

Searching for many defective edges in hypergraphs

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen
University zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften
genehmigte Dissertation

vorgelegt von

Diplom-Mathematikerin

Jessica Emonts, geb. Troes

aus

Trier.

Berichter: Universitätsprofessor Dr. Eberhard Triesch
Universitätsprofessor Dr. Ir. Arie M.C.A. Koster

Tag der mündlichen Prüfung: 05. Juli 2013

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftliche Mitarbeiterin am Lehrstuhl II für Mathematik an der RWTH Aachen. Für die Unterstützung, die ich in dieser Zeit erhalten habe, möchte ich mich gerne bedanken.

Zuerst gilt mein Dank meinem Doktorvater, Professor Eberhard Triesch. Mit seinen vielen wertvollen Ratschlägen und seiner großen Begeisterung für die Mathematik trug er nicht nur entscheidend zum Gelingen dieser Arbeit bei, sondern auch zur Freude an der Forschung an den Suchproblemen.

Des Weiteren möchte ich Herrn Professor Arie Koster herzlich danken, der sich als Korreferent dieser Arbeit zur Verfügung gestellt hat.

Allen meinen Kollegen am Lehrstuhl danke ich für das großartige Arbeitsklima und die schöne Zeit, die ich am Lehrstuhl hatte.

Für das Korrekturlesen und für Ihre Unterstützung nicht nur beim Schreiben dieser Arbeit möchte ich meiner lieben Freundin Anke Reuters danken.

Nicht zuletzt geht der Dank an meine Familie.

An meinen Bruder Christoph, an meine Eltern Sylvie und Werner Troes, die mir immer die Freiheit gelassen haben, meinen eigenen Weg zu gehen und mir auf diesem stets zu Seite standen.

Und ganz besonders an meinen Mann, Michael Emonts, der immer an das Gelingen und mich glaubte. Geduldig hat er alle Höhen und Tiefen der Promotion mit mir durchgestanden, meine 'Hieroglyphen' durchgesehen und sich meine Vorträge immer wieder angehört.

Contents

1	Introduction	1
1.1	(Hyper-) graphs: notations and definitions	1
1.2	Search processes	1
1.3	Group testing on graphs and hypergraphs	2
1.4	A search algorithm on graphs	4
2	Hypergraphs of bounded rank	7
2.1	Hypergraphs of rank 3	8
2.1.1	A search algorithm on hypergraphs of rank 3	10
2.1.2	An upper bound for the number of tests	26
2.2	Hypergraphs of bounded rank r	29
2.2.1	A search algorithm on hypergraphs of bounded rank r	30
2.2.2	An upper bound for the number of tests	45
3	Uniform hypergraphs	47
3.1	3-uniform hypergraphs	47
3.1.1	A search algorithm on 3-uniform hypergraphs	73
3.1.2	An upper bound for the number of tests	85
4	Conclusions and Outlook	97

Preface

Scarcely a day passes - at least for me - without searching for all sort of things: my keys, my left shoe, the shortest way, or an article that I have recently read. More generally, I am searching for an object x^* in a non-empty set \mathcal{S} . For example, the magazine with the article in question among all magazines I have ever read. To find the unknown object, one may take certain steps, like, for example, reading the table of contents of a magazine. These steps provide us with information such as “yes, the article is in the magazine” or “no, the article is not in this magazine”. We call the set of possible steps *tests*, and we assume that both the set \mathcal{S} and the range of possible answers to a test are finite. The pair of \mathcal{S} and a family of admissible tests \mathcal{F} is termed a *search process*.

One form of search process arose from the following problem: During World War II, the United States Public Health Service examined blood samples of all draftees for syphilis. By introducing an antigen to the blood samples, the blood clumped if antibodies were contained in the sample which indicated the disease. Since those tests were costly and in most cases negative, Dorfman (s. [Dor43]) suggested that one should pool several blood samples and test the pool instead of individual blood samples. If a pool of blood samples is clear of antibodies, so are all blood samples that are contained in the pool. Otherwise, if a pool of several blood samples contains antibodies, then at least one of those samples contains antibodies and further tests are necessary. Consequently, the following question arises: How can you minimize the number of tests by pooling samples?

Although Dorfman’s suggestion has never been implemented, his idea was applied to different tasks and he established a new topic in search theory, the group testing problem:

We are given a set N of n items and some of these items are somehow distinguished. Let M be the set of those items; we will call them *defective*. To discern between defective and non-defective (we refer to them as *good*) elements, a family of tests is at our disposal which maps any subset $X \subset M$ to “1” if at least one defective element lies in X , and “0” otherwise. For small m , group testing can certainly save tests. Now, this raises the question of how many tests an optimal algorithm needs to find m defective elements among $n = |N|$ items? The answer surely depends on which of the items are defective. To compare algorithms, either the average number of tests or the number of tests in the worst case is usually considered. In the following, we will always regard the number of tests in the worst case.

In contrast to the blood testing, we will assume that we know the exact number $m = |M|$ of defective items. If $m = 1$, we can find the defective item by always testing approximately half of the candidates. By doing so we need $\lceil \log_2 n \rceil$ tests, which is optimal. While the problem for $m = 1$ is quite simple, it is far more difficult for $m = 2$. Several papers had dealt with the problem (including [CH80], [Tos80], [CH81], [CHL82], and [Tos82]) until Aigner reformulated

the problem. Aigner suggested in [Aig86] that searching for two defective items can be viewed as the search for one defective edge in a graph. This problem was considered again by [AT93] and [Dam94] who finally proved an upper bound for the worst-case number of tests that is tight for infinitely many graphs.

By exchanging the roles of positive tests and negative tests the number of tests does not change and so the problem presents itself as follows: Given is a graph $G = (V, E)$ with a defective edge $e^* \in E$ and we have to find e^* . This exchange makes it more comfortable to extend the problem to larger unknown edge sets. If more than one edge is defective, say $d \geq 1$ edges, Du and Hwang (s. [DH93]) conjectured that at most

$$d \cdot (\lceil \log_2(|E|/d) \rceil + c)$$

tests are necessary to find all defective edges, this was proven by Johann (s. [Joh02]) for $c = 7$. Korneffel (s. [Kor07]) presented another solution that improved the result by an iterative algorithm. With a slight modification, Korneffel and Triesch (s. [KT08]) gave an algorithm that finds all defective edges by

$$d \cdot \lceil \log_2(|E|/d) \rceil + 9$$

tests, even if the number of defective edges is unknown.

If, in the first place, not pairs but triples or larger sets are defective, then we can interpret the problem as a group testing problem on hypergraphs. Triesch proved in [Tri96] that finding a defective edge in a hypergraph of rank $\leq r$ costs at most

$$\lceil \log_2 |E| \rceil + r - 1$$

tests (for $r = 2$ it yields Damaschke's result). Combining those two extensions of the problem, we are looking for d unknown defective hyperedges in a hypergraph. Chen and Hwang gave in [CH07] an algorithm that finds all defective edges in a hypergraph of rank r by at most

$$d \cdot \lceil \log_2 |E| \rceil + (r - 1)^{\lfloor r/2 \rfloor} \cdot d^r + o(d^r)$$

tests. In this thesis, we will show in Chapter 2 that on the one hand there is some constant $c > 0$ and infinitely many hypergraphs such that in the worst case at least

$$d \cdot \log_2 |E| + c \cdot d^{\frac{1}{2}}$$

tests are needed to find d defective edges. On the other hand, there is some constant $C > 0$ such that there is a search algorithm that finds all defective edges in a hypergraph of rank r by at most

$$d \cdot \lceil \log_2 |E| \rceil + C \cdot d^{\frac{1}{2}}$$

tests. In the third chapter, we will consider 3-uniform hypergraphs and we will give an algorithm that proves the conjecture of Du and Hwang for 3-uniform hypergraphs.

Chapter 1

Introduction

1.1 (Hyper-) graphs: notations and definitions

We start this chapter with some basic terms of graph and hypergraph theory, for more details please see [Bol98] or [Ber89].

A *hypergraph* H is an ordered pair $H = (V, E)$ comprising a *vertex set* $V = V(H)$ and a set of *hyperedges* (short: *edges*) $E = E(H)$ where $E \subseteq 2^V$. An edge e is called a *loop* if $|e| = 1$. We say a hypergraph is *simple*, if none of its edges is included in another. If not stated otherwise, we assume that all hypergraphs considered in the following are simple. Two vertices v and w are *adjacent* or *neighbors* if there is an edge $e \in E$ with $\{v, w\} \subseteq e$; we also say e *joins* v and w . Two edges $e, f \in E$ are *incident*, if $e \cap f \neq \emptyset$ and every edge e is incident to all vertices $v \in e$. Further, let $r(H) := \max\{|e| : e \in E\}$ denote the *rank* of H . If all edges in H have cardinality r , we say H is *r-uniform*. A *graph* is a hypergraph of rank 2. A graph in which every pair of distinct vertices is connected by an edge is called a *complete graph*. Let $S \subset V$, then set $E(S) := \{e \in E : e \subset S\}$. A vertex set $S \subset V$ is *independent* if $e \not\subset S$ for every $e \in E$ with $|e| \geq 2$. Is, moreover, $|e \cap S| \leq 1$ for every $e \in E$, then we call S *strongly independent*. Finally, we call a set $X \subset V$ a *vertex cover* if $e \cap X \neq \emptyset$ for each $e \in E$.

1.2 Search processes

Let \mathcal{S} be a non-empty set with a distinguished element $x^* \in \mathcal{S}$ and \mathcal{F} a family of functions on \mathcal{S} . We call \mathcal{S} a *search domain* and \mathcal{F} a *test family*. Functions in \mathcal{F} are called *tests* or *questions*. The pair $(\mathcal{S}, \mathcal{F})$ is a *search process*, and if both \mathcal{S} and \mathcal{F} are finite, the search is said to be *combinatorial*. Our objective is to identify x^* . On this account, we choose tests $f_1, f_2, \dots \in \mathcal{F}$ and receive as answer the values $f_1(x^*), f_2(x^*), \dots$ depending on x^* . If these values $f_1(x^*), f_2(x^*), \dots$ determine x^* uniquely, we call $\{f_1, f_2, \dots\}$ a (successful) *search algorithm*.

Among other things, the range of possible answers classifies different search processes. We will consider test functions with only two possible answers; such a search process is called *binary*. A crucial distinction concerns the nature of admissible search algorithms. An algorithm is called *predetermined* if all tests $f_1, f_2, \dots \in \mathcal{F}$ are stipulated in advance of the search process. Otherwise, if the choice of a function $f_i \in \mathcal{F}$ depends on the previous answers $f_1(x^*), f_2(x^*), \dots, f_{i-1}(x^*)$, the algorithm is said to be *sequential*.

Suppose the search domain $\mathcal{S} = \{x_1, x_2, \dots, x_n\}$ is linearly ordered $x_1 < x_2 < \dots < x_n$. A test $f \in \mathcal{F}$ is monotone if $f(x_1) \leq f(x_2) \leq \dots \leq f(x_n)$. Let \mathcal{F}_{mon} be the set of monotone tests. A search process $(\mathcal{S}, \mathcal{F})$ with $\mathcal{F} \subseteq \mathcal{F}_{mon}$ is *alphabetic*. Let $(\mathcal{S}, \mathcal{F})$ be a search process and \mathcal{A} a set of admissible algorithms that perform a successful search for every $x^* \in \mathcal{S}$, then $l(A, x^*)$ denotes the number of tests an algorithm $A \in \mathcal{A}$ needs to determine $x^* \in \mathcal{S}$ uniquely. The complexity of a search process $(\mathcal{S}, \mathcal{F})$ can be described by

$$c(\mathcal{S}, \mathcal{F}) = \min_{A \in \mathcal{A}} \max_{x^* \in \mathcal{S}} l(A, x^*),$$

called the *worst-case complexity*. A binary search process $(\mathcal{S}, \mathcal{F})$ is bounded from below by

$$c(\mathcal{S}, \mathcal{F}) \geq \lceil \log_2 |\mathcal{S}| \rceil$$

where $\lceil x \rceil$ denotes the smallest integer not less than x . This bound is usually called the *information-theoretic bound*. A proof for the information-theoretic bound can be found for example in [Aig88]. We also refer to [Aig88] for more details about search theory.

1.3 Group testing on graphs and hypergraphs

In the thesis at hand, we consider one form of combinatorial search: the *group testing problem*. Suppose, we are given a set N of n items and some of them are somehow distinguished. We call the distinguished items *defective*, and we say that all other items are *good*. Let $M \subset N$ be the set of defective items. To discern good and defective items, we choose subsets $X \subset N$ and ask whether $X \cap M = \emptyset$. Let $f : 2^N \rightarrow \{0, 1\}$ with

$$f(X) = 1 \Leftrightarrow X \cap M \neq \emptyset$$

for each $X \subset N$. We call $f(X)$ a *test* of $X \subset N$ and we say the test is *positive* if $f(X) = 1$, or, otherwise if $f(X) = 0$, it is *negative*. Usually, we assume that the number m of defective items is known. In that case, the search domain \mathcal{S} comprises all subsets $X \subset N$ with $|X| = m$. Suppose $m = 1$, then we find the defective element by the following procedure:

Test a set $X \subset N$ with $|X| = \lceil n/2 \rceil$ elements. If $f(X) = 1$, then the defective element lies in X , thus proceed the search on X . Otherwise, if $f(X) = 0$, then the defective element lies in $N \setminus X$ and the search proceeds of course on $N \setminus X$. By this so called *halving procedure*, we find a defective element by at most

$$\lceil \log_2 n \rceil$$

tests, which is the information-theoretic lower bound. Thus, the algorithm is optimal. While the case $m = 1$ is quite easy, the case $m = 2$ is far more difficult. In 1980, Chang and Hwang conjectured in [CH80] that finding two defective elements lying in two disjoint sets, one of cardinality l and the other of cardinality t , costs in the worst case

$$\lceil \log_2(l \cdot t) \rceil$$

tests, which they proved one year later in [CH81]. In the meantime, Tošić ([Tos80]) gave an optimal strategy for infinitely many values of n in order to find two defective elements in a set of n elements. Let n_k denote the largest n such that k tests suffice to find 2 defective elements

in a set of n elements. Chang, Hwang and Lin [CHL82] determined two years later an upper bound u_k and a lower bound l_k for n_k such that $l_k/u_k > 0.95$.

The search of two elements can be interpreted as the search of one defective edge in a complete graph. The first one to formulate the search on graphs in an explicit way was Aigner in [Aig86], where he considered a slightly different problem: a test has three possible outcomes, namely the tested set contains two, one, or no defective elements.

However, the group testing problem on graphs can be described as follows: Let $G = (V, E)$ be a graph containing one unknown defective edge $e^* = \{u, v\}$. The test of $A \subset V$ has now two possible outcomes, either $f(A) = 0$, if $\{u, v\} \cap A = \emptyset$ or otherwise $f(A) = 1$ in case of $\{u, v\} \cap A \neq \emptyset$. In the first case $e^* \in E(V \setminus A)$ and in the second case $e^* \notin E(V \setminus A)$. For convenience, the roles of positive and negative tests are interchanged: a test of any set $A \subset V$ is positive, if and only if $e^* \in E(A)$. Let $c(G)$ denote the worst-case cost of finding one defective edge in a graph. It is obvious that the predefinition of positive and negative tests has no influence on $c(G)$. Aigner conjectured (s. [Aig88], p. 144) that

$$c(G) \leq \lceil \log_2 |E(G)| \rceil + c$$

for some constant $c > 0$, which was proven in 1993 by Althöfer and Triesch in [AT93], who showed that

$$c(G) \leq \lceil \log_2 |E(G)| \rceil + 3.$$

In 1994 Damaschke [Dam94] improved the result and showed that

$$c(G) \leq \lceil \log_2 |E(G)| \rceil + 1.$$

This represents a tight bound for infinitely many graphs G (s. e.g. [Aig88], p. 133). Two years later, in 1996, Triesch generalized the search problem to hypergraphs. Let $H = (V, E)$ be a hypergraph with a rank of at most r , and let $c(H)$ denote the worst-case complexity of finding one defective edge in H . Triesch proved in [Tri96] that

$$c(H) \leq \lceil \log_2 |E| \rceil + r - 1. \quad (1.1)$$

Suppose now, not only one but $d > 1$ edges are defective. Let $c(G, d)$ and $c(H, d)$ denote the worst-case cost of finding d defective edges in a graph G or in a hypergraph H . We may repeat the search for one defective edge until we have found all defective edges. Since the average of defective edges is one out of $|E|/d$, Du and Hwang ([DH93], p. 242) conjectured:

Conjecture 1.1. [DH93] *Let $H = (V, E)$ be a hypergraph of rank r with $d \geq 1$ defective edges, then there is a constant c_r such that*

$$c(H, d) \leq d \cdot \left(\left\lceil \log_2 \frac{|E|}{d} \right\rceil + c_r \right)$$

holds.

For $r = 2$, Johann ([Joh02]) gave an algorithm that proved that the conjuncture is true for $c_2 = 7$. This result was improved five years later by Korneffel ([Kor07]), who gave another algorithm that works iteratively and uses (1.1) which shows that

$$c(G, d) \leq d \cdot \left(\left\lceil \log_2 \frac{|E|}{d} \right\rceil + 5 \right) + \sqrt{2d}.$$

With a slight adaption of the algorithm Korneffel and Triesch were able to prove in [KT08] that the conjecture remains true if d is unknown; in that case we have $c_2 = 9$. For unknown d , Hwang ([Hwa05]) proposed an algorithm as early as 2005, which was revised by Chen [Che11] in 2011. Eventually, the algorithm of Hwang / Chen identified all defective edges by at most

$$d \cdot \lceil \log_2 |E| \rceil + d^2 + 3d + 1$$

tests.

1.4 A search algorithm on graphs

In the next chapter we will use the search algorithm for graphs and its ideas to find defective edges in arbitrary hypergraphs. Therefore, we summarize the results of [KT08] and [Tri96] which we will apply in the following.

Let $G = (V, E)$ be a graph and $D \subset E$ be the set of defective edges where $d := |D|$ is known. Further, let $>$ be a total order on a subset $X \subset V$. We say $x \in X$ lies left of $y \in X$ (and y lies right of x) if $x > y$, and in case of $e = \{x, y\} \in E$ we call x the left and y the right endvertex of e . The rightmost vertex of any set $Y \subset X$ is the one vertex $y \in Y$ that lies right of all other vertices $z \in Y \setminus \{y\}$. Define for any subset X with a total order $>$ and a vertex $x \in X$ the following two sets, firstly,

$$X_x := \{y \in X : \exists \{x, y\} \in E(X) \text{ and } x > y\}$$

and secondly

$$A_x := \{e \in E(X) : x \text{ is left endvertex of } e\}.$$

Please note that $|X_x| = |A_x|$ and $E(X) = \bigcup_{x \in X} A_x$. We call $Y \subseteq X$ a rightmost set if there is a vertex x with $Y = X_x$. A vertex set X is alphabetically sorted, if

$$x > y \Rightarrow |A_x| \geq |A_y|.$$

We call the union $Y := Y' \cup \{x\}$ of a rightmost set $Y' \subset X$ and a single vertex $\{x\} \in X \setminus Y'$ selectable if Y is alphabetically sorted with respect to X .

Let now $X \subseteq V$ be an alphabetically sorted set. Using an alphabetical search on the sets A_x according to ([Tri96]), we find a defective edge $e \in E(X)$ with rightmost left endvertex $x \in e$ by at most

$$\lceil \log_2 |E(X)| \rceil + 1$$

tests. For arbitrary hypergraphs we have:

Theorem 1.2. [Tri96] *Let $H = (V, E)$ be a hypergraph with rank at most r and at least one defective edge. Further let V be alphabetically sorted. Then there is an algorithm that detects a defective edge with rightmost left endvertex by at most*

$$\lceil \log_2 |E| \rceil + r - 1$$

tests.

Since x is rightmost among all left endvertices in X , all edges in A_y are good for $x > y$. Good edges do not interfere with the result of a test, hence, we consider known good edges as deleted. Finally, if X is a vertex set with $E(X) = \emptyset$, we call X *free*.

We quote the following lemmas without proof.

Lemma 1.3. [KT08] *Let W be an alphabetically ordered set and $X \subset W$ a selectable subset. Further let $\{x, y\} \in E(X) \cap D$, $x > y$, be the defective edge that we have found with the above procedure. Suppose we have deleted all known good edges, then we can sort $W' := W \setminus \{x\}$ alphabetically with respect to the new edge set such that $W'_y = \emptyset$ and $W'_z = \emptyset$ for all $z \in W'$ with $W_z = \emptyset$.*

Lemma 1.4. [KT08] *For every alphabetically sorted set W with $|E(W)| \geq |E|/2d$, there exists a selectable subset $Y \subset W$ with*

$$\frac{|E|}{d} \geq |E(Y)| \geq \frac{|E|}{2d}.$$

Lemma 1.5. [KT08] *Let X be a free vertex set, $y \notin X$ and $\{x_1, y\}, \dots, \{x_n, y\}$ known defective edges in $E(X \cup \{y\})$ for some n . If there is at least one unknown defective edge in $E(X \cup \{y\})$, then we can find another defective edge in $X \cup \{y\}$ with at most*

$$\lceil \log_2 |E(X \cup \{y\})| - n \rceil$$

tests.

Next, we present the search algorithm of Korneffel and Triesch (compare: [KT08]).

A search algorithm on graphs:

Let V be alphabetically sorted. Set $W := V$, $X := \emptyset$. If

1. $E(W) \geq |E|/2d$: Construct a selectable subset $Y \subset W$ with $|E|/d \leq |E(Y)| \leq |E|/2d$ according to Lemma 1.4.
 $E(W) < |E|/2d$: Set $Y := W$.
2. Test Y . If
 $f(Y) = 1$: Find a defective edge $\{x, y\}$, ($x > y$) with rightmost left end vertex by Triesch's algorithm. Sort $W' := W \setminus \{x\}$ according to Lemma 1.3, then replace W by W' and set $X := X \cup \{x\}$. Delete all good edges and repeat step 1.
 $f(Y) = 0$: Delete all good edges. If $E(W) = \emptyset$ set $\bar{X} := X$ and go to step 3, else, repeat step 1.
3. Choose an $x \in \bar{X}$, set $\bar{X} := \bar{X} \setminus \{x\}$ and $\bar{W} := W \setminus \{w \in W : \{x, w\} \text{ is a known defective edge}\}$.
4. Choose a maximum set $Y \subset \bar{W}$ such that $|E(\{x\} \cup Y)| \leq |E|/d$.
5. Test $\{x\} \cup Y$. If
 $f(\{x\} \cup Y) = 0$: Remove the good edges and set $\bar{W} := \bar{W} \setminus Y$.
 $f(\{x\} \cup Y) = 1$: Find a defective edge $\{x, z\}$ according to Lemma 1.5 and set $\bar{W} := \bar{W} \setminus \{z\}$.
 $\bar{W} \neq \emptyset$: Repeat step 4.
 $\bar{W} = \emptyset$: $\bar{X} \neq \emptyset$: Repeat step 3.
 $\bar{X} = \emptyset$: $E(X) \neq \emptyset$: Test X if $f(X) = 1$: Set $W := X$ and repeat step 1.
 $f(X) = 0$: Stop.
 $E(X) = \emptyset$: Stop.

The algorithm performs the alphabetic search only on selectable sets $Y \subseteq W$, that are sets composed of a rightmost set and a single vertex. Suppose $e \in E(Y)$ is some defective edge that has been found in $E(Y)$ and let x be its left endvertex. Then,

$$W_y = Y_y = \emptyset$$

for all $y \in Y$ that lie right of x . Due to Lemma 1.3, a vertex $v \in V$ which is not the left endvertex of any edge, is in all later orders still not the left endvertex of any edge. Therefore, the algorithm will not add v to X . Consequently, the set $E(X)$ contains no known defective edges.

Theorem 1.6. [Kor07] *Let $G = (V, E)$ be a graph. Then the complexity $c(G, d)$ of finding d defective edges is bound by*

$$c(G, d) \leq d \cdot \left(\left\lceil \log_2 \left(\frac{|E|}{d} \right) \right\rceil + 5 \right) + \sqrt{2d}.$$

When the algorithm enters step 2, W is free. Lemma 1.3 prevents the right endvertex of a defective edge, that has been found in step 1, left endvertex of any edge with respect to all future sorting.

Suppose now we do not know the number of defective edges. Then, the algorithm can be modified as follows:

Let d_K be the number of defective edges found so far. Then compute $|E|/d$ with $\max\{1, d_K\}$ instead of d . Thus, $|E|/d$ decreases and we receive:

Theorem 1.7. [KT08] *Let $G = (V, E)$ be a graph. Then the complexity $c(G, d)$ of finding d defective edges in case of d unknown is bound by*

$$c(G, d) \leq d \left(\left\lceil \log_2 \left(\frac{|E|}{d} \right) \right\rceil + 9 \right).$$

Alternatively, we can perform each search of defective edges with the maximum number of edges; then we receive:

Theorem 1.8. [KT08] *Let $G = (V, E)$ be a graph. Then the complexity $c(G, d)$ of finding d defective edges in case of d unknown is bounded by*

$$c(G, d) \leq d \left(\left\lceil \log_2 (|E|) \right\rceil + 3 \right) + \sqrt{2 \cdot d + \frac{1}{4}} - \frac{1}{2}.$$

Chapter 2

Hypergraphs of bounded rank

In Chapter 2 we consider the group testing problem on hypergraphs of bounded rank. The first results are delivered by Chen and Hwang who presented an algorithm in [CH07] that identifies all defective edges with at most

$$d \cdot \lceil \log_2 |E| \rceil + (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + o(d^r)$$

tests, where d is not necessarily known and r denotes the rank of H .

Before we present an alternative search algorithm, we will consider the following hypergraph:

Let $H_{r,n} = H(V_{r,n}, E)$ be a hypergraph with vertex set $V_{r,n}$ and edge set E for $r \geq 3$ and $n > 0$, $r, n \in \mathbb{N}$. Further let $V_{r,n} := V_1 \cup V_2 \cup \dots \cup V_r$ be the union of r disjoint vertex sets each of cardinality n and let E be given by

$$E := \bigcup_{i=1}^r \binom{V_i}{2} \cup \underbrace{\{\{v_1, v_2, \dots, v_r\} : v_1 \in V_1, v_2 \in V_2, \dots, v_r \in V_r\}}_{=: E_r},$$

where $\binom{V_i}{2} := \{(x, y) : x, y \in V_i\}$ denotes all vertex pairs in V_i . For any edge $e \in E_r$ of cardinality r let

$$D := \bigcup_{i=1}^r \binom{V_i}{2} \cup \{e\}$$

be the set of defective edges.

Every test of a vertex set U with

$$\max_{1 \leq i \leq r} |U \cap V_i| \geq 2$$

receives a positive answer. Consequently, to identify e , every edge in E_r has to be tested individually. In the worst case, one would have to carry out $|E_r| = n^r$ tests. (One test less is needed, if it is known that exactly one edge in E_r is defective.) Let $n \geq r \geq 3$, then

$$|E| = r \cdot \binom{n}{2} + n^r \leq 2 \cdot n^r$$

and

$$d = r \cdot \binom{n}{2} + 1 \leq \frac{n^2 r}{2}.$$

For n sufficiently large, that is with $n \geq r^2 \cdot \log_2 n + r$ (for example $n \geq r^4$), we receive for $r \geq 3$

$$\begin{aligned} n^r &\geq \frac{1}{2}n^3 + \frac{1}{2}n^r \\ &= \frac{1}{2}n^2 \cdot n + \frac{1}{2}(n^2)^{\frac{r}{2}} \\ &\geq \frac{1}{2}n^2 \cdot (r^2 \cdot \log_2 n + r) + \frac{1}{2} \left(\frac{2}{r}\right)^{\frac{r}{2}} \cdot \left(\frac{n^2 r}{2}\right)^{\frac{r}{2}} \\ &\geq \frac{n^2 r}{2} \cdot (r \cdot \log_2 n + 1) + \frac{1}{2} \left(\frac{2}{r}\right)^{\frac{r}{2}} \cdot d^{\frac{r}{2}} \\ &\geq d \cdot \log_2(2 \cdot n^r) + \frac{1}{2} \left(\frac{2}{r}\right)^{\frac{r}{2}} \cdot d^{\frac{r}{2}} \\ &\geq d \cdot \log_2(|E|) + \frac{1}{2} \left(\frac{2}{r}\right)^{\frac{r}{2}} \cdot d^{\frac{r}{2}}. \end{aligned}$$

Thus, we can conclude the following:

Theorem 2.1. *For each natural number $r \geq 3$ there are infinitely many hypergraphs H of rank r with d defective edges, such that*

$$c(H, d) \geq d \cdot \log_2(|E|) + c \cdot d^{\frac{r}{2}},$$

where $c = \frac{1}{2} \left(\frac{2}{r}\right)^{\frac{r}{2}}$.

In the following we show that for every natural number r there exists also some constant C_r such that the worst-case costs of finding d defective edges in a hypergraph H of rank r are bounded by

$$c(H, d) \leq d \cdot \lceil \log_2 |E| \rceil + C_r \cdot d^{\frac{r}{2}}.$$

We will however present, in the following, two search algorithms to find all d defective edges in a hypergraph. The first algorithm only works for hypergraphs of rank 3; in return it provides a much better constant than the second algorithm that works on hypergraphs of any bounded rank.

2.1 Hypergraphs of rank 3

In this section we adapt the search algorithm proposed in [KT08] for hypergraphs of rank 3. On that account, let us recall the rough concept of the algorithm: It is operated iteratively in two steps. In the first step it finds defective edges by Triesch's alphabetical search [Tri96] and deletes one vertex from every defective edge that was found. This step is repeated as long as the vertex set contains defective edges. Let X be the set of deleted vertices, and let W denote the set of the other vertices, i.e., the vertices that were not deleted. In the second step the algorithm examines for every $x \in X$ the edge set $E(\{x\} \cup W)$ and finds all defective edges in $E(\{x\} \cup W)$ by a halving procedure. Hence, after step 2, all defective edges that join a vertex in X to a vertex in

W are detected. In consequence of the alphabetical search, X does not contain known defective edges, and the algorithm continues its search on X beginning again with step 1.

What happens now if we use the algorithm on a hypergraph of rank 3?

Step 1 works analogously on hypergraphs since Triesch's search finds defective edges in hypergraphs of any bounded rank. In the second step the situation differs: an edge e of rank 3 that joins a vertex $x \in X$ to a vertex $y \in W$ contains one further vertex, say v . If $v \in W$, e will be scanned in step 2. Otherwise, if $v \in X$, the algorithm will miss e .

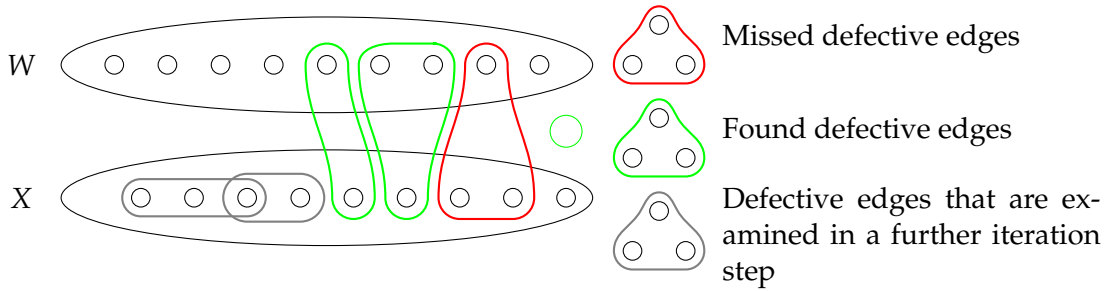


Figure 2.1: Using the graph algorithm on a hypergraph of rank 3.

The idea is the following: We start our search by the search algorithm for graphs, which finds a number of defective edges and partitions V into s sets that do not contain defective edges. Let X_1, X_2, \dots, X_s be those sets; then, for each edge of rank 3, one of the following cases is true:

- $|e \cap X_i| = 2$ and $|e \cap X_j| = 1$ for $1 \leq i < j \leq s$,
- $|e \cap X_i| = 1$ and $|e \cap X_j| = 2$ for $1 \leq i < j \leq s$, or
- $|e \cap X_i| = |e \cap X_j| = |e \cap X_k| = 1$ for $1 \leq i < j < k \leq s$.

We will see that the algorithm finds all defective edges of rank ≤ 2 as well as all defective edges of the first type. In order to find all defective edges of the second type, we test each set X_i with suitable subsets of $\bigcup_{j=1}^{i-1} X_j$, ($i = 2, \dots, s$). Afterwards, we find all defective edges of the latter kind by testing each set $(X_i)_{2 \leq i \leq s-1}$ with sets U_j , that cover all pairs in

$$(x, y) \in \bigcup_{j=1}^{i-1} X_j \times \bigcup_{k=i+1}^s X_k,$$

$\{x, y\} \notin D$, ($i = 2, \dots, s$).

Let us start by adapting some definitions.

Definition 2.2. Let $H = H(V, E)$ be a hypergraph and $D \subset E$ a set of defective edges with cardinality $d := |D|$. Once we have found a defective edge e , we say e is a known defective edge. Let $D_K \subset D$ denote the set of known defective edges with cardinality $d_K := |D_K|$. We will always use capital K to indicate that the set of known defective edges is considered. We call a set $X \subset V$ D_K -independent if $e \not\subseteq X$ for every $e \in D_K$ with $|e| \geq 2$. Thus every free vertex set is also D_K -independent. Please note that a test of a set which is not D_K -independent is pointless since we know that the answer will be positive. Hence we

test only sets that are D_K -independent, we will call such tests reasonable.

Define for a vertex set $X \subset V$ the following edge sets:

$$\begin{aligned} D(X) &:= \{e \in D : e \cap X \neq \emptyset\}, \\ \dot{D}(X) &:= \{e \in D : e \subset X\}, \\ D(X, Y) &:= \{e \in \dot{D}(X \cup Y) : e \cap X \neq \emptyset\}, \\ D(X, Y, Z) &:= \{e \in \dot{D}(X \cup Y \cup Z) : |e \cap X| = |e \cap Y| = |e \cap Z| = 1\}, \\ D_K(X) &:= \{e \in D_K : e \cap X \neq \emptyset\}, \\ \dot{D}_K(X) &:= \{e \in D_K : e \subset X\}, \\ D_K(X, Y) &:= \{e \in \dot{D}_K(X \cup Y) : e \cap X \neq \emptyset\}, \\ D_K(X, Y, Z) &:= \{e \in \dot{D}_K(X \cup Y \cup Z) : |e \cap X| = |e \cap Y| = |e \cap Z| = 1\} \end{aligned}$$

with cardinalities $d(X) := |D(X)|$, $\dot{d}(X) := |\dot{D}(X)|$, $d(X, Y) := |D(X, Y)|$, $d(X, Y, Z) := |D(X, Y, Z)|$, $d_K(X) := |D_K(X)|$, $\dot{d}_K(X) := |\dot{D}_K(X)|$, $d_K(X, Y) := |D_K(X, Y)|$, $d_K(X, Y, Z) := |D_K(X, Y, Z)|$. Let $X \subset V$ be a free vertex set, then set

$$\Gamma(X) := \{y \in V \setminus X : \exists e \in D \text{ with } e \subset \{y\} \cup X\}$$

and

$$\Gamma_K(X) := \{y \in V \setminus X : \exists e \in D_K, e \subset \{y\} \cup X\}.$$

In the following, if not stated otherwise, let $H = H(V, E)$ be a simple hypergraph of rank 3 with defective edge set $D \subset E$, for which the number of defective edges $d := |D|$ is not necessarily known. Please note, if H is not simple, defective edges might not be distinguishable. Assume e, g are two edges with $e \subset g$, further let e be defective. Then every test that contains g tests also e and is thus positive regardless whether g is defective or not.

2.1.1 A search algorithm on hypergraphs of rank 3

Our search algorithm performs three stages one at a time, which we will consider separately.

Stage I

The first stage is more or less the search algorithm for graphs (see [KT08]) with some minor modifications and reads as follows:

Let W be an alphabetically sorted set and $e \in \dot{D}(W)$ a defective edge with rightmost left endvertex $x \in e$ (x is left endvertex of e if x lies left of all vertices $y \in e \setminus \{x\}$). Further, let $W' := W \setminus \{x\}$, $W^> := \{y \in W : y > x\}$ and $W^< := \{z \in W : z < x\}$. Then Lemma 1.3 remains true for arbitrary hypergraphs:

Lemma 2.3. *After removing all good edges, W' can be sorted alphabetically with respect to the new edge set such that $A_z(W') = \emptyset$ for $z \in W^<$.*

Proof. Since all edges in $E(W^<)$ are identified as good and are thus removed, $E(W^<) = \emptyset$. Hence, it suffices to reorder the vertices in $W^>$ with respect to the new edge set and leave $W^<$ unchanged and right of $W^>$. Then $A_z(W') = \emptyset$ for $z \in W^<$, that is in particular, $A_v(W') = \emptyset$ for all vertices $v \in e \setminus \{x\}$, as well as for all vertices for which $A_v(W) = \emptyset$ already holds with respect to the former ordering. \square

The above Lemma makes sure that all vertices right of x are reordered such that they cannot be the left endvertex of any edge in $E(W')$.

Lemma 2.4. *Let X be a free vertex set and $x \notin X$ with $\{x\} \notin D$. Further let $\{x\} \cup X$ be D_K -independent and $\mathring{D}(\{x\} \cup X) \neq \emptyset$. Then there is a search algorithm that detects all defective edges in $\mathring{D}(\{x\} \cup X)$ by at most*

$$\mathring{d}(\{x\} \cup X) \cdot (\lceil \log_2 |E| \rceil + 4)$$

tests.

Proof. Since X is free, all edges in $E(\{x\} \cup X)$ are incident to x . By deleting x from every edge, we receive a graph on X . Lemma 2.4 follows now directly from Theorem 1.8. \square

Lemma 2.5. *Let U be a free vertex set and $x \notin U$ with $\{x\} \notin D$ such that $\mathring{d}_K(\{x\} \cup U) = 1$. Further let $d_\Delta := \mathring{d}(\{x\} \cup U) - 1$ denote the number of unknown defective edges in $\mathring{D}(\{x\} \cup U)$. There is an algorithm that needs at most*

$$d_\Delta \cdot (\lceil \log_2 |E| \rceil + 4)$$

tests to find all defective edges in $\mathring{D}(\{x\} \cup U)$ if $d_\Delta > 0$. Otherwise, the algorithm needs at most two tests to prove that there is just one defective edge in $\mathring{D}(\{x\} \cup U)$.

Proof. Let e be the one known defective edge in $\mathring{D}_K(\{x\} \cup U)$.

- $|e| = 2$: Seeing that H is a simple hypergraph and that U is free, we have

$$\mathring{D}(\{x\} \cup U) = \{e\} \cup \mathring{D}(\{x\} \cup [U \setminus e]).$$

Hence, test $\{x\} \cup (U \setminus e)$. If $f(\{x\} \cup (U \setminus e)) = 0$, then e is the only defective edge in $\mathring{D}(\{x\} \cup U)$ and we are done. Otherwise, use Lemma 2.4 to find all defective edges in $\mathring{D}(\{x\} \cup (U \setminus e))$.

- $|e| = 3$: Let $e = \{x, y, z\}$. With the same argumentation as before in Lemma 2.4, we can use the graph algorithm on U to find all defective edges in $\mathring{D}(\{x\} \cup U)$. Then sort the vertices in W alphabetically such that z is the rightmost vertex in W and go to step 1. Following the argumentation of Korneffel and Triesch in [KT08], we need at most

$$d_\Delta \cdot (\lceil \log_2 |E| \rceil + 2) + \mathring{d}(\{x\} \cup U) + \sqrt{2 \cdot \mathring{d}(\{x\} \cup U) + \frac{1}{4}} - \frac{1}{2} \stackrel{d_\Delta \geq 1}{\leq} d_\Delta \cdot (\lceil \log_2 |E| \rceil + 4)$$

tests to find all defective edges in $\mathring{D}(\{x\} \cup U)$. Please note that for $d_\Delta = 0$ this means that exactly two tests are necessary. \square

Algorithm - Stage I

Let V be alphabetically sorted. Set $X_1 := V, i := 1$.

1. Test X_i if
 - $f(X_i) = 1$: Find a defective edge $e \in \mathring{D}(X_i)$ with rightmost left endvertex by Triesch's search and remove all good edges. Let $x \in e$ be the left endvertex of e , set X_i to $X_i \setminus \{x\}$, and resort X_i according to Lemma 2.3. If $|e| \geq 2$, set X_{i+1} to $X_{i+1} \cup \{x\}$. Repeat step 1.
 - $f(X_i) = 0$: If $X_{i+1} = \emptyset$, set $s := i$, and go to stage II; else, go to step 2.
2. For each $x \in X_{i+1}$ find all defective edges in $\mathring{D}(\{x\} \cup X_i)$ by means of Lemma 2.5.
3. Set $i := i + 1$, sort X_i alphabetically, and go back to step 1.

In step 1 the algorithm tests X_i , finds, in case of $f(X_i) = 1$, one defective edge $e \in \mathring{D}(X_i)$, removes one vertex $x \in e$ from X_i , and repeats this step until X_i is free ($i = 1, \dots, s$). Since the algorithm adds x to X_{i+1} under the condition that $|e| \geq 2$, the sets X_1, X_2, \dots, X_s form a partition of $\{v \in V : \{v\} \notin D\}$, consisting of exactly those vertices that are not incident to a defective loop. Certainly, at the time when $f(X_1) = 0$, all defective loops are found and all unknown defective edges lie in

$$\mathring{D}(\{v \in V : \{v\} \notin D\}).$$

Hence, by continuing the search only on $\{v \in V : \{v\} \notin D\}$, we do not miss any defective edges. For convenience, we assume in the following that H does not contain any loops at all.

Consider Stage I for some fixed i : In step 1 the algorithm removes the left endvertex of each new-found defective edge and afterwards it resorts the remaining vertices such that

$$A_y(X_i) = \{e \in E(X_i) : y \text{ is left endvertex of } e\} = \emptyset$$

for all $y \in (g \cap X_i)$, $g \in D_K$. Consequently, the algorithm removes exactly one vertex of every known defective edge from X_i . That guarantees three things:

- Firstly, $\mathring{D}(X_{i+1})$ never contains known defective edges,
- secondly, for each $x \in X_{i+1}$ exactly one defective edge in $\mathring{D}(\{x\} \cup X_i)$ is known, and
- finally, no defective edge $e \in \mathring{D}(X_i \cup X_{i+1})$ with $|e \cap X_{i+1}| = 2$ is known.

Due to the second point, all constraints of Lemma 2.5 are satisfied and the algorithm finds indeed all defective edges in $\mathring{D}(\{x\} \cup X_i)$ during step 2 for each $x \in V \setminus \bigcup_{j=1}^i X_j$. So at the end of Stage I, the free sets X_1, X_2, \dots, X_s form a partition of V and all defective edges $e \in \mathring{D}(X_i \cup X_j)$, with $|e \cap X_j| = 1$, are known, whereas all defective edges $e \in \mathring{D}(X_i \cup X_j)$ with $|e \cap X_j| = 2$ are unknown ($1 \leq i < j \leq s$), see also Figure 2.2.

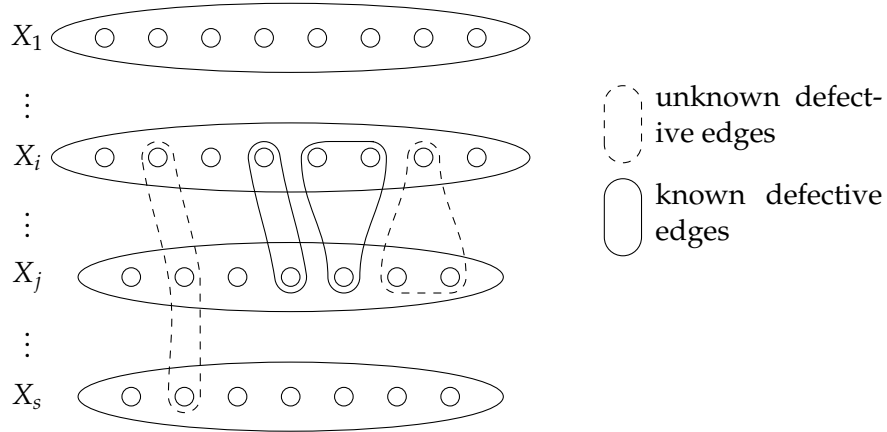


Figure 2.2: Sketch of known and unknown defective edges after Stage I

Constructing test sets

Following the idea that we have described at the beginning, we construct test sets out of X_1, X_2, \dots, X_s to identify the remaining unknown defective edges:

Lemma 2.6. *Let $X, Y, Z \subset V$ be three pairwise disjoint non-empty vertex sets and let X be free. Then there is an algorithm that constructs r sets U_1, U_2, \dots, U_r such that*

- (1) $r \leq 6 \cdot d_K$,
- (2) every pair $\{y, z\}$ with $y \in Y, z \in Z$, and $\{y, z\} \notin D_K$ is subset of some U_i ,
- (3) every vertex $v \in Y \cup Z$ lies in at most $6 \cdot \sqrt{d_K}$ different sets U_i , and
- (4) $|e \cap U_i| = 1$ or $|e \cap X| = |e \cap U_i \cap Y| = |e \cap U_i \cap Z| = 1$ for every $e \in \mathring{D}_K(X \cup U_i)$, ($i = 1, \dots, r$).

Proof. Instead of constructing such sets directly, we construct, for convenience, a graph G on $Y \cup Z$ such that any number of independent sets that cover all non-adjacent pairs in G satisfy (2) and (4): Therefore, let

$$D_K^2 \subset \binom{Y \cup Z}{2}$$

be a set of vertex pairs (cf.: Figure 2.3) such that

- for every $e \in D_K$ with $|e \cap Y| \geq 2$ there is exactly one pair $\{u, v\} \subseteq e \cap Y$ in D_K^2 ,
- for every $e \in D_K$ with $|e \cap Z| \geq 2$ there is exactly one pair $\{u, v\} \subseteq e \cap Z$ in D_K^2 ,
- for every $e \in D_K$ with $|e \cap Y| = |e \cap Z| = 1$ and $|e| = 2$ it is $e \in D_K^2$.

Consider now the graph G on $Y \cup Z$ with edge set D_K^2 :

If $y \in Y$ and $z \in Z$ such that $\{y, z\} \notin D_K$, then y and z are not adjacent in G . On the other hand, if two vertices $u, v \in Y \cup Z$ are not adjacent in G , then, on the one hand, $\{u, v\} \notin D_K$ and, on the other hand, there is no edge $e \in D_K$ with either $\{u, v\} = e \cap Y$ or $\{u, v\} = e \cap Z$. Thus, $|e \cap \{u, v\} \cap Y| \leq 1$ and $|e \cap \{u, v\} \cap Z| \leq 1$ for all $e \in \mathring{D}_K(\{u, v\} \cup X)$.

That is, if $U_1, U_2, \dots, U_r \subset (Y \cup Z)$ are independent sets in G that cover all non-adjacent pairs in G , then they satisfy (2) and (4). Now set for any $W \subseteq (Y \cup Z)$

$$d_2(W) := \left| \left\{ \{u, v\} \in D_K^2 : \{u, v\} \cap W \neq \emptyset \right\} \right|$$

and

$$\Gamma_2(W) := \left\{ y \in (Y \cup Z) \setminus W : \text{there is an } e \in D_K^2 \text{ with } y \in e \text{ and } e \cap W \neq \emptyset \right\}.$$

Please note, D_K^2 is usually not uniquely defined.

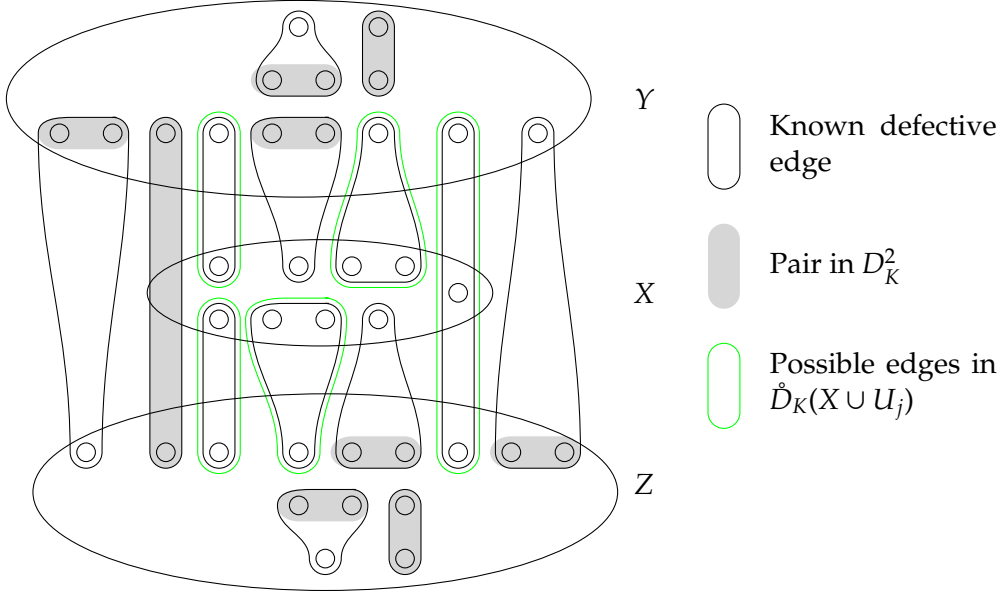


Figure 2.3: Sketch of D_K^2 for some hypergraph

The following algorithm constructs the desired sets:

Set $\mathcal{W} := \{\{v\} : v \in Y \cup Z\}$, $C_n := \emptyset$ for $1 \leq n \leq |Y| + |Z|$, and $i, j := 1$.

1. If there are two sets $W, W' \in \mathcal{W}$, $W \neq W'$, with $W \cup W'$ independent in G , set $\mathcal{W} := \mathcal{W} \cup \{W \cup W'\} \setminus \{W, W'\}$ and repeat step 1.
Else, choose an arbitrary $W \in \mathcal{W}$, set $\mathcal{W} := \mathcal{W} \setminus \{W\}$, and go to step 2.
2. If there is a vertex $w \in W$ with $d_2(\{w\}) \geq \sqrt{d_K}$, set $B_i := \{w\}$, $i := i + 1$, $W := W \setminus \{w\}$, and repeat step 2; else, go to step 3.
3. If there is a vertex $w \in W$ with $d_2(\{w\} \cup C_j) \leq 2 \cdot \sqrt{d_K}$, set $C_j := C_j \cup \{w\}$, $W := W \setminus \{w\}$, and repeat step 3, else, if
 $W \neq \emptyset$: Set $j := j + 1$ and repeat step 3.
 $W = \emptyset$: $\mathcal{W} \neq \emptyset$: Set $j := j + 1$, choose an arbitrary $W \in \mathcal{W}$, set $\mathcal{W} := \mathcal{W} \setminus \{W\}$, and repeat step 2.
 $\mathcal{W} = \emptyset$: Set $p := i$, $q := j$, and $B := \bigcup_{i=1}^p B_i$, afterwards set $i, n := 1$ and $j := 2$.
4. Set $U_n := B_i \cup (B_j \setminus \Gamma_2(B_i))$ and $n := n + 1$. If
 $j < p$: Set $j := j + 1$ and repeat step 4.
 $j = p$: Set $k := 1$ and go to step 5.

5. Set $U_n := B_i \cup (C_k \setminus \Gamma_2(B_i))$ and $n := n + 1$. If
 - $k < q$: Set $k := k + 1$, then repeat step 5.
 - $k = q$:
 - $i < p - 1$: Set $i := i + 1$ and $j := i + 1$, then repeat step 4.
 - $i = p - 1$: Set $i := i + 1$ and $k := 1$, then repeat step 5.
 - $i = p$: Set $k := 1$ and $l := 2$, then go to step 6.
6. Set $U_n := C_k \cup (C_l \setminus \Gamma_2(C_k))$ and $n = n + 1$. If
 - $l < q$: Set $l := l + 1$ and repeat step 6.
 - $l = q$: Set $\Gamma_{C_k} := \Gamma_2(C_k)$, choose a vertex $v \in \Gamma_{C_k}$, and go to step 7.
7. Set $U_n := \{v\} \cup (C_k \setminus \Gamma_2(\{v\}))$, $\Gamma_{C_k} := \Gamma_{C_k} \setminus \{v\}$ and $n := n + 1$. If
 - $\Gamma_{C_k} \neq \emptyset$: Choose a vertex $v \in \Gamma_{C_k}$ and repeat step 7.
 - $\Gamma_{C_k} = \emptyset$:
 - $k < q - 1$: Set $k := k + 1$ and $l := k + 1$, then repeat step 6.
 - $k = q - 1$: Set $U_{n+1} := C_q$ and $r := n + 1$, then stop.

In the first three steps the algorithm partitions $Y \cup Z$ into

$$B_1, B_2, \dots, B_p, C_1, C_2, \dots, C_q,$$

such that

- $|B_i| = 1$ for $i = 1, \dots, p$,
- C_j is independent in G and
- $d_2(C_j) \leq 2 \cdot \sqrt{d_K}$ for $j = 1, \dots, q$.

Now let us show that the sets U_1, U_2, \dots, U_r which are constructed in steps 4 - 7 cover indeed all non-adjacent pairs in G :

All non-adjacent pairs in G are covered

Let $\{u, v\} \subset (Y \cup Z)$ be some vertex pair which is not independent in G . Obviously, $u \notin \Gamma_2(\{v\})$ and also $v \notin \Gamma_2(\{u\})$. Besides, every vertex $x \in Y \cup Z$ lies in some set B_i or C_k .

1. Case $\{u, v\} \cap \mathbf{B} \neq \emptyset$:

Without loss of generality, we assume that $u \in B$. Let B_i be the set with $\{u\} = B_i$. If

- $v \in B_j$ with $j < i$ then $\{u, v\} \subseteq B_j \cup (B_i \setminus \Gamma_2(B_j))$,
- $v \in B_j$ with $j > i$ then $\{u, v\} \subseteq B_i \cup (B_j \setminus \Gamma_2(B_i))$,
- $v \in C_k$ then $\{u, v\} \subseteq B_i \cup (C_k \setminus \Gamma_2(B_i))$.

2. Case $\{u, v\} \cap \mathbf{B} = \emptyset$:

Let C_k be the set with $u \in C_k$ and C_l the set with $v \in C_l$, without loss of generality let $k \leq l$. If

- $k < l$ and $v \in C_l \setminus \Gamma_2(C_k)$ then $\{u, v\} \subseteq C_k \cup (C_l \setminus \Gamma_2(C_k))$,
- $k < l$ and $v \in C_l \cap \Gamma_2(C_k)$ then $\{u, v\} \subseteq \{v\} \cup (C_k \setminus \Gamma_2(\{v\}))$,
- $k = l < q$ then $\{u, v\} \subseteq C_k \cup (C_m \setminus \Gamma_2(C_k))$ for any $m > k$,
- $k = l = q$ then $\{u, v\} \subseteq C_q$.

Hence, in either case, $\{u, v\}$ is covered by some set U_n .

Number of sets \mathcal{U}_n

For any two sets $W, W' \in \mathcal{W}$ there is an edge $e \in D_K^2$ that joins a vertex from W and a vertex from W' and thus

$$\binom{|\mathcal{W}|}{2} \leq d_K,$$

which implies that $|\mathcal{W}| \leq 2 \cdot \sqrt{d_K}$. In step 2 the algorithm defines a set $B_i := \{w\}$ for each $w \in W$ with $d_2(W) \geq \sqrt{d_K}$. Afterwards the algorithm removes w from W . Accordingly, by the time the algorithm enters step 3,

$$d_2(\{w\}) < \sqrt{d_K}$$

for every $w \in W$, $W \in \mathcal{W}$. Hence, there are only two possible reasons for not enlarging C_j by some vertex $w \in W$ in step 3: either $d_2(C_j \cup \{w\}) > 2 \cdot \sqrt{d_K}$ for each $w \in W$, which implies that $d_2(C_j) > \sqrt{d_K}$ or $W = \emptyset$. Thus, for every $W \in \mathcal{W}$ there is at most one set $C_j \subseteq W$ with $d_2(C_j) \leq \sqrt{d_K}$. Consequently, we partition W into at most

$$\left\lceil \frac{d_2(W)}{\sqrt{d_K}} \right\rceil$$

smaller sets. Altogether, we receive an upper bound for the number of sets:

$$p + q \leq \sum_{W \in \mathcal{W}} \left\lceil \frac{d_2(W)}{\sqrt{d_K}} \right\rceil \leq |\mathcal{W}| + \frac{\sum_{W \in \mathcal{W}} d_2(W)}{\sqrt{d_K}} \leq 2 \cdot \sqrt{d_K} + \frac{2 \cdot d_K}{\sqrt{d_K}} = 4 \cdot \sqrt{d_K}.$$

The algorithm constructs the following sets

- $B_i \cup (B_j \setminus \Gamma_2(B_i))$ for $i = 1, \dots, p-1$ and $j = i+1, \dots, p$ in step 4,
- $B_i \cup (C_k \setminus \Gamma_2(B_i))$ for $i = 1, \dots, p$ and $k = 1, \dots, q$ in step 5,
- $C_k \cup (C_l \setminus \Gamma_2(C_k))$ for $k = 1, \dots, q-1$ and $l = k+1, \dots, q$ in step 6,
- $\{v\} \cup (C_k \setminus \Gamma_2(\{v\}))$ for $v \in \Gamma_2(C_k)$ and $k = 1, \dots, q-1$ in step 7.
- C_q in step 7

This amounts to

$$\begin{aligned} r &= \sum_{k=1}^p \left(\overbrace{p-i}^{\text{step 4}} + \overbrace{q}^{\text{step 5}} \right) + \sum_{k=1}^{q-1} \left(\overbrace{q-k}^{\text{step 6}} + \overbrace{|\Gamma_2(C_k)|}^{\text{step 7}} \right) + 1 \\ &= \binom{p}{2} + p \cdot q + \binom{q}{2} + 1 + \sum_{k=1}^q |\Gamma_2(C_k)| \\ &\leq \binom{p+q}{2} + 1 + q \cdot 2 \sqrt{d_K} \\ &\leq 2 \cdot d_K + 4 \cdot d_K \\ &\leq 6 \cdot d_K. \end{aligned}$$

Number of different sets U_n containing the same vertex

Certainly, the number of different sets U_n that contain the same vertex $u \in (Y \cup Z)$ depends on whether $u \in B$ or not:

1. Case $u \in B$:

Let B_{i_0} be the set with $u \in B_{i_0}$, then u lies in the following sets:

- $u \in B_i \cup (B_{i_0} \setminus \Gamma_2(B_i))$ for $i = 1, \dots, i_0 - 1$ with $u \notin \Gamma_2(B_i)$,
- $u \in B_{i_0} \cup (B_j \setminus \Gamma_2(B_{i_0}))$ for $j = i_0 + 1, \dots, p$,
- $u \in B_{i_0} \cup (C_k \setminus \Gamma_2(B_{i_0}))$ for $k = 1, \dots, q$.

These are in total at most

$$p - 1 + q \leq 2 \cdot \sqrt{d_K}$$

sets.

2. Case $u \notin B$:

Let C_{k_0} be the set with $u \in C_{k_0}$, then u lies at most in the following sets:

- $u \in B_i \cup (C_{k_0} \setminus \Gamma_2(B_i))$ for $i = 1, \dots, p$ with $u \notin \Gamma_2(B_i)$,
- $u \in C_k \cup (C_{k_0} \setminus \Gamma_2(C_k))$ for $k = 1, \dots, k_0 - 1$ with $u \notin \Gamma_2(C_k)$,
- $u \in C_{k_0} \cup (C_l \setminus \Gamma_2(C_{k_0}))$ for $l = k_0 + 1, \dots, q$,
- $u \in \{u\} \cup (C_k \setminus \Gamma_2(\{u\}))$ for $k = 1, \dots, q$ with $u \in \Gamma_2(C_k)$,
- $u \in \{v\} \cup (C_{k_0} \setminus \Gamma_2(\{v\}))$ for $v \in \Gamma_2(C_{k_0})$ with $u \notin \Gamma_2(\{v\})$,
- $u \in C_q$, if $k_0 = q$.

These are in total at most

$$\begin{aligned} & p + q - 1 + d_2(\{u\}) + d_2(C_{k_0}) + 1 \\ & \leq p + q + 2 \cdot d_2(C_{k_0}) \\ & \leq 2 \cdot \sqrt{d_k} + 2 \cdot 2 \cdot \sqrt{d_K} \\ & = 6 \cdot \sqrt{d_K} \end{aligned}$$

sets.

Thus, the sets U_1, U_2, \dots, U_r satisfy (1) - (4). □

Corollary 2.7. *Let X and Y be two non-empty sets, further let X be free. Then there is a partition of Y into*

$$p \leq 2 \cdot \sqrt{d_K}$$

D_K -independent sets

$$C_1, C_2, \dots, C_p,$$

such that for every $1 \leq i \leq p$

$$|e \cap C_i| \leq 1 \text{ for all } e \in \mathring{D}_K(X \cup C_i)$$

holds.

Proof. This follows directly from the proof of the previous Lemma by setting $Z = \emptyset$. The partition \mathcal{W} that the algorithm constructs in step 1 satisfies the constraint in Corollary 2.7. \square

Stage II

In Stage II our search algorithm finds all unknown defective edges that lie in the union of two free sets. Let us begin with some lemmas.

Lemma 2.8. *Let X and Y be two disjoint vertex sets such that*

- X is free,
- $X \cup Y$ is D_K -independent, and
- $\mathring{D}(X \cup Y) \neq \emptyset$.

Then there is a search algorithm that finds all defective edges $e \in \mathring{D}(X \cup Y)$ with $|e \cap X| = 2$ by at most

$$d_\Delta \cdot (\lceil \log_2 |E| \rceil + 5)$$

tests, where d_Δ denotes the number of defective edges in $\mathring{D}(X \cup Y)$ that are actually found by the search algorithm.

Proof. Set $Y' := Y$.

1. Find a defective edge $e \in \mathring{D}(X \cup Y')$ by Triesch's search.
2. Choose a vertex $y \in (Y' \cap e)$, find all defective edges in $\mathring{D}(\{y\} \cup X)$, and set $Y' := Y' \setminus \{y\}$.
3. Test $X \cup Y'$. If
 $f(X \cup Y') = 0$: stop.
 $f(X \cup Y') = 1$: repeat step 1.

Obviously, the algorithm finds at least all those defective edges $e \in \mathring{D}(X \cup Y)$ with $|e \cap X| = 2$ and $|e \cap Y| = 1$.

Now, for every $y \in Y \setminus Y'$, the algorithm finds one defective edge by Triesch's search, which costs at most

$$\lceil \log_2 |E| \rceil + 2$$

tests (cf. Theorem 1.2). That is, in particular, $|Y \setminus Y'| \leq d_\Delta$. If there is just one defective edge in $\mathring{D}(\{y\} \cup X)$, we need, according to Lemma 2.5, at most two tests for proof. Otherwise, we need

$$\lceil \log_2 |E| \rceil + 4$$

further tests per defective edge. Finally, in step 3, there is one test for each $y \in Y \setminus Y'$. This amounts to at most

$$\begin{aligned} & \sum_{\substack{y \in Y \setminus Y' : \\ \mathring{d}(\{y\} \cup X) = 1}} (\lceil \log_2 |E| \rceil + 2 + 2 + 1) \\ & + \sum_{\substack{y \in Y \setminus Y' : \\ \mathring{d}(\{y\} \cup X) > 1}} (\lceil \log_2 |E| \rceil + 2 + (\mathring{d}(\{y\} \cup X) - 1) \cdot (\lceil \log_2 |E| \rceil + 4) + 1) \\ & \leq d_\Delta \cdot (\lceil \log_2 |E| \rceil + 5) \end{aligned}$$

tests. \square

Lemma 2.9. *Let $X, C \subset V$ be two disjoint vertex sets such that*

- $\mathring{D}(C \cup X)$ contains no loops,
- X is free, and
- all defective edges $e \in \mathring{D}(C \cup X)$ with either $|e \cap C| \neq 1$ or $|e \cap X| \neq 1$ are unknown.

Then there is an algorithm that finds at least all those defective edges in $e \in \mathring{D}(C \cup X)$ with $|e \cap X| = 2$ by at most

$$d_\Delta \cdot (\lceil \log_2 |E| \rceil + 5) + \mathring{d}_K(C \cup X) + 1$$

tests, where d_Δ denotes the number defective edges which the algorithm actually finds.

Proof. Set $C' := C$

1. If $\mathring{D}_K(C' \cup X) = \emptyset$, go to step 3.
Otherwise, choose a vertex $v \in C'$ with $v \in e$ for some $e \in \mathring{D}_K(C' \cup X)$, set $C' := C' \setminus \{v\}$.

2. Test

$$\mathcal{T} := \{v\} \cup (X \setminus \Gamma_K(\{v\}))$$

and find, in case of $f(\mathcal{T}) = 1$, all defective edges $e \in \mathring{D}(\mathcal{T})$ by Lemma 2.4. Repeat step 1.

3. Test

$$\mathcal{T} := X \cup C'$$

and find in case of $f(\mathcal{T}) = 1$ all defective edges $e \in \mathring{D}(\mathcal{T})$ with $|e \cap X| = 2$ by Lemma 2.8.

Since X is free, every edge in $\mathring{D}(C \cup X)$ is incident to some vertex in C' . Thus, as long as $\mathring{D}_K(C' \cup X) \neq \emptyset$, there is some vertex $v \in C'$ which is incident to some known defective edge in $\mathring{D}_K(C' \cup X)$.

Let now $v \in C'$ be incident to some known edge $e \in \mathring{D}_K(C' \cup X)$. Certainly, since all defective edges $e \in \mathring{D}(C \cup X)$ have cardinality 2 and since X is free,

$$\mathring{D}_K(\{v\} \cup X \setminus \Gamma_K(\{v\})) = \emptyset.$$

Consequently, in step 2, all requirements of Lemma 2.4 are satisfied and the algorithm finds all defective edges in $\mathring{D}(\{v\} \cup X \setminus \Gamma_K(\{v\}))$ for each $v \in C \setminus C'$. These are in particular - since we consider only simple hypergraphs¹ - all defective edges $e \in \mathring{D}((C \setminus C') \cup X)$ with $|e \cap X| = 2$ and $|e \cap (C \setminus C')| = 1$.

In step 1 the algorithm removes vertices from C' until $C' \cup X$ is D_K -independent, which is possible because each defective edge in $\mathring{D}(C \cup X)$ is incident to some vertex in C . Now, X is free and $X \cup C'$ is D_K -independent. If $f(X \cup C') = 1$, the requirements of Lemma 2.8 are satisfied, and the algorithm finds all defective edges $e \in \mathring{D}(X \cup C)$ with $|e \cap X| = 2$ and $|e \cap C'| = 1$. That is, after step 3, all defective edges $e \in \mathring{D}(X \cup C)$ with $|e \cap X| = 2$ and $|e \cap C| = 1$ are known.

Number of tests:

There is one test $\mathcal{T} := \{v\} \cup (X \setminus \Gamma_K(\{v\}))$ for each $v \in C \setminus C'$ in step 2 and one test $\mathcal{T} := X \cup C'$ in step 3. Moreover, for every $v \in C \setminus C'$ there is at least one known defective edge and thus

¹Assume there is some unknown defective edge $e \in \mathring{D}(\{v\} \cup X)$ with $e \notin \mathring{D}(\{v\} \cup X \setminus \Gamma_K(\{v\}))$. Then $e \cap \Gamma_K(\{v\}) \neq \emptyset$, say $u \in e \cap \Gamma_K(\{v\})$. But then $\{u, v\} \in D_K$ and $\{u, v\} \subseteq e$ with $e \notin D_K$, which is a contradiction.

$|C \setminus C'| \leq \mathring{d}_K(C \cup X)$. If a test is positive, the algorithm finds defective edges by either Lemma 2.4 or Lemma 2.8. In the first case, the algorithm needs at most $\lceil \log_2 |E| \rceil + 4$ and in the second case at most $\lceil \log_2 |E| \rceil + 5$ tests per defective edge. This amounts to at most

$$d_\Delta \cdot (\lceil \log_2 |E| \rceil + 5) + \mathring{d}_K(C \cup X) + 1$$

tests. □

Algorithm - Stage II

Set $i := 2$.

1. Set $Y_i := \bigcup_{j=1}^{i-1} X_j$ and partition Y_i into p_i vertex sets $C_1^i, C_2^i, \dots, C_{p_i}^i$ such that

- $p_i \leq 2 \cdot \sqrt{d_K}$ and
- $|e \cap C_j^i| = 1$ for all $e \in \mathring{D}_K(X_i \cup C_j^i)$, ($j = 1, \dots, p_i$)

(cf. Corollary 2.7, Figure 2.4). Set $j := 1$.

2. Find all defective edges $e \in \mathring{D}(X_i \cup C_j^i)$ with $|e \cap X_i| = 2$ (cf. Lemma 2.9). If

$j < p_i$: Set $j := j + 1$ and repeat step 2.

$j = p_i$: $i < s$: Set $i := i + 1$ and repeat step 1.

$i = s$: Go to Stage III.

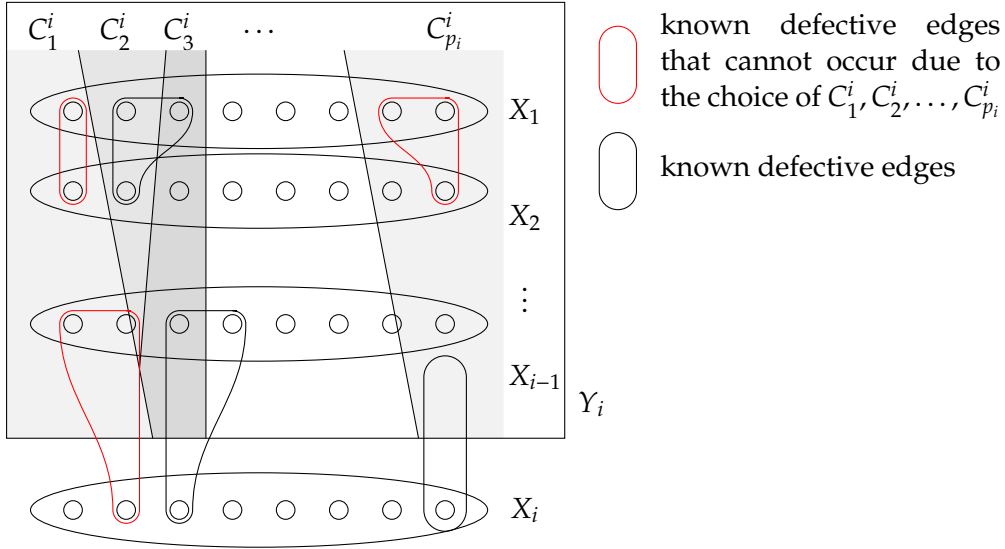


Figure 2.4: sketch of stage II

In step 1 the algorithm partitions

$$Y_i = \bigcup_{j=1}^{i-1} X_j$$

into p_i sets $C_1^i, C_2^i, \dots, C_{p_i}^i$ such that $|e \cap C_j^i| = 1$ for all $e \in \mathring{D}_K(X_i \cup C_j^i)$, ($i = 2, \dots, p$, $j = 1, \dots, p_i$). To find all defective edges in $\mathring{D}(X_i \cup C_j^i)$ by Lemma 3.28, the following constraint must be satisfied:

- there are no loops allowed in $\mathring{D}(X_i \cup C_j^i)$ (which we assume),
- X_i must be free (which we have already ascertained), and
- all $e \in \mathring{D}_K(X_i \cup C_j^i)$ with $|e \cap X_i| = 2$ must be unknown.

We show that at the beginning of step 2 in fact all $e \in \mathring{D}_K(X_i \cup Y_i)$ with $|e \cap X_i| = 2$ are unknown by induction on i :

$i = 2$:

We have already stated that when the algorithm starts Stage II with $i = 2$, all defective edges $e \in \mathring{D}(X_i \cup X_j)$ with $|e \cap X_i| = 2$ are unknown. That are in particular all defective edges $e \in \mathring{D}(X_2 \cup X_1) = \mathring{D}(X_2 \cup Y_2)$ with $|e \cap X_2| = 2$.

$i \rightarrow i + 1$:

While the algorithm performs Stage II for some i , only subsets of

$$X_i \cup Y_i = \bigcup_{k=1}^i X_k$$

are tested, that is, the algorithm does not find any defective edges in $D(X_{i+1})$. Therefore, all defective edges with $e \in \mathring{D}(\bigcup_{k=1}^{i+1} X_k)$ and $|e \cap X_{i+1}| = 2$ remain unknown. ■

Since the sets $C_1^i, C_2^i, \dots, C_{p_i}^i$ are pairwise disjoint, every edge $e \in \mathring{D}(X_i \cup Y_i)$ with $|X_i \cap e| = 2$ lies in exactly one set $\mathring{D}(X_i \cup U_j^i)$, and due to Lemma 3.28, the algorithm finds all defective edges

$$e \in \mathring{D}(X_i \cup C_j^i) \text{ with } |e \cap X_i| = 2,$$

($i = 2, \dots, s$, $j = 1, \dots, r_i$). These are all defective edges $e \in \mathring{D}(X_i \cup X_j)$ with $|e \cap X_i| = 2$, $1 \leq j < i \leq s$, and thus after Stage II, all defective edges

$$\mathring{D}(X_i \cup X_j)$$

are known ($1 \leq i < j \leq s$).

Please note that there has been no need to split Y_i into $C_1^i, C_2^i, \dots, C_{p_i}^i$ in order to find all defective edges $e \in \mathring{D}(X_i \cup Y_i)$ with $|e \cap X_i| = 2$. But since we need one test for each known defective edge in $\mathring{D}(X_i \cup Y_i)$, splitting Y_i beforehand saves tests.

Stage III

In Stage III our search algorithm finds all unknown defective edges that lie in the union of three free sets, which are the only remaining unknown defective edges. Again, we start with a lemma.

Lemma 2.10. *Let $X, Y, Z \subset V$ be three pairwise disjoint vertex sets such that*

- $\mathring{D}(X \cup Y \cup Z)$ contains no loops,

- X is free,
- all defective edges in $\mathring{D}(X \cup Y)$ are known,
- all defective edges in $\mathring{D}(X \cup Y \cup Z)$ with cardinality 2 are known, and
- every known defective edge $e \in \mathring{D}(X \cup Y \cup Z)$ satisfies either $|e \cap (Y \cup Z)| = 1$ or $|e \cap X| = |e \cap Y| = |e \cap Z| = 1$ (cf. Figure 2.5).

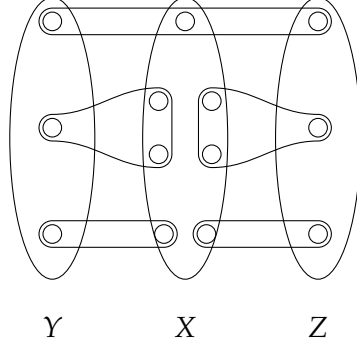


Figure 2.5: Sketch of admissible known defective edges.

Then there is a search algorithm on $X \cup Y \cup Z$ that finds at least all defective edges in $D(X, Y, Z)$ by at most

$$d_{\Delta}(\lceil \log_2 |E| \rceil + 5) + \mathring{d}_K(X \cup Y \cup Z) + d_K(X, Y, Z) + 1$$

tests, where d_{Δ} denotes the number of defective edges which the algorithm actually finds.

Proof. Set $X' := X$.

1. If $\mathring{D}_K(X' \cup Y \cup Z) = \emptyset$, go to step 5.
Otherwise, choose a vertex $x \in X'$ with $x \in e$ for some $e \in \mathring{D}_K(X' \cup Y \cup Z)$, set $X' := X' \setminus \{x\}$, and $Y^x := Y$.
2. If $\mathring{D}_K(\{x\} \cup Y^x \cup Z) \setminus \Gamma_K(\{x\}) = \emptyset$, go to step 4.
Otherwise, choose a vertex $y \in Y^x$ with $y \in g$ for some $g \in \mathring{D}_K(\{x\} \cup Y^x \cup Z) \setminus \Gamma_K(\{x\})$, set $Y^x := Y^x \setminus \{y\}$, and go to step 3.

3. Test

$$\mathcal{T} := \{x, y\} \cup (Z \setminus \Gamma_K(\{x, y\}))$$

and find in case of $f(\mathcal{T}) = 1$ all defective edges $e \in \mathring{D}(\mathcal{T})$ with $\{x, y\} \subset e$ by Lemma 2.8. Repeat step 2.

4. Test

$$\mathcal{T} := [\{x\} \cup Y^x \cup Z] \setminus \Gamma_K(\{x\})$$

and find in case of $f(\mathcal{T}) = 1$ all defective edges $e \in \mathring{D}(\mathcal{T})$ with $|e \cap (\{x\} \cup Y^x)| = 2$ by Lemma 2.8. Repeat step 1.

5. Test

$$\mathcal{T} := X' \cup Y \cup Z$$

and find in case of $f(\mathcal{T}) = 1$ all defective edges $e \in \mathring{D}(\mathcal{T})$ with $|e \cap (X' \cup Y)| = 2$ by Lemma 2.8.

Every edge $e \in \mathring{D}_K(X \cup Y \cup Z)$ satisfies either $|e \cap (Y \cup Z)| = 1$ or $|e \cap X| = |e \cap Y| = |e \cap Z| = 1$. Moreover, there are no loops in $\mathring{D}(X \cup Y \cup Z)$. Consequently, every edge $e \in \mathring{D}_K(X \cup Y \cup Z)$ is incident to some vertex $x \in X$. Hence, as long as $\mathring{D}_K(X' \cup Y \cup Z) \neq \emptyset$, there is some vertex $v \in X'$ which is incident to some known defective edge in $\mathring{D}_K(X' \cup Y \cup Z)$.

Let now $x \in X$ and $Y^x \subseteq Y$ such that there is some edge $g \in \mathring{D}_K([\{x\} \cup Y^x \cup Z] \setminus \Gamma_K(\{x\}))$. Since $g \cap \Gamma_K(\{x\}) = \emptyset$ and $x \in g$, we have $|g \setminus \{x\}| \geq 2$, which implies that $|g \cap Y^x| = |g \cap Z| = 1$. Thus, for every edge $g \in \mathring{D}_K([\{x\} \cup Y^x \cup Z] \setminus \Gamma_K(\{x\}))$ there is some vertex $y \in Y^x$ which is incident to g . Please note $y \notin \Gamma_K(\{x\})$ and thus $\{x, y\} \notin D$.

The constraints of Lemma 2.8 are satisfied:

In step 3:

We have already seen that $\{x, y\} \notin D$. As there are no loops in $\mathring{D}_K(X \cup Y \cup Z)$, the set $\{x, y\}$ is free. Since $|e \cap Z| \leq 1$ for all $e \in \mathring{D}_K(X \cup Y \cup Z)$, the test set

$$\mathcal{T} = \{x, y\} \cup (Z \setminus \Gamma_K(\{x, y\}))$$

is D_K -independent. That means that in step 3 the constraints of Lemma 2.8 are satisfied for $f(\mathcal{T}) = 1$.

In step 4:

The algorithm removes vertices y from Y^x until $\mathring{D}_K([\{x\} \cup Y^x \cup Z] \setminus \Gamma_K(\{x\})) = \emptyset$, that is, until $[\{x\} \cup Y^x \cup Z] \setminus \Gamma_K(\{x\})$ is D_K -independent. Since all defective edges in $\mathring{D}(X \cup Y)$ are known, $(\{x\} \cup Y^x) \setminus \Gamma_K(\{x\})$ is free. If $f(\mathcal{T}) = 1$, the constraints of Lemma 2.8 are satisfied for

$$\mathcal{T} := [\{x\} \cup Y^x \cup Z] \setminus \Gamma_K(\{x\}).$$

In step 5:

The same holds for step 5: The algorithm removes vertices x from X' until $X' \cup Y \cup Z$ is D_K -independent. Since all defective edges in $\mathring{D}(X \cup Y)$ are known, $X' \cup Y$ is free. If $f(\mathcal{T}) = 1$, the constraints of Lemma 2.8 are satisfied for

$$\mathcal{T} := X' \cup Y \cup Z.$$

The algorithm finds all defective edges in $D(X, Y, Z)$:

Due to Lemma 2.8, the algorithm finds all defective edges

- $e \in \mathring{D}(\{x, y\} \cup [Z \setminus \Gamma_K(\{x, y\})])$ with $\{x, y\} \subset e$ for each $x \in X \setminus X'$ and $y \in Y \setminus Y^x$ in step 3,
- $e \in \mathring{D}([\{x\} \cup Y^x \cup Z] \setminus \Gamma_K(\{x\}))$ with $|e \cap (\{x\} \cup Y^x)| = 2$ for each $x \in X \setminus X'$ in step 4, and
- $e \in \mathring{D}(X' \cup Y \cup Z)$ with $|e \cap (X' \cup Y)| = 2$ in step 5.

Let now e be some unknown defective edge in $D(X, Y, Z)$. If $e \in D(X', Y, Z)$, then the algorithm finds e in step 5 at the latest. Otherwise, there is some vertex $x \in X \setminus X'$ with $x \in e$ and there are again two possibilities:

Either $e \in D(\{x\}, Y^x, Z \setminus \Gamma_K(\{x\}))$, then the algorithm finds e in step 4 at the latest. Else, there is some vertex $y \in Y \setminus Y^x$ with $y \in e$ and the algorithm finds e in $D(\{x\}, \{y\}, Z \setminus \Gamma_K(\{x, y\}))$ in step 3.

The number of tests:

The algorithm tests the following sets:

- $\{x, y\} \cup [Z \setminus \Gamma_K(\{x, y\})]$ for each $x \in X \setminus X'$ and $y \in Y \setminus Y^x$,
- $[\{x\} \cup Y^x \cup Z] \setminus \Gamma_K(\{x\})$ for each $x \in X \setminus X'$, and
- $X' \cup Y \cup Z$.

If a test is positive, the algorithm uses Lemma 2.8 to find defective edges, which costs per defective edge at most

$$\lceil \log_2 |E| \rceil + 5$$

tests. Now, for every $x \in X \setminus X'$ there is at least one known defective edge $e \in \mathring{D}_K(X \cup Y \cup Z)$, and for every vertex $y \in Y \setminus Y^x$ for some $x \in X$ there is at least one known defective edge $e \in D_K(X, Y, Z)$ with $\{x, y\} \subseteq e$. Hence, there are at most

$$d_\Delta \cdot (\lceil \log_2 |E| \rceil + 5) + \overbrace{d_K^{\mathring{}}(X \cup Y \cup Z)}^{\text{step 3}} + \overbrace{d_K(X, Y, Z)}^{\text{step 4}} + \overbrace{1}^{\text{step 5}}$$

tests. □

Algorithm - Stage III

Set $i = 2$.

1. Set

$$Y_i := \bigcup_{j=1}^{i-1} X_j \text{ and } Z_i := \bigcup_{j=i+1}^s X_j.$$

For X_i, Y_i , and Z_i , construct r_i D_K -independent sets $U_1^i, U_2^i, \dots, U_{r_i}^i$ according to Lemma 2.6 such that

- $r_i \leq 6 \cdot d_K$,
- every pair $\{y, z\}$ with $y \in Y_i, z \in Z_i$ and $\{y, z\} \notin D_K$ is subset of some U_j^i ,
- every vertex $v \in Y_i \cup Z_i$ lies in at most $6\sqrt{d_K}$ different sets U_j^i , and
- $|e \cap U_j^i| = 1$ or $|e \cap X_i| = |e \cap U_j^i \cap Y_i| = |e \cap U_j^i \cap Z_i| = 1$ for every $e \in \mathring{D}_K(X_i \cup U_j^i)$, ($j = 1, \dots, r_i$) (cf. Figure 2.6).

Set $U_j^{Y_i} := U_j^i \cap Y_i$ and $U_j^{Z_i} := U_j^i \cap Z_i$ for $1 \leq j \leq r_i$ and $j := 1$.

2. Find all defective edges $e \in D(X_i, U_j^{Y_i}, U_j^{Z_i})$ (cf. Lemma 3.36). If
 $j < r_i$: set $j := j + 1$ and repeat step 2.
 $j = r_i$: $i < s - 1$: set $i := i + 1$ and repeat step 1.
 $i = s - 1$: stop.

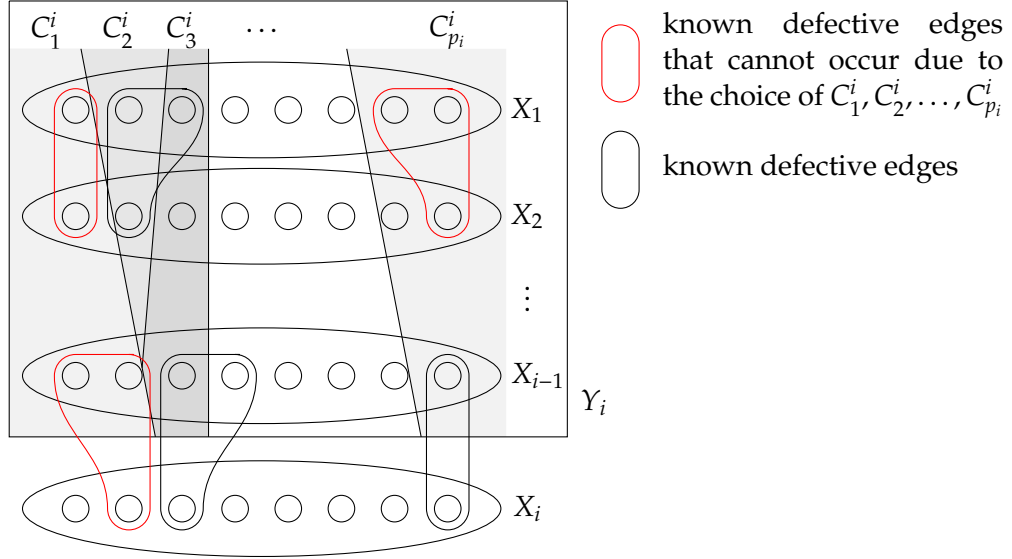


Figure 2.6: Sketch of Stage II

In step 1 the algorithm constructs, according to Lemma 2.6, sets $U_1^i, U_2^i, \dots, U_{r_i}^i$ such that

$$(*) : |e \cap U_j^i| = 1 \text{ or } |e \cap X_i| = |e \cap U_j^{Y_i}| = |e \cap U_j^{Z_i}| = 1$$

for every $e \in \mathring{D}_K(X_i \cup U_j^i)$, ($i = 2, \dots, s - 1$, $j = 1, \dots, r_i$). That is, one of the constraints of Lemma 3.36 is already satisfied, some others are also clear:

We have already mentioned that H contains no loops, X_i ($i = 1, \dots, s$) is free and all edges of cardinality 2 are known. Thus, it remains to show that all defective edges in

$$\mathring{D}_K(X_i \cup U_j^i)$$

are known. We show by induction on i that, as soon as the algorithm enters step 2 for i , $\mathring{D}_K(X_i \cup Y_i) = \mathring{D}(X_i \cup Y_i)$.

$i = 2$:

For $i = 2$ we have $Y_i = Y_2 = X_1$. After Stage II, all defective edges in $\mathring{D}(X_i \cup X_j)$ are known, these are certainly all edges in

$$\mathring{D}(X_1 \cup X_2) = \mathring{D}(Y_2 \cup X_2).$$

$i \rightarrow i + 1$:

Assume that all defective edges in $\mathring{D}(X_i \cup Y_i)$ are known. In that case all constraints of Lemma 3.36 are fulfilled, and the algorithm finds all defective edges in

$$\mathring{D}(X_i, U_j^{Y_i}, U_j^{Z_i}),$$

($j = 1, \dots, r_i$). Let now $e \in D(X_i, Y_i, Z_i)$ be some unknown defective edge. The sets $U_1^i, U_2^i, \dots, U_{r_i}^i$ cover all pairs $\{y, z\}$ with $y \in Y_i, z \in Z_i$ and $\{y, z\} \notin D_K$. Hence there is some set

$$U_j^i \text{ with } e \setminus X_i \subseteq U_j^i.$$

and thus

$$|e \cap X_i| = |e \cap Y_i \cap U_j^i| = |e \cap Z_i \cap U_j^i| = 1.$$

That is, for every unknown defective edge $e \in D(X_i, Y_i, Z_i)$ there is some set U_j^i such that $e \in D(X_i, Y_i \cap U_j^i, Z_i \cap U_j^i)$. Due to Lemma 3.36, the algorithm finds all defective edges in $D(X_i, Y_i \cap U_j^i, Z_i \cap U_j^i)$. Since

$$D(X_i, Y_i, X_{i+1}) \subseteq D(X_i, Y_i, Z_i),$$

all defective edges in

$$\mathring{D}(X_i \cup Y_i \cup X_{i+1}) = \mathring{D}(Y_{i+1} \cup X_{i+1})$$

are known before the algorithm sets $i := i + 1$. ■

As a consequence, the algorithm finds all unknown defective edges in $D(Y_i, X_i, Z_i)$, ($i = 2, \dots, s - 1$) at Stage III. Since these were the only defective edges that were (probably) unknown after stage II, the algorithm eventually finds all defective edges in a hypergraph of rank 3.

2.1.2 An upper bound for the number of tests

Let us count the number of tests individually for each stage.

Stage I

Consider the steps of Stage I for some fixed i : In step 1 the algorithm tests X_i and finds, in case of $f(X_i) = 1$, a defective edge $e \in \mathring{D}(X_i)$ which costs, due to Theorem 1.2, at most

$$\lceil \log_2 |E| \rceil + 2$$

tests. After removing some $x \in e$ from X_i and adding it to X_{i+1} the algorithm repeats step 1 until eventually $f(X_i) = 0$.

Then the algorithm goes on to step 2. We have already stated that, right after step 1, $\mathring{d}_K(\{x\} \cup X_i) = 1$ for all $x \in X_{i+1}$. If $\mathring{d}(\{x\} \cup X_i) = \mathring{d}_K(\{x\} \cup X_i) = 1$, the algorithm needs, according to Lemma 2.5, at most 2 tests for proof. Otherwise, if $\mathring{d}(\{x\} \cup X_i) > 1$, the algorithm finds all defective edges in $\mathring{D}(\{x\} \cup X_i)$ which costs, according to Lemma 2.5, at most

$$\lceil \log_2 |E| \rceil + 4$$

tests per defective edge. That are at most

$$\begin{aligned} & \sum_{\substack{x \in X_{i+1} : \\ \mathring{d}(\{x\} \cup X_i) = 1}} \left[1 + \lceil \log_2 |E| \rceil + 2 + 2 \right] \\ + & \sum_{\substack{x \in X_{i+1} : \\ \mathring{d}(\{x\} \cup X_i) > 1}} \left[1 + \lceil \log_2 |E| \rceil + 2 + (\mathring{d}(\{x\} \cup X_i) - 1) \cdot (\lceil \log_2 |E| \rceil + 4) \right] + 1 \end{aligned}$$

$$\leq \sum_{x \in X_{i+1}} \mathring{d}(\{x\} \cup X_i) \cdot (\lceil \log_2 |E| \rceil + 5) + 1$$

tests while the algorithm performs Stage I for i . Let d_I denote the number of defective edges that the algorithm finds in step 1; then there are at most

$$d_I \cdot (\lceil \log_2 |E| \rceil + 5) + s$$

tests in stage I, where s denotes the number of sets X_i .

For every vertex that is removed from X_i and added to X_{i+1} , the algorithm has found one defective edge. On the other hand, every vertex in X_i has been removed $i - 1$ times, namely from the set X_1, X_2, \dots, X_{i-1} . Since there are no empty sets X_i ,

$$d \geq \sum_{i=2}^s |X_i| \cdot (i-1) = \sum_{i=2}^s (i-1) + \underbrace{\sum_{i=2}^s (|X_i| - 1) \cdot (i-1)}_{\geq 0} \geq \binom{s}{2}$$

and thus

$$s \leq 1/2 + \sqrt{2d + 1/4} \stackrel{d \geq 1}{\leq} 2\sqrt{d}.$$

Stage II

The algorithm performs Stage II $s - 1$ times with increasing i starting at $i = 2$. In step 1 the algorithm splits the vertex set $Y_i = \bigcup_{j=1}^{i-1} X_j$ without any tests into $p_i \leq 2 \cdot \sqrt{d_K} \leq 2 \cdot \sqrt{d}$ disjoint sets such that

- $p_i \leq 2 \cdot \sqrt{d_K} \leq 2 \cdot \sqrt{d}$ and
- $|e \cap C_j^i| = 1$ for all $e \in \mathring{D}_K(X_i \cup C_j^i)$, ($j = 1, \dots, p_i$).

All sets $C_1^i, C_2^i, \dots, C_{p_i}^i$ are disjoint, accordingly every edge $e \in D(X_i, Y_i)$ lies in at most one set $\mathring{D}(X_i \cup C_j^i)$. Moreover, for each $e \in D$ there is exactly one i with $e \in D(X_i, Y_i)$ and thus

$$\sum_{i=2}^s \sum_{j=1}^{p_i} \mathring{d}(X_i \cup C_j^i) \leq \sum_{i=2}^s d(X_i, Y_i) = d.$$

In step 2 all defective edges in $e \in \mathring{D}(X_i \cup C_j^i)$ with $|e \cap X_i| = 2$ are found, which costs, according to Lemma 3.28, at most

$$d_\Delta(\lceil \log_2 |E| \rceil + 5) + \mathring{d}_K(X_i \cup C_j^i) + 1$$

tests, where d_Δ denotes the number of defective edges that are actually found in $\mathring{D}(X_i \cup C_j^i)$ ($i = 2, \dots, s$, $j = 1, \dots, p_i$). Let d_{II} denote the number of defective edges that the algorithm finds in Stage II; then at most

$$d_{II} \cdot (\lceil \log_2 |E| \rceil + 5) + \sum_{i=2}^s \sum_{j=1}^{p_i} (\mathring{d}_K(X_i \cup C_j^i) + 1)$$

$$\begin{aligned}
&\leq d_{II} \cdot (\lceil \log_2 |E| \rceil + 5) + \sum_{i=2}^s (d_K(X_i, Y_i) + p_i) \\
&\leq d_{II} \cdot (\lceil \log_2 |E| \rceil + 5) + d + \sum_{i=2}^s 2 \cdot \sqrt{d} \\
&\leq d_{II} \cdot (\lceil \log_2 |E| \rceil + 5) + 5 \cdot d
\end{aligned}$$

tests are needed at Stage II.

Stage III

The algorithm performs Stage III $s - 2$ times with increasing i starting at $i = 2$. In step 1 the algorithm constructs sets $U_1^i, U_2^i, \dots, U_{r_i}^i$ such that

- $r_i \leq 6 \cdot d_K \leq 6 \cdot d$,
- every pair $\{y, z\}$ with $y \in Y_i = \bigcup_{j=1}^{i-1} X_j$, $z \in Z_i = \bigcup_{j=i+1}^s X_j$ and $\{y, z\} \notin D_K$ is subset of some U_j^i ,
- every vertex $v \in Y_i \cup Z_i$ lies in at most $6 \cdot \sqrt{d_K} \leq 6 \cdot \sqrt{d}$ different sets U_j^i , and
- $|e \cap U_j^i| = 1$ or $|e \cap X_i| = |e \cap U_j^i \cap Y_i| = |e \cap U_j^i \cap Z_i| = 1$ for every $e \in \mathring{D}_K(X_i \cup U_j^i)$, ($j = 1, \dots, r_i$).

Afterwards, in step 2, the algorithm finds all defective edges in $D(X_i, U_j^{Y_i}, U_j^{Z_i})$ by Lemma 3.36 by at most

$$d_\Delta(\lceil \log_2 |E| \rceil + 5) + \mathring{d}_K(X_i \cup U_j^i) + d_K(X_i, U_j^{Y_i}, U_j^{Z_i}) + 1$$

tests, where d_Δ denotes the actual number of defective edges that are found in $\mathring{D}(X_i \cup U_j^i)$. Thus, already known defective edges cause extra tests if they lie in some set $X_i \cup U_j^i$. Now, let $e \in D_K$ be a known defective edge. In how many sets $X_i \cup U_j^i$ could e possibly lie? We distinguish between two cases:

1) $e \in \mathring{D}_K(X_i \cup X_k)$ for some $i \neq k$:

Without loss of generality, let $i < k$, then $e \in \mathring{D}_K(X_i \cup Z_i)$ and $e \in \mathring{D}_K(X_k \cup Y_k)$. If $|e \cap X_k| = 1$, then there are sets

$$U_j^i \text{ with } e \in \mathring{D}_K(X_i \cup U_j^i).$$

Since every vertex lies in at most $6 \cdot \sqrt{d}$ different sets, there are at most $6 \cdot \sqrt{d}$ different sets

$$\mathring{D}_K(X_i \cup U_j^i)$$

that contain e . The same holds if $|e \cap X_i| = 1$; then there are at most $6 \cdot \sqrt{d}$ sets

$$\mathring{D}_K(X_k \cup U_j^k)$$

containing e . That is, if $e \in D_K(X_i \cup X_k)$ for some $i \neq k$, there are at most

$$12 \cdot \sqrt{d}$$

different test sets in Stage III that contain e .

2)e $\in D_K(X_i, X_k, X_l)$ for some $i < k < l$:

In that case $e \in \mathring{D}_K(X_i \cup Z_i)$, $e \in D_K(X_k, Y_k, Z_k)$, and $e \in \mathring{D}_K(X_l \cup Y_l)$. Since $|e \cap Z_i| = |e \cap Y_l| = 2$, there are neither sets

$$U_j^i \text{ with } (e \setminus X_i) \subseteq U_j^i$$

nor sets

$$U_j^l \text{ with } (e \setminus X_l) \subseteq U_j^l.$$

So e lies at most in

$$6 \cdot \sqrt{d}$$

sets $D_K(X_k \cup U_j^k)$, if $e \in D_K(X_i, X_k, X_l)$ for some $i < k < l$. Please note that e is counted twice by

$$\mathring{d}_K(X_i \cup U_j^i) + d_K(X_i, U_j^{Y_i}, U_j^{Z_i})$$

if $e \in \mathring{D}_K(X_i \cup U_j^i)$.

Let d_{III} denote the number of defective edges that the algorithm finds in Stage III. Then at most

$$\begin{aligned} & d_{III} \cdot (\lceil \log_2 |E| \rceil + 5) + \sum_{i=2}^{s-1} \sum_{j=1}^{r_i} (\mathring{d}_K(X_i \cup U_j^i) + d_K(X_i, U_j^{Y_i}, U_j^{Z_i}) + 1) \\ & \leq d_{III} \cdot (\lceil \log_2 |E| \rceil + 5) + 12 \cdot \sqrt{d} \cdot d + \sum_{i=2}^{s-1} r_i \\ & \leq d_{III} \cdot (\lceil \log_2 |E| \rceil + 5) + 12 \cdot d^{\frac{3}{2}} + (2 \cdot \sqrt{d} - 2) \cdot 6 \cdot d \\ & = d_{III} \cdot (\lceil \log_2 |E| \rceil + 5) + 24 \cdot d^{\frac{3}{2}} - 12 \cdot d \end{aligned}$$

tests are needed.

Overall, we obtain:

Theorem 2.11. *Let H be a hypergraph of rank 3 with defective edge set D , for which the cardinality $d := |D|$ is not necessarily known. Then the complexity $c(H, d)$ of finding d defective edges is bounded by*

$$c(H, d) \leq d \lceil \log_2 |E| \rceil + 2 \cdot \sqrt{d} - 2 \cdot d + 24 \cdot d^{\frac{3}{2}}.$$

2.2 Hypergraphs of bounded rank r

In this chapter, we consider hypergraphs of bounded rank r . We present an algorithm \mathcal{A}_r that finds all defective edges in a hypergraph of rank r regardless whether the number of defective edges is known. The algorithm \mathcal{A}_r starts again with the graph algorithm (s. [KT08]) and partitions the vertex set of a given hypergraph into free sets. But then, unlike before, it is not reasonable to consider each possible distribution of the vertices of an edge over the free sets in single stages. The number of possible distributions increases exponentially, hence, we would need a correspondingly high number of stages.

The following algorithm organizes the search differently. After partitioning a given vertex set

into free sets, it finds at least all those defective edges that share pairwise not more than one vertex. The remaining defective edges are finally detected in the third and last stage. Let us start with some new terminologies:

Definition 2.12. Let $H = (V, E)$ be a hypergraph with defective edge set D and $D_K \subset D$ the set of known defective edges. We say a vertex set $X \subset V$ is strongly D_K -independent if $|e \cap X| \leq 1$ for every $e \in D_K$. Next, define for some $W \subseteq V$

$$\mathcal{P}_K^t(W) := \{X \subset W : |X| = t \text{ and } |e \cap X| \leq 1 \text{ for every } e \in D_K\},$$

the set of strongly D_K -independent vertex sets of cardinality t , set $\mathcal{P}_K^t := \mathcal{P}_K^t(V)$. We call a vertex $v \in V \setminus X$ a D_K -neighbour of a set $X \subset V$ if there is an edge $e \in D_K$ that joins v to some vertex in X . Finally, let $N_K(X)$ denote the set of D_K -neighbours:

$$N_K(X) = \{v \in V \setminus X : \exists e \in D_K \text{ with } v \in e, e \cap X \neq \emptyset\}.$$

In the following, if not stated otherwise, let $H = H(V, E)$ be a simple hypergraph of bounded rank r with defective edge set $D \subset E$, where the number of defective edges $d := |D|$ is not necessarily known.

2.2.1 A search algorithm on hypergraphs of bounded rank r

We will again consider all stages of the algorithm \mathcal{A}_r , separately.

Stage I

The first stage is almost the same as in the previous section, except, we will omit the second step.

\mathcal{A}_r - Stage I

Let V be alphabetically ordered.

Set $X_1 := V$, $X_j := \emptyset$ for $2 \leq j \leq |E|$ and $i := 1$.

1. Test X_i if

$f(X_i) = 1$: Use Triesch's search to find a defective edge $e \in \mathring{D}(X_i)$ with rightmost left endvertex and delete all good edges. Let $x \in e$ be the left endvertex of e , set X_i to $X_i \setminus \{x\}$ and re-sort X_i due to Lemma 2.3. If $|e| \geq 2$, set X_{i+1} to $X_{i+1} \cup \{x\}$.

Repeat step 1.

$f(X_i) = 0$: If $X_{i+1} = \emptyset$, set $s := i$ and go to Stage II, else, go to step 2.

2. Set $i := i + 1$, sort X_i alphabetically, and go back to step 1.

Please note that the rank of the considered hypergraph is not an issue for the proof of Lemma 2.3, hence the Lemma is also true for hypergraphs of any bounded rank r and thus we can use the Lemma on hypergraphs of any bounded rank.

The algorithm runs Stage I exactly s times. With the same argumentation as in the previous section, we receive

$$s \leq 2 \cdot \sqrt{d_K} \leq 2 \cdot \sqrt{d}, \quad (2.1)$$

and the resulting s free sets X_1, X_2, \dots, X_s form a partition of $\{v \in V : \{v\} \notin D\}$. Since all defective loops are certainly detected after step 1, we assume once more that H contains no loops at all.

Let d_I denote the number of defective edges that the algorithm identifies in Stage I, then, according to Theorem 1.2, at most

$$s + d_I \cdot (1 + \lceil \log_2 |E| \rceil + r - 1) \leq 2 \cdot \sqrt{d} + d_I \cdot (\lceil \log_2 |E| \rceil + r)$$

tests have to be carried out in Stage I.

Stage II

The aim of the second stage is to find defective edges, such that after Stage II for every defective edge $e \in D$ that is still unknown, there is some defective edge $g \in D$ with $|e \cap g| \geq 2$ which is known. In that case every strongly D_K -independent set is also free.

The rough idea of the second stage is now the following: The algorithm constructs in Stage II stepwise free sets $F_1^t, F_2^t, \dots, F_{p_t}^t$ such that every $X \in \mathcal{P}_K^t$ lies in some set F_i^t for $t = 1, \dots, r$. If some $X \in \mathcal{P}_K^t$ lies in a free set, certainly X is also free, and if all sets in $X \in \mathcal{P}_K^r$ are free, then every strongly D_K -independent set is free.

\mathcal{A}_r - Stage II

1. Set $C_n := \emptyset$ for all $1 \leq n \leq 2 \cdot d_K$ and $i, j, k := 1$.
2. If there is a vertex $x \in X_k$ with $|\mathcal{N}_K(\{x\})| \geq r/2 \cdot \sqrt{d_K}$, set $B_i := \{x\}$, $i := i + 1$, $X_k := X_k \setminus \{x\}$ and repeat step 2; else, go to step 3.
3. If there is a vertex $x \in X_k$ with $|\mathcal{N}_K(\{x\} \cup C_j)| \leq r \cdot \sqrt{d_K}$, set $C_j := C_j \cup \{x\}$, $X_k := X_k \setminus \{x\}$ and repeat step 3, else, if:
 - $X_k \neq \emptyset$: Set $j := j + 1$ and repeat step 3.
 - $X_k = \emptyset$: $k < s$: Set $j := j + 1$, $k := k + 1$, and repeat step 2.
 - $k = s$: Set $p := i$, $q := j$.
4. Set $F_i^1 := B_i$ for $i = 1, \dots, p$, $F_{p+j}^1 := C_j$ for $j = 1, \dots, q$, and $p_1 := p + q$, further set $B := \bigcup_{i=1}^p B_i$ and $i, j, k, n, t := 1$.
5. Test $\mathcal{T} := B_i \cup (F_k^t \setminus \mathcal{N}_K(B_i))$. If
 - $f(\mathcal{T}) = 1$: Find a defective edge in $\mathring{D}(\mathcal{T})$ by Triesch's search and repeat step 5. Please note that $\mathcal{N}_K(B_i)$ has increased and hence, we do not repeat the same test.
 - $f(\mathcal{T}) = 0$: Set $F_n^{t+1} := \mathcal{T}$ and $n := n + 1$. If
 - $i < p$: Set $i := i + 1$ and repeat step 5.
 - $i = p$: Go to step 6.
6. Test $\mathcal{T} := (F_k^t \setminus B) \cup (C_j \setminus \mathcal{N}_K(F_k^t \setminus B))$. If
 - $f(\mathcal{T}) = 1$: Find a defective edge in $\mathring{D}(\mathcal{T})$ by Triesch's search and repeat step 6.
 - $f(\mathcal{T}) = 0$: Set $F_n^{t+1} := \mathcal{T}$ and $n := n + 1$. If
 - $j < q$: Set $j := j + 1$ and repeat step 6.
 - $j = q$: $t < r - 1$: Go to step 5.
 - $t = r - 1$: Set $X := \mathcal{N}_K(F_k^t \setminus B) \setminus B$, choose a vertex $x \in X$, and go to step 7.

- If $|\mathcal{N}_K(C_j)| > 2r\sqrt{d_K}$ for at least one $1 \leq j \leq q$, repeat step 1. Otherwise, set $X := \mathcal{N}_K(F_k^t \setminus B) \setminus B$ and choose a vertex $x \in X$.
7. Test $\mathcal{T} := \{x\} \cup (F_k^t \setminus \mathcal{N}_K(\{x\}))$. If
 $f(\mathcal{T}) = 1$: Find a defective edge in $\dot{D}(\mathcal{T})$ by Triesch's search and repeat step 7.
 $f(\mathcal{T}) = 0$: Set $F_n^{t+1} := \mathcal{T}$, $X := X \setminus \{x\}$, and $n := n + 1$. If
 $X \neq \emptyset$: Choose a vertex $x \in X$ and repeat step 7.
 $X = \emptyset$: $k < p_t$: Set $k := k + 1$ and $i, j := 1$, then repeat step 5.
 $k = p_t$: $t < r - 1$: Set $t := t + 1$, $p_t := n$, and $i, j, k, n := 1$, then repeat step 5.
 $t = r - 1$: Set $p_r := n$ and stop.

The following picture shows a schema of the sets that are tested in Step 5, 6 and 7.

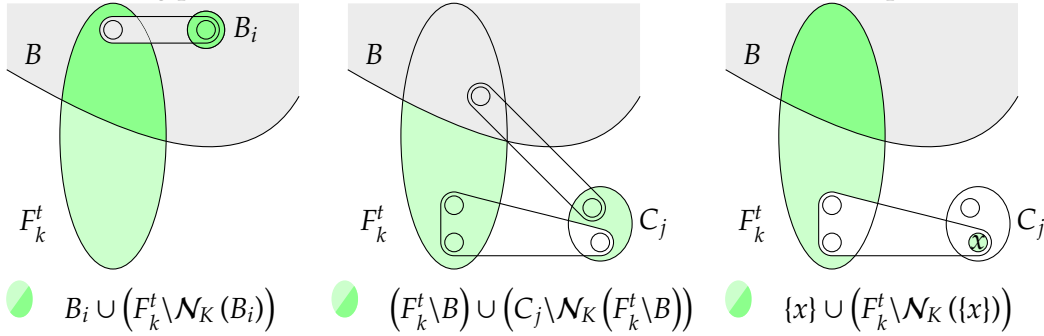


Figure 2.7: Sketch of the construction of test sets

Please note that between step 6 and step 7, there is a step in which we only check the number of neighbors and repeat if necessary, we therefore denote the step not by a number but by a symbol: □. Certainly all tests are D_K -independent and thus reasonable. Now, let us first of all take a look at step 7. In step 7 the algorithm constructs one set $\{x\} \cup (F_k^t \setminus \mathcal{N}_K(\{x\}))$ for each $x \in \mathcal{N}_K(F_k^t \setminus B) \setminus B$ ($t = 1, \dots, r - 1, k = 1, \dots, p_t$). To limit the number of sets, the algorithm limits the number of D_K -neighbors of the sets $F_1^t, F_2^t, \dots, F_{p_t}^t$ ($t = 1, \dots, r - 2$) (please note that for $t = r - 1$ the algorithm skips step □).

For that reason \mathcal{A}_r partitions in the first three steps all sets X_k with too many D_K -neighbors ($k = 1, \dots, s$). However, during the last three steps, we probably find defective edges that increase the number of D_K -neighbors again. Therefore, the algorithm checks at step □ whether the number of D_K -neighbors is still small enough or not. If the number is not small enough, that is, if there is at least one set C_j with $|\mathcal{N}_K(C_j)| > 2r\sqrt{d_K}$, \mathcal{A}_r repeats step 1 and reconstructs the sets $B_1, B_2, \dots, B_p, C_1, C_2, \dots, C_q$ with respect to the currently known defective edges.

Only in the last three steps, \mathcal{A}_r constructs the actual sets $F_1^t, F_2^t, \dots, F_{p_t}^t$ ($t = 2, \dots, r$). Now let us show that these sets $F_1^t, F_2^t, \dots, F_{p_t}^t$ cover indeed all sets in \mathcal{P}_K^t for $t = 1, \dots, r$:

All sets in \mathcal{P}_K^t are covered by some set F_k^t , ($t = 1, \dots, r$)

t = 1:

Every set in

$$\mathcal{P}_K^1 = \{\{v\} : v \in V \text{ and } |e \cap \{v\}| \leq 1 \text{ for every } e \in D_K\} = \{\{v\} : v \in V\}$$

is obviously covered by some set F_i^1 , since $\bigcup_{i=1}^{p_1} F_i^1 = V$.

$\mathbf{t} \rightarrow \mathbf{t} + 1$:

Next, let $t \leq r - 1$ and suppose that $F_1^t, F_2^t, \dots, F_{p_t}^t$ cover all sets in \mathcal{P}_K^t . Let $X \in \mathcal{P}_K^{t+1}$ be a strongly D_K -independent set with $X \cap B_i \neq \emptyset$ for some i , and F_k^t a set with $X \setminus B_i \subseteq F_k^t$. Since X is strongly D_K -independent, $(X \setminus B_i) \cap \mathcal{N}_K(B_i) = \emptyset$ and consequently

$$X \subseteq B_i \cup (F_k^t \setminus \mathcal{N}_K(B_i)).$$

Hence, in step 5 there is a test set $\mathcal{T} = B_i \cup (F_k^t \setminus \mathcal{N}_K(B_i))$ containing X . If $f(\mathcal{T}) = 0$, the algorithm sets $F_n^{t+1} := \mathcal{T}$ and we have $X \subseteq F_n^{t+1}$. Otherwise, if $f(\mathcal{T}) = 1$, there is at least one defective edge $e \in \mathring{D}(\mathcal{T})$ which is certainly incident to the only vertex in B_i because of F_k^t being free. Now, there are two possibilities: Either e is also incident to a further vertex in X , then X is no longer strongly D_K -independent. Or $e \cap X = B_i$, then again $X \subseteq B_i \cup (F_k^t \setminus \mathcal{N}_K(B_i))$ with respect to all currently known defective edges (e is now also a known edge). When eventually $f(\mathcal{T}) = 0$ (at the latest when $F_k^t \setminus \mathcal{N}_K(B_i) = \emptyset$), the algorithm sets $F_n^{t+1} := \mathcal{T}$ for some n and either $X \subseteq F_n^{t+1}$ or $X \notin \mathcal{P}_K^{t+1}$ (cf. Figure 2.8).

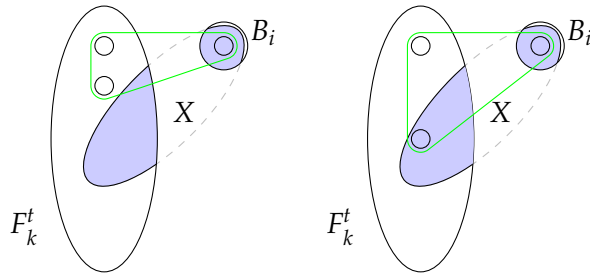


Figure 2.8: Two possibilities if $f(\mathcal{T}) = 1$

Let again $X \in \mathcal{P}_K^{t+1}$ be a strongly D_K -independent set, this time with $X \cap B = \emptyset$. Further, let F_k^t be a set with $X \setminus \{x\} \subseteq F_k^t$ for some $x \in X$, and C_j the set with $x \in C_j$. Then, we have either $x \notin \mathcal{N}_K(F_k^t \setminus B)$ or $x \in \mathcal{N}_K(F_k^t \setminus B)$. Therefore, in the first case,

$$X \subseteq (F_k^t \setminus B) \cup (C_j \setminus \mathcal{N}_K(F_k^t \setminus B))$$

and, in the second case,

$$X \subseteq \{x\} \cup (F_k^t \setminus \mathcal{N}_K(\{x\})).$$

The first set is tested in step 6, and the second set is tested in step 7. As before, after a positive test, there are two possibilities: Either the algorithm finds a defective edge such that X is no longer strongly D_K -independent, or X remains in the new test set, which is constructed with respect to the currently known defective edges set. So, when the algorithm defines F_n^{t+1} , we have either $X \subseteq F_n^{t+1}$ or $X \notin \mathcal{P}_K^{t+1}$.

Number of sets \mathbf{F}_k^t

Next, we show that

$$p_t \leq t! \cdot (2r)^t \cdot d_K^{\frac{t}{2}} \quad (2.2)$$

for $t \leq r - 1$.

$t = 1$:

Every (defective) edge has cardinality $\leq r$. Hence, every edge $e \in D_K$ joins each vertex $v \in e$ to at most $r - 1$ D_K -neighbors and thus, due to the Handshaking-Lemma,

$$\sum_{v \in V} |\mathcal{N}_K(\{v\})| \leq \sum_{v \in V} \sum_{e \in D_K(\{v\})} (|e| - 1) = \sum_{e \in D_K} |e| \cdot (|e| - 1) \leq d_K \cdot r(r - 1). \quad (2.3)$$

By the time the algorithm enters step 3,

$$|\mathcal{N}_K(\{x\})| < r/2 \cdot \sqrt{d_K}$$

for every $x \in X_k$ ($k = 1, \dots, s$). Further,

$$|\mathcal{N}_K(X \cup Y)| \leq |\mathcal{N}_K(X)| + |\mathcal{N}_K(Y)|$$

for any two sets $X, Y \subset V$. Hence, there are only two possible reasons for not enlarging C_j in step 3: Either $|\mathcal{N}_K(C_j)| > r/2 \cdot \sqrt{d_K}$ or $X_k = \emptyset$. Thus, for $k = 1, \dots, s$, there is at most one set $\emptyset \neq C_j \subseteq X_k$ with less than $r/2 \cdot \sqrt{d_K}$ D_K -neighbors, which in turn implies that we partition each set X_k into at most

$$\left\lceil \frac{|\mathcal{N}_K(X_k)|}{r/2 \cdot \sqrt{d_K}} \right\rceil$$

smaller sets. Altogether, we receive an upper bound for the number of sets:

$$p_1 = p + q \leq \sum_{k=1}^s \left\lceil \frac{|\mathcal{N}_K(X_k)|}{r/2 \cdot \sqrt{d_K}} \right\rceil \leq s + \frac{\sum_{k=1}^s |\mathcal{N}_K(X_k)|}{r/2 \cdot \sqrt{d_K}} \stackrel{(2.3)}{\leq} s + 2(r-1) \sqrt{d_K} \stackrel{(2.1)}{\leq} 2r \sqrt{d_K}. \quad (2.4)$$

$t \leq r - 1$:

Let us first of all take a look at step \square . Each time \mathcal{A}_r enters step \square for $t \leq r - 2$, it checks whether there is a set C_j with $|\mathcal{N}_K(C_j)| > 2r \sqrt{d_K}$. If at least one set C_j has too many D_K -neighbors, the algorithm repeats step 1 and constructs new sets $B_1, B_2, \dots, B_p, C_1, C_2, \dots, C_q$, such that again $|\mathcal{N}_K(C_j)| \leq r \sqrt{d_K}$ for $j = 1, \dots, q$. Consequently, every time the algorithm enters step 7 for $t \leq r - 2$, all sets C_j satisfy

$$|\mathcal{N}_K(C_j)| \leq 2r \sqrt{d_K}. \quad (2.5)$$

Every set F_n^{t+1} arises out of the union of two sets F_i^1 and F_k^t for some i and k , the second set itself is again, in case of $t > 1$, a subset of the union of two sets, etc. So, for every set F_k^t there is an index set $I \subset \{1, 2, \dots, p_1\}$, $|I| = t$ such that

$$F_k^t \subseteq \bigcup_{i \in I} F_i^1$$

and thus

$$\begin{aligned} |\mathcal{N}_K(F_k^t \setminus B) \setminus B| &\leq |\mathcal{N}_K(\bigcup_{i \in I} F_i^1 \setminus B)| \leq \sum_{i \in I} |\mathcal{N}_K(\underbrace{F_i^1 \setminus B}_{= \emptyset \text{ or } = C_j})| \leq t \cdot 2r \sqrt{d_K} \\ &= \emptyset \text{ or } = C_j \\ &\text{for some } j \end{aligned} \quad (2.6)$$

while $t \leq r - 2$. For each set F_k^t the algorithm constructs new sets by combining F_k^t with all p sets B_i (step 5), all q sets C_j (step 6), and, finally, with all vertices $x \in \mathcal{N}_K(F_k^t \setminus B) \setminus B$ (step 7). This amounts to:

$$\begin{aligned}
p_{t+1} &= \sum_{k=1}^{p_t} \left(\overbrace{p}^{\text{step 5}} + \overbrace{q}^{\text{step 6}} + \overbrace{|\mathcal{N}_K(F_k^t \setminus B) \setminus B|}^{\text{step 7}} \right) \\
&\stackrel{t < r-1}{\leq} p_t \cdot \left(2r \sqrt{d_K} + 2tr \sqrt{d_K} \right) \\
&\leq t! \cdot (2r)^t \cdot d_K^{\frac{t}{2}} \cdot \left((t+1) \cdot 2r \cdot \sqrt{d_K} \right) \\
&= (t+1)! \cdot (2r)^{t+1} \cdot d_K^{\frac{t+1}{2}}
\end{aligned} \tag{2.7}$$

t = r:

For $t = r - 1$ the algorithm no longer limits the number of D_K -neighbors of C_j ($j = 1, \dots, q$). Thus, we need a different approach to estimate the number p_r of sets $F_1^r, F_2^r, \dots, F_{p_r}^r$. For that reason we assess for each $v \in V$ the number of sets that contain v :

Definition 2.13. Set $\mathfrak{F}_t(v) := \{k : v \in F_k^t\}$ and $\mathfrak{f}_t(v) := |\mathfrak{F}_t(v)|$ for all $v \in V$.

We will show that for $1 \leq t \leq r - 1$, each vertex $v \in V$ lies in at most

$$\mathfrak{f}_t(v) \leq t! \cdot \left(\sum_{i=1}^t \frac{1}{i} \right) \cdot (2r)^{t-1} \cdot d_K^{\frac{t-1}{2}} \tag{2.8}$$

different sets F_k^t .

t = 1:

Since all sets $F_1^1, F_2^1, \dots, F_{p_1}^1$ are disjoint, $\mathfrak{f}_1(v) \leq 1 = (1! \sum_{i=1}^1 \frac{1}{i}) \cdot (2r)^0 \cdot d_K^0$ for all $v \in V$.

t ≤ r - 1:

Suppose (2.8) is true for $t - 1$; then: For every set F_k^{t-1} the algorithm constructs the following sets:

- $B_i \cup (F_k^{t-1} \setminus \mathcal{N}_K(B_i))$ for $i = 1, \dots, p$,
- $(C_j \setminus \mathcal{N}_K(F_k^{t-1} \setminus B)) \cup (F_k^{t-1} \setminus B)$ for $j = 1, \dots, q$ and
- $\{x\} \cup (F_k^{t-1} \setminus \mathcal{N}_K(\{x\}))$ for $x \in \mathcal{N}_K(F_k^{t-1} \setminus B) \setminus B$.

If $v \in F_k^{t-1}$, or, in other words, if $k \in \mathfrak{F}_{t-1}(v)$, then v lies in the worst case in each of the $p + q + |\mathcal{N}_K(F_k^{t-1} \setminus B) \setminus B|$ sets that the algorithm constructs for F_k^{t-1} . Otherwise, if $v \notin F_k^{t-1}$, only one of the following sets contains v since the sets

$$B_1, B_2, \dots, B_p, C_1 \setminus \mathcal{N}_K(F_k^{t-1} \setminus B), C_2 \setminus \mathcal{N}_K(F_k^{t-1} \setminus B), \dots, C_j \setminus \mathcal{N}_K(F_k^{t-1} \setminus B), \mathcal{N}_K(F_k^{t-1} \setminus B) \setminus B$$

are disjoint. Hence, for any $v \in V$ holds

$$\mathfrak{f}_t(v) \leq p_{t-1} + \sum_{k \in \mathfrak{F}_{t-1}(v)} (p + q + |\mathcal{N}_K(F_k^{t-1} \setminus B) \setminus B|).$$

According to (2.6), the number of neighbors is bounded by

$$|\mathcal{N}_K(F_k^{t-1} \setminus B) \setminus B| \leq 2r(t-1) \sqrt{d}$$

for each set F_k^t with $t \leq r-2$. Now, we can estimate the number of sets that contain some $v \in V$ for $t \leq r-1$:

$$\begin{aligned} \check{f}_t(v) &\leq p_{t-1} + \sum_{k \in \tilde{\mathcal{F}}_{t-1}(v)} (p+q + |\mathcal{N}_K(F_k^{t-1} \setminus B) \setminus B|) \\ &\stackrel{(2.2)}{\leq} (t-1)! \cdot (2r)^{t-1} \cdot d^{\frac{t-1}{2}} + \check{f}_{t-1}(v) \cdot (2r \sqrt{d} + 2r(t-1) \sqrt{d}) \\ &\leq (t-1)! \cdot (2r)^{t-1} \cdot d^{\frac{t-1}{2}} + \left((t-1)! \cdot \sum_{i=1}^{t-1} \frac{1}{i} \right) \cdot (2r)^{t-2} \cdot d^{\frac{t-2}{2}} \cdot 2rt \sqrt{d} \\ &\leq \frac{t!}{t} \cdot (2r)^{t-1} \cdot d^{\frac{t-1}{2}} + \left(t! \cdot \sum_{i=1}^{t-1} \frac{1}{i} \right) \cdot (2r)^{t-1} \cdot d^{\frac{t-1}{2}} \\ &= \left(t! \cdot \sum_{i=1}^t \frac{1}{i} \right) \cdot (2r)^{t-1} \cdot d^{\frac{t-1}{2}}, \end{aligned}$$

thus, (2.8) is also true for t .

Next, we consider p_r . For $r = t$ we have also

$$p_r = \sum_{k=1}^{p_{r-1}} (p+q + |\mathcal{N}_K(F_k^r \setminus B) \setminus B|).$$

Due to (2.8), these are at most

$$\begin{aligned} p_r &= \sum_{k=1}^{p_{r-1}} (p+q + |\mathcal{N}_K(F_k^{r-1} \setminus B) \setminus B|) \\ &\leq p_{r-1} \cdot (p+q) + \sum_{k=1}^{p_{r-1}} \sum_{v \in F_k^{r-1} \setminus B} |\mathcal{N}_K(\{v\})| \\ &\stackrel{(2.2)}{\leq} (r-1)! \cdot (2r)^{r-1} \cdot d^{\frac{r-1}{2}} \cdot 2r \sqrt{d} + \sum_{v \in V} \check{f}_{r-1}(v) \cdot |\mathcal{N}_K(\{v\})| \\ &\leq (r-1)! \cdot (2r)^r \cdot d^{\frac{r}{2}} + \left(\sum_{i=1}^{r-1} \frac{(r-1)!}{i} \right) \cdot (2r)^{r-2} \cdot d^{\frac{r-2}{2}} \cdot \sum_{v \in V} |\mathcal{N}_K(\{v\})| \\ &\stackrel{(2.3)}{\leq} (r-1)! \cdot (2r)^r \cdot d^{\frac{r}{2}} + (r-1) \cdot \frac{(r-1)!}{1} \cdot (2r)^{r-2} \cdot d^{\frac{r-2}{2}} \cdot r \cdot (r-1) \cdot d \\ &= (r-1)! \cdot (2r)^{r-2} \cdot \underbrace{(4r^2 + r \cdot (r-1)^2)}_{\substack{r \geq 1 \\ \leq 4r^3}} \cdot d^{\frac{r}{2}} \\ &= r! \cdot (2r)^r \cdot d^{\frac{r}{2}} \end{aligned}$$

sets. Hence (2.2) is also true for $t = r$.

Finally, let us count the number of tests that the algorithm performs at Stage II.

Number of tests

The algorithm performs all tests in step 5, 6, and 7. Each time a test of some set \mathcal{T} is positive, the algorithm finds a defective edge in $\mathring{D}(\mathcal{T})$ by, according to Theorem 1.2, not more than

$$\lceil \log_2 |E| \rceil + r - 1$$

tests per edge. Let d_{II} denote the number of defective edges that the algorithm finds at Stage II, then there are in total at most

$$d_{II}(\lceil \log_2 |E| \rceil + r)$$

tests to find d_{II} defective edges.

Next, consider the negative tests: After a negative test, the algorithm defines a new set F_k^{t+1} ; hence, there is one negative test for each set. Assume for the moment that the algorithm never returns to step 1 once it has entered step 4. In that case there are obviously

$$p_2 + p_3 + \dots + p_r$$

negative tests. However, since the algorithm finds defective edges after positive tests, the number of known defective edges increases, which in turn increases the number of D_K -neighbors for different sets C_j .

If $|\mathcal{N}_K(C_j)| > 2r\sqrt{d_K}$ holds for some C_j while the algorithm enters step \boxtimes for some t , then \mathcal{A}_r returns from step \boxtimes to step 1 and reconstructs the sets $F_1^\tau, F_2^\tau, \dots, F_{p_\tau}^\tau$ ($\tau = 1, \dots, t$).

Let now Φ denote the number of recurrences to step 1 and let d_i denote the number of defective edges that are known immediately after the i -th recurrence ($i = 0, \dots, \Phi$).

Each time when the algorithm constructs the sets C_1, C_2, \dots, C_q , the sets satisfy $|\mathcal{N}_K(C_j)| \leq r\sqrt{d_K}$, or more precisely, after the i -th recurrence they satisfy $|\mathcal{N}_K(C_j)| \leq r\sqrt{d_i}$, ($j = 1, \dots, q$). The $(i+1)$ -th recurrence to step 1 is caused by a set C_j with $|\mathcal{N}_K(C_j)| > 2r\sqrt{d_K}$. At that time $d_K = d_{i+1}$, hence, the number of D_K -neighbors of at least one set C_j has increased by at least $2r\sqrt{d_{i+1}} - r\sqrt{d_i}$. Every new found defective edge, on the other hand, increases $|\mathcal{N}_K(C_j)|$ by at most $r-1$. Accordingly, when the algorithm returns to step 1 for the $(i+1)$ -th time, it has found at least

$$\frac{1}{r-1} (2r\sqrt{d_{i+1}} - r\sqrt{d_i}) > \sqrt{d_{i+1}}$$

defective edges since the i -th recurrence. That is

$$d_i + \sqrt{d_{i+1}} \leq d_{i+1} \Leftrightarrow \sqrt{d_{i+1}} \leq d_{i+1} - d_i,$$

for $i = 0, \dots, \Phi - 1$. In total we receive

$$d \geq d_\Phi = d_0 + \sum_{i=0}^{\Phi-1} (d_{i+1} - d_i) \geq \sum_{i=0}^{\Phi-1} \sqrt{d_{i+1}}.$$

Since $d_i \geq 1$ for $0 \leq i \leq \Phi - 1$,

$$(\sqrt{d_{i+1}})^2 = d_{i+1} \geq d_i + \sqrt{d_{i+1}} \geq d_i + \frac{2}{3}\sqrt{d_i} + \frac{1}{9} = \left(\sqrt{d_i} + \frac{1}{3}\right)^2$$

and, hence, $\sqrt{d_{i+1}} \geq \sqrt{d_i} + \frac{1}{3}$ for $i = 0, \dots, \Phi - 1$

$$\Rightarrow \sqrt{d_i} \geq \frac{i}{3} \Rightarrow d \geq \sum_{i=0}^{\Phi-1} \frac{i}{3} = \frac{1}{3} \cdot \binom{\Phi}{2}.$$

Consequently,

$$\Phi \leq \sqrt{6d + 1/4} + 1/2 \leq 3 \cdot \sqrt{d}.$$

In the worst case, the algorithm returns to step 1, just when $t = r - 1$ and $k = p_t$. So there are at most

$$\begin{aligned} & 3 \cdot \sqrt{d} \cdot (p_2 + p_3 + \dots + p_{r-1}) + p_r \\ \stackrel{(2.2)}{\leq} & 3 \cdot \sqrt{d} \cdot \sum_{t=2}^{r-1} (t! \cdot (2r)^t \cdot d^{\frac{t}{2}}) + r! \cdot (2r)^r \cdot d^{\frac{r}{2}} \\ \leq & r! \cdot (2r)^r \cdot \left(\sum_{t=1}^{r-1} \frac{3 \cdot t!}{r! \cdot (2r)^{r-t}} + 1 \right) \cdot d^{\frac{r}{2}} \\ \leq & r! \cdot (2r)^r \cdot \underbrace{\left((r-1) \cdot \frac{3}{r \cdot (2r)} + 1 \right)}_{\substack{r \geq 2 \\ \leq 1}} \cdot d^{\frac{r}{2}} \\ \leq & r \cdot r! \cdot (2r)^r \cdot d^{\frac{r}{2}} \end{aligned}$$

negative tests in Stage II.

Stage III

After Stage II, all unknown defective edges share at least two vertices with at least one known defective edge. Suppose now e is an unknown defective edge. Let $D' = \{g_1, g_2, \dots, g_\alpha\} \subset D_K$ be a set of known defective edges, such that firstly

$$\left| g_i \cap \left(e \setminus \bigcup_{\substack{1 \leq j \leq \alpha, \\ j \neq i}} g_j \right) \right| \geq 2, \text{ for } i = 1, \dots, \alpha$$

and secondly

$$\left| g \cap \left(e \setminus \bigcup_{j=1}^{\alpha} g_j \right) \right| \leq 1, \text{ for } g \in D_K \setminus D'$$

holds. Then we have

- $\alpha \leq \lfloor \frac{r}{2} \rfloor$,
- $e \setminus \bigcup_{j=1}^{\alpha} g_j$ is strongly D_K -independent, and
- $\left| e \setminus \bigcup_{j=1}^{\alpha} g_j \right| \leq r - 2\alpha$.

The idea of Stage III is now the following: For every $1 \leq \alpha \leq \lfloor \frac{r}{2} \rfloor$ and each set $D' \subset D_K$ with $|D'| = \alpha$, we construct free sets

$$G_1^{D'}, G_2^{D'}, \dots, G_{q_{D'}}^{D'}$$

that cover all sets in $\mathcal{P}_K^{r-2\alpha}(V \setminus \bigcup_{g \in D'} g)$. Then each unknown defective edge lies in a set

$$\mathring{D} \left(\bigcup_{g \in D'} g \cup G_k^{D'} \right)$$

for some $1 \leq \alpha \leq \lfloor \frac{r}{2} \rfloor$, $D' \subset D_K$ with $|D'| = \alpha$ and $1 \leq k \leq q_{D'}$.

Let us start by constructing free sets that cover all sets in $\mathcal{P}_K^t(W)$ for any $W \subset V$ and $1 \leq t \leq r-2$. As in the previous stage, we build those sets up step by step; but unlike before, these sets are themselves strongly D_K -independent. Since strongly D_K -independent sets are by now also free, we need no tests for their construction.

Lemma 2.14. *Let $W \subset V$ and $0 \leq t \leq r-2$. There is an algorithm that constructs*

$$q_t \leq \frac{(t+2)!}{2} \cdot r^t \cdot d^{\frac{t}{2}}$$

strongly D_K -independent sets $G_1^t, G_2^t, \dots, G_{q_t}^t$, such that every set $X \in \mathcal{P}_K^t(W)$ lies in some set G_k^t .

Proof. The following algorithm $\mathcal{A}_{G^{D'}}$ constructs the desired sets:

Set $\mathcal{X} := \{\{v\} : v \in W\}$, $C_n := \emptyset$ for $1 \leq n \leq |W|$, and $i, j := 1$.

1. If there are two sets $X, X' \in \mathcal{X}$, $X \neq X'$, with $X \cup X'$ strongly D_K -independent, set $\mathcal{X} := \mathcal{X} \cup \{X \cup X'\} \setminus \{X, X'\}$ and repeat step 1.
Else, choose an arbitrary $X \in \mathcal{X}$, set $\mathcal{X} := \mathcal{X} \setminus \{X\}$, and go to step 2.
2. If there is a vertex $x \in X$ with $|\mathcal{N}_K(\{x\})| \geq r/2 \cdot \sqrt{d_K}$, set $B_i := \{x\}$, $i := i + 1$, $X := X \setminus \{x\}$, and repeat step 2, else, go to step 3.
3. If there is a vertex $x \in X$ with $|\mathcal{N}_K(\{x\} \cup C_j)| \leq r\sqrt{d_K}$, set $C_j := C_j \cup \{x\}$, $X := X \setminus \{x\}$, and repeat step 3, else, if
 $X \neq \emptyset$: Set $j := j + 1$ and repeat step 3.
 $X = \emptyset$: $\mathcal{X} \neq \emptyset$: Set $j := j + 1$, choose an arbitrary $X \in \mathcal{X}$, set $\mathcal{X} := \mathcal{X} \setminus \{X\}$, and repeat step 2.
 $\mathcal{X} = \emptyset$: Set $p := i$, $q := j$.
4. Set $G_i^1 := B_i$ for $i = 1, \dots, p$, $G_{p+j}^1 := C_j$ for $j = 1, \dots, q$, and $q_1 := p+q$, further set $B := \bigcup_{i=1}^p B_i$ as well as $i, j, k, n, \tau := 1$.
5. Set $G_n^{\tau+1} := B_i \cup (G_k^\tau \setminus \mathcal{N}_K(B_i))$ and $n := n + 1$.
 $i < p$: Set $i := i + 1$ and repeat step 5.
 $i = p$: Go to step 6.
6. Set $G_n^{\tau+1} := (G_k^\tau \setminus B) \cup (C_j \setminus \mathcal{N}_K(G_k^\tau \setminus B))$ and $n := n + 1$.
 $j < q$: Set $j := j + 1$ and repeat step 6.
 $j = q$: Set $N := \mathcal{N}_K(G_k^\tau \setminus B) \setminus B$, choose a vertex $x \in N$, and go to step 7.

7. Set $G_n^{\tau+1} := \{x\} \cup (G_k^\tau \setminus \mathcal{N}_K(\{x\}))$, $X := X \setminus \{x\}$ and $n := n + 1$.
 $N \neq \emptyset$: Choose a vertex $x \in N$ and repeat step 7.
 $N = \emptyset$: $k < q_\tau$: Set $k := k + 1$ and $i, j := 1$, then repeat step 5.
 $k = q_\tau$: $\tau < t - 1$: Set $\tau := \tau + 1$, $q_\tau := n$, and $i, j, k, n := 1$, then repeat step 5.
 $\tau = t - 1$: Stop.

In step 1 the algorithm starts with $|W|$ sets, each containing a single vertex. Then $\mathcal{A}_{G^{D'}}$ joins two sets to a new set if their union is strongly D_K -independent. This process terminates if for any two sets X and $X' \in \mathcal{X}$, there is some known defective edge e with $e \cap X \neq \emptyset$ and $e \cap X' \neq \emptyset$. Every defective edge, on the other hand, joins vertices from at most r different sets. That is, every known defective edge joins vertices from at most $\binom{r}{2}$ pairs of sets $X \in \mathcal{X}$. Thus, for the number of sets $|\mathcal{X}|$ applies

$$\binom{|\mathcal{X}|}{2} \leq \binom{r}{2} \cdot d_K,$$

for $r \geq 2$ and $d_K \geq 1$ we have

$$|\mathcal{X}| \leq r \sqrt{d_K}.$$

Since $\mathcal{A}_{G^{D'}}$ constructs the sets $G_1^t, G_2^t, \dots, G_{q_t}^t$ similar to \mathcal{A}_r 's construction of $F_1^r, F_2^r, \dots, F_{p_r}^r$, we receive by the same argumentation:

Firstly, for $1 \leq \tau \leq t$, the sets $G_1^\tau, G_2^\tau, \dots, G_{q_\tau}^\tau$ cover all sets in $\mathcal{P}_K^\tau(W)$. Please note, unlike the D_K -neighbours of the sets F_k^t , the number of the D_K -neighbours of the sets $G_1^\tau, G_2^\tau, \dots, G_{q_\tau}^\tau$ does not change during the algorithm since no new defective edges are found.

Secondly, for every set G_k^τ there is an index set $I \subset \{1, 2, \dots, q_1\}$, $|I| = \tau$ such that

$$G_k^\tau \subseteq \bigcup_{i \in I} G_i^1$$

and thus

$$\left| \mathcal{N}_K(G_k^\tau \setminus B) \setminus B \right| \leq \left| \mathcal{N}_K\left(\bigcup_{i \in I} G_i^1 \setminus B\right) \right| \leq \sum_{i \in I} \left| \mathcal{N}_K(G_i^1 \setminus B) \right| \leq \tau \cdot r \sqrt{d_K}.$$

Thirdly,

$$q_1 \leq \sum_{X \in \mathcal{X}} \left\lceil \frac{\mathcal{N}_K(X)}{r/2 \cdot \sqrt{d_K}} \right\rceil \leq (3r - 2) \cdot \sqrt{d_K} \leq \frac{(1 + 2)!}{2} \cdot r^1 \cdot d_K^{\frac{1}{2}},$$

and finally we receive for $2 \leq \tau \leq t - 1$ inductively that

$$q_\tau \leq \frac{(\tau + 2)!}{2} \cdot r^\tau \cdot d_K^{\frac{\tau}{2}}$$

owing to

$$\begin{aligned} q_{\tau+1} &= \sum_{k=1}^{q_\tau} \left(p + q + \left| \mathcal{N}_K(G_k^\tau \setminus B) \setminus B \right| \right) \\ &\leq \frac{(\tau + 2)!}{2} \cdot r^\tau \cdot d_K^{\frac{\tau}{2}} \cdot \left((3r - 2) \sqrt{d_K} + \tau \cdot r \sqrt{d_K} \right) \end{aligned}$$

$$\leq \frac{(\tau + 3)!}{2} \cdot r^{\tau+1} \cdot d_K^{\frac{\tau+1}{2}}.$$

Unlike the sets $F_1^t, F_2^t, \dots, F_{p_t}^t$, the sets $G_1^\tau, G_2^\tau, \dots, G_{q_\tau}^\tau$ are obliged to be strongly D_K -independent, which we show by induction on τ .

$\tau = 1$:

Obviously the sets $G_1^1, G_2^1, \dots, G_{q_1}^1$ are strongly D_K -independent.

$\tau - 1 \rightarrow \tau$:

Suppose $\tau \leq t$ and all sets $G_k^{\tau-1}$ are strongly D_K -independent. For every set G_n^τ there is some $G_k^{\tau-1}$ and some G_i^1 such that

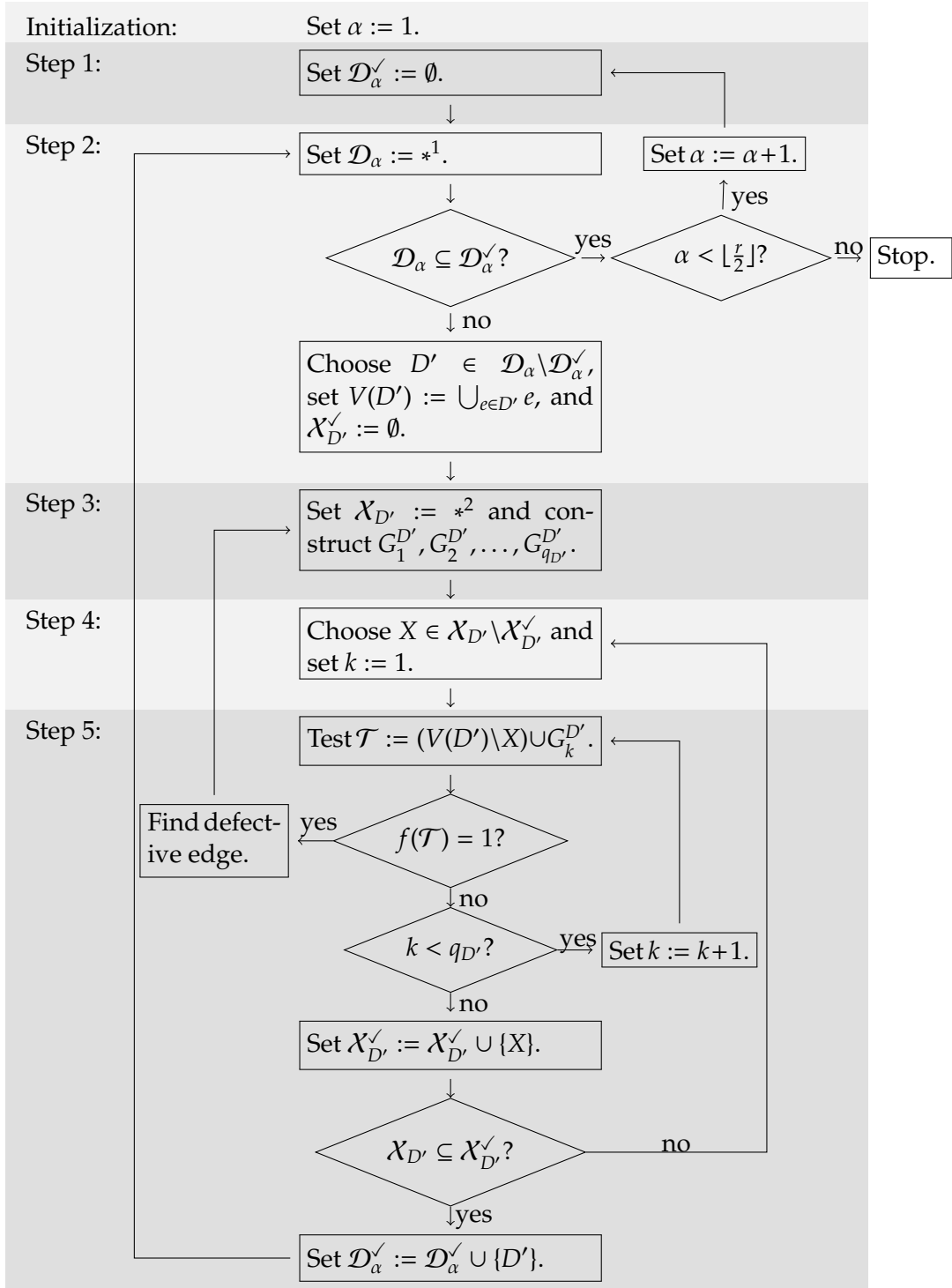
$$G_n^\tau \subseteq (G_k^{\tau-1} \cup G_i^1).$$

Since both $G_k^{\tau-1}$ and G_i^1 are strongly D_K -independent, G_n^τ is not strongly D_K -independent if and only if there is some known defective edge that joins a vertex in $G_k^{\tau-1} \cap G_n^\tau$ and a vertex in $G_i^1 \cap G_n^\tau$. Seeing that the algorithm subtracts among others all D_K -neighbours of either $G_k^{\tau-1}$ or G_i^1 from $G_k^{\tau-1} \cup G_i^1$, G_n^τ is of course strongly D_K -independent. \square

\mathcal{A}_r - Stage III

Set $\alpha := 1$.

1. Set $\mathcal{D}_\alpha^\vee := \emptyset$.
2. Set $\mathcal{D}_\alpha := \{D' \subset D_K : |D'| = \alpha, |e| \geq 3 \forall e \in D'\}$.
 $\mathcal{D}_\alpha \subseteq \mathcal{D}_\alpha^\vee$: $\alpha < \lfloor \frac{r}{2} \rfloor$: Set $\alpha := \alpha + 1$ and repeat step 1.
 $\alpha \geq \lfloor \frac{r}{2} \rfloor$: Stop.
 $\mathcal{D}_\alpha \not\subseteq \mathcal{D}_\alpha^\vee$: Choose an arbitrary edge set $D' \in \mathcal{D}_\alpha \setminus \mathcal{D}_\alpha^\vee$, $V(D') := \bigcup_{e \in D'} e$, and $\mathcal{X}_{D'}^\vee := \emptyset$.
3. Set
 - $\mathcal{X}_{D'} := \{Y \cup \Gamma_K(V(D') \setminus Y) : Y \subset V(D'), |Y \cap e| \geq 1 \forall e \in D'\}$ and
 - construct strongly D_K -independent sets $G_1^{D'}, G_2^{D'}, \dots, G_{q_{D'}}^{D'}$ that cover all sets in $\mathcal{P}_K^{r-2\alpha}(V \setminus V(D'))$
4. Choose an arbitrary vertex set $X \in \mathcal{X}_{D'} \setminus \mathcal{X}_{D'}^\vee$ and set $k := 1$.
5. Test $\mathcal{T} := (V(D') \setminus X) \cup G_k^{D'}$. If
 - $f(\mathcal{T}) = 1$: Find a defective edge by Triesch's search in $\mathring{D}(\mathcal{T})$ and repeat step 3.
 - $f(\mathcal{T}) = 0$: $k < q_{D'}$: Set $k := k + 1$, repeat step 5.
 $k = q_{D'}$: Set $\mathcal{X}_{D'}^\vee := \mathcal{X}_{D'}^\vee \cup \{X\}$.
 $\mathcal{X}_{D'} \not\subseteq \mathcal{X}_{D'}^\vee$: Repeat step 4.
 $\mathcal{X}_{D'} \subseteq \mathcal{X}_{D'}^\vee$: Set $\mathcal{D}_\alpha^\vee := \mathcal{D}_\alpha^\vee \cup \{D'\}$ and repeat step 2.



$$*^1 = \{D' \subset D_K : |D'| = \alpha, |e| \geq 3 \forall e \in D'\}$$

$$*^2 = \{Y \cup \Gamma_K(V(D') \setminus Y) : Y \subset V(D'), |Y \cap e| \geq 1 \forall e \in D'\}$$

Figure 2.9: \mathcal{A}_r - Stage III

All tests are reasonable

Suppose otherwise, suppose there is some test set $\mathcal{T} = (V(D') \setminus X) \cup G_k^{D'}$ which is not D_K -independent and let $e \in \mathring{D}_K(\mathcal{T})$. Since $X \cap e = \emptyset$, certainly $e \notin D'$. Suppose, $|e \setminus V(D')| = 1$ and let $Y := V(D') \cap X$. Then we have $e \cap \Gamma_K(V(D') \setminus Y) \neq \emptyset$, which is a contradiction, since $\Gamma_K(V(D') \setminus Y) \subset X$ and thus $\mathcal{T} \cap \Gamma_K(V(D') \setminus Y) = \emptyset$. Thus, we have $|e \setminus V(D')| \geq 2$. But $G_k^{D'}$ is strongly D_K -independent, that is $|e \cap G_k^{D'}| \leq 1$ for all $e \in D_K$, which is a contradiction. Accordingly, all tests are reasonable.

The algorithm finds all defective edges

Let $e \in D$ be some defective edge that is still unknown after Stage II. After Stage II there is for every unknown defective edge $e \in D$ some known defective edge $g \in D_K$ with $|e \cap g| \geq 2$. Assign to every edge $e \in D \setminus D_K$ some set $D'(e) = \{g_1, g_2, \dots, g_{\alpha(e)}\} \subset D_K$ with a minimum number $\alpha(e)$ of edges, such that firstly,

$$\left| g_i \cap \left(e \setminus \bigcup_{\substack{1 \leq j \leq \alpha(e) \\ j \neq i}} g_j \right) \right| \geq 2, \text{ for } 1 \leq i \leq \alpha(e)$$

and secondly,

$$\left| g \cap \left(e \setminus \bigcup_{j=1}^{\alpha(e)} g_j \right) \right| \leq 1, \text{ for } g \in D_K \setminus D'(e)$$

holds.

Consider now Stage III when the algorithm sets $\mathcal{D}_{\alpha(e)}^\vee := \mathcal{D}_{\alpha(e)}^\vee \cup \{D'(e)\}$. If the algorithm has found e in the meantime, there is nothing to show; hence, we assume that e is still unknown. If the algorithm has not yet found e , there are two possibilities:

Either no test set contains e so far, or there has been some set that contains e but the algorithm has found some other defective edge.

So far, no test contains e:

Since we consider only simple hypergraphs, there is surely some set $X \subset V(D'(e))$ with $e \cap X = \emptyset$. As there has been no test \mathcal{T} with $e \subset \mathcal{T}$, no set $G_k^{D'(e)}$ covers $e \setminus V(D'(e))$. Consequently, $e \setminus V(D'(e))$ is not strongly D_K -independent, which in turn implies that meanwhile the algorithm has found some defective edge $g_{\alpha(e)+1}$ with

$$\left| g_{\alpha(e)+1} \cap \left(e \setminus \bigcup_{j=1}^{\alpha(e)} g_j \right) \right| \geq 2. \quad (2.9)$$

There has been a test \mathcal{T} with $e \subseteq \mathcal{T}$:

We assume that the algorithm has found some other defective edge g . After finding g , the algorithm returns to step 3 without setting $\mathcal{X}_{D'(e)}^\vee = \mathcal{X}_{D'(e)}^\vee \cup \{X\}$. Hence, there will either be a new set \mathcal{T} with $e \subset \mathcal{T}$, or we find some defective edge $g_{\alpha(e)+1}$ with (2.9).

The algorithm does not set $\mathcal{D}_{\alpha(e)}^\vee := \mathcal{D}_{\alpha(e)}^\vee \cup \{D'(e)\}$ before it has set $\mathcal{X}_{D'(e)}^\vee = \mathcal{X}_{D'(e)}^\vee \cup \{X\}$, which only happens when all sets

$$(V(D'(e)) \setminus X) \cup G_k^{D'(e)}$$

are free ($k = 1, \dots, q_{D'(e)}$). But that means that $e \setminus V(D'(e))$ is no longer strongly D_K -independent. So, when the algorithm sets $\mathcal{D}'_{\alpha(e)} := \mathcal{D}'_{\alpha(e)} \cup \{D'(e)\}$ but has still not found e , the number $\alpha(e)$ has increased. Owing that α cannot exceed $\lfloor r/2 \rfloor$, the algorithm certainly finds all defective edges.

The number of tests

For each $X \in \mathcal{X}_{D'}$ with $D' \in \mathcal{D}'_{\alpha}$ and $1 \leq \alpha \leq \lfloor r/2 \rfloor$ there are tests

$$\mathcal{T} := (V(D') \setminus X) \cup G_k^{D'}$$

for increasing k until either $k = q_{D'}$ or $f(\mathcal{T}) = 1$. If $f(\mathcal{T}) = 1$, the algorithm detects one defective edge by Triesch's search and repeats step 3. Then, in step 3, \mathcal{A}_r renews $\mathcal{X}_{D'}$ and $G_1^{D'}, G_2^{D'}, \dots, G_{q_{D'}}^{D'}$ with respect to the enlarged D_K .

And there lies the difficulty of estimating the number of negative tests in Stage III: The sets \mathcal{D}'_{α} , $\mathcal{X}_{D'}$ and $G_1^{D'}, G_2^{D'}, \dots, G_{q_{D'}}^{D'}$ depend upon D_K . Since D_K is not static, neither are the other sets which the algorithm resets on several occasions. So, to avoid performing the same tests twice, the algorithm remembers sets $X \in \mathcal{X}_{D'}$ for which all tests

$$\mathcal{T} := (V(D') \setminus X) \cup G_k^{D'}$$

for $1 \leq k \leq q_{D'}$ were negative by adding it to $\mathcal{X}'_{D'}$. The same holds for all sets $D' \in \mathcal{D}'_{\alpha}$ with $\mathcal{X}_{D'} \subseteq \mathcal{X}'_{D'}$: \mathcal{A}_r adds them to \mathcal{D}'_{α} . Thus, we have for each $X \in \mathcal{X}_{D'}$ with $D' \in \mathcal{D}'_{\alpha}$ and $1 \leq \alpha \leq \lfloor r/2 \rfloor$ exactly $q_{D'}$ negative tests and this adds up to

$$\sum_{\alpha=1}^{\lfloor r/2 \rfloor} \sum_{D' \in \mathcal{D}'_{\alpha}} \sum_{X \in \mathcal{X}'_{D'}} q_{D'}. \quad (2.10)$$

But what happens now if for some set $X \in \mathcal{X}_{D'}$ not all tests $\mathcal{T} := (V(D') \setminus X) \cup G_k^{D'}$ were negative? We have already seen that the algorithm finds a defective edge and repeats step 3. It is crucial that \mathcal{A}_r repeats step 3 without adding X to $\mathcal{X}'_{D'}$. Hence, we have not counted those tests in (2.10). In the worst case, all test $\mathcal{T} := (V(D') \setminus X) \cup G_k^{D'}$ for $1 \leq k \leq q_{D'} - 1$ are positive and only $\mathcal{T} := (V(D') \setminus X) \cup G_{q_{D'}}^{D'}$ is negative. Accordingly, we have at most

$$\sum_{\alpha=1}^{\lfloor r/2 \rfloor} \sum_{D' \in \mathcal{D}'_{\alpha}} \sum_{X \in \mathcal{X}'_{D'}} q_{D'} + d \cdot \max_{D' \subset D, |D'| \leq \lfloor \frac{r}{2} \rfloor} q_{D'}$$

negative tests at Stage III. Let us now consider (2.10). First of all, it is

$$\mathcal{D}'_{\alpha} \subseteq \binom{D}{\alpha} \Rightarrow |\mathcal{D}'_{\alpha}| \leq d^{\alpha},$$

further, for $|D'| = \alpha$

$$|\mathcal{X}'_{D'}| \leq 2^{|V(D')|} \leq 2^{\alpha r},$$

and finally, due to Lemma 2.14,

$$q_{D'} \leq \frac{(r - 2\alpha + 2)!}{2} \cdot r^{r-2\alpha} \cdot d^{\frac{r-2\alpha}{2}}$$

for $D' \in \mathcal{D}'_\alpha$. That amounts to

$$\begin{aligned}
& \sum_{\alpha=1}^{\lfloor r/2 \rfloor} \sum_{D' \in \mathcal{D}'_\alpha} \sum_{X \in \mathcal{X}'_{D'}} q_{D'} \\
& \sum_{\alpha=1}^{\lfloor r/2 \rfloor} d^\alpha \cdot 2^{\alpha \cdot r} \cdot \frac{(r-2\alpha+2)!}{2} \cdot r^{r-2\alpha} \cdot d^{\frac{r-2\alpha}{2}} \\
& = \sum_{\alpha=1}^{\lfloor r/2 \rfloor} \frac{(r-2\alpha+2)!}{2} \cdot r^{r-2\alpha} \cdot 2^{\alpha \cdot r} \cdot d^{\frac{r}{2}} \\
& \leq \left\lfloor \frac{r}{2} \right\rfloor \cdot \frac{r!}{2} \cdot r^{r-2} \cdot 2^{\frac{r}{2} \cdot r} \cdot d^{\frac{r}{2}}.
\end{aligned}$$

In total, there are at most

$$\begin{aligned}
& \sum_{\alpha=1}^{\lfloor r/2 \rfloor} \sum_{D' \in \mathcal{D}'_\alpha} \sum_{X \in \mathcal{X}'_{D'}} q_{D'} + d \cdot \max_{D' \subset D, |D'| \leq \lfloor \frac{r}{2} \rfloor} q_{D'} \\
& \leq \left\lfloor \frac{r}{2} \right\rfloor \cdot \frac{r!}{2} \cdot r^{r-2} \cdot 2^{\frac{r}{2} \cdot r} \cdot d^{\frac{r}{2}} + d \cdot \frac{r!}{2} \cdot r^{r-2} \cdot d^{\frac{r-2}{2}} \\
& \leq \frac{r!}{2} \cdot r^{r-1} \cdot 2^{\frac{r^2}{2}} \cdot d^{\frac{r}{2}}
\end{aligned}$$

tests. Finally, let d_{III} denote the number of defective edges at Stage III, then there are, according to Theorem 1.2, at most

$$d_{III} \cdot (\lceil \log_2 |E| \rceil + r - 1)$$

additional tests to identify the defective edges in Stage III.

2.2.2 An upper bound for the number of tests

By adding up the tests that the algorithm requires in its three stages, we receive:

Theorem 2.15. *Let $H = (V, E)$ be a simple hypergraph of bounded rank r with defective edge set $D \subset V$. Then there is an algorithm that finds all defective edges with at most*

$$d \cdot \lceil \log_2 |E| \rceil + c_r \cdot d^{\frac{r}{2}}$$

tests for some constant $c_r > 0$.

Proof. We have already seen that the algorithm \mathcal{A}_r needs at most

$$\begin{aligned}
& 2\sqrt{d} + d_I \cdot (\lceil \log_2 |E| \rceil + r) + r \cdot r! \cdot (2r)^r \cdot d^{\frac{r}{2}} + d_{II} \cdot (\lceil \log_2 |E| \rceil + r) \\
& + \frac{r!}{2} \cdot r^{r-1} \cdot 2^{\frac{r^2}{2}} \cdot d^{\frac{r}{2}} + d_{III} \cdot (\lceil \log_2 |E| \rceil + r - 1) \\
& \leq \left(2 + r \cdot r! \cdot (2r)^r + \frac{r!}{2} \cdot r^{r-1} \cdot 2^{\frac{r^2}{2}} + r \right) \cdot d^{\frac{r}{2}} + d \cdot \lceil \log_2 |E| \rceil
\end{aligned}$$

tests to find all defective edges. Hence, \mathcal{A}_r satisfies Theorem 2.15 for

$$c_r = 2 + r \cdot r! \cdot (2r)^r + \frac{r!}{2} \cdot r^{r-1} \cdot 2^{\frac{r}{2}} + 3.$$

□

Remark 2.16. For $r = 3$ the above algorithm needs at most

$$d \cdot \lceil \log_2 |E| \rceil + 2\sqrt{d} + 3 \cdot d + (3888 + 432 \cdot \sqrt{2}) \cdot d^{\frac{3}{2}}$$

tests. These are, of course, considerably more tests than the algorithm of the previous section requires.

Chapter 3

Uniform hypergraphs

3.1 3-uniform hypergraphs

In this chapter we consider 3-uniform hypergraphs, that is, hypergraphs whose edges all have cardinality 3. Certainly, every 3-uniform hypergraph is a hypergraph of rank 3, and we have already presented two algorithms that find all defective edges in a hypergraph of rank 3.

In this chapter, though, we present an algorithm that proves the Conjecture of Du and Hwang (s. [DH93]) for 3-uniform hypergraphs and finds all defective edges in any 3-uniform hypergraph by at most $\lceil \log_2(|E|/d) \rceil + cd$ tests, where d (the number of defective edges) is known and c is some constant.

The rough idea is the following:

We partition V into free vertex sets and test each triple of sets to find all defective edges. To achieve the bound of at most $\lceil \log_2(|E|/d) \rceil + O(d)$ tests, the number of sets must not exceed $O(d^{1/3})$. In the following Lemma, we will present a construction of such sets. But first of all, let us start with some old and new definitions.

Definition 3.1. Let $H = (V, E)$ be a hypergraph with defective edge set $D \subset E$ of cardinality $d := |D|$. Once a defective edge e has been detected, we say e is a known defective edge and $D_K \subseteq D$ denotes the set of known defective edges; its cardinality is $d_K := |D_K|$. We call a vertex set $X \subset V$ D_K -independent if there is no $e \in D_K$ with $e \subseteq X$, and a set is said to be free if $f(X) = 0$. Thus, every free vertex set is also D_K -independent. Let $X, U, Y, Z \subset V$ be arbitrary vertex sets. We define the following edge sets

$$\begin{aligned} \mathring{D}(X) &:= \{e \in D : e \subseteq X\}, \\ D(X, U) &:= \{e \in \mathring{D}(X \cup U) : e \cap X \neq \emptyset\}, \text{ and} \\ D(X, Y, Z) &:= \{e \in \mathring{D}(X \cup Y \cup Z) : |e \cap X| = |e \cap Y| = |e \cap Z| = 1\} \end{aligned}$$

with cardinality $\mathring{d}(X) := |\mathring{D}(X)|$, $d(X, U) := |D(X, U)|$ and $d(X, Y, Z) := |D(X, Y, Z)|$, respectively. For some free set $X \subset V$ and arbitrary $U \subseteq V$, define

$$\begin{aligned} \Gamma(X, U) &:= \{v \in U \setminus X : \mathring{D}(\{v\} \cup X) \neq \emptyset\} \text{ and} \\ e(X, U) &:= \sum_{v \in U} \mathring{d}(\{v\} \cup X) = \sum_{v \in \Gamma(X, U)} \mathring{d}(\{v\} \cup X). \end{aligned}$$

Please note that $|\Gamma(X, U)| \leq e(X, U)$ for any free X and arbitrary U . For $U = V$ we write $D(X)$, $d(X)$, $\Gamma(X)$, and $e(X)$ instead of $D(X, V)$, $d(X, V)$, $\Gamma(X, V)$ and $e(X, V)$. Now let $D_K \subseteq D$ be the set of known defective edges; then we set analogously for $X, U, Y, Z \subseteq V$:

$$\begin{aligned} \mathring{D}_K(X) &:= \{e \in D_K : e \subseteq X\}, \\ D_K(X, U) &:= \{e \in \mathring{D}_K(X \cup U) : e \cap X \neq \emptyset\} \text{ and} \\ D_K(X, Y, Z) &:= \{e \in \mathring{D}_K(X \cup Y \cup Z) : |e \cap X| = |e \cap Y| = |e \cap Z| = 1\} \end{aligned}$$

with cardinality $\mathring{d}_K(X) := |\mathring{D}_K(X)|$, $d_K(X, U) := |D_K(X, U)|$ and $d_K(X, Y, Z) := |D_K(X, Y, Z)|$, respectively. For a free vertex set X and arbitrary $U \subseteq V$, define

$$\begin{aligned} \Gamma_K(X, U) &:= \{v \in U \setminus X : \mathring{D}_K(\{v\} \cup X) \neq \emptyset\} \text{ and} \\ e_K(X, U) &:= \sum_{v \in U} \mathring{d}_K(\{v\} \cup X) = \sum_{v \in \Gamma_K(X, U)} \mathring{d}_K(\{v\} \cup X) \end{aligned}$$

Again, $|\Gamma_K(X, U)| \leq e_K(X, U)$ for any free set X and arbitrary U . Write, as before, for $U = V$ instead of $D_K(X, V)$, $d_K(X, V)$, $\Gamma_K(X, V)$ and $e_K(X, V)$ the short forms $D_K(X)$, $d_K(X)$, $\Gamma_K(X)$ and $e_K(X)$.

Let now $H = (V, E)$ be a hypergraph of rank 3 without loops. Let $D_K = D_2 \dot{\cup} D_3$ be the set of known defective edges where

$$D_2 := \{e \in D_K : |e| = 2\} \text{ and } D_3 := \{e \in D_K : |e| = 3\},$$

and its cardinalities are denoted by $d_2 := |D_2|$ and $d_3 := |D_3|$, respectively. In analogy to $d_K(X, U)$, define for any two sets $X, U \subseteq V$:

$$\begin{aligned} d_2(X, U) &:= |\{e \in D_2 : e \subset X \cup U, e \cap X \neq \emptyset\}|, \\ d_3(X, U) &:= |\{e \in D_3 : e \subset X \cup U, e \cap X \neq \emptyset\}|, \\ \mathring{d}_2(X) &:= |\{e \in D_2 : e \subset X\}|, \text{ and} \\ \mathring{d}_3(X) &:= |\{e \in D_3 : e \subset X\}|, \end{aligned}$$

and finally for $X \subset V$ free and arbitrary $U \subseteq V$:

$$\begin{aligned} e_2(X, U) &:= d_2(X, U) \text{ and} \\ e_3(X, U) &:= \sum_{v \in U} \mathring{d}_3(\{v\} \cup X). \end{aligned}$$

Then, we have

$$e_K(X, U) = \sum_{v \in U} \mathring{d}_K(\{v\} \cup X) = d_2(X, U) + \sum_{v \in U} \mathring{d}_3(\{v\} \cup X) = e_2(X, U) + e_3(X, U).$$

We set $d_2(X)$ instead of $d_2(X, V)$ and also $d_3(X)$, $e_2(X)$ and $e_3(X)$ instead of $d_3(X, V)$, $e_2(X, V)$ and $e_3(X, V)$, respectively.

Let

$$\rho := 2^{\lceil \log_2(|E|/d) \rceil}.$$

Then we say that a test of a vertex set $X \subset V$ is *substantial* if either $f(X) = 1$ or $|E(X)| < \rho/2$. We have already seen that good edges do not affect the outcome of a test. Thus, we regard an edge that is known to be good as removed. That, in turn, implies that a search algorithm performs at most

$$\frac{|E|}{\rho/2} = \frac{2 \cdot |E|}{2^{\lceil \log_2 |E|/d \rceil}} \leq 2 \cdot d$$

non-substantial tests.

In the following, we will count only substantial tests. An upper bound for the total number of tests can be easily established by adding $2d$ tests to the number of substantial tests.

Further Approach

The search algorithm that we will present in this section starts by splitting V into free vertex sets C_1, C_2, \dots, C_p . Afterwards, the algorithm searches for each $1 \leq i \leq p$ for all defective edges in

$$D\left(C_i, \bigcup_{j=i+1}^p C_j\right).$$

All edges $e \in D\left(C_i, \bigcup_{j=i+1}^p C_j\right)$ can be assigned to one of the four cases below (cf. Figure 3.1):

1. $|e \cap C_i| = 2$ and $|e \cap \Gamma(C_i, \bigcup_{j=i+1}^p C_j)| = 1$,
2. $|e \cap C_i| = 1$ and $|e \cap \Gamma(C_i, \bigcup_{j=i+1}^p C_j)| = 2$,
3. $|e \cap C_i| = 1$, $|e \cap \Gamma(C_i, \bigcup_{j=i+1}^p C_j)| = 1$ and $|e \cap \left(\bigcup_{j=i+1}^p C_j \setminus \Gamma(C_i)\right)| = 1$, or
4. $|e \cap C_i| = 1$ and $|e \cap \left(\bigcup_{j=i+1}^p C_j \setminus \Gamma(C_i)\right)| = 2$.

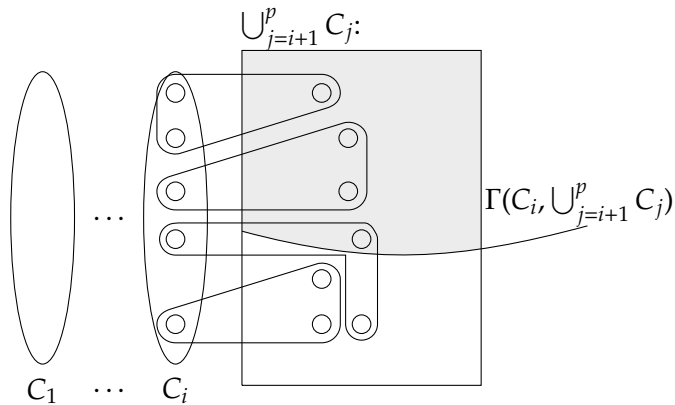


Figure 3.1: Scheme: Possible set-up for defective edges

Before we introduce the algorithm, first some lemmas:

- The first lemmas describe how to partition a given vertex set into free sets. Since the algorithm has to consider at times edges of rank 2, we will present a more general algorithm that partitions a vertex set of a hypergraph of rank 3.
- Afterwards, Lemma 3.7 describes how to identify all vertices in $\Gamma(C)$ for a given set C .
- Finally, we introduce some lemmas that describe how to find defective edges in the different set-ups.

Lemma 3.2. *Let $a, b \in \mathbb{N}$ such that $\binom{a-1}{3} < \max\{1, d_3\} \leq \binom{a}{3}$ and $\binom{b-1}{2} < \max\{1, d_2\} \leq \binom{b}{2}$. Then there is a D_K -independent set $X \subset V$ with*

$$(*) : \quad e_3(X) \leq \frac{d_3(X)}{a-2} + d_2(X) \text{ and} \\ d_2(X) \geq \frac{b-1}{28} \text{ or } d_3(X) \geq \frac{1}{14} \cdot \binom{a-1}{2}.$$

Proof. We differentiate between $a \geq b$ and $a < b$ and present for each case an algorithm that constructs a set $X \subset V$ that satisfies (*).

a \geq b:

Algorithm $\mathcal{A}_{a \geq b}$:

Let $\tilde{d}(X) := (a-2) \cdot d_2(X) + d_3(X)$ and $A := \frac{1}{7} \cdot \binom{a-1}{2}$.

1. Choose a vertex $x \in V$ with $\max_{v \in V} \tilde{d}(\{v\}) = \tilde{d}(\{x\})$ and set $X := \{x\}$.
2. If $\tilde{d}(X) \geq A$, stop. Else, choose a vertex $x \in W := V \setminus (X \cup \Gamma_K(X))$ with

$$\tilde{d}(X \cup \{x\}) \geq (a-2) \cdot e_K(X \cup \{x\})$$

and set $X := X \cup \{x\}$. Repeat step 2.

The procedure starts by choosing a single vertex which is, since there are no loops, certainly D_K -independent. We have $e_2(\{x\}) = e_K(\{x\})$ for all $x \in V$, therefore

$$\tilde{d}(\{x\}) = (a-2) \cdot d_2(\{x\}) + d_3(\{x\}) \geq (a-2) \cdot d_2(\{x\}) = (a-2) \cdot e_2(\{x\}) = (a-2) \cdot e_K(\{x\}).$$

The algorithm expands X only by vertices $x \in W = V \setminus (X \cup \Gamma_K(X))$ that satisfy

$$\tilde{d}(X \cup \{x\}) \geq (a-2) \cdot e_K(X \cup \{x\})$$

until eventually $\tilde{d}(X) \geq A$. Accordingly, each set X in the process satisfies

$$\tilde{d}(X) \geq (a-2) \cdot e_K(X).$$

As $W \cap \Gamma_K(X) = \emptyset$, $\{x\} \cup X$ is D_K -independent for each $x \in W$. So each set X in the process is D_K -independent. It remains for us to show, that as long as $\tilde{d}(X) < A$, there is at least one

admissible vertex to expand X .

Let $X \subset V$ with $\tilde{d}(X) < A$ and $\tilde{d}(X) \geq (a - 2) \cdot e_K(X)$.

W is not empty:

We prove indirectly that $W \neq \emptyset$ by showing that (cf. Figure 3.2)

$$D_3 \setminus (\mathring{D}_3(\Gamma_K(X)) \cup D_3(X)) = \underbrace{\{e \in D_3 : e \setminus \Gamma_K(X) \neq \emptyset \text{ and } e \cap X = \emptyset\}}_{\subseteq D_3(W)} \neq \emptyset.$$

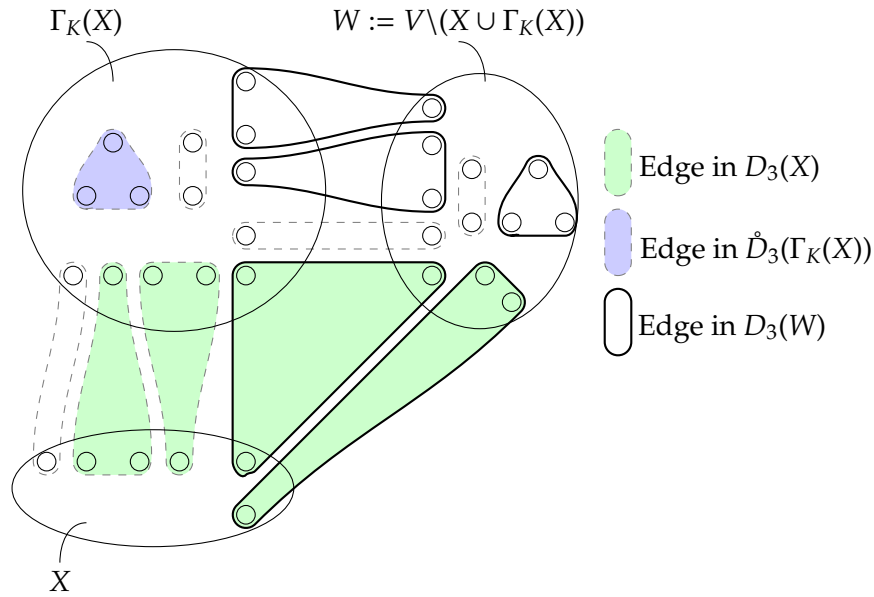


Figure 3.2: Sketch of possible edges between X , $\Gamma_K(X)$ and W

Since we assume that

$$A = \frac{1}{7} \cdot \binom{a-1}{2} > \tilde{d}(X) \geq (a-2) \cdot e_K(X),$$

it follows because of $e_K(X) \geq |\Gamma_K(X)|$ that $|\Gamma_K(X)| \leq \lfloor \frac{a-1}{14} \rfloor$ and therefore

$$\mathring{d}_3(\Gamma_K(X)) \leq \binom{|\Gamma_K(X)|}{3} \leq \binom{\lfloor \frac{a-1}{14} \rfloor}{3}. \quad (3.1)$$

Certainly, $d_3(X) \leq \tilde{d}(X)$ and thus

$$d_3(X) < \frac{1}{7} \cdot \binom{a-1}{2}. \quad (3.2)$$

We receive with (3.1) and (3.2)

$$\begin{aligned}
& |D_3 \setminus (\mathring{D}_3(\Gamma_K(X)) \cup D_3(X))| \\
& \geq d_3 - \mathring{d}_3(\Gamma_K(X)) - d_3(X) \\
& \stackrel{(3.1),(3.2)}{>} \binom{a-1}{3} - \binom{\lfloor \frac{a-1}{14} \rfloor}{3} - \frac{1}{7} \cdot \binom{a-1}{2} \\
& = \frac{2743}{16464} \cdot (a-1) \cdot \left(a - \underbrace{\frac{7426 - 7 \cdot \sqrt{96457}}{2743}}_{\approx 1.91} \right) \cdot \left(a - \underbrace{\frac{7426 + 7 \cdot \sqrt{96457}}{2743}}_{\approx 3.5} \right) \\
& \stackrel{a \geq 4}{>} 0.
\end{aligned}$$

For $a \leq 3$, we have $\binom{\lfloor \frac{a-1}{14} \rfloor}{3} = 0$ since $\lfloor \frac{a-1}{14} \rfloor \leq 3$, further is $\frac{1}{7} \cdot \binom{a-1}{2} < 1$. Since $d_3(X)$ is integers,

$$d_3 - \mathring{d}_3(\Gamma_K(X)) - d_3(X) = d_3 - 0 - 0 > 0.$$

That is, $D_3 \setminus (\mathring{D}_3(\Gamma_K(X)) \cup D_3(X))$ is in any case not empty, and thus, neither $D_3(W)$ nor W is empty.

There is at least one admissible $v \in W$ to expand X :

We use the pigeonhole principle to prove the existence of a vertex $v \in W$ with

$$\tilde{d}(\{v\} \cup X) - \tilde{d}(X) \geq (a-2) \cdot [e_K(\{v\} \cup X) - e_K(X)]. \quad (3.3)$$

Since $\tilde{d}(X) \geq (a-2) \cdot e_K(X)$, every $v \in W$ that satisfies inequality (3.3) satisfies also $\tilde{d}(\{v\} \cup X) \geq (a-2) \cdot e_K(\{v\} \cup X)$. We have already stated that for every $v \in W$, $\{v\} \cup X$ is D_K -independent and therefore $\mathring{d}_K(\{v\} \cup X) = 0$.

Now consider the right hand side of (3.3):

$$\begin{aligned}
& e_K(X \cup \{v\}) - e_K(X) \\
& = \sum_{u \in V} \mathring{d}_K(\{u, v\} \cup X) - \sum_{u \in V} \mathring{d}_K(\{u\} \cup X) \\
& = \sum_{u \in V} [\mathring{d}_K(\{u\} \cup X) + |\{e \in \mathring{D}_K(\{u, v\} \cup X) : v \in e\}|] - \sum_{u \in V} \mathring{d}_K(\{u\} \cup X) \\
& = d_2(\{v\}) + |\{e \in D_3(\{v\}) : e \cap X \neq \emptyset\}|
\end{aligned} \quad (3.4)$$

Adding up over all $v \in W$ this amounts to

$$\begin{aligned}
& \sum_{v \in W} [e_K(X \cup \{v\}) - e_K(X)] \\
& = \sum_{v \in W} [d_2(\{v\}) + |\{e \in D_3(\{v\}) : e \cap X \neq \emptyset\}|] \\
& \leq 2 \cdot d_3(X) + \sum_{v \in W} d_2(\{v\})
\end{aligned} \quad (3.5)$$

Now, take a look to the left hand side of (3.3). Please note that, since $\dot{d}_K(\{v\} \cup X) = 0$, of course also $\dot{d}_2(\{v\} \cup X) = 0$ and therefore $d_2(X \cup \{v\}) = d_2(X) + d_2(\{v\})$. Further, $d_3(X \cup \{v\}) = d_3(X) + |\{e \in D_3(\{v\}) : e \cap X = \emptyset\}|$, consequently

$$\begin{aligned} & \tilde{d}(X \cup \{v\}) - \tilde{d}(X) \\ &= (a-2) \cdot d_2(X \cup \{v\}) + d_3(X \cup \{v\}) - (a-2) \cdot d_2(X) - d_3(X) \\ &= (a-2) \cdot d_2(\{v\}) + |\{e \in D_3(\{v\}) : e \cap X = \emptyset\}|. \end{aligned}$$

Now we sum up $\tilde{d}(X \cup \{v\}) - \tilde{d}(X)$ for all $v \in W$ and receive

$$\begin{aligned} & \sum_{v \in W} [\tilde{d}(X \cup \{v\}) - \tilde{d}(X)] \\ &= \sum_{v \in W} [(a-2) \cdot d_2(\{v\}) + |\{e \in D_3(\{v\}) : e \cap X = \emptyset\}|] \\ &\geq |\{e \in D_3(W) : e \cap X = \emptyset\}| + (a-2) \cdot \sum_{v \in W} d_2(\{v\}) \\ &= |D_3 \setminus (\dot{D}_3(\Gamma_K(X)) \cup D_3(X))| + (a-2) \cdot \sum_{v \in W} d_2(\{v\}) \\ &> \binom{a-1}{3} - \binom{|\Gamma_K(X)|}{3} - d_3(X) + (a-2) \cdot \sum_{v \in W} d_2(\{v\}) \\ &\stackrel{(3.1), (3.2)}{>} \binom{a-1}{3} - \binom{\lfloor \frac{a-1}{14} \rfloor}{3} - \frac{1}{7} \cdot \binom{a-1}{2} + (a-2) \cdot \sum_{v \in W} d_2(\{v\}) \\ &= 2 \cdot (a-2) \cdot \frac{1}{7} \cdot \binom{a-1}{2} + \frac{391 \cdot (a-1)}{16464} \cdot \underbrace{\left(a - \frac{2722 - 7 \cdot \sqrt{79609}}{391}\right)}_{\approx 1.91} \cdot \underbrace{\left(a - \frac{2722 + 7 \cdot \sqrt{79609}}{391}\right)}_{\approx 12.01} \\ &\quad + (a-2) \cdot \sum_{v \in W} d_2(\{v\}) \\ &\stackrel{a \geq 13}{>} 2 \cdot (a-2) \cdot d_3(X) + (a-2) \cdot \sum_{v \in W} d_2(\{v\}) \\ &\geq (a-2) \cdot \left(2 \cdot d_3(X) + \sum_{v \in W} d_2(\{v\})\right) \\ &\stackrel{(3.5)}{\geq} (a-2) \cdot \sum_{v \in W} [e_K(X \cup \{v\}) - e_K(X)]. \end{aligned}$$

Accordingly, at least for $a \geq 13$, there is some $v \in W$ that satisfies (3.3) and therefore the algorithm terminates for $a \geq 13$ in case of $a \geq b$.

Suppose now $3 \leq a \leq 12$, then

$$\left\lfloor \frac{1}{7} \cdot \binom{a-1}{2} \right\rfloor \leq a-2.$$

Hence, for each vertex $v \in V$ with $d_2(\{v\}) > 0$, we have both

$$\tilde{d}(\{v\}) = (a-2) \cdot d_2(\{v\}) + d_3(\{v\}) \geq (a-2) \geq \left\lfloor \frac{1}{7} \cdot \binom{a-1}{2} \right\rfloor \text{ and}$$

$$\tilde{d}(\{v\}) = (a-2) \cdot d_2(\{v\}) + d_3(\{v\}) \geq (a-2) \cdot d_2(\{v\}) = (a-2) \cdot e_2(\{v\}) = (a-2) \cdot e_K(\{v\}).$$

So, if $a \geq b$, $3 \leq a \leq 12$ and $d_2 > 0$, the algorithm terminates.

Suppose now, $d_2 = 0$. Then we need to take a closer look at the case $3 \leq a \leq 12$: Since d_3 , $d_3(W)$, and $e_3(W)$ are integers for all $W \subset V$,

- $d_3 \geq \binom{a-1}{3} + 1$,
- $d_3(X) \leq \left\lceil \frac{1}{7} \cdot \binom{a-1}{2} \right\rceil - 1$ and,
- $\left\lfloor \frac{a-1}{14} \right\rfloor = 0$ for $a \leq 12$.

Hence,

$$|D_3 \setminus (\mathring{D}_3(\Gamma_K(X)) \cup D_3(X))| \geq \binom{a-1}{3} + 1 - \left(\frac{1}{7} \cdot \left\lceil \binom{a-1}{2} \right\rceil - 1 \right) - 0.$$

For $2 \leq a \leq 6$ and $8 \leq a \leq 12$ the inequality

$$\binom{a-1}{3} + 1 - \left(\left\lceil \frac{1}{7} \cdot \binom{a-1}{2} \right\rceil - 1 \right) - 0 \geq 2 \cdot (a-2) \cdot \left(\left\lceil \frac{1}{7} \cdot \binom{a-1}{2} \right\rceil - 1 \right)$$

is true. Also for $a = 7$ and $d_3 \geq 22 = \binom{7-1}{3} + 2$ we have

$$d_3 - \left(\left\lceil \frac{1}{7} \cdot \binom{7-1}{2} \right\rceil - 1 \right) - 0 \geq 2 \cdot (7-2) \cdot \left(\left\lceil \frac{1}{7} \cdot \binom{7-1}{2} \right\rceil - 1 \right).$$

Therefore, if $d_3 \neq 21$, it follows that

$$\sum_{v \in W} [\tilde{d}(X \cup \{v\}) - \tilde{d}(X)] \geq (a-2) \cdot \sum_{v \in W} [e_K(X \cup \{v\}) - e_K(X)]$$

and thus the algorithm terminates.

Finally, assume that $d_3 = 21$ and $d_2 = 0$. Then $a = 7$, $\left\lceil \frac{1}{7} \cdot \binom{7-1}{2} \right\rceil = 3$, and $a-2 = 5$.

- If $\max_{v \in V} d_3(\{v\}) \geq 3$, $\mathcal{A}_{a \geq b}$ chooses a vertex $x \in V$ with $d_3(X) \geq 3$, and $X = \{x\}$ satisfies the constraints of Lemma 3.2.
- In case of $\max_{v \in V} d_3(\{v\}) = 1$ all known defective edges are disjoint. $\mathcal{A}_{a \geq b}$ picks, thus, three vertices from three different known defective edges, say the algorithm picks v_1, v_2 and v_3 . We have of course $d_K(\{v_1, v_2, v_3\}) = 3$ and $e_K(\{v_1, v_2, v_3\}) = 0$, thus, $\{v_1, v_2, v_3\}$ obeys the requirements of the Lemma.
- Finally, suppose that $\max_{v \in V} d_3(\{v\}) = 2$. Let e_1 and e_2 be the two known defective edges that are incident to x . Then $e_K(\{x, y\}) \neq 0$ if and only if $y \in (e_1 \cup e_2) \setminus \{x\}$. Since $|(e_1 \cup e_2) \setminus \{x\}| \leq 4$ and $|V \setminus \{x\}| \geq 6$, there is at least one vertex $z \in V \setminus (e_1 \cup e_2)$ with $d_3(\{z\}) \geq 1$. Then, $\{x, z\}$ is a required set (it is $d_K(\{x, z\}) \geq 3$ and $e_K(\{x, z\}) = 0$).

Thus, $\mathcal{A}_{a \geq b}$ finds in any case a D_K -independent set $X \subset V$ with

$$\tilde{d}(X) \geq \frac{1}{7} \cdot \binom{a-1}{2} \text{ and } \tilde{d}(X) \geq (a-2) \cdot e_K(X).$$

According to the Pigeonhole Principle, $\tilde{d}(X) = (a-2) \cdot d_2(X) + d_3(X) \geq \frac{1}{7} \cdot \binom{a-1}{2}$ implies that

$$(a-2) \cdot d_2(X) \geq \frac{1}{14} \cdot \binom{a-1}{2} \Leftrightarrow d_2(X) \geq \frac{a-1}{28} \geq \frac{b-1}{28}$$

or

$$d_3(X) \geq \frac{1}{14} \cdot \binom{a-1}{2}.$$

Finally,

$$\begin{aligned} & \tilde{d}(X) \geq (a-2) \cdot e_K(X) \\ \Leftrightarrow & (a-2) \cdot d_2(X) + d_3(X) \geq (a-2) \cdot [e_2(X) + e_3(X)] \\ \Leftrightarrow & d_3(X) \geq (a-2) \cdot e_3(X) \\ \Leftrightarrow & \frac{d_3(X)}{a-2} \geq e_3(X), \end{aligned}$$

and all the more

$$e_3(X) \leq \frac{d_3(X)}{a-2} + d_2(X).$$

This concludes the case that $a \geq b$. Next, let us consider:

$b > a$:

Please note that, in case of $d_3 = 0$, we receive $a = 3$. Hence, if $b > a$, then $b \geq 4$.

Algorithm $\mathcal{A}_{a < b}$:

1. Choose a vertex $x \in V$ with $d_2(\{x\}) \geq 1$ and set $X := \{x\}$.
2. If $d_2(X) \geq \frac{b-1}{28}$ or $d_3(X) \geq \frac{1}{14} \cdot \binom{b-1}{2}$, stop. Else, choose a vertex $x \in W := V \setminus (X \cup \Gamma_K(X))$ with

$$2 \cdot d_2(X \cup \{x\}) \geq e_K(X \cup \{x\})$$

and set $X := X \cup \{x\}$. Repeat step 2.

As $\mathcal{A}_{a \geq b}$ before, the algorithm starts with a single vertex, which is of course D_K -independent. We obtain for all $x \in V$ that

$$2 \cdot d_2(\{x\}) = 2 \cdot e_2(\{x\}) = 2 \cdot e_K(\{x\}) \geq e_K(\{x\}).$$

Due to the choice of vertices that expand X , each set X in the process is D_K -independent and satisfies

$$2 \cdot d_2(X) \geq e_K(X).$$

It remains to show that, as long as

$$d_2(X) < \frac{b-1}{28} \text{ and } d_3(X) < \frac{1}{14} \cdot \binom{b-1}{2}, \quad (3.6)$$

there is at least one vertex $x \in W = V \setminus (X \cup \Gamma_K(X))$ with $2 \cdot d_2(X \cup \{x\}) \geq e_K(X \cup \{x\})$.

W is not empty:

In case of (3.6), we show again indirectly that, W is not empty by proving

$$D_2 \setminus [\mathring{D}_2(\Gamma_K(X)) \cup D_2(X)] \neq \emptyset.$$

Due to

$$\frac{b-1}{28} > d_2(X) \geq \frac{e_K(X)}{2} \geq \frac{|\Gamma_K(X)|}{2},$$

$$|\Gamma_K(X)| < \frac{b-1}{14} \text{ and thus } \mathring{d}_2(\Gamma_K(X)) \leq \binom{\frac{b-1}{14}}{2}.$$

Together we obtain

$$\begin{aligned} & \left| D_2 \setminus [\mathring{D}_2(\Gamma_K(X)) \cup D_2(X)] \right| \\ & > \binom{b-1}{2} - \binom{\frac{b-1}{14}}{2} - \frac{b-1}{28} \\ & = \frac{195}{392} \cdot \underbrace{\left(b - \frac{391}{195} \right)}_{\approx 2.005} \cdot (b-1) \\ & \stackrel{b \geq 4}{>} 0, \end{aligned}$$

and therefore W is not empty (cf. Figure 3.3).

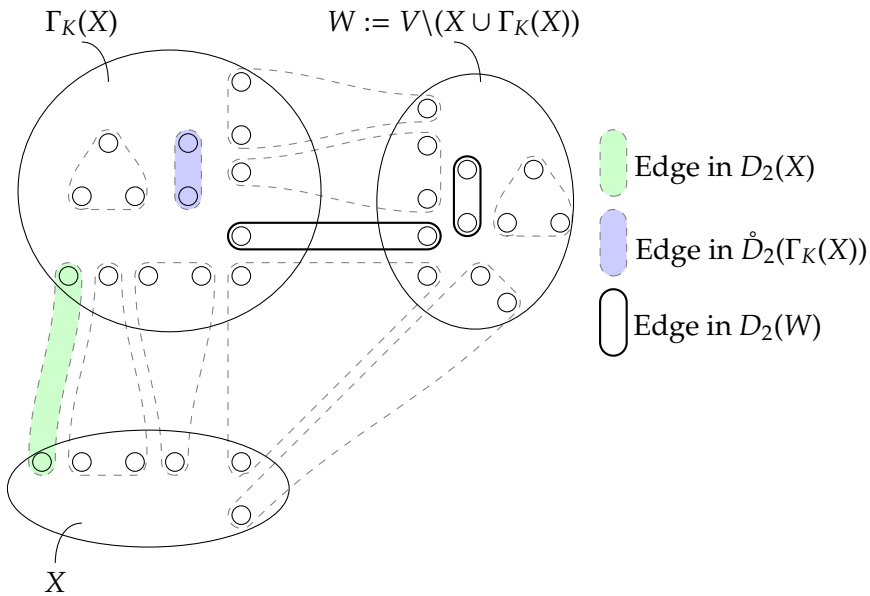


Figure 3.3: Sketch of possible edges between X , $\Gamma_K(X)$ and W

There is at least one admissible $v \in W$ to expand X :

Let now $X \subset V$ be some D_K -independent set with

$$d_2(X) < \frac{b-1}{28}, \quad d_3(X) < \frac{1}{14} \cdot \binom{b-1}{2}, \quad \text{and } 2 \cdot d_2(X) \geq e_K(X). \quad (3.7)$$

We show the existence of a vertex $x \in W$ with

$$2 \cdot d_2(X \cup \{x\}) \geq e_K(X \cup \{x\}) \quad (3.8)$$

by proving

$$2 \cdot \sum_{v \in W} [d_2(X \cup \{v\}) - d_2(X)] \geq \sum_{v \in W} [e_K(X \cup \{v\}) - e_K(X)]. \quad (3.9)$$

We have already seen (cf. (3.4)) that for arbitrary $v \in W$

$$e_K(X \cup \{v\}) - e_K(X) = d_2(\{v\}) + |\{e \in D_3(\{v\}) : e \cap X \neq \emptyset\}|.$$

Since $X \cup \{v\}$ is free, we have for any $v \in W$

$$d_2(X \cup \{v\}) - d_2(X) = d_2(\{v\}).$$

Summing up over $v \in W$, we receive on the right hand side

$$\begin{aligned} \sum_{v \in W} [e_K(X \cup \{v\}) - e_K(X)] &\stackrel{(3.3)}{=} \sum_{v \in W} [d_2(\{v\}) + |\{e \in D_3(\{v\}) : e \cap X \neq \emptyset\}|] \\ &\leq 2 \cdot d_3(X) + \sum_{v \in W} d_2(\{v\}) \\ &\stackrel{(3.7)}{<} \frac{1}{7} \cdot \binom{b-1}{2} + \sum_{v \in W} d_2(\{v\}) \end{aligned}$$

and on the left hand side

$$\begin{aligned} 2 \cdot \sum_{v \in W} [d_2(X \cup \{v\}) - d_2(X)] &= 2 \cdot \sum_{v \in W} d_2(\{v\}) \\ &\geq d_2(W) + \sum_{v \in W} d_2(\{v\}) \\ &> \binom{b-1}{2} - \binom{|\Gamma_K(X)|}{2} - d_2(X) + \sum_{v \in W} d_2(\{v\}) \\ &> \binom{b-1}{2} - \binom{\frac{b-1}{14}}{2} - \frac{b-1}{28} + \sum_{v \in W} d_2(\{v\}) \\ &= \frac{1}{7} \cdot \binom{b-1}{2} + \frac{167}{392} \cdot (b-1) \cdot \underbrace{\left(b - \frac{335}{167}\right)}_{\approx 2.006} + \sum_{v \in W} d_2(\{v\}) \\ &\stackrel{b \geq 4}{>} \frac{1}{7} \cdot \binom{b-1}{2} + \sum_{v \in W} d_2(\{v\}) \\ &> \sum_{v \in W} [e_K(X \cup \{v\}) - e_K(X)]. \end{aligned}$$

Hence, in case of (3.7), there is at least one $x \in W$ with

$$2 \cdot d_2(X \cup \{x\}) \geq e_K(X \cup \{x\}).$$

Consequently, the algorithm constructs a D_K -independent vertex set $X \subset V$ with

$$2 \cdot d_2(X \cup \{x\}) \geq e_K(X \cup \{x\}) \text{ and} \\ d_2(X) \geq \frac{b-1}{28} \text{ or } d_3(X) \geq \frac{1}{14} \cdot \binom{b-1}{2}.$$

In case of $b > a$ we obviously have

$$\frac{1}{14} \cdot \binom{b-1}{2} > \frac{1}{14} \cdot \binom{a-1}{2},$$

and since

$$\begin{aligned} 2 \cdot d_2(X) &\geq e_K(X) \\ \Leftrightarrow 2 \cdot d_2(X) &\geq e_2(X) + e_3(X) \\ \Leftrightarrow d_2(X) &\geq e_3(X), \end{aligned}$$

it applies

$$e_3(X) \leq \frac{d_3(X)}{a-2} + d_2(X),$$

which concludes our proof. \square

Let us return to 3-uniform hypergraphs.

Corollary 3.3. *Let $H = (V, E)$ be a 3-uniform hypergraph. Further let $C \subset V$ be a free set and $U \subset V \setminus (C \cup \Gamma(C))$ with $a, b \in \mathbb{N}$ such that*

$$\binom{a-1}{3} < \max\{1, \dot{d}_K(U)\} \leq \binom{a}{3} \text{ and } \binom{b-1}{2} < \max\{1, d_K(C, U)\} \leq \binom{b}{2}.$$

Then there is a set $X \subset U$ such that $X \cup C$ is D_K -independent, with

$$\begin{aligned} (*) : \quad e_K(X, U) &\leq \frac{d_K(X, U)}{a-2} + d_K(X, C, U \setminus X) \text{ and} \\ d_K(X, C, U \setminus X) &\geq \frac{b-1}{28} \quad \text{or} \quad d_K(X, U) \geq \frac{1}{14} \cdot \binom{a-1}{2}. \end{aligned}$$

Proof. Let $H_{(C,U)} = (U, E_{(C,U)})$ be the hypergraph on the vertex set U with the edge set

$$\begin{aligned} E_{(C,U)} &:= \{e \cap U : e \in E(U \cup C)\}, \\ D_2 &:= \{e \cap U : e \in D_K(C, U)\}, \text{ and} \\ D_3 &:= \dot{D}_K(U). \end{aligned}$$

Furthermore, define for any set $Y \subset U$, with $C \cup Y$ is D_K -independent, the following sets and numbers

$$\begin{aligned} D_2(Y) &:= \{e \in D_2 : e \cap Y \neq \emptyset\}, \\ D_3(Y) &:= D_K(Y, U), \\ e_2(Y) &:= d_2(Y) := |D_2(Y)|, \text{ and} \\ e_3(Y) &:= \sum_{v \in U} \dot{d}_K(\{v\} \cup Y). \end{aligned}$$

Please note that since $\Gamma(C) \cap U = \emptyset$, all edges in D_2 have cardinality 2. Let now $Y \subset U$ be any set such that $C \cup Y$ is D_K -independent, then

$$\begin{aligned} d_2(Y) &= |\{e \in D_2 : e \cap Y \neq \emptyset\}| \\ &= |\{e \cap U : e \in D_K(C, U), e \cap Y \neq \emptyset\}| \\ &= |D_K(Y, C, U \setminus Y)|, \end{aligned}$$

and

$$\begin{aligned} e_3(Y) &= \sum_{v \in U} \dot{d}_K(\{v\} \cup Y) \\ &= e_K(Y, U). \end{aligned}$$

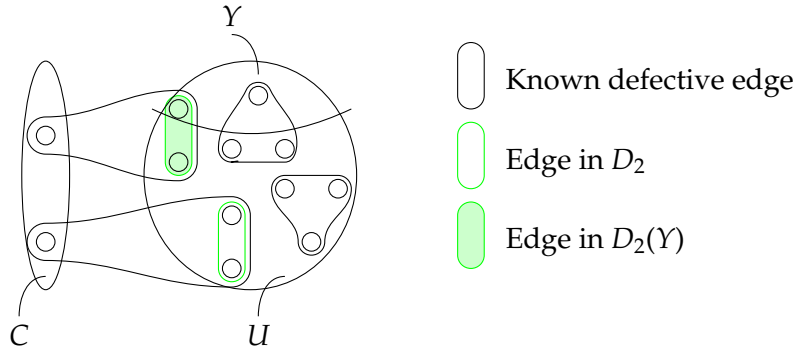


Figure 3.4: Sketch of $E_{(C,U)}$

Let $Y \subset U$ be D_K -independent in $H_{(C,U)}$.

CLAIM: Then $Y \cup C$ is D_K -independent in H .

Assume otherwise and let $e \in \dot{D}_K(Y \cup C)$. If $e \in \dot{D}_K(Y) \subset \dot{D}_K(U)$, then Y is not D_K -independent in $H_{(C,U)}$, which is a contradiction to our assumption. Thus $e \cap C \neq \emptyset$. Since C is free and $\Gamma(C) \cap Y = \emptyset$, it follows that $|e \cap C| = 1$ and $|e \cap Y| = 2$. Then $e \cap Y \in D_2$ which is again a contradiction. ■

Accordingly, Corollary 3.3 follows directly from Lemma 3.2. □

Lemma 3.4. *Let $H = (V, E)$ be a 3-uniform hypergraph. Further, let $C \subset V$ be a free vertex set and $U \subset V \setminus (C \cup \Gamma(C))$. Then there is a partition of U into r sets X_1, X_2, \dots, X_r such that*

- $r \leq 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(C, U)}$

- $C \cup X_i$ is D_K -independent for $i = 1, \dots, r$ and
- $\sum_{i=1}^r e_K(X_i, W_i) \leq 3 \cdot d^{\frac{2}{3}} + d_K(C, U)$,

where $W_i := U \setminus \bigcup_{n=1}^i X_n$.

Proof. Set $W_1 := U$ and $i := 1$.

1. Let $a_i, b_i \in \mathbb{N}$ with $\binom{a_i - 1}{3} < \max\{1, \dot{d}_K(W_i)\} \leq \binom{a_i}{3}$ and $\binom{b_i - 1}{2} < \max\{1, d_K(C, W_i)\} \leq \binom{b_i}{2}$.

Find a set $X_i \subseteq W_i$ such that

- $C \cup X_i$ is D_K -independent,
- $d_K(X_i, W_i) \geq \frac{1}{14} \cdot \binom{a_i - 1}{2}$ or $d_K(X_i, C, W_i) \geq \frac{b_i - 1}{28}$, and
- $e_K(X_i, W_i) \leq \frac{d_K(X_i, W_i)}{a_i - 2} + d_K(X_i, C, W_i)$.

2. Set $W_{i+1} := W_i \setminus X_i$. If $W_{i+1} = \emptyset$, set $r := i$ and stop, else, set $i := i + 1$ and repeat step 1.

Obviously X_1, X_2, \dots, X_r is a partition of U such that $X_i \cup C$ is D_K -independent for $1 \leq i \leq r$.

The number of sets:

Let us now consider r , the number of sets. It holds that

$$d_K \geq \dot{d}_K(U) = \dot{d}_K(W_1) = d_K(X_1, W_1) + \dot{d}_K(W_2) = \dots = d_K(X_1, W_1) + \dots + d_K(X_r, W_r)$$

and

$$d_K(C, U) = d_K(C, W_1) = d_K(X_1, C, W_1) + d_K(C, W_2) = \dots = d_K(X_1, C, W_1) + \dots + d_K(X_r, C, W_r).$$

Hence, $a_1 \geq a_2 \geq \dots \geq a_r \geq 3$ as well as $b_1 \geq b_2 \geq \dots \geq b_r \geq 2$. Set

$$I^a := \left\{ i : d_K(X_i, W_i) \geq \frac{1}{14} \cdot \binom{a_i - 1}{2} \right\}$$

and $I^b := \{1, 2, \dots, r\} \setminus I^a$, then

$$d_K(X_i, C, W_i) \geq \frac{b_i - 1}{28}$$

for each $i \in I^b$. Let $c, d \in \mathbb{N}$ with $|[c, d] \cap I^a| = 14$; then, according to Corollary 3.3,

$$\sum_{\substack{c \leq i \leq d : \\ i \in I^a}} d_K(X_i, W_i) \geq \binom{a_d - 1}{2}.$$

Hence,

$$\binom{a_c}{3} \geq \dot{d}(W_c)$$

$$\begin{aligned}
&= d(W_{d+1}) + \sum_{i=c}^d d_K(X_i, W_i) \\
&> \binom{a_{d+1}-1}{3} + \binom{a_d-1}{2} \\
&\geq \binom{a_{d+1}-1}{3} + \binom{a_{d+1}-1}{2} \\
&= \binom{a_{d+1}}{3}.
\end{aligned}$$

Consequently, $a_c \geq a_{d+1} + 1$ and thus $|I^a| \leq 14 \cdot a_1$. With the same argumentation, one receives $|I^b| \leq 28 \cdot b_1$. Thus,

$$r \leq 14 \cdot a_1 + 28 \cdot b_1.$$

Let us now put r in terms of d and $d_K(C, U)$. Due to $d \geq d_K \geq \binom{a_1-1}{3} + 1$ we know that

$$3 \cdot d^{\frac{1}{3}} \geq 3 \cdot \left(\binom{a_1-1}{3} + 1 \right)^{\frac{1}{3}} = \left(a_1^3 + \frac{7}{2} \cdot a_1 \cdot (a_1 - 3) \cdot \underbrace{\left(a_1 - \frac{33}{7} \right)}_{\approx 4.7} \right)^{\frac{1}{3}} \stackrel{a_1 \geq 5}{\geq} a_1,$$

and along with

$$2 \cdot \sqrt{d_K(C, U)} \geq 2 \cdot \left(\binom{b_1-1}{2} + 1 \right)^{\frac{1}{2}} = \left(b_1^2 + (b_1 - 2) \cdot (b_1 - 4) \right)^{\frac{1}{2}} \stackrel{b_1 \geq 4}{\geq} b_1,$$

we receive that

$$r \leq 14 \cdot a_1 + 28 \cdot b_1 \leq 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(C, U)}, \quad (3.10)$$

for at least all (a_1, b_1) with $a_1 \geq 5$ and $b_1 \geq 4$. Let now $a_1 \leq 4$, since $d_K(X_i, W_i) \geq 1$ for $i \in I_a$ with

$$|I_a| \leq d_K(W_1) \leq \binom{a_1}{3} \leq 4.$$

By the same argumentation $|I_b| \leq 3$ for $b_1 \leq 3$; consequently, (3.10) holds for any a_1 and b_1 .

$\sum_{i=1}^r \mathbf{e}_K(\mathbf{X}_i, \mathbf{W}_i)$:

Now, let us show that

$$\sum_{i=1}^r e_K(X_i, W_i) \leq 3 \cdot d^{\frac{2}{3}} + d_K(C, U).$$

Each set X_i satisfies $e_K(X_i, W_i) \leq \frac{d_K(X_i, W_i)}{a_i - 2} + d_K(X_i, C, W_i)$ for $i = 1, \dots, r$, hence,

$$\begin{aligned}
\sum_{i=1}^r e_K(X_i, W_i) &\leq \sum_{i=1}^r \left(\frac{d_K(X_i, W_i)}{a_i - 2} + d_K(X_i, C, W_i) \right) \\
&= d_K(C, U) + \sum_{i=1}^r \frac{d_K(X_i, W_i)}{a_i - 2}.
\end{aligned}$$

Let $I_k := \{n \mid a_n = k\}$ for $k = 3, \dots, a_1$ and set $f(k) := \sum_{n \in I_k} d_K(X_n, W_n)$, then

$$\sum_{i=1}^r \frac{d_K(X_i, W_i)}{a_i - 2} = \sum_{k=3}^{a_1} \sum_{n \in I_k} \frac{d_K(X_n, W_n)}{k - 2} = \sum_{k=3}^{a_1} \frac{f(k)}{k - 2}.$$

On the other hand, for $1 \leq k \leq a_1$ and $n = \max\{l \mid l \in I_k\}$,

$$\binom{k}{3} = \binom{a_n}{3} \geq \dot{d}_K(W_n) = \sum_{j=1}^n d_K(X_j, W_j) = \sum_{l=3}^k f(l)$$

and thus

$$\binom{k}{3} - \sum_{j=3}^{k-1} f(j) \geq f(k).$$

Altogether this amounts to

$$\begin{aligned} & \sum_{i=1}^r \frac{d_K(X_i, W_i)}{a_i - 2} \\ &= \sum_{k=3}^{a_1} \frac{f(k)}{k - 2} \\ &\leq \frac{1}{a_1 - 2} \cdot \left(\binom{a_1}{3} - \sum_{k=3}^{a_1-1} f(k) \right) + \sum_{k=3}^{a_1-1} \frac{f(k)}{k - 2} \\ &= \frac{1}{3} \cdot \binom{a_1}{2} + \sum_{k=3}^{a_1-1} \left(\frac{1}{k - 2} - \frac{1}{a_1 - 2} \right) \cdot f(k) \\ &\leq \frac{1}{3} \cdot \binom{a_1}{2} + \frac{1}{(a_1 - 3)(a_1 - 2)} \cdot \left(\binom{a_1 - 1}{3} - \sum_{k=3}^{a_1-2} f(k) \right) + \sum_{k=3}^{a_1-2} \left(\frac{1}{k - 2} - \frac{1}{a_1 - 2} \right) \cdot f(k) \\ &= \frac{1}{3} \cdot \binom{a_1}{2} + \frac{a_1 - 1}{6} + \sum_{k=3}^{a_1-2} \left(\frac{1}{k - 2} - \frac{1}{a_1 - 3} \right) \cdot f(k) \\ &\leq \frac{1}{3} \cdot \binom{a_1}{2} + \frac{a_1 - 1}{6} + \frac{1}{(a_1 - 4)(a_1 - 3)} \cdot \left(\binom{a_1 - 2}{3} - \sum_{k=3}^{a_1-3} f(k) \right) + \sum_{k=3}^{a_1-3} \left(\frac{1}{k - 2} - \frac{1}{a_1 - 3} \right) \cdot f(k) \\ &= \frac{1}{3} \cdot \binom{a_1}{2} + \frac{a_1 - 1}{6} + \frac{a_1 - 2}{6} + \sum_{k=3}^{a_1-3} \left(\frac{1}{k - 2} - \frac{1}{a_1 - 4} \right) \cdot f(k) \\ &\leq \dots \\ &\leq \frac{1}{3} \cdot \binom{a_1}{2} + \frac{a_1 - 1}{6} + \frac{a_1 - 2}{6} + \dots + \frac{3}{6} \\ &= \frac{1}{3} \cdot \binom{a_1}{2} + \frac{1}{6} \binom{a_1}{2} - \frac{3}{6} \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2} \left(\binom{a_1}{2} - 1 \right) \\
&\leq 3 \cdot d^{\frac{2}{3}}.
\end{aligned}$$

Consequently,

$$\sum_{i=1}^r e_K(X_i, W_i) \leq 3 \cdot d^{\frac{2}{3}} + d_K(C, U),$$

which concludes the proof. \square

Lemma 3.5. *Let $X \subset V$ be some D_K -independent set which is not free. Then finding a defective edge in $\mathring{D}(X)$ costs at most*

$$\log_2 \rho + 3$$

substantial tests.

Proof. If $|E(X)| \leq \rho$, finding a defective edge costs, according to Theorem 1.2, at most

$$\log_2 \rho + 2$$

tests. Hence, we assume that $|E(X)| > \rho$.

If there is a set $Y \subset X$ with $\rho/2 \leq E(Y) \leq \rho$, test Y and find, in case of $f(Y) = 1$, a defective edge in $\mathring{D}(Y)$. If $f(Y) = 0$, the test has not been substantial, otherwise

$$\log_2 \rho + 3$$

substantial tests are enough to find a defective edge.

If there is no set $Y \subset X$ with $\rho/2 \leq E(Y) \leq \rho$, choose some maximum set $Y \subset X$ such that $E(Y) \leq \rho/2$ and some $y \in X \setminus Y$, then $\{y\} \cup Y > \rho$. Test $\{y\} \cup Y$. If $f(\{y\} \cup Y) = 0$, we have found at least ρ good edges and the test has not been substantial. Therefore, assume that $f(\{y\} \cup Y) = 1$. Then, test also Y . In case of $f(Y) = 1$, we find a defective edge in $\mathring{D}(Y)$ by at most $\lceil \log_2 |E(Y)| \rceil + 2 \leq \log_2(\rho/2) + 2$ tests. Together with the first two tests this is at most

$$\log_2 \rho + 3$$

tests. Now, if $f(\{y\} \cup Y) = 1$ and $f(Y) = 0$, all edges in $E(Y \cup \{y\})$ are incident to y (we consider all good edges as deleted). So, by deleting y from every edge in $E(\{y\} \cup Y)$, we receive a graph. Due to Lemma 1.4, there is a set $Z \subset Y$ such that $\rho/2 \leq E(\{y\} \cup Z) \leq \rho$. Then test $\{y\} \cup Z$ and find, in case of $f(\{y\} \cup Z) = 1$, a defective edge which needs at most $\log_2 \rho + 1$ tests. This amounts to at most

$$2 + \log_2 \rho + 1$$

tests. Otherwise, if the test has been negative, the test has not been substantial.

By repeating the above procedure after each negative, non-substantial test, we eventually find a defective edge by at most

$$\log_2 \rho + 3$$

substantial tests. \square

Lemma 3.6. *Let $C \subset V$ be a free vertex set and $U \subset V \setminus (C \cup \Gamma(C))$. Then there is an algorithm that partitions U into q sets G_1, G_2, \dots, G_q such that*

- $q \leq 43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(C, U)}$,
- each set $C \cup G_j$ is free for $j = 1, \dots, q$, and
- $\sum_{j=1}^q e_K(G_j, U \setminus \bigcup_{n=1}^j G_n) \leq 3 \cdot d^{\frac{2}{3}} + d_K(C, U)$.

Further, the algorithm needs at most

$$42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(C, U)} + d_\Delta \cdot (\log_2 \rho + 60)$$

substantial tests, where $d_K(C, U)$ denotes the number of defective edges in $D(C, U)$ that are known at the end of the algorithm, and d_Δ denotes the number of defective edges that are found while performing the algorithm.

Proof. The following algorithm satisfies the constraints of the Lemma.

Set $\omega, j := 1$ and let $d_0 := d_K$ and $d_0(C, U) := d_K(C, U)$.

1. Partition U into r_ω sets $X_1, X_2, \dots, X_{r_\omega}$ such that

- $r_\omega \leq 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_{\omega-1}(C, U)}$,
- each set $C \cup X_i$ is D_K -independent for $i = 1, \dots, r_\omega$, and
- $\sum_{i=1}^{r_\omega} e_K(X_i, W_i) \leq 3 \cdot d^{\frac{2}{3}} + d_{\omega-1}(C, U)$, where $W_i := U \setminus \bigcup_{n=1}^i X_n$.

Set $i := 1$.

2. Test $C \cup X_i$ if

$f(C \cup X_i) = 0$: Set $G_j := X_i$, $i := i + 1$, and $j := j + 1$, then go to step 3.

$f(C \cup X_i) = 1$: Find a defective edge $e \in \mathring{D}(C \cup X_i)$ by Triesch's search, choose an incident vertex $x \in (e \cap X_i)$, and set $G_j := \{x\}$, $X_i := X_i \setminus \{x\}$ as well as $j := j + 1$. If $X_i \neq \emptyset$, repeat step 2, else, go to step 3.

3. If $i < r_\omega$, set $i := i + 1$ and repeat step 2, else, go to step \square .

\square If $j > 43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(C, U)}$, set $d_\omega := d_K$, $d_\omega(C, U) := d_K(C, U)$, $\omega := \omega + 1$, and $j := 1$, then repeat starting at step 1, else, set $q := j$, $V^j := V \setminus \bigcup_{n=1}^j G_n$ for $j = 1, \dots, q - 1$, $\Omega := \omega$, and $d_\Omega := d_K$, then, stop.

Suppose that for some ω all tests in step 2 are negative. When the algorithm enters step \square , there are r_ω sets G_j . Since $r_\omega < 43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(C, U)}$, the algorithm stops at that point. Accordingly, the algorithm terminates at the latest when all defective edges are known.

Certainly, when the algorithm stops, there are at most $43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(C, U)}$ sets G_j . For each set G_j with $|G_j| \geq 2$ there has been a negative test of the set $G_j \cup C$ and thus $G_j \cup C$ is free. Since

$U \cap \Gamma(C) = \emptyset$, we have $\dot{d}(\{v\} \cup C) = 0$ for all $v \in U$. Consequently, also all sets $G_j \cup C$ with $|G_j| = 1$ are free. Let now $\underline{i} := \min\{j : G_j \subseteq X_i\}$ and $\bar{i} := \max\{j : G_j \subseteq X_i\}$, then

$$X_i = G_{\underline{i}} \dot{\cup} \dots \dot{\cup} G_{\bar{i}}$$

with $|G_j| = 1$ for $\underline{i} \leq j < \bar{i}$. Since $e_K(\{x\}) = 0$ for any $x \in V$ and since $W_i = V^{\bar{i}}$, we have

$$\sum_{j=\underline{i}}^{\bar{i}} e_K(G_j, V^j) = e_K(G_{\bar{i}}, V^{\bar{i}}) = e_K(G_{\bar{i}}, W_i) \leq e_K(X_i, W_i)$$

and thus

$$\sum_{j=1}^q e_K(G_j, V^j) \leq \sum_{i=1}^{r_\Omega} e_K(X_i, W_i) \leq 3 \cdot d^{\frac{2}{3}} + d_K(C, U).$$

That is, the above procedure provides indeed a desired partition and only the number of tests remains to be examined.

The algorithm performs all tests in step 2, namely, each set X_i is tested and after a positive test, it finds a defective edge which costs, according to Lemma 3.5, at most

$$\log_2 \rho + 3$$

substantial tests. After finding a defective edge, the algorithm removes one vertex from X_i before the test of X_i is repeated. Let d_Δ denote the number of defective edges that are detected while performing the algorithm. Regardless of the outcome of the test, a new set G_j is defined. That is, we have one test for each set G_j plus

$$d_\Delta \cdot (\log_2 \rho + 3)$$

tests. Now, how many sets G_j are there? There is one set G_j for each X_i and one set for each defective edge that was found in the current iteration. These are

$$r_\omega + d_\omega - d_{\omega-1} \leq 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_{\omega-1}(C, U)} + d_\omega - d_{\omega-1}$$

sets / tests for $\omega = 1, \dots, \Omega$. The algorithm iterates if this number exceeds $43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(C, U)}$. Thus, we have

$$d_\omega - d_{\omega-1} > d^{\frac{1}{3}} + \sqrt{d_{\omega-1}(C, U)}. \quad (3.11)$$

for $\omega = 1, \dots, \Omega - 1$. Please note that

$$\sum_{\omega=1}^{\Omega} (d_\omega - d_{\omega-1}) = d_\Omega - d_0 = d_\Delta. \quad (3.12)$$

Hence, there are at most

$$\begin{aligned} & d_\Delta \cdot (\log_2 \rho + 3) + \sum_{\omega=1}^{\Omega} \left(42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_{\omega-1}(C, U)} + d_\omega - d_{\omega-1} \right) \\ & \stackrel{(3.12)}{=} d_\Delta \cdot (\log_2 \rho + 3) + d_\Delta + 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_\Omega(C, U)} + \sum_{\omega=1}^{\Omega-1} \left(42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_{\omega-1}(C, U)} \right) \end{aligned}$$

$$\begin{aligned}
&\leq d_\Delta \cdot (\log_2 \rho + 4) + 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_\Omega(C, U)} + 56 \cdot \sum_{\omega=1}^{\Omega-1} (d^{\frac{1}{3}} + \sqrt{d_{\omega-1}(C, U)}) \\
&\stackrel{(3.11)}{\leq} d_\Delta \cdot (\log_2 \rho + 4) + 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_\Omega(C, U)} + 56 \cdot \sum_{\omega=1}^{\Omega-1} (d_\omega - d_{\omega-1}) \\
&\stackrel{(3.12)}{=} d_\Delta \cdot (\log_2 \rho + 60) + 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_\Omega(C, U)}
\end{aligned}$$

substantial tests. So, the above algorithm provides a desired partition of U in the course of the permitted number of tests. \square

For any free set $C \subset V$ Lemma 3.6 works only on subsets of $V \setminus [C \cup \Gamma(C)]$. Therefore, we have to find all vertices in $\Gamma(C, W)$ to provide a feasible set $U \subset W$. The next algorithm finds all those vertices for any free set $C \subset V$ and proceeds thereby in much the same way as the previous algorithms do.

Lemma 3.7. *Let $C \subset V$ be a free vertex set and $U \subset V \setminus (C \cup \Gamma_K(C))$. Then there exists a search algorithm that identifies all vertices in $\Gamma(C, U)$ and needs at most*

$$42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(C, U)} + d_\Delta \cdot (\log_2 \rho + 5)$$

substantial tests, where $d_K(C, U)$ denotes the number of defective edges in $D(C, U)$ that are known at the end of the algorithm, and d_Δ denotes the number of defective edges that are found while performing the algorithm.

Proof. The following algorithm satisfies the constraints of the Lemma.

Set $W_1 := U$ and $i := 1$.

1. Partition U into r sets X_1, X_2, \dots, X_r such that

- $r \leq 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(C, U)}$ and
- each set $C \cup X_i$ is D_K -independent for $i = 1, \dots, r$.

Set $i := 1$.

2. Test $C \cup X_i$ if

$f(C \cup X_i) = 0$: If $i < r$, set $i := i + 1$ and repeat step 2, else, stop.

$f(C \cup X_i) = 1$: Find a defective edge $e \in \mathring{D}(C \cup X_i)$. If

$|e \cap X_i| = 1$: Set $X_i := X_i \setminus e$ and repeat step 2.

$|e \cap X_i| \geq 2$: Choose a vertex $x \in (e \cap X_i)$, test $\{x\} \cup C$, set $X_i := X_i \setminus \{x\}$, and repeat step 2.

For each $x \in U$ one of the following three cases is true:

- $f(X_i \cup C) = 0$ for some X_i with $x \in X_i \Rightarrow x \notin \Gamma(C, U)$,
- there is an edge $e \in \mathring{D}(\{x\} \cup C)$ that has been found in step 2 $\Rightarrow x \in \Gamma(C, U)$, or
- the algorithm tests $\{x\} \cup C \Rightarrow$ If $f(\{x\} \cup C) = 1$, then $x \in \Gamma(C, U)$, else $x \notin \Gamma(C, U)$.

$\Gamma(C, U) \subset U$ and since we know for each $x \in U$ whether $x \in \Gamma(C, U)$ or not, all vertices of $\Gamma(C, U)$ are known.

By the same argumentation as in the previous Lemma the algorithm needs at most

$$42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(C, U)}$$

substantial tests plus

$$\log_2 \rho + 5$$

further substantial tests for each defective edge which is found while performing the above algorithm. \square

Lemma 3.8. *Let U be a free vertex set and $x \notin U$, with $d_{xU} := \dot{d}_K(\{x\} \cup U) > 0$. Then there is an algorithm that either finds all unknown defective edges in $\dot{D}(\{x\} \cup U)$ or proves that there are no further defective edges in at most*

$$3 \cdot d_{xU} + d_\Delta \cdot (\log_2 \rho + 4)$$

substantial tests, where d_Δ denotes the number of defective edges that are found while performing the algorithm.

Proof. Since U is free, all edges in $E(\{x\} \cup U)$ are incident to x . Deleting x from every edge in $E(\{x\} \cup U)$ induces a graph on U . We thus use the graph algorithm described in [KT08] to find all unknown defective edges in $\dot{D}(\{x\} \cup U)$. First of all, let us repeat the algorithm:

Let V be alphabetically sorted. Set $W := V$ and $X := \emptyset$.

1. $E(W) \geq \rho/2$: Construct a selectable subset $Y \subset W$ with $\rho/2 \leq |E(Y)| \leq \rho$.
 $E(W) < \rho/2$: Set $Y := W$.
2. Test Y . If
 $f(Y) = 1$: Find a defective edge $\{u, y\}$ ($u > y$) with rightmost left endvertex. Set $W := W \setminus \{u\}$ and $X := X \cup \{u\}$. Then reorder W with respect to the new edge set such that the vertices that lie right of u form also a rightmost set due to the new ordering. Delete all good edges and repeat step 1.
 $f(Y) = 0$: If $E(W) \neq \emptyset$, repeat step 1; else, set $\bar{X} := X$ and go to step 3.
3. Choose a $u \in \bar{X}$, set $\bar{X} := \bar{X} \setminus \{u\}$ and $\bar{W} := W \setminus \{w \in W : \{u, w\} \in D_K\}$.
4. Choose a maximum set $Y \subset \bar{W}$ such that $|E(\{u\} \cup Y)| \leq \rho$.
5. Test $\{u\} \cup Y$. If
 $f(\{u\} \cup Y) = 0$: Remove the good edges and set $\bar{W} := \bar{W} \setminus Y$.
 $f(\{u\} \cup Y) = 1$: Find a defective edge $\{u, z\} \in \dot{D}(\{u\} \cup Y)$ and set $\bar{W} := \bar{W} \setminus \{z\}$.
Then, if $\bar{W} \neq \emptyset$: Repeat step 4.
 $\bar{W} = \emptyset$: $\bar{X} \neq \emptyset$: Repeat step 3.
 $\bar{X} = \emptyset$: test X . $f(X) = 1$: Set $W := X$ and repeat step 1.
 $f(X) = 0$: Stop.

To avoid redundant tests, we proceed as follows. Whenever the graph algorithm constructs a test set Y in step 1 such that $\{x\} \cup Y$ is not D_K -independent, we replace Y by $Y' := \{y \in Y : z > y\}$, where z is a leftmost vertex such that $\{x\} \cup \{y \in Y : z > y\}$ is D_K -independent. (Considering the graph on U that emerges by deleting x from each edge in $E(\{x\} \cup U)$, z is then the rightmost left endvertex of all known defective edges in Y .) Afterwards, proceed with Y' instead of Y with step 2. Do as the algorithm prescribes and if $f(Y') = 0$, set also $X := X \cup \{z\}$. Let d_Δ denote the number of defective edges that are detected by the adapted graph algorithm. Following the argumentation of [KT08], this causes at most

$$\begin{aligned} & (d_{xU} + d_\Delta) + d_\Delta \cdot (\log_2 \rho + 1) + (d_{xU} + d_\Delta) + \sqrt{2(d_{xU} + d_\Delta) + 1/4} - 1/2 \\ & \leq d_\Delta \cdot (\log_2 \rho + 4) + 3 \cdot d_{xU} \end{aligned}$$

substantial tests. Please note, that we only count substantial tests and, hence, the number of tests that identify at least $\rho/2$ good edges are not considered in the above estimation. \square

Lemma 3.9. *Let $C \subset V$ be a free vertex set and $x, y \in V \setminus C$ such that both sets $\{x\} \cup C$ and $\{y\} \cup C$ are also free. Then there is an algorithm that finds all unknown defective edges in $\mathring{D}(\{x, y\} \cup C)$ by at most*

$$1 + d_\Delta(\log_2 \rho + 1)$$

substantial tests, where d_Δ denotes the number of defective edges that are found while performing the algorithm.

Proof. The following algorithm finds all defective edges:

1. Test $\mathcal{T} := \{x, y\} \cup [C \setminus \Gamma_K(\{x, y\})]$. If
 $f(\mathcal{T}) = 0$: stop.
 $f(\mathcal{T}) = 1$: find a defective edge in $\mathring{D}(\mathcal{T})$ and repeat step 1.

Both sets $\{x\} \cup C$ and $\{y\} \cup C$ are free, thus $\{x, y\} \subset e$ for each $e \in \mathring{D}(\{x, y\} \cup C)$ and therefore $\mathring{D}(\{x, y\} \cup C) = \mathring{D}(\{x, y\} \cup \Gamma[\{x, y\}, C])$. Now, each unknown defective edge lies in $\mathring{D}(\{x, y\} \cup [C \setminus \Gamma_K(\{x, y\})])$ and can be found by a halving procedure by at most

$$\log_2 \rho$$

tests. With one final test, which terminates the algorithm, the Lemma is proved. \square

Lemma 3.10. *Let $C \subset V$ be a free vertex set and $U \subset V \setminus C$ such that $\Gamma(C, U) = \Gamma_K(C, U)$ and $e(C, U) = e_K(C, U)$ holds. In other words, all defective edges $e \in D(C, U)$ with $|e \cap C| = 2$ are known. Then there is an algorithm that finds all unknown defective edges in $\mathring{D}(U \cup \Gamma(C, U))$ with $|e \cap \Gamma(C, U)| = 2$ by at most*

$$3 \cdot e(C, U)^2 + d_\Delta \cdot (\log_2 \rho + 2)$$

substantial tests, where d_Δ denotes the number of defective edges that are found while performing the algorithm.

It seems reasonable to test each vertex pair $\{x, y\} \subset \Gamma(C, U)$ with C . Since $\{x\} \cup C$ is certainly not free if $x \in \Gamma(C, U)$, we cannot use Lemma 3.9 on $\{x, y\} \cup C$. Thus, before we prove Lemma 3.10, we present a greedy algorithm that partitions C for each pair $\{x, y\} \subset \Gamma(C, U)$ into disjoint sets G_1, G_2, \dots, G_s such that x, y and G_i satisfy the constraints of Lemma 3.9 for $i = 1, \dots, s$.

Lemma 3.11. *Let $C \subset V$ be a free vertex set and $x, y \in V \setminus C$ such that all edges in both $\mathring{D}(\{x\} \cup C)$ and $\mathring{D}(\{y\} \cup C)$ are known. Then there is an algorithm that partitions $C \setminus \Gamma_K(\{x, y\})$ into*

$$s \leq \mathring{d}(\{x\} \cup C) + \mathring{d}(\{y\} \cup C) + 1$$

sets G_1, G_2, \dots, G_s such that for $1 \leq i \leq s$ both sets $\{x\} \cup G_i$ and $\{y\} \cup G_i$ are free.

Proof. The following greedy procedure satisfies the constraints of the lemma: Set $G_i := \emptyset$ for $i = 1, \dots, \mathring{d}(\{x\} \cup C) + \mathring{d}(\{y\} \cup C) + 1$ and $i := 1$.

1. As long as possible, choose a vertex $z \in C \setminus \Gamma_K(\{x, y\})$ such that $\{x, y, z\} \cup G_i$ is D_K -independent and set $G_i := G_i \cup \{z\}$ as well as $C := C \setminus \{z\}$.
2. If $C \neq \emptyset$, set $i := i + 1$ and repeat step 2. Otherwise, set $s := i$ and stop.

Let $z \in \bigcup_{j=i+1}^s G_j$ for some $1 \leq i < s$, then $\mathring{D}_K(\{x, y, z\} \cup G_i) \neq \emptyset$. Thus, there is at least one known defective edge in $\mathring{D}_K(\{x, y\} \cup G_i \cup G_j)$ for each pair $1 \leq i < j \leq s$. Since each edge in $\mathring{D}_K(\{x, y\} \cup G_i \cup G_j)$ is incident to x or y , there is one particular edge in $\mathring{D}_K(\{x, y\} \cup G_i \cup G_j)$ for each pair $1 \leq i < j \leq s$. Hence,

$$\begin{aligned} \binom{s}{2} &\leq \mathring{d}(\{x\} \cup C) + \mathring{d}(\{y\} \cup C) \\ \Rightarrow s &\leq \mathring{d}(\{x\} \cup C) + \mathring{d}(\{y\} \cup C) + 1. \quad \square \end{aligned}$$

Corollary 3.12. *Let $C \subset V$ be a free vertex set and $x \in V \setminus C$ such that all edges in $\mathring{D}(\{x\} \cup C)$ are known. There is an algorithm that partitions C into*

$$r \leq 2\mathring{d}(\{x\} \cup C) + 1$$

sets F_1, F_2, \dots, F_r such that $\{x\} \cup F_i$ is free for $i = 1, \dots, r$.

Proof. Using Lemma 3.11 for $y = x$, we receive

$$r \leq \mathring{d}(\{x\} \cup C) + \mathring{d}(\{x\} \cup C) + 1. \quad \square$$

Proof of Lemma 3.10. The following algorithm finds all unknown defective edges $e \in \mathring{D}(U \cup \Gamma(C, U))$ with $|e \cap \Gamma(C, U)| = 2$:

$$\text{Set } \mathcal{X} := \binom{\Gamma(C, U)}{2}.$$

1. Choose a vertex pair $\{x, y\} \in \mathcal{X}$ and set $\mathcal{X} := \mathcal{X} \setminus \{\{x, y\}\}$.
2. Partition $C \setminus \Gamma_K(\{x, y\})$ into s_{xy} sets $G_1^{xy}, G_2^{xy}, \dots, G_{s_{xy}}^{xy}$ such that both sets $\{x\} \cup G_i^{xy}$ and $\{y\} \cup G_i^{xy}$ are free for $i = 1, \dots, s_{xy}$.
3. Test $\mathcal{T} := \{x, y\} \cup G_i^{xy}$ and find, in case of $f(\mathcal{T}) = 1$, all edges in $\mathring{D}(\{x, y\} \cup G_i^{xy})$ by Lemma 3.9.
 If
 $i < s_{xy}$: Set $i := i + 1$ and repeat step 3.
 $i = s_{xy}$: $\mathcal{X} \neq \emptyset$: Repeat from step 1.
 $\mathcal{X} = \emptyset$: Stop.

For all $\{x, y\} \in \binom{\Gamma(C, U)}{2}$ there are s_{xy} different tests

$$\mathcal{T} := \{x, y\} \cup G_i^{xy},$$

$i = 1, \dots, s_{xy}$, which are certainly all D_K -independent. In case of $f(\{x, y\} \cup G_i^{xy}) = 1$ the algorithm finds, according to Lemma 3.9, all defective edges in $\mathring{D}(\{x, y\} \cup G_i^{xy})$ by at most

$$\mathring{d}(\{x, y\} \cup G_i^{xy}) \cdot (\log_2 \rho + 1) + 1$$

substantial tests. Knowing that $e(C, U) \geq \Gamma(C, U)$, this amounts to

$$\begin{aligned} & \sum_{\{x, y\} \in \binom{\Gamma(C, U)}{2}} \sum_{i=1}^{s_{xy}} \left(1 + \mathring{d}(\{x, y\} \cup G_i^{xy}) \cdot (\log_2 \rho + 1) + 1\right) \\ &= d_\Delta \cdot (\log_2 \rho + 1) + \sum_{\{x, y\} \in \binom{\Gamma(C, U)}{2}} 2 \cdot s_{xy} \\ &\stackrel{L 3.11}{\leq} d_\Delta \cdot (\log_2 \rho + 1) + 2 \cdot \sum_{\{x, y\} \in \binom{\Gamma(C, U)}{2}} \left(\mathring{d}(\{x\} \cup C) + \mathring{d}(\{y\} \cup C) + 1\right) \\ &\leq d_\Delta \cdot (\log_2 \rho + 1) + 2 \cdot \left| \binom{\Gamma(C, U)}{2} \right| + 2 \cdot (|\Gamma(C, U)| - 1) \cdot \sum_{x \in \Gamma(C, U)} \mathring{d}(\{x\} \cup C) \\ &\leq d_\Delta \cdot (\log_2 \rho + 1) + e(C, U)^2 + 2 \cdot (|\Gamma(C, U)| - 1) \cdot e(C, U) \\ &\leq d_\Delta \cdot (\log_2 \rho + 1) + 3 \cdot e(C, U)^2 \end{aligned}$$

substantial tests. □

Lemma 3.13. *Let C_1, C_2, \dots, C_p be a partition of V such that*

- $p \leq 43 \cdot d^{\frac{1}{3}}$,
- $e_K(C_i, V^i) = e(C_i, V^i)$, and
- $\sum_{i=1}^p e(C_i, V^i) \leq 4 \cdot d^{\frac{2}{3}}$, where $V^i := V \setminus \bigcup_{j=1}^i C_j$.

Then we can refine the partition into $\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{\bar{p}}$ such that

- $\bar{p} \leq 44 \cdot d^{\frac{1}{3}}$,
- $e(\bar{C}_i, \bar{V}^i) \leq 4 \cdot d^{\frac{1}{3}}$, for $i = 1, \dots, \bar{p}$,
- $e_K(\bar{C}_i, \bar{V}^i) = e(\bar{C}_i, \bar{V}^i)$ for $i = 1, \dots, \bar{p}$, and finally
- $\sum_{i=1}^{\bar{p}} e(\bar{C}_i, \bar{V}^i) \leq 4 \cdot d^{\frac{2}{3}}$, where $\bar{V}^i := V \setminus \bigcup_{j=1}^i \bar{C}_j$.

Proof. Set $\bar{C}_j = \emptyset$ for $j = 1, \dots, |V|$ and $i, j := 1$.

1. If $e(C_i, V^i) > 4 \cdot d^{\frac{1}{3}}$, go to step 2. Otherwise, set $\bar{C}_j := C_j$. If $i < p$, set both $i := i + 1$ and $j := j + 1$, then repeat step 1. Else, that is, if $i = p$, set $\bar{p} := j$ and $\bar{V}^i := V \setminus \bigcup_{j=1}^i \bar{C}_j$ for $i = 1, \dots, \bar{p}$, and stop.
2. If there is a vertex $x \in C_i$ with $e(\bar{C}_j \cup \{x\}, V^i) \leq 4 \cdot d^{\frac{1}{3}}$, set $\bar{C}_j := \bar{C}_j \cup \{x\}$, $C_i := C_i \setminus \{x\}$ and repeat step 2. Otherwise, if
 - $C_i \neq \emptyset$: Set $j := j + 1$ and repeat step 2.
 - $C_i = \emptyset$: $i < p$: Set $j := j + 1$, $i := i + 1$ and repeat step 1.
 - $i = p$: Set $\bar{p} := j$ and $\bar{V}^i := V \setminus \bigcup_{j=1}^i \bar{C}_j$ for $i = 1, \dots, \bar{p}$, then stop.

Since $e(\{x\}) = 0$ for any $x \in V$, the algorithm definitely terminates and gives us a partition of V into free sets $\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{\bar{p}}$ such that $e(\bar{C}_i, \bar{V}^i) \leq 4 \cdot d^{\frac{1}{3}}$, for $i = 1, \dots, \bar{p}$.

It remains to be shown that those sets satisfy the constraints of Lemma 3.13. Therefore set $g(i) := \min\{j : \bar{C}_j \subseteq C_i\}$ and $h(i) := \max\{j : \bar{C}_j \subseteq C_i\}$ for $i = 1, \dots, p \leq |V|$. Then,

$$\bigcup_{j=g(i)}^{h(i)} \bar{C}_j = C_i, \text{ for } i = 1, \dots, p.$$

Let $g(i) \leq j \leq h(i)$ for some $1 \leq i \leq p$, then

$$\bar{V}^j = V \setminus \bigcup_{l=1}^j \bar{C}_l = V \setminus \left(\bigcup_{l=1}^{i-1} C_l \cup \bigcup_{l=g(i)}^j \bar{C}_l \right) = \left(V \setminus \bigcup_{l=1}^i C_l \right) \cup \bigcup_{l=j+1}^{h(i)} \bar{C}_l = V^i \cup \bigcup_{l=j+1}^{h(i)} \bar{C}_l.$$

Since $\bar{C}_j \cup \bigcup_{l=j+1}^{h(i)} \bar{C}_l$ is free,

$$\sum_{v \in \bigcup_{l=j+1}^{h(i)} \bar{C}_l} \hat{d}(\{v\} \cup \bar{C}_j) = 0,$$

and thus $e(\bar{C}_j, \bar{V}^j) = e(\bar{C}_j, V^i)$. Moreover, from $\hat{D}(\{v\} \cup \bar{C}_j) \subseteq \hat{D}(\{v\} \cup C_i)$ and $\hat{D}_K(\{v\} \cup C_i) = \hat{D}(\{v\} \cup C_i)$ it follows that

$$e_K(\bar{C}_j, \bar{V}^j) = e(\bar{C}_j, V^i).$$

Consequently,

$$\begin{aligned} e(C_i, V^i) &= e\left(\bigcup_{j=g(i)}^{h(i)} \bar{C}_j, V^i\right) \\ &= \sum_{v \in V^i} \hat{d}\left(\{v\} \cup \bigcup_{j=g(i)}^{h(i)} \bar{C}_j\right) \\ &\geq \sum_{v \in V^i} \sum_{j=g(i)}^{h(i)} \hat{d}(\{v\} \cup \bar{C}_j) \\ &= \sum_{j=g(i)}^{h(i)} \sum_{v \in \bar{V}^j} \hat{d}(\{v\} \cup \bar{C}_j) \end{aligned}$$

$$= \sum_{j=g(i)}^{h(i)} e(\bar{C}_j, \bar{V}^j)$$

for $i = 1, \dots, p$ and thus

$$\sum_{i=1}^p e(C_i, V^i) \geq \sum_{j=1}^{\bar{p}} e(\bar{C}_j, \bar{V}^j).$$

Finally, we count the number of sets. If $g(i) < h(i)$ for some i , then

$$e(\bar{C}_j \cup \{x\}) > 4 \cdot d^{\frac{1}{3}} \text{ for } g(i) \leq j < h(i) \text{ and each } x \in \bar{C}_{h(i)}.$$

Let now $i \in \{1, \dots, p\}$ with $g(i) < h(i)$, further let $x \in \bar{C}_{h(i)}$ and $v \in \Gamma(C_i, V^i)$, then

$$\begin{aligned} \sum_{j=g(i)}^{h(i)} \dot{d}(\{v, x\} \cup \bar{C}_j) &= \sum_{j=g(i)}^{h(i)-1} \dot{d}(\{v\} \cup \bar{C}_j) + \sum_{j=g(i)}^{h(i)-1} \dot{d}(\{v\}, \{x\}, \bar{C}_j) + \dot{d}(\{v\} \cup \bar{C}_{h(i)}) \\ &\leq \dot{d}(\{v\} \cup C_i), \end{aligned}$$

and thus

$$\begin{aligned} \sum_{j=g(i)}^{h(i)} e(\bar{C}_j \cup \{x\}, \bar{V}^j) &= \sum_{j=g(i)}^{h(i)} \sum_{v \in \bar{V}^j} \dot{d}(\{v, x\} \cup \bar{C}_j) \\ &= \sum_{v \in V^i} \sum_{j=g(i)}^{h(i)} \dot{d}(\{v, x\} \cup \bar{C}_j) \\ &\leq \sum_{v \in V^i} \dot{d}(\{v\} \cup C_i) \\ &\leq e(C_i, V^i). \end{aligned}$$

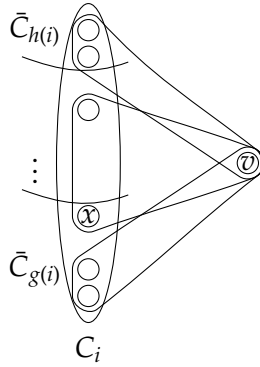


Figure 3.5: Sketch of edges counted by $e(C_i, V^i)$

On the other hand

$$\sum_{j=g(i)}^{h(i)} e(\bar{C}_j \cup \{x\}, \bar{V}^j) > (h(i) - g(i)) \cdot 4 \cdot d^{\frac{1}{3}} + e(\bar{C}_{h(i)}, V^{h(i)}) \geq (h(i) - g(i)) \cdot 4 \cdot d^{\frac{1}{3}}.$$

This leads to

$$\frac{e(C_i, V^i)}{4 \cdot d^{\frac{1}{3}}} > h(i) - g(i)$$

for $i = 1, \dots, p$. Added up, we receive

$$\bar{p} = \sum_{i=1}^p (h(i) - g(i) + 1) = \sum_{i=1}^p (h(i) - g(i)) + p < \sum_{i=1}^p \frac{e(C_i, V^i)}{4 \cdot d^{\frac{1}{3}}} + p \leq \frac{4 \cdot d^{\frac{2}{3}}}{4 \cdot d^{\frac{1}{3}}} + 43 \cdot d^{\frac{1}{3}} = 44 \cdot d^{\frac{1}{3}}.$$

□

Remark 3.14. If $e_K(X, U) = e(X, U)$ for any two sets X, U , then $\Gamma_K(C, U) = \Gamma(C, U)$. Hence, the sets from the previous lemma also satisfy

$$\Gamma_K(\bar{C}_j, \bar{V}^j) = \Gamma(\bar{C}_j, \bar{V}^j)$$

for $j = 1, \dots, \bar{p}$.

3.1.1 A search algorithm on 3-uniform hypergraphs

The following search algorithm $\mathcal{A}_{3\text{-uni}}$ works in four self-contained stages, which we will consider separately.

Stage I

The aim of the first stage is to partition V into free sets $\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{\bar{p}}$ and, furthermore, to find all defective edges in $D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i))$, $i = 1, \dots, \bar{p}$.

Algorithm - Stage I:

(cf. Figure 3.6)

1. Partition V into p disjoint free vertex sets C_1, C_2, \dots, C_p such that

- $p \leq 43 \cdot d^{\frac{1}{3}}$ and
- $\sum_{i=1}^p e_K(C_i, V^i) \leq 3 \cdot d^{\frac{2}{3}}$

holds (cf. Lemma 3.6 for $C = \emptyset$), where $V^i := V \setminus \bigcup_{j=1}^i C_j$, $i = 1, \dots, p$. Set $i := 1$.

2. Identify every vertex in $\Gamma(C_i, V^i)$ for $i = 1, \dots, p-1$ (cf. Lemma 3.7).

□₁ If $\sum_{i=1}^p e_K(C_i, V^i) > 4 \cdot d^{\frac{2}{3}}$, go back to step 1; else, set $d_{xi} := \mathring{d}_K(\{x\} \cup C_i)$ for $i = 1, \dots, p$ and $x \in \Gamma(C_i, V^i)$; further, set $i := 1$ and $X := \Gamma(C_i, V^i)$, then go to step 3.

3. Choose an arbitrary vertex $x \in X$, set $X := X \setminus \{x\}$, and find all defective edges in $\mathring{D}(\{x\} \cup C_i)$ (cf. Lemma 3.8). If

$X \neq \emptyset$: Repeat step 3.

$X = \emptyset$: $i < p-1$: Set $i := i+1$, $X := \Gamma(C_i, V^i)$, and repeat step 3.

$i = p-1$: Go to step □₂.

□₂ If $\sum_{i=1}^p e(C_i, V^i) > 4 \cdot d^{\frac{2}{3}}$, go back to step 1, else, set $i, j := 1$ and go to step 4.

4. Construct according to Lemma 3.13 new free sets $\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{\bar{p}}$ such that

- \bar{C}_i is free for $i = 1, \dots, \bar{p}$,
- $\bar{p} \leq 44 \cdot d^{\frac{1}{3}}$,
- $e_K(\bar{C}_i, \bar{V}^i) = e(\bar{C}_i, \bar{V}^i) \leq 4 \cdot d^{\frac{1}{3}}$ for $i = 1, \dots, \bar{p}$, and
- $\sum_{j=1}^{\bar{p}} e\left(\bar{C}_j, V \setminus \bigcup_{n=1}^j \bar{C}_n\right) \leq 4 \cdot d^{\frac{2}{3}}$.

5. Find all unknown defective edges $e \in D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i))$ with $|e \cap \bar{C}_i| = 1$ and $|e \cap \Gamma(\bar{C}_i, \bar{V}^i)| = 2$ for $i = 1, \dots, \bar{p} - 1$ (cf. Lemma 3.10).

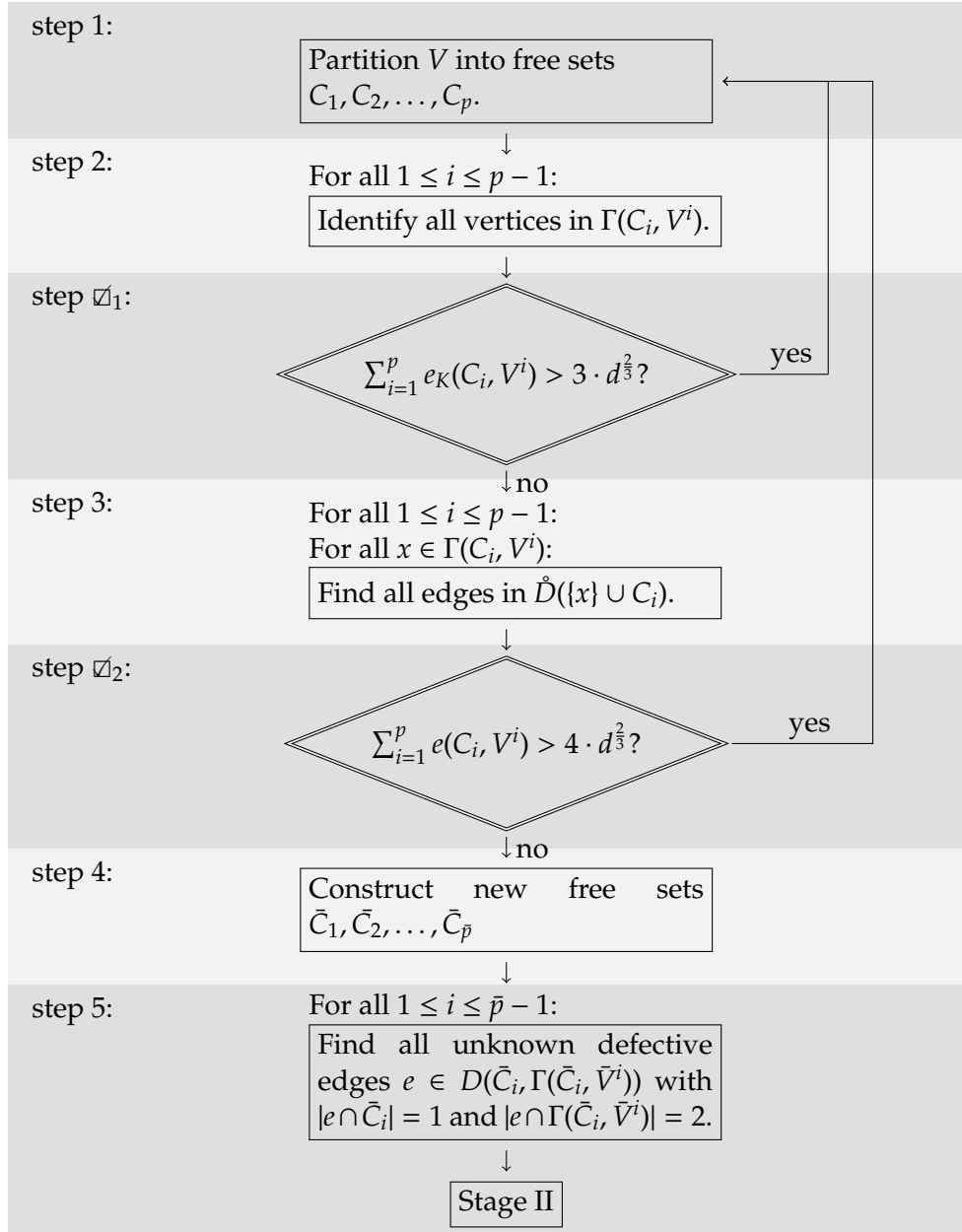


Figure 3.6: Steps in Stage I

The algorithm starts by partitioning V into p free sets C_1, C_2, \dots, C_p . Later, in step 4, the algorithm refines this partition obtaining $\{\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{\bar{p}}\}$. But beforehand, in step 2, the algorithm identifies, according to Lemma 3.7, all vertices that lie in $\Gamma(C_i, V^i)$. Thereafter, $\mathcal{A}_{3\text{-uni}}$ finds for each $x \in \Gamma(C_i, V^i)$ all defective edges in $\hat{D}(\{x\} \cup C_i)$ using Lemma 3.8. Consequently, after step 3 all defective edges in $e \in D(C_i, \Gamma(C_i, V^i))$ with $|e \cap C_i| = 2$ are known for $i = 1, \dots, p$. As above mentioned, the algorithm refines the former partition in step 4 and obtains a new partition $\{\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{\bar{p}}\}$. According to Lemma 3.13, the following assumption holds for the new partition: all defective edges in $e \in D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i))$ with $|e \cap \bar{C}_i| = 2$ are known for $i = 1, \dots, \bar{p}$. Finally, in step 5, the algorithm finds (due to Lemma 3.10) all defective edges in $D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i))$

with $|e \cap \bar{C}_i| = 1$ for $i = 1, \dots, \bar{p}$. Thus, after Stage I, $D_K(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i)) = D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i))$ (cf. Figure 3.7).

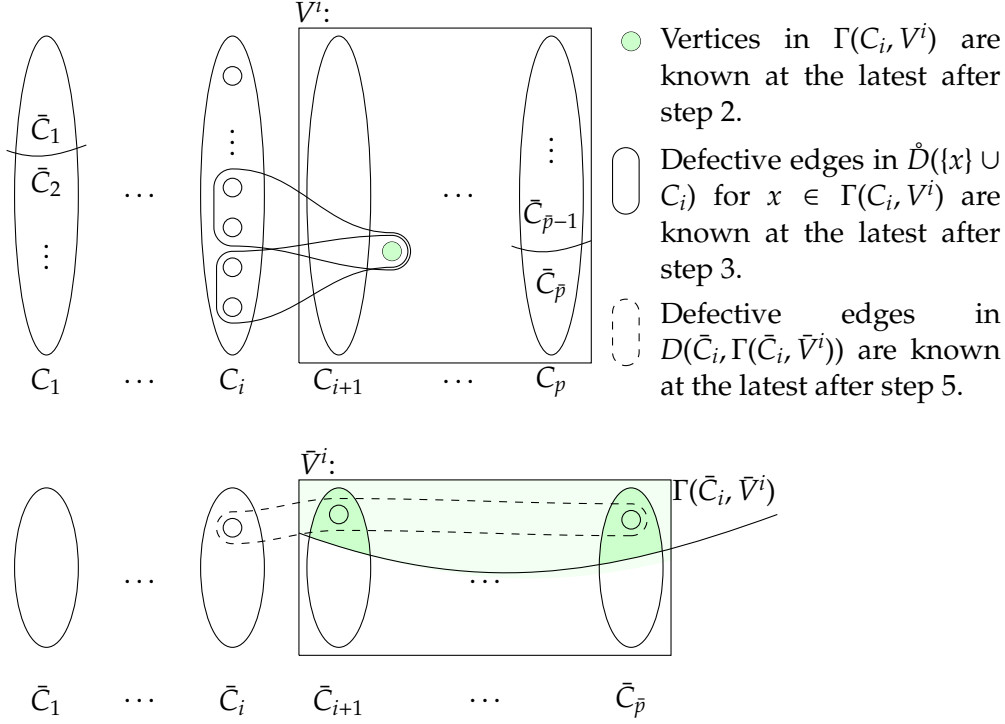


Figure 3.7: Sketch of Stage I.

Besides the steps 1-5, there are two additional steps: step \sphericalangle_1 and \sphericalangle_2 . Due to Lemma 3.8, the number of tests in step 3 depends linearly on the number of defective edges in $\mathring{D}(\{x\} \cup C_i)$ for $i = 1, \dots, p, x \in \Gamma(C_i, V^i)$ that are already known. But these are exactly the edges that are counted by

$$\Sigma_{e_K} := \sum_{i=1}^p e_K(C_i, V^i) = \sum_{i=1}^p \sum_{v \in \Gamma(C_i, V_i)} \mathring{d}_K(\{v\} \cup C_i) = \sum_{i=1}^p \sum_{v \in \Gamma(C_i, V_i)} d_{xi}.$$

To avoid performing too many tests, the algorithm limits Σ_{e_K} to $\Sigma_{e_K} \leq 4 \cdot d^{\frac{2}{3}}$. When the algorithm splits V in step 1 the resulting partition obeys $\Sigma_{e_K} \leq 3 \cdot d^{\frac{2}{3}}$. Since Σ_{e_K} counts only known defective edges, newly found defective edges (could) increase Σ_{e_K} . For that reason \mathcal{A}_{3-uni} verifies after step 2 and 3, that is, at step \sphericalangle_1 and \sphericalangle_2 whether $\Sigma_{e_K} \leq 4 \cdot d^{\frac{2}{3}}$. If $\Sigma_{e_K} > 4 \cdot d^{\frac{2}{3}}$, the algorithm restarts at the beginning and constructs new sets C_1, C_2, \dots, C_p , which again satisfy $\Sigma_{e_K} \leq 3 \cdot d^{\frac{2}{3}}$.

But how often could this possibly happen in the worst case? Every new found defective edge either leaves Σ_{e_K} unchanged or increases Σ_{e_K} by exactly one. The algorithm starts with a partition that satisfies $\Sigma_{e_K} \leq 3 \cdot d^{\frac{2}{3}}$ and restarts, when $\Sigma_{e_K} > 4 \cdot d^{\frac{2}{3}}$. That is, if \mathcal{A}_{3-uni} re-enters Stage I, it has found at least $d^{\frac{2}{3}}$ unknown defective edges in step 2 and 3, which, of course, only can happen

$d/d^{\frac{2}{3}} = d^{\frac{1}{3}}$ times at most. Please note that after step 3

$$\Sigma_{e_K} = \sum_{i=1}^p e_K(C_i, V^i) = \sum_{i=1}^p e(C_i, V^i)$$

holds, since all defective edges in $D(C_i, \Gamma(C_i, V^i))$ with $|e \cap C_i| = 2$ for $i = 1, \dots, p$ are found. That is in particular, Σ_{e_K} remains constant after step 3. To find the remaining unknown defective edges in $D(C_i, \Gamma(C_i, V^i))$, the algorithm uses Lemma 3.10, where the number $e(C_i, V^i)$ makes quadratic contribution to the total number of tests needed ($i = 1, \dots, p$). Up to step 4, the algorithm has only monitored the sum of $e(C_i, V^i)$ over all $1 \leq i \leq p$. To limit the number of tests in step 5, the algorithm splits, as early as in step 4, all sets C_i with $e(C_i, V^i) > 4 \cdot d^{\frac{1}{3}}$ into smaller sets and in this way receives the new partition $\{\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{\bar{p}}\}$.

Stage II

Set $V_L := \{x \in V : d_K(\{x\}) > 2 \cdot d^{\frac{2}{3}}\}$ the set of vertices which we denote as *large vertices*, further set $V_S := V \setminus V_L$. Please note that the set V_L is dynamic and likely to increase during the search process.

The aim of the second stage is to find all defective edges in

$$D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L), \bar{V}^i \setminus \Gamma(\bar{C}_i))$$

for $i = 1, \dots, \bar{p}$.

Algorithm - Stage II:

(cf. Figure 3.8)

Set $i := 1, X^i := \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)$. Choose a vertex $x \in X^i$ and set $X^i := X^i \setminus \{x\}$.

1. Split \bar{C}_i into r_{ix} disjoint vertex sets $F_1^{ix}, F_2^{ix}, \dots, F_{r_{ix}}^{ix}$ such that

- $r_{ix} \leq 2d(\{x\} \cup \bar{C}_i) + 1$ and
- $\{x\} \cup F_j^{ix}$ is free for $j = 1, \dots, r_{ix}$

(cf. Corollary 3.12). Set $j := 1$ and go to step 2.

2. Identify every vertex in $\Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))$ according to Lemma 3.7. Afterwards, set $Y := \Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))$. Choose a vertex $y \in Y$, set $Y := Y \setminus \{y\}$, and go to step 3.

3. Find all defective edges in $\mathring{D}(\{x, y\} \cup F_j^{ix})$ (cf. Lemma 3.9). Then, if

$Y \neq \emptyset$: Choose a vertex $y \in Y$, set $Y := Y \setminus \{y\}$, and repeat step 3.

$Y = \emptyset$: $j < r_{ix}$ and $x \notin V_L$: Set $j := j + 1$ and repeat step 2.

$j = r_{ix}$ or $x \in V_L$: $X^i \setminus V_L \neq \emptyset$: Choose a vertex $x \in X^i \setminus V_L$, set $X^i := X^i \setminus \{x\}$, and repeat step 1.

$X^i \setminus V_L = \emptyset$: $i < \bar{p} - 1$: Set $i := i + 1$, choose a vertex $x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)$, set $X^i := \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L) \setminus \{x\}$, and repeat step 1.

$i = \bar{p} - 1$: Go to Stage III.

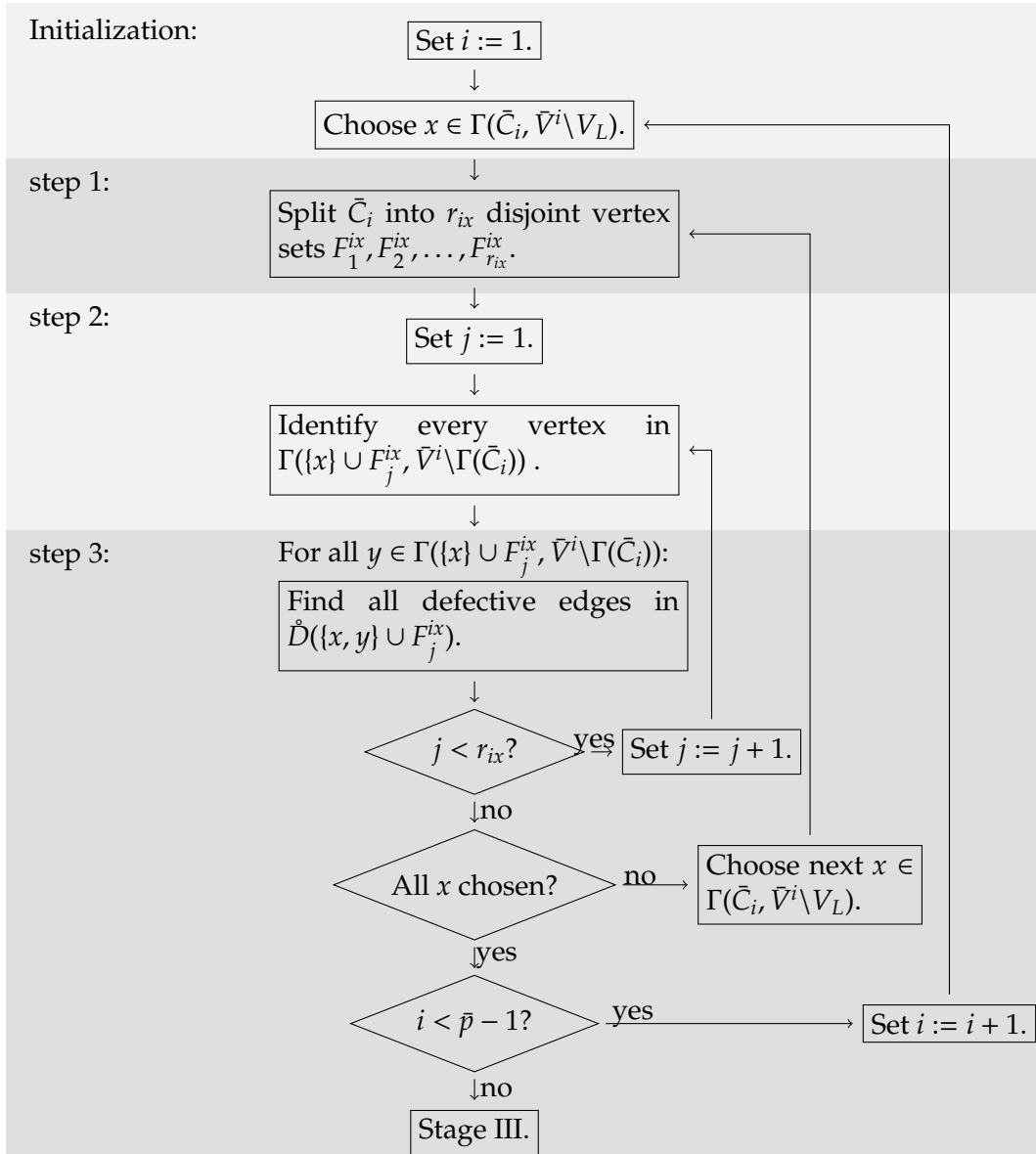


Figure 3.8: Steps in Stage II

The idea of the second stage is the following. For any three vertex sets $X, Y, Z \subset V$

$$D(X, Y, Z) = D(X, Y, \Gamma(X \cup Y, Z))$$

holds. Hence, all defective edges in $D(X, Y, Z)$ can be found by the following two steps: First, identify all vertices in $\Gamma(X \cup Y, Z)$ and second, find for each $z \in \Gamma(X \cup Y, Z)$ all defective edges in $\mathring{D}(\{z\} \cup X \cup Y)$. Using Lemma 3.7 to identify all vertices in $\Gamma(X \cup Y, Z)$ is only possible if $X \cup Y$ is free.

Since $\bar{C}_i \cup \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)$ is certainly not free, the algorithm partitions \bar{C}_i for each $x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)$ into r_{ix} sets $F_1^{ix}, F_2^{ix}, \dots, F_{r_{ix}}^{ix}$ such that $\{x\} \cup F_j^{ix}$ is free ($j = 1, \dots, r_{ix}$). According to Lemma 3.7 and 3.9, the algorithm identifies all vertices in

$$\Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))$$

(in step 2) and finds all defective edges in

$$\mathring{D}(\{x, y\} \cup F_j^{ix})$$

for $z \in \Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))$ (in step 3). Seeing that

$$\begin{aligned} & \bigcup_{i=1}^{\bar{p}} D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L), \bar{V}^i \setminus \Gamma(\bar{C}_i)) \\ = & \bigcup_{i=1}^{\bar{p}} \bigcup_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} D(\bar{C}_i, \{x\}, \bar{V}^i \setminus \Gamma(\bar{C}_i)) \\ = & \bigcup_{i=1}^{\bar{p}} \bigcup_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} \bigcup_{j=1}^{r_{ix}} D(F_j^{ix}, \{x\}, \bar{V}^i \setminus \Gamma(\bar{C}_i)) \\ = & \bigcup_{i=1}^{\bar{p}} \bigcup_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} \bigcup_{j=1}^{r_{ix}} D(F_j^{ix}, \{x\}, \Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))), \end{aligned}$$

certainly, all defective edges in

$$D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L), \bar{V}^i \setminus \Gamma(\bar{C}_i))$$

for $i = 1, \dots, \bar{p}$ are known after Stage II (cf. Figure 3.9).

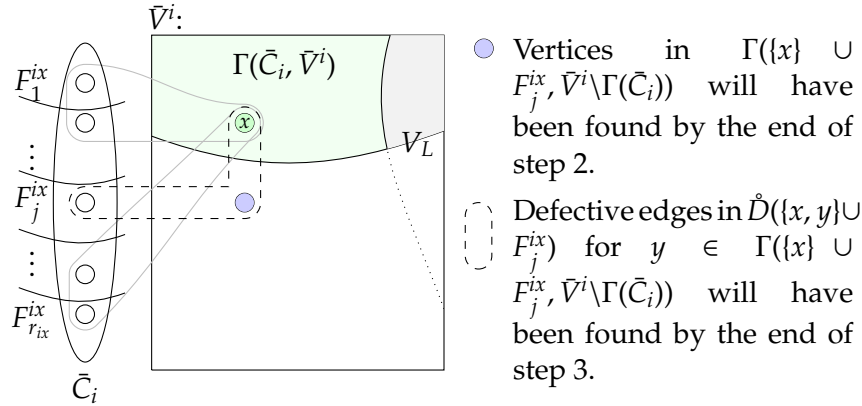


Figure 3.9: Sketch of Stage II.

Stage III

The aim of the third stage is to find all defective edges in

$$D(V_L).$$

These are in particular all edges in $D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i \cap V_L), \bar{V}^i \setminus \Gamma(\bar{C}_i))$, $i = 1, \dots, \bar{p}$. Consequently, after Stage III, all unknown defective edges lie in a set $D(\bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i))$ for some $1 \leq i \leq \bar{p}$.

Algorithm - Stage III:

(cf. Figure 3.10)

Number the vertices in V_L consecutively, let $V_L = \{x_1, x_2, \dots, x_{|V_L|}\}$, further set $V^{x_i} := V \setminus \bigcup_{j=1}^i \{x_j\}$ for all $i = 1, \dots, |V_L|$ and $i := 1$.

1. Partition V^{x_i} into q_{x_i} disjoint sets $G_1^{x_i}, G_2^{x_i}, \dots, G_{q_{x_i}}^{x_i}$ such that

- $\{x_i\} \cup G_j^{x_i}$ is free for $j = 1, \dots, q_{x_i}$,
- $q_{x_i} \leq 43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(\{x_i\})}$, and
- $\sum_{j=1}^{q_{x_i}} e_K(G_j^{x_i}, V^{x_{ij}}) \leq 3 \cdot d^{\frac{2}{3}} + d_K(\{x_i\})$

holds (cf. Lemma 3.6), where $V^{x_{ij}} = V^{x_i} \setminus \bigcup_{k=1}^j G_k^{x_i}$ for $j = 1, \dots, q_{x_i} - 1$. Set $j := 1$ and go to step 2.

2. Identify every vertex in $\Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_{ij}})$ according to Lemma 3.7. If $j < q_{x_i} - 1$, set $j := j + 1$ and repeat step 2; else, go to \square .

\square If $\sum_{j=1}^{q_{x_i}} e_K(G_j^{x_i}, V^{x_{ij}}) > 4 \cdot d^{\frac{2}{3}} + 2 \cdot d_K(\{x_i\})$, go back to step 1; else, set $d_{x_i j y} := \mathring{d}_K(\{y\} \cup G_j^{x_i})$ for $j = 1, \dots, q_{x_i}$, $y \in \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_{ij}})$. Further, set $j := 1$, $Y := \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_{ij}})$, and go to step 3.

3. Choose a vertex $y \in Y$, set $Y := Y \setminus \{y\}$, and find every defective edge in $\mathring{D}(\{x_i, y\} \cup G_j^{x_i})$ according to Lemma 3.8. If

$Y \neq \emptyset$: Repeat step 3.

$Y = \emptyset$: $j < q_{x_i} - 1$: Set $j := j + 1$, $Y := \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_{ij}})$, and repeat step 3.

$j = q_{x_i} - 1$: $i < |V_L|$: Set $i := i + 1$ and repeat step 1.

$i = |V_L|$: Go to Stage IV.

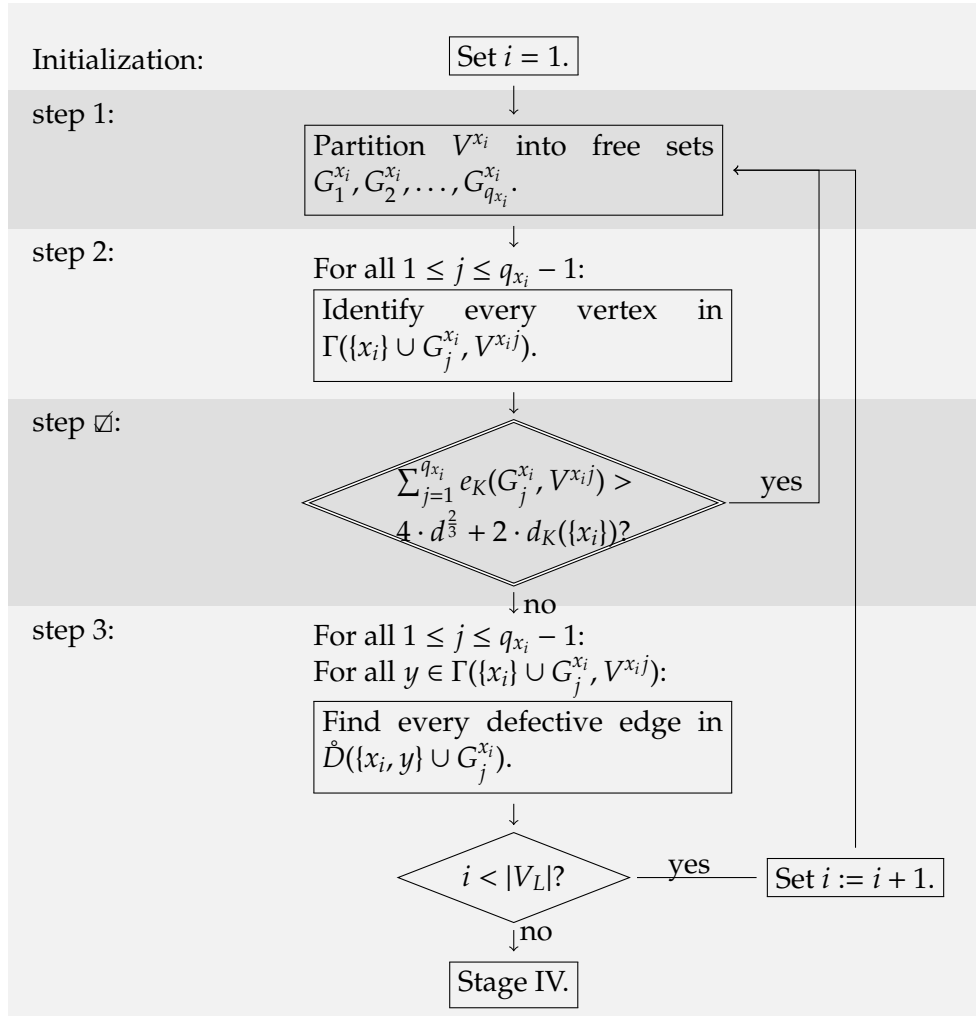


Figure 3.10: Steps in Stage III

The algorithm partitions V^{x_i} into q_{x_i} sets $G_1^{x_i}, G_2^{x_i}, \dots, G_{q_{x_i}}^{x_i}$ such that the sets $\{x_i\} \cup G_j^{x_i}$ are free for $j = 1, \dots, q_{x_i}$ and $i = 1, \dots, |V_L|$. Hence, for every defective edge $e \in D(\{x_i\}, V^{x_i})$ there is some set $G_j^{x_i}$ with

$$e \in D(\{x_i\}, G_j^{x_i}, V^{x_i j}) = D(\{x_i\}, G_j^{x_i}, \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})).$$

Now, the idea is the same as in the previous stage. The algorithm identifies for all $x_i \in V_L$ and $j = 1, \dots, q_{x_i}$ the vertices in

$$\Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})$$

and finds afterwards, for each $y \in \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})$, all edges in

$$\hat{D}(\{x_i, y\} \cup G_j^{x_i}).$$

That is to say, after Stage III all defective edges in

$$\bigcup_{i=1}^{|V_L|} D(\{x_i\}, V^{x_i}) = D(V_L)$$

are known. To avoid performing too many tests, the algorithm limits, in accordance to Stage I, the sum

$$\sum_{j=1}^{q_{x_i}} e_K(G_j^{x_i}, V^{x_{ij}}) \leq 4 \cdot d^{\frac{2}{3}} + 2 \cdot d_K(\{x_i\}).$$

After Stage II, all defective edges in

$$D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L), \bar{V}^i \setminus \Gamma(\bar{C}_i))$$

are known. Consequently, after Stage III, when all defective edges in $D(V_L)$ have been detected, all the more are all edges known which lie in

$$D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i), \bar{V}^i \setminus \Gamma(\bar{C}_i)), (i = 1, \dots, \bar{p})$$

(cf. Figure 3.11)

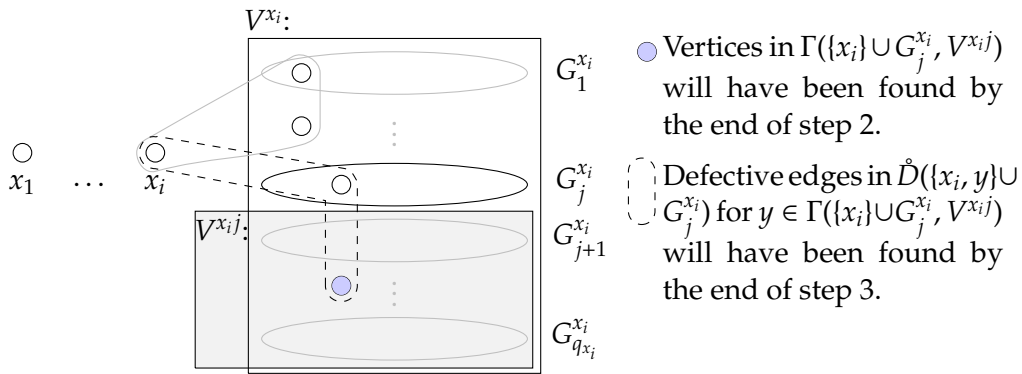


Figure 3.11: Sketch of Stage III.

Stage IV

At the last stage the algorithm finds all defective edges in

$$D(\bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i)), 1 \leq i \leq \bar{p}$$

and thus the remaining unknown defective edges.

Algorithm - Stage IV:

(cf. Figure 3.12)

Set $i := 1$.

1. Split $\bar{V}^i \setminus \Gamma(\bar{C}_i)$ into q_i disjoint sets $G_1^i, G_2^i, \dots, G_{q_i}^i$ such that

- $\bar{C}_i \cup G_j^i$ is free for $1 \leq j \leq q_i$,
- $q_i \leq 43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(\bar{C}_i)}$ and
- $\sum_{k=1}^{q_i} e_K(G_k^i, V^{ij}) \leq 3 \cdot d^{\frac{2}{3}} + d_K(\bar{C}_i)$

holds (cf. Lemma 3.6), where $V^{ij} = \bar{V}^i \setminus (\Gamma(\bar{C}_i) \cup \bigcup_{k=1}^j G_k^i)$. Set $j := 1$ and go to step 2.

2. Identify every vertex in $\Gamma(\bar{C}_i \cup G_j^i, V^{ij})$ (cf. Lemma 3.7). If $j < q_i - 1$, set $j := j + 1$ and repeat step 2; else, go to \square .

\square If $\sum_{j=1}^{q_i} e_K(G_j^i, V^{ij}) > 4 \cdot d^{\frac{2}{3}} + 2 \cdot d_K(\bar{C}_i)$, go back to step 1; else, set $d_{ijx} := \mathring{d}_K(\{x\} \cup G_j^i)$ for $1 \leq j \leq q_i, x \in \Gamma(G_j^i, V^{ij})$, further set $j := 1, X := \Gamma(\bar{C}_i \cup G_j^i, V^{ij})$ and go to step 3.

3. Choose a vertex $x \in X$, set $X := X \setminus \{x\}$, and find every defective edge in $\mathring{D}(\{x\} \cup \bar{C}_i \cup G_j^i)$ by means of Lemma 3.8. If
 $X \neq \emptyset$: Repeat step 3.
 $X = \emptyset$: $j < q_i - 1$: Set $j := j + 1, X := \Gamma(\bar{C}_i \cup G_j^i, V^{ij})$, and repeat step 3.
 $j = q_i - 1$: $i < \bar{p} - 1$: Set $i := i + 1$ and repeat step 1.
 $i = \bar{p} - 1$: Stop.

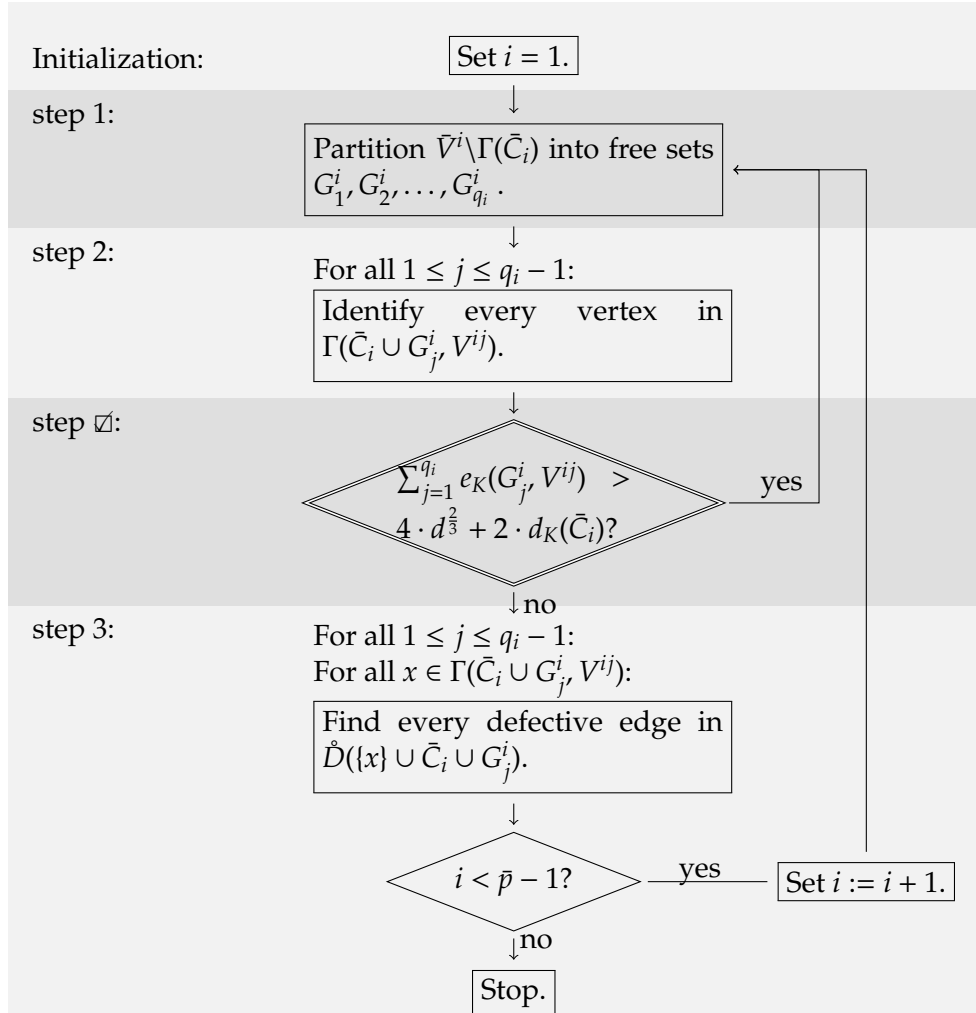


Figure 3.12: Steps in Stage IV

Stage IV works for \bar{C}_i ($i = 1, \dots, \bar{p}$) as Stage III works for $x_i \in V_L$: The algorithm partitions $\bar{V}^i \setminus \Gamma(\bar{C}_i)$ into q_i sets $G_1^i, G_2^i, \dots, G_{q_i}^i$ such that all sets $\bar{C}_i \cup G_j^i$, $1 \leq j \leq q_i$ are free for $i = 1, \dots, \bar{p}$. Again, there is for every defective edge $e \in D(\bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i))$ some set G_j^i with

$$e \in D(\bar{C}_i, G_j^i, V^{ij}).$$

In step 2 and 3, \mathcal{A}_{3-uni} identifies for $i = 1, \dots, \bar{p}$ and $j = 1, \dots, q_i$ all vertices in

$$\Gamma(\bar{C}_i \cup G_j^i, V^{ij})$$

and finds afterwards for each $x \in \Gamma(\bar{C}_i \cup G_j^i, V^{ij})$ all edges in

$$\mathring{D}(\{x\} \cup \bar{C}_i \cup G_j^i).$$

To avoid too many tests, the algorithm limits (similar to stage I and III) the sum

$$\sum_{j=1}^{q_i} e_K(G_j^i, V^{ij}) \leq 4 \cdot d^{\frac{2}{3}} + 2 \cdot d_K(\bar{C}_i)$$

for each \bar{C}_i .

Resulting from the above, the algorithm finds all defective edges (cf. Figure 3.13) in

$$\begin{aligned} & \bigcup_{i=1}^{\bar{p}} \left(\overbrace{D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i))}^{\text{Stage I}} \cup \overbrace{D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i), \bar{V}^i \setminus \Gamma(\bar{C}_i))}^{\text{Stage II / III}} \cup \overbrace{D(\bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i))}^{\text{Stage IV}} \right) \\ &= \bigcup_{i=1}^{\bar{p}} D(\bar{C}_i, \bar{V}^i) \\ &= D. \end{aligned}$$

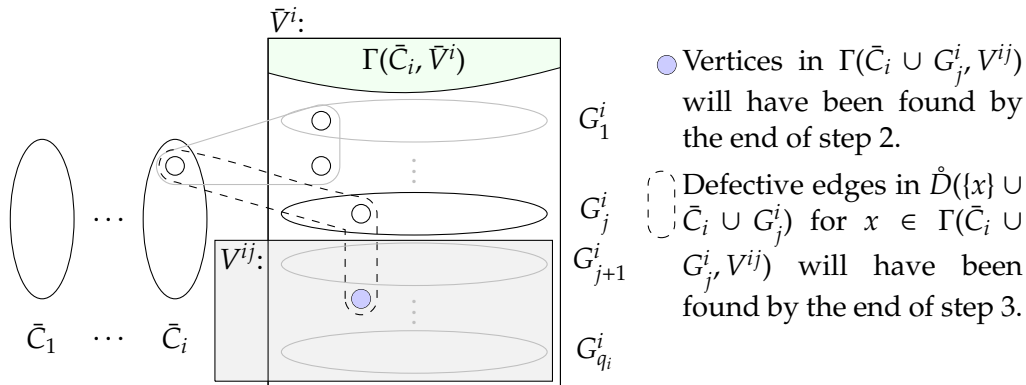


Figure 3.13: Sketch of Stage IV.

3.1.2 An upper bound for the number of tests

Let us now deduce an upper bound for the number of tests that the algorithm performs. We have already mentioned that there are at most

$$2d \tag{3.13}$$

non-substantial tests.

The number of substantial tests that the algorithm performs in the different steps is given by Lemmas 3.6 - 3.10. According to these Lemmas, finding a defective edge costs

$$(\log_2 \rho + c),$$

substantial tests, where (owing to the situation) $c \in \{1, 2, 3, 4, 60\}$. Since every defective edge is detected only once, this adds up to at most

$$d \cdot (\lceil \log_2(|E|/d) \rceil + 60) \tag{3.14}$$

substantial tests. Besides those tests each lemma causes further, negative substantial tests, independent of the number of defective edges that are found. In the following we will count these tests separately for each step.

But beforehand, let us start with some estimations:

Let X_1, X_2, \dots, X_s be a partition of some vertex set $U \subseteq V$. Every edge joins vertices from at most three different sets X_i , therefore

$$\sum_{i=1}^s d_K(X_i) \leq 3 \cdot d_K(U) \text{ and } \sum_{i=1}^s d(X_i) \leq 3 \cdot d(U). \tag{3.15}$$

For the sum of square roots one receives by the Cauchy-Schwarz inequality that

$$\sum_{i=1}^s \sqrt{d_K(X_i)} \leq \sqrt{\sum_{i=1}^s d_K(X_i)} \cdot \sqrt{\sum_{i=1}^s 1} \stackrel{(3.15)}{\leq} \sqrt{3 \cdot d_K(U)} \cdot \sqrt{s}. \tag{3.16}$$

Stage I:

In step 1 the algorithm partitions V and needs therefore, according to Lemma 3.6, at most

$$42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(\emptyset, V)} = \mathcal{O}\left(d^{\frac{1}{3}}\right) \tag{3.17}$$

substantial tests¹.

Afterwards, in step 2, the algorithm identifies all vertices in $\Gamma(C_i, V^i)$ which costs, due to Lemma 3.7, at most

$$42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(C_i, V^i)} = \mathcal{O}\left(d^{\frac{1}{3}} + \sqrt{d_K(C_i, V^i)}\right) \tag{3.18}$$

¹Plus those substantial tests that are already counted in (3.14).

substantial tests¹ for $i = 1, \dots, p$.

Consider next step 3 for some fixed set C_i , $1 \leq i \leq p$ and some vertex $x \in \Gamma(C_i, V^i)$. The number of tests that are needed to find all defective edges in

$$\mathring{D}(\{x\} \cup C_i)$$

depends, according to Lemma 3.8, on the number of defective edges in $\mathring{D}(\{x\} \cup C_i)$ that are known beforehand. In step \square_1 , the algorithm sets $d_{xi} := \mathring{d}_K(\{x\} \cup C_i)$. Hence, d_{xi} denotes the number of defective edges in $\mathring{D}(\{x\} \cup C_i)$ that are known immediately after step 2 and that are limited due to step \square_1 by

$$\sum_{i=1}^p \sum_{x \in \Gamma(C_i, V^i)} d_{xi} = \sum_{i=1}^p e_K(C_i, V^i) \leq 4 \cdot d^{\frac{2}{3}}. \quad (3.19)$$

Now, could $\mathring{d}_K(\{x\} \cup C_i)$ increase between step \square_1 and the beginning of the search on $\mathring{D}(\{x\} \cup C_i)$ in step 3? No, certainly not, since all sets

$$\mathring{D}(\{x\} \cup C_i), \quad 1 \leq i \leq p, \quad x \in \Gamma(C_i, V^i)$$

are disjoint. Due to Lemma 3.8, the algorithm needs at most

$$\sum_{i=1}^p \sum_{x \in \Gamma(C_i, V^i)} 3 \cdot d_{xi} \stackrel{(3.19)}{\leq} 12 \cdot d^{\frac{2}{3}} = O(d^{\frac{2}{3}}) \quad (3.20)$$

substantial tests¹ in step 3.

In step 4, there are no tests, the algorithm refines the partition $\{C_1, C_2, \dots, C_p\}$ and obtains $\{\bar{C}_1, \bar{C}_2, \dots, \bar{C}_{\bar{p}}\}$ such that

$$(|\Gamma(\bar{C}_i, \bar{V}^i)| \leq) \quad e(\bar{C}_i, \bar{V}^i) \leq 4 \cdot d^{\frac{1}{3}} \text{ for } i = 1, \dots, \bar{p}. \quad (3.21)$$

Finally, in step 5, the algorithm finds all defective edges in $D(\bar{C}_i, \Gamma(\bar{C}_i, \bar{V}^i))$ with $|e \cap \bar{C}_i| = 1$ and $|e \cap \Gamma(\bar{C}_i, \bar{V}^i)| = 2$ for $i = 1, \dots, \bar{p}$. Due to Lemma 3.10, this costs at most

$$3 \cdot e(\bar{C}_i, \bar{V}^i)^2 \stackrel{(3.21)}{\leq} 48 \cdot d^{\frac{2}{3}} = O(d^{\frac{2}{3}}) \quad (3.22)$$

substantial tests¹ ($i = 1, \dots, \bar{p}$).

Before we sum up the number of tests of Stage I, let us first of all turn to some observations:

- $p \leq 43 \cdot d^{\frac{1}{3}} = O(d^{\frac{1}{3}})$ and
- $\bar{p} \leq 44 \cdot d^{\frac{1}{3}} = O(d^{\frac{1}{3}})$

We have stated that the algorithm repeats step 1 at most $d^{\frac{1}{3}}$ times, either after step 2 or 3. That is, in the worst case, the algorithm performs the steps 1 - 3 $d^{\frac{1}{3}}$ times. Altogether, we obtain that

¹Plus those substantial tests that are already counted in (3.14).

the algorithm performs at most

$$\begin{aligned}
& d^{\frac{1}{3}} \cdot \left[\overbrace{O(d^{\frac{1}{3}})}^{\text{step 1}} + \overbrace{\sum_{i=1}^p O\left(d^{\frac{1}{3}} + \sqrt{d_K(C_i, V^i)}\right)}^{\text{step 2}} + \overbrace{O(d^{\frac{2}{3}})}^{\text{step 3}} \right] + \overbrace{\sum_{i=1}^{\bar{p}} O(d^{\frac{2}{3}})}^{\text{step 5}} \\
&= d^{\frac{1}{3}} \cdot \left[O\left(p \cdot d^{\frac{1}{3}} + \sum_{i=1}^p \sqrt{d_K(C_i, V^i)}\right) + O(d^{\frac{2}{3}}) \right] + O(d) \\
&\stackrel{(3.16)}{=} d^{\frac{1}{3}} \cdot \left[O\left(d^{\frac{2}{3}} + \sqrt{\bar{p}} \cdot \sqrt{\bar{d}}\right) \right] + O(d) \\
&= O(d)
\end{aligned}$$

substantial tests¹ at Stage I.

Stage II

At this stage there are no iterations that are caused by too many defective edges. Nevertheless, the algorithm repeats Stage II exactly $\bar{p} - 1$ times since $\mathcal{A}_{3\text{-uni}}$ performs Stage II once for each set \bar{C}_i , $1 \leq i \leq \bar{p} - 1$. So let us now consider the three steps of stage II for some fixed set \bar{C}_i :

In the first step the algorithm performs no tests. The algorithm partitions \bar{C}_i for each $x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)$ into r_{ix} sets $F_1^{ix}, F_2^{ix}, \dots, F_{r_{ix}}^{ix}$ such that

- $r_{ix} \leq 2\bar{d}(\{x\} \cup \bar{C}_i) + 1$ and
 - $\{x\} \cup F_j^{ix}$ is free for $j = 1, \dots, r_{ix}$.
- (3.23)

In the next step the algorithm identifies for each $x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i))$ all vertices in

$$\Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))$$

which costs for each x , due to Lemma 3.7, at most

$$42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))}$$

substantial tests for $j = 1, \dots, r_{ix}$.

Owing that

$$d_K(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i)) \leq d_K(\{x\}) + d_K(F_j^{ix}),$$

we obtain all the more

$$\sqrt{d_K(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))} \leq \sqrt{d_K(\{x\})} + \sqrt{d_K(F_j^{ix})}.$$

Let now $x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)$, then in particular $x \notin V_L$, which has been confirmed every time the algorithm returns to step 2, and thus

$$d_K(\{x\}) \leq 2 \cdot d^{\frac{2}{3}} \Rightarrow \sqrt{d_K(\{x\})} \leq 2 \cdot d^{\frac{1}{3}}.$$

¹Plus those substantial tests that are already counted in (3.14).

Consequently,

$$\sqrt{d_K(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))} = 2 \cdot d^{\frac{1}{3}} + \sqrt{d_K(F_j^{ix})}. \quad (3.24)$$

By (3.16) we obtain

$$\sum_{j=1}^{r_{ix}} \sqrt{d_K(F_j^{ix})} \leq \sqrt{r_{ix}} \cdot \sqrt{3 \cdot d_K(\bar{C}_i)} \leq r_{ix} \cdot \sqrt{3 \cdot d_K(\bar{C}_i)}.$$

With

$$\begin{aligned} \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} r_{ix} &\stackrel{(3.23)}{\leq} \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} (2\mathring{d}(\{x\} \cup \bar{C}_i) + 1) \\ &= 2e(\bar{C}_i, \bar{V}^i \setminus V_L) + 2|\Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)| \\ &\stackrel{(3.21)}{\leq} 16 \cdot d^{\frac{1}{3}} \end{aligned} \quad (3.25)$$

it follows that

$$\begin{aligned} \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} \sum_{j=1}^{r_{ix}} \sqrt{d_K(F_j^{ix})} &\leq \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} r_{ix} \cdot \sqrt{3 \cdot d_K(\bar{C}_i)} \\ &= \sqrt{3 \cdot d_K(\bar{C}_i)} \cdot \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} r_{ix} \\ &\stackrel{(3.25)}{\leq} 14 \cdot d^{\frac{1}{3}} \cdot \sqrt{d_K(\bar{C}_i)}. \end{aligned}$$

So throughout step 2 the algorithm needs at most

$$\begin{aligned} &\sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} \sum_{j=1}^{r_{ix}} \left(42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))} \right) \\ &\stackrel{(3.24)}{\leq} 42 \cdot d^{\frac{1}{3}} \cdot \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} r_{ix} + 56 \cdot \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i))} \sum_{j=1}^{r_{ix}} \left(2 \cdot d^{\frac{1}{3}} + \sqrt{d_K(F_j^{ix})} \right) \\ &\stackrel{(3.25)}{\leq} 42 \cdot d^{\frac{1}{3}} \cdot 16 \cdot d^{\frac{1}{3}} + 56 \cdot 2 \cdot d^{\frac{1}{3}} \cdot \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i))} r_{ix} + 56 \cdot \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i))} \sum_{j=1}^{r_{ix}} \sqrt{d_K(F_j^{ix})} \\ &\stackrel{(3.25)}{\leq} 42 \cdot d^{\frac{1}{3}} \cdot 16 \cdot d^{\frac{1}{3}} + 56 \cdot 2 \cdot d^{\frac{1}{3}} \cdot 16 \cdot d^{\frac{1}{3}} + 56 \cdot 14 \cdot d^{\frac{1}{3}} \cdot \sqrt{d_K(\bar{C}_i)} \\ &= O\left(d^{\frac{2}{3}} + d^{\frac{1}{3}} \cdot \sqrt{d_K(\bar{C}_i)}\right) \end{aligned} \quad (3.26)$$

substantial tests¹.

Finally, in step 3, there is according to Lemma 3.9, at most one substantial test¹ for each $x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)$, $j = 1, \dots, r_{ix}$ and $y \in \Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))$ to find all edges in

$$\mathring{D}(\{x, y\} \cup F_j^{ix}).$$

¹Plus those substantial tests that are already counted in (3.14).

They add up to at most

$$\sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} \sum_{j=1}^{r_{ix}} |\Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))|$$

substantial tests¹. Certainly, for every $y \in \Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))$ there is at least one defective edge $e \in D$ with $e \in \mathring{D}(\{x, y\} \cup F_j^{ix})$. Since $\{x\} \cup F_j^{ix}$ is free and since $y \notin \Gamma(\bar{C}_i)$ which implies that $y \notin \Gamma(F_j^{ix})$, it follows that $e \in D(\{x\}, \{y\}, F_j^{ix})$. Therefore,

$$|\Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))| \leq d(\{x\}, F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i)).$$

Please note that $\Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)$, \bar{C}_i and $\bar{V}^i \setminus \Gamma(\bar{C}_i)$ are disjoint, accordingly, the algorithm needs at most

$$\begin{aligned} & \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} \sum_{j=1}^{r_{ix}} |\Gamma(\{x\} \cup F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i))| \\ & \leq \sum_{x \in \Gamma(\bar{C}_i, \bar{V}^i \setminus V_L)} \sum_{j=1}^{r_{ix}} d(\{x\}, F_j^{ix}, \bar{V}^i \setminus \Gamma(\bar{C}_i)) \\ & = d(\Gamma(\bar{C}_i, \bar{V}^i \setminus V_L), \bar{C}_i, \bar{V}^i \setminus \Gamma(\bar{C}_i)) \\ & \leq d(\bar{C}_i) \end{aligned} \tag{3.27}$$

substantial tests¹ in step 3.

Let us now add up the number of tests over all sets \bar{C}_i , $1 \leq i \leq \bar{p} - 1$. We reach maximum of

$$\begin{aligned} & \sum_{i=1}^{\bar{p}-1} \left(\overbrace{\mathcal{O}\left(d^{\frac{2}{3}} + d^{\frac{1}{3}} \cdot \sqrt{d_K(\bar{C}_i)}\right)}^{\text{step 2}} + \overbrace{d(\bar{C}_i)}^{\text{step 3}} \right) \\ & \stackrel{(3.15)}{=} \mathcal{O}\left((\bar{p}-1) \cdot d^{\frac{2}{3}} + \left(d^{\frac{1}{3}} \cdot \sum_{i=1}^{\bar{p}-1} \sqrt{d_K(\bar{C}_i)}\right) + 3d\right) \\ & \stackrel{(3.16)}{=} \mathcal{O}\left(d + d^{\frac{1}{3}} \cdot \sqrt{\bar{p}-1} \cdot \sqrt{3d}\right) \\ & = \mathcal{O}(d) \end{aligned} \tag{3.28}$$

substantial tests¹ during Stage II.

Stage III

The algorithm performs stage III once for each large vertex $x_i \in V_L$. So, let us consider Stage III for some fixed $x_i \in V_L$:

The algorithm starts by partitioning V^{x_i} into q_{x_i} sets $G_1^{x_i}, G_2^{x_i}, \dots, G_{q_{x_i}}^{x_i}$ such that

- $\{x_i\} \cup G_j^{x_i}$ is free for $j = 1, \dots, q_{x_i}$,

¹Plus those substantial tests that are already counted in (3.14).

$$\bullet \quad q_{x_i} \leq 43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(\{x_i\})} \stackrel{d_K(\{x_i\}) > 2 \cdot d^{2/3}}{\leq} 79 \cdot \sqrt{d_K(\{x_i\})} \text{ and} \quad (3.29)$$

$$\bullet \quad \sum_{j=1}^{q_{x_i}} e_K(G_j^{x_i}, V^{x_i j}) \leq 3 \cdot d^{\frac{2}{3}} + d_K(\{x_i\}). \quad (3.30)$$

For the partitioning of V^{x_i} in step 1, the algorithm needs, due to Lemma 3.6, at most

$$42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(\{x_i\})} \stackrel{d_K(\{x_i\}) > 2 \cdot d^{2/3}}{<} 77 \cdot \sqrt{d_K(\{x_i\})}. \quad (3.31)$$

substantial tests¹.

In step 2, \mathcal{A}_{3-uni} identifies all vertices in $\Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})$, which costs, according to Lemma 3.7, at most

$$42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(\{x_i\} \cup G_j^{x_i}, V^{x_i j})} < 77 \cdot \sqrt{d_K(\{x_i\} \cup G_j^{x_i})} \quad (3.32)$$

substantial tests¹ for $j = 1, \dots, q_{x_i}$. Thereby, the algorithm is likely to find defective edges such that the left hand side of inequality (3.30), that is, the sum

$$\Sigma_{e_K}^{x_i} := \sum_{j=1}^{q_{x_i}} e_K(G_j^{x_i}, V^{x_i j}),$$

increases. Every new found defective edge contributes at most one to $\Sigma_{e_K}^{x_i}$. Hence, when the partition $\{G_1^{x_i}, G_2^{x_i}, \dots, G_{q_{x_i}}^{x_i}\}$ violates the inequality

$$\sum_{j=1}^{q_{x_i}} e_K(G_j^{x_i}, V^{x_i j}) \leq 4 \cdot d^{\frac{2}{3}} + 2 \cdot d_K(\{x_i\}) \quad (3.33)$$

in step \square , the algorithm has found at least

$$d^{\frac{2}{3}} + d_K(\{x_i\})$$

defective edges. In that case the algorithm repeats step 1 and constructs new sets $G_1^{x_i}, G_2^{x_i}, \dots, G_{q_{x_i}}^{x_i}$ which satisfy again (3.30). Let us now introduce two new quantities for every $x_i \in V_L$: let

- $d_{new}(\{x_i\})$ denote the number of defective edges that have been found and
- $\Phi(\{x_i\})$ denote the number of times that \mathcal{A}_{3-uni} returns to step 1

while the algorithm performs Stage III for x_i . Please note that the edges that are counted by $d_{new}(\{x_i\})$ are not necessarily incident to x_i . As mentioned before, every time the algorithm returns to step 1, it has found at least $d^{2/3} + d_K(\{x_i\})$ defective edges and thus

$$\Phi(\{x_i\}) \leq \frac{d_{new}(\{x_i\})}{d_K(\{x_i\}) + d^{\frac{2}{3}}} < \frac{d_{new}(\{x_i\})}{d_K(\{x_i\})}.$$

¹Plus those substantial tests that are already counted in (3.14).

Consequently, when the algorithm enters step 3 for some x_i , it has performed at most

$$(\Phi(x_i) + 1) \cdot \left[77 \cdot \sqrt{d_K(\{x_i\})} + \sum_{j=1}^{q_{x_i}} \left(77 \cdot \sqrt{d_K(\{x_i\} \cup G_j^{x_i})} \right) \right]$$

substantial tests¹. Because of $\sqrt{d_K(\{x_i\} \cup G_j^{x_i})} \leq \sqrt{d_K(\{x_i\})} + \sqrt{d_K(G_j^{x_i})}$, we have

$$\begin{aligned} & \sum_{j=1}^{q_{x_i}} \sqrt{d_K(\{x_i\} \cup G_j^{x_i})} \\ \leq & q_{x_i} \cdot \sqrt{d_K(\{x_i\})} + \sum_{j=1}^{q_{x_i}} \sqrt{d_K(G_j^{x_i})} \\ \stackrel{(3.29),(3.16)}{\leq} & 79 \cdot \sqrt{d_K(\{x_i\})} \cdot \sqrt{d_K(\{x_i\})} + \sqrt{q_{x_i}} \cdot \sqrt{3 \cdot d} \\ \stackrel{d_K(\{x_i\}) > 2 \cdot d^{\frac{2}{3}}}{\leq} & 79 \cdot d_K(\{x_i\}) + \sqrt{79 \cdot \sqrt{d_K(\{x_i\})} \cdot 3 \cdot (1/2 \cdot d_K(\{x_i\}))^{\frac{3}{2}}} \\ < & 89 \cdot d_K(\{x_i\}). \end{aligned}$$

And therefore

$$\begin{aligned} & (\Phi(x_i) + 1) \cdot \left[77 \sqrt{d_K(\{x_i\})} + \sum_{j=1}^{q_{x_i}} 77 \cdot \sqrt{d_K(\{x_i\} \cup G_j^{x_i})} \right] \\ \leq & (\Phi(x_i) + 1) \cdot \left[77 \left(\sqrt{d_K(\{x_i\})} + 89 \cdot d_K(\{x_i\}) \right) \right] \\ < & \frac{d_{new}(\{x_i\})}{d_K(\{x_i\})} \cdot \left[77 \left(\sqrt{d_K(\{x_i\})} + 89 \cdot d_K(\{x_i\}) \right) \right] + \mathcal{O}(d_K(\{x_i\})) \\ = & \mathcal{O}(d_{new}(\{x_i\}) + d_K(\{x_i\})). \end{aligned} \tag{3.34}$$

The algorithm finds in step 3 for each $y \in \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})$ all edges in

$$\mathring{D}(\{x_i, y\} \cup G_j^{x_i})$$

for $j = 1, \dots, q_{x_i}$. Altogether, that costs, according to Lemma 3.8, at most

$$\sum_{j=1}^{q_{x_i}} \sum_{y \in \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})} 3 \cdot \mathring{d}_K(\{x_i, y\} \cup G_j^{x_i})$$

substantial tests¹, where $\mathring{d}_K(\{x_i, y\} \cup G_j^{x_i})$ denotes the number of defective edges in $\mathring{D}(\{x_i, y\} \cup G_j^{x_i})$ that are known right before the algorithm starts its search on $\mathring{D}(\{x_i, y\} \cup G_j^{x_i})$. Since $\{x_i\} \cup G_j^{x_i}$ is free,

$$\mathring{d}_K(\{x_i, y\} \cup G_j^{x_i}) = \mathring{d}_K(\{y\} \cup G_j^{x_i}) + d_K(\{x_i, \{y\}, G_j^{x_i})$$

for each $y \in \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})$, ($j = 1, \dots, q_{x_i}$). In analogy to Stage I, the algorithm stores the number of defective edges that are known in $\mathring{D}(\{y\} \cup G_j^{x_i})$ right before step 3 in $d_{x_i j y}$. By the

¹Plus those substantial tests that are already counted in (3.14).

same argumentation as in Stage I, this number does not change until the algorithm performs a search on $\mathring{D}(\{x_i, y\} \cup G_j^{x_i})$ in step 3 and we have

$$\begin{aligned}
& \sum_{j=1}^{q_{x_i}} \sum_{y \in \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})} \mathring{d}_K(\{x_i, y\} \cup G_j^{x_i}) \\
&= \sum_{j=1}^{q_{x_i}} \sum_{y \in \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})} d_{x_i j y} \\
(3.33) \quad &\leq 4 \cdot d^{\frac{2}{3}} + 2 \cdot d_K(\{x_i\}) \\
&= \mathcal{O}(d_K(\{x_i\})).
\end{aligned}$$

Furthermore, because of $\Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j}) \subset V^{x_i j}$ and since $V^{x_i j}$ and $G_j^{x_i}$ are disjoint for $j = 1, \dots, q_{x_i}$,

$$\begin{aligned}
& \sum_{j=1}^{q_{x_i}} \sum_{y \in \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j})} d_K(\{x_i\}, \{y\}, G_j^{x_i}) \\
&= \sum_{j=1}^{q_{x_i}} d_K(\{x_i\}, \Gamma(\{x_i\} \cup G_j^{x_i}, V^{x_i j}), G_j^{x_i}) \\
&\leq \sum_{j=1}^{q_{x_i}} d_K(\{x_i\}, V^{x_i j}, G_j^{x_i}) \\
&\leq d_K(\{x_i\}).
\end{aligned}$$

That is, the algorithm performs at most

$$\mathcal{O}(d_K(\{x_i\})). \quad (3.35)$$

substantial tests¹ in step 3.

Let us now add up the tests at Stage III. Since we find every defective edge only once, of course

$$\sum_{i=1}^{|\bar{V}_L|} d_{new}(\{x_i\}) \leq d$$

holds. Taking into account (3.16), we receive for the total number of substantial tests at Stage III

$$\begin{aligned}
& \sum_{i=1}^{|\bar{V}_L|} \left[\overbrace{\mathcal{O}(d_{new}(\{x_i\}) + d_K(\{x_i\}))}^{\text{step 1 and 2}} + \overbrace{\mathcal{O}(d_K(\{x_i\}))}^{\text{step 3}} \right] \\
&= \mathcal{O} \left(\sum_{i=1}^{|\bar{V}_L|} d_{new}(\{x_i\}) + \sum_{i=1}^{|\bar{V}_L|} d_K(\{x_i\}) \right) \\
&= \mathcal{O}(d). \quad (3.36)
\end{aligned}$$

¹Plus those substantial tests that are already counted in (3.14).

Stage IV

The algorithm performs Stage IV once for each set \bar{C}_i , $1 \leq i \leq \bar{p} - 1$. As mentioned already, Stage IV works for \bar{C}_i ($i = 1, \dots, \bar{p}$) as Stage III works for $x_i \in V_L$. Let us hence define analogously two quantities for \bar{C}_i , $1 \leq i \leq \bar{p} - 1$: Let

- $d_{new}(\bar{C}_i)$ denote the number of defective edges that have been found and
- $\Phi(\{\bar{C}_i\})$ denote the number of times that \mathcal{A}_{3-uni} returns to step 1

while the algorithm performs Stage IV for \bar{C}_i . We obtain, by the same argumentation as for Stage III,

- $\Phi(\bar{C}_i) \leq \frac{d_{new}(\bar{C}_i)}{d_K(\bar{C}_i) + d^{\frac{2}{3}}},$

- that the algorithm performs at most $42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(\bar{C}_i, \bar{V}^i)} < 56 \cdot (d^{\frac{1}{3}} + \sqrt{d_K(\bar{C}_i)})$ substantial tests¹ in step 1 to receive a partition $G_1^i, G_2^i, \dots, G_{q_i}^i$ of $\bar{V}^i \setminus \Gamma(\bar{C}_i)$ such that

- $\bar{C}_i \cup G_j^i$ is free for $j = 1, \dots, q_i,$
- $q_i \leq 43 \cdot d^{\frac{1}{3}} + 57 \cdot \sqrt{d_K(\bar{C}_i, \bar{V}^i)} < 57 \cdot \left(d^{\frac{1}{3}} + \sqrt{d_K(\bar{C}_i, \bar{V}^i)} \right),$ (3.37)

- $\sum_{j=1}^{q_i} e_K(G_j^i, V^{ij}) \leq 3 \cdot d^{\frac{2}{3}} + d_K(\bar{C}_i),$ and (3.38)

- after step $\square \sum_{j=1}^{q_i} e_K(G_j^i, V^{ij}) \leq 4 \cdot d^{\frac{2}{3}} + 2 \cdot d_K(\bar{C}_i),$ and (3.39)

- that \mathcal{A}_{3-uni} needs at most at most $\sum_{j=1}^{q_i} 42 \cdot d^{\frac{1}{3}} + 56 \cdot \sqrt{d_K(\bar{C}_i \cup G_j^i, V^{ij})}$ substantial tests¹ in step 2 to identify all vertices in $\Gamma(\bar{C}_i \cup G_j^i, V^{ij})$ for $j = 1, \dots, q_i.$

Altogether, these are less than

$$(\Phi(\bar{C}_i) + 1) \cdot \left[56 \cdot \left(d^{\frac{1}{3}} + \sqrt{d_K(\bar{C}_i)} \right) + \sum_{j=1}^{q_i} 56 \cdot \left(d^{\frac{1}{3}} + \sqrt{d_K(\bar{C}_i \cup G_j^i)} \right) \right]$$

substantial tests¹ for each \bar{C}_i in the first two steps ($i = 1, \dots, \bar{p} - 1$).

Analogously to Stage III, we have

$$\begin{aligned} & \sum_{j=1}^{q_i} \sqrt{d_K(\bar{C}_i \cup G_j^i)} \\ & \leq 43 \cdot d^{\frac{1}{3}} \cdot \sqrt{d_K(\bar{C}_i)} + 57 \cdot d_K(\bar{C}_i) + \sqrt{43 \cdot d^{\frac{1}{3}} \cdot 3 \cdot d} + \sqrt{57 \cdot \sqrt{d_K(\bar{C}_i, \bar{V}^i)} \cdot 3 \cdot d} \end{aligned}$$

¹Plus those substantial tests that are already counted in (3.14).

$$\begin{aligned}
&< 43 \cdot d^{\frac{1}{3}} \cdot \sqrt{d_K(\bar{C}_i)} + 57 \cdot d_K(\bar{C}_i) + 12 \cdot d^{\frac{2}{3}} && + 14 \cdot \sqrt[4]{d_K(\bar{C}_i, \bar{V}^i)} \cdot \sqrt{d} \\
&\leq 114 \cdot (d_K(\bar{C}_i) + d^{\frac{2}{3}}),
\end{aligned}$$

due to $\sqrt{d_K(\bar{C}_i \cup G_j^i)} \leq \sqrt{d_K(\bar{C}_i)} + \sqrt{d_K(G_j^i)}$, (3.16) and (3.37). Further applies

$$\sum_{j=1}^{q_i} d^{\frac{1}{3}} \leq 57 \cdot \left(d^{\frac{2}{3}} + \sqrt{d_K(\bar{C}_i, \bar{V}^i)} \cdot d^{\frac{1}{3}} \right),$$

and thus

$$\begin{aligned}
&(\Phi(\bar{C}_i) + 1) \cdot \left[56 \cdot \left(d^{\frac{1}{3}} + \sqrt{d_K(\bar{C}_i)} \right) + \sum_{j=1}^{q_i} 56 \cdot \left(d^{\frac{1}{3}} + \sqrt{d_K(\bar{C}_i \cup G_j^i)} \right) \right] \\
&\leq \left(\frac{d_{new}(\bar{C}_i)}{d_K(\bar{C}_i) + d^{\frac{2}{3}}} + 1 \right) \cdot \left[56 \cdot \left(d^{\frac{1}{3}} + \sqrt{d_K(\bar{C}_i)} + 57 \cdot \left(d^{\frac{2}{3}} + \sqrt{d_K(\bar{C}_i, \bar{V}^i)} \cdot d^{\frac{1}{3}} \right) + 114 \cdot (d_K(\bar{C}_i) + d^{\frac{2}{3}}) \right) \right] \\
&= O(d_{new}(\bar{C}_i) + d_K(\bar{C}_i) + d^{\frac{2}{3}}). \tag{3.40}
\end{aligned}$$

In step 3 the algorithm finds for all $x \in \Gamma(\bar{C}_i \cup G_j^i, V^{ij})$ all edges in

$$\hat{D}(\{x\} \cup \bar{C}_i \cup G_j^i)$$

for $j = 1, \dots, q_i$, and, due to Lemma 3.8, this costs in total at most

$$\sum_{j=1}^{q_i} \sum_{x \in \Gamma(\bar{C}_i \cup G_j^i, V^{ij})} 3 \cdot \hat{d}_K(\{x\} \cup \bar{C}_i \cup G_j^i)$$

substantial tests¹. Since $\bar{C}_i \cup G_j^i$ is free and $x \in \Gamma(\bar{C}_i \cup G_j^i, V^{ij})$ with $\Gamma(\bar{C}_i \cup G_j^i, V^{ij}) \cap \Gamma(\bar{C}_i) = \emptyset$, we receive

$$\hat{d}_K(\{x\} \cup \bar{C}_i \cup G_j^i) = d_{ijx} + d_K(\{x\}, \bar{C}_i, G_j^i)$$

and so the algorithm enters step 3 only if

$$\sum_{j=1}^{q_i} \sum_{x \in \Gamma(\bar{C}_i \cup G_j^i, V^{ij})} d_{ijx} = \sum_{j=1}^{q_i} e_K(G_j^i, V^{ij}) \leq 4 \cdot d^{\frac{2}{3}} + 2 \cdot d_K(\bar{C}_i).$$

Hence, there are at most

$$\begin{aligned}
&\sum_{j=1}^{q_i} \sum_{x \in \Gamma(\bar{C}_i \cup G_j^i, V^{ij})} 3 \cdot \hat{d}_K(\{x\} \cup \bar{C}_i \cup G_j^i) \\
&= 3 \cdot \sum_{j=1}^{q_i} \sum_{x \in \Gamma(\bar{C}_i \cup G_j^i, V^{ij})} d_{ijx} + 3 \cdot \sum_{j=1}^{q_i} \sum_{x \in \Gamma(\bar{C}_i \cup G_j^i, V^{ij})} d_K(\{x\}, \bar{C}_i, G_j^i)
\end{aligned}$$

¹Plus those substantial tests that are already counted in (3.14).

$$\begin{aligned}
&= \mathcal{O}\left(d_K(\bar{C}_i) + d^{\frac{2}{3}}\right) + 3 \cdot \underbrace{\sum_{j=1}^{q_i} d_K(\Gamma(\bar{C}_i \cup G_j^i, V^{ij}), \bar{C}_i, G_j^i)}_{\leq d_K(\bar{C}_i)} \\
&= \mathcal{O}\left(d_K(\bar{C}_i) + d^{\frac{2}{3}}\right)
\end{aligned} \tag{3.41}$$

substantial tests¹ in step 3. It applies again that

$$\sum_{i=1}^{\bar{p}} d_{new}(\bar{C}_i) \leq d$$

and therefore

$$\begin{aligned}
&\sum_{i=1}^{\bar{p}} \left[\overbrace{\mathcal{O}\left(d_{new}(\bar{C}_i) + d_K(\bar{C}_i) + d^{\frac{2}{3}}\right)}^{\text{steps 1 and 2}} + \overbrace{\mathcal{O}\left(d_K(\bar{C}_i \cup d^{\frac{2}{3}})\right)}^{\text{step 3}} \right] \\
&= \mathcal{O}(d).
\end{aligned} \tag{3.42}$$

Thus, we obtain:

Theorem 3.15. *Let $H = (V, E)$ be a 3-uniform hypergraph with d defective edges where d is known. Then, it exists a search algorithm that finds all defective edges in H by at most*

$$d \cdot \left\lceil \log_2 \left(\frac{|E|}{d} \right) \right\rceil + \mathcal{O}(d)$$

tests.

¹Plus those substantial tests that are already counted in (3.14).

Chapter 4

Conclusions and Outlook

In this thesis, we have examined a lower and an upper bound for the group testing problem on hypergraphs. We have shown, that for hypergraphs of rank r there is an algorithm that finds all defective edges by at most

$$d \cdot \log_2 |E| + c \cdot d^{\frac{r}{2}}$$

tests for some constant $c > 0$. On the other hand, we have constructed a hypergraph of rank r , namely $H_{r,n}$ (compare: p. 7), with a defective edge set D , $|D| = d$, such that in the worst case no search algorithm can do better than using at least

$$d \cdot \log_2 |E| + \frac{1}{2} \left(\frac{2}{r} \right)^{\frac{r}{2}} \cdot d^{\frac{r}{2}}$$

tests.

To prove the upper bound, we have introduced two algorithms. The first one works only for hypergraphs of rank ≤ 3 , in return it provides a much better constant than the second algorithm, which works for any hypergraphs of bounded rank.

Future work might shrink the gap between upper and lower bound by a more lavishly algorithm and a finer estimation.

In Chapter 3, we have proven a conjuncture of Du and Hwang for 3-uniform hypergraphs. Although we have become quite convinced that the conjuncture of Du and Hwang is true for r -uniform hypergraphs, the question remains unsolved for $r \geq 4$.

Thus we narrow the conjuncture of Du and Hwang to:

Conjecture 4.1. *Let H be a r -uniform hypergraph, then*

$$c(H, d) = d \cdot \left\lceil \log_2 \left(\frac{|E|}{d} \right) \right\rceil + O(d).$$

With a lot of effort, one could probably adopt the algorithm for 3-uniform graphs to find all defective edges in 4- or 5-uniform hypergraphs by inventing additional stages and generalizing the partitioning lemmas (Lemma 3.2 to Lemma 3.6). But for a general proof, it seems more reasonable to look for a different approach.

The main idea of all algorithm that we have presented is the following: First of all, partition the vertex set V of a given hypergraph $H = (V, E)$ into free sets and secondly, combine these sets to construct tests that cover all edges.

The dimension of the number of sets that are at least needed to partition V into free sets depends upon the cardinality of the smallest edges in H , while the number of sets that we have to combine depends on the cardinality of the largest edges in H , the rank of H .

Hence, we conjecture:

Conjecture 4.2. *Let H be a hypergraph of rank r and let ℓ be the smallest cardinality of edges in H , then*

$$c(H, d) = d \cdot \left\lceil \log_2 \left(\frac{|E|}{d} \right) \right\rceil + O(d^{\frac{r}{\ell}}).$$

A proof of this statement would imply all our results.

Bibliography

- [Aig86] Martin Aigner. Search problems on graphs. Discrete Applied Mathematics, 14, 1986.
- [Aig88] Martin Aigner. Combinatorial Search. Wiley-Teubner, New York-Stuttgart, 1988.
- [AT93] Ingo Althöfer and Eberhard Triesch. Edge search in graphs and hypergraphs of bounded rank. Discrete Mathematics, 115(1-3):1–9, 1993.
- [Ber89] Claude Berge. Hypergraphes. North-Holland, Amsterdam [u.a.], 1989.
- [Bol98] Béla Bollobás. Modern graph theory. Springer, New York [u.a.], 1998.
- [CH80] Gerard J. Chang and F. K. Hwang. A group testing problem. SIAM Journal on Algebraic and Discrete Methods, 1(1):21–24, 1980.
- [CH81] Gerard J. Chang and Frank K. Hwang. A group testing problem on two disjoint sets. SIAM J. Algebraic Discrete Methods, 2(1):35–38, March 1981.
- [CH07] Ting Chen and Frank K. Hwang. Note: A competitive algorithm in searching for many edges in a hypergraph. Discrete Appl. Math., 155(4):566–571, February 2007.
- [Che11] Ting Chen. A revised algorithm for searching for all defective edges in a graph. Discrete Applied Mathematics, 159(18):2266–2268, 2011.
- [CHL82] Gerard J. Chang, Frank K. Hwang, and S. Lin. Group testing with two defectives. Discrete Applied Mathematics, 4:97–102, 1982.
- [Dam94] Peter Damaschke. A tight upper bound for group testing in graphs. Discrete Applied Mathematics, 48(2):101–109, 1994.
- [DH93] Ding-Zhu Du and Frank K. Hwang. Combinatorial Group Testing and Its Applications (Applied Mathematics). World Scientific Publishing Company, 1993.
- [Dor43] Robert Dorfman. The detection of defective members of large populations. Annals of Mathematical Statistics, pages 436–440, 1943.
- [Hwa05] Frank K. Hwang. A competitive algorithm to find all defective edges in a graph. Discrete Applied Mathematics, 148:273–277, June 2005.
- [Joh02] Petra Johann. A group testing problem for graphs with several defective edges. Discrete Applied Mathematics, 117(1-3):99–108, 2002.

- [Kor07] Torsten Korneffel. On Combinatorial Search Problems Which Involve Graphs. PhD thesis, RWTH Aachen, Aachen, 2007.
- [KT08] Torsten Korneffel and Eberhard Triesch. An iterative algorithm to find defective edges in a graph. Preprint, 2008.
- [Tos80] Ratko Tasic. An optimal search procedure. Journal of Statistical Planning and Inference, 4(2):169–171, 1980.
- [Tos82] Ratko Tasic. An optimal group-testing procedure. Studia Scientiarum Mathematicarum Hungarica, pages 319–323, 1982.
- [Tri96] Eberhard Triesch. A group testing problem for hypergraphs of bounded rank. Discrete Applied Mathematics, 66(2):185–188, 1996.

List of notations

$ A $	cardinality of a set A , 1
$ e $	cardinality of an edge e , 1
$>$	total order, 4
$\lceil x \rceil$	the smallest integer not less than x , 2
2^V	$\{e : e \subseteq V\}$, 1
$\binom{V}{2}$	$\{e \subset V : e = 2\}$, 7
A_x	$\{e \in E(X) : x \text{ is left endvertex of } e\}$, 4
$c(\mathcal{S}, \mathcal{F})$	worst-case complexity of $(\mathcal{S}, \mathcal{F})$, 2
$c(H, d)$	worst-case complexity of finding d defective edges in H , 3
d	number of defective edges, 9
D	set of defective edges, 9
d_K	number of known defective edges, 9
D_K	set of known defective edges, 9
$D(X)$	$\{e \in D : e \cap X \neq \emptyset\}$, 10
$d(X)$	cardinality of $D(X)$, 10
$D_K(X)$	$\{e \in D_K : e \cap X \neq \emptyset\}$, 10
$d_K(X)$	cardinality of $D_K(X)$, 10
$\dot{D}(X)$	$\{e \in D : e \subset X\}$, 10
$\dot{d}(X)$	cardinality of $\dot{D}(X)$, 10
$\dot{D}_K(X)$	$\{e \in D_K : e \subset X\}$, 10
$\dot{d}_K(X)$	cardinality of $\dot{D}_K(X)$, 10
$D(X, Y)$	$\{e \in \dot{D}(X \cup U) : e \cap X \neq \emptyset\}$, 10
$d(X, Y)$	cardinality of $D(X, Y)$, 10
$D_K(X, Y)$	$\{e \in \dot{D}_K(X \cup U) : e \cap X \neq \emptyset\}$, 10
$d_K(X, Y)$	cardinality of $D_K(X, Y)$, 10
$D(X, Y, Z)$	$\{e \in \dot{D}(X \cup Y \cup Z) : e \cap X = e \cap Y = e \cap Z = 1\}$, 10
$d(X, Y, Z)$	cardinality of $D(X, Y, Z)$, 10
$D_K(X, Y, Z)$	$\{e \in \dot{D}_K(X \cup Y \cup Z) : e \cap X = e \cap Y = e \cap Z = 1\}$, 10
$d_K(X, Y, Z)$	cardinality of $D_K(X, Y, Z)$, 10
$E = E(H)$	edge set (of hypergraph H), 1
$E(S)$	$\{e \in E : e \subset S\}$, 1

$e_K(X)$	$ \{e \in D_K \mid \exists v \in \Gamma(X), \text{ with } e \subset \{v\} \cup X\} $, 48
\mathcal{F}	test family, 1
\mathcal{F}_{mon}	set of monotone tests, 2
$\tilde{f}_t(v)$	$ \tilde{\mathcal{F}}_t(v) $, 35
$\tilde{\mathcal{F}}_t(v)$	$\{k : v \in F_k^t\}$, 35
$\Gamma(X)$	$\{y \in V \setminus X : \exists e \in D, e \subset \{y\} \cup X\}$, 10
$\Gamma_K(X)$	$\{y \in V \setminus X : \exists e \in D_K, e \subset \{y\} \cup X\}$, 10
$H(V, E)$	hypergraph on vertex set V with edge set E , 1
$L(A, x^*)$	number of tests to determine x^* , 2
$\mathcal{N}_K(X)$	set of d_K -neighbors of X , 30
$\mathcal{P}_K^k(V)$	set of all strongly D_K -independent vertex sets of cardinality k , 30
$r(H)$	rank of hypergraph H , 1
\mathcal{S}	search domain, 1
$(\mathcal{S}, \mathcal{F})$	search process, 1
$V = V(H)$	vertex set of hypergraph H , 1
X_x	$\{y \in X : \exists \{x, y\} \in E(X) \text{ and } x > y\}$, 4

Index

Index

- D_K -independent, 9
- d_K -neighbour, 30
- r -uniform, 1
- (successful) search algorithm, 1
- strongly D_K -independent, 30

- adjacent, 1
- alphabetic search, 2
- alphabetically sorted, 4

- binary search, 1

- combinatorial search process, 1
- complete graph, 1

- defective element, 2

- edge, 1

- free vertex set, 5

- good element, 2
- graph, 1
- group testing problem, 2

- halving procedure, 2
- hyperedge, 1
- hypergraph, 1

- independent, 1
- information-theoretic bound, 2

- joined by an edge, 1

- known defective edge, 9

- large vertex, 77
- left / right endvertex, 4
- loop, 1
- lying right of a vertex, 4

- negative test, 2

- neighbor, 1

- positive test, 2
- predetermined, 1

- questions, 1

- rank of a hypergraph, 1
- reasonable test, 10
- rightmost set, 4

- search domain, 1
- search process, 1
- selectable vertex set, 4
- sequential algorithm, 1
- simple hypergraph, 1
- strongly independent, 1

- test, 1
- test family, 1

- vertex, 1
- vertex cover, 1

- worst-case complexity, 2