

# Preface

The topics of my research in mathematics have varied a lot over time as a PhD student. Beginning in Umeå with extremal graph theory, then combining graph theory with image analysis in order to do image segmentation and finally investigating and improving the modified perceptron algorithm, an algorithm that can be used for segmentation.

The results in these different subjects have especially one property in common, that is that they are mainly proved with different discrete methods. Sometimes, as in the case of the geometric properties of the unit sphere in the part concerning the perceptron algorithm, some standard analysis is used and concerning the results about a general perceptron algorithm we get a constructive method of how to achieve a solution to a problem taken from functional analysis.

But, the main topics will be discrete, as well in methods used as in the results achieved.

Below is a list of my papers that has had importance for the results presented in the thesis.

- O. Barr, On extremal graphs without compatible triangles or quadrilaterals, *Discrete Mathematics* **125** (1994) 31–43.

This paper studies a concept of a more general edge colouring, local edge colourings. The results is about how many edges in a graph that will enforce a cycle of length three or four to be compatible with the local edge colouring.

- O. Barr, *Erdős-Sós Conjecture for Graphs with High Minimum Degree*, Research reports, No. 7 (1996), Umeå University, Sweden.

This paper studies the well known conjecture by P. Erdős and V. Sós. Here we get an extension of Sidorenkos earlier result and thereby results for graphs with large minimum degree.

- O. Barr, *Some Results in Extremal Graph Theory*, Research reports, No. 10 (1996), Umeå University, Sweden.

My Licentiate thesis that was presented at Umeå University in 1996, containing the two above papers.

- O. Barr, *A Note on the Loebel-Komlós-Sós Conjecture*, Research Reports, No. 14 (1997), Umeå University, Sweden.

---

A first note concerning the conjecture by Loeb, Komlós and Sós. A conjecture that is related to the previous one by Erdős and Sós.

- O. Barr and R. Johansson, *Another Note on the Loeb-Komlós-Sós Conjecture*, Research reports, No 22 (1997), Umeå University, Sweden.

A second note concerning the same conjecture as above. My contribution to this note is to approximately half of its content.

- O. Barr, Properly Coloured Hamiltonian Paths in Edge-coloured Complete Graphs without Monochromatic Triangles, *Ars Combinatorica*, Vol. 50, (1998).

A note showing the similarities between finding a properly coloured hamiltonian path in a complete graph without monochromatic triangles and finding a directed hamiltonian path in a tournament.

- A. Eriksson, O. Barr and K. Åström, Image Segmentation Using Minimal Graph Cuts, in: *Proceedings SSBA* (2006), 45–48.

Here we present some results on how the max-flow min-cut theorem in graph theory can be used for segmenting images combined with using prior information concerning what type of pixels that should belong to a certain class.

- O. Wigelius and O. Barr, *New Estimates Correcting an Earlier Proof of the Perceptron Algorithm to be Polynomial*, ISSN 1403-9338, LUTFMA-5041-2004.

This paper gives correct estimates of the probability that two  $n$ -dimensional unit vectors have an inner product of at least  $1/\sqrt{n}$ . Beside the collaboration between me and Oscar Wigelius in this paper, we are in debt to Jan Gustavsson in Lund, who found a flaw in the first proof and also helped us to produce a correct version of it.

- O. Barr and O. Wigelius, *A Deterministic Perceptron-Rescaling Algorithm*, published in Oskar Wigelius' Licentiate Thesis *Some Results on Online Learning Algorithms*, LUTFMA-2013-2004.

Here we present the general idea of how to transform the modified perceptron algorithm by Dunagan and Vempala from a randomised one to a deterministic one. The main idea of how to achieve this result was mine, even though we helped each other with all the details in the paper.

- O. Barr and O. Wigelius, *A Deterministic Perceptron-Rescaling Algorithm Finding an Optimal Solution*, published in Oskar Wigelius' Licentiate Thesis *Some Results on Online Learning Algorithms*, LUTFMA-2013-2004.

---

As a consequence of the paper above, we show how to find optimal solutions, instead of only feasible ones, with the modified perceptron algorithm. My work in this paper was to prove the correctness of the generalised  $\alpha$ -perceptron and the generalised  $\alpha$ -wobble procedure.

- O. Barr, A Deterministic and Polynomial Modified Perceptron Algorithm, *Computer Science Journal of Moldova*, Vol. 13, No. 3(39), 254–267, (2005).

By rewriting the deterministic modified perceptron algorithm developed in the papers above, the algorithm is here speeded up using the achieved freedom (from earlier results) to change some of the constants within the algorithmic structure.

---

# Acknowledgements

Different kinds of acknowledgements are needed, mathematical ones and social ones. The intersection of these two sets is not empty, but some people who will fall into both categories will perhaps only be mentioned once.

From Umeå I would first of all thank my supervisor Roland Häggkvist who introduced me to numerous graph theoretic problems and encouraged me in the mathematical studies. Among the PhD students in Umeå at that time I especially want to thank Robert Johansson, who contributed to some of the results concerning the Loebel-Komlós-Sós conjecture, but also Nina Rudälv and Tobias Adelgren.

Beside that I want to thank all the members of the informal interdisciplinary coffee discussion group outside the university library in Umeå. I thank them for distracting me from my research and realise there were other things in the world apart from mathematical structures. Thank you.

From Lund I would like to thank my supervisors Viktor Ufnarovski and Kalle Åström. I also would like to thank Amiran Ambroladze for introducing me to the mathematics concerning the perceptron algorithm. And looking back on the work done in this thesis, I remember most the joyful work together with Oskar Wigelius combined with memorable days in both Axelvold and Hanö.

Other mathematical acknowledgements that has to be made is the collaboration with Anders Eriksson in chapter 6 and the help by Jan Gustavsson in making correct estimates in 8.2.1. Thank you. Especially I also would like to express my thanks to Anna Torstensson for being a good friend combined with being an inspiring mathematical colleague introducing me to new areas of research, not presented in this thesis though.

Also, all the staff at Centre for Mathematical Sciences, in particular people involved in the Mathematical Imaging Group, thanks for help of any kind and entertaining discussions during coffee breaks.

Last, but the opposite of least, I want to express my warmest thanks to my family who has helped me with everything during this time (except the mathematics). Emeli, Nils, Erik and Agnes, thank you for being there all the time.



# Contents

<b>1</b>	<b>Graph Theory</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Notation and Definitions . . . . .	2
<b>2</b>	<b>Extremal Graphs without Compatible Triangles or Quadrilaterals</b>	<b>5</b>
2.1	Notation and Definitions . . . . .	6
2.2	Upper and Lower Bounds for Triangles . . . . .	8
2.2.1	Upper Bound for the Case $s=1$ . . . . .	12
2.3	Upper and Lower Bounds for Quadrilaterals . . . . .	16
2.3.1	Remarks . . . . .	20
<b>3</b>	<b>Erdős-Sós Conjecture for Graphs with High Minimum Degree</b>	<b>23</b>
3.1	Introduction . . . . .	23
3.2	Earlier Results on the Erdős-Sós Conjecture . . . . .	24
3.3	Extension of Sidorenkos Result . . . . .	26
3.4	Erdős-Sós Graphs with High Minimal Degree . . . . .	30
3.5	Trees in Coloured Graphs . . . . .	37
3.5.1	Heuristics about Graph Colouring . . . . .	39
3.5.2	Definitions of Some Classes of Trees . . . . .	40
<b>4</b>	<b>The Loebel-Komlós-Sós Conjecture</b>	<b>41</b>
4.1	Introduction . . . . .	41
4.2	Results on the Loebel-Komlós-Sós Conjecture . . . . .	42
4.2.1	Caterpillars with Many Legs . . . . .	42
4.2.2	Trees with Diameter 4 and Low Central Degree . . . . .	43
4.2.3	Caterpillars with All Legs at One Vertex . . . . .	43
4.2.4	Caterpillars with Diameter at most 5 . . . . .	44
4.2.5	Trees with Diameter at most 4 . . . . .	45
<b>5</b>	<b>Paths in Complete Graphs</b>	<b>47</b>

<b>6</b>	<b>Graphs, Segmentation and Linear Programs</b>	<b>49</b>
6.1	Introduction . . . . .	49
6.2	Segmentation and Classification . . . . .	49
6.3	Maximal Flow and Minimal Cut . . . . .	50
6.3.1	Introduction . . . . .	50
6.3.2	Theory of Graph Cuts . . . . .	50
6.3.3	The Image Seen as a Graph . . . . .	51
6.3.4	Image Descriptors, Pixel Models and Prior Knowledge . . . . .	53
6.3.5	Experiments . . . . .	54
6.3.6	Conclusions . . . . .	55
6.3.7	Graph Theory and Linear Programs . . . . .	56
<b>7</b>	<b>The Perceptron Algorithm and Linear Programs</b>	<b>59</b>
7.1	Introduction . . . . .	59
7.1.1	A Brief History of the Perceptron Algorithm . . . . .	59
7.1.2	The Notion of Margin . . . . .	61
7.2	Reduction of a Feasible Solution to an LP to a Hyperplane Through Origin with Positive Examples . . . . .	61
7.3	A Classical Theorem Concerning the Perceptron Algorithm . . . . .	63
7.4	The Classical Problem Concerning the XOR-function . . . . .	65
7.5	Neural Networks . . . . .	66
7.6	Linear Programs . . . . .	67
7.6.1	Computational Complexity . . . . .	67
7.6.2	The Case of Semidefinite Programming . . . . .	68
<b>8</b>	<b>Results on the Perceptron Algorithm</b>	<b>71</b>
8.1	Geometry Questions for the Modified Perceptron Algorithm . . . . .	71
8.1.1	Inner Product Between Two Random Unit Vectors . . . . .	71
8.2	Lower Bound on the Probability of Two Vectors Having Large Inner Product . . . . .	73
8.2.1	$\Pi_n$ is Always Greater than $\frac{1}{2}(1 - \text{erf}(1/\sqrt{2}))$ . . . . .	74
8.3	How to Make the Perceptron Algorithm Polynomial . . . . .	76
8.3.1	Making a Deterministic and Polynomial Perceptron Algorithm . . . . .	77
8.3.2	How to Make it Deterministic . . . . .	78
8.4	How to Speed Up the Perceptron Algorithm . . . . .	78
8.4.1	The Deterministic Modified Perceptron Algorithm . . . . .	79
8.4.2	Parallel Processors . . . . .	86
8.4.3	Expected Time for the Modified Perceptron Algorithm . . . . .	90

8.5	A Generalized Perceptron . . . . .	92
8.5.1	Inner Product Spaces . . . . .	92
8.5.2	Real Inner Product Spaces of Finite Dimension . . . . .	92
8.5.3	Real Inner Product Spaces of Infinite Dimensions . . . . .	93
<b>9</b>	<b>Finding Maximal Margins with the Modified Perceptron Algorithm</b>	<b>97</b>
9.1	Introduction to the Iterated Perceptron-Rescaling Algorithm . . . . .	97
9.1.1	Comparison of Solutions . . . . .	98
9.1.2	Outline of the Algorithmic Strategy . . . . .	98
9.1.3	The Black Box . . . . .	99
9.2	A Deterministic Perceptron-Rescaling Algorithm . . . . .	99
9.3	The Black Box for a General $\alpha$ . . . . .	100
9.3.1	The Perceptron Phase . . . . .	100
9.3.2	The Wiggling Phase . . . . .	102
9.3.3	The Rescaling Phase . . . . .	103
9.4	The Iterated Perceptron-Rescaling Algorithm . . . . .	104
9.4.1	Inside the Black Box . . . . .	104
<b>10</b>	<b>Conclusions</b>	<b>107</b>
10.1	Graph Theory . . . . .	107
10.2	The Perceptron Algorithm . . . . .	108

## CONTENTS

---

## Chapter 1

# Graph Theory

## 1.1 Introduction

In the beginning of my PhD studies the main focus of my research was directed towards graph theory and especially some of the problems in the area of extremal graph theory. In this area you could describe a typical kind of question as "When can you guarantee that a graph  $G$  has a certain property  $P$ ?". The property in question is something that can be decided if a graph has or has not. Examples of this is for instance if it contains a certain subgraph or has a proper vertex colouring with a certain number of colours. The way you guarantee this is by a function  $f$  on the graph  $G$ . Classical formulations in extremal graph theory can be exemplified by the theorem that says that a certain minimal degree  $\delta(G)$  implies that the graph in question must contain a hamiltonian cycle as a subgraph.

**Theorem 1.1.1.** (*Dirac, 1952*)

*A simple graph with at least  $n \geq 3$  vertices and minimal degree  $\delta(G) \geq \frac{n}{2}$  contains a hamiltonian cycle as a subgraph.*

Also we can exemplify by reformulating a classical theorem by Vizing.

**Theorem 1.1.2.** (*Vizing, 1964*)

*A simple graph with maximal degree  $\Delta(G) \leq k - 1$  can always be given a proper vertex colouring using at most  $k$  different colours.*

In these cases the function used on the specified graph  $G$  was the minimal degree  $\delta(G)$ , or only  $\delta$ , and the maximal degree  $\Delta(G)$ , or only  $\Delta$ , respectively. There are many other functions to choose from but in our case we will throughout the thesis look at functions that in one way or another describe the number of edges in proportion to how many vertices there are in the graph.

Extremal graph theory is a branch of graph theory developed by hungarian mathematicians. Its study was initiated by Turán in 1940, although a special case of his theorem and several other extremal results had been proved many years earlier. The main exponent has been Paul Erdős who, through his many papers and lectures, has virtually created the subject.

In extremal graph theory one is interested in relations between various graph invariants, such as order, size, connectivity, minimum degree, maximum degree, chromatic number and diameter, and also in the values of these invariants which ensure the graph to have certain properties.

In the chapters 2-4 of this thesis, results concerning the area of extremal graph theory will be presented. Chapter 2 will introduce the notion of local edge colourings and investigate the size of a graph, depending on the order of the graph, that is needed to ensure triangles and quadrilaterals respectively in the graph to be compatible with any local edge colouring contained in a certain family of colourings. This result has been published in *Discrete Mathematics* and been presented at the British Combinatorial Conference (BCC), [9].

Chapter 3 contains a study of the following well-known conjecture of Paul Erdős and Vera Sós: If a graph  $G$  of order  $n$  has a size strictly greater than  $\frac{n}{2}(k-1)$ , then every tree of size  $k$  is contained in  $G$  as a subgraph. By adding extra conditions to the conjecture concerning the minimum degree  $\delta$  of the graph  $G$ , we solve the conjecture for a family of graphs and also answer the conjecture for every  $k \leq 9$  in the affirmative. This result was published in a preprint, [10], and the above two papers together were presented as my Licentiate thesis, [11], at Umeå University in 1996.

Chapter 4 contains a study of the conjecture by Loebel, Komlós and Sós assuming the following statement to be true: Any simple graph  $G$  of order  $n$  with at least  $\frac{n}{2}$  vertices of degree at least  $k$ , contains each tree of size  $k$  as a subgraph. Here we will collect the results from two notes, the first one written by myself, [12], and the second one written together with Robert Johansson at Umeå University, [13]. The results from the second note can be estimated to be mine to an extent of approximately fifty percent.

Also a fourth short chapter concerning graph theory is included in this thesis, chapter 5. It contains a result concerning a sufficient condition on an edge colouring of a complete graph ensuring us to find a properly coloured hamiltonian path. This result has been published in *ARS Combinatoria* [14] and presented at British Combinatorial Conference. It is an often cited fact in surveys concerning this area of graph theory, see for example [44].

## 1.2 Notation and Definitions

To avoid confusion in the graph theoretic chapters we will first recall what can be said to be standard notation, a notation that will be used throughout these chapters. First of all we have the concept of a graph, usually denoted  $G$ . A graph consists of two sets, a vertex set denoted  $V(G)$  and an edge set denoted  $E(G)$ , where the edge set consists of unordered pairs from the vertex set  $V(G)$ . We write this as  $G = (V(G), E(G))$ .

We will only consider finite graphs, meaning that the vertex set  $V(G)$  is finite. On top of this we only consider simple graphs, meaning that we do not allow neither loops nor multiple edges in our edge set  $E(G)$ . The size of the vertex set  $V(G)$  is called the order of the graph, often denoted by  $n$ , and the size of the edge set  $E(G)$  is called the size of the graph, usually denoted by  $\varepsilon(G)$  or only  $\varepsilon$ .

In many cases it is of great interest to know how many edges in  $E(G)$  that have an endpoint in a specified vertex  $v$  from the vertex set  $V(G)$ . This number is called the

degree of the vertex  $v$  and is denoted  $d(v)$ . Sometimes we will be interested in how large or how small the degrees in a specified graph  $G$  can be and we introduce  $\delta(G)$  and  $\Delta(G)$  to represent the minimal respectively the maximal degree that can be achieved in  $G$ . If it is clear from the context what graph  $G$  we are discussing we will use the notation  $\delta$  and  $\Delta$  instead. The mean degree in the graph  $G$  will be denoted  $\overline{d(v)}$ .

Using the notation above, we can introduce the notion of degree of an edge. If  $\{u, v\}$  is an unordered pair in  $E(G)$  we usually denote this edge  $e$  as  $uv$ . The degree of an edge is defined by the sum of the degree of its endpoints, so  $d(e) = d(uv) = d(u) + d(v)$ . Here we will also make use of the notation  $\overline{d(e)}$  to denote the mean degree of the edges in the studied graph.

Now, if we study a subset  $A$  of the vertex set  $V(G)$  we have the following notation:  $E(A)$  is the set of edges induced by  $A$ ; the set of edges that have both ends in  $A$ . Here we denote the size of  $E(A)$  with  $\varepsilon(A)$ . With the induced graph on  $A$ , we mean the graph  $(A, E(A))$ . If we have two disjoint subsets,  $A$  and  $B$  say, from the vertex set  $V(G)$ , then  $\varepsilon(A, B)$  denotes the number of edges between  $A$  and  $B$ , or in other words, the number of edges with one end in  $A$  and one end in  $B$ .

Also we will make use of the concept neighbour in a graph. If  $A$  is a subset of the vertex set, the neighbours to  $A$ , denoted  $N(A)$ , will be defined by those vertices in  $V(G) \setminus A$  such that there is an edge  $uv$  where  $u \in A$  and  $v \in V \setminus A$ .

Other useful notations are the following:  $K_n$  denotes a complete graph on  $n$  vertices, where every possible edge is contained in the edge set.  $K_{m,n}$  denotes a complete bipartite graph on  $m + n$  vertices. Here the vertex set is divided into two parts, containing  $m$  and  $n$  vertices respectively. The edge set contains all possible edges with one end in each vertex set.  $\mathbb{C}_r$  denotes a cycle in a graph on  $r$  vertices. Especially we have that  $\mathbb{C}_3 = K_3$  and  $\mathbb{C}_4 = K_{2,2}$ .

Further on, a *forest* will be an acyclic graph; a graph containing no cycles. A *tree* will be a connected forest; a connected acyclic graph. Note that a tree on  $k$  edges has  $k + 1$  vertices, or in other words, a tree of size  $k$  has order  $k + 1$ .

For the reader not familiar with certain classes of trees mentioned later on in the thesis, we will present proper definitions of stars, double-stars, comets and spiders.

**Definition 1.2.1.** (star)

A star on  $k$  edges is the complete bipartite graph  $K_{1,k}$ .

**Definition 1.2.2.** (double-star)

A double-star with distance  $r$  between the centres is a tree obtained from two stars and a path by identifying the leaves of a path of length  $r$  to the centre of each star. If  $r = 1$  we say that the tree is a double-star omitting the length of the path between the centres of the two stars.

**Definition 1.2.3.** (comet)

A comet is a tree obtained from a star and a path by identifying one leaf of the star with one leaf of the path.

**Definition 1.2.4.** (spider)

A spider is a tree obtained from a star by subdividing its edges.

Finally we introduce the following notation for the number of cycles in a graph  $G$  with different restrictions. Let  $\tau(k)$  denote the number of cycles in  $G$  of length  $k$ ,  $\tau_A(k)$  the number of cycles in  $G$  of length  $k$  with at least one vertex in the subset of vertices  $A \subset V(G)$  and  $\tau^B(k)$  the number of cycles of length  $k$  with at least one edge in the subset of edges  $B \subset E(G)$ .

## Chapter 2

# Extremal Graphs without Compatible Triangles or Quadrilaterals

The results presented here concerns a variant of a standard type of problem in extremal graph theory. The standard problem is the following: Given a graph  $H$ , determine the maximal number of edges in a graph  $G$  on  $n$  vertices without a copy of  $H$ . Or you could turn things around and ask for the minimal number of edges in  $G$  that still enforces  $H$  to be contained in the studied graph  $G$  as a subgraph. We will be studying our problems from both of these perspectives, in order to achieve upper and lower bounds on the number of edges needed to enforce a copy of  $H$  in  $G$ .

In the problems considered here we will allow copies of  $H$  in  $G$  according to the following rule: For each pair of incident edges in  $G$  we decide in advance whether this pair of edges is an allowed transition (pair) to use or not. After this preliminary step we look for a copy of  $H$  satisfying the restriction that no pair of incident edges in  $H$  forms a forbidden transition. The family of pairs of edges which we allow in the copy of  $H$  forms what is called a transition system in  $G$ .

A natural question arises: What is the smallest number of edges we need in the graph  $G$  of order  $n$  (no matter where they are situated and which transition system  $X$  in some family of transition systems  $\mathcal{X}$  we impose on  $G$ ) to assure us of having a copy of  $H$  in  $G$  where all the paths in the copy of  $H$  are allowed by the transition system  $X$ .

We can also ask us the question how to construct a graph  $G$  of order  $n$  and decrease the size by at least one and then find a transition system  $X$  in a specified family of transition systems  $\mathcal{X}$  such that all copies of  $H$  in  $G$  have at least one path forbidden according to the given transition system  $X$ .

The above questions are deeply dependent on how we define  $\mathcal{X}$ , the family of transition system for a general graph  $G$ . Here we will consider transition systems defined through local edge colourings. This will be defined in the next section.

One common way to obtain a transition system is to partition the set of edges  $E(G)$  into a number of parts,  $E_1, E_2, \dots, E_m$ , and prescribe that forbidden transitions (paths of length two) are exactly those pairs of incident edges that belong to the same part  $E_i$  in the partition. This way of constructing a transition system is equivalent to an ordinary edge colouring where two incident edges must have different colours to be counted as an allowed transition. Later on, it will be clear that this is a more restricted family of transition system than those obtained through a local edge colouring, which we will deal with below.

There has been substantial research in this area before and below you will find some of the results that have most similarities with the work presented here. Chen and Daykin,

in [30], give a condition on  $n$  and  $r$  so that  $K_n$ , the complete graph on  $n$  vertices, and  $K_{r,r}$ , the complete bipartite graph on  $r$  plus  $r$  vertices, contain an  $X$ -compatible cycle of every possible length; here  $X$  is derived from an edge colouring where each vertex is incident with at most  $k$  edges of the same colour (in an  $X$ -compatible cycle two incident edges are of distinct colours). For the case  $k = 2$ , (plus some bounds on  $n$  to take care of small degenerated cases), Daykin shows in [33] that  $K_n$  contains an alternating cycle of every possible length.

Bankfalvi and Bankfalvi give in [8] a condition on a two-colouring of  $K_{2n}$  which is satisfied if and only if this complete graph on order  $2n$  contains an alternating hamiltonian cycle.

G. Sabidussy has conjectured the existence of an  $X$ -compatible cycle decomposition, when  $X$  is derived from an eulerian tour. (I have no direct reference for this conjecture; it is mentioned by Fleischner in [41].) Also to be mentioned is Hellgren's result in [47] that if  $\delta \geq \frac{1}{2}n$  then  $(G_n, X_{n/16})$  has a compatible 2-factor.

For the reader looking for a closer examination of transition systems in general, we recommend Fleischner's book [41].

Here we will consider the cases when  $H$  is  $\mathbb{C}_3 = K_3$  or  $\mathbb{C}_4 = K_{2,2}$ , a triangle or a quadrilateral, and  $\mathcal{X}$  is the family of transition systems of the form  $\mathcal{X}_s$ , which will be defined below.

The main results consist of giving upper and lower bounds for  $\varepsilon = |E(G)|$  where  $G$  is the extremal graph of order  $n$  with no  $\mathcal{X}_s$ -compatible triangle or cycle of length four, respectively.

## 2.1 Notation and Definitions

Apart from a few new definitions we will use the standard notation presented in the previous chapter.

**Definition 2.1.1.** (Transition System Based on Local Edge Colourings)

Let  $G = (V(G), E(G))$  be a simple graph of order  $n$ . A local edge colouring  $X(v)$  at a vertex  $v \in V(G)$  is a partition of the edges incident to  $v$ . A transition system for  $G$ , defined through local edge colourings, is defined by

$$X = \bigcup_{v \in V(G)} X(v).$$

Moreover, if for all vertices  $v \in V(G)$  the sets in the partition  $X(v)$  are of cardinality at most  $s + 1$ , we say that  $X$  is of type  $s$ . Such a transition system is denoted  $X_s$  and belongs to the family  $\mathcal{X}_s$  containing all transition systems of type  $s$ .

We also have to describe what types of transitions (paths of length two) that become forbidden under the restriction of our transition system.

**Definition 2.1.2.** (Forbidden Transitions)

Let  $A_1, A_2, \dots, A_r$  be the partition  $X(v)$ , the partition of vertices incident to the vertex  $v$ . A path  $uvw$  in the graph  $G$  is said to be forbidden if  $\{vu, vw\} \subseteq A_i \in X(v)$ . The set of all forbidden transitions in  $G$  with a specified transition system  $X$  is denoted by  $F(G, X)$ . The statement 'xyz and uyv are forbidden transitions' is depicted in the figure below.

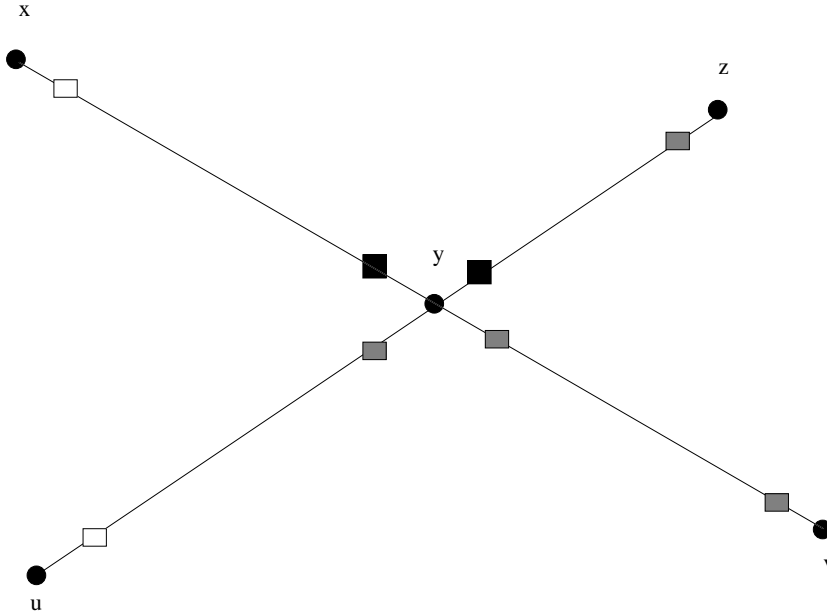


Figure 2.1: The paths  $xyz$  and  $uyv$  are forbidden transitions, indicated with local edge colours surrounding the vertex  $y$ .

**Definition 2.1.3.** (Compatible with a Transition System)

A subgraph  $H \subseteq G$  is said to be compatible with the transition system  $X$ , written  $H \subseteq (G, X)$ , if no paths of length two in  $H$  are forbidden.

**Remark 1.**  $F(G, X) = \emptyset$  when  $X$  is contained in the family  $\mathcal{X}_0$ . This is the same as saying that no forbidden transitions will be induced from a partition emerging from a proper (local) edge colouring. Or, in other words, in a proper edge colouring, all edges that meet at a vertex have different colours. In this case no forbidden pairs, of incident edges having the same colour, will emerge.

**Remark 2.** The reason for cardinality at most  $s+1$  in a colouring contained in the family  $\mathcal{X}_s$ , is that for every edge, incident to a specified vertex  $v$ , there is at most  $s$  forbidden ways, when passing the vertex  $v$  to another edge.

**Definition 2.1.4.** Let  $f(n, H, \mathcal{X})$  denote the maximal number of edges in a graph of order  $n$ , such there exists a transition system  $X \in \mathcal{X}$ , such that  $G$  does not contain a subgraph  $H$  compatible with the transition system  $X$ , i.e.

$$f(n, H, \mathcal{X}) = \max_{G, X \in \mathcal{X}, H \not\subseteq (G, X)} \varepsilon(G),$$

where  $\varepsilon(G) = |E(G)|$ .

Later on we will do some constructions of graphs to help us estimate the above expression. Therefore we need a few other definitions as well. First we need the concept of blowing up a graph by replacing every vertex by  $s$  vertices and replace every edge by a complete bipartite graphs. A proper definition of this concept is:

**Definition 2.1.5.**  $G(s)$  is a graph on  $ns$  vertices according to the following rule:

$$V(G(s)) = \cup_{i=1}^n \cup_{k=1}^s v_i^k$$

and the edges in  $E(G(s))$  are defined through the following:

$$v_i v_j \in E(G) \quad \text{if and only if} \quad v_i^k v_j^l \in E(G(s))$$

for all  $k, l \in \{1, 2, \dots, s\}$ .

Also we would like to add, to the previous construction, edges such that each vertex in the original graph  $G$  will be replaced by a complete graph on  $s$  vertices. More formally:

**Definition 2.1.6.**  $G^*(s) = (V(G(s)), E(G(s)) \cup A)$  where

$$A = \cup_{i=1}^n A_i \quad \text{and} \quad A_i = \{v_i^k v_i^l; k \neq l\}.$$

## 2.2 Upper and Lower Bounds for Triangles

In this section we will give upper and lower bounds for  $f(n, \mathbb{C}_3, \mathcal{X}_s)$ . First of all we need a simple estimate of how many forbidden transitions there can be in total in a graph with a transition system  $X$  taken from the family  $\mathcal{X}_s$ .

**Lemma 2.2.1.** *The number of forbidden transitions for a graph  $G$  with a transition system from  $\mathcal{X}_s$  is at most  $s\varepsilon$ , or*

$$|F(G, X)| \leq s\varepsilon.$$

*Proof:* For every vertex  $v_i$  in  $V(G)$  choose  $b_i$  and  $r_i$  such that  $d(v_i) = b_i(s + 1) + r_i$  and  $0 \leq r_i \leq s$ . We get that

$$\begin{aligned} |F(G, S)| &\leq \sum_{i=1}^n b_i \binom{s+1}{2} + \binom{r_i}{2} = \frac{1}{2} \sum_{i=1}^n (b_i(s+1)s + r_i(r_i - 1)) = \\ &= \frac{1}{2} \sum_{i=1}^n (d(v_i) - r_i)s + r_i(r_i - 1) \leq \frac{s}{2} \sum_{i=1}^n d(v_i) = s\varepsilon. \end{aligned}$$

□

In the estimates made below we also need to have a lower bound on how many different triangles that use a specified edge  $e$  in a simple graph of order  $n$ . Here we let  $e$  be a short notation for the set  $\{e\}$  in order to use the above definition in a proper way.

**Lemma 2.2.2.** *If  $d(e) \geq n + m$ , where  $m$  is a positive integer, then  $\tau^e(3) \geq m$ .*

*Proof:* Let the edge  $e$  have ends  $u$  and  $v$ , and let  $d(e) \geq n + m$  for a positive integer  $m$ . We have that  $\varepsilon(\{u, v\}, \{V(G) \setminus \{u, v\}\}) \geq n - 2 + m$  and  $|V(G) \setminus \{u, v\}| = n - 2$ . Since there are no multiple edges in our simple graph  $G$ , there will be at least  $m$  vertices in  $V(G) \setminus \{u, v\}$  which are neighbours to both  $u$  and  $v$ . Thus we have that  $\tau^e(3) \geq m$  and the proof is finished. □

Now, before we prove an upper bound for  $f(n, \mathbb{C}_3, \mathcal{X}_s)$ , we also need a lemma that gives us a lower bound for the mean edge degree that enforces a compatible triangle in the graph  $G$ .

**Lemma 2.2.3.** *If  $X$  is a transition system from  $\mathcal{X}_s$  and  $\overline{d(e)} > (n + 3s)$ , then  $(G, X)$  contains a compatible triangle.*

*Proof:* Let  $\overline{d(e)} > (n + 3s)$ , so that in particular

$$\sum_{e_i \in E(G)} (d(e_i) - n) > 3s\varepsilon.$$

This implies that each edge belongs to more than  $3s$  triangles on average using the previous lemma. Since every triangle is counted three times, we get that

$$\tau(3) > s\varepsilon \geq |F(G, X)|,$$

when using the first lemma of this section. But each transition in  $X$  can prevent at most one triangle from being compatible with  $X$ . Thus we have a compatible triangle in  $(G, X)$  and the proof is finished. □

Using this lemma, we can now prove an upper bound for the number of edges in a graph of order  $n$  not containing a triangle compatible to a transition system from  $\mathcal{X}_s$ .

**Theorem 2.2.1.**

$$f(n, \mathbb{C}_3, \mathcal{X}_s) \leq \frac{1}{4}n^2 + \frac{3}{4}sn.$$

*Proof:* The proof is done by contradiction.

Let  $X$  be a transition system from  $\mathcal{X}_s$  and suppose there exist positive integers  $n$  and  $s$  such that  $f(n, \mathbb{C}_3, \mathcal{X}_s) > \frac{1}{4}n^2 + \frac{3}{4}sn$ . Take a graph  $G$  and a transition system  $X$  from  $\mathcal{X}_s$  with this property. We know now that

$$\varepsilon > \frac{1}{4}n^2 + \frac{3}{4}sn$$

and that  $(G, X)$  contains no compatible triangles. But this inequality is equivalent to

$$\frac{n}{\varepsilon} \left( \frac{2\varepsilon}{n} \right)^2 > n + 3s,$$

and since  $2\varepsilon/n = \overline{d(v)}$ , we get that

$$\frac{1}{\varepsilon} \sum_{i=1}^n \left( \overline{d(v)} \right)^2 > n + 3s.$$

But since the sum over the separate degrees squared, will become an even larger entity, we get that

$$\frac{1}{\varepsilon} \sum_{i=1}^n d(v_i)^2 > n + 3s.$$

Now let every edge  $e_i$  have ends  $u_i$  and  $w_i$  for all  $i \in \{1, 2, \dots, \varepsilon\}$  and rewrite the sum to

$$\frac{1}{\varepsilon} \sum_{u_i w_i \in E(G)} (d(u_i) + d(w_i)) = \frac{1}{\varepsilon} \sum_{e_i \in E(G)} d(e_i) > n + 3s.$$

This reformulation can be done by using the fact that

$$\sum_{i=1}^n (d(v_i))^2 = \sum_{u_i w_i \in E(G)} (d(u_i) + d(w_i)),$$

which is true since when you sum over all edges degrees, every single vertex degree  $d(v)$  will contribute with its square value to the sum.

This gives that  $\overline{d(e)} > n + 3s$ , which by a previous lemma implies that there exists a compatible triangle in  $(G, X)$  and we have a contradiction.  $\square$

The last theorem is sharp in the case of  $s = 0$ , which is the case when we look for a triangle free graph  $G$  with as many edges as possible and no transition system is considered. The last theorem is also sharp when  $n = 5$  and  $s = 1$  in view of the local edge colouring as seen in figure 2.2.

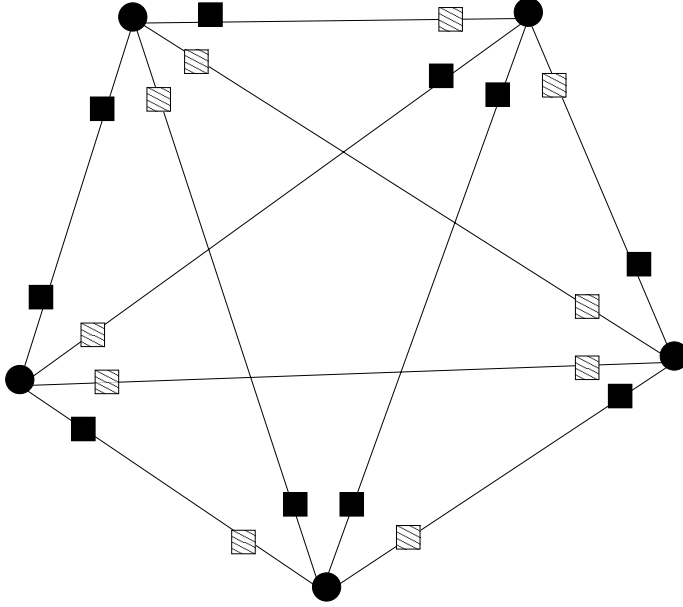


Figure 2.2: Every triangle in this complete graph on five vertices has a forbidden transition, indicated with local colours where edges meet at a vertex.

After this result, yielding an upper bound for  $f(n, \mathbb{C}_3, \mathcal{X}_s)$ , we will prove a lower bound as well. A natural way to achieve a lower bound is by construction. If we can construct a graph  $G$  of order  $n$  and find a transition system  $X$  from  $\mathcal{X}_s$  that make all present triangles forbidden, we have found a lower bound on  $f(n, \mathbb{C}_3, \mathcal{X}_s)$  by construction. This is done in a later proposition with help of the following lemma.

**Lemma 2.2.4.** *If  $G$  is a triangle-free graph, then we can define a transition system  $X$  from  $\mathcal{X}_s$  such that  $(G^*(s+1), X)$  has no triangle compatible with the transition system.*

*Proof:* Take a graph  $G$  without triangles and construct  $G^*(s+1)$  according to the definition above. Define the transition system  $X$  from  $\mathcal{X}_s$  according to the following: For every vertex  $v_i^k \in V(G^*(s+1))$  and every fixed  $j$ , let  $v_i^k v_j^l$  belong to the same set in the partition  $X(v_i^k)$  for every  $l \in \{1, 2, \dots, l\}$ . Assume now that  $(G^*(s+1), X)$  contains a compatible triangle,  $v_{i_1}^{k_1} v_{i_2}^{k_2} v_{i_3}^{k_3}$  say. If all  $i_j$ 's were different, then  $v_{i_1} v_{i_2} v_{i_3}$  would constitute a triangle in  $G$ , which is in conflict with the assumption that  $G$  is triangle-free. Thus, at least two  $i_j$ 's are equal and at least one path of length two in the triangle is a forbidden transition. Consequently,  $(G^*(s+1), X)$  has no compatible triangles.  $\square$

**Proposition 2.2.1.** *For every integer  $n$ , choose integers  $r' \leq s + 1$  and  $r'' \leq s + 1$  such that*

$$n = 2(s + 1) \left\lfloor \frac{n}{2(s + 1)} \right\rfloor + r' + r'' \quad \text{and} \quad |(r' - r'') \bmod (s + 1)| \leq 1,$$

*then it follows that*

$$f(n, \mathbb{C}_3, \mathcal{X}_s) \geq \left\lfloor \frac{n^2}{4} \right\rfloor + 2 \left\lfloor \frac{n}{2(s + 1)} \right\rfloor \binom{s + 1}{2} + \binom{r'}{2} + \binom{r''}{2}.$$

*Proof:* This is done by using the idea from the previous lemma and a complete bipartite graph. First partition the integer  $n$  in two as equal integer parts as possible. Each of these two parts will be subdivided into as many parts containing  $s + 1$  vertices as possible. The remainders will be  $r'$  and  $r''$  respectively. Group the vertices of a graph of order  $n$  according to this partition of the integer  $n$ . Add edges to form a complete graph of every subdivision of  $(s + 1)$ ,  $r'$  and  $r''$  vertices. Then add edges to form a complete bipartite graph between the first two parts that occurred in our partition.

The local edge colouring, that will define our transition system is defined as follows. For every vertex  $v$  in the constructed graph, let the edges  $vu$  and  $vw$ , now only studied at the local view around the vertex  $v$  be in the same partition  $X(v)$  if and only if  $u$  and  $w$  are contained in the very same  $K_{s+1}$ ,  $K_{r'}$  or  $K_{r''}$ . Each colour is represented at most  $s + 1$  times at each vertex and  $(G, X)$  does not contain a compatible triangle according to our previous lemma.

Now we count the number of edges in the constructed graph that has the required properties. First we have the number of edges in a maximal complete bipartite graph, that is  $\lfloor n^2/4 \rfloor$ , then we have the number of edges in a complete graph  $K_{s+1}$  which is  $\binom{s+1}{2}$  occurring  $2\lfloor n/2(s + 1) \rfloor$  times, and finally two complete graphs of size  $r'$  and  $r''$  respectively, contributing with  $\binom{r'}{2}$  and  $\binom{r''}{2}$  edges to the sum.  $\square$

### 2.2.1 Upper Bound for the Case $s=1$

In the main section we made upper and lower bounds for a general  $s$ . In this subsection we will consider the case  $s = 1$  and make a sharper upper bound for this more restricted local edge colouring.

To achieve this we will first prove three preliminary lemmas and one corollary before we can actually prove a sharper upper bound.

The first lemma is about how many edges there can be in the subgraph induced by the neighbours to a vertex  $v$ , if we have a transition system  $X$  from  $\mathcal{X}_s$  and know that there are no triangles in the graph compatible to the transition system.

**Lemma 2.2.5.** *If  $(G, X)$  has no compatible triangle and  $X \in \mathcal{X}_s$ , then  $|E(N(v))| \leq \frac{3}{2}sd(v)$  for every vertex  $v$  in the graph  $G$ .*

*Proof:* Take a graph  $G$  with properties as said in the lemma and pick any vertex  $v$  from  $V(G)$ . Each edge in the edge set  $E(N(v))$ , together with a pair of edges incident to  $v$ , constitutes a triangle with  $v$  as one out of three vertices and, conversely, each triangle having  $v$  as a vertex has exactly one edge in  $E(N(v))$ . At the vertex  $v$  there are at most  $\frac{1}{2}sd(v)$  transitions of type  $xvy$  not compatible with the transition system  $X$  from  $\mathcal{X}_s$ . For each vertex  $u \in N(v)$  there are at most  $s$  transitions of type  $xuv$  not compatible with the transition system  $X$ , since there are at most  $s$  edges incident to  $u$  contained in the same set of the partition  $X(u)$  as  $uv$ . Let  $v$  denote the set  $\{v\}$  below, then we get that

$$\tau_v(3) = \varepsilon(N(v)) \leq \frac{1}{2}sd(v) + sd(v) = \frac{3}{2}sd(v)$$

and the proof is finished.  $\square$

The second lemma gives an upper bound on the maximum degree,  $\Delta$ , of a graph with a transition system,  $(G, X)$ , not containing a compatible triangle, when we have a lower bound of the minimum degree,  $\delta$ , given.

**Lemma 2.2.6.** *Let  $(G, X)$  be a graph, with a transition system  $X$  from  $\mathcal{X}_s$ , not containing a compatible triangle and with a minimum degree  $\delta \geq \frac{1}{2}n + s$ , then is the maximum degree  $\Delta$  bounded above by  $\frac{1}{2}n + 2s$ .*

*Proof:* First we assume that  $\Delta > \delta$ , otherwise the statement is trivial. Now, take a graph with a transition system  $(G, X)$  as in the lemma and pick a vertex  $v \in V(G)$  such that  $d(v) = \Delta = \frac{1}{2}n + s + a$ , where  $a$  is a positive number. Let  $T = V(G) \setminus (N(v) \cup \{v\})$ .

First assume that  $T \neq \emptyset$ . For every vertex  $u \in N(v)$ , we get that

$$\varepsilon(u, N(v) \setminus u) \geq d(u) - |T| - 1 \geq 2s + a$$

and consequently

$$\overline{\varepsilon(u, N(v) \setminus u)} \geq 2s + a,$$

where the mean value is taken over all vertices  $u \in N(v)$ . But since  $\varepsilon(N(v)) \leq \frac{3}{2}s\Delta$  according to the previous lemma, we get that

$$2s + a \leq \overline{\varepsilon(u, N(v) \setminus u)} \leq 3s,$$

which implies that  $a \leq s$  and  $\Delta \leq \frac{1}{2}n + 2s$ .

On the other hand, if  $|T| = \emptyset$ , we have for each vertex  $u \in N(v)$  that

$$\overline{\varepsilon(u, N(v) \setminus u)} \geq \delta - 1 \geq \frac{1}{2}n + s - 1$$

and thus

$$\frac{1}{2}n + s - 1 \leq \overline{\varepsilon(u, N(v) \setminus u)} \leq 3s.$$

This yields that  $n \leq 4s + 2$  and we get  $\Delta \leq n - 1 \leq 4s + 1$  or, as claimed,  $\Delta \leq \frac{1}{2}n + 2s$ , by taking the mean value of the two different upper bounds.  $\square$

The third lemma on our way to an upper bound tells us that too many edges in a graph of even order  $n > 11$ , expressed as lower bounds on  $\delta$  and  $\Delta$ , in a graph with a transition system  $X$  from  $X_1$ , will enforce a compatible triangle in  $(G, X)$ .

**Lemma 2.2.7.** *Let  $G$  be a graph of even order  $n$  and let  $n > 10$ ,  $\delta \geq \frac{1}{2}n + 1$ ,  $X \in X_1$  and  $\Delta > \frac{1}{2}n + 1$ . Then  $(G, X)$  contains a compatible triangle.*

*Proof:* By contradiction. Take a graph  $G$  with properties as in the lemma and assume that it has no compatible triangles. According to the previous lemma we have that  $\Delta \leq \frac{1}{2}n + 2$  but, since  $\Delta > \frac{1}{2}n + 1$ , it must be the case that  $\Delta = \frac{1}{2}n + 2$ .

Pick a vertex  $v \in V(G)$  such that  $d(v) = \Delta$ . Since  $d(u) \geq \frac{1}{2}n + 1$ , we have that  $\varepsilon(u, N(v) \setminus u) \geq 3$  for all vertices  $u \in N(v)$ . But from an earlier lemma we know that  $\varepsilon(N(v)) \leq \frac{3}{2}\Delta$ , which can be rewritten as

$$\overline{\varepsilon(u, N(v) \setminus u)} \leq 3.$$

Thus  $\varepsilon(u, N(v) \setminus u) = 3$  for all  $u \in N(v)$ .

Now, if we take  $T$  as in the previous lemma, we have for each vertex  $u \in T$  that  $\varepsilon(u, T) \geq \frac{1}{2}n - 3$  and consequently

$$\varepsilon(N(v), T) \geq \left(\frac{1}{2}n + 2\right) \left(\frac{1}{2}n - 3\right).$$

On the other hand, we have for each vertex  $u \in T$  that  $\varepsilon(u, N(v)) \leq \frac{1}{2}n + 2 = \Delta$  and we get that

$$\varepsilon(N(v), T) \leq \left(\frac{1}{2}n + 2\right) \left(\frac{1}{2}n - 3\right).$$

This implies that there must be equality in these two inequalities to avoid contradiction and we have that  $d(u) = \frac{1}{2}n + 1$  if  $u \in N(v)$  and  $d(u) = \frac{1}{2}n + 2$  if  $u \in T$ . Moreover,  $T' = \{v\} \cup T$  is an independent set and  $|T'| = \frac{1}{2}n - 2$ .

## 2.2. UPPER AND LOWER BOUNDS FOR TRIANGLES

---

Since there is no triangle compatible with  $X$  in  $G$ , there is especially no compatible triangle with a vertex in  $T'$ . Each edge in  $E(N(v))$  gives exactly  $|T'|$  triangles with a vertex in  $T'$ . At each vertex  $v \in T'$  there are at most

$$\frac{1}{2}d(v) = \frac{1}{2} \left( \frac{1}{2}n + 2 \right)$$

transitions not compatible with  $X$ . For each vertex  $u \in N(v) = N(T')$  there are at most 3 transitions not compatible with  $X$  of type  $xuy$ , where  $x \in N(T')$  and  $y \in T'$ . Thus we get that

$$\tau_{T'}(3) = |T'| \frac{3}{2} \left( \frac{1}{2}n + 2 \right) \leq |T'| \frac{1}{2} \left( \frac{1}{2}n + 2 \right) + 3 \left( \frac{1}{2}n + 2 \right),$$

which is the same as  $(\frac{1}{2}n - 2) = |T'| \leq 3$ , but this fact contradicts the assumption that  $n > 11$  and the proof is finished.  $\square$

We get the following corollary as an immediate consequence:

**Corollary 2.2.1.** *If  $n$  is odd,  $n \geq 13$  and  $\Delta > \delta \geq \lceil \frac{1}{2}n \rceil + 1$ , then  $(G, X)$  has a compatible triangle.*

*Proof:* Take a graph  $G$  with the mentioned properties, choose any vertex  $v \in V(G)$  with minimum degree, and let  $V(G_{n-1}) = V(G) \setminus v$ . Then  $\delta(G_{n-1}) \geq \lceil \frac{1}{2}n \rceil = \frac{1}{2}(n-1) + 1$  and  $\Delta(G_{n-1}) \geq \lceil \frac{1}{2}n \rceil + 1 > \frac{1}{2}(n-1) + 1$ . Since  $(n-1)$  is even and  $(n-1) \geq 11$ , the previous lemma gives that  $(G, X)$  contains a compatible triangle.  $\square$

Finally we can prove the upper bound for  $f(n, \mathbb{C}_3, X_1)$  for graphs of order at least 11.

**Theorem 2.2.2.** *If  $n \geq 11$ , then  $f(n, \mathbb{C}_3, X_1) \leq \lceil \frac{1}{4}n^2 + \frac{1}{2}n \rceil + 2$ .*

*Proof:* By contradiction. Let  $X \in X_1$  and suppose that we have a graph  $G$  without compatible triangles on  $n \geq 11$  vertices where  $\varepsilon \geq \lceil \frac{1}{4}n^2 + \frac{1}{2}n \rceil + 3$ . We divide the problem in two cases, when  $n$  is odd and when  $n$  is even.

If  $n$  is odd and  $n \geq 13$  there exists a vertex  $v \in V(G)$  such that  $d(v) \leq \lceil \frac{1}{2}n \rceil + 1$  since if  $d(v) \geq \lceil \frac{1}{2}n \rceil + 1$  for all  $v \in V(G)$  we have a compatible triangle by the previous corollary. Let  $V(G_{n-1}) = V(G) \setminus v$ .

Clearly,  $(G_{n-1}, X)$  has no compatible triangle and

$$\varepsilon(G_{n-1}) \geq \left\lceil \frac{1}{4}n^2 + \frac{1}{2}n \right\rceil + 3 - \left\lfloor \frac{1}{2}n \right\rfloor - 1 = \left\lceil \frac{1}{4}(n-1)^2 + \frac{1}{2}(n-1) \right\rceil + 3.$$

If  $n$  is even and  $n \geq 12$  there exists a vertex  $v \in V(G)$  such that  $d(v) \leq \frac{1}{2}n$ , since if  $d(v) \geq \frac{1}{2}n + 1$  for all  $v \in V(G)$  there will be at least two vertices of degree at least

$\frac{1}{2}n + 2$  giving that  $\delta \geq \frac{1}{2}n + 1$  and  $\Delta > \frac{1}{2}n + 1$ , thus forcing a nonexistent compatible triangle to be in the graph according to our latest lemma. Let  $V(G_{n-1}) = V(G) \setminus v$ .

Again  $(G_{n-1}, X)$  has no compatible triangle and

$$\varepsilon(G_{n-1}) \geq \left\lceil \frac{1}{4}n^2 + \frac{1}{2}n \right\rceil + 3 - \frac{1}{2}n = \left\lceil \frac{1}{4}(n-1)^2 + \frac{1}{2}(n-1) \right\rceil + 3.$$

Repeating the argument we build a chain of graphs embedded in each other which can be viewed as

$$G = G_n \supset G_{n-1} \supset G_{n-2} \supset \cdots \supset G_{12} \supset G_{11}.$$

In this chain, no  $(G_k, X)$  has a compatible triangle, since  $(G, X)$  had none, and by construction all of them have the property that

$$\varepsilon(V(G_k)) \geq \left\lceil \frac{1}{4}k^2 + \frac{1}{2}k \right\rceil + 3.$$

But

$$\varepsilon(V(G_{11})) \geq \left\lceil \frac{1}{4}11^2 + \frac{1}{2}11 \right\rceil + 3 = 39 > \frac{1}{4}11^2 + \frac{3}{4}11,$$

which conflicts with the theorem for an upper bound with a general  $s$  in  $\mathcal{X}_s$  and we have a contradiction.  $\square$

## 2.3 Upper and Lower Bounds for Quadrilaterals

In this section we will change focus from triangles to cycles of length four and thus find upper and lower bounds for  $f(n, \mathbb{C}_4, \mathcal{X}_s)$ . The technique becomes somewhat different since the whole cycle is not known when we have fixed a path of length two. We start with a lemma, studying a small case with a complete bipartite graph  $K_{2,s+2}$  with a transition system  $X$  taken from  $\mathcal{X}_s$ .

**Lemma 2.3.1.** *If  $(K_{2,s+2}, X)$ , where  $X \in \mathcal{X}_s$  and  $s \geq 1$ , has no compatible cycle of length four, then there exists a forbidden transition of type  $v_1xv_2 \in F(K_{2,s+2})$ , where  $\{v_1, v_2\}$  is the small part of the bipartite graph.*

*Proof:* By contradiction. Take a complete bipartite graph  $K_{2,s+2}$  with a transition system  $X$  from  $\mathcal{X}_s$  and assume that it is enough to have forbidden transitions of type  $xv_iy$  to assure that there is no compatible cycle of length four. Note that any pair of vertices from the large part of the bipartite graph constitutes a cycle of length four together with the

vertices  $v_1$  and  $v_2$ . Also note that both  $v_i$  has at least two classes in its local colouring, since the larger part of the bipartite graph has  $s + 2$  vertices and each class in the local colouring contains at most  $s + 1$  edges. Let  $A$  be the vertices in the larger part of the bipartite graph such that it coincides with a nonempty colour class defined for the vertex  $v_1$ . Choose a vertex  $a \in A$  and study all cycles of form  $v_1av_2b$ , where  $b \notin A$ . Since we assumed all cycles of length four to be forbidden at  $v_1$  or  $v_2$ , we conclude that  $av_2b$  is a forbidden transition for every  $b$  as above. So  $v_2a$  has the same colour as every  $v_2b$ . By a symmetric argument we can conclude that  $v_1a$  has the same colour as any chosen  $v_1b$ . Now  $|A| = s + 2$ , and we have a contradiction.  $\square$

With this lemma proved we can now prove an upper bound for the number of edges in a graph with a transition system not containing a compatible cycle of length four. Note that the lemma is only for complete bipartite graphs of type  $K_{2,s+2}$ , but in the theorem below, we repeat the use of the lemma to handle complete bipartite graphs of type  $K_{2,m}$  when  $m > s + 2$ .

**Theorem 2.3.1.**  $f(n, \mathbb{C}_4, \mathcal{X}_s) \leq \frac{1}{2}n \left( \sqrt{(s+1)(n-1) + \frac{1}{4}(s+1)^2} + \frac{1}{2}(s+1) \right)$ .

*Proof.* Let  $X \in \mathcal{X}_s$  and let  $(G, X)$  be without compatible cycles of length four. Taken an unordered pair of vertices  $u$  and  $v$  from  $V(G)$  such that  $u \neq v$  and let  $|N(u) \cap N(v)| = m$ . The repeated use of the previous lemma gives that if  $m \geq s + 1$  there must be at least  $m - s - 1$  forbidden transitions  $uxv$ , where  $x \in \{N(u) \cap N(v)\}$ . Counting all forbidden transitions we get that

$$\sum_{u \neq v} (|N(u) \cap N(v)| - s - 1) \leq |F(G, X)| \leq s\varepsilon$$

according to the result of the maximum number of forbidden transitions when the transition system  $X$  is taken from the family  $\mathcal{X}_s$ . This in turn can be rewritten as

$$\sum_{u \neq v} |N(u) \cap N(v)| \leq s\varepsilon + (s+1) \binom{n}{2},$$

which is equivalent to

$$\sum_{i=1}^n \binom{d(v_i)}{2} \leq s\varepsilon + (s+1) \binom{n}{2},$$

since we are simply counting the number of paths of length two, and we get

$$\sum_{i=1}^n d(v_i)^2 - (s+1)d(v_i) \leq (s+1)n(n-1).$$

Consequently,

$$\left(\overline{d(v)}\right)^2 - (s+1)\overline{d(v)} \leq (s+1)(n-1).$$

Completing the square gives us

$$\overline{d(v)} \leq \sqrt{(s+1)(n-1) + \frac{1}{4}(s+1)^2} + \frac{1}{2}(s+1),$$

or as claimed

$$f(n, \mathbb{C}_4, \mathcal{X}_s) \leq \frac{1}{2}n \left( \sqrt{(s+1)(n-1) + \frac{1}{4}(s+1)^2} + \frac{1}{2}(s+1) \right).$$

□

Now, after this upper bound, we are about to make a construction of a graph together with a transition system to get a lower bound on  $f(n, \mathbb{C}_4, \mathcal{X}_s)$ . First we start with a preliminary lemma that will help us with the construction.

**Lemma 2.3.2.** *If  $G$  has no cycle of length four as a subgraph, then there exists a transition system  $X$  from  $\mathcal{X}_s$  such that  $(G(s+1), X)$  has no compatible cycle of length four.*

*Proof:* Take a simple graph  $G$  of order  $n$  without any cycles of length four and construct the graph  $G(s+1)$  according to the earlier definition. Remember that we get a graph on  $n(s+1)$  vertices where we have edges in  $G(s+1)$  depending on where the edges in  $G$  were situated as described in the equivalence below. Now

$$v_i v_j \in E(G) \Leftrightarrow v_i^k v_j^l \in E(G(s+1)),$$

which holds for all  $i, j \in \{1, 2, \dots, n\}$  and all  $k, l \in \{1, 2, \dots, (s+1)\}$ . Also note that  $v_i^k v_j^k \notin E(G(s+1))$  since  $G$  was a loop-less graph.

Now construct the local edge colouring  $X$  from  $\mathcal{X}_s$  according to the rule: For every fixed vertex  $v_i^k \in V(G(s+1))$ , and for every fixed  $j \in \{1, 2, \dots, n\}$ , let  $v_i^k v_j^l$ , for all  $j \in \{1, 2, \dots, s+1\}$  constitute a set in the partition  $X(v_i^k)$ . This gives that  $X \in \mathcal{X}_s$ .

Assume there is a compatible cycle of length four in  $(G(s+1), X)$ , say  $v_{i_1}^{k_1} v_{i_2}^{k_2} v_{i_3}^{k_3} v_{i_4}^{k_4}$ . If all  $i_j$ 's are different then, because of the construction of  $G(s+1)$ ,  $v_{i_1} v_{i_2} v_{i_3} v_{i_4}$  would constitute a cycle of length four in  $G$  which conflicts with the assumption on  $G$ .

Thus  $i_j = i_k$ , for some  $j$  and  $k$ , and since  $v_{i_j}^{k_j}$  and  $v_{i_k}^{k_k}$  cannot be neighbours in the cycle we have that  $\mathbb{C}_4 = v_{i_1}^{k_1} v_{i_2}^{k_2} v_{i_1}^{k_3} v_{i_4}^{k_4}$ . But this cycle of length four is not compatible to the given transition system since both  $v_{i_1}^{k_1} v_{i_2}^{k_2} v_{i_1}^{k_3}$  and  $v_{i_1}^{k_1} v_{i_4}^{k_4} v_{i_1}^{k_3}$  belong to the set of forbidden transitions  $F(G(s+1), X)$  and the proof is finished. □

This lemma is helpful when you are about to construct a graph with a transition system with as many edges as possible not containing a compatible cycle of length four. You start with a graph  $G$  with many edges but still without any cycles of length four, and then blow it up and impose a transition system as described. In the coming theorem 2.3.2, giving a lower bound on  $f(n, \mathbb{C}_4, \mathcal{X}_s)$ , we use an idea from [57] to construct extremal  $\mathbb{C}_4$ -free graphs from a finite projective plane and then using the above lemma. First, let us recall the definition of a finite projective plane:

**Definition 2.3.1.** A finite projective plane of order  $k$  is formally defined as a set of  $k^2 + k + 1$  points with the properties that:

- Any two points determine a line
- Any two lines determine a point
- Every point has  $k + 1$  lines on it
- Every line contains  $k + 1$  points

(Note that some of these properties are redundant. Also note that these properties can be fulfilled when  $k$  is a prime power)

Out of this definition, we can construct the bipartite point-line-graph  $P_{2m}$ , where  $m = k^2 + k + 1$ . Letting  $V(P_{2m}) = \{\cup_i p_i\} \cup \{\cup_i l_i\}$  be the vertex set for our graph, where each  $p_i$  is a point and each  $l_i$  is a line in our finite projective plane. The edges will be defined by

$$p_i l_j \in E(P_{2m}) \text{ if and only if } p_i \text{ is on line } l_j.$$

Now, if  $k$  is a prime power, we can make use of the notion of polarity and arrange the indices such that the following holds:

$$p_i l_j \in E(P_{2m}) \text{ if and only if } p_j l_i \in E(P_{2m}).$$

The details can be found in [57].

Now we are ready for the theorem:

**Theorem 2.3.2.** *If  $n = (s + 1)(k^2 + k + 1)$  and  $k$  is a prime power, then*

$$f(n, \mathbb{C}_4, \mathcal{X}_s) \geq \frac{1}{2}(n - 1) \sqrt{(s + 1)n - \frac{3}{4}(s + 1)^2} + \frac{1}{2}(s + 1)$$

*Proof:* By construction. Let  $m = k^2 + k + 1$  and take a point-line-graph  $P_{2m}$  of a projective plane of order  $k$  with a polarity, i.e. we can label the points  $p_1, \dots, p_m$  and the lines  $l_1, \dots, l_m$  such that if  $p_i l_j \in E(P_{2m})$  then  $p_j l_i \in E(P_{2m})$ . Such projective

planes exist for every prime power  $k$ , see [57] for further information. Note that there are exactly  $k + 1$  edges of type  $p_i l_i$  in  $E(P_{2m})$ .

Now construct  $P_{2m}(s + 1)$  and add all  $(s + 1)k^2$  missing edges of type  $p_i^k l_i^k$  to the edge set  $E(P_{2m}(s + 1))$ . Then contract all  $(s + 1)(k^2 + k + 1)$  edges of this type and replace all occurrences of multiple edges by simple ones.

The result is a graph  $G$  on  $n = m(s + 1)$  vertices which we label  $v_i^k$  where  $i \in \{1, 2, \dots, m\}$  and  $k \in \{1, 2, \dots, s + 1\}$ , and  $v_i^k$  is the result of contracting the edge  $p_i^k l_i^k$ . We also have that

$$\varepsilon(G) = \frac{1}{2} \left( (s + 1)^2 (k + 1)(k^2 + k + 1) - (s + 1)(k + 1) \right).$$

Define the local edge colouring as follows: For all vertices  $v_i^k \in V(G)$  and for every fixed  $j \in \{1, 2, \dots, m\}$ , colour  $v_i^k v_j^l$  for all  $l \in \{1, 2, \dots, s + 1\}$  in the same colour, unique for this vertex. This gives that our transition system  $X$  is of type  $\mathcal{X}_s$ .

Assume that  $G$  has a compatible cycle of length four, say  $v_{i_1}^{k_1} v_{i_2}^{k_2} v_{i_3}^{k_3} v_{i_4}^{k_4}$ . If all  $i_j$ 's are different, then  $v_{i_1} v_{i_2} v_{i_3} v_{i_4}$  forms a cycle in a contracted  $P_{2m}$ , but this is impossible since  $P_{2m}$  is a point-line-graph of a projective plane.

Thus  $i_j = i_l = i$  for some  $j$  and  $l$ . If  $v_i^{k_j}$  and  $v_i^{k_l}$  are not neighbours in the cycle, say  $v_i^{k_j} v_{i_2}^{k_2} v_i^{k_l} v_{i_4}^{k_4}$ , then  $v_i^{k_j} v_{i_2}^{k_2} v_i^{k_l} \in F(G, X)$  and is a forbidden transition.

Finally, if  $v_i^{k_j}$  and  $v_i^{k_l}$  were neighbours in the cycle, say  $v_i^{k_j} v_i^{k_l} v_{i_3}^{k_3} v_{i_4}^{k_4}$ , then  $p_i^{k_j} l_i^{k_l} p_{i_3}^{k_3} l_{i_4}^{k_4}$  forms a cycle in  $P_{2m}(s + 1)$  and  $p_i l_i p_{i_3} l_{i_4}$  is a cycle in  $P_{2m}$ , which is impossible by assumption.

Thus  $(G, X)$  has no compatible cycle of length four and

$$\varepsilon(G) = \frac{1}{2} \left( (s + 1)^2 (k + 1)(k^2 + k + 1) - (s + 1)(k + 1) \right),$$

which can be rewritten as

$$\frac{1}{2}(n - 1) \sqrt{(s + 1)n - \frac{3}{4}(s + 1)^2} + \frac{1}{2}(s + 1),$$

and the theorem follows.  $\square$

### 2.3.1 Remarks

It can be worth noting that for all fixed  $s$ ,

$$\lim_{n \rightarrow \infty} \frac{f(n, \mathbb{C}_3, \mathcal{X}_s)}{f(n, \mathbb{C}_3, X_0)} = 1,$$

while

$$\lim_{n \rightarrow \infty, n \in Q} \frac{f(n, \mathbb{C}_4, \mathcal{X}_s)}{f(n, \mathbb{C}_4, X_0)} = \sqrt{s + 1},$$

### 2.3. UPPER AND LOWER BOUNDS FOR QUADRILATERALS

---

where  $Q$  is the set of all positive integers of the form  $n = (s + 1)(k^2 + k + 1)$  where  $s$  is an integer and  $k$  is a prime power.

Also we have the not so bold conjecture about transition systems not having compatible odd cycles of certain lengths:

**Conjecture 2.3.1.** *For every odd  $k$ ,  $k \geq 5$ , and for every  $s \geq 0$  there exists an integer  $N_{k,s}$ , depending on the chosen integers, such that  $f(n, \mathbb{C}_k, \mathcal{X}_s) = \lfloor \frac{1}{4}n^2 \rfloor$  if  $n \geq N_{k,s}$ .*

A more informal way of describing this conjecture is to say that: When we try to forbid all cycles of a specified odd length of at least 5, and  $s$  in our transition system  $\mathcal{X}_s$  is held fixed, we will end up with the largest complete bipartite graph (not containing any odd cycles at all) when we let the number of vertices be large enough.



## Chapter 3

# Erdős-Sós Conjecture for Graphs with High Minimum Degree

This chapter also deals with extremal graph theory and the question dealt with is how many edges there can be in a graph (depending on the number of vertices) before we can be sure of having any tree on  $k$  edges present in the graph as a subgraph.

As in many other cases of extremal graph theory, Paul Erdős has made a lot of work and of course presented loads of conjectures in the area. For this specific problem, Erdős and Sós conjectured in 1963 that if  $G$  is a (simple) graph of order  $n$  with more than  $\frac{n}{2}(k-1)$  edges, then every tree on  $k$  edges is contained in  $G$  as a subgraph.

In this chapter we will present some partial results concerning this conjecture. In particular we prove that the conjecture is true for graphs with minimum degree  $\delta(G) \geq k-4$ . A consequence of this result is that the Erdős-Sós conjecture is true for every  $k \leq 9$ .

### 3.1 Introduction

We shall use standard graph theory notation as presented in the first chapter. For other theoretic notions not explicitly defined in the chapter, see for example [24], [21] and [37].

In 1959 Erdős and Gallai [38] proved that every graph  $G$ , with

$$\varepsilon(G) > \frac{n}{2}(k-1),$$

contains a path of length  $k$ . Later on, in view of this result, Erdős and Sós [37] conjectured the following in 1963:

**Conjecture 3.1.1.** (*Erdős-Sós*)

*If  $G$  is a graph and  $\varepsilon(G) > \frac{n}{2}(k-1)$ , then every tree  $T$  on  $k$  edges is contained in  $G$  as a subgraph.*

As many other conjectures, it can be restated in different ways. A common way to restate this conjecture is to see it as a packing problem as in both [28] and [37]:

**Conjecture 3.1.2.** (*restated*)

*Suppose that  $G$  is a graph of order  $n$  and  $T$  is any tree of size  $k$ . If  $\varepsilon(G) < \frac{n}{2}(n-k)$ , then  $G$  and  $T$  are packable into the complete graph  $K_n$ .*

Here, *packable* is the graph theoretic notion defined by the following:

**Definition 3.1.1.** Two simple graphs  $G$  and  $H$  are packable into a complete graph  $K_n$  if and only if there exist two edge-disjoint subgraphs of  $K_n$ ,  $G'$  and  $H'$  such that  $G$  is isomorphic to  $G'$  and  $H$  to  $H'$ .

Other, less evident, reformulations have been made, e.g. in [63] where Sidorenko proved that the following statement about the density of the graph is equivalent to the Erdős-Sós conjecture.

**Conjecture 3.1.3.** (*restated*)

*Every graph  $G \in Z(T)$  has the property  $p^*(G) \leq k - 2$ . (See [63] for explicit definitions.)*

We shall use the first stated conjecture throughout this chapter and therefore make use of the following definition:

**Definition 3.1.2.** Every graph  $G$  with size  $\varepsilon(G) > \frac{n}{2}(k - 1)$  is called an Erdős-Sós graph for trees on  $k$  edges. (Or only an Erdős-Sós graph if it is clear from the context what  $k$  is.)

## 3.2 Earlier Results on the Erdős-Sós Conjecture

The Erdős-Sós conjecture has been proven for many particular families of trees. As mentioned in the introduction, Erdős and Gallai proved in [38] that a path on  $k$  edges is contained in every Erdős-Sós graph. The following trivial result shows that a star on  $k$  edges also is contained in every Erdős-Sós graph:

**Proposition 3.2.1.** *If  $G$  is an Erdős-Sós graph, then is  $\Delta(G) \geq k$ .*

*Proof:* In an Erdős-Sós graph is the mean degree strictly greater than  $k - 1$ . Consequently, there must be a vertex of degree at least  $k$ .  $\square$

There are also proofs for double-stars (a consequence of Sidorenko's result in [63]), caterpillars with one leg (proven in [25]), comets (proven in [70]), double-stars with length 2 and 3 respectively between the centres (also proven in [70]), spiders of diameter at most 4 (proven in [69]) and trees with a vertex incident to at least  $\frac{1}{2}(k - 1)$  leaves (proven in [63]). Also, there is a result for general caterpillars (mentioned in [53] to be proved by Perles in 1990), but since the proof of this fact never has been published, the result will not be used throughout this chapter. Proper definitions of comets, double-stars and spiders are presented below in subsection 3.5.2, and also in the beginning of the thesis.

By the above list we can make the following claim:

**Theorem 3.2.1.** *The Erdős-Sós conjecture is true for every  $k \leq 5$ .*

*Remark:* Woźniak does count the result of Perles in [53] and gets  $k \leq 6$  in the claim above. Without the result about general caterpillars we are left to prove the conjecture for the tree  $T$  where the vertex set  $V(T) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$  and the set of edges  $E(T) = \{v_1v_2, v_2v_3, v_3v_4, v_4v_5, v_2v_6, v_3v_7\}$  to close the case when  $k = 6$ .

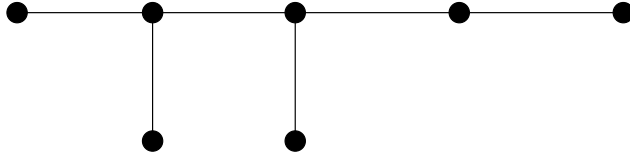


Figure 3.1: The last remaining tree to close the case  $k = 6$ .

By summing up information from [69],[60],[73],[64] and [71] we can also conclude the following:

**Theorem 3.2.2.** *The Erdős-Sós conjecture is true for every  $k \geq n - 3$ .*

It is easy to see that these two claims together implies the following corollary:

**Corollary 3.2.1.** *The Erdős-Sós conjecture is true for every  $n \leq 9$ .*

The Erdős-Sós conjecture has also been shown to be true for special families of graphs. Brandt and Dobson proved in [29] that:

**Theorem 3.2.3.** *The Erdős-Sós conjecture is true for graphs with girth at least 5.*

*Remark:* An earlier, considerably longer proof by Dobson can be found in [36].

Saclé and Woźniak showed that the condition on the graph  $G$  could be slightly weaker and presented the following theorem in [59]:

**Theorem 3.2.4.** *The Erdős-Sós conjecture is true for all graphs that do not contain a subgraph isomorphic to the cycle  $C_4$ .*

In the last two theorems, set aside the new conditions concerning girth and subgraphs respectively, it is sufficient for the graph  $G$  to degree majorise  $(\frac{k}{2}, \dots, \frac{k}{2}, k)$  to contain all trees of size  $k$ . We will consider other degree majorizing graphs in connection to the Erdős-Sós conjecture in next section.

To be mentioned is also Gyárfás, Szemerédi and Tuza's result in [46]:

**Theorem 3.2.5.** *A  $k$ -chromatic graph without triangles and quadrangles contains every tree on  $k$  vertices as an induced subgraph.*

This result is to be compared with the result of Brandt and Dobson above that allowed us to find the trees as subgraphs of  $G$ , but not as induced subgraphs.

There are also results on the Erdős-Sós conjecture in an approximate form as next theorem by Ajtai, Komlós and Szemerédi. For details see [51] and [1].

**Theorem 3.2.6.** *For every  $\varepsilon > 0$  there is a threshold  $k_0$  such that the following statement holds for all  $k \geq k_0$ : Every graph with average degree more than  $(1 + \varepsilon)(k - 1)$  contains, as subgraphs, all trees with  $k$  edges.*

It is important to note that the author's 1991 manuscript contains only the dense case, that is, when  $n \leq Ck$ . The sparse case needs a modified form of the Regularity Lemma that is not as compact and as generally applicable as the original Regularity Lemma.

Also to be mentioned is the following approximate result in [48], concerning a conjecture by Bollobás, with a proof by Komlós, Sárközy and Szemerédi.

**Theorem 3.2.7.** *For every  $\varepsilon > 0$  and  $\Delta$  there is a threshold  $n_0$  such that the following statement holds for all  $n \geq n_0$ : If  $T$  is a tree of order  $n$  and maximum degree  $\Delta$ , and  $G_n$  has minimum degree at least  $(1 + \varepsilon)\frac{n}{2}$ , then  $T$  is a subgraph of  $G_n$ .*

### 3.3 Extension of Sidorenkos Result

In this section we are about to make a generalisation of Sidorenkos theorem from [63], here presented below:

**Theorem 3.3.1.** *(Sidorenko)*

*Let  $G$  be a simple graph on  $n$  vertices. If  $G$  degree majorise  $(\lceil \frac{k}{2} \rceil, \dots, \lceil \frac{k}{2} \rceil, k)$ , then every tree of size  $k$  with at least  $\lfloor \frac{k}{2} \rfloor$  leaves situated at the same vertex is contained as a subgraph in  $G$ , when  $G$  is a reduced Erdős-Sós graph.*

Now, when approaching the Erdős-Sós conjecture it is often convenient to reduce the problem to a certain class of Erdős-Sós graphs. The following proposition yields that we only have to consider connected Erdős-Sós graphs with minimum degree  $\lceil \frac{k}{2} \rceil$ .

**Proposition 3.3.1.** *A smallest counterexample  $G$ , i.e. minimal  $|V(G)|$ , to the Erdős-Sós conjecture is connected and has minimum degree  $\delta(G) \geq \lceil \frac{k}{2} \rceil$ .*

*Proof:* First assume that the graph  $G$  has more than one component. By the size of  $G$  we know that at least one component has degree strictly greater than  $k - 1$  in average. This yields that a smallest counterexample is connected.

Now assume that we have a smallest counterexample with minimum degree  $\delta(G) < \lceil \frac{k}{2} \rceil$ . That is, there exists a tree  $T_k$  on  $k$  edges not present as a subgraph in the graph  $G$  and  $\varepsilon(G) > \frac{n}{2}(k - 1)$ . Removing a vertex of minimum degree together with its incident edges yields a new graph  $G'$  on  $n - 1$  vertices having more than  $\frac{n-1}{2}(k - 1)$  edges that

does not contain the earlier specified  $T_k$ , contradicting the fact that  $G$  was a minimal counterexample. By this, the proposition must be true, since the conjecture is true for  $n = k + 1$ .  $\square$

This proposition makes it natural to consider the following class of Erdős-Sós graphs:

**Definition 3.3.1.** A graph  $G$  is called a reduced Erdős-Sós graph for trees on  $k$  edges if it satisfies

$$\varepsilon(G) > \frac{|V(G)|}{2}(k - 1)$$

and no induced subgraph of  $G$  satisfy the same condition. It follows from the proof in the above proposition that such a graph is connected and has minimum degree  $\delta(G) \geq \lceil \frac{k}{2} \rceil$ . (As before,  $k$  is omitted if it is clear from the context what it is.)

Now we are prepared to generalise the result of Sidorenko in [63]. First a helpful lemma, where say that  $G$  degree majorise  $(k - m, \dots, k - m)$ , instead of simply saying that  $\delta(G) \geq r = k - m$ , to give the reader a hint on how the lemma is used further on in the chapter:

**Lemma 3.3.1.** *Let  $G$  be a simple graph on  $n$  vertices and let  $n > k - m$ . If  $G$  degree majorises  $(k - m, \dots, k - m)$ , then every forest  $F$  on  $k - m$  vertices is contained in  $G$  as a subgraph.*

*Moreover, any set of vertices in the forest  $F$  obtained by picking one vertex from every component can be identified with any other set of vertices of equal size from  $V(G)$ . Also, any other vertex in  $V(G)$ , outside this chosen set, can be avoided by the forest  $F$ .*

*Proof:* By induction on the number of edges in the forest  $F$ . The lemma is trivially true if the forest  $F$  does not contain any edges. Now choose a forest  $F$  of order  $k - m$  with  $c$  components. Then choose one vertex in each component of  $F$  and identify this set of vertices with any set of vertices of equal size from the graph  $G$ . Also choose one vertex,  $v$  say, in  $V(G)$  to be avoided by the forest  $F$ . Assume that we from this starting position have built the forest  $F$  in  $G$  except for one edge and one of its incident vertices. Left is to add one edge and a vertex neither present in  $F$  nor equal to  $v$ , to a specified vertex  $u \in V(F)$ . Since  $d(u) \geq k - m$  there will be at least one edge from  $u$  neither incident with  $v$  nor with any vertex in  $V(F)$ , due to the minimal degree. Adding this edge together with its incident vertex completes the proof.  $\square$

By the preceding lemma we can prove the following:

**Theorem 3.3.2.** *Let  $G$  be a simple graph on  $n$  vertices. If  $G$  degree majorise  $(k - m, \dots, k - m, k)$ , then every tree of size  $k$  with at least  $m$  leaves situated at a single vertex, is contained as a subgraph in  $G$ .*

CHAPTER 3. ERDŐS-SÓS CONJECTURE FOR GRAPHS WITH HIGH  
MINIMUM DEGREE

---

*Proof:* We only have to consider the case when  $k > m$ . Now, pick a graph  $G$  and a tree  $T$  as described above and delete a vertex in  $T$  with at least  $m$  leaves and its incident edges. Also delete all isolated vertices created by this process. Left is a forest  $F$  that fulfils the lemma above. Choose a vertex  $v$  with degree  $d(v) \geq k$  to be the avoided vertex in the lemma. For every component  $c_i$  in the forest  $F$ , choose the vertex  $v_i$  that was neighbour to the deleted vertex to be a chosen representative of the component. In  $G$ , choose only neighbours to  $v$  to be identified with each  $v_i$ . According to our lemma, we can find this forest with these prescribed vertices and avoid  $v$  at the same time. Now adding  $v$ , the edges to each  $v_i$  and all leaves, which is possible due to the degree of  $v$ , yields our tree  $T$ .  $\square$

This is a generalisation of Sidorenkos theorem in [63], since putting  $m = \lfloor \frac{k}{2} \rfloor$  and letting  $G$  be a reduced Erdős-Sós graph gives exactly his theorem.

But the proof of the theorem above does not take in account that our object of interest is reduced Erdős-Sós graphs. In such a graph  $G$  we know that the number of edges is strictly greater than a minimal graph  $G'$  that degree majorises  $(\lceil \frac{k}{2} \rceil, \dots, \lceil \frac{k}{2} \rceil, k)$  if  $k > 2$ . The question to answer is where these edges can be situated. This motivates the following propositions:

**Proposition 3.3.2.** *Let  $G$  be a reduced Erdős-Sós graph. Then  $d(u) + d(v) \geq k + 1$  if  $uv \in E(G)$ .*

*Proof:* Assume that  $d(u) + d(v) \leq k$  and that  $uv \in E(G)$ . Then  $H$ , the induced subgraph on  $V(G) \setminus \{u, v\}$ , is also an Erdős-Sós graph and consequently  $G$  was not properly reduced.  $\square$

**Proposition 3.3.3.** *Let  $G$  be a reduced Erdős-Sós graph and let the vertices  $u, v$  and  $w$  constitute a triangle in  $G$ . Then  $d(u) + d(v) + d(w) > 3\lceil \frac{k}{2} \rceil$ .*

*Proof:* Let  $G$  be a reduced Erdős-Sós graph. Assume that the vertices  $u, v$  and  $w$  constitutes a triangle in  $G$  and also assume that  $d(u) + d(v) + d(w) \leq 3\lceil \frac{k}{2} \rceil$ . Then is  $H$ , the induced graph on  $V(G) \setminus \{u, v, w\}$  also an Erdős-Sós graph and consequently  $G$  was not properly reduced.  $\square$

With this in mind we can improve our last theorem to enlarge the class of trees for which the Erdős-Sós conjecture is true. First a lemma concerning degree majorizing graphs.

**Lemma 3.3.2.** *Let  $G$  be a graph that degree majorise  $(\lceil \frac{k}{2} \rceil, \dots, \lceil \frac{k}{2} \rceil, k)$  and let  $T$  be a tree with at least  $\lfloor \frac{k}{2} \rfloor - 1$  leaves at a vertex  $v$ . If  $u$  is a leaf at distance  $r$  from  $v$  in  $T$  one you can find two vertices  $x$  and  $y$  in the graph  $G$  such that they constitute the two ends of a path of length  $r - 1$  such that  $d(x) \geq k$  and  $d(y) \geq \lceil \frac{k}{2} \rceil + 1$ , then  $T$  is contained in  $G$  as a subgraph.*

*Proof:* Take a graph  $G$  and a tree  $T$  as described above. Let  $w$  be the neighbour to the leaf  $u$  in the tree  $T$ . The vertices  $v$  and  $w$  are at distance  $r - 1$  in the tree  $T$ .

Now identify the vertex  $v$  with  $x \in V(G)$  and  $w$  with  $y \in V(G)$  and use the prescribed path in  $G$  as the path between  $v$  and  $w$  in the tree  $T$ .

Our aim is to find a copy of  $T$  in  $G$ . So, start with the prescribed path and add all edges (to achieve  $T$ ) not constituting leaves at  $v$  nor the last leaf at  $w$ . This is possible due to the minimum degree in  $G$ . Then add the last leaf at the vertex  $w$  and finally add the remaining leaves at the vertex  $v$  which is possible due to the prescribed degrees at these two vertices.  $\square$

By this we can prove the following:

**Theorem 3.3.3.** *Every tree  $T$  on  $k$  edges that has a vertex incident to at least  $\lfloor \frac{k}{2} \rfloor - 1$  leaves is contained in any Erdős-Sós graph.*

*Proof:* It is enough to prove the theorem for reduced Erdős-Sós graphs.

Later on in the proof we will denote the set of vertices being leaves adjacent to a vertex  $v$  in a tree  $T$  by  $L(v)$ .

*Case 1:* If  $k$  is even.

Take a tree  $T$  as above and a reduced Erdős-Sós graph. Putting  $m = \lfloor \frac{k}{2} \rfloor$  in our theorem 3.3.2 above, gives that we can find any tree  $T$ , if it contains a vertex  $v$  incident to at least  $\lfloor \frac{k}{2} \rfloor$  leaves. Thus we can find a subtree  $T'$  to our tree  $T$  in the graph  $G$ , missing only the last edge that should be added at a specified vertex  $u$  in the corresponding forest  $F$ , the forest induced on the vertex set  $V(F) = V(T) \setminus \{v, L(v)\}$ , as obtained in the first lemma of this section. If  $d(u) > \frac{k}{2}$  we can add the edge to our forest, due to the degree of  $u$ , and then join the remaining star with centre at  $v$  as in the preceding lemma 3.3.2.

Now assume that  $d(u) = \frac{k}{2}$ . The neighbours to  $u$ , the set  $N(u)$ , must be contained in  $V(F) \cup v$ , where  $v$  is the specified vertex in  $V(T)$  with at least  $\lfloor \frac{k}{2} \rfloor - 1$  leaves, since otherwise there is nothing left to prove. But in this case we know due to an earlier proposition 3.3.2 that all other vertices in the forest  $F$  must have degree at least  $\frac{k}{2} + 1$ , otherwise the graph  $G$  was not properly reduced. If one of these vertices in  $V(F) \cup v$ , say  $w$ , is a leaf with its edge situated at  $y$ , different from both  $u$  and  $v$ , let  $T''$  be  $T' \cup \{uw\} \setminus wy$ . Since  $d(y) > \frac{k}{2}$  we are finished due to our preceding lemma 3.3.2 and the degree of  $y$ .

If no such vertex  $w$  is present in the forest, we conclude that the tree we are looking for is two stars joined by a path. If the path is of length at most three, we are finished due to earlier results found in [63] and [70]. If not, we have a path  $v x_1 x_2 \dots x_r$  where the centre of the smallest star,  $x_r$ , has all vertices in the path as neighbours. With help of proposition 3.3.2 we know that  $v x_r x_1 x_2 \dots x_{r-1}$  is a path in  $G$  where  $d(v) \geq k$  and  $d(x_{r-1}) > \frac{k}{2}$ . Thus our preceding lemma 3.3.2 closes this case.

*Case 2: If  $k$  is odd.*

Take a tree as above and a reduced Erdős-Sós graph. The proof of this case follows almost the case when  $k$  is even. If there are two vertices  $x_1$  and  $x_2$  with properties as  $x$  in Case 1, we can use proposition 3.3.3 and the technique from the first case. Otherwise we know that the tree has, set aside the vertex identified with  $v$ , at most two vertices having leaves as neighbours. First, if there are two of these vertices, both of them has degree  $\lceil \frac{k}{2} \rceil$  and have all their neighbours in  $V(F) \cup v$  yielding that all other vertices in the forest  $F$  has larger degree by proposition 3.3.3. Unless not both  $x_1$  and  $x_2$  are neighbours to  $v$  in the subtree we can find a path of appropriate length from  $v$  to a vertex  $u$  such that we can use the proof technique in lemma 3.3.2 and we are finished. If  $x_1vx_2$  is a path in the subtree we can start choosing  $x_1$  and  $x_2$  such that either is  $d(x_1) > \frac{k}{2}$  or  $x_1x_2$  not an edge in  $G$  as a consequence of the earlier proposition 3.3.3. By the previous lemma 3.3.2 we have finished this sub-case as well.

At last we have to consider the case when  $T$  consists of two stars joined by a path. If the path is of length at most three we are finished due to earlier results in [70]. Otherwise we have a path  $vx_1x_2 \dots x_r$  where the centre of the smallest star,  $x_r$ , has all vertices in the path as neighbours. With help of proposition 3.3.3 we know that  $vx_r x_1 x_2 \dots x_{r-1}$  or  $vx_1 x_r x_{r-1} \dots x_2$  is a path in  $G$  where  $d(v) \geq k$  and  $d(x_{r-1}) > \frac{k}{2}$ . Thus our preceding lemma 3.3.2 closes this case.  $\square$

This enlarges the set of trees for which the Erdős-Sós conjecture is true and we can also conclude the following:

**Corollary 3.3.1.** *The Erdős-Sós conjecture is true for every  $k \leq 6$ .*

*Proof:* The only tree left to prove the conjecture for, as remarked in the above section about earlier results and pictured above in figure 3.1, has a vertex incident to at least  $\lfloor \frac{6}{2} \rfloor - 1 = 2$  leaves.  $\square$

### 3.4 Erdős-Sós Graphs with High Minimal Degree

The theorem 3.3.2 in the previous section allows us to tell that any Erdős-Sós graph with minimum degree at least  $k - 1$  contains every tree on  $k$  edges as a subgraph.

**Corollary 3.4.1.** *An Erdős-Sós graph  $G$  with minimum degree  $\delta(G) \geq k - 1$  contains every tree on  $k$  edges as a subgraph.*

*Proof:* Since every tree has a leaf, putting  $m = 1$  in theorem 3.3.2 we get our result.  $\square$

But this result can be strengthened by a small observation and by lowering the minimum valency to  $k - 2$ .

**Proposition 3.4.1.** *An Erdős-Sós graph  $G$  with minimum degree  $\delta(G) \geq k - 2$  contains every tree on  $k$  edges as a subgraph.*

*Proof:* Using theorem 3.3.2 from the previous section, putting  $m = 2$ , we get that the proposition is true for all trees with at least two leaves situated at the same vertex. Thus we can assume that  $T$  is a tree where all vertices have at most one leaf as a neighbour. In particular, this implies that the tree  $T$  has a path of length two,  $x_1x_2x_3$  say, as a subgraph where one end of the path is identified with a leaf in  $T$  and  $d(x_2) = 2$ .

Now take a vertex  $v$  in the graph  $G$  with degree at least  $k$  and a largest complete subgraph  $K_r$  in  $G$  containing  $v$ . We will try to build the tree  $T$  as a subgraph to  $G$  such that  $v$  is a neighbour to a leaf in  $T$ . As in earlier proofs we will wait to add that specific leaf until the last step. Start at  $v$  and build the tree by first using the vertices and edges in the complete graph  $K_r$  containing  $v$ . By the minimum degree of  $G$  we can add the first  $k - 2$  edges of the tree  $T$  creating a subgraph  $T'$  of  $T$ . We can assume that  $x_2$  is the last added vertex. Leaving the leaf at  $v$  for a moment we try to add the second last edge at a specified vertex  $u$ . If that is not possible, the vertex  $u$  has exactly the vertex set  $V(T') \setminus u$  as neighbours. This yields that either the chosen complete graph was not the largest one containing  $v$ , in this case we repeat the argument, or else it was a complete graph of order at least  $k - 1$ . It is enough to consider the case when the chosen complete graph has order exactly  $k - 1$ .

If  $v$  belongs to a path of length two, edge-disjoint from the complete graph  $K_r$ , we identify the earlier mentioned path with this one. Using the complete graph attached to the path yields the specified tree  $T$  as a subgraph of  $G$ .

If there is no such path, then  $G$  is either not an Erdős-Sós graph with the supposed minimum degree or else there is a path  $vxy$  of length two from  $v$  with its middle vertex  $x$  outside the complete graph. But this implies that we can build our tree  $T$  with  $yx$  as a leaf attached to the complete graph  $K_r$  and finally add the last leaf situated at  $v$  because of the chosen degree of  $v$ .  $\square$

With a little more effort we can also prove the following result:

**Theorem 3.4.1.** *An Erdős-Sós graph  $G$  with minimum degree  $\delta(G) \geq k - 3$  contains every tree  $T$  on  $k$  edges as a subgraph.*

*Proof:* Putting  $m = 3$  in theorem 3.3.2 gives that the claim is true if the tree  $T$  has a vertex adjacent to at least three leaves. Also we can assume that  $k \geq 7$  by corollary 3.3.1. We consider two cases:

*Case 1:* The tree  $T$  has a vertex adjacent to two leaves.

Take a vertex  $v$  in the graph  $G$  with degree at least  $k$  and the largest complete graph  $K_r$  in  $G$  containing  $v$ . We will try to build the tree  $T$  in  $G$  such that  $v$  is a vertex adjacent to two leaves. As before, the leaves adjacent to  $v$  will be added in the last step in

this process and we always build the tree by using edges from the chosen complete graph containing  $v$  as long as possible.

Start at a vertex  $v$  and build the tree by first using the edges and vertices in the chosen complete graph containing  $v$ . By the minimum degree of  $G$  we can add the first  $k - 3$  edges of the tree  $T$ , creating a subtree  $T'$  as a subgraph of  $G$ . Leaving the leaves at  $v$  for a moment we try to add the third last edge at a specified vertex  $u$  in  $T'$ .

First consider the sub-case when  $u \notin K_r$ . If it is not possible to add an edge to  $u$ , the vertex  $u$  has exactly the vertex set  $V(T') \setminus u$  as neighbours. This yields that either the chosen complete graph  $K_r$  was not the largest one containing  $v$ , in which case we repeat the argument, or else it is a complete graph of order at least  $k - 2$ .

In the second sub-case, assume first that the tree  $T$  has a path of length two,  $x_1x_2x_3$  say, ending in a leaf  $x_3$  where  $d(x_2) = 2$ . Then let  $u = x_2$  be the last added vertex in the preceding process and we are back to our previous case. So we can assume that  $T$  has at least two vertices,  $x_1$  and  $x_2$  say, with exactly two leaves as neighbours and that at least one of the,  $x_2$  say, is the centre of a star,  $S_3$ , on three edges such that the remaining graph, when all edges in  $S_3$  are removed as well as all created isolated vertices, also is a tree. Assume that  $x_1$  will be identified with  $v$  when we build our tree  $T$  in  $G$  and let  $x_2 = u$  be the second last vertex added in our building process. If  $r < k - 3$  there will be an edge to add at  $u$ , else  $r$  was not chosen to be maximal. If  $r = k - 3$  all vertices in  $K_r \setminus v$  will have at least one neighbour outside the complete graph  $K_r$ . Let  $y$  be such a vertex and identify  $u$  with  $y$ . If  $|N(y) \cap \{V(K_r) \setminus v\}| = 1$  we will find our subtree  $T'$  on  $k - 2$  edges and then we can add the last two leaves at  $x_1 = v$  because of the degree of that vertex. If  $|N(y) \cap \{V(K_r) \setminus v\}| > 1$  we can choose a vertex from this set to be one of the leaves from  $x_2$  and build a subtree  $T'$  on  $k - 3$  edges using all vertices in the complete graph  $K_r$  and having  $u$  outside the complete graph. Left is only to add the last leaf at  $u$ , which is possible due to the choice of a maximal  $r$ , and then to add the leaves at  $x_1 = v$  which is possible due to the degree of  $v$ .

So, it is enough to consider the case when the complete graph has order exactly  $k - 2$ . Note that no vertex in the complete graph but  $v$ , has a neighbour outside the complete graph since otherwise we could enlarge our subtree and finally add the two remaining leaves at the vertex  $v$ .

Now, if we can find a star  $S_3$  on three edges, edge disjoint from the complete graph  $K_r$ , with a leaf at  $v$  we can find our tree  $T$  with the vertex adjacent to two leaves moved to the centre of the star  $S_3$ . Assume that this is not possible. The vertex  $v$  has a neighbour, say  $x$ , outside the complete graph  $K_r$  due to the degree of  $v$ . If  $x$  is not centre of a star as described above, all but at most one neighbour to  $x$  are in the complete graph. But since  $k \geq 7$  we have a contradiction to the fact that  $x$  was not a neighbour to a vertex in  $V(K_r) \setminus v$ .

*Case 2:* The tree  $T$  has no vertex adjacent to more than one leaf.

Take a vertex  $v$  in the graph  $G$  with degree at least  $k$  and the largest complete graph  $K_r$  containing  $v$ . We will try to build the tree  $T$  in  $G$  such that the vertex  $v$  is adjacent

to a leaf. As before, the leaf adjacent to  $v$  will be added as a last step in this process and we will always build the tree by using vertices and edges from the complete graph containing  $v$  as long as possible. By the minimum degree we can add the first  $k - 3$  edges of  $T$  creating a subtree  $T'$  as a subgraph in  $G$ . The next edge of  $T$  can be added unless  $r = k - 2$  and all the vertices in the complete graph  $K_r$  except  $v$  have all their neighbours inside  $K_r$ . In that case we can find an appropriate tree  $T''$  on three vertices rooted in  $v$  outside the complete graph  $K_r$  such that  $T \subset (K_r \cup T'')$ , otherwise there are vertices in  $V(K_r) \setminus v$  having neighbours outside the complete graph.

So, we can assume that we have a tree  $T'$  on  $k - 2$  edges and try to add the second last edge at the specified vertex  $u$ . If this is not possible,  $u$  has all its neighbours in  $V(T') \setminus u$ . Since the vertex  $u$  does not have all vertices in the maximal complete graph as neighbours we know as well that  $u$  has every vertex in  $V(T') \setminus (u \cup V(K_r))$  as a neighbour. At this stage we can start to build a largest complete graph  $K_s$  outside  $K_r$  where all vertices have degree  $r - 1$  into the previous complete graph  $K_r$ . Now, when trying to build up our tree  $T$ , we start with using all vertices in  $K_r$  and continue by using vertices and edges in our  $K_s$ . Assume that we have found the maximal complete graph  $K_s$  and that we try to add next edge to a specified vertex  $u$  in our tree  $T'$ . If this is not possible,  $u$  has  $r - 1$  neighbours to  $K_r$  and  $s - 1$  neighbours to the complete graph  $K_s$  implying that  $r + s = k - 1$ . Note that the complete graph  $K_s$  only has neighbours inside itself and in the other complete graph  $K_r$ , since otherwise we would be able to add one edge to get a larger subtree on  $k - 1$  edges.

Now, since every vertex in the tree  $T$  is adjacent to at most one leaf we know especially that  $T$  has a path of length two as a subgraph where one end of the path is identified with a leaf and the other end is connected to the remaining of the tree. Due to the minimal degree in the graph induced by  $V(K_r) \cup V(K_s)$  we can find any tree on  $k - 2$  edges with any vertex rooted at  $v$  using only vertices from the induced graph. If we can find a path of length two starting at  $v$ , not using any other vertices from  $V(K_r) \cup V(K_s)$ , we are finished. Otherwise, since the vertex  $v$  is of degree at least  $k$ ,  $v$  has at least two neighbours,  $x_1$  and  $x_2$  say, having all their neighbours in  $V(K_r) \cup V(K_s)$ . We know also that both  $x_1$  and  $x_2$  has no neighbours in  $K_s$  and at most  $r - 1$  neighbours in  $K_r$ . Due to the minimal degree we get that  $r \geq k - 2$ . We end up with a complete graph on at least  $k - 2$  vertices and at least another three vertices, namely  $x_1$ ,  $x_2$  and  $V(K_s)$  with at least  $k - 3$  neighbours inside the complete graph  $K_{k-2}$ . If the tree  $T$  has at least three leaves, we identify these with the vertices outside the complete graph  $K_r$  and build the tree inside the complete graph using the fact that  $k \geq 7$ . If  $T$  only has two leaves it is a path and proven to be in  $G$  for every Erdős-Sós graph.  $\square$

Naturally we have the following corollary:

**Corollary 3.4.2.** *The Erdős-Sós conjecture is true for every  $k \leq 7$ .*

*Proof:* Consider a reduced Erdős-Sós graph  $G$ . Since  $k - 3 \leq \lceil \frac{k-1}{2} \rceil \leq \delta(G)$  if  $k \leq 7$ , the

corollary is true by the previous theorem.  $\square$

Even though the argument becomes more and more technical we can lower the minimal degree once again.

**Theorem 3.4.2.** *An Erdős-Sós graph  $G$  with minimum degree  $\delta(G) \geq k - 4$  contains every tree  $T$  on  $k$  edges as a subgraph.*

*Proof:* We examine case by case different classes of trees depending on the vertex in the tree  $T$  with maximal number of leaves adjacent to it.

*Case 1:* The tree  $T$  has a vertex adjacent to at least four leaves.

Putting  $m = 4$  in theorem 3.3.2 yields that this case is true. By the previous corollary we can also assume that  $k \geq 8$  in all cases below.

*Case 2:* The tree  $T$  has a vertex adjacent to three leaves.

By using the same argument as in Case 1 in theorem 3.4.1 we only have to consider the situation when we have at least two vertices in  $T$ ,  $x_1$  and  $x_2$  say, such that  $d(x_1) = d(x_2) = 4$  and both  $x_i$  have exactly three adjacent leaves.

Take a vertex  $v$  in the graph  $G$  with degree at least  $k$  and the largest complete graph containing  $v$ . We will try to build the tree  $T$  in  $G$  such that  $v$  is a vertex adjacent to three leaves,  $v = x_2$  say. As in earlier proofs the leaves in the tree  $T$  adjacent to  $v$  will be added as a last step in this process and we will always build the tree  $T$  by using vertices and edges from the complete graph  $K_r$  containing  $v$  as log as possible.

Start at the vertex  $v$  and build the tree  $T$  by first using the edges and vertices in the chosen complete graph  $K_r$  containing  $v$  as possible. By the minimum degree of  $G$  we can add the first  $k - 4$  edges of the tree  $T$ , creating a subtree  $T'$  as a subgraph of  $G$ . Leaving the leaves at  $v$  for a moment we try to add the fourth last edge at a specified vertex  $u$  in  $T'$ .

First consider the sub-case when  $u \notin K_r$ . If it is not possible to add this edge at  $u$ , the vertex  $u$  has exactly the vertex set  $V(T') \setminus u$  as neighbours. This yields that either the chosen complete graph  $K_r$  was not the largest one containing  $v$ , in which case we repeat the argument, or else it is a complete subgraph of order at least  $k - 3$ .

Now consider when  $u \in K_r$ . We can assume that the vertex  $u$  is identified with  $x_1$  in this case and that this vertex is added as late as possible in the building process. If  $r < k - 5$  there will be three leaves to add at  $u$ , else  $r$  was not chosen to be maximal. If  $r = k - 5$  or  $r = k - 4$  all vertices in  $K_r \setminus v$  will have at least one neighbour outside the complete graph  $K_r$ . Let  $y$  be such a vertex and identify  $u$  with  $y$ . If  $|N(y) \cap \{V(K_r) \setminus v\}| = 1$  we will find our subtree  $T'$  on  $k - 3$  edges, and then we can add the last three leaves from  $x_2 = v$  because of the degree of that vertex. If  $|N(y) \cap \{V(K_r) \setminus v\}| > 1$  we can choose a vertex from this set to be one of the leaves from  $x_1$  and build a subtree  $T'$  on  $k - 5$  or  $k - 4$  edges respectively, using all vertices in  $K_r$  and having  $u$  outside the complete graph. Left is now to add the last leaf (or leaves) at  $u$ , which is possible due to

the choice of a maximal  $r$ , and then to add the leaves at  $x_2 = v$  which is possible due to the degree of the earlier chosen vertex  $v$ .

Since the case is trivial if  $r > k - 3$  it is enough to consider when the complete graph has order exactly  $k - 3$ . Note that no vertex in the complete graph  $K_r$  but  $v$ , has a neighbour outside the complete graph, since otherwise we could enlarge our subtree  $T'$  and finally add the remaining leaves at the earlier specified vertex  $v$ .

Now, if we can find a star  $S_4$  on four edges, edge disjoint from the complete graph  $K_r$  with a leaf at  $v$  we can find our tree  $T$  with a vertex adjacent to three leaves identified with the centre of the star  $S_4$ . Assume that this is not possible. The vertex  $v$  has a neighbour,  $y$  say, outside the complete graph  $K_r$  due to the degree of  $v$ . If  $y$  is not a centre of a star as described above, all but at most two neighbours to  $y$  belong to the complete graph  $K - r$ . But since  $k > 8$  we have a contradiction to the fact that the vertex  $y$  was not a neighbour to a vertex in the vertex set  $V(K_r) \setminus v$ .

*Case 3:* The tree  $T$  has a vertex adjacent to two leaves.

By using the same argument as in Case 2 in our previous theorem we can assume that the tree  $T$  does not contain a path of length two,  $x_1x_2x_3$  say, such that  $x_3$  is a leaf and  $d(x_2) = 2$ . Thus there are at least two vertices  $y_1$  and  $y_2$  say, such that both are neighbours with exactly two leaves and  $d(y_i) = 3$ .

Now take a vertex  $v$  in the graph  $G$  with degree at least  $k$  and the largest complete graph  $K_r$  containing  $v$ . We will try to build the tree  $T$  in  $G$  such that  $v$  belongs to the subgraph  $T$  and has two leaves as neighbours,  $v = y_1$  say. As before, the leaves adjacent to  $v$  will be added as a last step in this process and we will always build the tree  $T$  using vertices and edges from the complete graph  $K_r$  as long as possible. By the minimum degree of  $G$  we can add the first  $k - 4$  edges of  $T$  creating a subtree  $T'$  as a subgraph in  $G$  using the vertex  $v$  as described above. We can now assume that there are two leaves left to add at the last added vertex  $u = y_2$  in  $T$  and two leaves to add at  $y_1 = v$ .

First assume there is no edge present in  $G$  to add at  $u = y_2$  to enlarge our subtree  $T'$ . By the choice of a maximal  $r$  we know that  $r \geq k - 3$ , but if it was an inequality we would get a contradiction so we can assume that  $r = k - 3$ . This implies that no vertex in  $V(K_r) \setminus v$  has a neighbour outside the complete graph. But this information gives that we can build a small tree, a subtree to  $T$ , rooted at  $v$  edge disjoint from the complete graph  $K_r$  due to the minimal degree of  $v$  and then finally add vertices and edges from the complete graph to maintain our desired tree  $T$  as a subgraph of  $G$ .

Now assume there is only one edge to add at the specified vertex  $u$ . At this stage we can look at the largest complete graph  $K_s$  outside the complete graph  $K_r$  where all vertices have degree  $r - 1$  into the previous complete graph  $K_r$ . Now, when trying to build our tree  $T$ , we start by using all vertices in  $K_r$  and continue by using vertices and edges in our  $K_s$ . Assume that, after finding the maximal  $K_r$  have found a maximal  $K_s$  and that we try to add the third last edge to our subtree in our process to find the tree  $T$  as a subgraph of  $G$ . If this is not possible,  $u$  has  $r - 1$  neighbours in  $K_r$  and  $s - 1$  neighbours in  $K_s$  implying that  $r + s \geq k - 2$ . Note that the complete graph  $K_s$  only has

neighbours inside itself and in the other complete graph  $K_r$ , since otherwise we would be able to add one edge to get a larger subtree on  $k - 2$  edges and finally add the remaining leaves at the vertex  $v$ .

If we can find a star  $S_3$  edge disjoint from the complete graphs with a leaf at  $v$  we are done, so assume this is not the case. But the vertex  $v$  has a neighbour,  $w$  say, outside the complete graphs due to its degree. Since  $w$  is not a centre of a star  $S_3$  as mentioned earlier, all but at most one neighbour is inside  $V(K_r) \cup V(K_s)$ . But  $w$  can not have a neighbour inside  $K_s$ , and  $w$  is not a neighbour to all vertices in the complete graph  $K_r$ , yielding that  $r \geq k - 4$ . But in this case we can find any tree on  $k - 2$  edges using  $K_r$ ,  $K_s$  and  $w$  with the vertex  $v$  identified with  $y_1$  as before. Finally we add the remaining two leaves at the vertex  $v$ .

*Case 4:* The tree  $T$  has vertices only adjacent to at most one leaf.

In the last case we know that no vertex in the tree  $T$  is adjacent to more than one leaf. Therefore there must be at least two paths in  $T$ ,  $x_1x_2x_3$  and  $y_1y_2y_3$  say, such that  $x_3$  and  $y_3$  are leaves and  $d(x_2) = d(y_2) = 2$ .

Take a vertex  $v$  in the graph  $G$  with degree at least  $k$ , this vertex will later on be identified with  $x_2$ . Consider that we have chosen two maximal complete graphs  $K_r$  and  $K_s$  both containing  $v$  such that  $V(K_r) \cap V(K_s) = v$  and all vertices in  $K_s$  have at least  $r - 1$  neighbours in  $K_r$ . Also choose a maximal complete graph  $K_t$  outside the earlier chosen complete graphs that has at least  $r - 1$  neighbours in  $K_r$  and at least  $s - 1$  neighbours in  $K_s$ .

By earlier cases and the theorem above we can assume that we are about to add the second last edge at a specified vertex  $u$ . We are also free to choose  $u = y_2$ . As before, during the building process we will always build the tree by first using the edges and vertices from the complete graph  $K_r$  as long as possible, then from  $K_s$  and finally from  $K_t$ .

First assume that  $u \notin V(K_r) \cap V(K_s) \cap V(K_t)$ . In this case we can add our second last edge since otherwise one of the complete graphs was not chosen to be maximal.

So, by earlier cases and theorem 3.3.2 we can conclude that  $u \in V(K_t)$  and consequently that  $r + s + t \geq k - 1$  and we can assume to have equality. Note that no vertex in  $K_t$  has a neighbour outside the three chosen complete graphs.

If we can find a path of length two starting at  $v$  not using any other vertex in the three chosen complete graphs, there is nothing more to prove. So, assume this is not the case. But  $v$  has due to its degree two neighbours,  $w_1$  and  $w_2$  say, outside the chosen complete graphs. By the earlier arguments both  $w_i$  must have all its neighbours inside  $K_r$  and  $K_s$  and therefore is  $r + s \geq k - 2$  leaving  $t = 1$  as a last case. Then we will find our tree  $T$  as a subgraph of the induced graph on  $V(K_r) \cup V(K_s) \cup V(K_t) \cup \{w_1, w_2\}$  by an argument similar to the last case in the previous theorem.  $\square$

Naturally we have the following corollary:

**Corollary 3.4.3.** *The Erdős-Sós conjecture is true for every  $k \leq 9$ .*

*Proof:* Consider a reduced Erdős-Sós graph  $G$ . Since  $k - 4 \leq \lceil \frac{k}{2} \rceil \leq \delta(G)$  if  $k \leq 9$ , the corollary is true by the previous theorem.  $\square$

Also we have a corollary for small  $n$ :

**Corollary 3.4.4.** *The Erdős-Sós conjecture is true for every  $n \leq 13$ .*

*Proof:* True by the previous corollary and theorem 3.2.2  $\square$

Another, almost trivial, result is the following:

**Proposition 3.4.2.** *If  $G$  is an Erdős-Sós graph with at least  $\lceil \frac{k-9}{2} \rceil$  vertices of degree  $n - 1$ , then every tree  $T$  of size  $k$  is contained in  $G$ .*

*Proof:* By induction. If  $k \leq 9$  we know by corollary 3.4.3 that the theorem is true. Now assume the theorem to be true for all  $k$  less than  $k'$ , pick  $k = k'$  and take a graph as above. Removing one vertex  $v$  with degree  $n - 1$  and all its incident edges yields a graph  $G'$  with  $\varepsilon > \frac{n}{2}(k - 3)$  where all trees on at most  $k - 2$  edges are present as subgraphs. Take any tree  $T$  on  $k$  edges and remove a star  $S_r$  to create a subtree on at most  $k - 2$  edges. By induction is  $T'$  contained in  $G'$  as a subgraph. Adding the earlier removed vertex  $v$ , we let  $v$  be identified with the centre of the star  $S_r$  and add the remaining edges incident to  $v$  to find that our desired tree  $T$  was contained as a subgraph in  $G$ .  $\square$

### 3.5 Trees in Coloured Graphs

There seems to be many connections between vertex colouring, the greedy algorithm and the Erdős-Sós conjecture, but very few results are known in this area. However, Gyárfás, Szemerédi and Tuza points out the following in [46] as well as Sumner does in [67]:

**Theorem 3.5.1.** *Let  $G$  be a  $k$ -chromatic graph whose vertices are labelled with  $1, 2, \dots, k$  according to a good  $k$ -colouring. If  $T$  is a labelled tree of order  $k$ , then  $G$  contains a subgraph isomorphic to  $T$ . (The isomorphism is understood to be between labelled graphs.)*

Naturally we have the following corollary as an immediate consequence:

**Corollary 3.5.1.** *A  $k$ -chromatic graph contains every tree of order  $k$  as a subgraph.*

But the proof of the theorem above does work with a weaker condition on the graph  $G$  which calls for a new definition:

**Definition 3.5.1.** A graph  $G$  is called  $k$ -greedy if  $k$  is the maximal number of colours used to colour  $G$  or any of its subgraphs, using the greedy algorithm. Or more formally,  $G$  is greedy of order

$$\max_{\omega, G' \subseteq G} \chi(G', \omega),$$

where  $\omega$  is an order of the vertex set  $V(G)$  and  $\chi(G', \omega)$  denotes the number of colours the greedy algorithm uses to colour  $G'$  with the vertices in order  $\omega$ .

*Remark:* Beside having a weaker condition, the use of looking at all subgraphs of  $G$  is illustrated by the following example:

The complete bipartite graph  $K_{n,n}$  is always coloured by two colours when using the greedy algorithm, independent of the vertex order  $\omega$ . On the other hand, by the following proposition we know it is  $(n + 1)$ -greedy!

**Proposition 3.5.1.** *The complete bipartite graph  $K_{n,n}$  is  $(n + 1)$ -greedy.*

*Proof:* Take a complete bipartite graph  $K_{n,n}$  with vertices  $x_1, x_2, \dots, x_n$  and  $y_1, y_2, \dots, y_n$  in respectively part of the partition  $(X, Y)$ . Remove all edges  $x_i y_i$  when  $i < n$  and let  $\omega = \{x_1, y_1, x_2, y_2, \dots, x_n, y_n\}$ . This particular subgraph with this particular order  $\omega$  yields an  $(n + 1)$ -colouring using the greedy algorithm. Since this value attains the upper bound  $\Delta(K_{n,n}) + 1$  of how many colours the greedy algorithm can result in, we know that  $K_{n,n}$  is  $(n + 1)$ -greedy.  $\square$

Using the definition above, we obtain another theorem concerning the colouring of a graph  $G$  and the problem if the graph contains all trees of a certain size.

**Theorem 3.5.2.** *Let  $G$  be a  $k$ -greedy graph whose vertices are labelled with  $1, 2, \dots, k$  according to a  $\chi(G', \omega)$ -colouring. If  $T$  is a labelled tree of order  $k$  with a specified vertex  $v$ , then  $T$  is contained as a subgraph such that  $v$  is contained in colour class  $k$  and all other vertices are contained in different colour classes.*

*Proof:* Take a graph  $G$  with a  $\chi(G', \omega)$ -colouring as above. Identify the vertex  $v$  in the tree  $T$  with any vertex in colour class  $k$ . Assume that a subtree  $T'$  on  $k - 1$  vertices has been found having the vertex  $v$  in colour class  $k$  and the other vertices use colour classes  $2, 3, \dots, k - 1$ . The last vertex to add is a leaf that should be situated at a vertex  $u$  in colour class  $c$ . By the structure of the greedy algorithm we know that there is an edge from  $u$  to colour class 1 and so we have found  $T$  as a subgraph in  $G$  with the prescribed colourings.  $\square$

*Remark:* We can not obtain an isomorphism between labelled graphs as in theorem 3.5.1 since the notion of  $k$ -greedy is weaker than the notion of  $k$ -chromatic.

Since our interest in this paper only is to find a subgraph  $T$  in  $G$ , not depending on labels, we can make a statement with slightly weaker conditions as well. We will denote the set of vertices that have been coloured with colour  $i$  by the greedy algorithm  $C_i$ .

**Corollary 3.5.2.** *If a graph  $G$  contains a vertex  $v$  such that the induced graph on  $V(G) \setminus v$  is a  $k$ -greedy graph and  $v \in N(C_k)$ , then is every tree  $T$  of size  $k$ , i.e. order  $k + 1$ , contained in  $G$  as a subgraph.*

*Proof:* Take a tree  $T$ , a graph  $G$  and a vertex  $v$  as above. Choose any leaf  $u$  in  $T$  and create the subtree  $T \setminus u$  and the induced graph  $V(G) \setminus v$ . Let  $x$  be a neighbour to  $v$  in  $C_k$  and  $y$  the neighbour to  $u$  in  $T$ . By the preceding theorem we can identify  $y$  in the subtree  $T' = T \setminus u$  with  $x$  in the induced graph on  $V(G) \setminus v$  and find the subtree  $T'$  as a subgraph. Adding the leaf  $v$  to its neighbour  $u$  gives the result.  $\square$

Even though this result covers a great deal of the Erdős-Sós graphs, it is hard to determine if a graph is contained in the class described in the theorem above or not. We are left with rather non-informative conclusions as the following proposition, a modified result from [18].

**Proposition 3.5.2.** *For any graph  $G$ ,*

$$\chi(G) \leq \max_{\omega, G' \subseteq G} \chi(G', \omega) \leq \Delta(G) + 1.$$

*Proof:* Since there is at least one ordering  $\omega$  of the vertex set  $V(G)$  such that  $\chi(G, \omega) = \chi(G)$  we get the lower bound. The upper bound is a trivial remark about the greedy algorithm.  $\square$

But the stated corollary above could suggest that the following conjecture is true:

**Conjecture 3.5.1.** *Every Erdős-Sós graph contains a vertex  $v$  such that the induced graph on  $V(G) \setminus v$  is a  $k$ -greedy graph and  $v \in N(C_k)$ .*

Of course, this would imply that the Erdős-Sós conjecture, the subject of this chapter, is true as well.

*Remark:* The conjecture above is easily proven for  $k \leq 3$ .

### 3.5.1 Heuristics about Graph Colouring

The conjecture above has grown up from thoughts about different results and conjectures in the area concerning colouring random graphs. For a longer discussion about this subject, see [18].

Bollobás showed in [22] that for a random graph  $G$  where each edge is present with probability  $p$  has a chromatic number  $\chi(G)$  close to  $n/2 \log_b n$  where  $b = 1/(1 - p)$ .

One can ask how well the greedy algorithm will work on such a random graph  $G$  with a random order  $\omega$  of the vertices  $V(G)$ . Grimmet and McDiarmid showed in [43] that  $\chi(G, \omega)$  is close to  $n / \log_b n$ , telling us that it is high probability that the greedy algorithm will use twice as many colours as needed in a random graph.

In the appendix to [2] Erdős has a discussion about what chromatic number certain classes of random graphs has. One class, not equal, but pretty close to reduced Erdős-Sós graphs, is there conjectured to have chromatic number close to its minimum degree.

This heuristic argument could suggest a solution for almost all Erdős-Sós graphs, since using the greedy algorithm on a reduced Erdős-Sós graph would be close to twice its minimal degree and then use the above knowledge about those graphs.

But even if this heuristic argument would work in detail there will still be, as in the case of proofs using the Regularity Lemma, not enough to prove the conjecture in all its instances.

Also it would be interesting to see results concerning the introduced concept greedy, or how to colour a graph as bad as possible using the greedy algorithm being allowed to omit edges in the graph  $G$ .

### 3.5.2 Definitions of Some Classes of Trees

For the reader not familiar with certain classes of trees mentioned in the section about earlier results we will present proper definitions of double-stars, comets and spiders. These definitions were also made in the first chapter about graph theory.

**Definition 3.5.2.** (double-star)

A double-star with distance  $r$  between the centres is a tree obtained from two stars and a path by identifying the leaves of a path of length  $r$  to the centre of each star. If  $r = 1$  we say that the tree is a double-star omitting the length of the path between the centres of the two stars.

**Definition 3.5.3.** (comet)

A comet is a tree obtained from a star and a path by identifying one leaf of the star with one leaf of the path.

**Definition 3.5.4.** (spider)

A spider is a tree obtained from a star by subdividing its edges.

## Chapter 4

# The Loeb-Komlós-Sós Conjecture

### 4.1 Introduction

In the last chapter we dealt with a conjecture that said that any tree of a specified size was contained as a subgraph in a graph  $G$  if the mean degree was large enough. This chapter is about a similar problem, but here is the concept of mean degree changed to the concept of median degree for the studied graph.

As before we shall use standard graph theory notation and we consider only finite undirected graphs of order  $n = |V(G)|$  and size  $\varepsilon(G) = |E(G)|$ . All graphs will be assumed to have neither loops nor multiple edges. For other graph theoretic notions not defined in this or previous chapters, see [21], [24] and [37].

The below conjecture was firstly formulated by Loeb in 1995 in the case  $k = \frac{n}{2}$  and next generalised by Komlós and Sós.

**Conjecture 4.1.1.** (*Loeb-Komlós-Sós*)

*If  $G$  is a graph of order  $n$  and at least  $\frac{n}{2}$  vertices have degree at least  $k$ , then  $G$  contains all trees of size  $k$ .*

This conjecture has some similarities with the conjecture by Erdős and Sós which was discussed in the previous chapter:

**Conjecture 4.1.2.** (*Erdős-Sós*)

*If  $G$  is a graph of order  $n$  with size strictly greater than  $\frac{n}{2}(k-1)$ , then  $G$  contains all trees of size  $k$ .*

As pointed out by the authors in [49], the condition on the average degree in the Erdős-Sós conjecture is replaced by a condition on the median degree in the Loeb-Komlós-Sós conjecture. For the reader interested in special cases of the Erdős-Sós conjecture see for example [10] and [69]. So far, not much has been proved for the Loeb-Komlós-Sós conjecture, but in [17] we find the following non-trivial results by Bazgan, Li and Woźniak:

**Theorem 4.1.1.** *If  $G$  is a graph of order  $n$  and it has at least  $\frac{n}{2}$  vertices with degree at least  $k$ , then  $G$  contains a path of length at least  $k$ .*

**Theorem 4.1.2.** *Let  $G$  be a graph of order  $n$  where at least  $\frac{n}{2}$  vertices have degree at least  $k$ , then  $G$  contains all stars and double-stars of size  $k$ .*

Since any tree of size at most 4 is a path, star or a double-star, we also have the following result for small values of  $k$ .

**Corollary 4.1.1.** *The Loeb-Komlós-Sós conjecture is true for every  $k \leq 4$ .*

To be mentioned is the approximate result concerning the case  $k = \frac{n}{2}$  by Ajtai, Komlós and Szemerédi using the Regularity Lemma [48].

**Theorem 4.1.3.** *For every  $\epsilon > 0$  there is a threshold  $n_0$  such that for all  $n \geq n_0$ , if  $G$  is a graph of order  $n$  and it has at least  $\frac{(1+\epsilon)n}{2}$  vertices of degree at least  $\frac{(1+\epsilon)n}{2}$ , then  $G$  contains all trees with  $\frac{n}{2}$  edges.*

This result was strengthened later on by Yi Zhao who proved an exact result for the case  $k = \frac{n}{2}$  and sufficiently large  $n$  in [72].

## 4.2 Results on the Loeb-Komlós-Sós Conjecture

### 4.2.1 Caterpillars with Many Legs

The aim of this subsection is to show that the Loeb-Komlós-Sós conjecture is true for all caterpillars with at least  $k - 4$  legs. But since we already know the conjecture to be true for stars and double-stars, it suffices to prove it true for caterpillars with exactly  $k - 4$  legs, this will be defined below. We will use the same notation as in [17] letting  $B = \{v \in V(G) | d(v) \geq k\}$  and  $S = V(G) - B$  in a graph satisfying the hypothesis of the Loeb-Komlós-Sós conjecture. But first we state a theorem of Barr from the previous chapter, also found in [10], which we will use in the proof of theorem 4.2.2 below.

**Theorem 4.2.1.** *Let  $G$  be a graph of order  $n$  with more than  $\frac{n}{2}(k - 1)$  edges. If the graph  $G$  has minimum degree  $\delta \geq k - 4$ , then  $G$  contains every tree of size  $k$ .*

To avoid confusion we define a caterpillar as follows:

**Definition 4.2.1.** A caterpillar is a tree such that if we remove all leaves and their incident edges, the remaining tree is a path. If a caterpillar has  $r$  leaves, seen as a tree, we say that the caterpillar has  $r - 2$  legs. (Assuming all caterpillars to have one head and one tail.)

*Remark:* In the case, when the tree only consists of one edge and the two vertices at its ends, we define it as the smallest possible caterpillar. Removing the two vertices and the edge will result in an empty graph. Questions concerning an empty graph being a path could be risen, but here we answer that question with a yes, in order to create a proper definition.

**Theorem 4.2.2.** *If  $G$  is a graph of order  $n$  and it has at least  $\frac{n}{2}$  vertices of degree at least  $k$ , then  $G$  contains all caterpillars with exactly  $k - 4$  legs.*

*Proof:* Let  $G$  be a graph as above and assume that the induced graph on  $B$  contains a path of length 2 as a subgraph. Due to the degree of the vertices in  $B$  we can add leaves to the existing path constructing any caterpillar with  $k - 4$  legs.

Now assume that no such path exists. This implies that every vertex in  $B$  sends at least  $k - 1$  edges into  $S$ , forcing  $S$  to have order  $\frac{n}{2}$  and every vertex in  $S$  to have degree  $k - 1$ . This graph satisfies the hypothesis of the Erdős-Sós conjecture and has minimum degree  $k - 1$ . By theorem 4.2.1 we know that this graph contains all trees of size  $k$ .  $\square$

### 4.2.2 Trees with Diameter 4 and Low Central Degree

The aim of this subsection is to prove the Loebel-Komlós-Sós conjecture for trees of diameter four where the central vertex has a degree of at most  $\frac{k}{2}$ . The proof is quite straightforward, using the fact from [17] that we only have to consider graphs where  $S$  is an independent set.

**Theorem 4.2.3.** *If  $G$  is a graph of order  $n$  and it has at least  $\frac{n}{2}$  vertices of degree at least  $k$ , then it contains all trees of size  $k$  with a diameter 4 where the central vertex is of degree at most  $\frac{k}{2}$ .*

*Proof:* Take a graph  $G$  and a tree  $T$  as above. First assume that the induced graph on  $B$  contains a star of size  $\frac{k}{2}$ , then, due to the degrees of the vertices in  $B$ , we are done. So, we can assume that every vertex in  $B$  sends at least  $\frac{k}{2}$  edges into  $S$ . But since  $S$  is an independent set and  $|S| \leq |B|$  there must be a vertex  $v$  in  $S$  sending at least  $\frac{k}{2}$  edges into  $B$ . Now, due to the degrees of the vertices in  $B$ , we can find our desired tree  $T$  with its central vertex identified with the vertex  $v$ .  $\square$

### 4.2.3 Caterpillars with All Legs at One Vertex

The next family of trees we will prove the Loebel-Komlós-Sós conjecture for is another subset of caterpillars. We will consider the class of trees of all caterpillars with all legs at one vertex. Then we conclude that the theorem below, together with the earlier ones in this section, allow us to say that the Loebel-Komlós-Sós conjecture is true for every  $k \leq 6$ .

**Theorem 4.2.4.** *If  $G$  is a graph of order  $n$  and it has at least  $\frac{n}{2}$  vertices of degree at least  $k$ , then  $G$  contains all caterpillars of size  $k$  with all legs at one vertex.*

*Proof:* Take a graph  $G$  and a tree  $T$  as above. By the theorem of Bazgan, Li and Woźniak we know there is a path  $P$  of length  $k$  in the graph  $G$ ,  $P = v_1 v_2 \dots v_k$  say. Assuming that  $T$  is not a path, we continue our proof. Knowing that  $S$  is an independent set we know that at least one of two consecutive vertices on the path  $P$  must be contained in  $B$ . Let  $u_1 u_2 \dots u_r$  be the longest path in our tree  $T$  and let all legs of this caterpillar be situated at  $u_s$ . Now if  $v_s$  is contained in  $B$  we identify  $u_i$  with  $v_i$  for all  $i \in \{1, 2, \dots, r\}$  and use the degree of  $v_s$  to add all remaining leaves to our desired tree. Otherwise is  $v_{s+1}$

contained in  $B$  and we identify  $u_i$  with  $v_{i+1}$  for all  $i \in \{1, 2, \dots, r\}$  and use the degree of  $v_{s+1}$  to add all remaining leaves to  $T$ .  $\square$

**Corollary 4.2.1.** *The LoebL-Komlós-Sós conjecture is true for every  $k \leq 6$ .*

*Proof:* Every tree of size at most 6 is a caterpillar with at least  $k - 4$  legs, a caterpillar with all legs situated at the same vertex, a tree of diameter 4 with a central vertex of degree at most  $\frac{k}{2}$ , or a path.  $\square$

Combining this with the result of Bazgan, Li and Woźniak we get a result for small  $n$ .

**Corollary 4.2.2.** *The LoebL-Komlós-Sós conjecture is true for every graph of order at most 10.*

#### 4.2.4 Caterpillars with Diameter at most 5

In this subsection we will extend the family of caterpillars where the LoebL-Komlós-Sós conjecture is true. We will make use of some partial results on the Erdős-Sós conjecture made earlier in this thesis, that also can be found in [10]. As before, we define  $B = \{v \in V(G) \mid d(v) \geq k\}$  and  $S = V(G) \setminus B$ . Also, we can suppose that  $S$  is an independent set, otherwise the graph obtained from  $G$  by removing the edges between the vertices in  $S$  also would satisfy the hypothesis of the LoebL-Komlós-Sós conjecture, a fact taken from [17].

**Theorem 4.2.5.** *If  $G$  is a graph of order  $n$  and it has at least  $\frac{n}{2}$  vertices of degree at least  $k$ , then  $G$  contains all caterpillars with diameter at most 5.*

*Proof:* Due to earlier results we only have to consider caterpillars with diameter equal to 5. Take a graph  $G$  and a tree  $T$  as above. First assume that the induced graph on  $B$  contains a path of length 3 as a subgraph. Due to the degrees of the vertices in  $B$  we can add leaves to the existing path to construct any caterpillar with diameter 5.

Now assume no such path exists and that we are unable to find our tree  $T$  in  $G$ . Since the Erdős-Sós conjecture is true for paths we know that the mean degree in the induced graph on the set  $B$  is at most 2. This implies that the mean degree of the vertices in  $S$  is at least  $k - 2$ . Let  $B_0$  be the set of vertices in  $B$  only having neighbours in  $S$  and let  $B_1 = B \setminus B_0$ . Also, let  $S_0$  be the vertices in  $S$  having degree at most  $k - 3$  and let  $S_1 = S \setminus S_0$ .

Note that there are no edges between  $B_0$  and  $B_1$ , else we could easily find a copy of our tree  $T$  in the graph  $G$ . Moreover, there are no edges from  $B_1$  to  $S_1$  since otherwise we could construct a path on four vertices, all of them of degree  $k$  but one (not being an

end vertex) of degree  $k - 2$  or  $k - 1$ . This path could always be extended to our tree  $T$ , using the degrees of the vertices, contradicting our assumption.

We can also establish the fact that every vertex  $v$  in  $B_0$  has at most one neighbour in  $S_1$ . Assuming  $v$  to have at least two neighbours in  $S_1$  yields that we can extend this to a path on four vertices alternating between the two sets  $B_0$  and  $S_1$ . This path can always be extended to our caterpillar with exactly  $k - 5$  legs, using the degree of the vertices and the fact that the two vertices in  $S_1$  are not joined by an edge, contradicting our assumption.

This implies that every vertex in  $B$  has, in average, at least  $k - 2$  neighbours in  $S_0$  contradicting the construction of  $S_0$  unless it is an empty set. But, if  $S_0$  is an empty set,  $B_0$  must be an empty set as well and no vertex in  $B$  has a neighbour in  $S$  giving us a path of appropriate length in  $B$  which proves our theorem.  $\square$

### 4.2.5 Trees with Diameter at most 4

Above, the Loebel-Komlós-Sós conjecture has been proved true for a subset of the trees of diameter 4. With some new observations we can conclude that the conjecture is true for all trees of diameter 4.

**Theorem 4.2.6.** *If  $G$  is a graph of order  $n$  and it has at least  $\frac{n}{2}$  vertices of degree at least  $k$ , then  $G$  contains every tree on  $k$  edges of diameter at most 4 as a subgraph.*

*Proof:* Take a graph  $G$  as above and a tree  $T$  on  $k$  edges of diameter at most 4. Let  $w$  be a vertex of  $T$  such that the distance to every other vertex in  $T$  is at most 2. Put  $m = d(w)$ ,  $N(w) = \{L_1, L_2, \dots, L_m\}$  where  $d(L_1) \leq \dots \leq d(L_m)$ ,  $p = \max\{i : d(L_i) = 0\}$  and  $b = |B|$ . Note that

$$p \geq 2m - k.$$

If a vertex in  $S$  has degree at least  $m$ , then, since  $S$  is an independent set, it has at least  $m$  neighbours in  $B$ . In this case it is easy to find an embedding of  $T$  in  $G$  due to the degrees of the vertices in  $B$ . Hence suppose that  $d(v) \leq m - 1$  for all  $v \in S$ . Then

$$|E(B, S)| \leq (m - 1)(n - b).$$

By the same argument, if a vertex  $u \in B$  has degree at least  $m - p$  in  $B$  we can find an embedding of  $T$ . Hence, assume that  $d(u, B) \leq m - p - 1$  for all  $u \in B$ . Then we get the following lower bound:

$$|E(B, S)| \geq b(k - m + p + 1) \geq b(m + 1).$$

Combining our upper and lower bound on  $|E(B, S)|$  we get that  $2mb \leq n(m - 1)$ , implying that  $b \leq \frac{n}{2}$ , contradicting the hypothesis of the theorem.  $\square$

These results leave us with one remaining tree of size 7 to close the case when  $k \leq 7$ .

**Proposition 4.2.1.** *The Loeb-Komlós-Sós conjecture is true when  $k \leq 7$ .*

*Proof:* The remaining tree  $T$ , to close the case  $k \leq 7$ , can be constructed from a star on three edges by subdividing two of its edges once and one of its edges twice. Take a graph with minimum degree 7 and a tree  $T$  as described. Assume that  $G$  is a minimal counter example, hence we can not find  $T$  as a subgraph in  $G$ . Now, let  $T^*$  be the remaining tree after deleting all leaves and their incident edges from  $T$ . If  $T^*$  is contained in  $B$  we are done due to the degrees of the vertices in  $B$ . Since the Erdős-Sós is true for  $T^*$  we know that the mean degree in the graph induced on  $B$  is at most 3. Consequently, there is a vertex  $v$  in  $S$  with degree at least 4. We embed  $T$  as follows: Let the centre of the subdivided star be a vertex  $v$  as above with neighbourhood  $N(v)$  and continue with any three neighbours in  $B$ . If any of these vertices in  $N(v)$  has a neighbour in  $B$  we are done due to the degrees of the vertices in  $B$ . So,  $N(v)$  must be an independent set and its neighbourhood is a subset of  $S$ . Now, if  $|N(N(N(v)))| > |N(v)|$  it is clear we can embed our tree  $T$  in  $G$ , so assume equality holds. In this case is  $G \setminus \{N(v) \cup N(N(v))\}$  a smaller counter example than the chosen minimal one and we have a contradiction.  $\square$

## Chapter 5

# Paths in Complete Graphs

This note concerns simple complete graphs,  $K_n$  with a given edge colouring. We will let  $G^c$  denote a graph  $G$  whose edges are coloured in an arbitrary way with  $c$  colours. A properly coloured cycle in  $G^c$  is a cycle in which adjacent edges have different colours. Such cycles are termed *alternating cycles*. In particular we are here interested in alternating hamiltonian cycles; alternating cycles that pass once through every vertex of the graph. If this is too hard to achieve, we look for an alternating hamiltonian path; an alternating path that pass once through every vertex of the graph.

Bollobás and Erdős, in [23] considered this problem in coloured complete graphs  $K_n^c$ , that is, a complete graph coloured with  $c$  colours. Let  $\Delta(G^c)$  denote the maximum degree in  $G^c$  in any one colour. Bollobás and Erdős made the following conjecture:

**Conjecture 5.0.1.** (*Bollobás and Erdős*)

*Every  $K_n^c$  with  $\Delta(K_n^c) \leq \lfloor n/2 \rfloor - 1$  contains an alternating hamiltonian cycle.*

This conjecture, if true, would provide a sharp bound, as can be seen by letting  $n = 4k + 1$ . Then there clearly exists a  $K_n^2$  so that both its monochromatic subgraphs are regular of degree  $2k$ . However, such a graph does not contain an alternating hamiltonian cycle, since  $n$  is odd and there are only two colours to choose from.

The best known result so far towards the above conjecture is due to Alon and Gutin in [4]:

**Theorem 5.0.7.** *For every  $\epsilon \in (0, 1 - 1/\sqrt{2})$  there exists an  $n_0 = n_0(\epsilon)$  so that, for every  $n > n_0$ , every  $K_n^c$  satisfying*

$$\Delta(K_n^c) \leq (1 - 1/\sqrt{2} - \epsilon)n$$

*contains an alternating hamiltonian cycle.*

Instead of searching for properly coloured hamiltonian cycles as above, we can restrict ourself to look for properly coloured hamiltonian paths. We drop the restriction on one single edge, and in this case there are positive results for complete graphs fulfilling the Bollobás-Erdős condition above as in [7].

**Theorem 5.0.8.** (*Bang-Jensen, Gutin and Yeo*)

*Every edge-coloured complete graph satisfying the Bollobás-Erdős condition has an alternating hamiltonian path.*

While many other results in this area has concerned necessary conditions on a colouring of a complete graph containing a properly coloured hamiltonian path, we will here

present, as described in a survey [44], 'a somewhat unexpected result', a sufficient condition on a complete graph with the mentioned property. The proof is very similar to the well known Redei Theorem [56] that ensures a directed hamiltonian path through any tournament. This result has earlier been published in *ARS Combinatoria* [14].

**Theorem 5.0.9.** *Every complete graph  $K_n^c$  with an edge colouring in  $c$  colours, not containing monochromatic triangles, has a properly coloured hamiltonian path.*

*Proof:* By induction. If  $n = 1$  the result is trivial. Now assume the theorem to be true for  $n = k - 1$  and take a complete graph on  $k$  vertices with an edge colouring as above. By assumption, any induced subgraph on  $k - 1$  vertices has a properly coloured hamiltonian path,  $v_1v_2 \dots v_{k-1}$  say. Assume that it is impossible to enlarge the path to a properly coloured hamiltonian path on  $k$  vertices.

Now we know that the edge  $v_kv_1$  has the same colour as the edge  $v_1v_2$ , otherwise we could easily do our extension. So, since there are no monochromatic triangles in the graph, the edge  $v_kv_2$  must have a different colour from those mentioned earlier. But now, the colour of  $v_kv_2$  must be the same as  $v_2v_3$  has, else is  $v_1v_kv_2v_3 \dots v_{k-1}$  a properly coloured hamiltonian path. By the same argument we get that every pair of edges  $v_kv_i$  and  $v_iv_{i+1}$  has the same colour when  $i \in \{1, 2, \dots, k - 2\}$ .

In particular we know that  $v_kv_{k-2}$  and  $v_{k-2}v_{k-1}$  have the same colour, so the edge  $v_{k-1}v_k$  has a different colour from those two edges. But then we know that  $v_1v_2v_3 \dots v_{k-1}v_k$  is a properly coloured hamiltonian path, contradicting the assumption that no such path existed.  $\square$

This result, with a local restriction implying a global result, could give rise to new questions concerning which local restrictions that could enforce an alternating hamiltonian path or cycle in a  $K_n^c$ .

## Chapter 6

# Graphs, Segmentation and Linear Programs

In this chapter we will suggest a method for automatic detection, segmentation and classification of textured regions in colour images. Beside using the graph theoretic framework, which has been used before, we will here describe how prior information can be brought into this framework through the use of terminal node weights and learning techniques.

### 6.1 Introduction

Graph theory is a relatively new area in mathematics, starting with Euler's paper from 1736 about the bridges of Königsberg (today Kaliningrad), and has been growing rapidly during the last fifty years. Numerous of theorems has been proved about the structure of graphs and other properties. At the same time, along the development of computers, graph theory has been a very successful way of describing discrete problems and many graph theoretic algorithms have been developed to solve these problems. Among these algorithms we find for example the simplex method for solving problems of linear programming. This method has an equivalent formulation that uses the graph theoretic framework.

Here, we will not enumerate all other different kinds of applications of graph theory that have been made throughout the years, but only discuss a couple of different ways of segmenting images with the help of graph theory. The first one uses the Ford-Fulkerson theorem about maximal flow and minimal cut as a main ingredient, and the other method discussed will be the above mentioned graph theoretic method to solve linear programming, before entering the second half of this thesis that is covering new developments of the perceptron algorithm.

### 6.2 Segmentation and Classification

Many day to day tasks for us humans can be seen as tasks where we automatically do segmentation from images and classifications of what has been seen. For instance, when we read a text, we have done a lot of preprocessing before we actually can read the words. The different letters have been segmented out from the background and after this classified into which letter it intends to represent. These tasks are made with great speed and they have been modelled by computers to work at great speed as well. Nevertheless, we can think of much more complex situations where it would take a trained expert many

minutes to do a proper segmentation and classification. Later on in this chapter we will exemplify this with the task of segmenting out corals from an underwater image and classify the coral segments into different species.

This chapter will primarily consider a method to segment an image, even though we as well take the problem of classification in account. In the coming chapters, that to largest extent are about the perceptron algorithm, we are instead focusing on the problem of classification.

## 6.3 Maximal Flow and Minimal Cut

### 6.3.1 Introduction

Image segmentation can be defined as the task of distinguishing objects from background in unseen images. Typically this division is based on low-level cues such as intensity, homogeneity or contours. Four popular approaches based on such cues are threshold techniques, edge-based methods, region-based techniques and connectivity-preserving relaxation methods.

Regardless of the approach, the difficulty lies in formulating and including prior knowledge into the segmentation process. How does one describe ones perception of what constitutes foreground in an arbitrary image through low level cues? As distinguishing between foreground and background becomes harder and requires a higher level of scene understanding this task becomes increasingly difficult.

In this work we attempt to address this issue. By combining existing image segmentation approaches with simple learning techniques we seek to include prior knowledge into this visual grouping process. We wish to partition images into two parts based on previously seen example segmentations.

The approach taken here is based on graph cut techniques. This was motivated by the simple fact that it has been one of the more successful approaches in image segmentation. In addition, as it will be seen, it also allowed for a straightforward incorporation of prior knowledge into its formulation. A suggestion for an efficient implementation along with some preliminary results on two different types of images are also given.

In terms of computer vision subfields, the proposed technique could be seen as being placed somewhere between segmentation, classification and detection.

### 6.3.2 Theory of Graph Cuts

A graph cut is the process of partitioning a directed or undirected graph into disjoint sets. The concept of optimality of such cuts is usually introduced by associating an energy to each cut. Problems of this kind have been well studied within the field of graph theory but can for graphs with more than only a few nodes be notoriously difficult. Nevertheless, ever since it became apparent that many low-level vision problems can be posed as finding

energy minimising cuts in graphs these techniques have received increased attention in the computer vision community. Graph cut methods have been successfully applied to stereo problems, image restoration, texture synthesis and image segmentation. Below we give a brief overview of graph cuts for image segmentation as well as an introduction to some basic definitions.

Given a graph  $G = \{V, E, W\}$ , where  $V$  denotes its nodes,  $E$  its edges and  $W$  the affinity matrix, which associates a weight to each edge in  $E$ . A cut on a graph is a partition of  $V$  into two subsets  $A$  and  $B$  such that

$$A \cup B = V, \quad A \cap B = \emptyset.$$

Perhaps the simplest and best known graph cut method is the min-cut formulation. The min-cut of a graph is the cut that partitions  $G$  into disjoint segments such that the sum of the weights associated with edges between the different segments is minimised. That is, the partition that minimises

$$C_{min}(A, B) = \sum_{u \in A, v \in B} W_{uv}. \quad (6.1)$$

However, as this is an NP-hard combinatorial optimisation problem, the task of finding the solution can be a formidable one. In order to overcome this one can relax (6.1) into a semi-definite program [68], resulting in a convex problem for which efficient solvers exist. However, the task of finding the solution to the original problem from the relaxed one still remains an open issue. Another commonly used approach is based on a slight reformulation of the original min-cut problem. By adding the requirement that two predefined nodes, denoted terminal nodes or source and sink nodes, in  $G$  must be in separate sets, the complexity of the problem is significantly reduced. Finding the min-cut separating the source and the sink, the so called s-t cut, can be achieved in polynomial time [26]. If one views the weights associated to each node as a flow capacity it can be shown that the maximal amount of a flow from source to sink is equal to the capacity of a minimal cut. Therefore the min-cut problem is also known as the max-flow problem.

### 6.3.3 The Image Seen as a Graph

The general approach to constructing an undirected graph from an image is shown below in figure 6.1.

Basically each pixel in the image is viewed as a node in a graph, edges are formed between nodes with weights corresponding to how similar two pixels are, given some measure of similarity, as well as the distance between them. In attempt to reduce the number of edges in the graph only pixels within a smaller, predetermined neighbourhood  $\mathcal{N}$  of each other

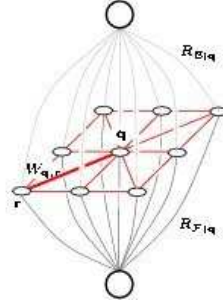


Figure 6.1: Graph representing a 3-by-3 image. (Image courtesy of Yuri Boykov.)

are considered. The two terminal nodes, the source and the sink does not correspond to any pixel in the image but instead are viewed as representing the object and background respectively. Edges are formed between the source and sink and all other non-terminal nodes, where the corresponding weights are determined using models for the object and background.

The min-cut of the resulting graph will then be the segmentation of the image at hand. This segmentation should then be a partition such that, owing to the definition of image-pixel resemblance, similar pixels close to each other will belong to the same partition. In addition, as a result of the terminal weights, pixels should also be segmented in such a manner so they end up in the same partition as the terminal node corresponding to the model (object or background) they are most similar to. An illustration of the segmentation process can be seen in figure 6.2.

The edge weight between pixel  $i$  and  $j$  will be denoted  $W_{ij}^I$  and the terminal weights between pixel  $i$  and the source (s) and sink (t) as  $W_i^s$  and  $W_i^t$  respectively and are given by

$$W_{ij}^I = e^{-\frac{r(i,j)}{\sigma_R}} e^{-\frac{\|\mathbf{w}(i) - \mathbf{w}(j)\|^2}{\sigma_W}}, \quad (6.2)$$

$$W_i^s = \frac{p(\mathbf{w}(i)|i \in s)}{p(\mathbf{w}(i)|i \in s) + p(\mathbf{w}(i)|i \in t)}, \quad (6.3)$$

$$W_i^t = \frac{p(\mathbf{w}(i)|i \in t)}{p(\mathbf{w}(i)|i \in s) + p(\mathbf{w}(i)|i \in t)}. \quad (6.4)$$

Here  $\|\cdot\|$  denotes the Euclidean norm,  $r(i, j)$  the distance between pixel  $i$  and  $j$  and  $\lambda$ ,  $\sigma_R$  and  $\sigma_W$  are tuning parameters weighting the importance of the different features. Hence,  $W_{ij}^I$  contains the inter-pixel similarity, that ensures that the segmentation is more

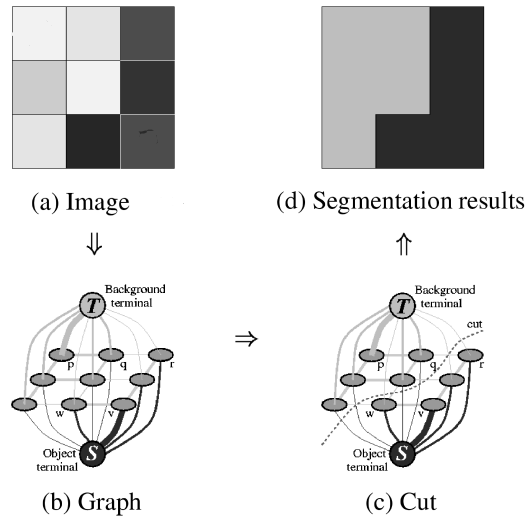


Figure 6.2: Example segmentation of a very simple 3-by-3 image. Edge thickness corresponds to the associated edge weight. (Image courtesy of Yuri Boykov.)

coherent.  $W_i^s$  and  $W_i^f$  describes how likely a pixel is to being background and foreground respectively.

#### 6.3.4 Image Descriptors, Pixel Models and Prior Knowledge

As mentioned in the previous section prior knowledge is incorporated into the graph cut framework through the terminal nodes. For this purpose we need a way to describe each pixel as well as model the probability of that pixel belonging to the foreground or the background.

The image descriptors used in the current implementation are based on texture and colour. For texture descriptors we used the output of a bank of 30 Gabor filters. These are a type of complex valued filters that are defined by harmonic functions modulated by a Gaussian distribution. Their close relation to processes in the primal visual cortex along with a number of additional desirable filter properties has made them very popular within the image processing community. The three colour channels are simply appended to this 60-dimensional, real-valued vector resulting in a 63-dimensional descriptor vector  $v$  for each pixel in the image  $I$ .

The probability distribution for these descriptors is modelled using a Gaussian Mix-

ture Model (GMM).

$$p(v|\Sigma, \mu) = \sum_{i=1}^k \frac{1}{\sqrt{2\pi|\Sigma_i|}} e^{(-\frac{1}{2}(v-\mu_i)^T \Sigma_i^{-1} (v-\mu_i))}.$$

From a number of training images, as exemplified in figure 6.3, with hand labelled regions the GMM parameters are then fitted through Expectation Maximisation, [34]. This fitting is only carried out once and can be viewed as the learning phase of our proposed method.

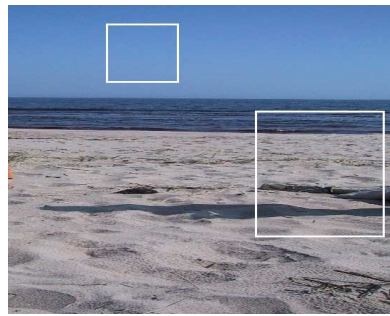


Figure 6.3: Example training image with two hand labelled regions. Left: sky. Right: non-sky.

### 6.3.5 Experiments

The examples presented below are all preceded by a training phase, one per object class, as described above. This is the a priori information that will determine the weights of the edges of the graph that represents the image. The inter-pixel similarity is computed according to (6.2) and the terminal weights from (6.3-6.4) and section 3 to form the affinity matrix. The resulting graph can then be cut, or segmented in low order polynomial time by the algorithm proposed by [26].

We have evaluated our method on two different images, an underwater image of a coral reef and an ordinary holiday picture, and three different object categories. In the coral images the goal was to detect and segment out bleached coral and for the holiday snaps the two object categories were sky and sand. The results are shown in figs. 6.4 and 6.5.

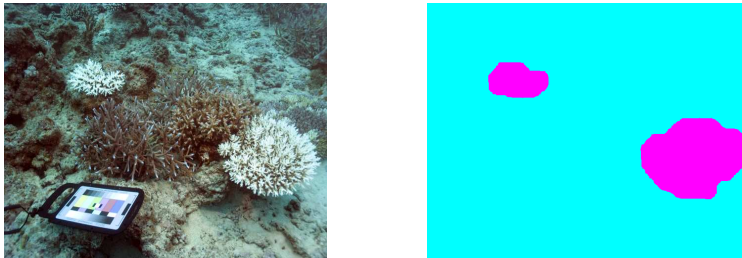


Figure 6.4: Segmentation of an image of a coral reef into diseased coral and background respectively.

For average size images, 320-by-320 pixels, the run-time for this method on a standard PC is approximately 2 minutes. On the other hand, Håkan Ardö, a colleague in Lund, has implemented the algorithm with an optimised code for this particular type of graph, [3], resulting in a solution for the algorithm with at least two frames per second for images of size 640-by-480 pixels, which in turn can be used for different applications.

### 6.3.6 Conclusions

In this chapter we have suggested a method for automatic detection, segmentation and classification of textured regions in colour images. It describes how prior information can be brought into a graph cut framework through the use of terminal node weights and learning techniques. An efficient implementation is also presented along with some very promising results on an underwater image of a coral reef as well as an ordinary holiday picture.

Future work includes an more thorough examination of different object and background models. The choice of model order of the Gaussian mixture models should be made automatic. The image descriptors and the issue of scale invariance needs addressing. Finally, multiway segmentation as well as the possibilities of including prior shape information into the segmentation process could also prove to be promising candidates for continued research.

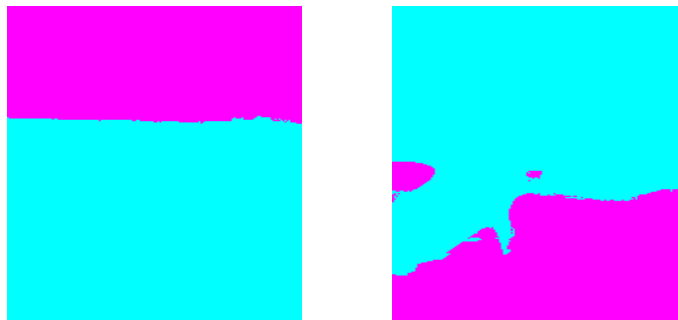


Figure 6.5: Segmentation of an ordinary holiday picture (top) into sky and background respectively, and into sand and background respectively.

### 6.3.7 Graph Theory and Linear Programs

In this chapter we have presented some results on how the graph theoretic theorem about maximal flow and minimal cut can be used to segment images. But this graph theoretic theorem can be used as a more general tool when constructing algorithms for solving mathematical problems.

For instance we can solve problems in linear programming, instead of using the well-known simplex algorithm, by translating the problem to the one of finding a minimal cut in a graph. For the reader interested in the technical details concerning the translation of the problem to a graph problem we recommend [31]

Since many of the problems concerning segmentation and classification can be regarded as special instances of solving linear programs we will now, for the rest of the thesis, concentrate on linear programming instead. And even though there are strong

connections between this subject and graph theory, this will hardly be mentioned later on. We will take a closer look at the perceptron algorithm instead, another discrete algorithm that has been used for solving these types of questions.

#### **Acknowledgements**

The work in this chapter has been funded by the European Commission's Sixth Framework Programme under grant no. 011838 as part of the Integrated Project SMERobot.



## Chapter 7

# The Perceptron Algorithm and Linear Programs

## 7.1 Introduction

The Perceptron Algorithm was introduced by Rosenblatt in [58] and has been well-studied by mathematicians and computer scientists since then. The algorithm is mistake-driven and has a very simple form of updating.

The perceptron as such was proposed and designed to model the human brain and the human recognition of visual patterns. The model involved a retina whose elements (cells) were linked to so called predicates whose purpose was the recognition of elementary patterns occurring on the retina.

Today, even though the perceptron no longer is thought of as a good model for the human brain, the algorithm and its developments are used in areas as machine learning, pattern recognition, separation and segmentation.

By connecting several layers of perceptrons, computer scientists have developed neural networks, and the simple single layer perceptron is still used for finding feasible solutions to linear programs.

This thesis will only consider the perceptron algorithm as such and throughout this work, the interpretation of the algorithm will be as a solver of linear programs, even though we will discuss some other interpretations briefly.

Here we will show earlier modifications to make it work at a greater speed, but notice that this modification has made the algorithm a randomised one. Further on we will present a way of dealing with this side-effect, finally yielding an algorithm that is both faster than the previous modification and at the same time having a deterministic behaviour. This result in turn can be used for approximating an optimal solution to the problem of finding a maximal margin for the feasibility version of a linear program, and so approximate an optimal solution.

### 7.1.1 A Brief History of the Perceptron Algorithm

The first iterative algorithm for learning (binary) linear classifications is the procedure proposed by Frank Rosenblatt in 1956 for the perceptron, see [58]. It is an on-line and mistake-driven procedure, starting at an initial vector  $w$ , updating the vector with the procedure every time a training point is misclassified.

In the case of the ( $n$ -dimensional) data being linearly separable, the process ends after a finite number of steps by presenting a hyperplane, an affine subspace of dimension

$n - 1$ , which divides the space into two parts, corresponding to the two distinct classes that we aimed to separate.

To describe the perceptron by Rosenblatt, we introduce some notation: Let  $X$  denote the input space and  $Y$  denote the output domain. Usually we have  $X \subseteq \mathbb{R}^n$  and  $Y = \{-1, 1\}$  in the case of a binary classification. A training set is a collection of training examples, which we will denote as

$$S = ((x_1, y_1), \dots, (x_m, y_m)) \subseteq (X \times Y)^m,$$

where  $m$  is the number of examples. Also we let  $(w, b)$  denote a hyperplane with normal vector  $w$ , the distance  $b$  from the origin to the hyperplane in the direction of  $w$ . further on, the learning rate will be denoted  $\eta$ . With this we can describe the algorithm as in [32] for instance:

**Algorithm 7.1.1.** *The Perceptron Algorithm (primal form)*

*Input: A linearly separable training set  $S$  and a learning rate  $\eta \in \mathbb{R}^+$ .*

*Output: A hyperplane  $(w, b)$  such that  $y_i(w \cdot x_i + b) > 0$  for every  $i$ .*

1.  $w \leftarrow 0, b \leftarrow 0$  and  $R \leftarrow \max_i \|x_i\|$
2. repeat
3.   for  $i = 1$  to  $m$
4.     if  $y_i(w \cdot x_i + b) \leq 0$  then
5.        $w \leftarrow w + \eta y_i x_i$
6.        $b \leftarrow b + \eta y_i R^2$
7.     end if
8.   end for
9. until no mistakes,  $y_i(w \cdot x_i + b) \leq 0$ , are made within the for loop
10. return  $(w, b)$

For convenience, we will in these chapters discuss the version of the algorithm that, given a set of points (constraints, or examples)  $A = \cup_i a_i$  from  $\mathbb{R}^n$  finds, if any, a normal  $z$  to a hyperplane through the origin such that  $z \cdot a_i > 0$  for every  $i$ . (Further on we will let  $A$  also denote the matrix with  $a_i$  as its row  $i$ . Note that we do not have any zero rows in the matrix  $A$  since the inequality is strict). This is, so to say, a hyperplane through origin such that all points are on the very same side of the hyperplane. Here, the constraints are all considered to be in the same class, while we on the other hand have added an extra condition on the hyperplane, namely that it passes through the origin. Also, the learning rate  $\eta$  is set to 1 for convenience.

This version of the algorithm can easily be described as follows:

## 7.2. REDUCTION OF A FEASIBLE SOLUTION TO AN LP TO A HYPERPLANE THROUGH ORIGIN WITH POSITIVE EXAMPLES

---

**Algorithm 7.1.2.** *The Perceptron Algorithm*

*Input:* A set of points (constraints)  $A = \cup_i a_i$  from  $\mathbb{R}^n$ .

*Output:* A normal  $z$  to a hyperplane such that  $z \cdot a_i > 0$  for every  $i$ , if there is such a solution.

1. Let  $z=0$ .
2. If there is a point  $a_i \in A$  such that  $z \cdot a_i \leq 0$ , then  $z \leftarrow z + a_i$ .
3. Repeat step 2 until no such points are found and output  $z$ .

On top of this, we will normalise the conditions such that all of them have the same length ( $\|a_i\| = 1$  for simplicity reasons). It is easy to see that if  $z$  has positive inner product with all  $a_i$ 's, it would also have positive inner product with all  $a_i/\|a_i\|$  since the denominator is positive for all constraints.

Do note that the perceptron algorithm will never halt if there is no solution to the stated problem. In this case there is always at least one constraint misclassified, giving that we always have to return to step 2.

### 7.1.2 The Notion of Margin

There is a need to have a way of comparing two different solutions to our studied problem, in order to be able to choose the best one. To do this we will use the concept of margin. In this case we let  $z$  be a solution to a linear program such that  $z \cdot a_i > 0$  for every  $a_i \in A$  and let  $P$  be the hyperplane through origin with  $z$  as normal.

**Definition 7.1.1.** The margin  $\gamma(z, A)$  of a specific solution  $z$  to a linear program is the minimum distance between members of  $A$  and the hyperplane  $P$ .

But for every posed problem about a linear program we are interested in getting as large margin as possible, thus we introduce the concept of the maximal margin.

**Definition 7.1.2.** The maximal margin  $\gamma$  to a linear program is the largest possible margin that can be attained by letting  $z$  vary over all possible solutions.

## 7.2 Reduction of a Feasible Solution to an LP to a Hyperplane Through Origin with Positive Examples

In the general case a solution to a Linear Program is a hyperplane that separates positive and negative examples. But in this thesis we will, as mentioned above, transform the problem and instead study linear programs with only positive examples, moreover will the transformation guarantee that there is a solution through the origin. Here is a short description of how the transformation is made.

1. Let us start with a linear program with both positive and negative examples and a separating  $(n - 1)$ -dimensional hyperplane  $P$  as a solution to the problem, that is:  $P$  separates the negative examples from the positive ones.

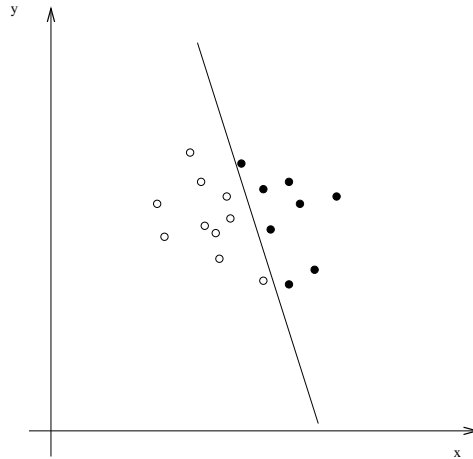


Figure 7.1: Hyperplane that separates positive and negative examples.

2. For all  $m$  constraints: add a new coordinate, always with the same nonzero value (here chosen to be equal to one), to every constraint  $a$ , or formally

$$a \leftarrow \{a_1, \dots, a_n, 1\}.$$

Now we have  $m$  constraints in  $n + 1$  dimensions. Since the original problem had an  $n - 1$ -dimensional hyperplane  $P$  as a solution, we get that the  $n$ -dimensional hyperplane  $P'$ , which is spanned by  $P$  and origin, is a solution to the new problem.

3. Now we construct a new set  $A$  of examples by negating all of the negative examples in the previous set. So, if  $a_i$  is a negative example, we make the replacement  $a_i \leftarrow -a_i$ . Now all examples are positive and we know that  $z \cdot a_i > 0$  for every  $i$ , when  $z$  is a feasible solution to our studied problem.

### 7.3. A CLASSICAL THEOREM CONCERNING THE PERCEPTRON ALGORITHM

---

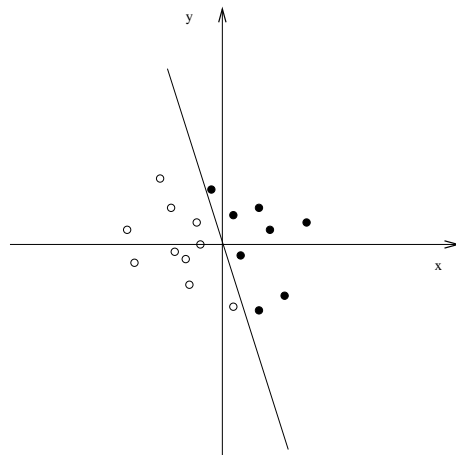


Figure 7.2: Hyperplane through origin that separates positive and negative examples.

4. Finally we normalise the length of all constraints such that they all have length one. This operation does not interfere with our separating hyperplane since  $z \cdot a_i > 0$  implies that  $(z \cdot a_i) / \|a_i\| > 0$ .
5. This yields a hyperplane through the origin such that the inner product between a normal to the hyperplane and all examples from  $A$  is strictly positive.

Note that this way of restating the original problem only preserves a feasible solution. An optimal solution, yielding a maximal margin, might not be preserved during this transformation of the problem.

### 7.3 A Classical Theorem Concerning the Perceptron Algorithm

Obviously this algorithm will never halt if there is no solution to the problem, that would imply that there always exists a misclassified example yielding a new update. On the other hand, Novikoff proved the following: If there is a solution to the problem, the algorithm will halt in a finite number of steps, even if the number of constraints is infinite. We will now present this classical theorem, but rewritten for the case where we have no bias,  $b = 0$ , and only positive examples of unit length.

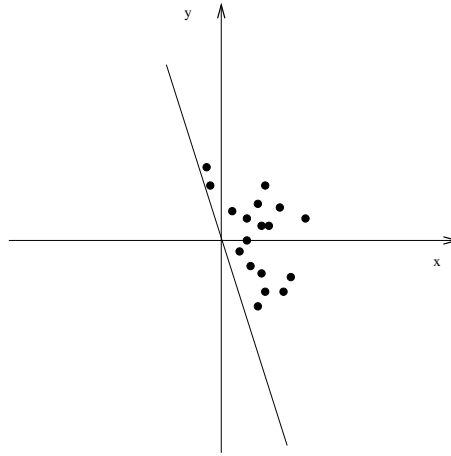


Figure 7.3: Hyperplane through origin having all examples at the same side.

**Theorem 7.3.1.** *The number of mistakes made by the on-line perceptron algorithm, on a non-trivial set  $A$  that has a solution to the problem, is at most  $(1/\gamma)^2$ , where  $\gamma$  is the size of the maximal margin according to the above definition.*

Now, since the margin can be any positive number close to zero, this upper bound of the performance does not say very much. And even though there are many results showing that the algorithm runs much faster in the usual case, there are constructions of constraint sets such that the behaviour of the algorithm is exponential in terms of the number of constraints [6].

Nevertheless, the way the theorem is proved is of interest for further developments of the Perceptron Algorithm later on in this thesis, and therefore we present a proof of the theorem.

*Proof:* We consider the simpler form of the algorithm where the desired hyperplane passes through origin, thus there is no bias. We further assume that all examples are of unit length and also positive so we do not have to consider the sign of any constraint.

Now, let  $z$  denote a solution where  $z \cdot a_i \geq \gamma$  for every  $i$ , and let  $x_t$  denote the hypothesis  $x$  in the algorithm after  $t$  updates. First of all we conclude that

$$x_t \cdot z = x_{t-1} \cdot z + a_i \cdot z \geq x_{t-1} \cdot z + \gamma$$

for the  $a_i$  that was the last instance of misclassification yielding an update. This implies, by induction, that  $x_t \cdot z \geq t\gamma$ .

Similarly, we have

$$\|x_t\|^2 = \|x_{t-1}\|^2 + 2(x_{t-1} \cdot a_i) + \|a_i\|^2 \leq \|x_{t-1}\|^2 + 1,$$

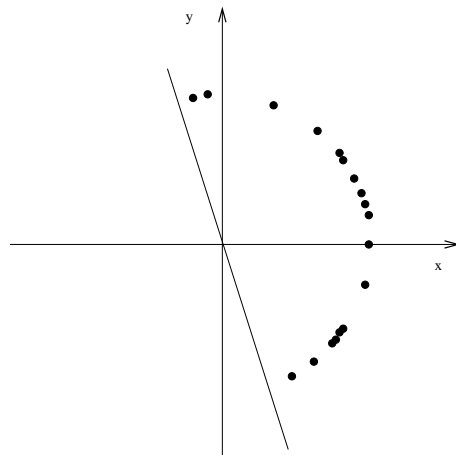


Figure 7.4: A hyperplane through origin having all examples, now of unit length, at the same side. This illustrates the transformation of the linear program that is used throughout the thesis.

since  $(x_{t-1} \cdot a_i)$  is negative, which implies that

$$\|x_t\|^2 \leq t.$$

Combining these two results we get the following squeezing relation:

$$\|z\|\sqrt{t} \geq \|z\|\|x_t\| \geq (x_t) \cdot z \geq t\gamma,$$

which implies that  $t \leq 1/\gamma^2$ . □

## 7.4 The Classical Problem Concerning the XOR-function

Although the perceptron initially seemed promising, it was quickly proved that a single perceptron could not be trained to recognise many classes of patterns. This led to the field of neural network research stagnating for many years, before it was recognised that a feed-forward neural network with three or more layers (also called a multilayer perceptron) had far greater processing power than perceptrons with one layer (also called a single layer perceptron) or two. Single layer perceptrons are only capable of learning linearly separable patterns. In 1969 Marvin Minsky and Seymour showed in a famous monograph entitled *Perceptrons*, [52] that it was impossible for these classes of network to learn an XOR-function. They conjectured (incorrectly) that a similar result would hold

for a perceptron with three or more layers. The discovery in the 1980s that multi-layer neural networks did not, in fact, have these problems contributed to the resurgence of neural network research. More recently, interest in the perceptron learning algorithm has increased again after Freund and Schapire (1998) presented a voted formulation of the original algorithm (attaining large margin) and suggested that one can apply the kernel trick to it. The kernel-perceptron not only handles nonlinearly separable data but can also go beyond vectors and classify instances having a relational representation (e.g. trees, graphs or sequences).

## 7.5 Neural Networks

Neural networks have great advantages of adaptability, flexibility, and universal nonlinear functional approximators. One of the most popular algorithms in the neural networks category is the back-propagation algorithm. The back-propagation algorithm performs supervised learning on a multilayer feed-forward neural network. It is based on Perceptron algorithm introduced by Rosenblatt that is considered as one of the first machine learning algorithms. The perceptron algorithm expresses linear decision surfaces. In contrast, the kind of multilayer networks learned by the back-propagation algorithm are capable of expressing a rich variety of nonlinear decision surfaces. Its discovery made the neural networks a tool for solving a wide variety of problems ranging from speech recognition to complex tasks of particle separation in high energy physics experiments. A multilayer neural network consists of sensory units (neurons or nodes) divided into 3 layers - an input layer, one or more hidden layers and an output layer. The units in each layer are fully connected with the units in the next layer. These connections have weights associated with them. Each signal travelling along the link is multiplied by the connection weight. The input signal propagates through the network in a forward direction, on a layer-by-layer basis. The training of a network by back-propagation involves three stages: The feed-forward of the input training pattern, the calculation and back-propagation of the associated error, and the adjustment of the weights. After training, application of the net involves only the computations of the feed-forward phase. Even if the training is slow, a trained net can produce its output very rapidly. An activation function for a back-propagation net should have several important characteristics: It should be continuously differentiable, and monotonically non-decreasing. Furthermore, for computational efficiency, it is desirable that its derivative is easy to compute. For the most commonly used activation functions, the value of the derivative can be expressed in terms of the value of the function. Usually the function is expected to saturate, i.e. approach finite maximum and minimum values asymptotically.

## 7.6 Linear Programs

The problem solved by the version of the Perceptron Algorithm discussed here, is the homogenised form of a feasible standard form of a linear program: Find  $x$  such that  $Ax \geq 0$  and  $x \neq 0$ . Here,  $A$  is the matrix with  $a_i$  as row  $i$  and  $x = z^T$ , where  $z$  is the normal of the hyperplane described above. This problem is of great importance, since it solves  $\max c^T x, Ax \leq b, x \geq 0$  by repeatedly solve the homogenised version and perform a binary search.

Due to the importance of the problem, many different methods have evolved to find solutions to it. To mention here is the simplex algorithm, the interior point method, ellipsoid methods, the perceptron algorithm and the modified perceptron algorithm.

One of the main results of this part of the thesis, is that we present a modified perceptron algorithm that is both deterministic and faster than previous versions. This makes the modified perceptron algorithm even more competitive than before.

### 7.6.1 Computational Complexity

Popular algorithms for solving standard LP are versions of the simplex algorithm [66]; ellipsoid methods [45]; interior point algorithms [65]; and the perceptron-rescaling algorithm (with possible modifications) [35]. There exist different measures of the complexity of an algorithm. We will mention three different of them here: *worst case*, which is given by the worst probable running time over all datasets, *expected complexity*, described as the expected time used by the algorithm when a problem is picked from a uniform distribution, and *smoothed complexity*, which is defined as the maximum over all inputs of the expected running time over an input under a Gaussian perturbation with variance  $\sigma^2$ . The following figures can be found in the literature:

- Simplex algorithm: exponential in worst case, but polynomial smoothed complexity [66] (see the same article for a definition of smoothed analysis).
- Interior point methods:  $O(m^3 \log(m/\sigma))$  (smoothed complexity) [65], where  $\sigma$  is defined above.
- Ellipsoid methods:  $O(mn^3 \langle A, b \rangle)$  in worst case, where  $\langle A, b \rangle \geq mn$ , [45]. However, this method does not work on-line, which can be a disadvantage for very large datasets.
- Perceptron algorithm with rescaling [19]:  $O(mn^4 \log n \log(1/\gamma^*))$  where

$$\gamma^* = \max_{x \in \mathbb{R}^n} \min_i \frac{a_i \cdot x}{\|a_i\| \|x\|}.$$

(Here  $a_i$  denotes the  $i$ :th row of  $A$ .)

This algorithm is not deterministic, but terminates in worst case, with high probability, in the given time.

- Modified perceptron algorithm, presented in this thesis, terminates in worst case in time:  $O(mn^3 \log n \log(1/\gamma^*))$ .
- Modified perceptron algorithm implemented on parallel processors, presented in this thesis, terminates in worst case in:  $O(mn^2 \log n \log(1/\gamma^*))$ .
- Modified perceptron algorithm, presented in this thesis, terminates in expected time:  $O(mn^2 \log n \log(1/\gamma^*))$ .

### 7.6.2 The Case of Semidefinite Programming

Semidefinite programming is an active research area in optimisation and it has important applications in combinatorial optimisation and control theory. The main discussion in this subsection is taken from [42] and is included to the thesis to show one important consequence of speeding up the modified perceptron algorithm.

Semidefinite programming has been discovered as a useful tool in machine learning with applications in pattern separation, ellipsoids and invariant learning.

Methods for solving semidefinite programs has in many cases been developed in analogy to linear programming. Algorithms, with the main idea taken from the simplex method, has been developed for semidefinite programs, but they seem to be of mere theoretical interest. The ellipsoid method works by searching for a feasible point by repeatedly halving an ellipsoid and creating a new smaller ellipsoid in each step, in such manner that, after repeating the process enough many times, the centre of the last ellipsoid becomes a feasible point. This method usually attains its worst-case bound and can be said to have poor performance in practice. The currently most efficient method for solving a semidefinite program in practise, are different interior point methods. These methods minimise a linear function on convex sets provided the sets are endowed with self-concordant barrier functions.

Quite surprising was that Graepel, Herbrich, Kharechko and Shawe-Taylor [42] could show that the old perceptron algorithm could be modified to solve semidefinite programs. This was combined with the modified perceptron algorithm by Dunagan and Vempala to reach a polynomial time algorithm using the perceptron algorithm to solve this problem. The running time was not at a great speed, but the reason to mention this result here is that speeding up the modified perceptron algorithm, as done in this thesis, would speed up the algorithm for solving semidefinite programs using this method as well.

Their aim was to solve a semidefinite program, here expressed by linear matrix inequalities:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \mathbf{c}'\mathbf{x} \quad \text{subject to} \quad \mathbf{F}(\mathbf{x}) := \mathbf{F}_0 + \sum_{i=1}^n \mathbf{x}_i \mathbf{F}_i \succeq \mathbf{0},$$

where  $\mathbf{c} \in \mathbb{R}^n$  and  $\mathbf{F}_i \in \mathbb{R}^{m \times m}$  for all  $i \in \{0, \dots, n\}$ .

The main ideas in their work can be described firstly by the following three propositions:

**Proposition 7.6.1.** *Every semidefinite program is a linear program with infinitely many linear constraints.*

**Proposition 7.6.2.** *Any semidefinite program can be solved by a sequence of homogenised semidefinite constraint satisfaction problems of the following form:*

$$\text{find } \mathbf{x} \in \mathbb{R}^{n+1} \quad \text{subject to} \quad \mathbf{G}(\mathbf{x}) := \sum_{i=0}^n x_i \mathbf{G}_i \succ \mathbf{0}.$$

**Proposition 7.6.3.** *If the feasible set has a positive radius, then the perceptron learning algorithm solves a linear constraint satisfaction problem in finitely many steps.*

This fact, that the ordinary perceptron algorithm ends in a finite number of steps, if there is a solution with positive radius, irrespective of the cardinality of the number of constraints, is an old but very useful result when it comes to study the perceptron algorithm.

Secondly the authors combine these facts to construct an algorithm to solve the semidefinite program. In order to make the algorithm terminate in polynomial time, they use the modified perceptron algorithm, first constructed by Dunagan and Vempala. So, speeding up the modified perceptron algorithm, as done below, will in turn speed up this algorithm for solving semidefinite programs.



## Chapter 8

# Results on the Perceptron Algorithm

From now on we will discuss changes and new results concerning the modified perceptron algorithm in order to make it both deterministic and faster than before. We will improve the performance with a factor of  $O(n)$ , if looking at the worst case scenario, but also show that the expected time used by the algorithm will be a factor  $O(n^2)$  faster than the bound made by Dunagan and Vempala. The improvements of the algorithm has also changed it from a randomised one into a deterministic algorithm. The algorithm presented by Dunagan and Vempala will be presented in detail later on in the thesis, after we have discussed some geometrical aspects of the unit sphere in  $n$  dimensions.

### 8.1 Geometry Questions for the Modified Perceptron Algorithm

In the proof of the polynomial termination of the modified perceptron algorithm, Dunagan and Vempala use an estimate for the probability of two random unit vectors having an inner product with each other of at least  $1/\sqrt{n}$ . They say that a simple geometric argument yields that this entity, here denoted  $\Pi_n$  is at least  $1/8$  for all  $n$ . Here follows a more extensive investigation of this probability for two reasons. Firstly, it is an interesting property when you are about to speed up an algorithm to have as good estimates as possible. Secondly, we could never find this simple geometric argument giving their estimate, so we had to ensure us, in one way or the other, that the estimate was correct.

#### 8.1.1 Inner Product Between Two Random Unit Vectors

In the algorithm described by Dunagan and Vempala, the key to prove the performance is to find a unit vector  $v$  having inner product with the optimal, but unknown, unit vector  $w$  of at least  $1/\sqrt{n}$ , with a high probability. The value  $1/\sqrt{n}$  originates from a proof made in [35]. Here it is of great importance to, at first, know the probability of two unit vectors, picked from a uniform distribution, having an inner product of at least  $1/\sqrt{n}$ .

First we conclude that we can fix one of the two  $n$ -dimensional vectors, say  $w = e_1 = (1, 0, 0, \dots, 0)$  which is a vector on the unit sphere. Now, let  $v = (x_1, \dots, x_n)$  be a randomly chosen unit vector in our  $n$ -dimensional space, chosen from a uniform distribution on the  $n$ -dimensional sphere.

In simulations where we pick a random vector of length one, with uniform distribution on the unit ball surface, we first construct  $\tilde{v} = (\tilde{x}_1, \dots, \tilde{x}_n)$ , where each  $\tilde{x}_i$  is chosen with the same Gaussian distribution, in our case  $\tilde{x} \in N(0, 1)$ . Our vector  $v$  is a

scaling of  $\tilde{v}$  such that it has length one, see [54] for more details on the construction of a uniform distribution on the  $n$ -dimensional sphere. The above construction will also be used in the calculations below.

Let  $\Pi_n$  denote the probability that two  $n$ -dimensional unit vectors have an inner product of at least  $\frac{1}{\sqrt{n}}$ . We calculate the probability of having the desired inner product and assume that  $n \geq 2$  since the case  $n = 1$  is trivial.

$$\begin{aligned} \Pr\left(v \cdot w \geq \frac{1}{\sqrt{n}}\right) &= \Pr\left(x_1 \geq \frac{1}{\sqrt{n}}\right) = \frac{1}{2}\Pr\left(x_1^2 \geq \frac{1}{n}\right) = \frac{1}{2}\Pr\left(\frac{\tilde{x}_1^2}{\sum_{i=1}^n \tilde{x}_i^2} \geq \frac{1}{n}\right) = \\ &= \frac{1}{2}\Pr\left(\tilde{x}_1^2 \geq \frac{\sum_{i=1}^n \tilde{x}_i^2}{n}\right) = \frac{1}{2}\Pr\left(\tilde{x}_1^2\left(1 - \frac{1}{n}\right) \geq \frac{\sum_{i=2}^n \tilde{x}_i^2}{n}\right) = \frac{1}{2}\Pr\left(\tilde{x}_1^2 \geq \frac{\sum_{i=2}^n \tilde{x}_i^2}{n-1}\right). \end{aligned}$$

This can be used for calculating the probability  $\Pi_n$  when  $n$  tends to infinity. We get the following proposition, where  $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ :

**Proposition 8.1.1.**

$$\lim_{n \rightarrow \infty} \Pi_n = \frac{1}{2} \left(1 - \operatorname{erf}\left(\frac{1}{\sqrt{2}}\right)\right).$$

*Proof:* When  $n$  tends to infinity we get that the distribution

$$\frac{\sum_{i=2}^n \tilde{x}_i^2}{n-1}$$

tends to 1, having zero variance, and so,

$$\lim_{n \rightarrow \infty} \Pi_n = \frac{1}{2} \Pr\left(\tilde{x}_1^2 \geq 1\right).$$

Then  $\tilde{x}_1^2$  is  $\chi^2(1)$ -distributed, hence

$$\lim_{n \rightarrow \infty} \Pi_n = \frac{1}{2} \int_1^\infty \frac{1}{\sqrt{2\pi}} \frac{e^{-\frac{x}{2}}}{\sqrt{x}} dx = \dots = \frac{1}{2} \left(1 - \operatorname{erf}\left(\frac{1}{\sqrt{2}}\right)\right).$$

□

Moreover, there is an other, more direct, way of calculating this probability, which also gives us an analytical expression for every  $n$ . By elementary geometry, we get that the probability can be expressed by considering sectors of the unit sphere in  $\mathbb{R}^n$ . The surface element  $dS$  on this hyper-sphere is given by

$$dS = C_{n-1} \sqrt{(1-x^2)^{n-3}} dx,$$

where  $C_{n-1}$  is the area of the  $(n-1)$ -dimensional unit sphere. We get the following theorem:

8.2. LOWER BOUND ON THE PROBABILITY OF  
TWO VECTORS HAVING LARGE INNER PRODUCT

---

**Theorem 8.1.1.**

$$\Pi_n = \Pr\left(v \cdot w \geq \frac{1}{\sqrt{n}}\right) = \frac{\int_{1/\sqrt{n}}^1 \sqrt{(1-x^2)^{n-3}} dx}{\int_{-1}^1 \sqrt{(1-x^2)^{n-3}} dx},$$

when  $n > 2$ . Both integrals can be calculated by using the substitution  $x = \cos t$ .

We list the first known exact values and present their approximations for a few finite cases. A proof showing that  $\Pi_n$  always is greater than  $\frac{1}{2}(1 - \text{erf}(1/\sqrt{2}))$  will be included in a section below.

$n$	$\Pi_n$	appr. value
1	1/2	0.5000 ...
2	1/4	0.2500 ...
3	$\frac{1}{2}(1 - 1/(\sqrt{3}))$	0.2113 ...
4	$1/3 - \sqrt{3}/(4\pi)$	0.1955 ...
5	$\frac{1}{2}(1 - 7/(5\sqrt{5}))$	0.1870 ...
$\vdots$	$\vdots$	$\vdots$
$\infty$	$\frac{1}{2}(1 - \text{erf}(1/\sqrt{2}))$	0.1587 ...

The first version of Dunagan and Vempalas algorithm that we read used an incorrect estimate of the entity  $\Pi_n$ , saying that  $\Pi_n \geq 1/4$ . Here we see that the first version of the proof of Algorithm 1.1 in [35] will fail for  $n \geq 3$ .

(The proof was corrected with a new lower bound,  $\Pi \geq 1/8$ , directly after the mistake was discovered. But a positive effect of this minor mistake was that it triggered us to a detailed investigation of this geometric property.)

The failure of the first proof was the case since they needed a lower bound of the probability of at least 1/4 in the estimations on the increase of the margin made in the proof. Below we will give the exact lower bound, better than the rude estimate 1/8, for the probability which in turn could be used to estimate optimal choices of different pre-chosen constants in the algorithm.

## 8.2 Lower Bound on the Probability of Two Vectors Having Large Inner Product

For purposes below, we need to show a lower bound for  $\Pi_n$  for every  $n$ . This will be proved by first showing that the sequences  $\Pi_{2n}$  and  $\Pi_{2n+1}$  are decreasing sequences and then add the earlier information about the first few values of  $\Pi_n$  and the limit when  $n$  tends to infinity.

We were not able to show that the sequence  $\Pi_n$  itself is a decreasing sequence, but we conjecture that this is the case. Our results yields anyhow a lower bound, even though the sequence  $\Pi_n$  might be of a sawtooth shape for large  $n$ .

### 8.2.1 $\Pi_n$ is Always Greater than $\frac{1}{2}(1 - \operatorname{erf}(1/\sqrt{2}))$

In order to show that  $\Pi_n$  always is big enough for our purposes, we first show the following:

**Theorem 8.2.1.** *The series  $\Pi_{2n}$  and  $\Pi_{2n+1}$  are decreasing.*

*Proof:* First put

$$I_n = \int_{\frac{1}{\sqrt{n}}}^1 \sqrt{(1-x^2)^{n-3}} dx \text{ and } J_n = \int_{-1}^1 \sqrt{(1-x^2)^{n-3}} dx.$$

According to the definition of  $\Pi_n$  above, we want to show that  $\frac{I_n}{J_n} \geq \frac{I_{n+2}}{J_{n+2}}$  when  $n > 1$ . Partial integration yields:

$$\begin{aligned} J_{n+2} &= \left[ x\sqrt{(1-x^2)^{n-1}} \right]_{-1}^1 - \int_{-1}^1 x \frac{n-1}{2} (1-x^2)^{\frac{n-3}{2}} (-2x) dx = \\ &= -(n-1)J_{n+2} + (n-1)J_n. \end{aligned}$$

We get that

$$J_{n+2} = \frac{n-1}{n} J_n.$$

In the same manner we get that:

$$\begin{aligned} I_{n+2} &= \left[ x\sqrt{(1-x^2)^{n-1}} \right]_{\frac{1}{\sqrt{n+2}}}^1 + (n-1) \int_{\frac{1}{\sqrt{n+2}}}^1 x(1-x^2)^{\frac{n-3}{2}} x dx = \\ &= -\frac{1}{\sqrt{n+2}} \sqrt{\left(1 - \frac{1}{n+2}\right)^{n-1}} - (n-1) \int_{\frac{1}{\sqrt{n+2}}}^1 (1-x^2)^{\frac{n-3}{2}} (1-x^2) dx + \\ &\quad + (n-1) \int_{\frac{1}{\sqrt{n+2}}}^1 (1-x^2)^{\frac{n-3}{2}} dx = \\ &= -\frac{1}{\sqrt{n+2}} \sqrt{\left(1 - \frac{1}{n+2}\right)^{n-1}} - (n-1)I_{n+2} + (n-1)I_n + \end{aligned}$$

8.2. LOWER BOUND ON THE PROBABILITY OF  
TWO VECTORS HAVING LARGE INNER PRODUCT

---

$$+(n-1) \int_{\frac{1}{\sqrt{n+2}}}^{\frac{1}{\sqrt{n}}} (1-x^2)^{\frac{n-3}{2}} dx.$$

Consequently,

$$I_{n+2} = \frac{n-1}{n} I_n + \frac{n-1}{n} \left( \int_{\frac{1}{\sqrt{n+2}}}^{\frac{1}{\sqrt{n}}} \sqrt{(1-x^2)^{n-3}} dx - \frac{1}{(n-1)\sqrt{n+2}} \sqrt{1 - \left(\frac{1}{n+2}\right)^{n-1}} \right).$$

We are finished if we can show that

$$\int_{\frac{1}{\sqrt{n+2}}}^{\frac{1}{\sqrt{n}}} \sqrt{(1-x^2)^{n-3}} dx \leq \frac{1}{(n-1)\sqrt{n+2}} \sqrt{1 - \left(\frac{1}{n+2}\right)^{n-1}}.$$

First we estimate the integral

$$\begin{aligned} \int_{\frac{1}{\sqrt{n+2}}}^{\frac{1}{\sqrt{n}}} \sqrt{(1-x^2)^{n-3}} dx &\leq \int_{\frac{1}{\sqrt{n+2}}}^{\frac{1}{\sqrt{n}}} x \sqrt{n+2} \sqrt{(1-x^2)^{n-3}} dx \leq \\ &\leq \left[ -\sqrt{n+2} \frac{1}{n-1} (1-x^2)^{\frac{n-1}{2}} \right]_{\frac{1}{\sqrt{n+2}}}^{\frac{1}{\sqrt{n}}} = \\ &= \sqrt{n+2} \frac{1}{n-1} \sqrt{\left(1 - \frac{1}{n+2}\right)^{n-1}} - \sqrt{n+2} \frac{1}{n-1} \sqrt{\left(1 - \frac{1}{n}\right)^{n-1}}. \end{aligned}$$

Now, we are ready if we can show that

$$\begin{aligned} \sqrt{n+2} \frac{1}{n-1} \sqrt{\left(1 - \frac{1}{n+2}\right)^{n-1}} - \sqrt{n+2} \frac{1}{n-1} \sqrt{\left(1 - \frac{1}{n}\right)^{n-1}} &\leq \\ &\leq \frac{1}{(n-1)\sqrt{n+2}} \sqrt{\left(1 - \frac{1}{n+2}\right)^{n-1}}. \end{aligned}$$

This inequality can be rewritten in a few steps as follows:

$$\begin{aligned} (n+1) \sqrt{\left(1 - \frac{1}{n+2}\right)^{n-1}} &\leq (n+2) \sqrt{\left(1 - \frac{1}{n}\right)^{n-1}}, \\ \left(1 - \frac{1}{n+2}\right) \sqrt{\left(1 - \frac{1}{n+2}\right)^{n-1}} &\leq \sqrt{\left(1 - \frac{1}{n}\right)^{n-1}}, \end{aligned}$$

$$\left(1 - \frac{1}{n+2}\right)^{n+1} \leq \left(1 - \frac{1}{n}\right)^{n-1}$$

and by inverting both sides we get that

$$\left(1 + \frac{1}{n-1}\right)^{n-1} \leq \left(1 + \frac{1}{n+1}\right)^{n+1},$$

which is true since  $\left(1 + \frac{1}{n}\right)^n$ ,  $n = 1, 2, 3, \dots$  is a monotone increasing sequence.  $\square$

Summing up facts concerning  $\Pi_n$ , we now know the first few values, the value when  $n$  tends to infinity and that  $\Pi_{2n}$  and  $\Pi_{2n+1}$  are decreasing sequences when  $n$  grows. We get the following corollary:

**Corollary 2.4** *For every  $n$ ,*

$$\Pi_n \geq \frac{1}{2}(1 - \operatorname{erf}(1/\sqrt{2})).$$

### 8.3 How to Make the Perceptron Algorithm Polynomial

The first polynomial algorithm using the perceptron algorithm was created by Dunagan and Vempala in [35]. Even though the algorithm they constructed was not as fast as other algorithms mentioned above, it must be said it was a breakthrough for the Perceptron Algorithm. After this result, the algorithm could not be counted out, it could be competitive.

**Algorithm 8.3.1.** *The Modified Perceptron Algorithm (Dunagan & Vempala)*

*Input:* An  $m \times n$  matrix  $A$ .

*Output:* A point  $x$  such that  $Ax \geq 0$  and  $x \neq 0$ .

1. Let  $B = I$ ,  $\sigma = 1/(32n)$ .
2. (Perceptron)
  - (a) Let  $x$  be the origin in  $\mathbb{R}^n$ .
  - (b) Repeat at most  $16n^2$  times: If there exists a row  $a$  such that  $a \cdot x \leq 0$ , set  $x = x + \bar{a}$ . Here  $\bar{a}$  denotes the normalised vector  $a/\|a\|$ .
3. If  $Ax \geq 0$ , then output  $Bx$  as a feasible solution and stop.
4. (Perceptron Improvement)
  - (a) Let  $x$  be a random unit vector in  $\mathbb{R}^n$ .

### 8.3. HOW TO MAKE THE PERCEPTRON ALGORITHM POLYNOMIAL

---

(b) Repeat at most  $(\ln n)/\sigma^2$  times: If there exists a row  $a$  such that  $\bar{a} \cdot \bar{x} < -\sigma$ , set  $x \leftarrow x - (\bar{a} \cdot x)\bar{a}$ . If  $x = 0$ , go back to step (a). (This is to assure us of not having the vector  $x$  set to zero)

(c) If there still exists a row  $a$  such that  $\bar{a} \cdot \bar{x} < -\sigma$ , restart at step (a). (This takes care of the situation of a bad choice of the randomly chosen vector  $x$ )

5. If  $Ax \geq 0$ , then output  $Bx$  as a feasible solution and stop.

6. (Rescaling)

Set  $A \leftarrow A(I + \bar{x}\bar{x}^T)$  and  $B \leftarrow B(I + \bar{x}\bar{x}^T)$ .

7. Go back to step 2.

It is not evident that this algorithm will terminate. But Dunagan and Vempala proved that the margin of the Linear Program will increase in mean, when many rescalings are made inside the algorithm. This will in turn make the margin so large such that the Perceptron part (2) of the algorithm will return a feasible solution to the problem of the Linear Program.

Beside the size of the running time ( $O(mn^4 \log n \log(1/\rho))$ , where  $\rho$  is approximately the size of margin  $\gamma$ ), the algorithm they presented was randomised, as seen in step 4 (a) above. This means that the result was given in the mentioned time with very high probability. This is unfortunate, since exceeding the specified time will not always imply that no solution exists. This property will be desired later on in the thesis when we are about to estimate the maximal margin.

Dunagan and Vempala posed an open question whether or not there was a deterministic version of their algorithm.

#### 8.3.1 Making a Deterministic and Polynomial Perceptron Algorithm

Here, we answer the above stated question in the affirmative, according to a result by Barr and Wigelius presented in [16]. The drawback of the first deterministic and polynomial perceptron algorithm was that it ran a factor  $2n$  slower than the algorithm presented by Dunagan and Vempala. On the other hand it gave us a freedom to alter different constants in the algorithm such that it could run even faster than before. This will be dealt with later on.

A positive consequence was that the deterministic algorithm could present a definitive answer if the stated linear program was solvable or not. A fact that will make us construct an algorithm that approximates a solution to the stated linear program with maximal margin.

Also, the construction of the algorithm suggests a natural way to implement the problem on parallel processors in order to speed up the process even further.

### 8.3.2 How to Make it Deterministic

First of all we can conclude that it is due to the random choice of a unit vector inside the algorithm that makes the algorithm of Dunagan and Vempala a randomised one. The question to put is if there is a way of choosing appropriate vectors so that we can keep control of the number of iterations being made inside the algorithm.

The success of the algorithm depends on choosing a unit vector that has an inner product of at least  $1/\sqrt{n}$  with a feasible solution  $z$  to the posed problem. But since we do not know a solution to the problem, we can ask us if we can have a set  $V$  of vectors, where at least one vector  $v \in V$  has the mentioned property. The answer to this is positive. The set  $V = \cup_{i=1}^n \{e_i, -e_i\}$  where  $\cup_{i=1}^n \{e_i\}$  constitutes an orthonormal basis for  $\mathbb{R}^n$  will do, according to the following proposition.

**Proposition 8.3.1.** *Let  $V = \cup_{i=1}^n \{e_i, -e_i\}$ , where  $\cup_{i=1}^n \{e_i\}$  constitutes an orthonormal basis for  $\mathbb{R}^n$ . Now, for every unit vector  $w \in \mathbb{R}^n$ ,*

$$\max_{v \in V} w \cdot v \geq \frac{1}{\sqrt{n}}.$$

*Proof:* First assume that  $e_i$  is the vector with a 1 in the  $i$ :th coordinate and zero elsewhere. Now let  $w = (w_1, \dots, w_n)$ . At least one  $w_i$  must have an absolute value of at least  $1/\sqrt{n}$ , otherwise  $\|w\| < 1$ . This yields that there exists at least one vector  $v \in V$  such that  $w \cdot v \geq 1/\sqrt{n}$  as stated above. To generalise this statement for any orthonormal basis, we only have to consider the rotation symmetry of  $\mathbb{R}^n$ .  $\square$

To make the algorithm deterministic, we will now run the algorithm in  $2n$  parallel tracks. And instead of using the origin in step 2(a) and a random unit vector in step 4(a), we will use vectors from our set  $V$  and update them for each iteration in the algorithm. Since one of parallel tracks will come closer and closer to a solution for each round, this specific track will terminate in the time mentioned above. But, we are running  $2n$  tracks, and the total running time will be a factor  $2n$  greater than before.

In practice, this is an advantage since the algorithm will tell us how to make use of parallel processors in a practical situation, making the algorithm fast when implemented.

## 8.4 How to Speed Up the Perceptron Algorithm

In order to speed up the algorithm, a deep analysis of all estimates done by Dunagan and Vempala has been done. As a result of this  $\sigma$  can be enlarged to  $1/(32\sqrt{n})$  and the  $16n^2$  in step 2(b) can be reduced to  $4n$ . These alterations will speed up the process with a factor of order  $O(n)$  for one start vector. Also, new estimates in the margin growth causes another factor of order  $O(n)$  for one start vector. So, in total, the algorithm will run a factor  $2n$  slower to make it deterministic, but a factor  $O(n^2)$  faster due to new

estimates, yielding a modified perceptron algorithm running a factor  $O(n)$  faster than previous results made. These calculations are upper bounds on the worst time taken by the algorithm to produce a feasible solution.

### 8.4.1 The Deterministic Modified Perceptron Algorithm

Now, we are ready to go into details with the topic of this chapter.

Below we can study the general structure of the algorithm, breaking it up into smaller parts that will be presented further on. Important is though that we are running the algorithm in  $2n$  parallel tracks. This does not imply that we have to run the algorithm on parallel processors, only that we do each step for every single track before going to the next step.

**Algorithm 8.4.1.** *The Modified Perceptron Algorithm*

*Input:* An  $m \times n$  matrix  $A$ .

*Output:* A vector  $x$  such that  $Ax \geq 0$  and  $x \neq 0$  or "No solution exists".

1. *Initials*

Choose  $R = 4n$  (where  $n$  is the dimension of the Linear Program) and put  $\sigma = \frac{1}{32\sqrt{n}}$ .

Let  $B = I$  and  $V = \cup_{i=1}^n \{e_i, -e_i\}$ , where  $\cup_{i=1}^n \{e_i\}$  constitutes an orthonormal basis for  $\mathbb{R}^n$ .

2. *Wiggle Phase*

Let  $U = \emptyset$ . For each vector  $v \in V$ , run the Wiggle Algorithm, collecting all returned corresponding vectors  $u$  in  $U$ .

$V \leftarrow U$ .

3. *Check for no solution*

If  $V = \emptyset$ , then output "No solution exists" and stop.

4. *Rescaling Phase*

Normalise every vector in  $V$ . That is, for every  $v \in V$ , let  $v \leftarrow \frac{v}{\|v\|}$ .

For each vector  $v$  in  $V$  (at most  $2n$ ), together with its corresponding matrices  $A$  and  $B$ , run the Rescaling Algorithm.

5. *Perceptron Phase*

For each vector  $v \in V$ , run the Perceptron Algorithm for at most  $R$  rounds with  $v$  as the initial vector.

6. *If no feasible solution was obtained in the last iteration of the algorithm, go back to step 2.*

Note that it is only during the first time in the wiggle phase, the set  $V$  consists of an orthonormal basis. The important thing is that there will always be at least one vector  $v \in V$  having an inner product with  $z$  of at least  $1/\sqrt{n}$ , if there is a solution to the problem. The concept could be described as follows: sometimes we throw away a vector from the set  $V$ , but never throw away the good ones.

Also, we must remind the reader that, even though we start with the two matrices,  $A$  and  $B$ , we will do different updates to these matrices for every vector in  $V$ . So we will have up to  $2n$  parallel tracks of  $A$  and  $B$  matrices.

Note that there are more things to show than only to describe the smaller algorithms used inside the larger structure. We have to show that they work, to calculate the complexity and to prove that the main algorithm always will return a correct answer in the time mentioned above.

First we describe the Wiggle Algorithm:

**Algorithm 8.4.2.** *Wiggle Algorithm*

*Input:* An  $m \times n$  matrix  $A$ , a vector  $v \in \mathbb{R}^n$  and a set of vectors  $U$ .

*Output:* One of the following three: A solution  $x$  to  $Ax \geq 0$ , an updated vector  $v$  that will be put in  $U$  or the empty set  $\emptyset$  (also to be put in  $U$ ).

1. If  $\frac{a \cdot v}{\|a\| \|v\|} < -\sigma$  for some row  $a \in A$ ,  
then  $v \leftarrow v - \left( \frac{a}{\|a\|} \cdot v \right) \frac{a}{\|a\|}$ .
2. Repeat step 1 at most  $(\log n)/\sigma^2$  times.
3. If  $\frac{a \cdot v}{\|a\| \|v\|} \geq -\sigma$  for every row  $a \in A$ , then  $U \leftarrow U \cup v$ .
4. If  $Av \geq 0$ , then output  $Bv$  as a feasible solution and stop. (Here you choose the matrices  $A$  and  $B$  that corresponds to the specified vector  $v$ .)

This algorithm is shown, in [20], to output a vector  $v$  in at most  $(\ln n)/\sigma^2$  steps, if the input vector  $v$  satisfies  $v \cdot z \geq 1/\sqrt{n}$ , where  $z$  is a unit vector that solves  $Ax \geq 0$ . Thus, we have to insure us that at least one of our starting vectors in  $V$  does have this property. But this was shown in the earlier proposition presented above.

It is of interest for later calculations to see how this estimate is done, so we will here present a proof for the upper bound number of iterations in the  $\sigma$ -relaxed wiggle algorithm.

**Theorem 8.4.1.** *The number of updates made within the  $\sigma$ -relaxed wiggle algorithm is bounded above with  $t \leq (\ln n)/\sigma^2$ .*

*Proof:* Let  $z$  be a unit vector that correctly classifies all examples  $a_i$  in  $A$ . Suppose that an initial unit vector  $x_0$  satisfies  $x \cdot z \geq 1/\sqrt{n}$ . Denote the vector achieved in  $t$  updates

within the algorithm by  $x_t$ . Notice first that in each update made,  $x_t \cdot z$  does not decrease with increasing  $t$  because

$$x_t \cdot z = (x_{t-1} - (x_{t-1} \cdot a_i)a_i) \cdot z = x_{t-1} \cdot z - (x_{t-1} \cdot a_i)(z \cdot a_i) \geq x_{t-1} \cdot z$$

where the last inequality holds because  $x_{t-1}$  misclassifies  $a_i$ . On the other hand,  $|x_t|^2$  does decrease significantly because

$$\begin{aligned} |x_t|^2 &= |x_{t-1} - (x_{t-1} \cdot a_i)a_i|^2 = |x_{t-1}|^2 - 2(x_{t-1} \cdot a_i)^2 + (x_{t-1} \cdot a_i)^2 \leq \\ &\leq |x_{t-1}|^2(1 - \sigma^2) \end{aligned}$$

using the Pythagorean Theorem. Thus, after  $t$  iterations  $|x_t| \leq (1 - \sigma^2)^{\frac{t}{2}}$ . Since  $|x_t|$  cannot be less than  $|x \cdot z|$ , this means that the number of iterations  $t$  satisfies

$$(1 - \sigma^2)^{\frac{t}{2}} \geq \frac{1}{\sqrt{n}},$$

which implies that  $t \leq (\ln n)/\sigma^2$ . □

Also we must calculate the complexity of running through the Wiggle Algorithm once:

**Proposition 8.4.1.** *The number of iterations inside the Wiggle Algorithm is of order*

$$O(mn^2 \log n).$$

*Proof:* The inner loop of the algorithm requires at most one matrix-vector multiplication, time  $O(mn)$ , and a constant number of vector manipulations, time  $O(n)$ . This is repeated at most  $(\log n)/\sigma^2 = 32^2 n \log n$  times. So, the overall time bound is  $O(mn^2 \log n)$ . □

The following rescaling procedure is a simple matrix-matrix multiplication, being of order  $O(n^2)$ .

**Algorithm 8.4.3.** *Rescaling Phase*

*Input:* a vector  $v$  and its corresponding matrices  $A$  and  $B$ .

*Output:* rescaled matrices  $A$  and  $B$ .

1.  $A \leftarrow A(I + \frac{v}{\|v\|} \frac{v}{\|v\|}^T)$ .
2.  $B \leftarrow B(I + \frac{v}{\|v\|} \frac{v}{\|v\|}^T)$ .

The important thing to prove for this part, is that for at least one of our  $2n$  parallel processes, the matrices will be stretched in a direction such that the margin increases in the new problem  $Ax \geq 0$ . The proof of this follows substantially the proof in [35], but using some other in-data and new estimates. The reason for letting  $\rho \leq 1/(2\sqrt{n})$  in the following theorem is that if  $\rho$  would be larger, the algorithm will halt later on in the Perceptron phase.

**Theorem 8.4.2.** *Suppose  $\rho \leq 1/(2\sqrt{n})$ ,  $\sigma = 1/(32\sqrt{n})$  and  $n \geq 2$ . Let  $A'$  be obtained from  $A$  by one iteration of the algorithm (where the problem is not solved). Let  $\rho'$  and  $\rho$  be the margins of the problems  $A'x \geq 0$  and  $Ax \geq 0$  respectively. Also assume that we are studying one of those processes running parallel, where  $v \cdot z \geq 1/\sqrt{n}$  and  $z$  is a feasible solution, of length one, to  $Ax \geq 0$ . Then  $\rho' \geq (1 + \frac{1}{4})\rho$ .*

*Proof:* Let  $a_i$ ,  $i = 1, \dots, m$  be the rows of  $A$  at the beginning of some iteration, for one of the parallel processes having  $v \cdot z \geq 1/\sqrt{n}$ , where  $z$  is a feasible solution. Below we drop the index  $i$ , and usually denote a row  $a_i$  only with  $a$ ). Let  $z$  be the unit vector satisfying  $\rho = \min_i \frac{a}{\|a\|} \cdot z$ , and let  $\sigma_i = \frac{a}{\|a\|} \cdot v$ . After the wiggle phase, we get a vector  $v$  such that  $\frac{a}{\|a\|} \cdot v = \sigma_i \geq -\sigma$  for every  $i$ .

As described in the algorithm, let  $A'$  be the matrix obtained after the rescaling step, i.e.  $a'_i = a_i + (a_i \cdot v)v$ . Finally define  $z' = z + \alpha(z \cdot v)v$ , where

$$2\alpha + 1 = \rho\sqrt{n}$$

or to put it another way

$$\alpha = (\rho\sqrt{n} - 1)/2.$$

Even though  $z'$  might not be an optimal choice, it is enough to consider this one element to lower bound  $\rho'$ . We have  $\rho' \geq \min_j \frac{a'_j}{\|a'_j\|} \cdot \frac{z'}{\|z'\|}$ . Below we are about to get an estimate where we can drop the variable  $j$  and show that  $\rho'$  grows as stated in the theorem.

We will first prove that  $\frac{a'_i}{\|a'_i\|} \cdot z'$  cannot be too small. Note that

$$\frac{a'}{\|a'\|} \cdot z' = \frac{\frac{a}{\|a\|} + (\frac{a}{\|a\|} \cdot v)v}{\|\frac{a}{\|a\|} + (\frac{a}{\|a\|} \cdot v)v\|} \cdot z' = \frac{[\frac{a}{\|a\|} + (\frac{a}{\|a\|} \cdot v)v][z + \alpha(z \cdot v)v]}{\sqrt{1 + 3(\frac{a}{\|a\|} \cdot v)^2}}$$

by using the definitions of  $a'$  and  $z'$  respectively. Now, multiplying things together, and using that  $\frac{a}{\|a\|} \cdot z \geq \rho$ ,  $\frac{a}{\|a\|} \cdot v = \sigma_i$  and that  $v \cdot v = 1$  we get that the expression is at least

$$\frac{\rho + (2\alpha + 1)\sigma_i(z \cdot v)}{\sqrt{1 + 3\sigma_i^2}} = \rho \frac{1 + \sqrt{n}\sigma_i(z \cdot v)}{\sqrt{1 + 3\sigma_i^2}}$$

when substituting the chosen value of  $\alpha$ .

So, our aim now is to show that this expression is always greater than

$$\rho \frac{1 - \sigma}{\sqrt{1 + 3\sigma^2}},$$

no matter of what value we have on  $\sigma_i$ . First we start with the case of a positive  $\sigma_i$ .

In this case we would like to show that

$$\frac{1 + \sqrt{n}\sigma_i(z \cdot v)}{\sqrt{1 + 3\sigma_i^2}} \geq \frac{1 - \sigma}{\sqrt{1 + 3\sigma^2}} = C(\sigma) = C(1/32\sqrt{n}) = C$$

where we know that  $0 \leq \sigma_i \leq 1$  and  $1/\sqrt{n} \leq (z \cdot v) \leq 1$ . So, we want to show that

$$1 + \sqrt{n}\sigma_i(z \cdot v) \geq C\sqrt{1 + 3\sigma_i^2},$$

and since everything is positive we can square both sides yielding

$$1 + 2\sqrt{n}\sigma_i(z \cdot v) + n\sigma_i^2(z \cdot v)^2 \geq C^2(1 + 3\sigma_i^2).$$

Thus, we want to show that

$$f(\sigma_i, (z \cdot v)) = 1 + 2\sqrt{n}\sigma_i(z \cdot v) + n\sigma_i^2(z \cdot v)^2 - C^2(1 + 3\sigma_i^2)$$

is a positive function within the given limits of the variables  $\sigma_i$  and  $(z \cdot v)$ .

Examining the partial derivatives of the above function  $f$ , with respect to the two variables,  $\sigma_i$  and  $(z \cdot v)$ , we get two conditions,

$$2\sqrt{n}(z \cdot v) + 2n\sigma_i(z \cdot v)^2 - 6C^2\sigma_i = 0$$

$$2\sqrt{n}\sigma_i + 2n\sigma_i^2(z \cdot v) = 0,$$

for having extremal points within the rectangle defined by the limits of the two variables.

Since the second condition gives that  $(z \cdot v)$  is negative, we can conclude that our studied function will have its minimum value somewhere on the boundary of the rectangle defined by the restrictions of our two variables.

Checking the four sides of the rectangle, we get that our expression is positive on the boundary as long as  $n \geq 2$  which we know from the statement in the theorem. (Anyhow, solving the linear program in one dimension is a trivial task).

On the other hand, if  $\sigma_i$  is negative we get that the expression  $\frac{a'}{\|a'\|} \cdot z$  is at least

$$\rho \frac{31}{32\sqrt{1 + 3\sigma^2}},$$

since  $0 \geq \sigma_i \geq -\sigma$  and  $(z \cdot v) \leq 1$ .

Now we are about to bound  $\|z'\|$  from above, and for convenience we study the square of it,  $\|z'\|^2$ . We know that

$$\|z'\|^2 = \|z + \alpha(z \cdot v)v\|^2 = 1 + (\alpha^2 + 2\alpha)(v \cdot z)^2$$

Inserting our known  $\alpha = (\rho\sqrt{n} - 1)/2$  we get that

$$\|z'\|^2 = 1 + ((\alpha + 1)^2 - 1)(v \cdot z)^2 \leq 1 - \frac{7}{16} = \frac{9}{16},$$

since  $\alpha + 1 \leq 3/4$ .

Using the identity

$$\frac{1}{\sqrt{1+\beta}} \geq 1 - \frac{\beta}{2}$$

for  $\beta \in (-1, 1)$ , we find that the total estimate for the new margin is

$$\rho' \geq \rho \left(1 - \frac{1}{2^5\sqrt{n}}\right) \left(1 - \frac{3}{2^{11}n}\right) \left(\frac{4}{3}\right) \geq \frac{5}{4}\rho,$$

when  $\sigma_i$  is positive. Otherwise, when  $\sigma_i$  is negative we get that

$$\rho' \geq \rho \left(\frac{31}{32}\right) \left(1 - \frac{3}{2^{11}n}\right) \left(\frac{4}{3}\right) \geq \frac{5}{4}\rho.$$

This estimate will be enough to fulfill the demand we have on the margin to increase with a large proportion, giving us a guarantee for one of the  $2n$  different tracks to terminate fast enough to speed up the algorithm in total.  $\square$

Now we describe the classical Perceptron Algorithm, but reduced to at most  $n$  steps, being an important component of the modified algorithm.

**Algorithm 8.4.4.** *Perceptron Algorithm*

*Input:* A starting vector  $v$  from  $V$ .

*Output:* A feasible solution  $Bv$  or a new updated vector  $v$ .

1. If there is a row  $a$  in  $A$  such that  $v \cdot a \leq 0$ , then  $v \leftarrow v + a/\|a\|$ .
2. If  $Av \geq 0$ , then output  $Bv$  as a feasible solution and stop. (Here you choose the matrices  $A$  and  $B$  that corresponds to the specified vector  $v$ .)
3. Repeat the two steps above at most  $R = 4n$  times.

The behaviour of this algorithm is well-studied, and we know from, for example, [32] that it produces a feasible solution if the problem has a margin  $\rho$  of size at least  $1/\sqrt{R} = 1/(2\sqrt{n})$ . Also we can conclude that:

**Proposition 8.4.2.** *The number of iterations inside the Perceptron Algorithm is of order  $O(mn^2)$ .*

*Proof:* The algorithm will perform at most one matrix-vector multiplication in its inner loop (made at time  $O(mn)$ ) and a constant number of vector manipulations (in time  $O(n)$ ). This is done at most  $2n$  times. So we get  $O(mn^2)$ .  $\square$

**Lemma 8.4.1.** *The number of times we repeat step 2-5 in Algorithm 8.4.1 for a single start vector is at most of order  $O(\log(1/\rho))$ .*

*Proof:* As we have seen above, at least one of our  $2n$  parallel processes will start with a vector  $v$  satisfying  $v \cdot z \geq 1/\sqrt{n}$  where  $z$  is a feasible unit vector. So, after the wiggling phase, the resulting vector will be a proper direction for rescaling, this enlarges the radius  $\rho$  with a factor of size at least  $(1 + 1/4)$ . But since the perceptron stage will terminate, yielding a feasible solution if  $\rho \geq 1/(2\sqrt{n})$ , we know that our algorithm will terminate after  $k$  proper rescalings when

$$\rho \left(1 + \frac{1}{4}\right)^k \geq \frac{1}{2\sqrt{n}}.$$

This yields that  $k = O(\log(1/\rho))$   $\square$

Summing up the information we have got, we get the complexity of the algorithm in total.

**Theorem 8.4.3.** *The modified Perceptron Algorithm returns an answer in time*

$$O\left(mn^3 \log n \log\left(\frac{1}{\rho}\right)\right).$$

*Proof:* The algorithm runs in  $2n$  parallel processes. The wiggle algorithm runs for at most  $2^{10}n \log n$  times, each round taking time  $O(mn)$ . The rescale process takes  $O(n^2)$  and the Perceptron algorithm runs for at most  $n$  times, each round taking time  $O(mn)$ . All these three stages are repeated at most  $O(\log(1/\rho))$  times. So we get that the total time is

$$\begin{aligned} &O\left(2nO\left(\log\left(\frac{1}{\rho}\right)\right)(2^{10}mn^2 \log n + O(n^2) + mn^2)\right) = \\ &= O\left(mn^3 \log n \log\left(\frac{1}{\rho}\right)\right). \end{aligned}$$

$\square$

### 8.4.2 Parallel Processors

The way the algorithm is structured, it falls naturally into  $2n$  parts that can run on separate processors, since the algorithm or the outcome from any one of these parts is not dependent on any result or subresult from the other parts. This implies that if we run the algorithm on  $2n$  processors, we will solve the problem even faster. We conclude this remark with the following corollary:

**Corollary 8.4.1.** *The modified Perceptron Algorithm, when ran on  $2n$  parallel processors, will return an answer in time*

$$O\left(mn^2 \log n \log\left(\frac{1}{\rho}\right)\right).$$

But when the algorithm is used for a problem in a large number of dimensions, this would result in twice as large number of processors. What you would like, is to reduce the number of processors if possible. And since the structure of the algorithm indicate that each processor is using one of the  $2n$  start vectors, we would ask ourselves if it is possible to reduce the number of start vectors in some way. Certainly we could reduce it by one, since if we rotate all  $m$  examples in such a way that  $a_1 = e_1$ , a unit vector from the used orthonormal basis, we know for sure that the start vector  $-e_1$  does not have an inner product with the optimal solution  $z$  greater than  $1/\sqrt{n}$ , since  $z \cdot e_1 \geq 0$ . But running on  $2n$  or  $2n - 1$  processors does not make a big difference.

Instead, we continue this idea of rotating a random example, and rotate example  $a_1$  to the point  $(1/\sqrt{n}, \dots, 1/\sqrt{n})$ . Now we have to ask ourselves if this would reduce the number of start vectors. If this was possible we would have to prove that any solution  $z$  of unit length, where  $z \cdot (1/\sqrt{n}, \dots, 1/\sqrt{n}) \geq 0$ , fulfills the following:  $z \cdot e_i \geq 1/\sqrt{n}$  for at least one  $i$ . Unfortunately this is not the case when the number of dimensions are greater than 2, so we have to analyse the problem even further.

The restriction for a start vector  $e_i$  or  $-e_i$  to end up as a solution to the problem when running the algorithm, is that its inner product with a unit length solution  $z$  is at least  $1/\sqrt{n}$ , comes from the estimates made within the  $\sigma$ -relaxed wiggle procedure, and is a number that can be changed without interfering with other parts in the algorithm. Though, changing this number does imply that we have to recalculate how many times this procedure should be ran.

The intention here is to show that changing  $1/\sqrt{n}$  to  $1/n$  would imply that a rotation, as described briefly above, reduces the number of needed start vectors by fifty percent. At the same time the running time will increase with at most a factor two, thus preserving the magnitude of the running time.

In order to achieve this we have to prove the following theorem:

**Theorem 8.4.4.** *If*

$$\sum_{i=1}^n x_i \geq 0 \quad \text{and} \quad \sum_{i=1}^n x_i^2 = 1$$

then is  $x_i \geq 1/n$  for some  $i$ .

To prove the above statement we will start with mentioning Jensen's inequality that will be used later on.

**Theorem 8.4.5.** (*Jensen's inequality*)

If  $f(x)$  is a convex function and  $\sum_{i=1}^k \alpha_i = 1$ , then

$$\sum_{i=1}^k \alpha_i f(x_i) \geq f\left(\sum_{i=1}^k \alpha_i x_i\right).$$

We will use this inequality for the following corollary:

**Corollary 8.4.2.** For every combination of realvalued  $x_i$ 's we have that

$$\frac{1}{k} \sum_{i=1}^k x_i^2 \geq \left(\frac{1}{k} \sum_{i=1}^k x_i\right)^2.$$

*Proof:* Use Jensen's inequality by letting  $f(x) = x^2$ , a convex function, and let  $\alpha_i = 1/k$  for every  $i$ . □

With this in mind we can prove the following helpful lemma:

**Lemma 8.4.2.** If  $\sum_{i=1}^k x_i^2 = B$  and  $\tilde{x}_i = \sqrt{B/k}$  for every  $i$ , then

$$\sum_{i=1}^k \tilde{x}_i \geq \sum_{i=1}^k x_i.$$

*Proof:* We can assume that  $\sum_{i=1}^k x_i \geq 0$ , since the statement becomes trivial otherwise. Then is the statement in the lemma equivalent to

$$\left(\sum_{i=1}^k \sqrt{\frac{B}{k}}\right)^2 \geq \left(\sum_{i=1}^k x_i\right)^2,$$

which in turn is equivalent to

$$kB \geq \left(\sum_{i=1}^k x_i\right)^2.$$

But this is equivalent to

$$\frac{1}{k} \sum_{i=1}^k x_i^2 \geq \left( \frac{1}{k} \sum_{i=1}^k x_i \right)^2,$$

which was proven in the above corollary.  $\square$

Also we will need the following easily proven propositions:

**Proposition 8.4.3.** *If  $\sum_{i=1}^k x_i^2 = B$  and  $x_i \geq 0$  for every  $i$ , then there exists a  $j$  such that  $x_j \geq \sqrt{B/k}$ .*

*Proof:* By contradiction. Assume that  $x_j < \sqrt{B/k}$  for every  $j$ . Then we get

$$B = \sum_{i=1}^k x_i^2 < k \frac{B}{k} = B,$$

which is a contradiction.  $\square$

**Proposition 8.4.4.** *If  $y_j \geq 0$  for every  $j$ , then is*

$$\sum_{j=1}^l y_j^2 \leq \left( \sum_{j=1}^l y_j \right)^2.$$

*Proof:* The proposition follows from the fact that

$$\left( \sum_{j=1}^l y_j \right)^2 = \sum_{j=1}^l y_j^2 + 2 \sum_{i < j} y_i y_j \geq \sum_{j=1}^l y_j^2,$$

when every  $y_j \geq 0$ .  $\square$

Now we are prepared to prove theorem 8.4.4 stated above:

*Proof:* We set out to prove the following equivalent statement: If  $\sum_{i=1}^n a_i \geq 0$  and  $\sum_{i=1}^n a_i^2 = A$ , then is  $a_i \geq \sqrt{A/n}$  for some  $i$ . First we sort and renumber the  $a_i$ 's according to the following:

$$\{a_1, \dots, a_n\} = \{x_1, \dots, x_k, -y_1, \dots, -y_l\},$$

where  $x_i \geq 0$  and  $y_i \geq 0$  for every  $i$ ,  $k \geq 1$  and  $l \geq 0$ . Then we have that

$$\sum_{i=1}^k x_i \geq \sum_{j=1}^l y_j \quad \text{and} \quad \sum_{i=1}^k x_i^2 + \sum_{j=1}^l y_j^2 = A.$$

Now, let  $\sum_{i=1}^k x_i^2 = B$ . If  $\tilde{x}_i = \sqrt{B/k}$  for every  $i$  we have, according to our lemma above, that

$$\sum_{i=1}^k \tilde{x}_i \geq \sum_{j=1}^l y_j \quad \text{and that} \quad \sum_{i=1}^k \tilde{x}_i^2 + \sum_{j=1}^l y_j^2 = A.$$

If we now show the theorem for  $\tilde{x}_i$  instead, we are about to show that  $\tilde{x}_i = \sqrt{B/k} \geq \sqrt{A}/n$ , where  $n = k + l$ . According to the proposition above, we have in this case also proved the theorem for some  $x_i$ . Therefore we can suppose that

$$x_1 = x_2 = \dots = x_k = x$$

and that our aim is to show that  $x \geq \sqrt{A}/n$ .

The case  $l = 0$  is trivial since that implies that  $x = \sqrt{A}/n$ . If  $l \geq 1$  we can, instead of consider  $y_1, \dots, y_l$ , consider  $\tilde{y}_1 = \sum_{j=1}^l y_j$  and let  $\tilde{y}_2 = \tilde{y}_3 = \dots = \tilde{y}_l = 0$ . This is possible since  $\sum_{j=1}^l \tilde{y}_j = \sum_{j=1}^l y_j$  and  $\tilde{A} = \sum_{j=1}^l \tilde{y}_j^2 \geq \sum_{j=1}^l y_j^2 = A$  according to our second proposition above.

Since  $n = k + l$  and  $\sqrt{\tilde{A}}/(k + 1) \geq \sqrt{A}/(k + l)$  it will be enough to consider the case when  $l = 1$ , where we denote  $y_1$  by  $y$ . Now we have got that

$$\sum_{i=1}^k x_i = kx \geq y \quad \text{and} \quad kx^2 + y^2 = A.$$

We want to show that

$$x \geq \sqrt{kx^2 + y^2}/(k + 1),$$

but this is equivalent to

$$x^2 \geq \frac{kx^2 + y^2}{(k + 1)^2},$$

which is equivalent to

$$x^2(k^2 + k + 1) \geq y^2,$$

which is true since  $kx \geq y$  and  $k^2x^2 \geq y^2$ . □

**Proposition 8.4.5.** *The  $\sigma$ -relaxed wiggle procedure, starting with a vector  $x_0$  that satisfies  $x_0 \cdot z \geq 1/n$ , where  $z$  is a solution of unit length, will end in at most  $2(\ln n)/\sigma^2$  iterations.*

*Proof:* Let  $z$  be a unit vector that correctly classifies all examples  $a_i$  in  $A$ . Suppose that an initial unit vector  $x_0$  satisfies  $x \cdot z \geq 1/n$ . Denote the vector achieved in  $t$  updates within the algorithm by  $x_t$ . Notice first that in each update made,  $x_t \cdot z$  does not decrease with increasing  $t$  because

$$x_t \cdot z = (x_{t-1} - (x_{t-1} \cdot a_i)a_i) \cdot z = x_{t-1} \cdot z - (x_{t-1} \cdot a_i)(z \cdot a_i) \geq x_{t-1} \cdot z$$

where the last inequality holds because  $x_{t-1}$  misclassifies  $a_i$ . On the other hand,  $|x_t|^2$  does decrease significantly because

$$\begin{aligned} |x_t|^2 &= |x_{t-1} - (x_{t-1} \cdot a_i)a_i|^2 = |x_{t-1}|^2 - 2(x_{t-1} \cdot a_i)^2 + (x_{t-1} \cdot a_i)^2 \leq \\ &\leq |x_{t-1}|^2(1 - \sigma^2) \end{aligned}$$

using the Pythagorean Theorem. Thus, after  $t$  iterations  $|x_t| \leq (1 - \sigma^2)^{\frac{t}{2}}$ . Since  $|x_t|$  cannot be less than  $x \cdot z$ , this means that the number of iterations  $t$  satisfies

$$(1 - \sigma^2)^{\frac{t}{2}} \geq \frac{1}{n},$$

which implies that  $t \leq (2 \ln n)/\sigma^2$ .  $\square$

This implies that, if we rotate the set of examples such that one example becomes  $(1/\sqrt{n}, \dots, 1/\sqrt{n})$  and change the number of rounds taken through the wiggle procedure, we only have to consider the  $n$  unit vectors in our orthonormal base as start vectors.

**Corollary 8.4.3.** *An  $n$ -dimensional problem can be solved on  $n$  processors in time*

$$O\left(mn^2 \log n \log\left(\frac{1}{\rho}\right)\right).$$

### 8.4.3 Expected Time for the Modified Perceptron Algorithm

The above discussions about the performance of the modified perceptron algorithm has all the time been a discussion about the worst case scenario. An interesting question to ask, after the above discussion, is how often we are ending up in this scenario. In some algorithms you end up in the worst case scenario most of the time, but in others not. Here we will show that the expected performance of the modified perceptron algorithm is in the general case a factor  $n$  faster than the previous calculated worst case time that was of order

$$O\left(mn^3 \log n \log\left(\frac{1}{\rho}\right)\right)$$

when not using parallel processors.

Here we are going to use the  $2n$  different start vectors as in the described algorithm, but we are going to choose them randomly and then calculate the expected time the algorithm will use before it returns a feasible solution to the linear program. We will use the earlier worst case estimates for each part of the algorithm and only make an estimate for how many start vectors that will be used before the algorithm terminates.

So, we start with a linear program having a feasible solution  $z$  of unit length, a solution that is picked from a uniform distribution on the  $n$ -dimensional sphere. We know from earlier results in this chapter that when we choose one of our  $2n$  start vectors at random, we have a probability of  $\Pi_n$  of choosing a correct one from start and the algorithm will terminate. Otherwise, with probability  $1 - \Pi_n$ , we will choose another of our start vectors to run the algorithm with. Same again, with probability at least  $\Pi_n$  (we have taken away one bad start vector) the algorithm will terminate, and with a probability of at most  $1 - \Pi_n$  we choose yet another start vector for our algorithm. We know the time complexity for the worst case, running through all these  $2n$  cases, but the expected time for terminating the algorithm will be bounded above according to the following:

**Proposition 8.4.6.** *The expected number of start vectors used by the modified perceptron algorithm, used on a problem with a solution taken from a uniform distribution, is bounded above by 7.*

*Proof:* Let  $T$  denote the worst time taken for the algorithm used on one start vector. If it ends after the  $k$ :th choice, the algorithm has used at most time  $kT$  to find a solution and halt. The expected time used will be bounded by

$$\begin{aligned} & \Pi_n T + \Pi_n(1 - \Pi_n)2T + \Pi_n(1 - \Pi_n)^2 3T + \cdots + \Pi_n(1 - \Pi_n)^{2n} 2nT \\ &= \Pi_n T (1 + 2(1 - \Pi_n) + 3(1 - \Pi_n)^2 + \cdots + 2n(1 - \Pi_n)^{2n}) \leq \\ & \leq \Pi_n T \sum_{n=1}^{\infty} n(1 - \Pi_n)^{n-1}. \end{aligned}$$

But using that

$$f(y) = \sum_{n=0}^{\infty} y^n = \frac{1}{1-y}$$

and

$$f'(y) = \sum_{n=1}^{\infty} n y^{n-1} = \frac{1}{(1-y)^2}$$

by differentiating, we can substitute with  $y = 1 - \Pi_n$ , yielding that our first expression is bounded above by

$$\Pi_n T \frac{1}{\Pi_n^2} = T \frac{1}{\Pi_n} \leq 7T$$

for every  $n$ , since

$$\Pi_n \geq \frac{1}{2}(1 - \operatorname{erf}(1/\sqrt{2})) > \frac{1}{7}.$$

□

This implies that the expected time taken for the algorithm to return an answer is at most  $7T$  while the worst case scenario gave an upper bound on  $2nT$ , where  $n$  is the number of dimensions on the stated problem.

**Corollary 8.4.4.** *The expected time for the modified perceptron algorithm to solve a problem is at most*

$$O\left(mn^2 \log n \log\left(\frac{1}{\rho}\right)\right).$$

## 8.5 A Generalized Perceptron

One of the things mathematicians are up to, is to investigate if there are possibilities to generalize the results achieved. In this short section we will mention some natural ways to extend the results concerning the perceptron algorithm presented in this thesis. Instead of speaking about vectors in euclidian space, we will consider more general mathematical entities, and find some non-trivial results concerning these spaces.

### 8.5.1 Inner Product Spaces

When you go through the details of the proofs concerning the perceptron algorithm and its modifications, there is one thing they seem to have in common. All the proofs (maybe for simplicity) concern points and/or vectors in the euclidian space  $\mathbb{R}^n$ , but in most cases there are no restrictions to instead speak about real inner product spaces. However, when starting to look at spaces with an infinite number of dimensions, we could perhaps end up in some difficulties.

### 8.5.2 Real Inner Product Spaces of Finite Dimension

In the case of a real inner product space with a finite number of dimensions, there are really no problems to generalize the earlier theorems concerning the perceptron algorithm. We start with the classical theorem, that informs us about the number of updates that can be made at most by the on-line perceptron algorithm. Here we make some conclusions where the algorithm is as before, but changing the dot-product for an inner product throughout the algorithm.

**Theorem 8.5.1.** *The number of mistakes made by the on-line perceptron, on a non-trivial set  $A$  (from a real inner product space of finite dimension) that has a solution to the problem, is at most  $(1/\gamma)^2$ , where  $\gamma$  is the size of the maximal margin according to the above definition.*

*Proof:* First recall the definition of the margin made earlier in the thesis, and conclude that the definition is suitable for our purposes when switching to an inner product space. As before we consider the simpler form of the algorithm where the hyperplane to find passes through origin, thus there are no bias, and that all examples are of unit length and also positive, so we do not have to consider the sign of any constraint.

Now let  $z$  denote a solution where  $\langle z, a_i \rangle \geq \gamma$  for every  $i$ , and let  $x_t$  denote the hypothesis  $x$  in the algorithm after  $t$  updates.

$$\langle x_t, z \rangle = \langle x_{t-1}, z \rangle + \langle a_i, z \rangle \geq \langle x_{t-1}, z \rangle + \gamma$$

for the  $a_i$  that was the last instance of misclassification yielding an update. This implies, by induction, that  $\langle x_t, z \rangle \geq t\gamma$ .

Similarly, we have

$$\|x_t\|^2 = \|x_{t-1}\|^2 + 2\langle x_{t-1}, a_i \rangle + \|a_i\|^2 \leq \|x_{t-1}\|^2 + 1$$

since  $\langle x_{t-1}, a_i \rangle$  is negative, which implies that

$$\|x_t\|^2 \leq t.$$

Combining these two results we get the following squeezing relation:

$$\|z\|\sqrt{t} \geq \|z\|\|x_t\| \geq \langle x_t, z \rangle \geq t\gamma,$$

which imply that  $t \leq 1/\gamma^2$ . □

In a similar way we can conclude that all previous results concerning finite euclidian space can be modified to any real inner product space of finite dimension. The generalization is a bit more intricate though if we start to consider spaces of infinite dimensions.

### 8.5.3 Real Inner Product Spaces of Infinite Dimensions

To illustrate the new situation when dealing with an infinite number of dimensions in an inner product space we start with the following proposition as an example:

**Proposition 8.5.1.** *Let  $A = \cup_i e_i$  be an orthogonal basis of unit vectors of our infinite dimensional inner product space. Also let  $A$  be our set of examples for an on-line perceptron algorithm. Even though there is a solution  $z$ , such that  $\langle z, e_i \rangle > 0$  for every  $i$ , the algorithm will never halt.*

*Proof:* First we can conclude that

$$z = \left\{ 1, \frac{1}{2}, \dots, \frac{1}{n}, \dots \right\}$$

is a feasible solution to our problem. But as long as the algorithm has passed through a finite number of steps, there are still an infinite number of examples being misclassified by the algorithm.  $\square$

But there might be ways to deal with this problem. One way could be to only look for solutions where  $\langle z, e_i \rangle \geq 0$ , then the algorithm in this case would halt after the first hypothesis. But this could still get us into trouble due to the number of dimensions.

What is needed to consider, to assure us of having a terminating algorithm, is problems with strictly positive margin  $\gamma$ . In the example given above we get that the margin is  $\gamma(z, A) = \inf_n \frac{1}{n} = 0$  which in turn made our algorithmic procedure to go on for ever. The observant reader has already noted that the property of having a solution with a strictly positive margin is a necessary condition for our algorithm to end in a finite number of steps from the proof above, but it is worth to point out here in the case of an infinite number of dimensions in our problem.

So with the above discussion we can conclude the same theorem for real inner product spaces of infinite dimensions:

**Theorem 8.5.2.** *The number of mistakes made by the on-line perceptron, on a non-trivial set  $A$  (from a real inner product space of infinite dimension) that has a solution to the problem, is at most  $(1/\gamma)^2$ , where  $\gamma$  is the size of the maximal margin according to the above definition.*

*Proof:* Same as in theorem 8.5.1.  $\square$

This is perhaps a bit surprising. Not only that the perceptron algorithm could handle an infinite number (of any cardinality) of examples and halt after a finite number of updates, but also an infinite number of dimensions and still halt after a finite number of updates, as long as the solution to the problem has a strictly positive margin.

Anyhow, now we have an algorithm that in a practical way can find a separating hyperplane between two disjoint convex sets, as done in the area of functional analysis. We know the following theorem from any introduction in functional analysis, (usually in a much more general form):

**Theorem 8.5.3.** *Two disjoint, convex and compact sets  $A$  and  $B$  in a real inner product space can be strictly separated by a hyperplane.*

The proof of this existence theorem uses the Hahn-Banach theorem, that in its own turn uses a Zorn's lemma argument.

But now, by the above discussion, we can construct such a hyperplane, with a finite number of updates made within the perceptron algorithm, as long as the minimum distance between the sets is strictly positive.

Restating the problem without bias and no negative examples, we can for instance use the generalized perceptron algorithm to solve the following example problem:

**Proposition 8.5.2.** *Assume we have a collection  $A = \cup_{i \in I} f_i$  of real-valued functions in  $L^2[x]$  and we know that there exists a real-valued function  $g \in L^2[x]$  such that  $\langle f_i, g \rangle \geq$*

$\gamma > 0$  for every  $i \in I$ . Then we can find a real-valued function  $g' \in L^2[x]$  such that  $\langle f_i, g' \rangle > 0$  for every  $i \in I$  with a finite number of updates made within the perceptron algorithm.

*Proof:* The space made up by all real-valued  $L^2[x]$ -functions is a real inner product space under its usual inner product  $\langle f, g \rangle = \int fg dx$ . So, the generalized perceptron algorithm will end after a finite number of updates since the margin  $\gamma$  is strictly positive.  $\square$

**Remark: Complex Inner Product Spaces**

In the case of complex inner product spaces we immediately get a problem of interpretation of the original problem. Say that we are looking for an element  $x$  such that  $\langle x, a_i \rangle \geq 0$  for every element  $a_i \in A$ . Then we do not really know how to interpret  $\langle x, a_i \rangle \geq 0$ , since the inner product returns a complex number. But we could restrict ourselves to only look at the real part, and solve the problem as before.



## Chapter 9

# Finding Maximal Margins with the Modified Perceptron Algorithm

In this chapter we construct an iterated perceptron-rescaling algorithm that approximates an optimal solution, among the feasible solutions, for a linear program, terminating in median time  $O(k^2mn^{7/2})$ . Here  $n$  is the number of dimensions,  $m$  is the number of constraints and  $k$  reflects that the answer given by the algorithm differs at most with  $2^{-k}$  from an optimal solution to the problem.

Linear programs (LP) arise naturally in many areas. The standard form of LP is

$$\max c^T x, \quad Ax \leq b, \quad x \geq 0,$$

where  $x, c \in \mathbb{R}^n$ ,  $b \in \mathbb{R}^m$  and  $A$  is an  $m \times n$  matrix of real numbers. The feasibility version of a standard form LP neglects the objective function  $c^T x$ . It can be shown (cf. [45]) that an approximate solution to an LP always can be found by repeatedly solving its feasibility version. The feasibility standard form of an LP, in its turn, can be reduced to the homogenised form

$$Ax \geq 0, \quad x \neq 0.$$

These methods of rewriting standard LP allow for a wide variety of solution techniques for linear programs to exist.

Here we are, for simplicity, only considering linear programs where each constraint (or example)  $a_i$  has unit length. The different parts of the algorithm could be adjusted for the more general case, but this is not done here for clarity reasons. It is important though to note that you can not transform a general problem to one with every constraint having unit length, as before, and guarantee the same optimal solution. The transformation is only successful when looking for a feasible solution.

## 9.1 Introduction to the Iterated Perceptron-Rescaling Algorithm

The algorithms presented earlier in the thesis are for finding feasible solutions to the above stated problem. But many times we are not only interested in finding a solution, but to find as good solution as possible. First of all we must define what makes a solution better than another, and then outline a strategy for how to achieve this.

The aim of this chapter is to show that a generalised form of the deterministic algorithm presented in [15] can be used to construct an iterative algorithm that converges to

an optimal solution. This perceptron-rescaling algorithm will work in polynomial time, speaking in median behaviour terms.

### 9.1.1 Comparison of Solutions

A feasible solution to the problem  $Ax \geq 0$  is a hyperplane through origin having all datapoints  $a_i$  on the same side of the hyperplane. A measure of how good this solution is, is to determine the minimal distance from all datapoints to the hyperplane. This distance is usually called the margin. So if  $x$  is the unit vector normal to the hyperplane, pointing towards the half-space containing the data set, the margin  $\alpha$  for this feasible solution is defined as

$$0 < \alpha = \min_i x \cdot \frac{a_i}{\|a_i\|}.$$

An optimal solution is consequently defined as a feasible solution that maximises the margin above. So if  $X$  is the set of all unit vectors being a feasible solution to the problem, we have that an  $x \in X$  that maximises the margin is an optimal solution to the problem. The margin for an optimal solution will be denoted  $\gamma^*$ . Also, if  $x_1$  and  $x_2$  are two different unit normal vectors, both defining feasible solutions for our problem with margins  $\alpha_1$  and  $\alpha_2$  respectively, we say that  $x_1$  is better than  $x_2$  if  $\alpha_1 > \alpha_2$ . The idea of using an iterated perceptron to find such an optimal solution originates from [5].

### 9.1.2 Outline of the Algorithmic Strategy

Before going into details of the final algorithm, we begin with an outline of our strategy. We can think of the important part of the algorithm as a Black Box where the input is an  $m \times n$  matrix  $A$  and a margin  $\alpha$ , and the output is either 0 (zero), saying that no feasible solution with margin at least  $\alpha$  exists, or 1, where 1 says that there is a feasible solution with a margin of size at least  $\alpha$ .

This oracle, our Black Box, makes it possible for us to determine  $\gamma^*$  with accuracy  $2^{-k}$  with  $k$  questions given to the oracle. This is the case since the margin in our reformulation of the problem never exceeds 1.

#### Algorithm 9.1.1. Approximating the Optimal Margin with a Black Box

*Input:* An  $m \times n$  matrix  $A$  and a positive integer  $K$ .

*Output:* An interval of size  $2^{-K}$  containing the optimal margin  $\gamma^*$ .

1. Let  $I = [0, 1]$ ,  $k = 1$  and  $\alpha = 2^{-1}$ .
2. Let  $k \leftarrow k + 1$ .
3. *Input*  $A$  and  $\alpha$  in the Black Box.  
 If the output of the Black Box is 0, let  $I \leftarrow I \setminus (\alpha, 1]$  and  $\alpha \leftarrow \alpha - 2^{-k}$ .  
 If the output of the Black Box is 1, let  $I \leftarrow I \setminus [0, \alpha)$  and  $\alpha \leftarrow \alpha + 2^{-k}$ .

4. If  $k \leq K$ , goto 2, otherwise output  $I$ .

The resulting output is an interval of length  $2^{-K}$ , giving an approximation to  $\gamma^*$ . We have omitted to output the normal  $x$  giving this solution in order to simplify the later coming algorithms in this paper.

### 9.1.3 The Black Box

The most crucial part in the construction of the Black Box is to both have the possibility of a definitive YES and a definitive NO in the output answer. Many algorithms, finding a feasible solution to a linear program, have only a definite answer in the affirmative, but for our purposes this is not a sufficient condition. The algorithm inside the Black Box can be based on any linear program solver having the above properties. Here we will use a modified perceptron-rescaling algorithm, based on results by Dunagan and Vempala in [35], made deterministic by the authors in [15], which is a necessary condition for this Black Box to behave properly, and made working at a greater speed in the previous chapter.

## 9.2 A Deterministic Perceptron-Rescaling Algorithm

Now we present a deterministic version of the perceptron-rescaling algorithm suggested in [35]. This construction will correspond to a Black Box as described above, but only made for one single question, namely if there exists a feasible solution at all, that is having  $\alpha = 0$  as input. Further on we will prove that this algorithm can be generalized to handle questions for a general  $\alpha$ .

### Algorithm 9.2.1. (Dunagan & Vempala, modified by Barr & Wigelius)

*Input:* An  $m \times n$  matrix  $A$ .

*Output:* A point (vector)  $x$  such that  $Ax \geq 0$  and  $x \neq 0$ .

1. Let  $B = I$ ,  $\sigma = \frac{1}{32\sqrt{n}}$  and  $V = \bigcup_{i=1}^n \{e_i, -e_i\}$ .
2. (Perceptron Phase)
  - (a) Let  $x$  be the origin in  $\mathbb{R}^n$ .
  - (b) Repeat at most  $4n$  times:  
If there exists a row  $a$  in  $A$  such that  $a \cdot x \leq 0$ ,  
then  $x \leftarrow x + a/\|a\|$ .
3. If  $Ax \geq 0$ , then output  $Bx$  as a feasible solution and stop.
4. (Wiggling Phase)
  - (a) For each vector  $v_0 \in V$ :

- Let  $v = v_i$ . Repeat at most  $\frac{\ln n}{\sigma^2}$  times the following wiggling procedure :  
 If there exists a row  $a$  such that  $\frac{a \cdot v}{\|a\| \|v\|} < -\sigma$ ,  
 then  $v \leftarrow v - \left(\frac{a}{\|a\|} \cdot v\right) \frac{a}{\|a\|}$ .
- If  $Av \geq 0$ , then output  $Bv$  as a feasible solution and stop.

(b) Let  $V = \bigcup \{v \mid \frac{a \cdot v}{\|a\| \|v\|} \geq -\sigma\}$ .

5. (Rescaling Phase; in at most  $2n$  directions)

For each  $i$ :

(a) let

$$A \leftarrow A \left( I + \frac{v}{\|v\|} \frac{v}{\|v\|}^T \right) \text{ and}$$

$$B \leftarrow B \left( I + \frac{v}{\|v\|} \frac{v}{\|v\|}^T \right)$$

(b) Run the perceptron at most  $4n$  times on each  $A$ .

If it terminates with the solution  $x$ , output  $Bx$  as a feasible solution and stop.

6. If no feasible solution has been obtained in step 5, go back to step 4.

### 9.3 The Black Box for a General $\alpha$

The above presented algorithm consists of three major parts; the perceptron phase, the wiggling phase and the rescaling phase. The aim now is to prove that we can make general updates, dependent of  $\alpha$  in the perceptron phase and in the wiggling phase. Also we must show that the rescaling phase still works. Furthermore, we are going to calculate the complexity of this iterated perceptron.

We could describe the algorithm as follows: A positive return from the perceptron phase will correspond to output 1 from the Black Box; and a negative return from the wiggling phase, meaning that we end up with  $V = \emptyset$ , will correspond to output 0 from the Black Box. The rescaling phase, combined with the other phases, is a construction to assure us of a polynomial behaviour of the algorithm, and at the same time a way of altering between the two phases that corresponds to a definite YES and a definite NO.

#### 9.3.1 The Perceptron Phase

The proofs and discussions made here can easily be generalised to a general perceptron with labels, bias and learning rate. But for clarity of our presentation we have chosen to use our reformulation of the problem where all constraints  $a_i$  are on the unit sphere. First we present the  $\alpha$ -Perceptron Algorithm, a general form of the perceptron made by Rosenblatt (cf [58]). Another similar generalization has been discussed and analysed by Amiran Ambroladze [5]. In the algorithm,  $a_i$  denotes the vector defined by the  $i$ :th row of  $A$ .

**Algorithm 9.3.1. The  $\alpha$ -Perceptron Algorithm**

*Input:* An  $m \times n$  matrix  $A$ , with each row vector on the unit sphere, and an  $\alpha \in [0, \gamma^*)$ .

*Output:* A point (vector)  $x$  such that  $Ax > \alpha$ .

1.  $x \leftarrow 0$ .
2. *for*  $i = 1$  *to*  $m$ 
  - if*  $a_i \cdot \frac{x}{\|x\|} \leq \alpha$ , *then*
    - $x \leftarrow x + a_i$
  - end if*
- end for*
3. *Repeat* 2 *until* no updates are made within the *for-loop*.

Now we need to know that this algorithm terminates and also how fast this happens in terms of the number of updates made within the algorithm in terms of  $\alpha$ ,  $\gamma^*$  and  $\gamma = \gamma^* - \alpha$ .

**Theorem 9.3.1.** *Suppose there exists a unit vector  $z$  such that  $a_i \cdot z \geq \gamma^*$  for every  $i$ . Then the number of updates made by the on-line  $\alpha$ -perceptron algorithm is at most  $\frac{2}{\gamma^2 \gamma^*}$ .*

*Proof:* Let  $x_t$  denote the vector  $x$  in the algorithm after  $t$  updates made. First of all we conclude that

$$x_t \cdot z = x_{t-1} \cdot z + a_i \cdot z \geq x_{t-1} \cdot z + \gamma^*,$$

which implies (by induction) that  $x_t \cdot z \geq t\gamma^*$ . Similarly, we have  $\|x_t\|^2 =$

$$\begin{aligned} \|x_{t-1}\|^2 + 2(x_{t-1} \cdot a_i) + \|a_i\|^2 &= \|x_{t-1}\|^2 + 2(x_{t-1} \cdot a_i) + 1 \leq \|x_{t-1}\|^2 + 2\alpha\|x_{t-1}\| + 1 \\ &\leq \|x_{t-1}\|^2 + 2\alpha\|x_{t-1}\| + \alpha^2 - \alpha^2 + 1 = (\|x_{t-1}\| + \alpha)^2 - \alpha^2 + 1 \leq (\|x_{t-1}\| + \gamma^* - \varepsilon)^2, \end{aligned}$$

if  $t$  is big enough (implying that  $\|x_{t-1}\|$  is big enough) and  $\gamma^* - \varepsilon \geq \alpha$ , for convenience we put  $\varepsilon = \frac{1}{2}(\gamma^* - \alpha) = \frac{1}{2}\gamma$ . This implies that  $\|x_t\|$  only grows with at most  $\gamma^* - \varepsilon$  if  $t$  is big enough. So when  $t$  is big enough,  $t > C$  say, we have that  $\|x_t\| \leq \|x_C\| + (t - C)(\gamma^* - \varepsilon)$  at the same time as  $x_t \cdot z \geq t\gamma^*$ .

The two inequalities combined give the relations

$$\|x_C\| + (t - C)(\gamma^* - \varepsilon) \geq \|x_t\| \geq x_t \cdot z \geq t\gamma^*,$$

which together imply the bound

$$\frac{\|x_C\|}{\varepsilon} \geq t.$$

An analysis of  $\|x_C\|$ , using that  $(\|x_C\| + \alpha)^2 - \alpha^2 + 1 \leq (\|x_C\| + \gamma^* - \varepsilon)^2$  is fulfilled, shows that

$$\|x_C\| \geq \frac{1 - (\gamma^* - \varepsilon)^2}{2\varepsilon}$$

which will be the case after at most  $\frac{1-(\gamma^*-\varepsilon)^2}{2\varepsilon\gamma^*}$  steps. But this implies that  $\|x_C\| \leq \frac{1}{2\varepsilon\gamma^*}$  and we get that

$$\frac{1}{2\varepsilon^2\gamma^*} \geq t.$$

This finishes the proof.  $\square$

### 9.3.2 The Wiggling Phase

The wiggling phase in the above described algorithm is an idea of a modified perceptron algorithm that was presented in [20]. This algorithm was later used in [35] to construct a perceptron-rescaling algorithm that terminates in polynomial time. In [15] the algorithm was made deterministic, a condition that is necessary for our purposes. Here we will, as in the subsection above, present a generalized form with an  $\alpha$ -dependent update.

#### Algorithm 9.3.2. The $\alpha$ -Wiggling Algorithm

*Input:* An  $m \times n$  matrix  $A$ ,  $\alpha \in [0, 1]$ ,  $\sigma$  and any  $n$ -dimensional unit vector  $x$ .

*Output:* A unit vector  $x$  such that  $\frac{x \cdot (a_i - \alpha x)}{\|x\| \|a_i - \alpha x\|} \geq -\sigma$  or  $\emptyset$ , the empty set.

1. If  $\frac{x \cdot (a_i - \alpha x)}{\|x\| \|a_i - \alpha x\|} \geq -\sigma$  for every  $a_i$ , halt and output  $x$ .

2. for  $i = 1$  to  $m$

    if  $\frac{x \cdot (a_i - \alpha x)}{\|x\| \|a_i - \alpha x\|} < -\sigma$ , then

$$x \leftarrow x - \frac{x \cdot (a_i - \alpha x)}{\|a_i - \alpha x\|} \frac{(a_i - \alpha x)}{\|a_i - \alpha x\|}$$

    end if

end for

3. Repeat 1 and 2 at most  $(\ln n)/\sigma^2$  times. Otherwise output  $\emptyset$ , the empty set.

The case when  $\emptyset$  is returned implies that the input vector  $x$  in the algorithm was a bad choice for the wiggling procedure. In every stage we will start to wiggle  $2n$  vectors, chosen such that at least one is a good initial vector for the wiggling procedure under the condition that  $\alpha \leq \gamma^*$ .

**Theorem 9.3.2.** *If  $Ax \geq \alpha$  has a solution  $z$  with norm 1 and the input vector  $x$  satisfies  $x \cdot z \geq 1/\sqrt{n}$ , then the  $\alpha$ -wiggling algorithm produces a vector  $x$  such that*

$$\frac{x \cdot (a_i - \alpha x)}{\|x\| \|a_i - \alpha x\|} \geq -\sigma$$

*is true for every row  $a_i$  in  $A$ . This is done in at most  $(\ln n)/\sigma^2$  steps.*

*Proof:* Let  $z$  be a unit vector that correctly classifies every row  $a_i$  in  $A$ , or in other words  $(z \cdot a_i) > 0$ , and suppose that  $x \cdot z \geq 1/\sqrt{n}$ . Let  $x_t$  denote the vector  $x$  in the algorithm after  $t$  updates. Notice that in each update made,  $x_t \cdot z$  does not decrease because

$$\begin{aligned} x_t \cdot z &= \left( x_{t-1} - \frac{x_{t-1} \cdot (a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \frac{(a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \right) \cdot z = \\ x_{t-1} \cdot z - \left( \frac{x_{t-1} \cdot (a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \frac{(a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \right) \cdot z &\geq x_{t-1} \cdot z \end{aligned}$$

since  $a_i \cdot z \geq \alpha$ ,  $x_{t-1} \cdot z \leq 1$  and  $x_{t-1} \cdot (a_i - \alpha x) < 0$  due to the last update made.

On the other hand,  $\|x_t\|^2$  does decrease significantly because

$$\begin{aligned} \|x_t\|^2 &= \left\| x_{t-1} - \left( x_{t-1} \cdot \frac{(a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \right) \frac{(a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \right\|^2 = \\ \|x_{t-1}\|^2 - 2 \left( x_{t-1} \cdot \frac{(a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \right)^2 + \left( x_{t-1} \cdot \frac{(a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \right)^2 &= \\ \|x_{t-1}\|^2 - \left( x_{t-1} \cdot \frac{(a_i - \alpha x_{t-1})}{\|a_i - \alpha x_{t-1}\|} \right)^2 &\leq \|x_{t-1}\|^2 (1 - \sigma^2). \end{aligned}$$

Thus, after  $t$  iterations,  $\|x_t\| \leq (1 - \sigma^2)^{t/2}$ . Since  $\|x_t\|$  cannot be less than  $x_t \cdot z$ , this means that the number of iterations  $t$  satisfies  $(1 - \sigma^2)^{t/2} \geq 1/\sqrt{n}$ , which implies that  $t \leq (\ln n)/\sigma^2$  (using a Taylor expansion).  $\square$

This procedure will be repeated inside the Black Box for each vector belonging to a certain vector set  $V$ , that from the beginning contains  $2n$  vectors. If the process terminates with all vectors in  $V$  being put to  $\emptyset$ , then  $V$  itself is the empty set. This will correspond to the answer 0, zero, from the Black Box.

### 9.3.3 The Rescaling Phase

Every vector that has not been set to the empty set in the wiggling phase is a potentially good direction to rescale the dataset with. This rescaling will increase  $\gamma$  and help the perceptron phase to terminate in its fixed number of steps. So, inside the Black Box, we rescale in the different directions contained in  $V$  and then run the perceptron phase for each rescaling made. If a solution exists and  $\gamma$  is big enough in one of these cases, the process will halt and return a vector for the solution (and the Black Box will output 1). If this is not the case, the wiggling phase will take over again.

Now we want to prove that  $\gamma = \gamma^* - \alpha$  grows after each rescaling phase, under the condition that we rescale in a correct direction. We do know that at least one of the directions in  $V$  is correct due to a proposition in [15]. We will also use the fact from [15] that the theorem is true for  $\alpha = 0$ .

**Algorithm 9.3.3. Rescaling Phase**

*Input:* An  $m \times n$  matrix  $A$  and a rescaling direction (a unit vector)  $x$ .

*Output:* An  $m \times n$  matrix  $A$ .

1.  $A \leftarrow N[A(I + xx^T)]$ ; where  $N[\cdot]$  is the function normalising each row in the input matrix.
2. Output  $A$

**Theorem 9.3.3.** Suppose that  $\gamma < 1/(2\sqrt{n})$  and let  $\sigma \leq 1/(32\sqrt{n})$ . If  $\frac{x \cdot (a_i - \alpha x)}{\|x\| \|a_i - \alpha x\|} \geq -\sigma$  is true for every row  $a_i$  in  $A$ , then  $\hat{\gamma} \geq \gamma(1 + \frac{1}{4})$ .

*Proof:* Suppose  $\frac{x \cdot (a_i - \alpha x)}{\|x\| \|a_i - \alpha x\|} \geq -\sigma$  is true for every row  $a_i$  in  $A$ . Then we know that

$$\begin{aligned} \frac{x}{\|x\|} \cdot a_i &\geq -\sigma \|a_i - \alpha x\| + \alpha \|x\| \geq \\ &\geq -\sigma(\|a_i\| + \alpha \|x\|) + \alpha \|x\| \geq -\sigma + (1 - \sigma)\alpha \|x\| \geq -\sigma. \end{aligned}$$

We know from the previous chapter that  $\hat{\gamma}^* \geq \gamma^*(1 + \frac{1}{4})$  and thus

$$\hat{\gamma} = \hat{\gamma}^* - \alpha \geq \gamma^* \left(1 + \frac{1}{4}\right) - \alpha \geq \left(1 + \frac{1}{4}\right) (\gamma^* - \alpha) = \left(1 + \frac{1}{4}\right) \gamma.$$

This concludes the proof. □

## 9.4 The Iterated Perceptron-Rescaling Algorithm

Now, after the sub-algorithms and theorems in the last section, we are able to put together the iterated perceptron-rescaling algorithm. The wiggling and rescaling phases increase  $\gamma$  and  $\gamma^*$  gradually to  $1/(2\sqrt{n})$  so that the perceptron will finally need at most  $8n^{3/2}$  steps to terminate.

### 9.4.1 Inside the Black Box

**Algorithm 9.4.1. The Black Box**

*Input:* An  $m \times n$  matrix  $A$  and an  $\alpha \in [0, 1]$ .

*Output:* 1 if there exists a feasible solution to  $Ax \geq \alpha$ , 0 otherwise.

1.  $\sigma \leftarrow 1/(32\sqrt{n})$ ,  $V = \cup_{1 \leq i \leq n} \{e_i, -e_i\}$ ,  $U = \emptyset$ .
2. Run the  $\alpha$ -perceptron phase with at most  $8n^{3/2}$  updates. If  $Ax \geq \alpha$  at any stage, halt and output 1.

3. for  $i = 1$  to  $|V|$   
     Run the  $\alpha$ -wiggling phase with  $v_i$ , yielding an updated  $v_i$  and let  $U \leftarrow U \cup v_i$   
     end for  
     Let  $V \leftarrow U$  and  $U \leftarrow \emptyset$ .
4. If  $V = \emptyset$ , halt and return 0.
5. Rescale in every direction stored in  $V$  and store each updated  $A$ .
6. Run the  $\alpha$ -perceptron phase with at most  $8n^{3/2}$  updates for each new  $A_i$ .  
     If  $Ax \geq \alpha$  at any time, halt and return 1.
7. Return to the wiggling phase (step 3) and wiggle each updated  $v_i$  with respect to its corresponding matrix  $A_i$ .

Putting this into the scheme of Algorithm 2.1, we now have an algorithm that approximates  $\gamma^*$  with precision  $2^{-k}$  if we run through the algorithm  $k$  times. Furthermore we have the following estimate of the median time complexity, when we use the result of the worst case behaviour discussed in the previous chapter. Note that the extra factor of  $O(\sqrt{n})$  origins from the generalized perceptron:

**Theorem 9.4.1.** *In median time  $O(k^2 mn^{7/2})$ , the iterated perceptron-rescaling algorithm will approximate  $\gamma^*$  with precision  $2^{-k}$ .*

*Proof:* Using the calculations from the previous chapter it is easy to show that running through the Black Box once, giving answer 1, takes time at most  $O(mn^{7/2} \log \frac{1}{\gamma})$ , where  $\gamma = \gamma^* - \alpha$ . Now assume that the optimal margin is of size  $\gamma^*$  and that we run the Black Box  $k$  times. Letting  $\gamma^*$  vary with a uniform distribution in  $(0, 1]$  we get that the size of  $\gamma = \gamma^* - \alpha \geq 1/2^{k+1}$  in all  $k$  runs for at least half of all  $\gamma^*$ . This yields that our algorithm will terminate in median time  $O(k^2 mn^{7/2})$ .  $\square$

*Remark:* While the perceptron part of this algorithm is not suitable for investigating a negative margin (since adding a misclassified vector might decrease the inner product with a solution  $z$ ), it is possible to interpret the wiggle procedure in this case. You can say that an example  $a_i$  is misclassified by  $x_t$  when  $x_t \cdot a_i \leq \gamma$ , even if  $\gamma$  is negative. All estimates in the proofs above will still hold, but there is no help to get in the interaction with the perceptron part. The details of this result will not be presented in this thesis though.



## Chapter 10

# Conclusions

Now we are about to sum up the different topics and results of this thesis. They can be subdivided into two major categories: Graph Theory and The Perceptron Algorithm.

### 10.1 Graph Theory

Most of the results falling into this category concerns the subtopic extremal graph theory. A topic that search answers to questions like "When can we guarantee a graph  $G$  to have a property  $P$ ?"

#### Short Cycles in Transition Systems

Here we give upper and lower bounds on the number of edges needed in a transition system to assure us of having a cycle of length three or four compatible with the transition system. This is made for a natural family of transition systems.

#### The Erdős-Sós Conjecture

The conjecture states how many edges a graph  $G$  needs, to assure us of having all trees of a certain size  $k$  contained as a subgraph. Here we make an extension of Sidorenkos earlier results and use that to prove the conjecture true for every  $k \leq 9$  and for graphs with high minimum degree.

#### The Loebel-Komlós-Sós Conjecture

The conjecture states how large median degree a graph  $G$  needs, to assure us of having all trees of a certain size  $k$  contained as a subgraph. With help from results concerning the Erdős-Sós conjecture we conclude the conjecture to be true for every  $k \leq 7$ .

#### Properly Coloured Hamiltonian Path

This result shows that every edge-coloured complete graph not containing a monochromatic triangle contains a properly coloured (or alternating) hamiltonian path.

#### Image Segmentation using Graph Theory

Here we show how the max-flow – min-cut theorem can help us to segment an image in a proper and considerably fast manner.

## 10.2 The Perceptron Algorithm

The perceptron algorithm was developed by Dunagan and Vempala into the Modified perceptron algorithm. Here we take the modifications and developments a couple of steps further.

### The Geometry of the $n$ -dimensional sphere

To sort things out for the modified algorithm, we get proper estimates of the probability that two  $n$ -dimensional unit vectors have an inner product of at least  $1/\sqrt{n}$ .

### Making the Algorithm Deterministic

The algorithm made by Dunagan and Vempala had the drawback of not being a deterministic one. Here we show how modifications can establish this desired property.

### Speeding up the Algorithm

The modifications of making the algorithm deterministic made it a factor  $O(n)$  slower. Here we speed up the algorithm to a pace  $O(n)$  faster than the modified algorithm presented by Dunagan and Vempala.

Also we show that the expected time for the algorithm is another factor  $O(n)$  faster than the upper bound made on the worst case.

### Finding an Optimal Margin

Here the two main ingredients in the algorithm, the Perceptron Phase and the Wiggle Phase, are generalized in such a manner that we can construct an algorithm approximating an optimal margin instead of only finding a feasible solution.

# Bibliography

- [1] M. Ajtai, J. Komlós and E. Szemerédi, *On the Erdős-Sós conjecture – the dense case*, manuscript 1991.
- [2] N. Alon and A. Naor, Approximating the Cut-Norm via Grothendieck's Inequality, in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, Chicago (2004).
- [3] H. Ardö, *Personal communication*.
- [4] N. Alon and G. Gutin, Properly colored hamiltonian cycles in edge colored complete graphs. *Random Struct. Algorithms* **11**, 179–186 (1997).
- [5] A. Ambroladze, *Personal communication*.
- [6] M. Anthony and J. Shawe-Taylor, Using the Perceptron Algorithm to Find Consistent Hypotheses *Combinatorics, Probability and Computing* (1993) 2:pp. 385-387.
- [7] J. Bang-Jensen, G. Gutin and A. Yeo Properly coloured Hamiltonian paths in edge-coloured complete graphs, *Discrete Applied Mathematics* **83** (1998) 267-270.
- [8] M. Bankfalvi and Z. S. Bankfalvi, Alternating hamiltonian circuits in two-coloured complete graphs, in: *Theory of Graphs*, Proc. Coll., Tihany, 1966 (Academic Press, New York, 1968) 11–18.
- [9] O. Barr, On extremal graphs without compatible triangles or quadrilaterals, *Discrete Mathematics* **125** (1994) 31–43.
- [10] O. Barr, *Erdős-Sós Conjecture for Graphs with High Minimum Degree*, Research reports, No. 7 (1996), Umeå University, Sweden.
- [11] O. Barr, *Some Results in Extremal Graph Theory*, Research reports, No. 10 (1996), Umeå University, Sweden.
- [12] O. Barr, *A Note on the Loebel-Komlós-Sós Conjecture*, Research Reports, No. 14 (1997), Umeå University, Sweden.
- [13] O. Barr and R. Johansson, *Another Note on the Loebel-Komlós-Sós Conjecture*, Research reports, No 22 (1997), Umeå University, Sweden.

## BIBLIOGRAPHY

---

- [14] O. Barr, Properly Coloured Hamiltonian Paths in Edge-coloured Complete Graphs without Monochromatic Triangles, *Ars Combinatorica*, Vol. 50, (1998).
- [15] O. Barr and O. Wigelius, *New Estimates Correcting an Earlier Proof of the Perceptron Algorithm to be Polynomial*, ISSN 1403-9338, LUTFMA-5041-2004.
- [16] O. Barr, A Deterministic and Polynomial Modified Perceptron Algorithm *Computer Science Journal of Moldova*, Vol. 13, No. 3(39), 254–267, (2005).
- [17] C. Bazgan, H. Li and M. Woźniak, On the LoebL-Komlós-Sós Conjecture (1997), Preprint.
- [18] N. Biggs, Some heuristics for graph colourings, in: *Graph colourings*, (R. Nelson and R. J. Wilson, ed.) Longman Group UK Ltd., London, (1990), 87–96.
- [19] A. Blum and J. Dunagan, Smoothed Analysis of the Perceptron Algorithm for Linear Programming, in *SODA'02*, 2002; 905–914.
- [20] A. Blum, A. Frieze, R. Kannan and S. Vempala, A Polynomial-time Algorithm for Learning Noisy Linear Threshold Functions *Algorithmica*, **22**(1/2):35–52, 1997.
- [21] B. Bollobás, *Graph Theory* Springer, New York (1979).
- [22] B. Bollobás, The Chromatic Number of Random Graphs, *Combinatorica* **8** (1988), 49–55.
- [23] B. Bollobás and P. Erdős, Alternating hamiltonian cycles, *Isr. J. Math.* **23**, 126–131 (1976).
- [24] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, The Macmillian Press Ltd., New York (1976).
- [25] M. Borowiecki and P. Vaderlind, Erdős-Sós graphs contain all caterpillars with one leg, *Reports of Dept. of Math.*, University of Stockholm, **4** (1993).
- [26] Y. Boykov, O. Veksler and R. Zabih, “Fast Approximate Energy Minimization via Graph Cuts,” *ICCV*, 377–384, (1999).
- [27] S. Brandt, Subtrees and subforests of graphs, *J. Combin. Theory Ser. B* **61** (1994), 63–70.
- [28] S. Brandt, An Extremal Result for Subgraphs with few Edges, *J. Combin. Theory Ser. B*, **64**, (1995), 288–299.
- [29] S. Brandt and E. Dobson The Erdős-Sós Conjecture for Graphs of Girth 5, *Discrete Math.* **150** (1996), 411–414.

- 
- [30] C. C. Chen and D. E. Daykin, Graphs with hamiltonian cycles having adjacent lines different colours, *J. Combin. Theory Ser. B* **21** (1976) 135–139.
- [31] V. Chvátal, *Linear Programming*, W. H. Freeman and Company, New York, (1983).
- [32] N. Cristianini and J. Shawe-Taylor, *Support Vector Machines*, Cambridge, 2000.
- [33] D. E. Daykin, Graphs with cycles having adjacent lines different colours, *J. Combin. Theory Ser. B* **20** (1976) 149–152.
- [34] A. Dempster, M. Laird and D. Rubin, “Maximum likelihood from incomplete data via the EM algorithm” *J. R. Stat. Soc.*, 1977.
- [35] J. Dunagan and S. Vempala, *A Simple Polynomial-time Rescaling Algorithm for Solving Linear Programs*, ACM STOC 2004, 315–320.
- [36] E. Dobson, Some problems in extremal and algebraic graph theory, Ph.D. Dissertation, Louisiana State University, Baton Rouge, 1995.
- [37] P. Erdős, Extremal problems in graph theory, in: *Theory of Graphs and its Applications* (M. Fielder ed.), Academic Press, New York (1965), 29–36.
- [38] P. Erdős and T. Gallai, On maximal paths and circuit of graphs, *Acta Math. Acad. Sci. Hungar.* **10** (1959), 337–356.
- [39] A. Eriksson, O. Barr and K. Åström, Image Segmentation Using Minimal Graph Cuts, in: *Proceedings SSBA* (2006), 45–48.
- [40] H. Fleischner, Eulersche Linien und Kreisüberdeckungen, die vorgegebene Durchgänge in den Kanten vermeiden, *J. Combin. Theory Ser. B* **29** (1980) 145–167.
- [41] H. Fleischner, *Eulerian Graphs and Related Topics*, Vol. 1, Ann. Discrete Mathematics, Vol. 45 (Elsevier, Amsterdam, 1990).
- [42] T. Graepel, R. Herbrich, A. Kharechko and J. Shawe-Taylor Semidefinite Programming by Perceptron Learning, in S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems* 16. MIT Press, 2004.
- [43] G. R. Grimmet and C. J. H. McDiarmid, On Colouring Random Graphs, *Math. Proc. Cambridge Philos. Soc.* **77** (1975), 313–324.
- [44] R. J. Gould, Advances on the Hamiltonian Problem – A Survey, in *Graphs an Combinatorics*, **19**, No 1, 2003, 7–52.
- [45] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer Verlag, Berlin Heidelberg, 1988.

## BIBLIOGRAPHY

---

- [46] A. Gyárfás, E. Szemerédi and Zs. Tusa, Induced Subtrees in Graphs of Large Chromatic Number, *Discrete Math.* **30** (1980), 235–244.
- [47] T. Hellgren, *Minimum degree conditions giving a Hamiltonian cycle without forbidden transitions*, Doctoral Thesis, University of Stockholm, Sweden.
- [48] J. Komlós, G. N. Sárközy, E Szemerédi, Proof of a packing conjecture of Bollobás, AMS Conference on Discrete Mathematics, DeKalb, Illinois (1993), *Combinatorics, Probability and Computing* **4** (1995), 241–255.
- [49] J. Komlós and M. Simonovits, Szemerédi's Regularity Lemma and its Applications in Graph Theory, in: D. Miklós, V. T. Sós and T. Szőni (eds), *Combinatorics, Paul Erdős is Eighty*, Vol. 2, János Bolyai Mathematical Society, Budapest (1996), 295–352.
- [50] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, Massachusetts, 1984.
- [51] D. Miklós, V. T. Sós and T. Szőnyi, *Combinatorics, Paul Erdős is Eighty*, János Bolyai Mathematical Society (1996), Vol.2, 321–325.
- [52] M. Minsky and S. Papert, *Perceptrons*, (expanded edition), MIT Press, Cambridge, Massachusetts, 1988.
- [53] W. Moser and J. Pach, Recent developments in combinatorial geometry, in: *New trends in Discrete and Computational Geometry*, Springer, New York (1993).
- [54] M. E. Muller, A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres *Comm. Assoc. Comput. Mach.* **2**, 19–20, 1959.
- [55] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming* Studies in Applied Mathematics, Vol. 13, Philadelphia, 1994.
- [56] L. Redei, Ein kombinatorischer satz. *Acta Litt. Sci. Szeged*, 7 39–43, 1934.
- [57] I. Reiman, *Acta Math. Acad. Sci. Hung.* **9** (1959) 269–279.
- [58] F. Rosenblatt, *Principles of Neurodynamics*. Spartan Books, 1962.
- [59] J. F. Saclé and M. Woźniak *A note on the Erdős-Sós Conjecture for Graphs without  $C_4$* , Preprint No. 964 (1995), Université de Paris Sud.
- [60] N. Sauer and J. Spencer, Edge disjoint placement of graphs, *J. Combin. Theory Ser. B* **25** (1978), 207–215.
- [61] J. Shi and J. Malik, “Normalized Cuts and Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 888-905, 2000.

- 
- [62] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*, Springer-Verlag, Berlin, Heidelberg, 1985.
- [63] A. F. Sidorenko, Asymptotic solution for a new class of forbidden  $r$ -graphs, *Combinatorica* **9** (1985), 213–216.
- [64] P. J. Slater, S. K. Teo and H. P. Yap, Packing a tree with a graph of the same size, *J. Graph Theory* **9** (1985), 207–215.
- [65] D. Spielman and S. Teng, Smoothed Analysis of Termination of Linear Programming Algorithms, in *Mathematical Programming, Series B*, Vol. **97**, 2003.
- [66] D. Spielman and S. Teng, Smoothed Analysis: Why The Simplex Algorithm Usually Takes Polynomial Time, in *Proc. of the 33rd ACM Symposium on the Theory of Computing*, 296–305, 2001.
- [67] D. P. Sumner, Subtrees of a graph and the Chromatic Number in: *The Theory and Applications of Graphs, Fourth International Conference, Western Michigan University* (ed. Chartrand, Alavi, Goldsmith, Lesniak-Foster and Lick), John Wiley & Sons, New York (1981), 557–576.
- [68] L. Vandenberghe and S. Boyd “Semidefinite programming,” *SIAM Review*, 38(1): 49–95, March 1996.
- [69] M. Woźniak, On the Erdős-Sós Conjecture, *J. Graph Theory*, **21**, (1996), 229–234.
- [70] M. Woźniak, Packing of graphs, in *Dissertationes Math. (Rozprawy Mat.)* **362**, (1997).
- [71] H. P. Yap, Packing of Graphs – a survey, *Discrete Math.* **72** (1988), 395–404.
- [72] Y. Zhao, *Proof of the  $(n/2 - n/2 - n/2)$  Conjecture for large  $n$ .*, Submitted.
- [73] Zhou Bing, A note on Erdős-Sós conjecture, *Acta Math. Scientia*, **4** (3) (1984), 287–289.