_e z_e & & (3.17)\\
\text{subject to } & \sum_{wu \in E} y_{wu}^{s,t} - \sum_{uw \in E} y_{uw}^{s,t} = 0 & \forall u \in V \setminus \{s,t\} \quad & (3.18)\\
& \sum_{ws \in E} y_{ws}^{s,t} - \sum_{sw \in E} y_{sw}^{s,t} = -1 & & (3.19)\\
& \sum_{wt \in E} y_{wt}^{s,t} - \sum_{tw \in E} y_{tw}^{s,t} = 1 & & (3.20)\\
& y_e^{s,t} \leq z_e & \forall e \in E \quad & (3.21)\\
& y_e^{s,t} \in \{0,1\} & \forall e \in E \quad & (3.22)\\
& z_e \in \{0,1\} & \forall e \in E, \quad & (3.23)
\end{aligned}
$$

where Constraints (3.18), (3.19), (3.20), (3.21) and (3.22) are written for every distinct pair of vertices $s, t \in V$. To write the IP obtained from Proposition 3.3.6 concisely we let $FLOW(E', z^i, y^i)$ denote the set of constraints in IP_{m-ECSS}, where the edge set of the graph is restricted to E' and the variables z and y are indexed by an additional index $i \in [k]$. Then the IP of the robust counterpart reads

$$
\begin{aligned}
(IP2_{ERCC(ST)}) \qquad & \min \sum_{e \in E} w_e x_e & \\
\text{subject to } & FLOW(E \setminus F_i, z^i, y^i) & \forall i \in [k]\\
& z_e^i = 0 & \forall i \in [k], \forall e \in F_i\\
& z_e^i \leq x_e & \forall i \in [k], \forall e \in E \setminus F_i\\
& x_e \in \{0,1\} & \forall e \in E.
\end{aligned}
$$

It is easy to verify that a fractional solution to the LP relaxation of $IP2_{ERCC(ST)}$ is also feasible for the LP relaxation of the CIP formulation of ERCC(ST). We conclude that the algorithm described in this section can be implemented with an LP with size which is polynomially bounded in the encoding length of the input. At the same time, the approach presented here requires solving very large LPs, whose size depends linearly on the number of scenarios. In the following sections we present several combinatorial algorithms, as well as algorithms which require solving significantly smaller LPs, and attain similar approximation guarantees.

3.4 A simultaneous covering approach

In this section we take a combinatorial approach. Theorem 3.1.4 shows that SC is inherently included in essentially every ERCC of a combinatorial problem. The idea behind the approach presented in this section is that SC can actually be used directly to solve the ERCCs of some problems. To this end we recall the *greedy algorithm for SC*. The input to SC is a certain ground set $S = \{a_1, \cdots, a_n\}$ of n, as well as a collection $\mathcal{F} = \{A_1, \cdots, A_m\}$ of subsets of S. Each subset A_i is associated with a cost w_i. The minimization variant of SC asks to find a cheapest collection $\mathcal{B} \subset \mathcal{F}$ of subsets, whose union equals S.

The greedy algorithm for SC works as follows. A current solution is initialized with the empty set. At each iteration the algorithm adds another set A_j to the solution. The set that is added is the one, which maximizes the *relative gain*

$$rg(A_j) = \frac{|A_j \setminus D|}{w_j},$$

where D is the union of all sets that were chosen by the algorithm so far. Ties are broken arbitrarily. The classical result of Chvátal [27] shows that the greedy algorithm is a $H(n)$-approximation algorithm, where $H(n)$ is the n-th Harmonic number. It holds that $H(n) \leq \log n + 1$, hence this algorithm is essentially the best possible, in light of the hardness result of Feige [35].

We start our exposition by showing a simple combinatorial algorithm for ERCC(ST). The key idea is to define a certain SC problem related to the ERCC(ST) instance and solve it to obtain an initial set of edges. If the solution is feasible stop. Otherwise, contract all edges in the initial solution in all graphs $G_i = (V, E \setminus F_i)$, define a new SC problem, corresponding to the new contracted instance and so on.

Let us describe the algorithm in detail. Initialize a set of edges $L = \emptyset$. Let $G_i = (V, E_i)$, where $E_i = E \setminus F_i$. With every pair $p = (v, i)$, where $v \in V$ is a vertex and $i \in [k]$ is a scenario, we associate a *constraint* C_p. The number of constraints is hence nk, where $n = |V|$, as usual. An edge $e \in E$ *covers* the constraint C_p if $p = (v, i)$, $v \in e$ and $e \in E_i$. In other words, an edge covers a constraint of some pair if it is not failed in the associated scenario and it is incident to the associated vertex. Consequently, each edge is associated with a subset R_e of the set of all constraints, which we denote by $\mathcal{C}$. The cost associated with the set R_e is w_e.

The former definitions give rise to a SC instance I_1. Concretely, we would

like to cover all constraints in $\mathcal{C}$ with the sets R_e, incurring a minimal total cost. Our first observation is that the optimal solution to this instance has at most the cost of the optimal solution to the ERCC(ST) instance.

Proposition 3.4.1. *Let OPT denote the optimal solution cost of the ERCC(ST) instance, and let o_1^* denote the optimal solution to the SC instance I_1. Then $o_1^* \leq OPT$.*

Proof. Consider an optimal solution $S^* \subset E$ to the ERCC(ST) instance. Consider the collection of subsets $\mathcal{B} = \{R_e : e \in S^*\}$. Clearly, the cost of this solution for the SC instance I_1 is $OPT = w(S^*)$. To see that this is a feasible solution to the SC instance consider any pair $p = (v, i)$ and its associated constraint C_p. Since S^* is a feasible solution to ERCC(ST), $(V, S^* \setminus F_i)$ is connected, and there is some edge e' in $S^* \setminus F_i$, which is connected to v. It follows that $R_{e'}$ covers C_p. □

Proposition 3.4.1 shows that every feasible solution to ERCC(ST) corresponds to a feasible solution to the SC instance with the same cost. Unfortunately, the converse is not true. Consider, for example, a simple case with $k = 1$. In this case the ground set of I_1 is simply associated with the vertex set V. The optimal solution to the SC instance corresponds to any set of edges, which is incident to every vertex. This set of edges can be a matching, which is not a set of edges that connects the graph.

The latter observation means that we might not be able to obtain a feasible solution to ERCC(ST) by only solving the SC instance I_1. We will, however, make significant progress, which will allow us to solve a smaller problem in the next iteration. Let $S_1 \subset E$ denote the solution obtained for the SC instance I_1. If S_1 is feasible for the ERCC(ST) instance the algorithm stops and reports S_1 as the solution. Otherwise, define G_i^1 to be the graph obtained by contracting the edges $S_1 \cap E_i$ in G_i. Define a new set cover instance I_2, which has as ground set all pairs $p = (v, i)$, where v ranges over the vertices in the contracted graph G_i^1. The sets R_e are defined analogously to the first iteration. The algorithm obtains S_2 for the instance I_2 and adds the corresponding edges to the solution obtained so far. This process continues until the solution becomes feasible for the ERCC(ST) instance. The complete algorithm is given as Algorithm 1.

We prove in the following theorem that Algorithm 1 is a $O(\log n(\log n + \log k))$-approximation algorithm.

Theorem 3.4.2. *Algorithm 1 is a $O(\log n(\log n + \log k))$-approximation algorithm for ERCC(SP).*

Algorithm 1 *Input: A ERCC(ST) instance* (G, Ω)*. Output: A feasible solution* $L \subset E$ *to ERCC(ST).*

1: $L \leftarrow \emptyset$.
2: $G_i \leftarrow (V, E \setminus F_i)$ for every $i \in [k]$.
3: **while** L is not a feasible solution **do**
4: $\mathcal{C} \leftarrow \cup_{i \in [k]} \{C_{(v,i)}\}_{v \in V[G_i]}$.
5: $R_e \leftarrow \cup_{F_i \in \Omega} \{C_{(v,i)} : e \notin F_i,\ v \in e\}$ and $w(R_e) = w_e$ for every $e \in E$.
6: $\mathcal{R} = \{R_e\}_{e \in E}$.
7: Solve the SC instance $I = (\mathcal{C}, \mathcal{R})$ using the greedy algorithm. Let S be the corresponding set of edges.
8: $L \leftarrow L \cup S$.
9: Contract the edges in $S \setminus F_i$ in G_i for every $i \in [k]$.
10: **end while**
11: Return L.

Proof. From Proposition 3.4.1 and from the approximation guarantee of the greedy algorithm for SC we know that the cost of the set of edges added in each iteration is at most $(\log n + \log k + 1)OPT$, where OPT is the cost of the optimal solution to the ERCC(ST) instance. It remains to bound the number of iterations performed by the algorithm. Consider some scenario $i \in [k]$. We claim that the number of vertices in the graph G_i drops by a factor of 2 or more in each iteration, until it is contracted to a single vertex. Indeed, consider the graph G_i in some iteration, and let $n' = |V[G_i]|$. For every $v \in V[G_i]$ the solution S to the SC instance corresponding to this iteration will include at least one edge $e \in E[G_i] \setminus F_i$, which is incident to v. It follows that the graph $(V[G_i], S)$ has no isolated vertices, hence the number of components it contains is at most $\frac{n'}{2}$. Since the number of vertices in G_i in the next iteration is exactly the number of such connected components the claim is proved. We conclude that after at most $\log n$ iterations, the number of vertices in every graph G_i is one. Consequently, there are at most $\log n$ iterations, each contributing at most $(\log n + \log k + 1)OPT$ to the cost of the solution. □

Although Algorithm 1 has an inferior worst-case approximation guarantee to the algorithm in Theorem 3.3.5, it still has several important advantages. Firstly, Algorithm 1 is combinatorial and extremely simple to implement. The basic operations required to carry out the algorithm are simple operations on arrays of numbers, such as summations and maximums, as well as simple manipulations of graphs, such as contraction. Secondly, Algorithm 1 has

an inherent distributed nature. This property is advantageous in very large graphs, in which no centralized storage of the entire graph is available.

Let us switch to a general discussion about the method underlying the previous algorithm. The main idea in the algorithm is to use the edges of the graph as covering sets for the constraints, defining the combinatorial problem. This approach can be generalized in the following way. Consider any instance P of a combinatorial graph problem $\mathcal{P}$. Let $C_1, \cdots, C_{r_P}$ be a set of *criteria* necessary and sufficient for the instance P. A criterion C_i is either satisfied or not satisfied by a set $R \subset E$ of edges. If a criterion is satisfied by a set R, then it is also satisfied by any superset of R. The latter property means that every criterion defines an upper ideal on the set of edges E. Finally, we require that a subset $R \subset E$ satisfies all criteria if and only if $R \in \mathcal{S}$, namely R is a feasible solution for the instance P.

Let us illustrate these definition by an example. Consider an instance P of SP. We argued before that $R \subset E$ is a feasible solution if and only if every s-t cut $\Delta(S)$ in the input graph intersects R. This gives rise to the a set of criteria as follows. For every $S \subset V$ defining an s-t cut we have a criterion C_S, which is satisfied by all sets of edges, which cross this cut. The SP problem can now be cast in our new terminology as finding the cheapest set of edges R^*, such that R^* satisfies all criteria defined above.

To this end we only restated certain combinatorial problems in a different way. The advantage is, that this new point of view casts all these problems as SC problems. Indeed, all we need in order to solve the instance P is to solve the SC problem with a ground set $\mathcal{C}_P = \{C_1, \cdots, C_{r_P}\}$, consisting of all criteria, the family $\mathcal{A} = \{A_R : R \subset E\}$ of covering sets, where

$$A_R = \{C \in \mathcal{C}_P : C \text{ is satisfied by } R\},$$

and the cost $w(R) = \sum_{e \in R} w_e$. There are several obvious difficulties concerning this approach. The first difficulty is that SC is a problem which is hard to approximate within a logarithmic factor of the cardinality of the ground set. This means that a direct application of this approach to solving a concrete problem is not likely to yield an exact (or indeed a constant-factor approximation) algorithm. In fact, if the number of criteria needed to define the problem is exponential in the size of the input (as is the case with our previous example of the SP problem), the best approximation guarantee that one can hope for is super-logarithmic.

Another difficulty concerns the cardinality of the set $\mathcal{A}$. This set is indexed by all subsets of edges in the input graph, and hence has exponential size. It follows that even applying the greedy algorithm is not possible without further

insights. In fact, performing an exact greedy choice in the SC instance defined above often means choosing the optimal solution of the modeled combinatorial problem directly.

Our final goal is to apply this approach to ERRCs of combinatorial problems. In light of the hardness result in Theorem 3.1.4, the first difficulty seems to be inherent to any efficient solution method for ERCC. Note that we will still need to keep the ground set of criteria small in order to get a logarithmic approximation guarantee.

To tackle the second difficulty we need to refine our framework. Our solution method for the SC problem will be the greedy algorithm. In order to perform this algorithm we need a subroutine, which chooses in each iteration the set in $\mathcal{A}$, which attains the highest relative gain. Let us denote the latter operation by *greedy selection*. For many combinatorial problems $\mathcal{P}$, the resulting greedy selection problem is NP-hard, and often hard to approximate. To address this issue we will restrict the domain of possible greedy choices to a smaller set $\bar{\mathcal{A}} \subset \mathcal{A}$. This restriction might allow us to perform the greedy selection efficiently, but it comes at a cost in the approximation guarantee we will achieve with the SC problem. To this end let us define a key parameter for controlling the aforementioned trade-of.

Definition 3.4.3. Let $\bar{\mathcal{A}} \subset \mathcal{A}$ be any subset of the family of covering subsets of the SC instance. Let OPT and $OPT_{\bar{\mathcal{A}}}$ be the costs of the optimal solution, and the cost of the optimal solution, when restricted to subsets from $\bar{\mathcal{A}}$. Then the *covering gap* is defined as

$$GAP(\bar{\mathcal{A}}) = \frac{OPT_{\bar{\mathcal{A}}}}{OPT}. \tag{3.24}$$

In simple words, the covering gap is the multiplicative factor by which the optimal solution to the SC instance deteriorates, when the family of covering sets is restricted to $\bar{\mathcal{A}}$.

Restricting the family of covering sets will allow us to reduce significantly the complexity of the greedy selection problem. In some cases we will still not be able to solve this problem exactly, but rather approximate it within a certain factor $\beta \geq 1$. In other words, we will be able to obtain a set $A_R \in \bar{\mathcal{A}}$ satisfying

$$\frac{|A_R \setminus D|}{w(A_R)} \geq \frac{1}{\beta} \max_{B \in \bar{\mathcal{A}}} \frac{|B \setminus D|}{w(B)}, \tag{3.25}$$

where D is the part of the ground set covered by previously chosen sets. We denote the resulting algorithm the *inexact greedy algorithm* and summarize it in Algorithm 2.

Algorithm 2 *Input: An instance P of $\mathcal{P}$. Output: A feasible solution $M \in \mathcal{S}$.*

```
D ← ∅.
M ← ∅.
Obtain a set of criteria C_P = {C_1, ··· , C_{r_P}} for P.
while D ≠ C_P do
    Obtain a set A_R ∈ Ā satisfying Equation (3.25).
    D ← D ∪ A_R.
    M ← M ∪ R.
end while
Return M.
```

A simple adaptation of the theorem of Chvátal [27] gives the following result.

Theorem 3.4.4. *Algorithm 2 is a γ-approximation algorithm for $\mathcal{P}$, with*

$$\gamma = \beta GAP(\bar{\mathcal{A}})(\log r_P + 1).$$

Let us finally return to the ERCC model. It is here that the advantage of using the SC problem comes to light. Put simply, the ERCC of a problem can be modeled in the current model without extra effort, by incurring only an additional logarithmic factor in the approximation guarantee. The only adaptation needed to Algorithm 2 is the introduction of a larger set of criteria. Concretely, given an instance (P, Ω) of ERCC($\mathcal{P}$), we obtain k sets of criteria $\mathcal{C}^1, \cdots, \mathcal{C}^k$, where $\mathcal{C}^i$ corresponds to the failure scenario $F_i \in \Omega$. Let $\mathcal{C}_P = \{C_1, \cdots, C_{r_P}\}$ denote the set of criteria corresponding to the nominal instance P. For each $i \in [k]$, the set $\mathcal{C}_i$ will contain r_P criteria $C_1^i, \cdots, C_{r_P}^i$; one for each criterion $C_j \in \mathcal{C}_P$. The criterion C_j^i is satisfied by a set of edges $R \subset E$ if and only if C_j is satisfied by $R \setminus F_i$. We can now take the union of all criteria

$$\mathcal{C}_P^\Omega = \bigcup_{i \in [k]} \mathcal{C}^i \tag{3.26}$$

to be the ground set of the SC problem. We call the algorithm obtained from Algorithm 2 by taking the latter ground set the *robust inexact greedy algorithm*. As a corollary of Theorem 3.4.4 we obtain the following theorem.

Theorem 3.4.5. *Algorithm 2 is a γ-approximation algorithm for ERCC($\mathcal{P}$), with*

$$\gamma = \beta GAP(\bar{\mathcal{A}})(\log r_P + \log k + 1).$$

Proof. Observe that any feasible solution to the SC problem corresponds to a feasible solution $S \subset E$ to ERCC($\mathcal{P}$). Consider any failure set $F_i \in \Omega$. Since

all criteria in $\mathcal{C}^i$ are satisfied by S, it holds that $S \setminus F_i$ satisfies all criteria in $\mathcal{C}_P$. It follows that $S \setminus F_i \in \mathcal{S}$, as required. The converse is also true. For some $S \subset E$ to be a feasible solution to the ERCC($\mathcal{P}$) instance, the set S needs to satisfy all criteria in $\mathcal{C}_P^\Omega$. Indeed, if some criterion $C_j^i \in \mathcal{C}_P^\Omega$ is not satisfied by S, then $S \setminus F_i$ does not satisfy criterion C_j, which means that $S \setminus F_i \notin \mathcal{S}$. We conclude that the robust SC problem models ERCC($\mathcal{P}$). The approximation guarantee follows from Theorem 3.4.4 and the fact that $|\mathcal{C}_P^\Omega| = kr_P$. □

Note that theorems 3.4.4 and 3.4.5 say nothing about the complexity of executing the greedy selection. In fact, it is often the case that the complexity of problem increases significantly when the static ground set $\mathcal{C}_P$ is replaced with its robust counterpart $\mathcal{C}_P^\Omega$. To illustrate this phenomenon consider the SP problem and the simple case $\bar{\mathcal{A}} = \mathcal{A}$. We use the set of criteria which correspond to s-t cuts, as we explained before. While the greedy selection problem for SP can be implemented by computing the shortest s-t path, the corresponding problem in the robust setup is NP-hard, since it contains the ERCC(SP) problem.

To demonstrate the implications of Theorem 3.4.5 we describe an algorithm for ERCC(S2S). The input to S2S (SkS) consists of a graph $G = (V, E)$, and the goal is to find the cheapest set of edges $S \subset E$, such that shortest-path distance between any two vertices $u, v \in V$ in $G' = (V, S)$ is at most twice (k times) this distance in G. We will assume unit costs, namely $w \equiv 1$. The problem ERCC(S2S) asks to find the cheapest set of vertices S, such that $S \setminus F_i$ is a 2-spanner of G. In the context of robust spanner problems it is often more realistic to solve the following less conservative problem (see e.g. Chechik, Langberg, Peleg and Roditty [24]). Find the cheapest set of edges S such that $(S \setminus F_i)$ is a 2-spanner of $G_i = (V, E \setminus F_i)$. In the context of spanners, the former setup is often called *strict robustness*, while the latter one is called *adaptive robustness*. Let us focus on the latter problem, which we denote AERCC(S2S). We remark that the technique presented here can be easily adapted to solve ERCC(S2S) as well.

The algorithm we present for AERCC(S2S) defines a certain SC problem and solves using the robust inexact greedy algorithm. The first work, which presented this algorithm for the nominal case of the S2S problem is the paper by Kortsarz and Peleg [53]. The authors presented this algorithm in an ad-hoc fashion and did not directly use the fact that the their algorithm solves a SC cover with a greedy heuristic. Our presentation focuses on the nominal case, followed by the extension to the robust case.

Let us start with a proper choice of a set of criteria for the S2S problem. The

following proposition states a well known fact about k-spanners.

Proposition 3.4.6. *A subgraph $G' = (V, S)$ of a graph $G = (V, E)$ ($S \subset E$) is a k-spanner of G if and only if for every edge $e = uv \in E$, the graph G' contains a path of length at most k from u to v.*

It is now easy to obtain a polynomial set of criteria $\mathcal{C}_P$ for an instance P of the S2S problem. For every $e \in E$, $\mathcal{C}_P$ will contain one criterion C_e, which is satisfied by a set of edges R if and only if R contains e, or any path of length 2 connecting the endpoints of e. Proposition 3.4.6 guarantees that this is indeed a valid set of criteria for S2S.

The next step is to obtain a good family $\bar{\mathcal{A}} \subset \mathcal{A}$ of covering sets. Recall that this set needs to have a small covering gap and a well-approximable greedy selection subroutine. Let us start with a simple definition.

Definition 3.4.7. A *star* in G is any subset $S \subset E$ of the edges, inducing a tree of diameter at most two. The *center* of a star with at least two edges is the only vertex in S with degree greater than one. The other vertices are the *leafs* of the star. The set of all stars in G is denoted by $STAR(G)$. The set of stars centered at a specific vertex $v \in V$ is denoted $STAR(v)$.

We are ready to fix

$$\bar{\mathcal{A}} = STAR(G),$$

and thus finish the description of the algorithm. It remains to compute the parameters $GAP(\bar{\mathcal{A}})$ and β corresponding to the chosen set $\bar{\mathcal{A}}$. Let us start by proving $GAP(\bar{\mathcal{A}}) \leq 2$. At this point we observe another important advantage of the current method. In order to prove an upper bound on the covering gap, one only needs to perform an existential argument. Concretely, consider any optimal 2-spanner $H^* = (V, E^*)$ of G. For every $v \in V$ let $S_v \subset E^*$ denote the star centered at v, which contains all edges incident to v in E^*. Note that

$$\sum_{v \in V} w(S_v) = 2w(E^*) \tag{3.27}$$

holds. Indeed, every edge $e = uv \in E^*$ is contained in exactly two of the stars: S_v and S_u. Note also that $\mathcal{B} = \{A_{S_v} : v \in V\}$ is a feasible solution to the SC problem. To see this consider any edge $e \in E$ and its corresponding criterion C_e. By Proposition 3.4.6 we need to show that either $e \in S_u$ for some u, or the endpoints of e are the leafs of some star S_u. Clearly, if $e \in E^*$, then $e \in S_u$ for any $u \in e$. Otherwise, from feasibility of H^*, there is a path of length 2 in H^*, connecting the endpoints of e. The middle vertex v of this path is such that S_v satisfies this criterion.

It remains to show that we can solve the greedy selection problem efficiently. To this end note that $|STAR(G)|$ is exponential in the size of the graph, hence a naive scan of this family cannot be done efficiently. We show a way to efficiently find the star around a given vertex u_0, which has the highest relative gain. The best star in the entire graph is then computed by performing the latter computation around every vertex $v \in V$.

Let us start by computing the relative gain $rp(A_S)$ of a single star $S \in STAR(u_0)$ with leafs $V' \subset V$. Recall that the relative gain of A_S is the ratio between the previously uncovered constraints it satisfies, and its cost $w(A_S) = w(S)$. Let $W = E \setminus M$ be the set of edges, whose criteria are not satisfied by the current solution in a given iteration of Algorithm 2. The set of criteria in W, which A_S covers is hence

$$A_S \setminus D = \{C_e : e \in E[V' + v] \cap W\}.$$

Let $R = E[V' + v] \cap W$ be the corresponding set of edges. Note that the edges in R can be divided into two parts. The first part, R_1, contains all edges, which are part of the star S itself, namely $R_1 = R \cap S$. The other part, $R_2 = R \setminus R_1$, contains all edges, which connect two leafs of the star S. With the latter definitions we can write the relative gain of A_S as follows.

$$rg(A_S) = \frac{|R_1| + |R_2|}{|S|}.$$

Let us assign *weights* θ_e and θ_u to the edges in $E[V']$ and the vertices in V' as follows. For every edge $e \in E[V']$ we assign the weight $\theta_e = 1$ if $e \in W$ and $\theta_e = 0$, otherwise. For every vertex $v \in V'$ we assign the weight $\theta_v = 1$ if $u_0 v \in R_1$ and $\theta_v = 0$ otherwise. With this definition and by using $|S| = |V'|$ we can rewrite the relative gain as

$$rg(A_S) = \frac{\theta(V') + \theta(E[V'])}{|V'|}.$$

The latter expression is known as the *weighted density* of the induced by the vertices V'. The greedy selection problem is hence to find the set $V' \subset \Gamma(u_0)$ with the maximal weighted density, with respect to the weight function θ. The latter problem is solvable in polynomial time using network flow techniques (see e.g. Gallo, Grigoriadis and Tarjan [38], or the paper of Kortsarz and Peleg [53] for faster constant-factor approximation, which can be used without changing the asymptotic guarantee of the S2S algorithm).

The latter discussion and Theorem 3.4.4 give the following theorem.

Theorem 3.4.8 ([53]). *There is a polynomial $O(\log n)$-approximation algorithm for S2S.*

We note that Kortsarz and Peleg give a slightly stronger statement of the previous theorem, which gives a better result for very sparse graphs. We omit the details of the required modifications. Let us turn back to the AERCC(S2S) problem. As we remarked before, an efficient procedure for the greedy selection problem in the nominal case does not guarantee the existence a similar procedure for the robust problem. Fortunately, in the case of AERCC(S2S) this generalization is simple. Let $\mathcal{C}^1, \cdots, \mathcal{C}^k$ be the sets of criteria corresponding to an instance of AERCC(S2S). Consider some iteration in the execution of Algorithm 2. Let

$$D^i = \mathcal{C}^i \cap D$$

denote the set of criteria corresponding to the i'th scenario, already covered by the current solution. The relative gain corresponding to some star S is given by

$$rg(A_S) = \frac{\sum_{i \in [k]} |A_S \setminus D^i|}{|S|}. \tag{3.28}$$

To this end we can define functions θ^i, one for each scenario, in the same way the function θ was defined for the nominal greedy selection problem. Concretely, we set $\theta^i_e = 1$ if and only if $e \in E \setminus F_i$, and $C^i_e \notin D^i$. Similarly, we set $\theta^i_v = 1$ if and only if $e \in E \setminus F_i$ and $C^i_{u_0 v} \in D^i$. Finally, we use the aggregated weight function

$$\theta = \sum_{i \in [k]} \theta^i$$

to represent the sum in numerator of Equation (3.28). The resulting problem can again be solved efficiently using any algorithm for the densest subgraph problem. This discussion in conjunction with Theorem 3.4.5 give the following theorem.

Theorem 3.4.9. *There is a polynomial $O(\log n + \log k)$-approximation algorithm for AERCC(S2S).*

The approximation guarantee in Theorem 3.4.9 is best possible up to a constant factor, given standard complexity assumptions. The optimality of the dependence on the parameter k follows from Theorem 3.1.4, while the optimality of the dependence on the number of vertices n, follows from the

hardness-of-approximation result by Kortsarz [52]. This paper shows that no $c \log n$-approximation algorithm exists for some constant $c > 0$ unless P=NP.

The results in theorems 3.4.8 and 3.4.9 are applicable to both directed and undirected graphs. Succeeding the work of Kortsarz and Peleg [53], Elkin and Peleg [58] used the latter technique to generate sparse directed k-spanners for $k \geq 3$. The challenge in this work is to prove a good bound on the covering gap. Their algorithm was later improved by Berman, Raskhodnikova and Ruan [18], who found a better set $\bar{\mathcal{A}}$ and proved a tighter covering gap. All these algorithms generalize to the robust versions using the algorithm we presented here, with a cost of an additional $\log k$ factor in the approximation guarantee. We note that none of the aforementioned papers treat the current framework in the generality it is presented here.

As we demonstrated, the method above is perfectly suited to treat ERCCs of certain combinatorial problems. In fact, it is likely that almost any nominal problem that can be cast in this setting without incurring a large approximation factor can be solved also in its robust version, albeit with a small adaptation of the greedy selection, and a seemingly unavoidable loss of a $\log k$ factor in the approximation guarantee. Another obvious advantage of the latter framework is its simplicity. In fact, the computational complexity of the algorithm solely depends on the complexity of the underlying greedy selection procedure. As we have seen, this gives combinatorial algorithms for robust spanner problems with an optimal approximation guarantee. It seems that a key feature of the applicability of the method is the ability to choose a covering set $\bar{\mathcal{A}}$, which allows 'efficient covers' of potential optimal solutions. Informally, a cover $A_{S_1}, \cdots, A_{S_t}$ of a feasible solution $A_O \in \mathcal{A}$ is efficient if $S_i \subset O$ for every $i \in [t]$ and every edge $e \in O$ appears in a small number of sets S_i. It is also required that the cover corresponds to a feasible solution to the SC problem. In the case of stars, we could easily bound the latter quantity by 2 for the S2S problem.

3.5 The robust matroid optimization problem

In this section we improve the algorithm for ERCC(ST) from the previous section in two ways. Firstly, we will achieve an algorithm with an optimal dependence on the parameter k. Secondly, we will be able to generalize the algorithm to the *matroid optimization problem (MO)*, which includes ST as a special case. We briefly recall the MO and ERCC(MO) problems. In the MO problem, given a matroid $\mathcal{M} = (A, \mathcal{I})$ and a linear cost function on its

ground set, the goal is to find a basis of the matroid with minimum weight. The ST problem is a special case of MO when $\mathcal{M}$ is the *graphic matroid* of a graph. In ERCC(MO) we are additionally given a collection Ω of scenarios and the goal is to find the cheapest subset S of the ground set, such that $S \setminus F_i$ contains a basis of the matroid $\mathcal{M}$ for every $F_i \in \Omega$. In fact, we will solve the following more general *simultaneous matroid basis (SMB)* problem. Given k matroids $\mathcal{M}_1, \cdots, \mathcal{M}_k$, with the same ground set A, find a minimum cost subset S, which contains a basis of each matroid. Note that ERCC(MO) can be transformed into SMB by setting $\mathcal{M}_i = (A, \mathcal{I}_i)$ and

$$\mathcal{I}_i = \{X \subset A : X \setminus F_i \in \mathcal{I}\},$$

for all $i \in [k]$. Recall that the *rank function* of a matroid $\mathcal{M}$ assigns a value to each subset of the ground set, namely

$$r(X) = \max\{|Y| : Y \subset X,\, Y \in \mathcal{I}\}.$$

We denote by $rank(\mathcal{M}) = r(A)$, the rank of the matroid. Let us fix an instance of SMB with k matroids $\mathcal{M}_1, \cdots, \mathcal{M}_k$ and a cost function w. Let r_i denote the rank function of the matroid $\mathcal{M}_i$ for $i \in [k]$. For brevity we let $\alpha_i = rank(\mathcal{M}_i)$ denote the rank of $\mathcal{M}_i$. Note that all matroids are defined with the same ground set A. Consequently, we can define the *aggregate rank* of a set $X \subset A$ as

$$f(X) = \sum_{i \in [k]} r_i(X). \tag{3.29}$$

Observe that $f(A) = \sum_{i \in [k]} \alpha_i$. Furthermore, a set $X \subset A$ is a feasible solution to the SMB instance if and only if

$$f(X) = f(A).$$

The function f is the key to our algorithm. An important feature of f is *submodularity*, namely the property that for every two sets $X, Y \subset A$ we have

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y). \tag{3.30}$$

Submodularity of f follows from submodularity of rank functions of matroids, and the fact that sums of submodular functions are also submodular. Let us briefly review an important operation on matroid called *contraction*. For a matroid $\mathcal{M} = (A, \mathcal{I})$ and $X \subset A$, the contraction of $\mathcal{M}$ by X is a matroid with ground set $A \setminus X$, denoted by $\mathcal{M}/X$, and defined by the rank function

$$r(Y) = r(Y \cup X) - r(X), \tag{3.31}$$

where r is the rank function of $\mathcal{M}$. [2]

Let us give a brief description of the algorithm. The algorithm is iterative. In each iteration, the algorithm maintains a partial solution obtained so far. Let S_j denote the set of elements comprising the partial solution in the end of the j'th iteration and let $S_0 = \emptyset$. The invariant that the algorithm maintains is that

$$f(A) - f(S_{j+1}) \leq \gamma(f(A) - f(S_j)) \tag{3.32}$$

holds for every iteration j and some constant $\gamma < 1$. Another feature is that

$$w(S_{j+1} \setminus S_j) \leq OPT, \tag{3.33}$$

where OPT is the optimal solution for this instance of SMB. Clearly, any algorithm satisfying the latter two conditions is a $O(\log f(A))$-approximation algorithm for SMB. Indeed, Equation (3.32) guarantees that after $O(\log f(A))$ the partial solution is feasible, and Equation (3.33) ensures that in each iteration, the set of elements added have a cost, which is bounded by OPT.

Let us focus next on a single iteration of the algorithm. In order to obtain the update set $Y_{j+1} = S_{j+1} \setminus S_j$ we will use an algorithm for the *submodular function maximization (SFM)* problem. Given a submodular set function g, the SFM problem asks to find a subset X of the ground set, maximizing $g(X)$. The SFM problem is *constrained (CSFM)* if in addition a linear budget constraint $w(X) \leq B$ is given, and the goal is to solve

$$\max\{g(X) : w(X) \leq B\}.$$

SFM (and CSFM) is an APX-hard problem. The best known algorithm for CSFM is a γ-approximation algorithm due Kulik, Shachnai and Tamir [55], with $\gamma = (1+\epsilon)\frac{e}{e-1}$ and any $\epsilon > 0$. [3]

The main loop of the algorithm is given as Algorithm 3. The algorithm assumes that the optimal solution value is known. We will remove this assumption later.

Let us prove the following lemma, which provides the approximation guarantee of Algorithm 3.

Lemma 3.5.1. *Algorithm 3 is a* $\log(\sum_{i\in[k]} \alpha_i)$*-approximation algorithm, given that in every computation in step 6., the CSFM problem is solved within approximation accuracy of at least* 2.

[2] It is possible to define a matroid via a rank function. For an introduction to matroid theory see, e.g., the book of Schrijver [62].

[3] This is also best possible, assuming standard complexity assumptions. In fact the hardness result for SC of Feige [35] implies that unless $NP \subset DTIME(n^{\log\log n})$, no approximation algorithm with a guarantee better than $\frac{e}{e-1}$ exists for SFM.

Algorithm 3 *Input: An instance $\mathcal{M}_1, \cdots, \mathcal{M}_k$ of SMB. Output: A feasible solution $S \subset A$ for SMB.*

1: $T \leftarrow \sum_{i\in[k]} \alpha_i$.
2: $S \leftarrow \emptyset$.
3: **while** $\sum_{i\in[k]} r_i(S) < T$ **do**
4: Define $g_i(X) = r_i(X \cup S) - r_i(S)$ for all $i \in [k]$.
5: Define $f(X) = \sum_{i\in[k]} g_i(X)$.
6: Solve approximately $Y \leftarrow \max\{f(X) : w(X) \leq OPT\}$.
7: $S \leftarrow S \cup Y$.
8: **end while**
9: Return S.

Proof. Note that in each iteration we increase the cost of the current solution by at most OPT. It remains to bound the number of iterations reformed by the algorithm. Observe that in each iteration of the algorithm it holds that $g_i(A \setminus S) + r_i(S) = \alpha_i$, since g_i is the rank function of the contracted matroid $\mathcal{M}/S$. It follows that

$$\sum_{i\in[k]} g_i(A \setminus S) = \sum_{i\in[k]} \alpha_i - \sum_{i\in[k]} r_i(S).$$

Denote the latter quantity by Δ. Let S^* be any optimal solution for the SMB problem with $w(S^*) = OPT$. Note that

$$\Delta = f(S^*) = \max\{f(X) : w(X) \leq OPT\}$$

holds, hence from our assumption on the approximation guarantee of the solution produced by the CSFM algorithm in step 6, we have

$$f(Y) \geq \frac{1}{2}\Delta.$$

It follows that

$$\sum_{i\in[k]} r_i(S \cup Y) = \sum_{i\in[k]} r_i(S) + r_i(Y) \geq \sum_{i\in[k]} r_i(S) + \frac{\Delta}{2}.$$

We conclude that Δ decreases by a factor of 2 in each iteration. Since $\Delta = \sum_{i\in[k]} \alpha_i$ holds in the first iteration, the total number of iterations is at most $\log(\sum_{i\in[k]} \alpha_i)$, which finishes the proof. □

We note that it is possible to obtain an approximation guarantee which is within any constant $\gamma > 1$ of $\ln(\sum_{i\in[k]} \alpha_i)$ (here $\ln(\cdot)$ denotes the natural logarithm).

It remains to remove the assumption that OPT needs to be known in advance. To this end, notice that for every value $T \geq OPT$ it holds that the optimal solution S^* to the SMB instance is feasible for the problem $\max\{f(X) : w(X) \leq T\}$. Note that the only requirement we have from the update set Y chosen in any iteration of Algorithm 3 is that $f(Y) \geq \frac{\Delta}{2}$ holds. Although it holds that S^* is not a feasible solution of $\max\{f(X) : w(X) \leq T\}$ whenever $T < OPT$ it can still happen, that a solution to the latter problem yields a set Y with $f(Y) \geq \frac{\Delta}{2}$. It follows that we cannot use binary search with the previous criterion to find OPT exactly. At the same time, we can still use this set Y to update the current solution S, since it satisfies both required conditions: It has a cost, which is smaller than OPT and it attains a large enough value for f. The latter discussion justifies the following approach.

Let us call a bound $T \in \mathbb{Z}_+$ *critical* if the update sets

$$Y_T \leftarrow \max\{f(X) : w(X) \leq T\} \quad \text{and} \quad Y_{T-1} \leftarrow \max\{f(X) : w(X) \leq T-1\}$$

obtained in step 6 of Algorithm 3 are such that

$$f(Y_T) \geq \frac{\Delta}{2} \quad \text{and} \quad f(Y_{T-1}) < \frac{\Delta}{2}.$$

We emphasize that Y_T and Y_{T-1} are not the optimal solutions to the corresponding problems, but rather the output of the approximation algorithm for the problem used in step 6. From our previous discussion, If T is critical at any iteration of Algorithm 3, it satisfies $T \leq OPT$. Note that it is possible to find a critical value T using binary search in the range $[0, \sum_{a\in A} w_a]$ in polynomial time. Consequently, in order to remove the dependence of Algorithm 3 on the knowledge of the value OPT, we simply need to replace step 6 with the aforementioned binary search. The set Y_T for any critical value T satisfies both necessary criteria for the correctness of the new algorithm: $w(Y_T) \leq OPT$ and $f(Y_T) \geq \frac{\Delta}{2}$.

This concludes the algorithm for SMB, and as a special case, for ERCC(MO). We obtain the following theorem.

Theorem 3.5.2. *For every $c > 1$, there is a polynomial $c\ln(\sum_{i\in[k]} \alpha_i)$-approximation algorithm for SMB.*

Note that in the case of the ERCC(ST) problem, which is a special case of SMB, we have $\sum_{i\in[k]} \alpha_i = k(n-1)$, and a hence $c(\ln k + \ln n)$-approximation

algorithm. The dependence of this factor on k is essentially the best possible in light of Theorem 3.1.4.

3.6 Restricted variants of ERCC

In this section we study the ERCC model with various restrictions. Concretely, we will address the following four types of restrictions. In the following definitions B is a fixed constant.

- **Fixed ERCC (FERCC):** The input is restricted to scenario sets Ω with $\sum_{F\in\Omega}|F| \leq B$.
- **Constant ERCC (CERCC):** The input is restricted to some constant number of scenarios, namely $k \leq B$.
- **Bounded ERCC (BERCC):** The input is restricted to scenario sets Ω with $|F| \leq B$ for every $F \in \Omega$.
- **Disjoint ERCC (DERCC):** The input is restricted to scenario sets Ω such that $F, F' \in \Omega$ and $F \neq F'$ implies $F \cap F' = \emptyset$.

We will focus on the SP problem in our study of restricted variants of ERCC. Let us start with a simple observation about the DERCC(SP). This variant of ERCC(SP) is the only variant, which admits no change in complexity by imposing the corresponding restriction. We prove this fact in the following proposition.

Proposition 3.6.1. *Every instance of ERCC(SP) can be transformed into an instance of DERCC(SP) with the same optimal cost and the same number of scenarios.*

Proof. Let G, w, s, t, Ω be an input to ERCC(SP) with k scenarios. Construct a new graph H from G by replacing each edge $e = uv$ in G with a u-v path of length k. Let the edges in this path be denoted by $f_1^e, \cdots, f_k^e$. Define a cost function w' on H by setting $w'(f_1^e) = w(e)$ and $w'(f_i^e) = 0$ for $i = 2, \cdots, k$ for every edge e in G. Note that there is a one-to-one correspondence between subsets of edges in G and subsets of edges in H, which identifies edges in

G with their corresponding paths in H. Also, by choice of w', the cost of a path in G is the same as the cost of the corresponding path in H. Finally, we define k failure scenarios for the graph H. For every failure scenario $F_i \in \Omega$ we define a single failure scenario F'_i by taking

$$F'_i = \{f_i^e : e \in F_i\}.$$

Clearly, $F'_i \cap F'_j = \emptyset$ whenever $i \neq j$, hence the instance with graph H, terminals s and t, weight function w' and scenarios $\Omega' = \{F'_1, \cdots, F'_k\}$ is an instance of DERCC(SP). In addition, note that the aforementioned correspondence between sets of edges also preserves the following property: A set of edges $S \subset E$ is feasible for the ERCC(SP) instance if and only if the corresponding set of edges is a feasible solution for the DERCC(SP) instance. In conjunction with the previous remark on this correspondence, this proves the claim.

□

Proposition 3.6.1 ensures that the hardness result in Theorem 3.1.4 also holds for DERCC(SP). In the remainder of this section we present results for the other three aforementioned restrictions of ERCC(SP). We will always assume disjointness of the scenario set in the algorithmic results. In light of Proposition 3.6.1, this does not compromise generality.

3.6.1 FERCC(SP)

Consider any problem $\mathcal{P}$ and its corresponding counterpart FERCC($\mathcal{P}$). Let A be the ground set of an instance (P, Ω) of FERCC($\mathcal{P}$). Consider an optimal solution $S^* \subset A$ to this instance. The assumption $\sum_{i\in[k]} |F_i| \leq B$ allows us to guess the set $M^* = S^* \cap (\bigcup_{i\in[k]} F_i)$. The knowledge of the set M^* can be transformed into a polynomial algorithm for SP, as shown in the following theorem.

Theorem 3.6.2. *There is a polynomial algorithm for FERCC(SP).*

Proof. Consider the optimal solution $S^* \subset E$ and the set of edges M^* defined above. From optimality of S^*, the graph $H = (V, S^* \setminus M^*)$ is a forest. Indeed, if this graph contained a cycle, removing any edge of this cycle would improve the cost of the solution, without breaking feasibility. Next, we fix some s-t path $P_i \subset S^* \setminus F_i$ for every $F_i \in \Omega$. Feasibility of S^* guarantees the existence of these paths, and optimality of S^* gives $S^* = \cup_{i\in[k]} P_i$. We will further

assume that the paths satisfy the following property. For every tree T in H and path P_i, the set of edges $T \cap P_i$ is a subpath of P_i. The latter assumption does not compromise generality. Indeed, if the assumption does not hold, we can replace the part of P_i which starts at the first vertex u that is in common to T and ends at the last vertex v that is in common to T with the unique u-v path in T. This new path is still intact in $S^* \setminus F_i$. To this end we identify for every tree T in H at most $2k$ *terminals* U_T by including in U_T the aforementioned two extreme vertices u and v, for every path P_i. We can now conclude from optimality of S^* that T is an optimal *Steiner Tree* on the set of terminals U_T, when edges are chosen from $E \setminus \cup_{i\in[k]} F_i$. To see this simply note that replacing T with any set of edges connecting the terminals in U_T results in a new feasible solution. Furthermore, every vertex $u \in U_T$, which is not s or t must be incident to some edge in M^*.

We are ready to state the algorithm. Let U' be the set of vertices incident to some edge in M^*, and let $U = U' \cup \{s, t\}$. Since $|M^*| \leq B$ we have $|U| \leq 2B + 2$. The algorithm guesses the partition of U into the subsets $U_1, \cdots, U_r$, such that U_j is the set of terminals of some tree in H. Then the algorithm finds minimum cost Steiner Trees $T_1, \cdots, T_r$, with T_j having terminals U_j. Since the number of terminals in each Steiner Tree is bounded, the latter computation can be performed in polynomial time. The optimality of the obtained solution for a correct guess of M^* and the partition into terminal sets follows from the previous discussion.

To claim the theorem we need to argue that it is possible to enumerate all possible partitions of the terminals in U in polynomial time. The latter claim is true since U contains a bounded number of vertices. □

3.6.2 CERCC(SP)

In this section we present two main results. On the positive side, we show that CERCC(SP) is solvable in polynomial time in directed acyclic graphs. On the other hand, when cycles are allowed in the graph, we show that the problem is NP-hard for every $B \geq 4$.

We start by presenting a simple algorithm for CERCC(SP) in directed acyclic graphs. In the heart of the algorithm is a simple reduction of the problem to the SP problem in a larger graph. Let us briefly describe the reduction next. Consider any optimal solution $S^* \subset E$ to an instance of CERCC(SP) with a directed acyclic graph G. For every $F_i \in \Omega$ we fix one s-t path $P_i \subset S^* \setminus F_i$. Let $U_i = V[P_i]$ be the set of vertices incident to the path P_i, and let $U = \cup_{i\in[k]} U_i$. We call a tuple $\alpha = (u_1, \cdots, u_k)$ of k vertices a *configuration*. A configuration

$\alpha_1 = (u_1, \cdots, u_k)$ *precedes* a configuration $\alpha_2 = (v_1, \cdots, v_k)$ if $\alpha_1 \neq \alpha_2$ and for every $i \in [k]$ either $u_i = v_i$, or $u_i v_i \in E \setminus F_i$. We say that an ordered pair of configurations $\ell = (\alpha_1, \alpha_2)$ is a *link* if α_1 precedes α_2. The cost of the link ℓ is defined as

$$\bar{w}(\ell) = w(\{u_i v_i : i \in [k]\}).$$

Note that $\bar{w}$ charges edges in a fixed-charge fashion, namely if $u_i v_i = u_j v_j = uv$ for some $i \neq j$, $\bar{w}(\ell)$ will incur the cost w_{uv} only once. We use $\alpha[j] = u_j$ to refer to specific vertices in a configuration $\alpha = (u_1, \cdots, u_k)$. For every $i \in [k]$ with $u_i \neq v_i$ we say that the edge $u_i v_i$ *appears* in the link ℓ. Finally, we define a directed graph $H = (W, R)$, where W is the set of all configurations and R contains an arc for every link $\ell = (\alpha_1, \alpha_2)$, directed from α_1 to α_2. The key to the algorithm is the following lemma.

Lemma 3.6.3. *H is acyclic. Furthermore, the optimal solution value to the CERCC(SP) instance is the same as the shortest α^s-α^t distance in H, where $\alpha^s = (s, \cdots, s)$ and $\alpha^t = (t, \cdots, t)$. Taking all edges e, which appear in some link in this path gives an optimal solution to CERCC(SP).*

Proof. The fact that H is acyclic easily follows from the fact that G is acyclic. Indeed if a cycle $\alpha_0, \alpha_1, \cdots, \alpha_r = \alpha_0$ exists in H, then for some index $i \in [k]$, the sequence $\alpha_0[i], \alpha_1[i], \cdots, \alpha_r[i] = \alpha_0[i]$ contains a cycle in G, leading to a contradiction.

Next, we prove that for every optimal solution $S^* \subset E$ to the CERCC(SP) instance, there is a α^s-α^t path in H, which has the same cost. By definition of P_i, every edge on this path is contained in the set $E \setminus F_i$. For a vertex $\alpha = (u_1, \cdots, u_k)$ of H and a path P_i, we say that α *is waiting for P_i in index j* if $j \neq i$, $u_j \in U_i$ and u_j appears after u_i on P_i, when it is traversed from s to t. For a vertex $\alpha = (u_1, \cdots, u_k)$ we define its *follower* $\alpha' = (u'_1, \cdots, u'_k)$ in the following way. If α is waiting for some path P_i in position j, then $u_i = u'_i$. Otherwise u'_i is the next vertex on the path P_i after u_i, namely u'_i is the unique vertex such that $u_i u'_i \in P_i$. The α^s-α^t path we define simply moves from the current vertex to its follower. To prove that α^t is indeed reached in this way we need to show that that a vertex is always different from its follower. To prove this we define an anti-symmetric relation $\propto_\alpha$ on the index set $[k]$ by setting $i \propto_\alpha j$ if and only if α is waiting for P_i in index j. Note that $\propto_\alpha$ is indeed anti-symmetric: If $i \propto_\alpha j$ and $j \propto_\alpha i$ holds at the same time, this implies the existence of a cycle in G, containing the vertices u_i and u_j. We conclude that any minimal element j of $\propto_\alpha$ is such that α is not waiting for any path P_i in index j. It remains to prove that the cost of this path is $w(S^*)$. To this end note that every $e \in S^*$ appears in exactly one link in the

α^s-α^t path we defined above. Indeed, for an edge uu' to appear in the link (α, α') in this path, it must hold that for all paths P_j which contain this edge we have $\alpha[j] = u$ and $\alpha'[j] = u'$. The bound on the cost of this path now follows from the way that the cost function $\bar{w}$ is defined (in particular, from the fact that edges are charged in a fixed-charge fashion).

To finish the proof of the lemma we need to show that every α^s-α^t path in H corresponds to a feasible solution S' with at most the same cost. Consider any such path $\alpha^s = \alpha_0, \alpha_1, \cdots, \alpha_m = \alpha^t$. We take S' to be the set of all edges $e \in E$, which appear in some link in this path. Feasibility of S' follows from the fact that for every $i \in [k]$, the sequence $s = \alpha_0[i], \alpha_1[i], \cdots, \alpha_m[i] = t$ defines an s-t path in G, which uses no edge of F_i. Since the cost of the path in H contains the cost of every edge, which is contained in some link at least once, we get that the cost of this path is at least $w(S')$.

We proved that every α^s-α^t path in H can be transformed into a feasible solution to CERCC(SP) with at most the same cost. On the other hand, we showed that any optimal solution S^* has a corresponding path in H, which achieves the same cost. This concludes the proof of the lemma. □

Lemma 3.6.3 along with the polynomiality of the size of H give the following theorem.

Theorem 3.6.4. *There is a polynomial algorithm for CERCC(SP), restricted to directed acyclic graphs.*

In the remainder of this section we show that if the graph is allowed to contain cycles, the problem becomes NP-hard for any bound $B \geq 4$. In fact, we show that this problem is APX-hard. We present a reduction from the SC problem, restricted to instances in which each covering set contains at most three elements. This problem contains the *vertex cover (VC)* problem restricted to cubic graphs (graphs with maximum degree three), which was proved to be APX-hard by Alimonti and Kann [9].

Theorem 3.6.5. *The CERCC(SP) problem is APX-hard for every $B \geq 4$.*

Proof. Consider an instance $(S, \mathcal{F})$ of SC with a ground set $S = \{a_1, \cdots, a_m\}$ and a family $\mathcal{F} = \{F_1, \cdots, F_m\}$ with $|F_i| \leq 3$ for every $i \in [m]$. We construct an instance of CERCC(SP) as follows. The graph G contains an s-t path $s = u_0, u_1, \cdots, u_n = t$. The edge $u_{i-1}u_i$ corresponds to the element a_i. In addition, G contains one edge $f_j = v_j w_j$ for every set A_j. Finally, for every a_i and A_j with $a_i \in A_j$ the graph contains the edges $a_{i-1}v_j$ and $w_j u_i$. The costs

of all edges are zero, except for the edges f_j, which have a cost of one. The failure scenarios are defined as follows. First, we obtain a partition of S into four sets S_l, $l \in [4]$, such that for every $l \in [4]$ and every distinct $a, a' \in S_l$ we have that a and a' appear in no set A_j, $j \in [m]$ simultaneously. This partition can be achieved by greedily constructing the sets S_l, one after the other. Next, we define four failure sets F_l, $l \in [4]$, corresponding to the sets S_l. The set F_l contains the edge $u_{i-1}u_i$ if $a_i \in S_l$. In addition, F_l contains all edges $a_{i-1}v_j$ and w_ju_i for every $a_i \notin S_l$ and every $j \in [m]$. This concludes the construction of the CERCC(SP) instance.

We claim that the cost of the optimal solution to the SC instance is identical to the optimal cost of the CERCC(SP) instance. Consider first an optimal solution $X^* = \{A_{i_1}, \cdots, A_{i_r}\}$ to the SC problem with cost r. The corresponding CERCC(SP) solution Y takes all edges in the graph, except for the edges f_j for $j \notin \{i_1, \cdots, i_r\}$. Clearly, the cost of this solution is r. To see that this solution is feasible consider any failure scenario F_l for $l \in [4]$. We claim that every vertex u_i on the path P is reachable from the preceding vertex u_{i-1} in $Y \setminus F_l$. Indeed, if $a_i \notin S_l$ and $u_{i-1}u_i \notin F_l$ this is obvious. Consider the case $a_i \in S_l$ and $u_{i-1}u_i \in F_l$. From feasibility of X^*, we know that there exists an index $j' \in \{i_1, \cdots, i_r\}$ such that $a_i \in A_{j'}$. Hence, also the path $u_{i-1}, v_{j'}, w_{j'}, u_i$ is intact in $Y \setminus F_l$. This concludes the first direction of the proof.

For the second direction, consider any optimal solution S^* to the CERCC(SP) instance. We claim that the SC solution X, which takes the set A_j if $f_j \in S^*$ is feasible. The cost of this solution is clearly the same as the cost of S^*. Consider any element a_i. Let $l \in [4]$ be such that $a_i \in S_l$. Consider any s-t path P in $S^* \setminus F_l$. Observe that by choice of the partition of S, the path P must contain all vertices u_i, $i \in [n]$. Indeed, if some u_i is not contained in this path, then it must hold that for some $i_- < i$, some $i_+ > i$, and some $j \in [m]$, the path P contains the edges $u_{i_-}v_j$ and $w_ju_{i_+}$. This means that both a_{i_-} and a_{i_+} are contained in S_l, and at the same time, they are contained in the set A_j. This contradicts the choice of the partition. Next note that by construction of G, the path P must contain the vertices u_i according to increasing order of their index. It follows that the vertex u_i appears after the vertex u_{i-1} on P. To prove that X contains some set covering a_i, simply note that the only way to get to u_i from u_{i-1} is via an edge f_j, such that $a_i \in A_j$. This concludes the proof of the theorem. □

Note that Theorem 3.6.5 works for both directed and undirected graphs. The remaining open cases of CERCC(SP) correspond to the cases $B \in \{2, 3\}$. Closing this gap is an interesting direction for future research.

3.6.3 BERCC(SP)

In this section we present various results for BERCC(SP). On the positive side we show that BERCC(SP) admits constant factor approximation algorithms in certain cases. We present algorithms for general graphs, as well as series-parallel graphs. We also show that the case $B = 1$ is solvable in polynomial time. From the computational complexity point of view, we show that BERCC(SP) is APX-hard for every $B \geq 2$. For $B \geq 3$ we show that achieving an factor $\frac{B}{2} - \epsilon$ is Unique Games Conjecture (UGC) hard.

Since BERCC(SP) with $B = 1$ is a special case of SIRCC(SP), we deffer the discussion about this special case to Section 3.7, which treats the SIRCC model. We comment here that this special case is the only one solvable in polynomial time, given that P≠NP, namely for $B \geq 2$ the problem becomes NP-hard. We deffer the discussion about the complexity of BERCC(SP) to a later stage. We focus first on approximation algorithms. The main results in this vein are a 17-approximation algorithm for the case $B = 2$ and a $(1 + B2^{B+1})$-approximation algorithm for the case $B \geq 3$ and restricted to series-parallel graphs.

The cut covering approach and the case $B = 2$

The most important technique, which is in the heart of both approximation algorithms discussed here, is an algorithm for a certain cut covering problem. Informally speaking, the cut covering problem is to find a minimum cost set of edges in a given graph, which crosses every cut in a certain collection of given cuts in the graph. This method is best depicted by presenting the algorithm for the case $B = 2$.

The algorithm operates in two phases. In each phase a relaxation of the original problem is solved approximately. The algorithm finally returns the union of both solutions as the solution of the original problem. Let us fix an instance $I = (G, \Omega)$ for BERCC(SP) with $G = (V, E), \Omega = \{F_1, \cdots, F_k\}$ and $|F_i| \leq 2$ for every $F_i \in \Omega$.

Define first the set $\Omega_0 = \{\{e\} : e \in F \in \Omega\}$. Observe that the problem $I_0 = (G, \Omega_0)$ is a relaxation of the problem I, since a feasible solution $S \subset E$ to the latter problem cannot admit an s-t cut, defined by some subset $F' \subset F \in \Omega$ of a failure scenario. The former problem can be solved exactly in polynomial time with the algorithm of Section 3.7.1. Obtaining this initial part of the solution concludes the first phase of the algorithm. Let $S_0 \subset E$ be such an optimal solution. Note that from the above discussion $w(S_0) \leq OPT$.

Let $\Omega' \subset \Omega$ denote the set of failures which form an s-t cut in (V, S_0). It remains to find a supplementary set of edges $S_1 \subset E$, so that $S_0 \cup S_1$ is a feasible solution for the instance I. We call the problem of finding a minimum cost such S_1 *the augmentation problem* $AUG(G, \Omega, S_0)$. In the remainder of this section we describe a 16-approximation algorithm for the augmentation problem.

Let $V' = V[S_0]$. To this end we need to use one result regarding the case $B = 1$, which is proved in Section 3.7.1. This result states that the subgraph $H = (V', S_0)$ is an s-t *bipath*, namely a union two s-t paths, which satisfy a certain property. The exact property is given in the following definition.

Definition 3.6.6. An s-t *bipath* in the graph $G = (V, E)$ is a union of two s-t paths $p_1, p_2 \subset E$, such that for every two nodes u, v incident to both p_1 and p_2, the order in which they appear on p_1 and p_2, respectively, when traversed from s to t is the same.

A bipath $Q = p_1 \cup p_2$ has the property that $p_1 \cap p_2$ is a set of bridges in (V, Q), and $Q \setminus (p_1 \cap p_2)$ is an edge-disjoint union of cycles. Again, we assume in this section that an optimal solution to BERCC(SP) with $B = 1$ can be obtained in polynomial time, leaving the details to Section 3.7.1.

Observe that by contracting all bridges in H, which cannot be part of any failure scenario, we obtain a graph decomposable into exactly two edge-disjoint paths. We fix any two such paths p_1, p_2. We also contract all edge on those paths, which are not contained in a failure scenario in Ω'. We denote the obtained graph by H'. Clearly, every failure set $F \in \Omega$ intersects each of the paths p_1, p_2 exactly once. This follows from the assumption $B = 2$. We conclude that $|p_1| = |p_2| = |\mathcal{F}'| =: d$. See Figure 3.1 for an illustration. Let the edges on the path p_i be numbered $e_1^i, \cdots, e_d^i$ according to their order on p_i from s to t. The following *prefix property* is clearly satisfied for every $i \in [2]$.

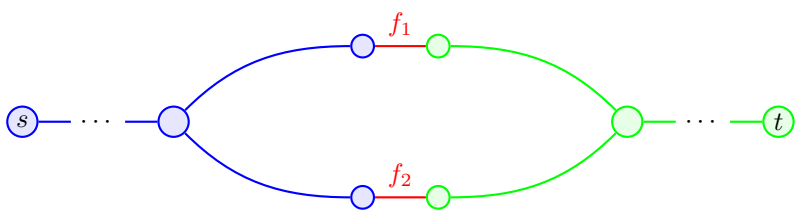

Figure 3.1: A failure set $F = \{f_1, f_2\}$ in Ω', which cuts the bipaths into two parts, $pre(F)$ on the left and $post(F)$ on the right.

Property 3.6.7. *For every $j, l \in [d]$, $l < j$ and $F \in \Omega'$ such that $e^i_j \in F$, the edge e^i_l is on the s-side of the cut F in H'.*

In fact, also the corresponding *suffix property* is satisfied: when $l > j$, the edge e^i_l is on the t-side of the cut F in H'. For every $F \in \Omega'$ denote by $pre(F)$ ($post(F)$, respectively) the subset of vertices in $V[H']$, which is on the s-side (t-side, respectively) of the cut F. A set of edges *crosses* a failure scenario F, if it contains a path from a vertex in $pre(F)$ to a vertex in $post(F)$.

In terms of the above notation, the augmentation problem $AUG(G, \Omega, S_0)$ is to find a cheapest set S_1 that crosses every failure scenario $F \in \Omega'$. It is obvious that an optimal solution S^*_1 to the augmentation problem is acyclic. Furthermore, every leaf of the forest $(V[S^*_1], S^*_1)$ is necessarily a vertex in $V[H']$. The following Lemma justifies using only unions of paths in a search for an approximate solution for $AUG(G, \Omega, S_0)$, with a potential multiplicative cost of 2 in the approximation guarantee.

Lemma 3.6.8. *Let $S^*_1 \subset E \backslash S_0$ be the optimal solution to $AUG(G, \Omega, S_0)$ with optimal cost $w(S^*_1) = w^*$. Then there exists a collection of paths $q_1, \cdots, q_r \subset E \setminus S_0$, which cross all scenarios $F \in \Omega'$ and such that $\sum_{i=1}^r w(q_i) \leq 2w^*$.*

Proof. We established that S^*_1 is a forest with leaves in $V[H']$. Consider any tree $T \subset S^*_1$. By doubling all edges in T, we obtain an Eulerian subgraph. Consider any Eulerian tour in this subgraph. We cut the tour into edge-disjoint paths in the following way. We start at some leaf of T and follow the tour until it reaches another leaf ot T. The edges traversed correspond to the first path q_1. We continue to follow the tour until the next leaf of T is reached. This part corresponds to q_2 etc.

Let $q_1, \cdots, q_r$ be the obtained paths. Clearly $\sum_{i=1}^r w(q_i) = 2w(T)$. Furthermore, every cut that was covered by T is clearly also covered by some path. By repeating the argument for every tree in S^*_1 we obtain the result.

□

We model $AUG(G, \Omega, S_0)$ as a SC problem with Ω' corresponding to the ground set and the aforementioned paths as the covering sets. A path between two vertices $u, v \in V[H']$ covers a failure $F \in \Omega'$ if $u \in pre(F)$ and $v \in post(F)$, or vise-verse. In the following lemma we describe a way to obtain a 8-approximation for this set cover problem, which we denote by $SC(G, \Omega, S_0)$.

Lemma 3.6.9. *There is a polynomial time algorithm, which accepts as input the graph H', the scenario set Ω' and the paths p_1, p_2 and outputs a 8-approximation to $SC(G, \Omega, S_0)$.*

Proof. We first compute for every distinct pair of vertices $u, v \in V[H']$ a shortest path from u to v in $(V, E \setminus S_0)$, and define a length function $\ell : \binom{V[H']}{2} \to \mathbb{R}_+$ according to these shortest path distances. The function ℓ represents the costs of the subsets in the problem $SC(G, \mathcal{F}, S_0)$. We denote the aforementioned shortest path from a node u to the node v by the *link* $\alpha_{u,v}$.

Let $s = u_1, u_2, \cdots, u_d = t$ and $s = v_1, v_2, \cdots, v_d = t$ be the vertices of p_1 and p_2, respectively, when traversing the paths from s to t. A failure F can be covered in four different *modes* by a link. The first two modes correspond to links of the type α_{u_i,u_j} and α_{v_i,v_j}, respectively. Note that those are links which connect vertices on the same path. The third mode corresponds to links of the type α_{u_i,v_j}, where $u_i \in pre(F)$ and $v_j \in post(F)$. The fourth mode corresponds to links of the type α_{v_i,u_j}, where $v_i \in pre(F)$ and $u_j \in post(F)$. Figure 3.2 illustrates this definition.

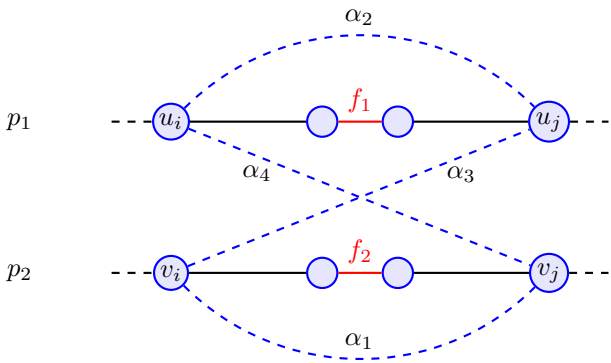

Figure 3.2: Four different modes of crossing a failure scenario $F = \{f_1, f_2\}$. The link α_i crosses F in the i'th mode for $i \in [4]$.

The key observation is that if one restricts all failures to be covered in one of the four modes, the resulting problem becomes an *interval covering (IC)* problem. For the first two modes this is obvious: The IC problem contains the faulty edges as points and the intervals stretch between pairs of vertices on the path. We prove the claim for the third mode. The proof for the fourth mode is identical. Recall that a failure F is covered in the third mode by a link α_{u_i,v_j} if $u_i \in pre(F)$ and $v_j \in post(F)$. To obtain the desired interval cover problem we first perform a certain uncrossing to Ω'. Consider two failures $F_1 = \{e^1_{i_1}, e^2_{j_1}\}$ and $F_2 = \{e^1_{i_2}, e^2_{j_2}\}$ such that $i_2 > i_1$

and $j_2 < j_1$. Clearly, if for some link α_{u_c,v_d} it holds that $u_c \in pre(F_1)$ and $v_d \in post(F_1)$ then it also holds that $u_c \in pre(F_2)$ and $u_d \in post(F_2)$, since $pre(F_1) \subset pre(F_2)$ and $post(F_1) \subset post(F_2)$. We conclude that we can simply eliminate F_2, since any solution that covers F_1 will also cover F_2. Let $\bar{\Omega} \subset \Omega'$ be the set of failures obtained after removing all redundant failures in this way. Note that for distinct $F_1 = \{e^1_{i_1}, e^2_{j_1}\}$ and $F_2 = \{e^1_{i_2}, e^2_{j_2}\}$ in $\bar{\Omega}$ it holds that $sign(i_1 - i_2) = sign(j_1 - j_2)$, thus they can be ordered according to their appearance on the paths p_1, p_2 from s to t. It is now straightforward to see that every link $\alpha_{u,v}$ crosses a set of failures in $\bar{\Omega}$, which corresponds to an interval in this order. This proves the claim. Figure 3.3 illustrates this procedure.

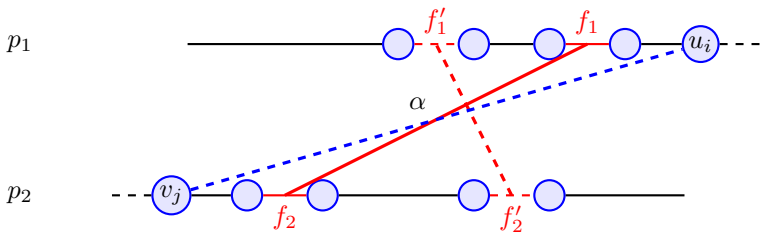

Figure 3.3: An illustration of the construction of $\bar{\Omega}$ for the fourth mode. The failure scenario $F' = \{f'_1, f'_2\}$ is redundant in the presence of the failure scenario $F = \{f_1, f_2\}$.

Consider next an IP formulation of the SC problem. The variables $x_{u,v}$ correspond to the decision whether or not to choose link $\alpha_{u,v}$ in the solution. The notation $\alpha_{u,v} \times F$ stands for 'link $\alpha_{u,v}$ crosses failure F'.

$$\min \left\{ \ell(x) : x_{u,v} \in \{0,1\}, \quad \sum_{\substack{u,v \in V[H'] \\ \alpha_{u,v} \times F}} x_{u,v} \geq 1 \quad \forall F \in \bar{\Omega} \right\} \tag{3.34}$$

Consider the fractional optimal solution x^* obtained by solving the LP relaxation of this IP. For $F \in \bar{\Omega}$ and $j \in [4]$ let A^j_F denote all pairs $\{u, v\}$ such that F is crossed in the j'th mode by $\alpha_{u,v}$. Since the union $\cup_{j \in [4]} A^j_F$ covers the set of links which cross F, it must hold that for some $j_F \in [4]$ we have

$$\sum_{\{u,v\} \in A^{j_F}_F} x^*_{u,v} \geq \frac{1}{4}.$$

We are now ready to partition the set cover problem into four problems with disjoint families of failure scenarios, $\bar{\Omega}_j$, $j \in [4]$. The set $\bar{\Omega}_j$ will simply contain all failures sets in $\bar{\Omega}$, for which $\sum_{\{u,v\} \in A_F^j} x^*_{u,v} \geq \frac{1}{4}$ holds (if for some $F \in \bar{\Omega}$ this holds for more than one value of j, one is chosen arbitrarily).

Consider next the solution $y^* = 4x^*$. For $j \in [4]$, let $B_j = \cup_{F \in \bar{\Omega}_j} A_F^j$ be the set of links that are used to fractionally cover some failure scenario in the j'th mode. From the previous discussion the following holds for every $F \in \bar{\Omega}_j$

$$\sum_{\substack{\{u,v\} \in B_j \\ \alpha_{u,v} \times F}} y^*_{u,v} \geq 1. \tag{3.35}$$

Since the SC instance restricted to elements in $\bar{\Omega}_j$ is an interval cover problem for every $j \in [4]$, it holds that there is an integral solution z_j with cost $\ell(z_j) = \sum_{\{u,v\} \in B_j} y^*_{u,v}$. This solution can be found with standard dynamic programming techniques in polynomial time. To this end the algorithm returns the union of all four solution z_j, $j \in [4]$ as the feasible solution to the SC problem. The cost of this solution can be bounded as follows.

$$\sum_{i \in [4]} \ell(z_i) \leq 4 \left(\sum_{\{u,v\} \in B_1 \cup B_2} x^*_{u,v} + 2 \sum_{\{u,v\} \in B_3 \cup B_4} x^*_{u,v} \right) \leq 8\ell(x^*) \leq 8OPT_{SC}.$$

In the last equation, the coefficient 8 is due to the fact that every link in B_3 might also appear in B_4, hence the cost in the scaled LP solution for this link might be incurred twice. The value OPT_{SC} is the optimal solution value of the SC problem. □

Lemmas 3.6.8 and 3.6.9, along with the preceding discussion give the following theorem.

Theorem 3.6.10. *There is a polynomial* 17*-approximation algorithm for BERCC(SP) with* $B = 2$.

The case $B \geq 3$ and series-parallel graphs

Our next goal is to develop a constant factor $\beta(B)$-approximation algorithm for BERCC(SP) for every constant bound $B \geq 3$, restricted to inputs with series-parallel graphs. The main idea is to extend the cut covering framework developed for the case $B = 2$. The driving motor of the algorithm is again a certain augmentation procedure, that given a previously acquired solution,

needs to obtain an approximation of the optimal augmentation of this set to a feasible solution to the BERCC(SP) problem. The number of such augmentation steps depends on the value of B. In each augmentation step, the algorithm solves a bounded number of IC problems. While the partition of the of augmentation problem into IC problems was reasonably simple in the case $B = 2$, the corresponding problem for higher values of B is significantly more involved. In fact, this issue poses the hardest technical difficulty.

Let us start with some useful definitions. We denote the set of edges obtained after the i'th augmentation step by S_i. The set S_0 is simply a shortest s-t path in G. The set of failure scenarios, which form a cut in the set S_i is denoted by Ω_i. In other words, the set of scenarios $F \in \Omega$, which satisfy that $S_i \setminus F$ contains an s-t path is exactly $\Omega \setminus \Omega_i$.

Definition 3.6.11 (redundant, near, critical and safe edges)**.** Let $S \subset E$ be a set containing an s-t path and let $F \in \Omega$. Let $red_S(F) \subset F$ be all edges $e \in F$, such that there is a path in $S \setminus F$ between the end-nodes of e. In case $S \setminus F$ contains an s-t path we let $red_S(F) = F$. Let $near_S(F) \subset F$ be all edges $e \in F$ such that there is a path in $S \setminus F$ from s to one endpoint of e, but not to the other. Let $crit_S(F) \subset near_S(F)$ be those edges $e \in near_S(F)$ for which there exists an s-t path $p_e \subset S$, such that $near_S(F) \cap p_e = \{e\}$. Finally, let $safe_S(F) = red_S(F) \cup crit_S(F)$.

Intuitively, edges in $red_S(F)$ do not represent problematic failed edges, assuming that S is contained in the solution, since in the case of a realization of the scenario F, the connectivity they provide can be substituted with a path in the solution, which is intact. The edges in $crit_S(F)$ form an s-t cut in the graph. Furthermore, they can be unambiguously directed from the node on the s-side of the cut to the other node. The invariant we would like the intermediate solution S_i to satisfy is that $|safe_{S_i}(F)| > i$ for every $F \in \Omega$ and $i \in [B]$. To this end it is clear that after B successful augmentation iterations we obtain a feasible solution to the BERCC(SP) instance. We are now ready to precisely define the augmentation problems.

Definition 3.6.12. The augmentation problem $AUG(G, \Omega, S_{i-1})$ is to find the cheapest set $A_i \subset E \setminus S_{i-1}$ such that $S_i = A_i \cup S_{i-1}$ satisfies $|safe_{S_i}(F)| > |safe_{S_{i-1}}(F)|$ for every $F \in \Omega_i$, where $\Omega^i \subset \Omega$ are all failures, which form an s-t cut in (V, S_{i-1}).

Note that $AUG(G, \Omega, S_{i-1})$ is a relaxation of the original problem, as every feasible solution for the BERCC(SP) instance, restricted to the edges not in S_{i-1} is a feasible choice for the augmenting set A_i. The following lemma

provides the required interpretation of $AUG(G, \Omega, S_{i-1})$ as a cut covering problem.

Lemma 3.6.13. *Let $A \subset E \setminus S_{i-1}$ be a set, which crosses every cut in S_{i-1} of the form $crit_{S_{i-1}}(F)$ for $F \in \Omega^i$. Then A is a feasible solution to the augmentation problem $AUG(G, \Omega, S_{i-1})$.*

Proof. To prove the claim we need to show that for $S_i = S_{i-1} \cup A$ and for every $F \in \Omega^i$ it holds that $|safe_{S_i}(F)| > |safe_{S_{i-1}}(F)|$. Note that $safe_X(F) \subset safe_Y(F)$ whenever $X \subset Y$, hence $safe_{S_{i-1}}(F) \subset safe_{S_i}(F)$ holds. We just need to prove that some other $e \in F \setminus safe_{S_{i-1}}(F)$ becomes safe after adding the edges in A. Consider some $F \in \Omega^i$ and let $C = crit_{S_{i-1}}(F)$. Since A crosses the s-t cut C, A contains a path p from some $u \in pre_{S_{i-1}}(C)$ to some vertex $v \in post_{S_{i-1}}(C)$. Let B_u and B_v be the connected components of $H_{i-1} - F$, which contain u and v respectively.

Consider the graph $Q = (W, F')$ obtained by contracting all edges in H_{i-1}, except for the edges in $F' = S_{i-1} \cap F$. We denote the vertices, which corresponds to the connected components B_u, B_v, the connected component which contains s and the connected component which contains t by u', v', s' and t', respectively. Note that all edges in $red_{S_{i-1}}(F)$ become loops in Q, and all edges in $near_{S_{i-1}}(F)$ are incident to s'. Consider a vertex $x \in W$ for which every x-t' path in Q goes through s'. Let $X \subset V[H_{i-1}]$ be the the set of vertices that were contracted to x. It is straightforward to see that $X \subset pre_{S_{i-1}}(C)$, hence v' is not such a vertex. We denote all vertices in W, which are of the aforementioned type by *left* vertices. All vertices, which are not left vertices are denoted by *right* vertices. Note that s' is a left vertex and t' is a right vertex. We henceforth view p as a path from u' to v'.

We distinguish two cases. In the first case $p \cap F \neq \emptyset$. Let $e \in p \cap F$ be the the edge in F closest to u' on p. If $u' = s'$ we have that e is a critical edge in $S_{i-1} \cup p$ and the claim is proved. Otherwise let q be any path from s' to u' in Q. Since u' is a left vertex, the first edge on this path is in $near_{S_{i-1}}(F) \setminus crit_{S_{i-1}}(F)$, and it becomes critical after adding the path p, which proves the claim in this case. The same argument works for the case $u' \neq s'$ and $p \cap F = \emptyset$.

The remaining case is $p \cap F = \emptyset$ and $s' = u'$. In this case if $v' = t'$, it holds that $H_i - F$ contains an s-t path, hence $safe_{S_i}(F) = F$ and the claim is proved. The other case is that $v' \neq t'$. Consider a v'-t' path q in Q, which does not pass through s'. Such a path exists since v' is a right vertex. Adding the path p forms the path which follows p from s' to v' and then the path q

from v' to t'. We conclude that the edge of q, which is incident to v' becomes critical in $S_{i-1} \cup p$. This concludes the proof of the lemma. $\square$

Lemma 3.6.13 justifies the following useful preprocessing step. We contract all edges of H_{i-1}, which are either in $S_{i-1} \setminus \cup_{F \in \Omega_{i-1}} F$, or faulty edges $f \in F$, which are not critical. This step is justified by the fact that we are only concerned with crossing the cuts of the form $crit_{S_{i-1}}(F)$ for $F \in \Omega_{i-1}$. To simplify matters, we allow a small abuse of notation by letting Ω_{i-1} denote the set of relevant cuts after the aforementioned contraction operation, namely the set

$$\{crit_{S_{i-1}}(F) : F \in \Omega_{i-1}\}.$$

Note that after the preprocessing step all failure scenarios in Ω_{i-1} are s-t cuts, which split the graph into exactly two connected components. We denote the resulting graph by $H = (W, M)$. To simplify notation we henceforth drop the subscript $i-1$, as the remaining part treat this augmentation step exclusively. Observe that $M = \cup_{F \in \Omega} F$.

From this point on it is useful to regard H as a directed graph. The direction of an edge $e \in M$ is defined in the following way. If $e \in F \in \Omega$ then we direct e from the node which is on the s-side of the cut F in H to the node on the t-side of this cut. The following are two easy observations about H. Firstly, H is acyclic. Indeed if there was a cycle C in H, it would mean that some cut $F \in \Omega$ is crossed twice: once in each direction. This however cannot happen, since all edges of F are directed from the s-side to the t-side of F. Secondly, every directed s-t path crosses each cut $F \in \Omega$ exactly once.

To this end recall that the algorithm for the case $B = 2$ relied on the fact that we could obtain two convenient sequences of edges, namely the paths p_1, p_2. The critical fact was that these sequences of edges satisfied the prefix and the suffix properties, on the one hand, and the they covered all edges in the current solution, on the other hand. For larger values of B it may not be possible to find a small number of such s-t paths, which cover all edges of the graph H. In fact, the size of a minimal such cover may depend on the number of vertices of the graph. Our next goal is to show that this situation cannot occur in series-parallel graphs.

Recall that a graph is called *series-parallel (SRP)* with terminal s and t if it can be composed from a collection of disjoint edges using the *series* and *parallel* compositions. The series composition of two SRP graphs with terminals s, t and s', t' respectively, takes the disjoint union of the two graphs, and identifies t with s'. The parallel composition takes the disjoint union of

the two graphs and identifies s with s' and t with t'. Given a SRP graph it is easy to obtain the aforementioned decomposition.

We will also assume that the terminals of the graph G are the vertices s and t that need to be connected by a robust path. Although this restricts further the class of instances, this assumption is realistic in many applications. In addition, the strongest hardness result for BERCC(SP), which we prove in Section 3.6.3, holds for this restricted case.

Let us proceed by stating and proving a lemma, which provides the aforementioned convenient decomposition of the graph H into s-t paths. Note that H is a directed SRP graph, since G is SRP.

Lemma 3.6.14. *H is a union of at most 2^{B-1} s-t paths. Furthermore, these paths can be obtained from H in polynomial time.*

Proof. We prove the claim by a double induction on the parameter B and on the number of edges m. In the base case $B = 1$ the graph must be a single s-t path, so the claim holds true for any m. Also, for any $B \geq 2$ and $m \leq 2^{B-1}$ the claim holds as well, since we can take one arbitrary s-t path containing every edge separately, and obtain at most 2^{B-1} paths in total.

Assume next that the claim is true for every $B' < B$ and every m, as well as for $B' = B$ and every $m' < m$. Assume first that H is a series composition of two SRP graphs H_1 and H_2, with terminals s, r and r, t, respectively. From the inductive assumption, since the number of edges in of these graphs is smaller than m, both these graphs can be obtained as a union of at most 2^{B-1} paths. Concatenating pair of paths from both these sets at the vertex r arbitrarily gives the desired decomposition for H.

Assume next that H is a parallel composition of H_1 and H_2. Let Ω_i denote the set of restrictions of cuts in Ω to edges in H_i for $i \in [2]$. Since H_1 and H_2 do not share edges, and every $F \in \Omega$ forms a cut in H, the restrictions of F in Ω_1 and Ω_2 contain at most $|F| - 1 = B - 1$ edges each. In other words we have $\max\{|F| : F \in \Omega_i\} \leq B - 1$ for $i \in [2]$. It follows from the inductive hypothesis that H_1 and H_2 are a union of at most 2^{B-2} s-t paths each. Finally, taking the union of all these paths gives at most 2^{B-1} paths.

We showed that the desired set of paths exists using an inductive argument on the structure of the graph. The algorithm to obtain these path is a recursive procedure that follows the exact same case distinction. This algorithm runs in time, which is linear in the size of the graph H. □

Assume next that a set of $R \leq 2^{B-1}$ paths $\mathcal{L} = \{P_1, \cdots, P_R\}$, the union of

which comprises H, was computed. Our next goal is to generalize Lemma 3.6.9 to our setting. Identically to the algorithm in Lemma 3.6.9, we define a SC problem, which contains the failure sets in Ω as the ground set of elements, and shortest paths in the remaining graph $(V, E \setminus E[H])$ as covering sets. The subsets of vertices $pre(F)$ and $post(F)$, the notion of links and the length function ℓ are defined as in Section 3.6.3. Again, the goal is to choose a set of links, such that for every $F \in \Omega$, there is some link $\alpha_{u,v}$ with $u \in pre(F)$ and $v \in post(F)$.

To obtain a better approximation guarantee we perform the following additional preprocessing step. We call a pair of vertices $u, v \in V[H]$ *clean* if every s-t path which is incident to u, is also incident to v, or vise-versa. We denote links which connect such pairs of vertices by *clean links*. Observe, that since G is a SRP graph, for every pair of vertices u, v, connected by a directed path in H, which is *not* clean, there exists a vertex w, such that u, w and w, v are clean pairs. At the same time we can assume that

$$\ell(u, v) \geq \ell(u, w) \quad \text{and} \quad \ell(u, v) \geq \ell(w, v),$$

since the link $\alpha_{u,v}$ covers *exactly* the set of cuts covered by $\alpha_{u,w}$ and $\alpha_{w,v}$ *together*. The SC problem we solve contains only clean links. We assume that the cost function ℓ was updated to satisfy the previous inequalities. In other words, we assume that the cost of a clean link is the minimum over the costs of all links, which cover the same set of cuts, or more. From the previous discussion it follows that the new SC problem (which contains only clean links) has an optimal objective value, which is at most twice that of the original SC problem. Let us denote the modified SC problem by $SC(H, \Omega)$.

The advantage of using the altered problem $SC(H, \Omega)$ is as follows. For every clean link $\alpha_{u,v}$, there is necessarily some path $P \in \mathcal{L}$, which contains both u and v. We use this fact to obtain a 2^{B-1}-approximation of for $SC(H, \Omega)$ in the following lemma, which is a natural generalization of Lemma 3.6.9 to the current setup.

Lemma 3.6.15. *There is polynomial 2^{B-1}-approximation algorithm for the SC problem $SC(H, \Omega)$, which accepts as input the graph H, the set Ω and the paths $\mathcal{L} = \{P_1, \cdots, P_R\}$.*

Proof. Let OPT_{SC} denote the value of the optimal solution to the problem $SC(H, \Omega)$. The algorithm proceeds analogously to the one in Lemma 3.6.9, namely by solving the LP relaxation of the natural IP formulation of the given SC problem. Let x^* be the obtained optimal fractional solution.

To this end let us make another important observation. Consider some link $\alpha_{u,v}$ and two paths $P, P' \in \mathcal{L}$, both incident to u and to v. Since H is a directed acyclic graph, the order in which the vertices appear on the paths, when traversed from s to t is the same. Let $S \subset P$ and $S' \subset P'$ be the two subpaths of P and P', respectively, connecting the vertices u and v. We claim that $|S| = |S'|$ and furthermore,

$$\{F \in \Omega : S \cap F \neq \emptyset\} = \{F \in \Omega : S' \cap F \neq \emptyset\}. \tag{3.36}$$

Both properties are the consequence of the fact that every path $P \in \mathcal{L}$ contains exactly one edge from each cut $F \in \Omega$. The fact $P \cap F \neq \emptyset$ follows from the fact that F is a cut, while $|P \cap F| \geq 2$ would contradict the fact that we reduced every failure set F to contain only critical edges. To simplify the remaining analysis we identify each clean link $\alpha_{u,v}$ with exactly one path $P[\alpha_{u,v}] \in \mathcal{L}$, which is incident to both u and v. Observe that by doing so, we define a partition of the set of all clean links into at most R parts: at most one for each path $P \in \mathcal{L}$. We can now conclude that for every $F \in \Omega$, the fractional solution x^* satisfies

$$\sum_{\alpha_{u,v} \times F} x^*_{u,v} = \sum_{P \in \mathcal{L}} \sum_{\substack{(u,v)\,:\,P=P[\alpha_{u,v}] \\ F \cap P[u,v] \neq \emptyset}} x^*_{u,v} \geq 1. \tag{3.37}$$

Note that in the previous equality we critically use the property in Equation (3.36). We use x^* to partition Ω into R sets $\Omega_1, \cdots, \Omega_R$: one for each path in $\mathcal{L}$. Equation (3.37) tells us that for every $F \in \Omega$, there is some path $P_i \in \mathcal{L}$, such that

$$\sum_{\substack{(u,v)\,:\,P_i=P[\alpha_{u,v}] \\ F \cap P_i[u,v] \neq \emptyset}} x^*_{u,v} \geq \frac{1}{R}.$$

Consequently, we include F in Ω_i. If two or more paths satisfy the previous condition for F, one path is chosen arbitrarily.

From this point on we repeat the ideas in Lemma 3.6.9. We split the problem into R IC problems, and solve each one separately. The cost of the solution Rx^*, restricted to the links associated with a given path P_i gives an upper bound on the cost of the solution to the corresponding IC problem. Consequently, the cost of the the union of the solutions to all IC problems is bounded from above by the cost of Rx^*, which in turn, is at most $2^{B-1}OPT_{SC}$. Since this solution is clearly feasible, the lemma is proved. □

We summarize the main result of this section in the following theorem.

Theorem 3.6.16. *There is a polynomial* $(1 + B2^{B+1})$*-approximation algorithm for BERCC(SP), restricted to SRP graphs.*

Proof. Let OPT denote the cost of the optimal solution. The algorithm starts by computing a shortest s-t path S_0 in G. Clearly, this path has a cost of at most OPT. Then the algorithm performs at most B augmentation steps. Each such step increases the cost of the current solution by at most $2^{B+1}OPT$. This is a direct consequence of Lemma 3.6.8, Lemma 3.6.15 and the fact that problem $SC(H, \Omega)$ has an optimal cost, which is bounded by twice the cost of the original SC problem, which is defined with all links. Lemma 3.6.13 guarantees that the solution at hand after B augmentation steps is a feasible solution. Since the cost of this solution is at most $(1+B2^{B+1})OPT$, the proof is complete. □

Remark 3.6.17. The approximation guarantee of Theorem 3.6.16 can be improved further in the following way. In the i'th augmentation step, we can restrict the augmentation to cuts F, which satisfy $|safe_{S_{i-1}}(F)| = i$. In this way we can still ensure $|safe_{S_i}(F)| \geq i + 1$ for all $F \in \Omega$, in the end of this augmentation step, which guarantees the feasibility of the solution after B such steps. At the same time, since the maximal cardinality of a cut in the i'th augmentation step is bounded by $|crit_{S_{i-1}}(F)| \leq |safe_{S_{i-1}}(F)| \leq i$, we can bound the total cost of this augmentation step by 2^{i+1} (instead of 2^{B+1}). This adaptation gives the approximation factor $1 + \sum_{i=1}^{B} 2^{i+1} = 2^{B+2}$.

Complexity of BERCC(SP)

This section presents two hardness-of-approximation results for BERCC(SP). For $B \geq 2$ we show that BERCC(SP) is APX-hard via a reduction from VC on cubic graphs. For $B \geq 3$ we show that BERCC(SP) is hard to approximate within a factor $\frac{B}{4} - \epsilon$ unless P=NP, and $\frac{B}{2} - \epsilon$ under the Unique Games Conjecture (UGC) via a reduction from the Hypergraph Vertex Cover (HVC) problem. Let us start by defining a key notion.

Definition 3.6.18. A graph $G = (V, E)$ is *r-monotone*, if its edge set can be written as a union of r not necessarily disjoint subsets $E_1, \cdots, E_r \subset E$ with r respective orders $\pi_1, \cdots, \pi_r$ of the subsets of edges, such that for every vertex $v \in V$, the set of edges incident to v is an interval in at least one of the orders.

Note that every n-vertex graph is n-monotone, every t-colourable graph is t-monotone etc. In the following lemma we prove that instances of VC on B-monotone graphs can be modeled as instances of BERCC(SP), given a certificate of B-monotonicity.

Lemma 3.6.19. *Given a B-monotone graph $G = (V, E)$ with a corresponding collection of ordered subsets $E_1, \cdots E_B \subset E$, and $\pi_1, \cdots, \pi_B$, there is a polynomial time transformation of the Minimum VC instance on G into an instance of BERCC(SP) with identical optimal objective values.*

Proof. We construct an instance (H, Ω) of BERCC(SP) in the following way. We describe first the graph H. We start by including in H B vertex-disjoint s-t paths P_i, $i \in [B]$, with $|P_i| = |E_i| + 1$. The first edge in the path P_i (the edge incident to s) is special and it is denoted by α_i. The remaining edges of P_i are associated to edges in E_i, such that the j'th edge on P_i is associated with $\pi_i(j-1)$ for $2 \leq j \leq |E_i| + 1$. All edges on the paths P_i have cost zero.

We associate every vertex $v \in V$ with an index $j(v) \in [B]$, which corresponds to the ordered subset of edges $E_{j(v)}$, which contains the set of incident edges $E_v = \{e \in E : v \in e\}$ as an interval, and denote by $beg(v)$ and $end(v)$ the end vertices of the corresponding interval on $P_{j(v)}$. We conclude the construction of H by adding the direct edge β_v from $beg(v)$ to $end(v)$ with cost one, for every vertex of the graph G.

We describe the failure scenarios next. Let $e = \{u, v\}$ be some edge with $a = j(u)$, $b = j(v)$ and $I_e = [B] \setminus \{a, b\}$. The BERCC(SP) instance will contain a failure scenario F_e for e, which will contain the edge f_a, which is associated with e on P_a, the edge f_b, which is associated with e on P_b, and all edges α_i with $i \in I_e$. Observe that it is possible that $a = b$ and $f_a = f_b$. In any case, we have $|F_e| = B$, as required. This completes the description of the transformation, which is illustrated in Figure 3.4

We claim that the VC instance has optimal value d if and only if the BERCC(SP) instance has optimal value d. Consider a vertex cover C of size d. We claim that the solution

$$S = \bigcup_{i \in [B]} P_i \cup \{\beta_u : u \in C\}$$

is feasible for the BERCC(SP) instance. Consider any failure set $F_e \in \Omega$, corresponding to some edge $e \in E$. Since C is a vertex cover, there is some $w \in C$ with $w \in e$. We assume without loss of generality that $j(w) = 1$. Observe that the path starting at s, following P_1 until $beg(w)$, then taking the edge β_w to the vertex $end(w)$, and then following P_1 until t is intact in S. Also note that the cost of S is exactly d.

Conversely, consider a feasible solution S for BERCC(SP) with cost d. Let

$$C = \{v \in V : \beta_v \in S\}.$$

We claim that C is a vertex cover of G. Consider any edge $e = \{z, w\} \in E$, and consider its associated failure scenario F_e. Assume first that $j(z) \neq j(w)$, and assume without loss of generality that e is associated with one edge f_1 on P_1 and another edge f_2 on P_2. It follows that $F_e = \{f_1, f_2\} \cup \{\alpha_i : 3 \leq i \leq B\}$. Observe that the set of edges $\{f_1, f_2, \beta_z, \beta_w\} \cup \{\alpha_i : 3 \leq i \leq B\}$ is an s-t cut in H. It follows that either $\beta_z \in S$ or $\beta_w \in S$, since $S \setminus F_e$ contains an s-t path. We conclude that either $w \in C$ or $z \in C$. The case $j(z) = j(w)$ is similar. This completes the proof. □

To this end recall that VC remains APX-hard, even when the graphs are restricted to be cubic, as was shown by Alimonti and Kann [9]. To prove that BERCC(SP) is APX-hard for $B = 2$ we need to prove that cubic graphs are 2-monotone, and that the certificate can be computed from the graph in polynomial time. We prove this fact in the following lemma.

Lemma 3.6.20. *Every cubic graph is 2-monotone. Furthermore, a corresponding pair of ordered sets of edges $E_1, E_2 \subset E$ can be obtained from G in polynomial time.*

Proof. Let $G = (V, E)$ be a cubic graph. We describe a procedure to obtain the ordered sets E_1, E_2. We start by obtaining a partition of the vertex set $V = V_1 \cup V_2$, which is obtained iteratively. We initialize by setting $V_1 = V$ and $V_2 = \emptyset$. We stop the procedure when it is true that for every $i \in [2]$ and for every $v \in V_i$, the number of neighbors of v in V_i is at most one. At every iteration we choose any vertex $v \in V_i$ for some $i \in [2]$, which violates the condition. Assume without loss of generality that $i = 1$. Since G is a cubic graph, v must have either two or three neighbors in V_1. In the former case, v might have a single neighbor in V_2, and in the latter case v has no neighbors in V_2. Consider moving v to V_2. From the latter argumentation, it follows that v is not a violating vertex anymore. Furthermore, this operation did not create a new violating vertex. We conclude that the number of violating vertices decreased after the aforementioned operation. The number of iterations needed to eliminate all violating vertices from the partition is, hence, at most $|V|$. Let V_1, V_2 be any final partition. To this end we define E_1 and E_2 to be the sets of edges incident to V_1 and V_2, respectively. Note that $E = E_1 \cup E_2$ holds. Let $\delta \subset E$ denote the edges in the cut with shores V_1 and V_2.

It remains to define the desired orders on the sets E_1 and E_2. To do so we first order the vertices in V_i ($i \in [2]$) such that if $u, v \in V_i$ and $\{u, v\} \in E$, then u

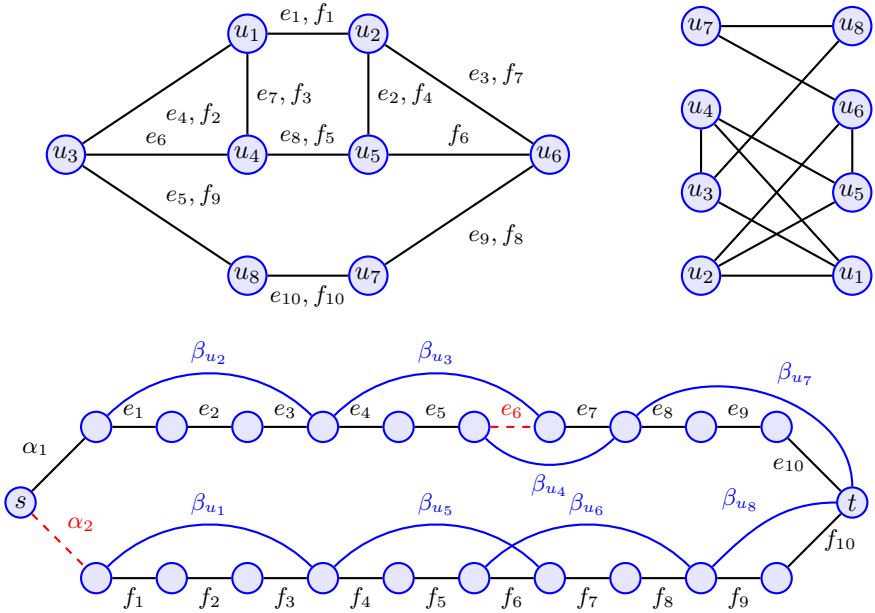

Figure 3.4: Top left: A cubic graph G. The edge set is covered by two subsets $E_1 = \{e_1, \cdots, e_{10}\}$ and $E_2 = \{f_1, \cdots, f_{10}\}$. The numbering of the edges provides the two corresponding orders, which are a certificate that the graph is 2-monotone. Top right: The partition into two sets V_1' and V_2' in Lemma 3.6.20, which gives rise to the ordered sets E_1, E_2. Bottom: the transformation in Lemma 3.6.19 corresponding to the 2-monotone graph G. The failure set F_e with $e = \{u_3, u_4\}$ is highlighted with dashed edges.

and v appear one after the other in the order. This can easily be achieved, since the induced graphs $G[V_i]$ are matchings. Let $\sigma_i = u_1^i, \cdots, u_{|V_i|}^i$ for $i \in [2]$ be the corresponding orders. We define the order on E_i in the following way. The first edges in the order are the edges incident to u_1^i, which cross the cut δ. Then we add the edge between u_1^i and u_2^i, if it exists. Then we add all edges which are incident to u_i^2, which cross the cut δ, etc. Clearly, the edges incident to a vertex $u \in V_i$ appear as an interval in the ordering of the set of edges E_i. See Figure 3.4 for an illustration.

□

In light of Lemma 3.6.19 and Lemma 3.6.20 we obtain the desired result for

$B \geq 2$.

Theorem 3.6.21. *BERCC(SP) is APX-hard for any $B \geq 2$.*

Remark 3.6.22. In the reduction in Theorem 3.6.21 we used a weighted graph. We remark that the reduction can be adapted to show APX-hardness of the unweighted case. This can be achieved by setting $D = \sum_{i=1}^{B} |P_i|$, and subdividing every edge β_u, not in one of the paths $\zeta(D)$ times for any sufficiently large polynomial ζ. This adaptation provides that the cost of any detour is much more expensive than the number of edges on all the paths P_i, $i \in [2]$ (concretely, $\zeta(D)$ times more expensive), hence we can assume again that all paths P_i are part of any optimal solution, potentially losing only a sub-constant (polynomial) fraction in optimality.

In fact, the result of Theorem 3.6.21 can be stated for the restricted class of SRP graphs in the case $B \geq 3$.

Theorem 3.6.23. *BERCC(SP) restricted SRP graphs is APX-hard for any $B \geq 3$.*

Proof. Consider the graph obtained in the proof of Lemma 3.6.20. The partition $V = V_1 \cup V_2$ was such that if V_1, V_2 is a bi-partition of the cubic graph (namely that the induced graphs $G[V_1]$ and $G[V_2]$ have no edges), then the resulting graph in the instance of BERCC(SP) is SRP. In fact, we established that the edges of $G[V_1]$ and $G[V_2]$ are matchings. Consider any set of vertices V_3, which contains exactly one vertex from $V = V_1 \cup V_2$, which is incident to each edge in either $G[V_1]$ and $G[V_2]$. Clearly, the partition $V = (V_1 \setminus V_3) \cup (V_2 \setminus V_3) \cup V_3$ is a tri-partition of the input graph. To this end, repeating the construction in Lemma 3.6.19 with three edge sets $E_1 = \{e : e \ni v \in (V_1 \setminus V_3)\}$, $E_2 = \{e : e \ni v \in (V_2 \setminus V_3)\}$ and $E_3 = \{e : e \ni v \in V_3\}$ results in an instance of BERCC(SP) with $B = 3$ and a SRP graph. □

To obtain the second hardness result we use a result of Guruswami and Saket [44], which states that it is NP-hard to approximate HVC within a factor $\frac{m}{4} - \epsilon$ and it is UGC-hard to approximate HVC within a factor $\frac{m}{2} - \epsilon$ for any $\epsilon \geq 0$, even when restricted to m-uniform and m-partite hypergraphs. Recall that a hypergraph is m-uniform if every hyperedge contains exactly m vertices, and it is m-partite if the vertices can be partitioned into m sets, such that no two vertices in the same set appear simultaneously in some hyperedge. Critical to our reduction is the fact that all aforementioned hardness results

hold even if the m-partition of the hypergraph is given in the input. Consequently, we will assume that every instance of HVC supplies this partition. The following lemma is a simple generalization of Lemma 3.6.19 to the setup of hypergraphs.

Lemma 3.6.24. *Given a B-uniform B-partite hypergraph H, there is a polynomial time transformation of the HVC instance on H into an instance of BERCC(SP) with identical optimal objective values.*

Proof. The construction is exactly the same as in Lemma 3.6.19, with one difference. Instead of the sets of edges in the monotone decomposition of the graph in Lemma 3.6.19, here we use a similar partition of the hyperedges defined as follows. Consider the B-partition of the vertex set $V = \cup_{i\in[B]} V_i$ of H. The set of hyperedges E_i contains all hyperedges incident to some vertex in V_i. Having fixed an arbitrary order for each vertex set V_i, the order on the set E_i is defined in the natural way: The first hyperedges in the order are those incident to the first vertex in V_i, in any order. Those are followed by the hyperedges incident to the second vertex in V_i etc. The rest of the construction is identical to the one in Lemma 3.6.19. □

The consequence of Lemma 3.6.24 is the desired result, summarized in the following theorem.

Theorem 3.6.25. *It is NP-hard to approximate BERCC(SP) within a factor $\frac{B}{4} - \epsilon$, and it is UGC-hard to approximate BERCC(SP) within a factor $\frac{B}{2} - \epsilon$ for every $\epsilon > 0$, even when the input graphs are restricted to be SRP, for any $B \geq 3$.*

Proof. This is immediate from Lemma 3.6.24. The fact that the graph obtained in the reduction is series-parallel is obvious from the fact that no hyperedge contains two vertices from any set of vertices in the partition. □

We conclude by stating the only special case of BERCC(SP), for which the complexity is open, namely the case $B = 2$, restricted to SRP graphs. Settling the complexity of this problem is an interesting direction for future research.

3.7 SIRCC(SP)

This section studies the SIRCC(SP) problem. We denote a variant of SIRCC(SP), restricted to a fixed k by k-SIRCC(SP). Recall that k-SIRCC(SP) is a special

case of BERCC(SP) with the bound $B = k$. As we showed before, the problem BERCC(SP) becomes APX-hard for $B \geq 2$. In the following discussion we analyze the complexity of the more restricted k-SIRCC(SP) problem, as well as the general SIRCC(SP) problem (when k is part of the input) with, and without restrictions of the class of input graphs. The class of SRP graphs is of particular interest, both because it is an important class of graphs from the application point of view, and it gives rise to a separation of complexity: while BERCC(SP) is APX-hard for $B \geq 3$ when restricted to SRP graphs, the SIRCC(SP) problem admits a polynomial algorithm for variable k. Let us start by recalling the exact definition of SIRCC(SP).

SIRCC(SP):

Input: A graph $G = (V, E)$, two vertices $s, t \in V$, an integer $k \in \mathbb{Z}_+$ and a set of edges $M \subset E$.

Problem: Find a minimum cost $S \subset E$, such that for every $F \subset M$ with $|M| \leq k$, the graph $(V, S \setminus F)$ contains an s-t path.

Clearly, IRCC($\mathcal{P}$) is SIRCC($\mathcal{P}$) with $M = A$ and BERCC($\mathcal{P}$) with $B = 1$ is SIRCC($\mathcal{P}$) with $k = 1$. The fact that the class SIRCC is a generalization of the natural and well-studied class IRCC makes it particularly interesting. Unfortunately, this class contains significantly harder robust counterparts. In particular, we will show that SIRCC(SP) in directed graphs contains the SC problem, and hence admits no constant factor approximation algorithms. This is in contrast to the polynomial problem IRCC(SP).

3.7.1 The case $k = 1$

We focus first on the case $k = 1$. This problem is the same as BERCC(SP) with $B = 1$, which is the only variant of BERCC(SP), solvable in polynomial time. The following lemma gives the key observation leading to a polynomial algorithm for SIRCC(SP) with $k = 1$. In fact, this lemma reveals the structure of any optimal solution of SIRCC(SP) for any k.

Lemma 3.7.1. *Let X^* be an optimal solution to SIRCC(SP) on the instance (G, M, k). Let $H' = (V', X')$ be the graph obtained by contracting all edges of $X^* \setminus M$ in $H^* = (V, X^*)$. Let s' and t' be the vertices in V', to which s and*

t were contracted, respectively. Then either $s' = t'$, or there are $k+1$ edge disjoint s'-t' paths in H'.

Proof. If $X^* \subset E \setminus M$ then $s' = t'$. Assume this is not the case, namely that $X^* \cap M \neq \emptyset$. Observe that H' must contain at least $k+1$ edge disjoint s'-t' paths: indeed if there was a cut of cardinality k or less in H, this would also be a cut in H^*. Since $X' \subset M$, there is a failure scenario $F \in \Omega(M, k)$, which contains all these edges. □

To this end recall the notion of a bipath, defined in Section 3.6.3. In our algorithm for BERCC(SP) treated in that section, we claimed that an optimal solution to a BERCC(SP) instance with $B = 1$ (which is identical to the SIRCC(SP) problem with $k = 1$) is necessarily a bipath. We prove this claim next. In the context of 1-SIRCC(SP), we call a bipath $Q = P_1 \cup P_2$ *robust*, if it holds that $P_1 \cap P_2 \cap M = \emptyset$. Note that every robust s-t bipath Q in G is a feasible solution to the SIRCC(SP) instance. Indeed, consider any failure edge $e \in M$. Since $e \notin P_1 \cap P_2$ it holds that either $P_1 \subset Q - e$, or $P_2 \subset Q - e$. It follows that $Q - e$ contains some s-t path. The next lemma shows that every feasible solution to the 1-SIRCC(SP) contains a robust s-t bipath.

Lemma 3.7.2. *Every feasible solution S^* to the 1-SIRCC(SP) contains an s-t bipath.*

Proof. We assume without loss of generality that S^* is a minimal feasible solution with respect to inclusion. Let $Y \subset S^*$ be the set of bridges in (V, S^*). From feasibility of S^*, we have $Y \cap M = \emptyset$. Consider any s-t path P in S^*. Let $u_1, \cdots, u_r$ be be the set of vertices incident to $Y = P \cap Y$. Let u_i and u_{i+1} be such that $u_i u_{i+1} \notin Y$. (if such an edge does not exist, we have $Y = P$, which means that P is a robust s-t bipath). Note that S^* must contain two edge-disjoint u_i-u_{i+1} paths L_1, L_2. Taking as the set Y together with all such pairs of paths L_1, L_2 results in a robust bipath. □

We can conclude from the previous discussion and Lemma 3.7.2 that all minimal feasible solutions to the 1-SIRCC(SP) instance are robust bipaths. This observations leads to the simple algorithm in the following theorem.

Theorem 3.7.3. *There is a polynomial algorithm for 1-SIRCC(SP).*

Proof. To solve 1-SIRCC(SP) we need to find the minimum cost robust s-t bipath. To this end let us define two length functions $\ell_1, \ell_2 : V^2 \to \mathbb{R}_+$. For two vertices $u, v \in V$ let $\ell_1(u, v)$ denote the shortest path distance

from u to v in the graph $(V, E \setminus M)$, and let $\ell_2(u,v)$ denote the cost of the shortest cycle in G containing u and v. Clearly, both length functions can be computed in polynomial time (e.g. using flow techniques). Finally, set $\ell(u,v) = \min\{\ell_1(u,v), \ell_2(u,v)\}$. Construct the complete graph on the vertex set V and associate the length function ℓ with it. Observe that by definition of ℓ, any s-t path in this graph corresponds to a robust s-t bipath with the same cost, and vise versa. It remains to find the shortest s-t bipath by performing a single shortest s-t path in the new graph. For every edge (u,v) in this shortest path, the optimal bipath contains the shortest u-v path in $(V, E \setminus M)$ if $\ell(u,v) = \ell_1(u,v)$, and the shortest cycle containing u and v in G, otherwise. □

3.7.2 Hardness for large k

Consider next the case $k \geq 2$. We start by exhibiting the complexity of SIRCC(SP). The following theorem proves that SIRCC(SP) on directed acyclic graphs contains the SC problem, and it is NP-hard when graphs are undirected.

Theorem 3.7.4. *Assuming that NP $\not\subset$ DTIME($n^{\log\log n}$), there is no polynomial $c \log k$-approximation algorithm for SIRCC(SP) on directed acyclic graphs for every $c < 1$. SIRCC(SP) on undirected graphs is NP-hard.*

Proof. We describe the reduction for directed graphs first. Consider and instance $I = (S, \mathcal{F})$ of SC with ground set $S = \{a_1, \cdots, a_n\}$ and a family $\mathcal{F} = \{A_1, \cdots, A_m\}$ of subsets of S. We construct an equivalent instance of SIRCC(SP) with a directed graph as follows. The graph G contains $m+n+2$ vertices $s, t, u_1, \cdots, u_n, v_i, \cdots, v_m$. The source s is connected by an edge to u_i for every $i \in [n]$. For every $i \in [n]$ and $j \in [m]$ with $a_i \in A_j$ we add the edge $u_i v_j$. Finally, for every $j \in [m]$ we add the edge $v_j t$. The cost of all edges are zero, except for the m edges $v_j t$, which have a cost of one. The set of faulty edges is chosen to be $M = \{su_i : i \in [n]\}$ and we set $k = n - 1$. Observe that the cost of a solution is simply the number of edges of the type $v_j t$ it contains. Moreover, the choice of k allows to remove all edges connected to s, except for any one edge su_i. Note that the obtained graph is indeed acyclic.

Consider any feasible solution X to the SC instance. We obtain a corresponding solution Y to the SIRCC(SP) instance by selecting all edges in G, except for the edges $v_j t$, such that $A_j \notin X$. Clearly, this solution has the same objective value as X. To see that this solution is feasible consider any failure set F, which fails all edges $su_{i'}$, except for the edge su_i. From feasibility of

X, we know that for some $j \in [m]$ and $A_j \in X$, we have $a_i \in A_j$. Hence, we can conclude that the path s, u_i, v_j, t is intact in $Y \setminus F$.

Conversely, if Y is a feasible solution to the SIRCC(SP) instance we set X to be all sets A_j such that $v_j t \in Y$. Consider some element a_i. The failure set F_i, which fails all edges in M, except for su_i does not disconnect some path s, u_i, v_j, t. Therefore, $a_i \in A_j$ and $A_j \in X$, as required.

The modification for undirected graphs is as follows. The instance is constructed in exactly the same way, except that edges are undirected. Also, the cost of the edges $u_i v_j$ is set to a large constant $D > m$. We claim that the CS instance has optimal cost T if and only if the SIRCC(SP) instance has optimal value $nD + T$. To this end observe that at least n of the edges $u_i v_j$ need to be selected in any feasible solution of the SIRCC(SP) instance: at least one incident to every vertex u_i. Since $D > m$, we can assume that an optimal solution always chooses n such edges. It is now obvious that a SC solution with cost T has a corresponding SIRCC(SP) solution of cost $nD + T$. To see the converse consider any optimal solution with cost $nD + T$ to the SIRCC(SP) instance and let X be the solution to the SC instance, which chooses A_j if and only if $v_j t$ is in the SIRCC(SP) solution. We claim that this is a cover. Consider any element a_i and the corresponding failure scenario F, which fails all edges $su_{i'}$, but the edge su_i. The SIRCC(SP) solution has an s-t path not using the edges in F. We claim that this path has the form s, u_i, v_j, t, which means that $a_i \in A_j \in X$, as required. Indeed, if this is not the case, the s-t path needs to cross the cut, which contains the edges $u_i v_j$ more than once, before reaching t. This means that some vertex u_r is incident to at least two edges $u_r v_{j_1}$ and $u_r v_{j_2}$, contradicting the fact that every such vertex needs to be incident to at most one edge in any optimal solution to the SIRCC(SP) instance. This finishes the proof.

□

We remark that the result in Theorem 3.7.4 can be adapted to the unweighted case by suitably subdividing the edges with positive cost in the reduction.

3.7.3 Fractional SIRCC(SP)

Note that the general SIRCC(SP) problem is not a special case of ERCC(SP), hence we did not describe an approximation algorithm for this problem up to now. The LP methods developed in Section 3.3 fail since the separation problem of the LP resulting in the application of Proposition 3.3.6 turns out

to be an NP-hard interdiction problem. In fact, the fractional variant of SIRCC(SP) poses a challenge in itself. Let us define this problem formally.

Definition 3.7.5. The fractional SIRCC(SP), denoted FRAC-SIRCC(SP), is given an instance comprising a weighted graph G with terminals s and t, a set of faulty edges M and an integer k, find an assignment of capacities $x \in [0,1]^E$ to the edges of the graph, such that for every $F \subset M$ with $|F| \leq k$, the maximum s-t flow in the graph G with capacities x^F defined as

$$x_e^F = \begin{cases} 0 & \text{if } e \in F \\ x_e & \text{otherwise.} \end{cases} \tag{3.38}$$

is at least one, so as to minimize $w(x)$.

Clearly, SIRCC(SP) is FRAC-SIRCC(SP) with the additional constraint that x should be integral, hence it is indeed a fractional relaxation. A natural question that relates SIRCC(SP) and FRAC-SIRCC(SP) is the integrality gap with respect to this relaxation. In the following theorem we prove a tight bound of $k+1$ on the integrality gap. Recall that the integrality gap is the maximal ratio between the optimal integral solution value and the optimal value of its relaxation.

Theorem 3.7.6. *The integrality gap of SIRCC(SP) is bounded by $k+1$. Furthermore, there is a family of instances with integrality gap arbitrarily close to $k+1$.*

Proof. Consider an input of $I = (G, M, k)$ to SIRCC(SP). Let x^* denote an optimal solution to the corresponding FRAC-SIRCC(SP) instance, and let $OPT = w(x^*)$ be its cost. Define a vector $y \in \mathbb{R}^E$ as follows.

$$y_e = \begin{cases} (k+1)x_e^* & \text{if } e \notin M \\ \min\{1, (k+1)x_e^*\} & \text{otherwise.} \end{cases} \tag{3.39}$$

Clearly, it holds that $w(y) \leq (k+1)OPT$. We claim that every s-t cut in G with capacities y has capacity of at least $k+1$. Consider any such cut $C \subset E$, represented as the set of edges in the cut. Let $M' = \{e \in M : x_e^* \geq \frac{1}{k+1}\}$ denote the set of faulty edges attaining high fractional values in x^*. Define $C' = C \cap M'$. If $|C'| \geq k+1$ we are clearly done. Otherwise, assume $|C'| \leq k$. In this case consider the failure scenario $F = C'$. Since x^* is a feasible solution it must hold that

$$\sum_{e \in C \setminus C'} x_e^* \geq 1. \tag{3.40}$$

Since for every edge $e \in C \setminus C'$ it holds that $y_e = (k+1)x_e^*$ we obtain

$$\sum_{e \in C \setminus C'} y_e \geq k+1, \tag{3.41}$$

as desired. From our observations it follows that maximum flow in G with capacities y is at least $k+1$. Finally, consider the minimum cost $(k+1)$-flow z^* in G with capacities defined by

$$c_e = \begin{cases} k+1 & \text{if } e \notin M \\ 1 & \text{otherwise.} \end{cases} \tag{3.42}$$

From integrality of c and the IMCF problem we can assume that z^* is integral. Note that $y_e \leq c_e$ for every $e \in E$, hence any feasible $(k+1)$-flow with capacities y is also a feasible $(k+1)$-flow with capacities c. From the previous observation it holds that $w(z^*) \leq w(y) \leq (k+1)OPT$. From Lemma 3.7.1 we know that z^* is a feasible solution to the SIRCC(SP) instance. This concludes the proof of the upper bound of $k+1$ for the integrality gap.

To prove the same lower bound we provide an infinite family of instances, containing instances with integrality gap arbitrarily close to $k+1$. Consider a graph with $D \gg k$ parallel edges with unit cost connecting s and t, and let $M = E$. The optimal solution to SIRCC(SP) on this instance chooses any subset of $k+1$ edges. At the same time, the optimal solution to FRAC-SIRCC(SP) assigns a capacity of $\frac{1}{D-k}$ to every edge. This solution is feasible, since in every failure scenario, the number of edges that survive is at least $D-k$, hence the maximum s-t flow is at least one. The cost of this solution is $\frac{D}{D-k}$. Taking D to infinity yields instances with integrality gap arbitrarily close to $k+1$. □

We remark that Theorem 3.7.6 can be easily generalized to the SIRCC(IMCF) problem. In fact, it is likely that the argument will work for many more problems, which admit a representation as a CIP. In addition, the proof of Theorem 3.7.6 implies a simple $(k+1)$-approximation algorithm for SIRCC(SP). This algorithm simply solves the IMCF problem, defined in proof of the theorem, and returns the optimal integral flow z^* as the solution. As we remarked before, this result can be directly extended to the SIRCC(IMCF) problem.

Theorem 3.7.7. *There is a polynomial $(k+1)$-approximation algorithm for SIRCC(SP).*

3.7.4 Series-parallel graphs

To this end let us switch our attention to the special case of SIRCC(SP) and FRAC-SIRCC(SP), which correspond to SRP graphs. As in Section 3.6.3, we will assume that s and t are the terminals of the input graph G.

Consider the SIRCC(SP) problem first. The algorithm we present has linear running time whenever the robustness parameter k is fixed. The algorithm is given as Algorithm 4. In fact, the algorithms computes the optimal solutions $S_{k'}$ for all parameters $0 \leq k' \leq k$. The symbol $\perp$ is returned if the problem is infeasible.

Theorem 3.7.8. *Algorithm 4 returns an optimal solution to the SIRCC(SP) problem on SRP graphs. The running time of Algorithm 4 is $O(nk)$.*

Proof. The proof of correctness is by induction on the depth of the recursion in Algorithm 4. Clearly the result returned by Algorithm 4 in lines 1-6 is optimal. Assume next that the algorithm computed correctly all optimal solutions for the subgraphs H_1, H_2, namely that for every $i \in [2]$ and $j \in [k]$, the set S_j^i computed in lines 7-8 is an optimal solution to the problem on instance $\mathcal{I}_i^j = (H_i, M \cap E[H_i], j)$.

Assume first that G is a series composition of H_1 and H_2, and let $0 \leq i \leq k$. If either $S_i^1 = \perp$ or $S_i^2 = \perp$ the problem with parameter i is clearly also infeasible, hence the algorithm works correctly in this case. Furthermore, since G contains a cut vertex (the terminal node, which is in common to H_1 and H_2), a solution S to the problem is feasible for G if and only if it is a union of two feasible solutions for H_1 and H_2. From the inductive hypothesis it follows that S_i is computed correctly in line 14.

Assume next that G is a series composition of H_1 and H_2. Consider any feasible solution S' to the problem on G with parameter i. Let S'_1 and S'_2 be the restrictions of S' to edges of H_1 and H_2 respectively, and let n_1 and n_2 be the maximal integers such that S'_1 and S'_2 are robust paths for H_1 and H_2 with parameters n_1 and n_2, respectively. Observe that $i \leq n_1 + n_2 + 1$ must hold. Indeed if this would not be the case, then taking any cut with $n_1 + 1$ edges in S'_1 and another cut with $n_2 + 1$ edges in S'_2 yields a cut with $n_1 + n_2 + 2$ edges in G, contradicting the fact that S' is a robust path with parameter i. We conclude that the algorithm computes S_i correctly in line 23. Finally note that the union any two robust paths for the graphs H_1 and H_2 with parameters n_1 and n_2 with $i \leq n_1 + n_2 + 1$ yield a feasible solution S_i. It follows that the minimum cost such robust path is obtained as a minimum

Algorithm 4 *Input: An instance* (G, M, k) *of SIRCC(SP). Output:* $(S_0, \cdots, S_k)$ *- the optimal solutions for the parameters* $0, 1, \cdots, k$.

```
if E = {e} ∧ e ∈ M then
    Return ({e}, ⊥, ··· , ⊥)
end if
if E = {e} ∧ e ∉ M then
    Return ({e}, ··· , {e})
end if
⇒ G is a composition of H_1, H_2.
(S_0^1, ··· , S_k^1) ← Algorithm 4(H_1, M ∩ E[H_1], k)
(S_0^2, ··· , S_k^2) ← Algorithm 4(H_2, M ∩ E[H_2], k)
if G is a series composition of H_1, H_2 then
    for i = 0, ··· , k do
        if S_i^1 = ⊥ ∨ S_i^2 = ⊥ then
            S_i ← ⊥
        else
            S_i ← S_i^1 ∪ S_i^2
        end if
    end for
end if
if G is a parallel composition of H_1, H_2 then
    m_1 ← max{i : S_i^1 ≠ ⊥}
    m_2 ← max{i : S_i^2 ≠ ⊥}
    for i = 0, ··· , k do
        if i > m_1 + m_2 + 1 then
            S_i ← ⊥
        else
            r ← argmin_{-1≤j≤i}{w(S_j^1) + w(S_{i-j-1}^2)}        // S_{-1}^1 = S_{-1}^2 := ∅
            S_i ← S_r^1 ∪ S_{i-r-1}^2
        end if
    end for
end if
Return (S_0, ··· , S_k)
```

cost of a union of two solutions for H_1 and H_2, with robustness parameters j and $i-j-1$ for some value of j. To allow $S_i = S_i^1$ or $S_i = S_i^2$ we let j range from -1 to k and set $S_{-1}^1 = S_{-1}^2 = \emptyset$. This completes the proof of correctness.

To prove the bound on the running time, let $T(m,k)$ denote the running time of the algorithm on a graph with m edges and robustness parameter k. We assume that the graph is given by a hierarchical description, according to its decomposition into single edges. The base case obviously takes $O(k)$ time. Furthermore we assume that the solution $(S_0, \cdots, S_k)$ is stored in a data structure for sets, which uses $O(1)$ time for generating empty sets and for performing union operations. If the graph is a series composition then the running time satisfies $T(m,k) \leq T(m',k) + T(m-m',k) + O(k)$ for some $m' < m$. If the graph is a parallel composition, then $T(m,k)$ satisfied the same inequality. We assume that the data structure, which stores the sets S_i also contains the cost of the edges in the set. This value can be easily updates in time $O(1)$ when the assignment into S_i is performed. It follows that $T(m,k) = O(mk) = O(nk)$ as required. □

Let us consider next the problem FRAC-SIRCC(SP) on SRP graphs. For a SRP subgraph H of a SRP graph G and a vector $x \in [0,1]^E$ let $MF_F(H,x)$ denote the value of the maximum flow in $H-F$ between the terminals of H, when the capacities are defined by x. We define the robust value $\alpha_i(H,x)$ with parameter $i \in [k]$ as the number

$$\alpha_i(H,x) = \min_{F \subset M,\, |F| \leq i} MF_F(H,x).$$

The following proposition provides a useful characterization of solutions to FRAC-SIRCC(SP) on SP graphs with respect to the functions α_i.

Proposition 3.7.9. *Let $x \in [0,1]^E$ be a fractional vector, and let $\mathcal{I} = (G = (V,E), M, k)$ be an instance of the FRAC-SIRCC(SP) problem. Then the following holds.*

1. *x is feasible if and only if $\alpha_k(G,x) \geq 1$.*

2. *If G is a series composition of H_1 and H_2 then*

$$\alpha_i(G,x) = \min\{\alpha_i(H_1,x), \alpha_i(H_2,x)\}$$

holds for every $0 \leq i \leq k$.

3. If G is a parallel composition of H_1 and H_2 then

$$\alpha_i(G,x) = \min_{0\le j\le i}\{\alpha_j(H_1,x)+\alpha_{i-j}(H_2,x)\}$$

holds.

Proof. 1. This is exactly the definition of a feasible solution.

2. Note first that $\alpha_i(G,x) \le \min\{\alpha_i(H_1,x),\alpha_i(H_2,x)\}$ holds: the flow between the terminal nodes of G must pass through the terminal vertex in common to H_1 and H_2. Hence any $j \in [2]$ and $F \subset E[H_j] \cap M$ with $|F| \le i$, which causes the flow in H_j to be at most $\alpha_i(H_j,x)$ will also reduce the flow to this value in G. To obtain $\alpha_i(G,x) \ge \min\{\alpha_i(H_1,x),\alpha_i(H_2,x)\}$ consider any $F \subset M$ with $|F| \le i$. Clearly removing this set of edges from H_j does not decrease the flow in H_j to a value, which is lower than $\alpha_i(H_j,x)$. This means that the maximum flow in G between the terminal nodes is indeed at least $\min\{\alpha_i(H_1,x),\alpha_i(H_2,x)\}$.

3. $\alpha_i(G,x) \le \alpha_j(H_1,x)+\alpha_{i-j}(H_2,x)$ holds for every $0 \le j \le i$: to obtain a failure set F, which decreases the flow between the terminal nodes of G to this value, one simply needs to take the union F of the two sets $F_1 \subset E[H_1] \cap M$ and $F_2 \subset E[H_2] \cap M$ with $|F_1| \le j$, $|F_2| \le i-j$, which satisfy $MF_{F_1}(H_1,x) = \alpha_j(H_1,x)$ and $MF_{F_2}(H_2,x) = \alpha_{i-j}(H_2,x)$, respectively. For the opposite direction Let $F \subset M$ be a set with $|F| \le i$ and $MF_F(G,x) = \alpha_i(G,x)$. F can be written as a disjoint union $F = F_1 \cup F_2$ with $|F_1| = j$, $|F_2| = i-j$, $F_1 \subset E[H_1]$ and $F_2 \subset E[H_2]$ for some $0 \le j \le i$. Observe that $\alpha_i(G,x) = MF_{F_1}(H_1,x) + MF_{F_2}(H_2,x) \ge \alpha_j(H_1,x)+\alpha_{i-j}(H_2,x)$. This finishes the proof. □

It remains to note that if G is a single edge then the following holds.

- If $e \in M$ and $i > 0$ then $\alpha_i(G,x) = 0$.
- If $e \notin M$ and $i > 0$ then $\alpha_i(G,x) = x(e)$.
- If $i = 0$ then $\alpha_i(G,x) = x(e)$.

We conclude that the following LP models the FRAC-SIRCC(SP) problem in SRP graphs. The LP uses a series-parallel decomposition $\mathcal{D} = (H_1,\cdots,H_r)$ of G. We use the notation $H = H_1||H_2$ if H is a part in this decomposition and it is a parallel composition of H_1 and H_2. If H is a series composition we write $H = H_1 \leftrightarrow H_2$. The variables of the LP are $x \in [0,1]^E$, which

corresponds to the solution of the fractional robust path problem, as well as variables $\alpha_i(H)$ for every subgraph $H \in \mathcal{D}$ and every $0 \leq i \leq k$, which satisfy

$$\alpha'_i(H) = \alpha_i(H, x')$$

for any feasible solution (x', α') of the LP. For every subgraph $H \in D$, which consists of a single edge e we write $\alpha_i(e)$ instead of $\alpha_i(H)$.

$$
\begin{aligned}
(LP^{SRP}_{SIRCC(SP)}) \min \quad & w(x) \\
\text{s. t.} \quad & \alpha_i(H) \leq \alpha_j(H_1) + \alpha_{i-j}(H_2) && \forall H = H_1 || H_2, \\
& && \forall i \in [k]_0, \forall j \in [i]_0 \\
& \alpha_i(H) \leq \alpha_i(H_1) && \\
& \alpha_i(H) \leq \alpha_i(H_2) && \forall H = H_1 \leftrightarrow H_2, \forall i \in [k]_0 \\
& \alpha_k(G) \geq 1 && \\
& \alpha_i(e) = x(e) && \forall e \in E \setminus M, \forall i \in [k] \\
& \alpha_i(e) = 0 && \forall e \in M, \forall i \in [k] \\
& \alpha_0(e) = x(e) && \forall e \in E \\
& 0 \leq x(e) \leq 1 && \forall e \in E \\
& 0 \leq \alpha_i(H) \leq 1 && \forall H \in \mathcal{D}, \forall i \in [k]_0
\end{aligned}
$$

Note that the number of variables and constraints in this LP is $O(kn)$, where n is the number of vertices of the graph. This compact LP is only achievable due to the implicit representation of the exponential scenario set, given by Proposition 3.7.9. We summarize the latter discussion in the following corollary.

Corollary 3.7.10. *$LP^{SRP}_{SIRCC(SP)}$ models correctly the FRAC-SIRCC(SP) problem in SRP graphs.*

3.8 Conclusion

In this chapter we developed near-optimal approximation algorithms for ERCCs of several important combinatorial optimization problems. We proved hardness-of-approximation of essentially all combinatorial problems in this model. Concretely, we showed that SC is a special case of almost any ERCC. The latter result suggested covering approaches for corresponding algorithmic solutions of these problems. We presented three main techniques, each

with a distinct covering inclination. The first is the well-known technique of randomly rounding a CIP (covering IP). The latter technique relied on the formulation of nominal combinatorial problems as CIPs. The constraints often corresponded to cuts in the input graph. This allowed us to apply of a careful analysis, employing Lemma 3.3.3. The latter lemma bounds the number of cuts of a particular deviation from the size of a minimal cut in the graph. Although this technique is simple, it yields prohibitively large linear programs, whose size grows linearly with the number of scenarios k.

In Section 3.4 we obtained simpler combinatorial algorithms for ERCC(ST) and ERCC(S2S). The former algorithm is extremely simple and fast, but at the same time guarantees an inferior approximation guarantee to the LP-based algorithms. For the latter algorithm we develop a framework, which relies of the greedy algorithm for SC. We show how the S2S problem can be modeled in this framework as an exponential size SC problem, which can be solved efficiently. This technique was applied to solve the S2S problem by Kortsarz and Peleg [53]. We observe that, in general, problems which admit a formulation in this framework admit efficient algorithms for the corresponding ERCC as well. We demonstrate this fact by obtaining an optimal approximation algorithm for ERCC(S2S).

In Section 3.5 we show that the robust matroid optimization problem admits an efficient approximation algorithm. The main tools in this algorithm are an approximation algorithm for SFM and a simple binary search procedure for obtaining good bounds for the optimal solution value.

All aforementioned algorithms carry a $O(\log n)$ factor in the approximation guarantee (where n is the number of vertices, or the cardinality of the ground set in matroid problems). Except for the ERCC(S2S) problem, in which this factor appears in known lower bounds, it may be possible to remove the dependence of the approximation factors of the algorithms for the ERCCs on the parameter n. This is an interesting topic for future research.

Some restricted variants of ERCC(SP) were studied in Section 3.6. In particular, constant factor approximation algorithms were developed for some variants of BERCC(SP). These results were complemented with hardness-of-approximation results for every bound $k \geq 2$. Exact algorithms were presented for FERCC(SP) and CERCC(SP) restricted to directed acyclic graphs. CERCC(SP) on general graphs and at least 4 scenarios was shown to be APX-hard. Although all results along this vein were presented for the robust counterparts of the SP problem, we remark that most techniques can be made to work for other problems as well.

Section 3.7 addresses the SIRCC model. We again focus on the SP problem,

showing that unlike the seemingly similar IRCC(SP) problem, this problem is hard to approximate. On the positive side, we show that the case $k = 1$ is solvable in polynomial time, and that a $(k + 1)-$approximation algorithm can be obtained using network flow techniques. The latter result is an implication of a result on the integrality gap of SIRCC(SP). We also show that, unlike the BERCC(SP) problem, SIRCC(SP) is solvable in polynomial time on SRP graphs. We provide a fast algorithm and a compact LP formulation for SIRCC(SP) and its fractional variant, respectively.

Chapter 4

Uniform Faults with Recourse

4.1 Introduction

This chapter studies the ARCC model for robust combinatorial optimization. Recall that the ARCC model assumes that, apart from the input to the nominal problem, two extra parameters k and r are supplied in the input. These parameters control the power of the adversary and the recovery capacity of the system, respectively. This is the first model studied in this thesis, which admits a *completely uniform failure model*, namely one in which every subset of at most k resources is a valid failure scenario. In this sense, ARCC is also an extension of the IRCC model, which is achieved by fixing the recovery parameter r to zero in the ARCC model. We stress that the despite its implicit two-stage nature, the optimization is only performed over the first stage decision. The second stage only comes into play as a feasibility requirement, namely it is required that the chosen first stage solution should be amendable to a feasible solution of the underlying combinatorial problem with at most r new resources, after the failure of any subset of k resources. Another important feature is the requirement that the first stage solution is a feasible solution for the underlying problem. This requirement is motivated by the fact that in most applications, the nominal solution does not face adversity in every circumstance, but rather in some rare cases. In these cases, however, the solution needs to be able to sustain an adversarial attack.

We focus on two concrete problems, ARCC(SP) and ARCC(ST). In both cases we study the unweighted case. This assumption is motivated by a significant simplification of the structure of the feasible set, which lead to simpler algorithmic solutions. In some cases the approaches presented can be extended to the weighted case, but as a whole, this remains a direction for future research. We denote fixed parameter variants of a problem ARCC($\mathcal{P}$) by (k, r)-ARCC($\mathcal{P}$).

4.2 ARCC(SP)

This section studies the two-stage robust problem ARCC(SP). Let us recall the formal statement of the problem. The results of this section are a summary of the results in [2].

ARCC(SP):

Input: A graph $G = (V, E)$, two vertices $s, t \in V$ and $k, r \in \mathbb{Z}_+$.

Problem: Find $S \subset E$ of minimum cardinality which connects s and t, such that for every $F \subset E$ with $|F| \leq k$ there exists $R \subset E \setminus F$ with $|R| \leq r$ such that s and t are connected in $(S \setminus F) \cup R$.

Note that we are concerned solely with the unweighted and undirected variants of the problem.

ARCC(SP) is motivated by the following application. Consider a network in which a fault-tolerant connection needs to established between two designated nodes (modeled as a path between the nodes). We are allowed to rent links in the network at a fixed cost per unit of time to establish the connection. Furthermore, there is a setup time for links. A certain number r of links can be set up immediately by a limited available crew of workers. More than r arcs cannot be set up in a single unit of time. We are also aware that the links in the network are failure prone and in a worst case scenario the network can be subject to an adversarial attack which can destroy the k most vital links in the network. In the event of failure we can immediately set up and rent r links that would bypass the failed links using the crew of workers. We refer to this as the *recourse action*. Our goal is to set up and rent a minimum number of links that would guarantee a continuous connection between the two designated nodes. We call this kind of solution an *adaptable robust solution*. Note that in the above setting allowing no recourse action would require the solution to contain $k + 1$ 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).

We stress that the optimal solution to ARCC(SP) need not contain a shortest s-t path in the graph. Our name for the problem draws from the following analogy to the SP problem. The SP problem asks to find the smallest number of edges, which constitute an s-t path in the graph, and ARCC(SP) asks to find a smallest number of edges that constitute a *recoverable robust path* from s to t in the graph.

Note that in the ARCC(SP) problem, the scenarios are given implicitly, where a scenario corresponds to a set $F \subset E$ with $|F| \leq k$.

We will consider more restricted variants of the problem in which either one or both of the parameters k and r are fixed. In particular we will focus on the case $k = 1$ and study the problem for fixed and variable r. We denote by $OPT_r(G)$ the cardinality of an optimal solution set to the above problem.

Note that the cases $k = 0$ and $r = 0$ are solvable in polynomial time. If $k = 0$ the problem is the classical shortest path problem from s to t. When $r = 0$ any solution must contains $k + 1$ 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. [62]).

Consider next the problem of deciding feasibility of a solution S to an ARCC(SP) instance. In the case of variable k and r we prove in Section 4.2.5 that the problem is NP-hard. In the case of fixed k 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 k is variable is seemingly open.

4.2.1 The feasible set for $k = 1$

In all forthcoming discussions paths are node-disjoint, i.e., they do not contain cycles.

We start by characterizing the feasible set in the case $k = 1$. 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 $s, t \in V$.

Definition 4.2.1. 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 (k, r)*-recoverable* if for every $F \subset E$ with $|F| \leq k$ 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 $k = 1$ is the family of all $(1, r)$-recoverable subsets of E, and hence in this case ARCC(SP) 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 4.2.2. 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 (k, r)-recoverable, then it is minimal if and only if $S \setminus \{e\}$ is not (k, r)-recoverable, for every $e \in S$. This is due to the monotonicity property of (k, r)-recoverable sets which states that any super-set of a (k, r)-recoverable set is a (k, 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 4.2.3. *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 4.2.4. *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 4.2.3 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 4.2.3 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 4.2.3 also $(1, r)$-recoverable, contradicting the minimality of S. □

Corollary 4.2.5. *$S \subset E$ is minimum $(1, r)$-recoverable if and only if S is an r-cyclic and connecting set of minimum cardinality.*

4.2.2 A polynomial algorithm for $k = r = 1$

In this section we provide and exact polynomial algorithm for the case $k = r = 1$. Corollary 4.2.5 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 the survey of Khuller [51] for a comprehensive treatment of the topic. Despite some similarities of $(1, 1)$-ARCC(SP) to the Minimum 2-Edge Connected Spanning Subgraph Problem

(e.g. every 2-edge connected spanning subgraph of G is a feasible solution to $(1,1)$-ARCC(SP)) the problems are inherently different. In particular, $(1,1)$-ARCC(SP) 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 $(1,1)$-ARCC(SP) to the latter problem are explained and exploited in remaining of this section.

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

Lemma 4.2.6. *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 4.2.4 that minimal connecting r-cyclic sets are acyclic. Combining this insight with Lemma 4.2.6 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 4.2.7. *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 4.2.6. 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 4.2.8. *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 $(1, 1)$**-ARCC(SP):**

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 4.2.9. *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 $j_1 < j_2$.

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. Figure 4.1 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 4.2.9 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.

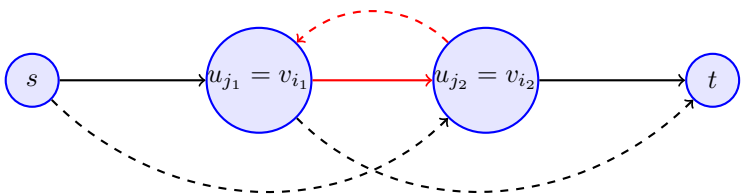

Figure 4.1: The thick subpaths and the dashed subpaths correspond to p and q respectively. The red cycle can be removed without breaking 2-edge connectivity.

Lemma 4.2.10. *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 $s = u_1, u_2, ..., u_l = t$ be the vertices which appear on both paths sorted according to their order on p from s to t. Then*

1. *The vertices $u_1, u_2, ..., u_l$ 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 4.2.9. 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 4.2.10 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 $O(n^2)$ 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 $O(n^2F(G))$.

Theorem 4.2.11. *There is a polynomial algorithm for* $(1,1)$*-ARCC(SP).*

4.2.3 A 2-approximation for $k = 1$ and variable r

In this section we consider the ARCC(SP) problem with $k = 1$ 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 4.2.12. 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 4.2.12. 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 4.2.13. *For every pair of vertices $u, v \in V$, $A_{u,v}$ 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 $(1,r)$-ARCC(SP) problem.

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

Theorem 4.2.14. *$d_r(s,t) \leq 2OPT_r(G)$.*

To prove Theorem 4.2.14, we give some auxiliary results which are insightful in their own right and allow to deduce Theorem 4.2.14 rather easily. We denote by $\mathcal{C}$ the set of all cycles in G.

Definition 4.2.15. 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 4.2.16. *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 4.2.17. *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 4.2.16 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 4.2.18. *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 $a_i = \min\{h \in [l] \mid v_h$ lies on $C_i\}$ and $b_i = \max\{h \in [l] \mid v_h$ 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 $D_1^i, \ldots, D_{m_i}^i$, all cycles obtained in this way by completing the different maximal paths of $C_i \setminus P^*$ to cycles. We call the cycle family $\{D_1^i, \ldots, D_{m_i}^1\}$ the *corresponding flattened cycle family* to C_i. We clearly have $\sum_{k=1}^{m_i} |D_k^i \setminus S^*| = |C_i \setminus S^*|$ for $i \in [m]$. This implies that $|D_k^i \setminus S^*| \leq r$ 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 $\{D_1^1, \ldots, D_{m_1}^1\}$ 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 $\mathcal{F}' = \{C_3, \ldots, C_m\} \cup \{D_1^1, \ldots, D_{m_1}^1\} \cup \{D_1^2, \ldots, D_{m_2}^2\}$. 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 $D \in \{D_1^1, \ldots, D_{m_1}^1\}$ 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 4.2.16 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 $\{C_3, \ldots, C_m\} \cup \{D_1^1, \ldots, D_{m_1}^1\}$ 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 $\{D_1^1, \ldots, D_{m_1}^1\}$ 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_j^1 such that $D_j^1 \setminus P^*$ has q as one of its endpoints. Hence, $D_j^1 \cap P^*$ 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_k^1 that has v_{b_1} as one of its endpoints. Again, $D_k^1 \cap P^*$ 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 Figure 4.2). Notice that $c_k < a_2$, since otherwise $D_k^1 \cap P^* \subset \cup_{i=1}^{m_2} D_i^2$, and hence D_k^1 can be removed from $\mathcal{F}'$. Furthermore $c_j < c_k$, since otherwise $D_j^1 \subset D_k^1$ and hence D_j^1 can be removed from $\mathcal{F}'$. However, under the current assumptions (see Figure 4.2 to recall the current order of the mentioned vertices on P^*) D_k^1 can be removed from $\mathcal{F}'$ since every edge $e \in D_k^1 \cap P^*$ is either contained in $\cup_{i=1}^{m_2} D_i^2$, which covers all edges on P^* between v_{a_2} and v_{b_2}, or $e \in D_j^1$. □

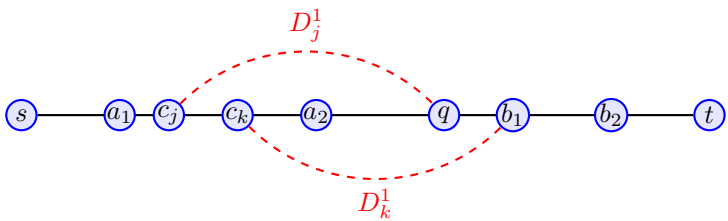

Figure 4.2: The black line represents the path P^*. The nodes between s and t are labeled with respect to their index.

Lemma 4.2.19. *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 $\max\{h \in [l] \mid v_h$ lies on $C_i\} \leq \max\{h \in [l] \mid v_h$ 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^*.*

Proof. 1.) For $k \in [m]$ we define $a_k = \min\{h \in [l] \mid v_h$ lies on $C_k\}$, $b_k = \max\{h \in [l] \mid v_h$ lies on $C_k\}$. By Lemma 4.2.18, $C_k \cap P^*$ 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 4.2.16 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 Figure 4.3). 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 4.2.16, $\mathcal{F}' \setminus \{C_{i+1}\}$ 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 $a_k = \min\{j \in [l] \mid C_k$ passes trough $v_j\}$ and $b_k = \max\{j \in [l] \mid C_k$ passes trough $v_j\}$. Since

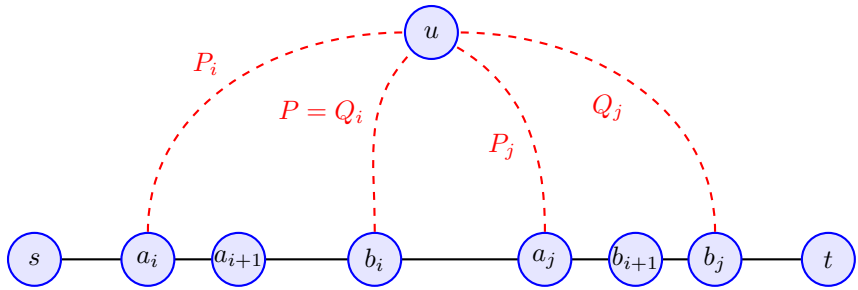

Figure 4.3: The black line represents the path P^*. The nodes between s and t are labeled with respect to their index.

$C_i \cap C_{i+1} \cap P^* \neq \emptyset$, 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 4.2.16 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 4.2.14. Let S^* be a minimum $(1, r)$-recoverable set, let P^* be the unique s-t path in S^* and let $\mathcal{F} = \{C_1, ..., C_m\}$ 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 4.2.19, 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 $i \in \{0, ..., m-1\}$, $l_r(w_i, w_{i+1}) \leq |C_i \cap S|$ 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 4.2.19 *1.)* every edge of G can be shared by at most 2 cycles in $\mathcal{F}$ we obtain

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

□

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 $O(|V|^2 F(G))$ ($F(G)$ is the running time of the minimum-cost flow algorithm on G).

4.2.4 The existence of connected optimal optimal solution for $k = 1$

In this section we prove the following structural result on optimal $(1, r)$-recoverable sets, namely that the problem can be restricted to connected sets. This additional condition is not needed for the suggested algorithm, but this property might be important in some applications.

Lemma 4.2.20. *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 $\mathcal{F} = \{C_1, ..., C_m\}$ be a minimum recoverable cycle family for S^* with minimum $\sum_{C \in \mathcal{F}} |P^* \cap C|$ (as in Lemma 4.2.19 *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 4.2.19 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 4.2.19 guarantees that these cycles are adjacent in the ordering defined in Lemma 4.2.19 (hence, say $j = i+1$). Furthermore, part *2.* of Lemma 4.2.19 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 4.2.18 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 $|C_{i+1} \setminus S| = |C_{i+1} \setminus S^*| \leq r$. $\square$

4.2.5 Complexity of ARCC(SP)

We show a reduction from the Most Vital Arcs problem (MVAP), which was introduced by Corley and Sha [28]. 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 $m \in \mathbb{N}$. 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 $|U| \leq m$ that maximizes $d_U(s, t)$. In [14] Bar-Noy, Khuller and Schieber showed that the following natural decision version of MVAP is NP-hard: decide for some given integers $k, D \in \mathbb{N}$ whether there is a set $U \subseteq E$, $|U| \leq k$ such that $d_U(s, t) > D$.

Theorem 4.2.21. *The ARCC(SP) problem is NP-hard if k 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 ARCC(SP) 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 $m, D \in \mathbb{N}$. Let $G' = (V, E')$ be the graph defined by $E' = E \cup \{e\}$, where e is an edge between s and t. Consider the ARCC(SP) problem on G' between s and t with parameters $k = m + 1$ and $r = D$. We will show that the thus defined ARCC(SP) problem 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 ARCC(SP) problem because for any failure set $F \subseteq E'$ with $|F| \leq k$ 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 $|U| \leq m$ such that there is no path in $(V, E \setminus U)$ of length $\leq D = r$. Let $F = U \cup \{e\}$. Consequently, an (k, r)-recoverable set $S \subseteq E$ for the ARCC(SP) problem 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 ARCC(SP) problem has no solution of cardinality one.

Note that the optimal solution value to the transformed ARCC(SP) problem 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 ARCC(SP) allows to decide MVAP. Hence the inapproximability result.

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

4.3 ARCC(ST)

We switch our attention to the ARCC(ST) problem, whose formal statement is as follows.

ARCC(ST):

Input: A graph $G = (V, E)$ and $k, r \in \mathbb{Z}_+$.

Problem: Find $S \subset E$ of minimum cardinality which spans G, such that for every $F \subset E$ with $|F| \leq k$ there exists $R \subset E \setminus F$ with $|R| \leq r$ such that $(S \setminus F) \cup R$ spans G.

We again focus on the unweighted case. The feasibility problem corresponding to ARCC(ST) is solvable in polynomial time via a simple computation of the minimum cut in G. Indeed, it is straightforward to see that the instance is feasible if and only if G is $(k+1)$-edge connected. We will henceforth assume that every instance at hand is feasible.

ARCC(ST) is a generalization of the *m-edge connected spanning subgraph problem (m-ECSS)*, which given a graph $G = (V, E)$, asks to find the cheapest possible set of edges $S \subset E$, which spans G, and which is m-edge connected (in other words, such that S contains at least m edge-disjoint paths between any pair of vertices $u, v \in V$). This can be seen by considering the instance $I = (G, m-1, 0)$ to ARCC(ST). As a matter of fact, ARCC(ST) presents a natural extension of the m-ECSS problem in the following sense. The most common application of m-ECSS lies in the domain of survivable network design, where the connectivity of a network needs to be guaranteed, even in the presence of a certain predetermined number $m-1$ of edge failures. ARCC(ST) models the additional capacity, available in many applications, to acquire additional network resources, albeit a limited amount r thereof, to amend the aforementioned disruption.

To highlight the observations in the latter discussion, we restate both problems in terms of cuts in the graph. While m-ERCC asks to find a subgraph, which contains at least m edges in every cut (into two parts), (k, r)-ARCC(ST) asks to find a subgraph, such that every $(r+2)$-cut contains at least $k+1$ edges. The latter formulation of (k, r)-ARCC(ST) gives rise to the

following simple IP formulation (for $k \geq r$). We use $\mathcal{C}_q(G)$ to denote the set of q-cuts in a graph G.

$$
\begin{array}{llll}
(IP_{ARCC(ST)}) & \min & \sum_{e \in E} x_e & \\
 & \text{subject to} & \sum_{e \in E(C)} x_e \geq k+1 & \forall C \in \mathcal{C}_{r+2}(G) \\
 & & x_e \in \{0,1\} & \forall e \in E.
\end{array}
$$

We will only consider the case $k > r$. Let us quickly discuss the other case $k \leq r$. In this case any spanning tree of G is an optimal solution to the problem. Indeed, every failure scenario F with at most k edges breaks the spanning tree into $k+1$ components, which can always be put back together by some $r \geq k$ other edges.

We define next the notion of a (k, r)-recoverable graph in the context of ARCC(ST) to be any graph, which satisfies the constraints of $IP_{ARCC(ST)}$. To this end observe that while deciding feasibility of an instance of ARCC(ST) is a polynomial problem, the problem of deciding if a graph is (k, r)-recoverable is polynomial only when r is fixed. In light of our previous discussion, this decision problem is equivalent to the decision problem: Does the minimum $(r+2)$-cut in the graph contain $k+1$ or more edges? We defer the details to Section 4.3.4.

While m-ECSS was extensively studied, and its complexity status settled (see e.g. the paper of Gabow, Goemans, Tardos and Williamson [37] and references therein), ARCC(ST) received no attention at all. Being a generalization of m-ECSS, ARCC(ST) does not admit better approximation algorithms. At the same time, it is possible to show that a constant factor approximation algorithm can be obtained for ARCC(ST), by taking as a solution an optimal solution to the corresponding m-ECSS instance for an appropriately chosen $m = m(k, r)$. Our goal is to derive independent approximation algorithm, with a potentially better approximation guarantee. The results of this sections are two algorithms for ARCC(ST). The first algorithm is a simple combinatorial algorithm, which achieves an approximation guarantee of 2. The second result is an LP-based algorithm, which achieves a better bound for large values of $\frac{k}{r}$ and fixed r. We also show a better analysis of the former algorithm for certain parameters k, r.

4.3.1 A combinatorial 2-approximation algorithm

Let $G = (V, E)$ and $k, r \in \mathbb{Z}_+$ be fixed as input to (k, r)-ARCC(ST). Let $p, q \in \mathbb{Z}_+$ be such that $k + 1 = p(r + 1) + q$ and $q < r + 1$. Let $|V| = n$. As a first step we bound the number of edges in any optimal solution to (k, r)-ARCC(ST) as a function of n, k and r.

Lemma 4.3.1. *Let $U^* \subset E$ be an optimal solution to (k, r)-ARCC(ST). Then*

$$|U^*| \geq \frac{1}{2}\left(k + 1 + (n - r - 1)\left\lceil \frac{k+1}{r+1} \right\rceil\right).$$

Proof. Consider an ordering of V according to increasing degree in $G' = (V, U^*)$, $v_1, v_2 \cdots, v_n$, and let the $d_1^*, d_2^*, \cdots, d_n^*$ be the corresponding degrees. Since U^* is (k, r)-recoverable it is impossible to disconnect the vertices $v_1, v_2, \cdots, v_{r+1}$ from each other by removing fewer than $k+1$ edges (In other words, the $(r+2)$-way cut $(\{v_1\}, \{v_2\}, \cdots, \{v_{r+1}\}, \{v_{r+2}, \cdots, v_n\})$ in U^* contains at least $k + 1$ edges). Clearly, one does not need more than $\sum_{i=1}^{r+1} d_i^*$ edges to separate those vertices. In particular since d_{r+1}^* is the largest of the terms in the sum, it must hold that $d_{r+1}^* \geq \lceil \frac{k+1}{r+1} \rceil$. Furthermore, for every $j > r + 1$ it holds that $d_j^* \geq d_{r+1}^* \geq \left\lceil \frac{k+1}{r+1} \right\rceil$. Hence,

$$2|U^*| = \sum_{i=1}^{n} d_i^* = \sum_{i=1}^{r+1} d_i^* + \sum_{i=r+2}^{n} d_i^* \geq k + 1 + (n - r - 1)\left\lceil \frac{k+1}{r+1} \right\rceil.$$

□

We use the bound in Lemma 4.3.1 to prove that the simple algorithm given as Algorithm 5 is a 2-approximation for ARCC(ST). The algorithm progresses by iteratively adding sufficiently many spanning forests of the remaining graph.

Theorem 4.3.2. *Algorithm 5 is a 2-approximation for ARCC(ST).*

Proof. Let $U \subseteq E$ be a set returned by Algorithm 5. We first show that U is an (k, r)-recoverable set before showing that the algorithm is a 2-approximation. To show that U is (k, r)-recoverable, we show that for any $(r+2)$-cut $(V_1, \ldots, V_{r+2})$ in G, we have $|E(V_1, \ldots, V_{r+2}) \cap U| \geq k+1$. Let $W = E \setminus \cup_{i=1}^{p} T_i$ and consider the following set $\overline{W}$, which represents the pairs of sets in the partition that are connected by edges of W, i.e.,

$$\overline{W} = \{\{V_i, V_j\} \mid \exists v_i \in V_i, v_j \in V_j \text{ with } \{v_i, v_j\} \in W\}.$$

Algorithm 5 *Input: Graph $G = (V, E)$, integers $k, r \in \mathbb{Z}_+$. Output: (k, r)-recoverable set U.*

```
Let p = ⌊(k+1)/(r+1)⌋ and q = k + 1 − p(r + 1).
U ← ∅.
for i = 1, ..., p do
    Choose a spanning forest T_i ⊆ E \ U of H = (V, E \ U).
    U ← U ∪ T_i.
end for
if q > 0 then
    Choose a spanning forest T ⊆ E \ U of H = (V, E \ U).
    if |T| ≥ r + 1 then
        Set T' ⊆ T to be any subset of G of |T| − (r + 1) + q edges.
        U ← U ∪ T'
    end if
end if
Return U.
```

The set $\overline{W}$ can be interpreted as edges on a graph with vertices $\overline{V} = \{V_1, \ldots, V_{r+2}\}$. We distinguish two cases: a) $(\overline{V}, \overline{W})$ has more than one connected component or b) $(\overline{V}, \overline{W})$ is a connected graph.

a) If $(\overline{V}, \overline{W})$ contains more than one connected component, let $I \subseteq [r+2]$ be such that $\cup_{i \in I}\{V_i\}$ is a connected component of $(\overline{V}, \overline{W})$. Let $A_1 = \cup_{i \in I} V_i$, $A_2 = V \setminus A_1$ be the corresponding partition of V. Since W contains no edges between A_1 and A_2, all edges between these sets are already contained in the previously chosen spanning forests $T_1, \ldots, T_p$, i.e., $E(A_1, A_2) \subseteq \cup_{i=1}^{p} T_i$. Furthermore, $E(A_1, A_2) \subseteq E(V_1, \ldots, V_{r+2})$ and $\cup_{i=1}^{p} T_i \subseteq U$, which implies

$$|U \cap E(V_1, \ldots, V_{r+2})| \geq |U \cap E(A_1, A_2)| = |E(A_1, A_2)| \geq k + 1,$$

where the last inequality follows from the assumption that every cut in G contains at least $k + 1$ edges.

b) If $(\overline{V}, \overline{W})$ is a connected graph, then every spanning forest $T_1, \ldots, T_p$ as well as T if $q > 0$ contain at least $r + 1$ edges of $E(V_1, \ldots, V_{r+2})$, since otherwise they would not be spanning. If $q = 0$, then

$$|U \cap E(V_1, \ldots, V_{r+2})| = \sum_{i=1}^{p} |T_i \cap E(V_1, \ldots, V_{r+2})| \geq p(r + 1) = k + 1.$$

Otherwise if $q > 0$, then since $|T| \geq n - 1 \geq r + 1$, the set U also contains $T' \subseteq T$. Since T' is obtained from T by removing $r + 1 - q$ edges, we have

$|T' \cap E(V_1, \ldots, V_{r+2})| \geq (r+1) - (r+1-q) = q$, and hence

$$\begin{aligned}|U \cap E(V_1, \ldots, V_{r+2})| &= |T' \cap E(V_1, \ldots, V_{r+2})| + \sum_{i=1}^{p} |T_i \cap E(V_1, \ldots, V_{r+2})| \\ &\geq q + p(r+1) = k+1.\end{aligned}$$

It remains to prove that U is a 2-approximation for the ARCC(ST) problem. We compare $|U|$ to the lower-bound for the size of an optimal edge set U^* given by Lemma 4.3.1. Consider first the case $q > 0$. The size of U can be upper-bounded by

$$|U| \leq p(n-1) + n - 1 - (r+1) + q, \tag{4.1}$$

since $|T_i| \leq n-1$ for $i \in [p]$, and $|T'| \leq n - 1 - (r+1) + q$. Combining this result with Lemma 4.3.1 we get

$$\begin{aligned}2|U^*| - |U| &\geq k + 1 + (n-r-1)(p+1) - (p(n-1) + n - 1 - (r+1) + q) \\ &= k + 2 - pr + q > k + 2 - p(r+1) - q = 1,\end{aligned}$$

where for the last equality we used $k + 1 = p(r+1) + q$. If $q = 0$, we have $|U| = |\cup_{i=1}^{p} T_i| \leq p(n-1)$, and additionally, the bound of Lemma 4.3.1 simplifies to $2|U^*| \geq k + 1 + (n-r-1)p$ since $\frac{k+1}{r+1} = p$, implying

$$2|U^*| - |U| \geq k + 1 + (n-r-1)p - p(n-1) = k + 1 - pr > 0,$$

since $k + 1 - p(r+1) = 0$. □

Let us show a tight example for the analysis of Algorithm 5. In fact, we will show that the lower bound of Lemma 4.3.1 cannot be improved simultaneously for all values of k and r.

Example 4.3.3. *Let $n \in \mathbb{Z}_+$ be some integer. The d-dimensional torus graph $T_n^d = (V_n^d, E_n^d)$ of size n is defined as follows. The vertex set is indexed by the set $\mathcal{I}_n^d = \{I = (i_1, \cdots, i_d) : i_j \in [n]\ \forall j \in [d]\}$, namely V contains the vertex v_I for every $I \in \mathcal{I}_n^d$. The edge set contains an edge $e = (v_I, v_J)$ if $I = (i_1, \cdots, i_d)$ and $J = (j_1, \cdots, j_d)$ differ in exactly one component $t \in [d]$, and such that*

$$|i_t - j_t| = 1 \mod n.$$

Note that T_n^d is a grid graph with periodic boundaries, and the number of vertices and edges it contains is n^d and dn^d, respectively. See Figure 4.4 for an illustration.

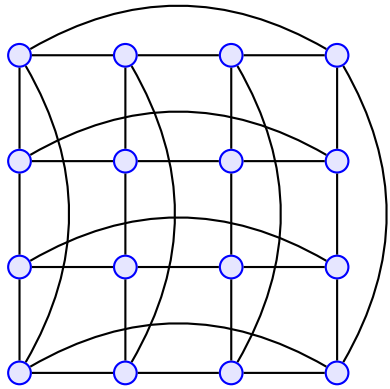

Figure 4.4: The Torus graph T_4^2.

We claim that T_n^2 is $(6,1)$-recoverable (for $n > 2$). Indeed, note that the minimum 3-cut $C = (V_1, V_2, V_3)$ in T_n^2 is attained by setting $V_1 = \{u\}$ and $V_2 = \{v\}$ for some adjacent u and v. This cut has cardinality 7, since it contains exactly all edges incident to u and v. At the same time we have $\lceil \frac{k+1}{r+1} \rceil = 4$, hence the lower bound in Lemma 4.3.1 is equal to $\frac{1}{2}(7+4(n^2-2)) = 2n^2 - \frac{1}{2}$, which matches $|E_n^2|$ exactly (considering the fact that lower bound can be rounded up).

The same graph is also $(9,2)$-recoverable. The minimal 4-cut corresponds to the partition $V_1 = \{v_1\}$, $V_2 = \{v_2\}$, $V_3 = \{v_3\}$ and $V_4 = V_n^2 \setminus \{v_1, v_2, v_3\}$, where v_1, v_2 and v_3 are any three vertices linked by two edges. The cardinality of this cut is 10. The lower bound of Lemma 4.3.1 gives $\frac{1}{2}(10 + 4(n^2 - 3)) = 2n^2 - 1$ in this case, which is smaller than the size of the graph by one.

Example 4.3.3 shows that for some values of k and r, there are (k,r)-recoverable graphs with n vertices, which contain $\frac{1}{2}\lceil \frac{k+1}{r+1} \rceil n$ edges. It is natural to ask whether this is true, perhaps up to some additive factors of k and r, for every combination of k and r. We give a negative answer to the former question in Section 4.3.3.

4.3.2 ARCC(ST) and m-edge connected graphs

As we mentioned before, the ARCC(ST) problem is strongly related to the m-ECSS problem. In this section we explore this connection further. Our

main goal is to identify a critical parameter, which controls the complexity of an instance $I = (G, k, r)$ of ARCC(ST). We will see that while m-ECSS admits better approximations when m is large, the same holds, to some extent for ARCC(ST), when the parameter $\frac{k}{r}$ is large. This fact is also reflected in the improved algorithm for ARCC(ST) we present in the next section.

A first simple observation is that the graph returned by Algorithm 5 is a $\lfloor \frac{k+1}{r+1} \rfloor$-edge connected graph. This raises the following interesting question: is every (k, r)-recoverable set of edges close to a t-edge connected graph, for some $t \approx \frac{k}{r}$? In the following lemma we answer this question in the affirmative.

Lemma 4.3.4. *Let X be a (k, r)-recoverable set of edges for G. Then there exists a set $Y \subset E$ with $|Y| \leq k$ such that $S \cup Y$ is $(\lfloor \frac{k}{r+1} \rfloor + 1)$-edge connected.*

Proof. Let $t = \lfloor \frac{k}{r+1} \rfloor + 1$. If X is a t-edge connected subgraph we are done with $Y = \emptyset$. Otherwise, let C_1 be a cut in X with $\sigma_1 = |C_1| < t$ edges. Let $G_1 = (S_1, X_1)$ and $G_2 = (V \setminus S_1, X_2)$ be the two shores of C_1. If both G_1 and G_2 are t-edge connected choose Y to be some $t - \sigma_1$ edges in $E \setminus X$ (such Y is guaranteed to exist since G is $(k + 1)$-edge connected). The fact that $X \cup Y$ is t-edge connected follows from the following claim.

Claim1: Let $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ be t-edge connected graphs, and let $u_1, \cdots, u_t$ and $v_1, \cdots, v_t$ be be two sets of vertices of H_1 and H_2 respectively. Then $H' = (V', E')$ is t-edge connected, where $V = V_1 \cup V_2$ and $E' = E_1 \cup E_2 \cup \{u_1 v_1, \cdots, u_k v_k\}$.

proof of Claim1: Consider any nontrivial cut $W \subset V'$. We need to show that the number of edges crossing this cut is at least t. If $W \cap V_i \notin \{\emptyset, V_i\}$ for some $i \in [2]$ the claim follows from t-edge connectedness of H_i. In the only other case $W \in \{V_1, V_2\}$ and hence $E(W, V' \setminus W) = \{u_1 v_1, \cdots, u_k v_k\}$, which proves the claim.

Assume next that either G_1 or G_2 are not t-edge connected. Assume without loss of generality that G_2 is not t-edge connected. We can choose another cut C_2 in G_2 with size $\sigma_2 = |C_2| < t$, the removal of which leaves three connected components. We continue in this way until either all remaining connected components are t-edge connected, or we found $r + 1$ cuts (equivalently, we performed $r + 1$ iterations of the latter procedure).

If the procedure stopped after $r + 1$ iterations we found a $(r + 2)$-cut in X

with size

$$\sum_{i=1}^{r+1} \sigma_i \leq \sum_{i=1}^{r+1} t - 1 = (r+1)(\lfloor \frac{k}{r+1} \rfloor) \leq (r+1)\frac{k}{r+1} = k,$$

contradicting the choice of X as a (k,r)-recoverable set.

Assume next that the procedure stopped with $d \leq r$ iterations, once all remaining connected components $X_1, ..., X_{d+1}$ were t-edge-connected. By Claim1, we can find d sets $Y_1, ..., Y_d$ of edges with sizes $|Y_i| = t - \sigma_i$, $i \in [d]$, such that $X \cup \bigcup_{i=1}^{d} Y_i$ is t-edge connected. We set $Y = \bigcup_{i=1}^{d} Y_i$. It holds that

$$|Y| = |\bigcup_{i=1}^{d} Y_i| \leq \sum_{i=1}^{d} t - \sigma_i \leq r(t-1) = r\lfloor \frac{k}{r+1} \rfloor < k,$$

which proves the lemma. □

It is natural to ask whether the converse is also true: is every t-edge connected subgraph of G almost (k,r)-recoverable, with $t \approx \frac{k}{r}$? As we show in the following example, this need not hold, in general.

Example 4.3.5. *Let $d \in \mathbb{Z}_+$ be an even integer and let $r \in \mathbb{Z}_+$ be any integer. Consider a graph $H_{n,d}$, which is obtained from the cycle graph on n vertices C_n, by duplicating each edge d times. Observe that the optimal d-edge connected subgraph of $H_{n,d}$ contains exactly $\frac{d}{2}$ edges from each set of d parallel edges connecting two vertices. At the same time, observe that the only $(rd-1, r-1)$-recoverable subgraph H of $H_{n,d}$ is $H_{n,d}$ itself. Indeed assume some edge e connecting two vertices, u and v is not included in H. Then, taking all $d-1$ edges parallel to e, along with $d(r-1)$ edges, comprising $r-1$ sets of parallel edges connecting $r-1$ respective pairs of vertices, results in a set of $rd-1$ edges, which corresponds to a $(r+1)$-cut in H. Note that the cost of this solution is exactly double that of the optimal d-edge connected subgraph, despite the fact that $\lfloor \frac{rd-1}{r} \rfloor + 1 = d$.*

As the previous example displayed, the reason why some t-edge connected subgraphs fail to be close to (k,r)-recoverable sets (with $t \approx \frac{k}{r}$) is that in many cases, removing a very small number of edges from the t-edge connected subgraph results in a graph with a large number of t'-cuts for some $t' \approx \frac{t}{2}$. Let us briefly sketch the tightness of this example in the special case that multiple copies of a given edge can be taken. Consider a t-edge connected subgraph H with $t = \lfloor \frac{k}{r+1} \rfloor + 1$, and consider any $(r+2)$-cut $(V_1, \cdots, V_{r+2})$

in H. Since H is t-edge connected, every component V_i is incident to at least t edges connecting it to other components. Let the number of such edges be d_i. We obtain

$$2|E(V_1, \cdots, V_{r+2})| = \sum_{i \in [r+2]} d_i \geq (r+2)t > k.$$

It follows that by doubling all edges in H we obtain an (k, r)-recoverable set. An identical argument shows that every $\lceil \frac{2k}{r+1} \rceil$-edge connected subgraph of G is a (k, r)-recoverable set. Mader [59] proved that any inclusion-wise minimal t-connected graph contains at most nt edges. Since $\lceil \frac{2k}{r+1} \rceil n \leq 4\frac{k+1}{2(r+1)}n \leq 4LB$, where LB is the lower bound obtained in Lemma 4.3.1, any algorithm, which returns an inclusion-wise minimal $\lceil \frac{2k}{r+1} \rceil$-edge connected subgraph of G is a 4-approximation (In some special cases where $\lceil \frac{2k}{r+1} \rceil < k+1$, G might admit a cut with fewer than $\lceil \frac{2k}{r+1} \rceil$ edges, in which case this algorithm does not directly work, but it can easily be adapted for these cases as well).

A special case in which no such gap exists is the case $(k, k-1)$-ARCC(ST) for any $k \in \mathbb{Z}_+$. Lemma 4.3.4 tells us that any $(k, k-1)$-recoverable set can be augmented to a 2-edge connected subgraph by adding at most two edges. In this special case the converse also holds: every 2-edge connected subgraph H is $(k, k-1)$-recoverable. To see this consider any $(k+1)$-cut in H. If this cut contains k or fewer edges, one of those edges must be a bridge in H, contradicting the assumption that H is 2-edge connected. The seemingly related problem $(k, k-2)$-ARCC(ST) fails to satisfy this property. In fact, it is easy to see that the gap grows back to 2 for every $k \in \mathbb{Z}_+$, via the argument in Example 4.3.5.

4.3.3 Minimal (k, r)-recoverable graphs

In this section we partially answer the question raised in Section 4.3.1. In particular, we show that for many parameters k and r, such that $\lceil \frac{k+1}{r+1} \rceil = 2$, the number of edges in any sufficiently large (k, r)-recoverable graph is at least $\frac{6}{5}n - O(r)$, which is significantly larger than the lower bound, $LB = n$, obtained from Lemma 4.3.1. Let us define a quantity that measures the tightness of the lower bound with respect to a given set of parameters k, r.

Definition 4.3.6. For two integers $k, r \in \mathbb{Z}_+$ with $k > r$ let $M_{k,r}(n)$ denote the minimal number of edges in any (k, r)-recoverable graph with n vertices. The (k, r)-*density* is defined as

$$\kappa(k, r) = \liminf_{n \to \infty} \frac{M_{k,r}(n)}{L(k, r, n)}, \tag{4.2}$$

where $L(k,r,n) = \frac{1}{2}\lceil\frac{k+1}{r+1}\rceil n$.

Note that $L(k,r,n)$ is essentially the lower bound in Lemma 4.3.1, so $\kappa(k,r) \geq 1$ for every $k > r$. Furthermore, Theorem 4.3.2 implies that $\kappa(k,r) \leq 2$ for every $k > r$, since the solution returned by Algorithm 5 contains at most $2L(k,r,n)$ edges for every input graph. In the following lemma we obtain a tighter bound for $\kappa(k,r)$ for some combinations of the parameters k and r.

Lemma 4.3.7. *Let $k, r \in \mathbb{Z}_+$ be given such that $k > r \geq 3$ and $\lceil\frac{3(r+2)}{2}\rceil \leq k \leq 2r+1$. Then $\kappa(k,r) \geq \frac{6}{5}$.*

Proof. Let $G = (V,E)$ be some (k,r)-recoverable graph with more than k vertices. Consider first the number of bridges in G. If this number is greater than $r+1$ we reach a contradiction, since any $r+1$ bridges disconnect the graph into $r+2$ connected components with fewer than $k+1$ edges. We assume next that G is bridge-less, perhaps after augmenting it with at most $r+1$ edges.

Let $U \subset V$ denote the set of vertices in G with degree two. Consider the graph H obtained from G by removing all edges, which are not incident to some vertex in U. By choice of U, H is a union of paths and cycles. Any cycle in H must be disconnected from the rest of the graph, hence we can assume that H is a collection of paths. Paths in H, which have three edges or more represent pairs of vertices in U, which are connected. Let $P_1, \cdots, P_t$ be the set of paths of length three or more in H. We claim that their combined length is at most $\lceil\frac{3(r+1)}{2}\rceil$. Let $t' < t$ be the largest index such that the paths $P_1, \cdots, P_{t'}$ have a combined length of at most $\frac{3(r+1)}{2}$. Let the total number of edges in these paths be d, and let B be a set of $\lceil\frac{3(r+2)}{2}\rceil - d$ edges from the path $P_{t'+1}$. Consider the cut $C = B \cup \bigcup_{i=1}^{t'} P_i$, which contains exactly $\lceil\frac{3(r+2)}{2}\rceil \leq k$ edges. We count the number of connected components formed by removing the cut C from G. Since they belong to U, all internal vertices of the paths $P_1, \cdots, P_{t'}$ are isolated. Since each such path has three or more edges, the total number of vertices that were isolated from those paths is at least $\frac{2d}{3}$, and the number of formed connected components is at least $\frac{2d}{3}+1$. The number of extra components that were created by removing the edges of B is exactly $|B|-1$, which gives a total of

$$\frac{2d}{3} + 1 + \lceil\frac{3(r+2)}{2}\rceil - d - 1 \geq \frac{3(r+2)}{2} - \frac{d}{3} \geq r+2,$$

where the last inequality is due to $d \leq \frac{3(r+2)}{2}$. We reached a contradiction to the assumption that G is (k,r)-recoverable.

Assume next that G has no neighboring vertices of degree two. From the previous discussion, G can be augmented by at most $\frac{3(r+1)}{4}$ extra edges to obtain such a graph. In other words, we assume next that U is an independent set in G. Consider the graph $G' = (V', E')$ obtained from G by contracting exactly one edge, which is incident to every vertex $u \in U$. Put differently, G' is obtained from G by removing the vertices U and all their incident edges, and inserting for every $u \in U$ a direct edge connecting the two neighbors of u in $V \setminus U$. Note that

$$|E'| = |E| - |U| \quad \text{and} \quad |V'| = |V| - |U|.$$

Observe that the degree of every vertex $v \in V \setminus U$ in G' is the same as the degree of the corresponding vertex in G. By choice of U, this degree is at least three, hence

$$|E'| \geq \frac{3|V'|}{2}.$$

We conclude that

$$|E| = |E'| + |U| \geq \frac{3|V'|}{2} + |U| = \frac{3|V|}{2} - \frac{|U|}{2}. \tag{4.3}$$

Finally, we distinguish two cases. In the first case assume $|U| \geq \frac{3}{5}|V|$. In this case, since no two vertices of U are adjacent we have $|E| \geq 2|U| \geq \frac{6}{5}|V|$, as required. In the other case $|U| \leq \frac{3}{5}|V|$ and Equation (4.3) gives

$$|E| = \frac{3}{2}|V| - \frac{|U|}{2} \geq \frac{3}{2}|V| - \frac{3}{10}|V| = \frac{6}{5}|V|,$$

which finishes the proof.

□

The study of the parameters $\kappa(k, r)$, apart from being interesting quantities in their own right, also give a better analysis of Algorithm 5. Note that Theorem 4.3.2 proves that this algorithm returns a set of cardinality $2L(k, r, n)$ for every graph with n vertices. It follows that the approximation guarantee of Algorithm 5 on instances (k, r)-ARCC(ST) gets arbitrarily close to

$$\frac{2}{\kappa(k, r)},$$

when the number of vertices of the graph grows to infinity. In fact a $\frac{2}{\kappa(k,r)} + \epsilon$ factor can be guaranteed by assuming that the input instances are restricted to have at least a certain minimal number $n_0 = n(k, r, \epsilon)$ of vertices. From the previous discussion and Lemma 4.3.7 we obtain the following result.

Corollary 4.3.8. *Under the conditions of Lemma 4.3.7 on k and r, there is an approximation algorithm for (k,r)-ARCC(ST) with factor $\frac{5}{3}+\epsilon$ for every $\epsilon \geq 0$.*

4.3.4 An approximation for large k and fixed r

In this section we analyze a simple algorithm, which employs the technique of randomized rounding of an LP solution. We start by considering the IP $IP_{ARCC(ST)}$ and its natural LP relaxation, which we denote by $LP_{ARCC(ST)}$. We assume throughout this section that $k = \Omega(\log n)$ and that r is fixed. Note that $\frac{k}{r} = \Omega(\log n)$ follows from these assumptions.

The algorithm starts by solving the LP $LP_{ARCC(ST)}$. Since this LP has exponential size, the Ellipsoid method needs to be employed. In order to use the Ellipsoid algorithm, one needs to design a separation oracle for $LP_{ARCC(ST)}$. In other words, we need to show a polynomial time procedure that given a fractional vector $x \in \mathbb{Q}^E$, either asserts that it is feasible for the LP, or reports a constraint, which is violated by x. In our case this amounts to solving the *minimum $(r+2)$-cut problem (MrC)* in G, capacitated by the vector x: if this cut has capacity $k+1$ or higher then, by definition, the vector x is feasible. Otherwise, if this minimum cut has capacity strictly smaller than $k+1$, it corresponds to a violated constraint.

Unfortunately, the MrC problem is NP-hard. The only polynomial case is that of fixed r. An algorithm for this problem is given in the paper of Goldschmidt and Hochbaum [40].

Theorem 4.3.9 ([40])**.** *Given a capacitated graph $G = (V, E)$, the minimum r-cut in G can be found in time, which is equivalent to $O(n^{r^2})$ MC computations.*

Note that the NP-hardness of the MrC problem is the only limiting factor, which restricts the current algorithm to the case of fixed r.

Consider next an optimal fractional solution x^* to $LP_{ARCC(ST)}$. The constrains in the LP guarantee that the minimal $(r+2)$-cut in G, capacitated by the vector x^* is $c^* = k+1$. We are interested in the probability that in a randomly rounded solution y (which is obtained from x^* by setting y_e to be one with probability x^*_e, for every $e \in E$ independently) some $(r+2)$-cut $C \in \mathcal{C}_{r+2}$ satisfies

$$\sum_{e \in E(C)} y_e < (1-\delta)(k+1)$$

for some small δ, which we fix later. In fact, for simplicity, we will compute the larger probability that this cut satisfies

$$\sum_{e \in E(C)} y_e < (1-\delta) \sum_{e \in E(C)} x_e^*.$$

To this end let us set

$$\delta = \sqrt{\frac{4(r+2)\log n}{k+1}},$$

and note that the approximation guarantee of our algorithm is of the form $1 + c\delta$, for some constant c, hence this guarantee improves when $\frac{k}{r}$ grows.

To this end we perform an analysis, which is similar to the one in Section 3.3. We utilize the bounds in Theorem 3.3.4, and we use a generalized version of Karger's Lemma 3.3.3, which we state next.

Lemma 4.3.10 ([48]). *Let G be a graph and let γ be the size of the minimal capacity r-cut in G. Then for every half-integral $c \geq 1$, the number of r-cuts in G, which have size at most $c\gamma$ is at most $n^{2c(r-1)}$.*

Let us start by bounding the probability that some $(r+2)$-cut C is violating the aforementioned condition. We let c be the largest half integer smaller or equal to $\frac{x^*(E(C))}{c^*}$, as in Lemma 4.3.10. Using Theorem 3.3.4 we obtain

$$Pr[y(E(C)) < (1-\delta)x^*(E(C))] \leq e^{-\frac{4(r+2)\log n}{k+1}\frac{(k+1)c}{2}} = \frac{1}{n^{2(r+2)c}}. \tag{4.4}$$

Lemma 4.3.10 and the union bound now imply that the probability that some $(r+2)$-cut C' with $x^*(E(C')) \leq c(k+1)$ is violating the condition is at most

$$\frac{n^{2c(r+1)}}{n^{2c(r+2)}} = \frac{1}{n^{2c}}.$$

Finally, we compute the bound on the probability that some cut violates the condition as follows.

$$Pr[y \text{ satisfies the condition for every } r\text{-cut}] \leq \sum_{c \geq 1,\ c \text{ half-integral}} \frac{1}{n^{2c}} \leq \frac{1}{n}.$$

We henceforth assume that $k \geq 2(r+1)\log n$. Let us bound next the probability that the rounded solution y is significantly costlier than x^*. We set $\gamma = \frac{2}{\sqrt{n}}$ and use the fact that $x^*(E) \geq \frac{n}{2}\lfloor\frac{k+1}{r+1}\rfloor + 1 - k \geq n\log n$, which can be proved as a continuous analog of Lemma 4.3.4, in exactly the same way.

Alternatively, we can guarantee this inequality by adding the following set on constraints to the LP which, by Lemma 4.3.4, increases the optimal solution value by at most an additive factor of k.

$$\sum_{e \in E(C)} x_e \geq \lfloor \frac{k}{r+1} \rfloor + 1 \quad \forall C = (V_1, V \setminus V_1) \in \mathcal{C}_2(G). \tag{4.5}$$

We can now bound the desired probability as follows.

$$Pr[y^*(E) \geq (1+\gamma)x^*(E)] \leq e^{-\frac{x^*(E)}{3n/4}} \leq e^{-\frac{n \log n}{n}} = \frac{1}{n}.$$

We conclude that with probability $1-\frac{2}{n}$ the rounded solution both satisfies the previous condition, and it admits no violating $(r+2)$-cuts. Let $S \subset E$ denote a set of edges corresponding to such a vector y. It remains to augment such a vector S to a feasible solution for ARCC(ST). Note that every $(r+2)$-cut in (V, S) contains at least $(1-\delta)(k+1)$ edges. To this end we use Algorithm 5 to complete S to a feasible solution. The complete description of the algorithm is given as Algorithm 6. The following theorem states the main result of this section.

Algorithm 6 *Input: A graph $G = (V, E)$, $k, r \in \mathbb{Z}_+$. Output: A subset of edges $U \subset E$.*

1: Solve $LP_{ARCC(ST)}$ and obtain a fractional solution x^*.
2: Randomly round x^* to obtain y.
3: $A \leftarrow \{e \in E : y_e = 1\}$.
4: $G' \leftarrow (V, E \setminus A)$.
5: $k' \leftarrow \lceil \delta(k+1) + r \rceil$.
6: $A' \leftarrow$ Algorithm 5(G', k', r)
7: $U \leftarrow A \cup A'$.
8: Return U.

Theorem 4.3.11. *Let $r \in \mathbb{Z}_+$ be fixed. Let $G = (V, E)$ and $k \in \mathbb{Z}_+$ such that $k \geq 2(r+1)\log n$. Then with probability greater than $1 - \frac{2}{n}$, Algorithm 6 returns an (k, r)-recoverable set of edges of size $(1 + O(\sqrt{\frac{\log n}{k}}))OPT$, where OPT denotes the size of an optimal (k, r)-recoverable set of edges for G.*

Proof. Let us prove the feasibility of U. Consider some $(r+2)$-cut C. We assume that A satisfies the required condition, namely that $|A \cap C| \geq (1 - \delta)(k+1)$. By choice of k' we know that the loop in step 3 of Algorithm 5

runs for at least $\frac{\delta(k+1)}{r+1}$ iterations. To this end the argument in the proof of Theorem 4.3.2 proves that $|A' \cap C| \geq \delta(k+1)$, unless A already contained an entire 2-cut $C' \subset C$ of G, implying that $|A \cap C| \geq k+1$ holds.

Let us compute the approximation factor. Let OPT denote the optimal solution value. From the discussion preceding the theorem we can assume $|A| \leq (1+\gamma)OPT \leq (1+\delta)OPT$. From Theorem 4.3.2 and Lemma 4.3.1 we can bound

$$|A'| \leq (\frac{k'+1}{r+1}+1)(n+1) \leq (\frac{\delta(k+1)+r+1}{r+1}+1)(n+1) \leq 4\delta OPT.$$

We conclude that $|U| = |A| + |A'| \leq (1+5\delta)OPT = (1+O(\sqrt{\frac{\log n}{k}}))OPT$, as required. □

4.4 Conclusions

This chapter treated the ST and SP problems in the ARCC model. We focused on the unweighted case, which revealed a clean combinatorial characterization of the feasible sets. This, in turn, led to several exact and approximate algorithms for various variants of the two problems.

The uniform scenario set, given by the upper bound k, allows for exponentially large sets of scenarios. As a result, and despite them being covering problems, this results in potentially intractable corresponding feasibility problems. Indeed, we showed that problem of deciding whether a set of edges is (k,r)-recoverable is NP-hard, both for the ARCC(SP) and the ARCC(ST) problems. A $2-\epsilon$ hardness-of-approximation result was obtained for ARCC(SP).

The feasible set the ARCC(SP) was characterized for the case $k=1$. The feasible set of the ARCC(ST) problem was considered in relation to the similar m-ECSS problem.

On the positive side we developed exact algorithms for the $(1,1)$-ARCC(SP) and 2-approximation algorithms for the $(1,r)$-ARCC(SP) and the ARCC(ST) problems. All these algorithms are combinatorial and simple to implement. A better approximation guarantee was attained for ARCC(SP) using an LP-based algorithm for certain parameters k and r.

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

- The complexity of ARCC(SP) in the cases of fixed k and fixed r remains open. In the case $k = 1$ 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 (k, 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.
- Study the ARCC(SP) problem without the requirement that s and t need to be connected in the first-stage solution.
- The complexity of the (k, r)-ARCC(ST) problem remains open for some combinations of the parameters k and r.
- Study the parameters $\kappa(k, r)$, namely find better lower bounds for the number of edges in a minimal (k, r)-recoverable graph with n vertices.

Chapter 5

Conclusions

This thesis studied several new models of robust combinatorial optimization. While most old models dealt with uncertainty in the underlying structure in an implicit manner, our new models try to model the structural failure as the essence of the uncertainty. In all models the decision maker assumes the existence of an adversary which, having observed the chosen initial solution, removed some of the structures' resources. The removed resources comprise the materialized failure scenario.

To allow the often imperative level of redundancy in the solutions, all our models assumed that the feasible set is an upper ideal. Put differently, every superset of a feasible solution is feasible as well. This feature classifies all problem studied in this thesis as covering problems. This approach helped to reduce the level of conservatism of the robust solutions.

We coarsely divided the different models into two parts, according to the nature of the failure scenario set. Chapter 3 dealt with models which admit nonuniform scenario sets. A failure scenario set is not uniform if it is not defined via an upper bound on the number of possible resource elements allowed in a single failure scenario. Chapter 4 treated the ARCC model, which assumes a uniform failure scenario set, but also allows recovery action. Hence the latter model can be viewed as a two-stage problem. Unlike classical two-stage optimization, however, in the ARCC model the decision maker is only required to specify a first stage decision, while the requirement is that every possible failure can be recovered by a suitable second stage augmentation, limited by the recovery budget.

The first model defined and analyzed is the ERCC model, which assumes an explicit description of all possible realizations of failures. An input in this model provides an explicit list of all possible subsets of resources, which may fail in the stage of solution implementation. This model treats resources in a nonuniform way, allowing e.g. some subsets of k resources to comprise a failure set, while disallowing other such subsets. Our first result shows

that the robust counterpart of essentially any graph problem in this model contains the SC problem as a special case. This result immediately puts a logarithmic lower bound on the best approximation algorithms for these problems. We then show that this lower bound can be matched with several algorithms. While some algorithms employ standard LP-rounding techniques, other algorithms are more combinatorial in nature, and often give a better worst case approximation guarantee. As a result we proved the existence of approximation algorithms with a logarithmic approximation guarantee for a number of problems, including ERCC(ST), ERCC(S2S) and ERCC(MO).

The general hardness-of-approximation result for ERCC motivated our search for better approximation algorithms for restricted variants of ERCC. Along these lines we focused on the ERCC(SP) problem, although several results can be generalized to other graph problems as well. We showed that by restricting admissible inputs to have a bounded number of scenarios, or a bounded cardinality of each failure scenario, one can obtain better approximation algorithms, and in some cases, even exact polynomial algorithms.

We later switched our attention to the SIRCC model, which defines the set of admissible failure scenarios implicitly. This model refines the well-studied IRCC model, in which the set of scenarios is given by a single integer upper bound k, and every set of k resources is one failure scenario. In SIRCC, the scenario set is given by an upper bound k, as well as a set U of faulty resources. The scenario set contains as a scenario every subset of U with at most k elements. While this modification seems insignificant, it turns out that the complexity of some robust counterparts changes dramatically. To demonstrate this we showed that while IRCC(SP) admits polynomial time algorithms, SIRCC(SP) is NP-hard, and in some cases, hard to approximate. On the positive side, we show that SIRCC(SP) can be solved exactly in polynomial for some special cases, including the case $k = 1$, the case of directed acyclic graphs and bounded k, and the case of SRP graphs. We also present a fractional variant of the problem, and display an integrality gap result, which translates into a factor $k + 1$ approximation algorithm for SIRCC(SP).

In Chapter 4 we focused on the ARCC(SP) and ARCC(ST) problems. Unlike all previously mentioned problems, it is shown that for both these problems the problem of deciding the feasibility of a certain subgraph is an NP-hard problem. The feasible set of ARCC(SP) was studied, and a constant factor approximation algorithm was given for the case of a single edge failure ($k = 1$). The ARCC(ST) problem was shown to admit a simple combinatorial 2-approximation algorithm, as well as an LP-based algorithm with a better approximation guarantee for certain combinations of the input parameters.

Further structural results were given for both problems.

Bibliography

[1] D. Adjiashvili, S. Stiller, and R. Zenklusen. The robust path problem. Technical report, Institute for Operations Research, ETH, Zürich, 2012.

[2] D. Adjiashvili and R. Zenklusen. An s-t connection problem with adaptability. *Discrete Applied Mathematics*, 159:695–705, 2011.

[3] D. Adjiashvili and R. Zenklusen. A graph connectivity problem with adaptability. Technical report, Institute for Operations Research, ETH, Zürich, 2012.

[4] D. Adjiashvili and R. Zenklusen. Structural robustness in combinatorial optimization. Technical report, Institute for Operations Research, ETH, Zürich, 2012.

[5] H. Aissi, C. Bazgan, and D. Vanderpooten. Complexity of the min-max (regret) versions of cut problems. In Xiaotie Deng and Ding-Zhu Du, editors, *Algorithms and Computation*, volume 3827 of *Lecture Notes in Computer Science*, pages 789–798. Springer Berlin / Heidelberg, 2005.

[6] H. Aissi, C. Bazgan, and D. Vanderpooten. Complexity of the minmax and minmax regret assignment problems. *Operations Research Letters*, 33(6):634 – 640, 2005.

[7] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation of minmax and minmax regret versions of some combinatorial optimization problems. *European Journal of Operational Research*, 179(2):281 – 290, 2007.

[8] H. Aissi, C. Bazgan, and D. Vanderpooten. Minmax and minmax regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427 – 438, 2009.

[9] P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. In Giancarlo Bongiovanni, Daniel Bovet, and Giuseppe Di Battista, editors, *Algorithms and Complexity*, volume 1203 of *Lecture Notes in Computer Science*, pages 288–298. Springer Berlin / Heidelberg, 1997.

[10] A. Armon and U. Zwick. Multicriteria global minimum cuts. *Algorithmica*, 46(1):15–26, 2006.

[11] A. Asadpour, M. X. Goemans, A. Madry, S. O. Gharan, and A. Saberi. An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 379–389, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[12] A. Atamtürk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55:662–673, July 2007.

[13] I. Averbakh and V. Lebedev. Interval data minmax regret network optimization problems. *Discrete Applied Mathematics*, 138:289–301, 2004.

[14] A. Bar-Noy, S. Khuller, and 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.

[15] A. Ben-Tal, S. Boyd, and A. Nemirovski. Extending scope of robust optimization: Comprehensive robust counterparts of uncertain problems. *Mathematical Programming*, 107(1):63–89, 2006.

[16] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton series in applied mathematics. Princeton Univ. Press, Princeton, NJ, 2009.

[17] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.

[18] P. Berman, S. Raskhodnikova, and G. Ruan. Finding sparser directed spanners. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 424–435. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

[19] D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM review*, 53:464–501, 2011.

[20] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming Series B*, 98:2003, 2003.

[21] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52:pp. 35–53, 2004.

[22] C. Büsing. The exact subgraph recoverable robust shortest path problem. In *Robust and Online Large-Scale Optimization*, pages 231–248. 2009.

[23] J. Byrka, A. Srinivasan, and C. Swamy. Fault-tolerant facility location: A randomized dependent lp-rounding algorithm. In Friedrich Eisenbrand and F. Shepherd, editors, *IPCO 2010*, volume 6080 of *Lecture Notes in Computer Science*, pages 244–257. Springer Berlin / Heidelberg, 2010.

[24] S. Chechik, M. Langberg, David Peleg, and L. Roditty. Fault-tolerant spanners for general graphs. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 435–444, New York, NY, USA, 2009. ACM.

[25] S. Chechik and D. Peleg. Robust Fault Tolerant uncapacitated facility location. In Jean-Yves Marion and Thomas Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science - STACS 2010*, pages 191–202, 2010.

[26] J. Cheriyan and R. Thurimella. Approximating Minimum-Size k-Connected Spanning Subgraphs via Matching. *SIAM J. Comput*, 30:292–301, 2000.

[27] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[28] H. W. Corley and D. Y. Sha. Most vital links and nodes in weighted networks. *Operations Research Letters*, 1(4):157–160, 1982.

[29] A. Czumaj and H. Zhao. Fault-tolerant geometric spanners. In *Proceedings of the nineteenth annual symposium on Computational geometry*, SCG '03, pages 1–10, New York, NY, USA, 2003. ACM.

[30] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):pp. 101–111, 1960.

[31] George Dantzig. *Linear Programming and Extensions.* Princeton University Press, 1998.

[32] K. Dhamdhere, V. Goyal, R. Ravi, and M. Singh. How to pay, come what may: approximation algorithms for demand-robust covering problems. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 367 – 376, October 2005.

[33] M. Dinitz and R. Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '11, pages 169–178, New York, NY, USA, 2011. ACM.

[34] M. Elkin and D. Peleg. $(1+\epsilon, \beta)$-spanner constructions for general graphs. *SIAM J. Comput.*, 33:608–631, March 2004.

[35] U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45:634–652, July 1998.

[36] U. Feige, K. Jain, M. Mahdian, and V. Mirrokni. Robust combinatorial optimization with exponential scenarios. In Matteo Fischetti and David Williamson, editors, *IPCO 2007*, volume 4513 of *Lecture Notes in Computer Science*, pages 439–453. Springer Berlin / Heidelberg, 2007.

[37] H. N. Gabow, M. X. Goemans, . Tardos, and D. P. Williamson. Approximating the smallest k-edge connected spanning subgraph by lp-rounding. *Networks*, 53(4):345–357, 2009.

[38] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989.

[39] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[40] O. Goldschmidt and D. S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of Operations Research*, 19(1):24–37, 1994.

[41] D. Golovin, V. Goyal, and R. Ravi. Pay today for a rainy day: Improved approximation algorithms for demand-robust min-cut and shortest path problems. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006*, volume 3884 of *Lecture Notes in Computer Science*, pages 206–217. Springer Berlin / Heidelberg, 2006.

[42] V. Goyal. *Combinatorial optimization under uncertainty.* PhD thesis, Carnegie Mellon University, USA, 2008.

[43] M. Guarisco, C. Friedrich, C. Balderer, M. Laumanns, and M. Zdrallek. A grid operation model: Resource demand for an adequate incident management in high-, medium-, and low-voltage grids. *International Journal of Electrical Power & Energy Systems*, 31(9):504 – 511, 2009.

[44] V. Guruswami and R. Saket. On the inapproximability of vertex cover on k-partite k-uniform hypergraphs. In *Proceedings of the 37th international colloquium conference on Automata, languages and programming*, ICALP'10, pages 360–371, Berlin, Heidelberg, 2010. Springer-Verlag.

[45] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17:36–42, February 1992.

[46] K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:448, 1998.

[47] K. Jain and V. Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. In Klaus Jansen and Samir Khuller, editors, *Approximation Algorithms for Combinatorial Optimization*, volume 1913 of *Lecture Notes in Computer Science*, pages 177–182. Springer Berlin / Heidelberg, 2000.

[48] D. R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-out algorithm. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 21–30, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[49] A. Kasperski and P. Zieliski. On the approximability of minmax (regret) network optimization problems. *Information Processing Letters*, 109(5):262 – 266, 2009.

[50] R. Khandekar, G. Kortsarz, V. Mirrokni, and M. Salavatipour. Two-stage robust network design with exponential scenarios. In Dan Halperin and Kurt Mehlhorn, editors, *ESA 2008*, volume 5193 of *Lecture Notes in Computer Science*, pages 589–600. Springer Berlin / Heidelberg, 2008.

[51] S. Khuller. Approximation algorithms for finding highly connected subgraphs. In *Approximation algorithms for NP-hard problems*, pages 236–265. PWS Publishing Co., Boston, MA, USA, 1997.

[52] G. Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30:432–450, 2001.

[53] G. Kortsarz and D. Peleg. Generating sparse 2-spanners. *Journal of Algorithms*, 17:222 – 236, 1994.

[54] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, Boston., 1997.

[55] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 545–554, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[56] C. Levcopoulos, G. Narasimhan, and M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, 32:2002, 1998.

[57] C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller. Recoverable robustness. Technical report, ARRIVAL-Project, August 2007.

[58] Elkin M. and D. Peleg. Approximating k-spanner problems for $k > 2$. *Theoretical Computer Science*, 337(1-3):249 – 277, 2005.

[59] W. Mader. Minimale n-fach zusammenhngende graphen mit maximaler kantenzahl. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1971:201–207, 2009.

[60] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 229–234, New York, NY, USA, 1988. ACM.

[61] C. Puhl. Recoverable robust shortest path problems. Preprint 034-2008, Intitute of Mathematics, Technische Universität Berlin, 2008.

[62] A. Schrijver. *Combinatorial Optimization — Polyhedra and Efficiency.* Springer, 2003.

[63] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):pp. 1154–1157, 1973.

[64] C. Swamy and D. B. Shmoys. Fault-tolerant facility location. *ACM Trans. Algorithms*, 4:51:1–51:27, August 2008.

[65] D. J. Thuente. Technical noteduality theory for generalized linear programs with computational methods. *Operations Research*, 28(4):1005–1011, 1980.

[66] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):pp. 70–79, 1987.

[67] G. Yu and J. Yang. On the robust shortest path problem. *Computers & Operations Research*, 25(6):457 – 468, 1998.

[68] P. Zieliński. The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research*, 158(3):570 – 576, 2004.

Appendix A

Acronyms

Computational problems

BOSP Bi-objective shortest path

CIP Covering integer program

CSFM Constrained submodular function maximization

EC Edge cover

FTUFL Fault-tolerant uncapacitated facility location

GLP Generalized linear programming

GSN Generalized Steiner network

HVC Hypergraph vertex cover

IMCF Integer minimum cost flow

IC Interval cover

IP Integer programming

LP Linear programming

IXLP Inexact linear programming

m**-ECSS** Minimum m-edge connected spanning subgraph

m**-EDP** m-edge disjoint paths

MC Minimum cut

MCF Minimum cost flow

MIP Mixed-integer programming

MMC Minimum multi-cut

MO Matroid optimization

MrC Minimum r-cut

SC Set cover

STF Steiner forest

SFM Submodular function minimization

SkS Sparsest k-spanner

SMB Simultaneous matroid basis

SP Shortest path

ST Spanning tree

STT Steiner tree

UFL Uncapacitated facility location

VC Vertex cover

Robust models

AERCC Adaptive explicit robust covering counterpart

ARCC Adaptable robust covering counterpart

CRC Cost robust counterpart

ECRC Explicit cost robust counterpart

ERCC Explicit robust covering counterpart

EDRC Explicit demand robust counterpart

ICRC Interval cost robust counterpart

IDRC Implicit demand robust counterpart

IRCC Implicit robust covering counterpart

RECRC Min-max regret explicit cost robust counterpart

SIRCC Subset implicit recoverable robust counterpart

Miscellaneous

SPR Series-parallel

Short Curriculum Vitae

David Adjiashvili

Born on March 4, 1984

citizen of Israel

Education

2008-2012	**Ph.D., Mathematics** ETH Zürich
2005-2007	**M.Sc., Computer Science** Weizmann Institute of Science, Rehovot
2003-2005	**B.Sc., Computer Science** Tel-Aviv University
1996-2002	**High school** Ironi Dalet Tel-Aviv

Work Experience

2008-2012	**Teaching Assistant** ETH Zürich
2008	**Algorithms and Software Developer** Primesense Ltd. Israel

Academic Visists

2010	Operations Research Center MIT Cambridge