

POLITECNICO DI MILANO
Master Degree in Computer Science and Engineering
Electronic, Information and Bioengineering
Department



Breaking Down Group Testing


A MaxSAT Based Framework with Theoretical Perspectives

AIR Lab
Artificial Intelligence
and Robotics Laboratory

Author:
Lorenzo Ciampiconi
SN 899140

Supervisor: Prof. Nicola Gatti
External Supervisor: Prof. Kuldeep Meel¹

Academic Year 2018-2019

¹National University of Singapore 

A mia madre Martina e mio padre Francesco...

Abstract

The success of MaxSAT (Maximum Satisfiability) solving in recent years has motivated researchers to apply MaxSAT solvers in diverse discrete combinatorial optimization problems. Group testing has been studied as a combinatorial optimization problem, where the goal is to find defective items among a set of items by performing tests on pools of items. In this thesis work, we propose a MaxSAT-based framework, called **MGT**, that solves the decoding phase of non-adaptive group testing. Such approach is motivated by past studies over the optimality of the so called **Smallest Satisfying Set**. We encode such formulation with partially weighted **MaxSAT** and extend it to the noisy variant of group testing. We prove equivalence of a compact MaxSAT-based encoding and derive guarantees of optimality of its tuning. We perform extensive experimental evaluation which shows that **MGT** grants unprecedented scalability in terms of number of items handled. Furthermore, **MGT** performance are impressive compared to all the other algorithms implemented as test bed. We leave a wide open window on theoretical perspective trying to give an interpretation to the observed phase transition with respect to the number of tests with an easy-hard-easy behaviour. This kind of transition has been already detected, but never deeply understood, in other artificial intelligence problem such as random k -SAT.

Abstract

Il successo nello sviluppo di risolutori per il problema MaxSAT (massima soddisfacibilità) negli ultimi anni ha motivato i ricercatori ad applicare tali codifiche, e quindi i relativi algoritmi, in diversi problemi discreti di ottimizzazione combinatoria. Il test di gruppo (Group Testing) è stato studiato come un problema di ottimizzazione combinatoria, in cui l'obiettivo è quello di trovare, data una serie di elementi, coloro i quali presentano una certa caratteristica, in generale chiamata difettosità, eseguendo test su sottogruppi di essi. In questo lavoro di tesi, proponiamo un framework basato su MaxSAT, chiamato MGT, che computa una soluzione per la fase di decodifica dei test in un contesto non adattativo. Tale approccio è motivato da studi precedenti sull'ottimalità del cosiddetto **Small Satisfying Set** (minor insieme di soddisfacibilità). Codifichiamo la citata formulazione attraverso una particolare versione di MaxSAT (Partially Weighted) e la estendiamo con una variante in grado di considerare e gestire errori nei risultati dei test. Dimostriamo l'equivalenza di due codifiche basate su MaxSAT, la prima triviale, data la formulazione del problema di ottimizzazione, e la seconda compatta, in termini di clausole generate, ottenendone garanzie di ottimalità. A partire da una valutazione sperimentale approfondita mostriamo che MGT offre una scalabilità senza precedenti in termini di numero di elementi considerabili nel problema. Inoltre le prestazioni di MGT sono impressionanti rispetto a tutti gli altri algoritmi implementati come banco di prova. Per concludere delineiamo una prospettiva teorica cercando di dare un'interpretazione alla

transizione di fase osservata, riguardante la probabilità di successo e il comportamento del problema in termini di complessità temporale come Facile-Difficile-Facile (in letteratura esattamente definita Easy-Hard-Easy), con dipendenza dal numero di test. Questo tipo di transizione è già stata osservata sperimentalmente, ma mai profondamente compresa, in altri problemi di intelligenza artificiale come la versione randomica di k -SAT.

Acknowledgments

I want to thank all the people with whom I spent my time, had discussions and shared experiences during those five years which this thesis is a result of. I'm deeply grateful to my hometown friends Andrea, Luca, Alex and Joseph for their friendship that has been supporting me throughout my studying and has been regenerating during the pauses back home. I want to thank my dearest friends from university Lorenzo, Alessandro, Simone and Mario (Mauro, Marione, Bobina) for the time, the suffering and the joy we spent together. I want to thank Enrico and his way of being curious all the time about what I have been studying, I keep and value all of our discussions where we shared the topics of our research and study which has funnily substituted partnering in a progressive metal band. I owe my deepest gratitude to my college friends Giulia, Elisa and Matteo as they have been a life loadstar, the deepest pool where to sink my doubts, bad feelings and a constant source of great moments in life. I would like to express my gratitude to Professor Kuldeep Meel which acquaintance has changed and determined my academic path. He has given me several unexpected chance as his student and has contributed to many achievements during the last two years of my career

culminated with the publication for AAAI2020, his dedication and passion for teaching are unmatched. I also want to thank Doc. Bishwamittra Ghosh and Professor Jonathan Scarlett for the effort and the support during the work on the publication. I also want to thank Prof. Nicola Gatti for accepting to co-supervise me during this final step of my master degree. I want to thank my bandmates Leoluca and Francesco, my time with you has been crucial to fuel me and I wish long life and tons of gigs to our music projects. I am particularly grateful to Daniele which friendship has enriched my life as friend, as a music listener, as a politic expert (well you can put whatever you want as probably we have discussed about it) since I have known him more than ten years ago and some of our discussions has for sure inspired me on understanding the problems I was working on. During those years I have greatly benefited from the time with my friend Guido, he has been a constant partner in study since the first year at Politecnico and my intellectual dept to him is priceless. I want to thank my roommates Andrea and Rachid for being my second family during the last years, I feel impossible to describe what has been living with you for all this time, my ride towards graduation has been made special by you, just Thank You. I want to thank my grandmothers Laura and Anna for their unconditional love. I want to thank my grandfather Angelo for showing every day me what hard work is. I want to thank my uncle Massimo for his sharp observations and the passionate discussions we had all over the years. I want to thank my sister Elena which pure and soulful love has been warming every time I was back home. I want to thank my brother Federico for his love, his unmatched curiosity and passion for science that has inspired me throughout

my journey. Finally I want to thank my mother Martina and my father Francesco without whom this achievement would have been, without a doubt, impossible. Thank you Mother for the constant understanding and the teaching on how to appreciate my achievements. Thank you Father for teaching me how to be a curious human being. Thank you both for constantly support my passions.

*Pasceva nell'orto ignorando la vita
Una vita nascosta che sotto cresceva
Ignorando il perchè
Non ci fosse ragione*

*Perchè la vita non fosse
Perchè non pensasse
Perchè l'intelletto neppure visesse
Perchè l'intelletto neppure vivrà*

Nomenclature

Algorithm for Group Testing Symbols

$\widehat{\mathcal{K}}_{\text{COMP}}$ Estimate given by SCOMP

$\widehat{\mathcal{K}}_{\text{DD}}$ Estimate given by DD

$\widehat{\mathcal{K}}$ Estimate of a defective set

\mathcal{K} Defective set

$\widehat{\mathcal{K}}_{\text{LP}}$ Estimate given by LP

$\widehat{\mathcal{K}}_{\text{SCOMP}}$ Estimate given by SCOMP

$\widehat{\mathcal{K}}_{\text{SSS}}$ Smallest satisfying set

DD DD algorithm

LP Relaxed-LP algorithm

P_s Probability of success of an algorithm

\widehat{P}_s Unbiased estimate of probability of success of an algorithm

SCOMP SCOMP algorithm

Group Testing Symbols

ξ	Vector of boolean noise
\mathbf{A}	Matrix of pooled measurements
\mathbf{x}	Boolean vector representing the items
\mathbf{y}	Vector of outcomes of tests
k	Number of faulty items
$k(\mathbf{x})$	Number of faulty items of the vector \mathbf{x}
m	Number of tests
n	Number of items
x_i	i -th items
y_i	i -th outcomes

Logical Operator and Symbols

\models	Is model operator
\oplus	Logical XOR
σ	Logical assignment to boolean variables
\vee	Logical OR
\wedge	Logical AND
$\text{wt}(\sigma)$	Weight of the assignment (the formula F is implied by the context)

$\text{wt}(C_i)$ Weight of the i -th clause

C_i i -th CNF clause

F Propositional Formula

CNF Conjunctive Normal Form

Other Symbols

MaxHS MaxHS solver

MaxSAT Maximum-Satisfiability problem

MGT MaxSAT for Group Testing implemented algorithm

$\mathbb{P}(\text{SAT}|X)$ Probability of SAT of a random k -SAT instance X

SAT Satisfiability problem

Contents

1	Introduction	1
1.1	Research Area and Motivation	1
1.2	Work and Contribution	3
1.3	Thesis Structure	6
2	Grounding Concepts	9
2.1	Introduction to Group Testing	10
2.1.1	An Intuitive Example of Group Testing .	11
2.1.2	Adaptive and Non-Adaptive Group Testing	13
2.2	Non-Adaptive Group Testing	16
2.2.1	Design Phase	17
2.2.2	Decoding Phase	22
2.2.3	Error Tolerant Model: Noiseless and Noisy Testing	23
2.3	Application of Group Testing	25
2.3.1	Biology	26
2.3.2	Communications	27
2.3.3	Information Technology	27
2.3.4	Computer Science	28

2.4	Maximum Satisfiability	29
2.4.1	Mathematical formulation	30
2.4.2	Solving Techniques	31
2.4.3	Applications	32
2.5	Phase Transition in Computer Science	32
2.5.1	SAT Phase Transition	33
3	Problem Formulation	37
3.1	Notation and Preliminaries	37
3.1.1	Boolean Logic and CNF	38
3.1.2	MaxSAT	39
3.1.3	Group Testing	40
3.1.4	Stochastic Models	42
3.1.5	Group Testing Bounds	43
3.2	Non-Adaptive Group Testing Decoding Phase . .	46
3.2.1	Compressed Sensing and $\widehat{\mathcal{K}}_{SSS}$	47
3.2.2	Related Decoding Algorithms	47
3.2.3	Optimal Solution: SSS	52
3.2.4	Noisy Extension	54
4	MaxSAT Encoding and Optimality	57
4.1	MaxSAT Encoding	58
4.1.1	Noiseless Setting:	58
4.1.2	Noisy Setting:	59
4.2	A Compact Encoding for Noisy Setting	61
4.2.1	Intuition and Formulation	61
4.2.2	Equivalence of Encodings	62
4.3	Optimality of $\widehat{\mathcal{K}}_{SSS}$ and λ Trade-Off	65
4.3.1	Overview	66
4.3.2	Background	66

4.3.3	Derivation	67
5	Experimental Results and Evaluation	69
5.1	Experiment Preliminaries	69
5.1.1	About the Implementation of Algorithms	70
5.1.2	Setup of Instances	71
5.1.3	Evaluation Metrics	72
5.1.4	Question Answered	73
5.1.5	Adoption of Bounds	73
5.2	Empirical Results on Accuracy	74
5.2.1	Varying the Number of Items:	74
5.2.2	Varying the Number of Defective Items: .	77
5.2.3	Scalability Analysis:	77
5.2.4	Efficiency of the Compact Encoding: . . .	79
5.3	Runtime Analysis and Phase Transition	79
5.3.1	Noiseless Phase Transition	82
5.3.2	Understanding Runtime Behaviour and \hat{C} Estimate	84
6	Conclusions and Future Research Direction	91
	Bibliography	95
A	Proof and Derivations	113
A.1	NP-Hardness of $\hat{\mathcal{K}}_{SSS}$	113
A.1.1	What is Vertex Cover:	114
A.1.2	Proof	114
A.2	Concepts and Derivation of C-Estimate	116
A.2.1	Probability of Picking a Masked Defective	117
A.2.2	From \mathbb{P}_{md} to \hat{C}_A	120

B Additional Plot

123

List of Important Theorems and Definitions

2.2.3	Definition (Symmetric Noise)	24
2.5.1	Definition (Random k -SAT)	34
3.1.6	Definition (MaxSAT)	39
3.1.7	Definition (Items Vector)	40
3.1.8	Definition (Pooling Matrix)	40
3.1.9	Definition (Outcomes Vector)	40
3.1.10	Definition (Boolean Noise Vector)	40
3.1.11	Definition (Defective Set)	41
3.1.12	Definition (Satisfying Set)	41
3.1.13	Definition (Decoding Algorithm)	41
3.1.15	Definition (Masked Defective)	41
3.1.16	Definition (Success Probability)	43
3.1.1	Theorem (Counting Bound)	44
3.1.17	Definition (Group Testing Rate)	45
3.1.18	Definition (Achievable Rate)	45
3.1.2	Theorem (Noisy Group Testing Bound)	46

LIST OF IMPORTANT THEOREMS AND DEFINITIONS

XVI

3.2.2 Definition (Unexplained Test)	51
3.2.3 Theorem (\mathcal{NP} -Hardness of $\widehat{\mathcal{K}}_{SSS}$)	52
4.1.1 Construction (Mapping of MaxSAT to $\widehat{\mathcal{K}}_{SSS}$) . .	59
4.1.2 Construction (Mapping of MaxSAT to $\widehat{\mathcal{K}}_{SSS}$ in Noisy Settings)	61
4.2.2 Theorem (Equivalence of XOR and Compact En- coding)	63
5.3.1 Conjecture (Hardness Threshold)	86
5.3.1 Definition (Masked Defective Probability)	87
5.3.2 Definition (C-Estimate)	88

List of Figures

2.1.1	Group Testing over Defective Bulb Problem . . .	12
2.1.2	Adaptive Group Testing and Binary Search . . .	13
2.1.3	Binary Tree Representation of a General Sequential Algorithm	15
2.5.1	Phase Transition in Random k-SAT	34
3.1.1	Combinatorial Group Testing Instance	42
5.2.1	Accuracy for Implemented Algorithms	75
5.2.2	Qualitative Accuracy Trend Variation with respect to n	76
5.2.3	Qualitative Trend Variation with respect to β	78
5.2.4	Accuracy Trend for MGT with Large Instances	80
5.2.5	XOR vs Compact Runtime	81
5.3.1	MGT and LP Noiseless Phase Transition	82
5.3.2	Phase Transition of Runtime and \mathbb{P}_s overlapped for MGT	83
5.3.3	Noiseless Runtime Phase Transition for all Algorithms	86
5.3.4	\widehat{C}_A Estimation Performances	89

A.2.1 Probability of Masked Defective	118
A.2.2 \hat{C}_A over all Algorithm	121
B.0.1 Scalability in Noiseless settings 1	124
B.0.2 Scalability in Noiseless settings 2	125
B.0.3 Runtime Trend in Noisy Settings	126
B.0.4 Easy-Hard-Easy Phase Transitions for MGT, LP, COMP, DD, SCOMP	127
B.0.5 Easy-Hard-Easy Phase Transition in Noiseless Settings for MGT and LP	128
B.0.6 Runtime Trend in Noisy Settings	129

Chapter 1

Introduction

“Bown: Prepare G-pod for EVA, Hal. Made radio contact with him yet?”

Hal: The radio is still dead.

Bown: Do you have a positive track on him?”

Hal: Yes. I have a good track.

Bown: Do you know what happened?”

Hal: I’m sorry Dave. I don’t have enough information.”

2001: A Space Odyssey

1.1 Research Area and Motivation

The transition to the new millennium has coincided with an heavy adoption of computer techniques to all sorts of applications and problems of real life in the technical, industrial, medical and economic fields. While computer science uncontrollably permeates our society, life and activities, a new frontier of application and

automation has been revealed in the last decade: Artificial Intelligence. Although artificial intelligence destroys and reinvents the classic role of the programmer and computer engineer, in parallel it fills computer science with new philosophical meaning and in particular renew an even closer relationship with mathematics, just think of the role of statistics in machine learning or logic in reasoning. In this sense, mathematical problems, which were formalized before artificial intelligence took hold, now acquire new importance and the careful study of their formalization and resolution techniques is due. In particular the study of the relation between each problem and their reducibility can lead to the exploitation of techniques developed for a specific one to solve other, apparently unrelated, ones by finding a feasible reduction. Group Testing is a mathematical problem, formalized during the Second World War, to which innumerable real problems coming from the most disparate fields are reduced and the research on its algorithmic solvability has had a rekindle in the last decade, in particular after the Human Genome Project. The success of MaxSAT (maximum satisfiability) solving in recent years has motivated researchers to apply MaxSAT solvers in diverse discrete combinatorial optimization problems. Group testing has been studied as a combinatorial optimization problem, where the goal is to find defective items among a set of items by performing tests on pool of items. The purpose of this work is to investigate non-adaptive Combinatorial Group Testing and to seek a valid non-approximated solution to its decoding phase.

1.2 Work and Contribution

Given a large set of items containing a subset of defective item(s), the problem of group testing concerns the design and evaluation of tests on pools of items (a pool is a selected subset of all items) to identify the defective item(s). A test is positive if the pool contains at least one defective item(s) and negative if it contains no defective item. Group testing can be viewed as a sparse inference problem with a combinatorial flavor with diverse applications, for example, clone screening [BT97], multichannel access in high speed computer networks [BHKK92], medical examination [Dor43], statistics [HHW81], compressed sensing and machine learning [AS12].

Group Testing can be approached in two ways: adaptive and non-adaptive [AJS19]. In adaptive testing, the test pools are designed sequentially, wherein each test depends on the outcome of the previous ones and each outcome drives the designs of the following pools to be tested. In the case of non-adaptive testing, all test pools are designed in advance, which allows parallel implementation. Although adaptive testing allows more freedom in the design, it does not improve upon non-adaptive testing by more than a constant factor in the number of required tests [AS12]. The group testing problem, in the non-adaptive variant, which is studied in this work, can be separated into two phases: *design* and *decoding* [AJS19]. The design phase in which the testing strategy is chosen, that is, which items to place in which pools. The decoding problem consists of determining which items are defective given the set of tests and their outcomes, and can be formulated as a combinatorial optimization problem [DHH00, MM12, ABJ14].

In a more practical setting, a noisy variant to group testing is considered where tests are inverted/flipped according to some specific random model or in an adversarial manner [AJS19]. In this work, we pursue the direction of solving the decoding phase in non-adaptive group testing in both noiseless and noisy settings by introducing a novel MaxSAT-based approach.

Robert Dorfman formalized group testing during World War II with the purpose of creating a model to pool sick soldiers without direct testing of every single candidate [Dor43]. Thus, group testing can be viewed as a pooling strategic problem [ZKMZ13] with the goal of designing an optimal set of tests of items efficiently such that the test results contain enough information to determine a small subset of defective items. Another body of work focuses on the decoding of the pooling results. In particular, Chan et al. [CCJS11, CJS14, MM12] find a similarity between compressed sensing [AAS16, Mal13a] and group testing, and present the COMP (combinatorial orthogonal matching pursuit) algorithm for the decoding phase of non-adaptive group testing in both noiseless and noisy settings. Aldridge et al. [ABJ14] consider non-adaptive noiseless group testing and propose two algorithms: DD (definite defectives) and SCOMP (sequential COMP), which require stronger evidence to declare an item defective; and an essentially optimal but computationally difficult SSS (smallest satisfying set) algorithm. Aldridge has shown the strict suboptimality of DD algorithm in case less than 0.41 of the items are defective, its optimality in case more than the half are defective and the optimality of SSS in case less than half of the items are defective [Ald17], this has been shown in the bernoulli non adaptive testing. The recent works [SC16, COGHKL19] have

established performance guarantees for similar computationally expensive algorithms as SSS. In addition, linear programming (LP) algorithms have been proposed to approximate the SSS algorithm with practical runtime [MM12, MS09, CJSA14].

The maximum satisfiability (MaxSAT) problem is an optimization analogue to the SAT (satisfiability) problem. MaxSAT is complete for the class FP^{NP} , which includes many practical optimization problems. The added scalability and improvement of MaxSAT solvers in recent years have encouraged researchers to reduce several optimization problems into MaxSAT, for example, optimal planning [RGPS10, ZB12], interpretable rule-based classifications in machine learning [GM19, MM18], automotive configuration [WZK13], data analysis and machine learning [BHJ15], automatic test pattern generation for cancer therapy [LK12], etc. However, to the best of my knowledge, there is no prior work on group testing that takes benefit from MaxSAT-based solution approach.

In this work Max-SAT encoding for the Group Testing decoding phase is identified in both noiseless and noisy settings, for the latter two encoding are proposed, a trivial one and one exploiting the objective function to dramatically reduce the logical constraints, they have been compared and their logical equivalence has been proved. The proposed logical models has been implemented using a Python based framework relying on a flexible MaxSAT solver (MaxHS) [DB11]. An extensive experimental evaluation has been performed comparing accuracy with state of the art Linear Programming algorithm [MM12] and with COMP, DD, SCOMP algorithms [ABJ14]; all those experiments has shown great accuracy performance of the proposed implementation and

impressive scalability as our model has been able to handle an unprecedented number of variables. Furthermore the performance in terms of time complexity of the solver has been studied which has revealed a phase transition behaviour similar to the one already observed in random k-SAT [MSL92] that widely open a door on the relation and difference between combinatorial problem, constraint satisfaction and optimization problems; in particular in future it is mandatory to investigate on the potential bridge between group testing and MaxSAT and such study will probably be useful for the improvement of MaxSAT solving. A paper version of this work has been submitted and approved for publication to AAAI¹ and will be presented at the next conference edition in february 2020.

1.3 Thesis Structure

- **Chapter 2, Grounding Concepts:** We present all the elements relevant to the understanding of this work. In particular we give an introduction to the Group Testing problem, we mark out some historical references and we illustrate the involved reasoning and techniques trying to get the reader used with Group Testing concepts. We also present MaxSAT and have a brief discussion over phase transitions in computer science.
- **Chapter 3, Problem Formulation:** The beginning of this chapter contains the notation and preliminaries necessary to formalize our work, in particular it provides defini-

¹Association for the Advancement of Artificial Intelligence

tion used by the community and theoretical results useful to give solid grounds to our work and understand experimental evaluation. We depict and discuss some state of the art algorithm that has been implemented to test our framework. Finally we unfold the optimization formulation of our solution.

- **Chapter 4, MaxSAT Encoding and Optimality:** We show the essential MaxSAT encoding, we extend it to the noisy settings and portray the insight that lead us to the design and formalization of a compact encoding. Furthermore we mathematically describe the essential optimality of our model under the assumption of Bernoulli testing and give a theoretically grounded tuning of its parameter.
- **Chapter 5, Experimental Results and Evaluation:** We illustrate the setup of our experiments in terms of implementations, problem instances, evaluations metrics and adopted bounds. We start with a study over the accuracy of the framework and analyze its behavior in order to delineate dependency from the problem parameters (number of items, number of faulty, noise etc). We then study the computation effort of the algorithms, we try to interpret the behaviour and we picture a theoretical perspective following the observed phase transition.
- **Chapter 6, Conclusion and Future Research Direction:** We summarize both the theoretical and the empirical results. We set forth all the possible future directions that we aim to pursue and we outline the novel open problems that we are proposing to the computer science community.

‘

Chapter 2

Grounding Concepts

*“Strangers passing in the street
By chance two separate glances meet
And I am you and what I see is me
And do I take you by the hand
And lead you through the land
And help me understand
The best I can”*

Pink Floyd, Echoes

The goal of this chapter is to introduce the reader to the mathematical and logical problems we are dealing with, mainly Group Testing. As those mathematical environment are rich of formalism and theoretical results we try to give an intuitive and brief introduction to them to picture a significant overview of the main implication and issues that arise when trying to solve them, leaving, when not strictly needed, the heavy formalism to the concrete contribution of our work. Furthermore this chapter

present real world application of the studied topic and tries to make the reader understanding its importance and why it is crucial.

2.1 Introduction to Group Testing

Robert Dorfman, in his seminal paper of 1943 [Dor43], published in the *Annals of Mathematics Statistics*, presented the Group Testing problem. The main contribution of his work is to describe a statistically grounded method to reduce the tediousness of the testing work and to make more elaborate and more sensitive inspections economically feasible. The context of his publication was the Second World War when the American army was encountering the spread of the syphilis among his battalions. The United States Public Health Service was able to test a single candidate with good accuracy, by the so-called "Wasserman-type" blood test, but the large size of the population made unpractical to test directly every single soldier because of the cost of each test. They commissioned Dorfman to work out a method to dramatically reduce the number of tests required to weed out all syphilitic men called up for induction. So Dorfman made the following claim:

“The germ of the proposed technique is revealed by the following possibility. Suppose that after the individual blood sera are drawn they are pooled in groups of, say, five and that the groups rather than the individual sera are subjected to chemical analysis. If none of the five sera contributing to the pool contains syphilitic antigen, the pool will not contain it either and will test negative. If, however, one or more of the sera contain syphilitic antigen, the pool will contain it also and the group test will reveal its pres-

ence. The individuals making up the pool must then be retested to determine which of the members are infected". From the above reasoning Group Testing was born. Dorfman was suggesting to compress the information through the testing phase in order to reduce the number of test (from n , where n is the number of candidates) required to discover the uncompressed one. The first step represents the higher degree of compression; when further testing phases are conducted the degree of compression reduces until the original information is recovered. While in computer science the advantage of compression is likely to be on the reduced size of the information¹, here is on the number of tests performed, so Group Testing is useful when performing a test is really expensive and the syphilis problem is the first practical problem which was attacked with this technique.

2.1.1 An Intuitive Example of Group Testing

To ease the reader understanding of the general problem we provide another simple practical example. Imagine to have a large set (size n) of electrical bulb which are connected in series and imagine that at least one bulb is faulty, so none of the bulbs will light up as the series is broken. In this case we are pooling all the bulb in a single group and performing a single test instead of n obtaining the information that at least 1 bulb is broken or, using the common terminology of the field, *defective*. At this point we want to identify precisely which are the defectives one(s). We can

¹To keep track of the belonging of item to each pool and the results of each test the information needed can be larger compared to encoding the original one. Information on how to encode a group testing instance can be seen in Section 3.1.3

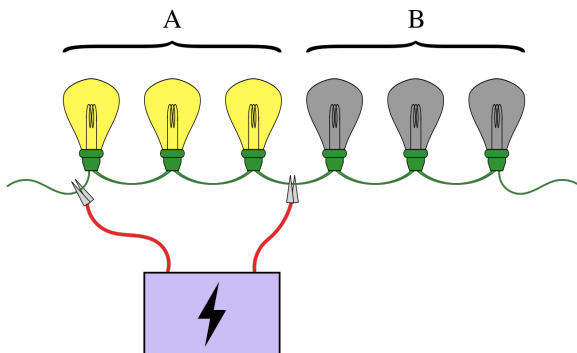


Figure 2.1.1: Group testing applied to the electric bulb problem

test each bulb individually, performing n tests or we can design pools in which we connect in series a subset² (size \tilde{n}) of bulbs. In case the series light up (negative outcome) we are ruling out \tilde{n} item, with certainty about their non-defectiveness. In case the series does not light up (positive outcome) it reveals that at least one of the \tilde{n} item is defective and we have to consider other subset of the pools to identify the broken bulb(s). This procedure to dynamically design pools coincide with adaptive testing previously mentioned in Section 1 and more deeply presented in Section 2.1.2. Note that if we have the information of the precise number of faulty item k and $k = 1$, the procedure become a divide et impera algorithm really similar to a dichotomic search and, if we proceed dividing by two the size of each pools, the number of tests needed to exactly identify the faulty one is exactly $\log_2(n)$.

²The subsets are the "pools" previously mentioned by Dorfman

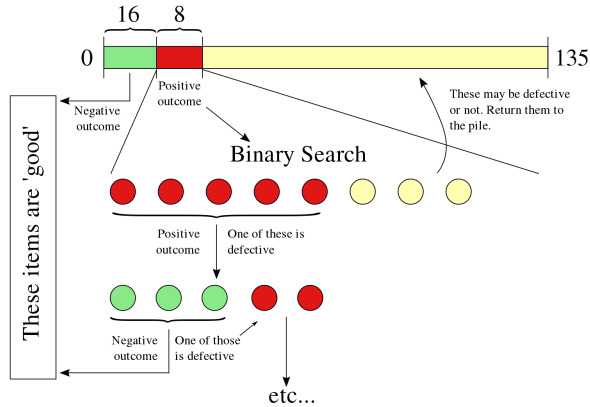


Figure 2.1.2: Adaptive Group Testing and binary search

2.1.2 Adaptive and Non-Adaptive Group Testing

Group testing, as anticipated in the introduction, can be approached in two different ways: adaptive and non-adaptive. In an adaptive algorithm, the tests are conducted one by one, and the outcomes of previous tests are assumed known at the time of determining the next pool to test. In a non-adaptive algorithm, no such information is available in determining a test. Thus, it can be interpreted as if all tests are specified simultaneously, although in practice the real reason can be the collection of such information is too time-consuming or too costly. A compromise between the two types of algorithms is realized by the so-called multistage algorithm in which tests are divided into several stages where the stages are considered sequential but all tests in the same stage are treated as nonadaptive. The early history of group testing was all based on sequential algorithms even though,

in the 90's, the focus has been shifting due to the general influence of the popularity of parallel computing, and also due to the need of nonadaptive algorithms in the application of clone library screening.

Adaptive Group Testing

Adaptive group testing can be seen as a recursive procedure in which a first design of pool is performed, and then, by the information retrieved from the tests the next pools are designed, and so on and so forth until the solution has been reconstructed. In practice a dynamic reasoning is performed over the partial information retrieved step by step. Since a sequential algorithm possesses more information at testing, clearly, it requires fewer number of tests than a nonadaptive algorithm in general; so the main advantage of adaptive group testing is the reduction of number of tests required to obtain a solution with no misclassified items. It is worth mentioning that it has been proved that this reduction is negligible as the number of tests reduces only by a constant factor with respect of number of items [AS12].

There are several formalized adaptive algorithm that extends the work proposed by Dorfman, one of the two most important is Li's s -stage algorithm [Li62], which extends the Dorfman procedure from 2 to s stages. The other one is Hwang's Generalized Binary Splitting algorithm which apply the concept of dichotomic search to Group Testing. Hwang [Hwa72] suggested a way to coordinate the applications of binary splitting such that the total number of tests can be reduced. The idea is, roughly, that there exists in average a defective in every $\frac{n}{k}$ items, where k is the number of defectives. Instead of catching a contaminated group of

Algorithm 1 General Adaptive Algorithm

```

1: procedure ADAPTIVEGROUPTESTING( $S$ )
2:    $H = \emptyset$  ▷ set of Healty items
3:    $F = \emptyset$  ▷ set of Faulty items
4:   if TESTPOOL( $S$ ) = True then
5:     if  $|S| = 1$  then
6:        $F \leftarrow S$ 
7:     else
8:        $listofpools \leftarrow$  DESIGNPOOLS( $S$ )
9:       for all  $P \in listofpools$  do
10:         $H_i, F_i \leftarrow$  ADAPTIVEGROUPTESTING( $P$ )
11:         $H \leftarrow H \cup H_i$ 
12:         $F \leftarrow F \cup F_i$ 
13:     else
14:        $H \leftarrow S$ 
15:   return  $H, F$ 

```

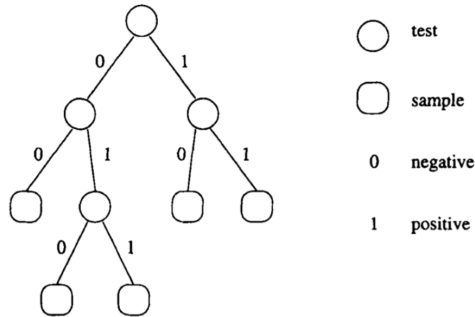


Figure 2.1.3: Binary Tree Representation of a General Sequential Algorithm

size about half of the original group, which is the spirit of binary splitting, one could expect to catch a much smaller contaminated group and thus to identify a defective therein in fewer number of tests. There are other algorithms such as nested class algorithms, merging algorithms [HD73], competitive algorithms [MA14] and doing more than cite them is beyond the scope of this section.

Historically, the main goal of group testing is to minimize the number of tests. Therefore sequential algorithms have dominated the literature. Note that for the (n, k) model, the asymptotic information-theoretical lower bound is (that will be presented later on in Section 3.1.5):

$$\log_2 \binom{n}{k} \approx k \log_2 \left(\frac{n}{k} \right) \quad (2.1.1)$$

Since we can use the binary splitting algorithm to identify a positive item in at most $\log_2(n)$ tests, $k \cdot \log_2(n)$ tests suffice for the (n, k) model. By the closeness of $k \cdot \log_2(n)$ with the information bound, we can say the (n, k) model is practically solved [AJS19]. On the other hand, the determination of exact optimal (n, k) algorithm is very difficult.

2.2 Non-Adaptive Group Testing

A group testing algorithm is non-adaptive in the sense that it does not adapt to the outcomes of previously performed tests, the constraint is that all tests must be specified without knowing the outcomes of other tests. But why are nonadaptive algorithms necessary? There are two scenarios. The first comes from a time constraint: all tests must be conducted simultaneously.

The second comes from a cost constraint: the cost of obtaining information on other tests could be prohibitive. A mathematical study of nonadaptive Combinatorial Group Testing algorithms does not distinguish these two causes. The main characteristic of a non-adaptive algorithm is that two phases are distinguished: *design* phase and *decoding* phase. There is implicitly a third phase, in between the two, which is the testing phase, but, as an algorithm is not in charge of such phase, it is not considered in the mathematical study and the tests outcomes are considered an implicit result of the design phase.

In order to introduce the design phase we give a brief definition of the representation of the design in a combinatorial fashion. In combinatorial group testing the pools are represented by a $m \times n$ boolean matrix \mathbf{A} (where m is the number of tests and n is the number of items). Each A_{ij} cell of the matrix represent the belonging of the j -th item to the i -th test, so $A_{ij} = 1$ if and only if the j -th item belongs to the pool of the i -th test.

2.2.1 Design Phase

The design phase is necessarily the first to be performed and is based on grouping all the candidates item in pools. This passage of the algorithm is the one that take care of the requirements from the application scenario, the cost-constraint and so the number of tests that can be performed. In general the design phase must compute a matrix \mathbf{A} that has the best trade-off between the number of tests and the quality of the information encoded after the collection of outcomes. The requirements from specific scenarios can vary a lot and we try to introduce the design phase in the more general way as possible.

The first detail to note, that distinguish the design of pools from adaptive testing, is that pools must be non-disjoint in order to retrieve relevant information and to have chance to recover the exact *defectiveness* of each item. Let's consider the opposite for once and let's have disjoint pools. If you know the exact number of faulty items (k) in your set, then you can feel lucky enough to design at least k unit pool and try your chance to get in those exactly the defective ones, in order to get an exact ³ information, and this is just a not so smart or statistically reasonable idea. What we get from this trivial observation is that pools must share some of their elements and we are looking for a sharp way to overlap pools in order to obtain the most information from a number of tests $m \ll n$.

So how to design pools with no, even partial, information about which are the most promising candidates? The techniques are divided in two main groups: deterministic design and random design.

Deterministic Design

The key point of a deterministic design is to obtain pools which satisfy some property that represents minimum requirements for **A** to be able to identify the k elements. Those property are called k -separability and k -disjunctness.

³Remember that a positive test tells you that at least one item in the pool is faulty, so the only chance to exactly identify the faulty ones, in case the pools are disjoint and you can not rule out other member of a positive pool with information from other tests, is that the positive pool is a unit pool, so it contains only one item.

Definition 2.2.1. Consider a testing matrix $\mathbf{A} \in \{0, 1\}^{m \times n}$, we write X for the set of all items:

\mathbf{A} is called k -disjunct if, for all subset $\Upsilon \subset X$ with $|\Upsilon| \leq k$:

$$\begin{aligned} \forall i \in \Upsilon^c, \forall j \in \Upsilon \\ \exists t \in \{1, m\} | A_{it} = 1, A_{jt} = 0 \end{aligned} \tag{2.2.1}$$

We can also define it by saying that \mathbf{A} is k -disjunct if the union of any k columns does not contain any other column⁴.

Definition 2.2.2. Consider a testing matrix $\mathbf{A} \in \{0, 1\}^{m \times n}$.

\mathbf{A} is called k -separable if, for all pairs of distinct subsets $B, C \subset X$ of cardinality $|B|, |C| \leq k$, we have:

$$\bigvee_{b \in B} A_b \neq \bigvee_{c \in C} A_c \tag{2.2.2}$$

We can also define it by saying that \mathbf{A} is k -separable, if boolean sums of sets of k columns are all distinct.

The k -separating property on \mathbf{A} implies that any set of items with up-to k non-zeros can be recovered exactly from the designed pools [DH06]. So we have guarantees that the compression is lossless and a recovery with no error is possible. However, in general the recovery problem requires searching over all subsets. The k -disjunct property simplifies the search, and a simple algorithm exactly recovers the faulty items [DH06] (see COMP in Section 3.2.2). The same algorithm can be used with pools that does not satisfy both properties, albeit losing the correctness guarantee. Following those statements we can say that, once the set of items and their defectiveness are fixed, the only element

⁴Of course not including the ones in the union

which determine the quality of compression and the recoverability of the defectiveness information is determined by \mathbf{A} alone. There are some results about the number of rows m (number of tests) and number of column n (number of items) required to obtain a k -disjunct and k -separable matrix, all those results are extensively studied and reported in *Pooling design and non-adaptive group testing* [DH06]. We cite the most important and representative. Let be, for a given set of parameters P , $m_d(P)$ the minimum number of rows (pools) and $n_d(P)$ the maximum number of columns (items) for a disjunct matrix, $m_s(P)$ and $n_s(P)$ for a separable one. Let's fix n and define the parameter $P = (n, k)$, where k is the number of defectives.

Theorem 2.2.1. *For a k -separable matrix*

$$\binom{n}{k} \leq \sum_{i=k-1}^m \binom{m}{i} \quad (2.2.3)$$

Corollary 2.2.1.1. *for $n \gg k$*

$$m_s(n, k) \geq k \log n (1 + o(1)) \quad (2.2.4)$$

Theorem 2.2.2. *For a k -disjunct matrix*

$$m_d(n, k) \geq \min \left\{ \binom{k+2}{2}, n \right\} \quad (2.2.5)$$

Given those results we recognize lower bounds on number of tests m that describe how effectively we can reduce the number of tests from n to keep guarantees of correctness (k -separability) and to have a trivial decoding phase (k -disjunct), so we know when a construction with such properties exist and we can apply techniques to obtain it. There are several deterministic techniques

to obtain those properties and their discussion is beyond the scope of this work. To whom may want to get more information about design we suggest *Pooling design and non-adaptive group testing* [DH06]. Those results also underline the fact that deterministic designs are not feasible in every settings and scenarios, this motivates the study and adoption of random designs.

Random Design

In the last section we briefly introduced k -disjunct and k -separable matrices. A general observation is that although the methods to construct those matrices are highly efficient when they exist, their existence is rare. On the other hand, a random design, regardless of which type, usually exists for all m and n . Of course, the price to be paid is that there is no guarantee that they will identify all items correctly. Therefore, there might be positive items unidentified, called unresolved positives, or negative items unidentified, called unresolved negatives. A surprising thing is that most random designs do well in restricting the number of unresolved positives or unresolved negatives. There are four types of random design:

- *Random Incidence Design* (RID). Each cell in \mathbf{A} has probability q of being 1.
- *Random k -set design* (RkSD). Each column in \mathbf{A} is a random k -subset of the set $[m] = \{1, 2, \dots, m\}$.
- *Random distinct k -set design* (RDkSD). RkSD except the columns are distinct.

- *Random r -size design (RrSD)*. Each row in \mathbf{A} is a random r -subset of the set $[n] = \{1, 2, \dots, n\}$.

In this work we focus on the random design, as it has the strongest guarantees of applicability. In particular we use the Bernoulli design which is a version of a RID where the probability is the same for all the cell of the matrix \mathbf{A} so that the belonging of items to a pool is sampled with a Bernoulli process $Be(q, m)$. Bounds and theorem of recoverability under such design will be presented in Section 3.1.5.

2.2.2 Decoding Phase

Once the outcome of test are known and fixed, given the matrix of pooled measurements \mathbf{A} , to recover the defectiveness of the candidates it's required to decode the information encoded after the design and testing phase. The previous phases basically give constraint over the defectiveness of items. We can see this phase as a reasoning over a knowledge base where new information can be deduced through an inference process. The main contribution of this work is on this process, so decoding phase will be extensively discussed later on in Section 3.2. We now mention the most important past contribution and related work.

Going back to the work of Malyutov and co-authors in the 1970s (see [Mal13b] for a review of their contribution), they established an analogy between noisy group testing and Shannon's channel coding theorem [Sha48]. The idea is to treat the recovery of the defective set as a decoding procedure for a message transmitted over a noisy channel, where the testing matrix \mathbf{A} represents the codebook used to translate the message. Atia and

Saligrama, recalling such ideas, [AS12] mimics the channel coding theorem's results and obtains an upper bound of $O(k \log n)$ on the number of tests required. In terms of decoding algorithms, the similarity between compressed sensing and group testing (as discussed in [Mal13b], [AAS16]) has been used in [CCJS11, CJS14] by Chan et al. to present testing algorithms for both noiseless and noisy non-adaptive group testing. In particular, the authors introduce the Combinatorial Basis Pursuit (CBP), Combinatorial Orthogonal Matching Pursuit (COMP) algorithms, and their noisy versions (NCBP and NCOMP). They prove universal lower bounds for the number of tests needed to get a certain success probability and upper bounds for the algorithms they are introducing. The COMP algorithm allows the strongest bounds in their paper to be rigorously proved, and will be considered as a baseline algorithm in this work (see section 3.2.2). Other approaches to classical instances of group testing have been proposed in the literature. In particular, its natural integer-programming (IP) formulation has been addressed by Malioutov and Malyutov [MM12], Malyutov and Sadaka [MS09] and Chan et al. [CJS14]: noticing that group testing allows an immediate IP formulation, it is possible to relax the integer program and solve the associated linear version [MM12].

2.2.3 Error Tolerant Model: Noiseless and Noisy Testing

"Someone thinks of a number between one and one million (which is just less than 2^{20}). Another person is allowed to ask up to twenty questions, to each of which the first person is supposed to answer only yes or no. Obviously the number can be guessed

by asking first: Is the number in the first half-million? and then again reduce the reservoir of numbers in the next question by one-half, and so on. Finally the number is obtained in less than $\log_2(1000000)$ questions. Now suppose one were allowed to lie once or twice, then how many questions would one need to get the right answer? One clearly needs more than n questions for guessing one of the 2^n objects because one does not know when the lie was told. This problem is not solved in general."

This citation is from Stanislaw M. Ulam (1909-1984) autobiography *Adventures of a Mathematician* [Ula76] and it introduces error inside a dichotomic search. In particular Ulam's problem is a group testing problem with one defective and at most one or two erroneous tests. In general, more defectives and more errors may be considered. Group Testing algorithm has been devised in order to control and handle error in the testing phase. In particular decoding technique in non-adaptive testing can consider error and inverted test. Those techniques are crucial because most of the real scenarios present some kind of error probability. The error in general is called noise, this derive by the fact that group testing has been paired with communication techniques and problems, in particular compressed sensing.

In order to introduce the used noise model we anticipate the notation that will be later on presented to represent the outcome of test. The vector \mathbf{y} of length m represents the outcome of tests, in particular $y_i = 1$ if the i -th test is positive, $y_i = 0$ otherwise. One of the simplest noise models simply considers the scenario where these values are flipped independently at random with a given probability.

Definition 2.2.3 (Symmetric Noise). In the binary symmetric

noise model, the i -th test outcome is given by

$$y_i = \begin{cases} \bigvee(x_j \mathbf{A}_{ij}) & \text{with probability } d \\ 1 - \bigvee(x_j \mathbf{A}_{ij}) & \text{with probability } d \end{cases} \quad (2.2.6)$$

2.3 Application of Group Testing

⁵ Sobel and Groll listed in their early paper [SG59] some basic applications to unit testing in industrial processes, such as the detection of faulty containers, capacitors, or Christmas tree lights. A prevalent type of fault in the manufacture of electrical circuits is the presence of a short circuit between two nets of a circuit. Short testing constitutes a significant part of the manufacturing process. Several fault patterns can be described as variations or combination of these types of faults. Applications of short testing ranges from printed circuit board testing and circuit testing to functional testing. Recently solutions based on group testing have been proposed in other manufacturing contexts, such as integrated circuits [KR06] and molecular electronics [SFG⁺03]. We list some additional applications here; note that this list is certainly not exhaustive, and is only intended to give a flavour of the wide range of contexts in which group testing has been applied, many of these applications motivate our focus on non-adaptive algorithms. In many settings, adaptive algorithms are impractical, and it is preferable to fix the test design in advance.

⁵This section is a revision of [AJS19, Section 1.7], a survey paper on group testing

2.3.1 Biology

Group testing was born to solve a problem in a medical scenario, so it is no surprise that it has found many more uses in this field, as summarised, for example, in [BBTK96, CH08, DH06]. Application in the medical fields are really common in DNA testing, for example modern sequencing methods search for particular subsequences of the genome in relatively short fragments of DNA [DHH00, SBC⁺02, SAZ09]; since samples from individuals can be easily mixed group testing allow to a significant reduction in the number of tests required to isolate individuals with rare genetic conditions [BBTK96, GGC91] and non-adaptive methods are the common one [DHH00, EGB⁺10, EGN⁺15].

A variant of adaptive testing algorithm that has been proposed to estimate the number of defectives [BBHH⁺17], instead of identify the faulty ones. The estimation can be useful to run a group testing algorithm as it gives an information about the number of faulty items and is an important problem in biological and medical applications [Swa85, CS90]; it is used to estimate the proportion of organisms capable of transmitting the aster-yellows virus in a natural population of leafhoppers [Tho62], estimating the infection rate of yellow-fever virus in a mosquito population [WHB80] and estimating the prevalence of a rare disease using grouped samples to preserve individual anonymity [GH89].

We briefly remark that group testing has also been used in many other biological contexts – see [DHH00, Section 1.3] for a review. For example, this includes the design of protein–protein interaction experiments [MDM13], high-throughput drug screening [KW09], and efficient learning of the Immune–Defective graphs in drug design [GJS17].

2.3.2 Communications

Group testing has had also an important roles in communication scenarios. A really common application is in Multiple Access Channel in which several user can communicate with a single receiver. Adaptive protocols based on group testing to schedule transmissions has been introduced by Hayes [Hay78] which were further developed by many authors (see [Gal85] for a review). There are subsequent complementary non-adaptive applications developed in works such as [KG85] (using random designs) and [DBV02] (using designs based on superimposed code constructions). A similar argument for the related problem of Code-Division Multiple Access (CDMA) is used in [Var95]. Another communication related scenario in which group testing is applied is Cognitive radio networks where ‘secondary users’ can opportunistically transmit on frequency bands which are unoccupied by primary users, see for example [AAES08]. Network tomography and anomaly discovery Group testing has been used to perform (loss) network tomography; that is, to detect faults in a computer network only using certain end-to-end measurements which has been modeled with a connected graph topology. This motivates the study of graph-constrained group testing, which is an area of interest in its own right [GH07, JD08].

2.3.3 Information Technology

Problems in computing has been paired with group testing one because of the shared discrete nature, for example in the Data Storage and Compression field Kautz and Singleton [KS64] describe early applications of superimposed coding strategies to

efficiently searching punch cards and properties of core memories. Hong and Ladner [HL02] describe an adaptive data compression algorithm for images with a parallel with Hwang's algorithms [Hwa72] and has been further extended by Hong in [YS04].

Several problems in the cybersecurity field have been attacked with group testing techniques, in particular with non-adaptive procedures. For example Xuan et al. [XSTZ10] describe how to detect DOS attacks with a group testing algorithm. Goodrich and Madej [GAT05, Mad89] applied group testing to the problem of identifying changed files using a collection of hashes.

Cormode and Muthukrishnan [CM03] show that the identification of "hot" items in a database can be achieved using both adaptive and nonadaptive group testing, even in the presence of noise. A related application is given in [WZC18], which considers the problem of identifying 'heavy hitters' (high-traffic flows) in Internet traffic. In [ZH17] we find an application to bloom filters.

2.3.4 Computer Science

Finally, group testing has been applied to a number of problems in statistics and theoretical computer science. The chronologically first examples are search problems [DHH00, Erd]. Furthermore, group testing has been related to compressed sensing and interpreted as a sparse inference problem in several works in both efforts to try to solve an abstract group testing instance or to apply it to a concrete problem. Gilbert, Iwen and Strauss [GIS08] discuss the relationship between group testing and compressed sensing, following such work [MV13] shows how group testing can be used to perform binary classification of objects, and [EM14] develops a framework for testing arrivals with decreasing defec-

tivity probability. Similar ideas can be used for classification by searching for similar items in high dimensional spaces [SFJ14]. A very interesting application is in learning classification rules that are interpretable by practitioners, such applications often use techniques similar to our contribution. For example, in medicine we may wish to develop a rule based on training data that can diagnose a condition or identify high-risk groups from a number of pieces of measured medical data (features). However, standard machine learning approaches such as support vector machines or neural networks can lead to classification rules that are complex, opaque and hard to interpret for a clinician. For reasons of simplicity, it can be preferable to use sub-optimal classification rules based on a small collection of AND clauses or a small collection of OR clauses. In [EVM15, MVED17], the authors show how such rules can be obtained using a relaxed noisy linear programming formulation of group testing 3.2.4. Lastly classical problems in theoretical computer science has been reduced to group testing, including pattern matching [CEPR10, Ind97, MP04] and the estimation of high degree vertices in hidden bipartite graphs [WLY13]. In addition, generalizations of the group testing problem are studied in this community in their own right, including the ‘k-junta problem’ (see for example [Bla10, BC16, MOS04]). Further studies of the K -junta problem and its variations are [BBHH⁺17], [MOS04] and [ABRW15, AS07].

2.4 Maximum Satisfiability

The maximum satisfiability problem (**MaxSAT**) is the problem of determining the maximum number of clauses, of a given Boolean

formula in conjunctive normal form, that can be made true by an assignment of truth values to the variables of the formula. The satisfiability problem (SAT) is the decision problem of determining if, given a boolean formula, it exists a satisfying assignment to variables such that every clause in the formula is satisfied. MaxSAT is an extension of SAT problem, since its completeness under the $\mathcal{FP}^{\mathcal{NP}}$ class it is able to represent optimization problems.

2.4.1 Mathematical formulation

Definition 2.4.1. Being F a boolean formula, $c \in F$ any clauses belonging to F and σ an assignment $\{0, 1\}^n$ ⁶ to each variable constrained in F , the general formulation of a MaxSAT problem is

$$\sigma_{\text{opt}} = \underset{\sigma}{\operatorname{argmax}} \sum_{C \in F} \mathbb{1}\{\sigma \models C\} \quad (2.4.1)$$

MaxSAT can be approached in different fashion and with different degree of expressivity. To add more expressivity is possible to give weight to each clause $c \in F$, so now the problem is to maximize the weight of the satisfied clauses.

Definition 2.4.2. Being F a boolean formula, $c \in F$ any clauses belonging to F , σ an assignment $\{0, 1\}^n$ to each variable constrained in F and $\text{wt}(C)$ the weight function which returns the weight associated to a clause C , the formulation of a *weighted* MaxSAT problem is

$$\sigma_{\text{opt}} = \underset{\sigma}{\operatorname{argmax}} \sum_{C \in F} \text{wt}(C) \cdot \mathbb{1} \quad (2.4.2)$$

⁶Here n denote the number of variables constrained by F

A further degree of expressivity is obtained dividing clauses in two classes: *Hard* and *Soft*. *Hard* clauses (F_h) are the one that must be mandatory satisfied, so any assignment must satisfy this subset of clauses. *Soft* clauses (F_s) are not mandatory to be satisfied, and they can be weighted or the weight can be uniform so that the problem is to maximize the number of *Soft* clauses satisfied, given that *Hard* ones are satisfied.

Definition 2.4.3. Being F a boolean formula, $F_H \subseteq F$ the subset of hard clauses, $F_s \subseteq F$ the subset of soft clauses, σ an assignment $\{0, 1\}^n$ to each variable constrained in F and $\text{wt}(C)$ the weight function which returns the weight associated to a clause C , the formulation of a *weighted MaxSAT* problem is

$$\sigma^* = \operatorname{argmax}_{\sigma \mid \forall C \in F_h, \sigma \models C} \sum_{C \in F_s} \text{wt}(C) \cdot \mathbb{1}\{\sigma \models C\}$$

Note that *Partially Weighted MaxSAT* can be encoded in *Weighted MaxSAT* by giving ∞ weight to *Hard* clauses.

2.4.2 Solving Techniques

There are two approaches to solve Max-SAT: approximation algorithms based on heuristics [AL97, BP98] that compute near-optimal solutions and exact algorithms that compute optimal solutions. Heuristic algorithms are fast and do not provide any guarantee about the quality of their solutions, while exact algorithms are not so fast but provide a guarantee about the quality of their solutions. Regarding exact algorithms, there have been substantial performance improvements in the last years with the annual Max-SAT competition. Now one of the challenge is to turn exact Max-SAT into a competitive generic approach to

solve combinatorial optimization problems and this work aims to push towards this direction. Max-SAT solvers are usually based on one the following techniques: branch and bound approach [AMP03, LW66], unsatisfiability-based approach [MSP07], and satisfiability-based approach [ABL13]. In our implementation we used a robust MaxSAT solver called Max-HS [DB11] which adopts satisfiability-based technique. Specifically, MaxHS solves the optimization problem by solving sub-instances of SAT using a SAT-oracle while performing arithmetic reasoning in order to exploit strength of both modern SAT-solver and integer programming solvers as CPLEX.

2.4.3 Applications

Solving with MaxSAT is important from a practical perspective. Many interesting real-world problems have been expressed as MaxSAT, from areas such as electronic design automation (EDA), planning, probabilistic inference, and software upgradeability. There are potential applications in bioinformatics, scheduling, and combinatorial auctions as well. Given the success story of SAT solvers, which can be treated as a black box to solve very large problems arising from industrial applications, and the close relationship of MaxSAT and SAT, now the application of MaxSAT to real-world problem are arising and are giving strong and surprising results.

2.5 Phase Transition in Computer Science

Phase transitions are familiar phenomena in physical systems. In general a phase transition is observed when a state of a system

tend to change with respect to some parameters. The term phase transition (or phase change) is most commonly used in physics to describe transitions between solid, liquid, and gaseous states of matter, as well as plasma in rare cases. A phase of a thermodynamic system and the states of matter have uniform physical properties. During a phase transition of a given medium, certain properties of the medium change, as a result of the change of external conditions, such as temperature, pressure, or others. Years of research has revealed that a phase transition also occur in many probabilistic and combinatorial models, including random versions of some classic problems in theoretical computer science. We briefly present SAT phase transition as it is useful to analyze observed behaviour in the experimental evaluation of our model.

2.5.1 SAT Phase Transition

Propositional satisfiability (SAT) is the problem of deciding if there is an assignment for the variables in a propositional formula that makes the formula true. SAT is of considerable practical interest as many AI tasks can be encoded quite naturally in SAT (also the the process of encoding reveal a problem called SMT). SAT is known for being a \mathcal{NP} -complete problem, so is in its worst case untractable, even if over the year efficient heuristic has been developed, and is at the core of study of \mathcal{NP} -completeness and complexity theory [Kar72]. Random instance of SAT has revealed, with fixed length k of clauses (this is called k -SAT)⁷ a phase transition from satisfiability to unsatisfiability and an

⁷Note that, when $k \geq 3$ every SAT problem is reducible to k -SAT, 1-SAT is trivial and 2-SAT is in the \mathcal{P} complexity class.

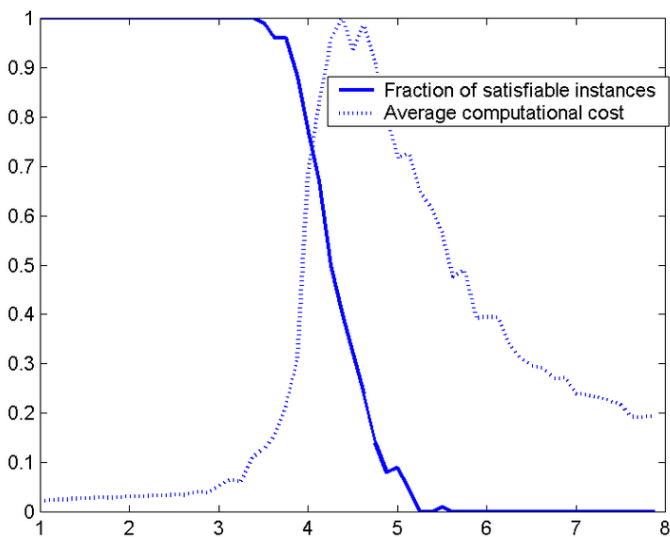


Figure 2.5.1: This figure illustrates the phase transition of a 3-SAT in terms of Satisfiability and Hardness of the problem. On the y axis there's $\mathbb{P}(\text{SAT}|X)$ with fixed $N = 40$ (for the dotted line the reference measure is computational cost in terms of time), and on the x -axis the ratio r .

easy-hard-easy behaviour.

Definition 2.5.1 (Random k -SAT). A random k -SAT problem consists of L clauses, each of which has k literals chosen uniformly from the N possible variables and the N possible negated variables. Let be $X = (N, L, k)$ a class, $R(X)$ denotes problems drawn from this class and $\mathbb{P}(\text{SAT}|X)$ denotes the probability that a problem drawn at random from the class X is satisfiable.

It has been conjectured that for a class X , fixing k and L

there exist a threshold on the ratio $\alpha = \frac{L}{N}$ that characterize the transition from $\text{SAT}(\mathbb{P}(\text{SAT}|X) = 1)$ to $\text{UNSAT}(\mathbb{P}(\text{SAT}|X) = 0)$. Mitchell, Selman, and Levesque [MSL92] observed that the median hardness of the instances is very nicely characterized by this single parameter α . They observed that instance hardness peaks in a critically constrained region determined by α alone. For random 3-SAT this region has been experimentally shown to be around $\alpha \approx 4.26$ and has provided challenging benchmarks as a test-bed for SAT solvers. Basically when the problem is under-constrained or over-constrained it is likely to be easy to solve, the critical region represents a confusion region where the state can be equally likely between SAT and UNSAT (in terms of probability) and so for the solver, even if is not calculating any probability, is getting harder to find a solution⁸. Figure 2.5.1 shows this phenomenon in both Satisfiability and Hardness transitions. Study on phase transition has been important for the theoretical understanding of SAT and in general Constraint Satisfaction Problem. Achlioptas and Oglan [ACO08] studied CSP algorithm and determined algorithmic barrier from phase transition. In particular they studied random graph coloring and proved that completely analogous phase transition also occurs both in random k -SAT and in random hypergraph 2-coloring. And that, for each problem, its location corresponds precisely with the point where all known polynomial-time algorithms fail.

At the heart of statistical physics, discrete mathematics, and theoretical computer science, lie mathematically similar counting

⁸Note that SAT-solvers are based on complete algorithms and always grant to find a solution in finite time, but not in polynomial time unless $\mathcal{P} = \mathcal{NP}$

and optimization problems. This situation leads to a transgression of boundaries so that progress in one discipline can benefit the others. Relating the phase transition phenomenon for 3-SAT to statistical physics, Kirkpatrick and Selman [KS99] showed that the threshold has characteristics typical of phase transitions in the statistical mechanics of disordered materials. Physicists have studied phase transition phenomena in great detail because of the many interesting changes in a system's macroscopic behavior that occur at phase boundaries and so Computer Scientists have done, even if not deep enough. In fact it is still not formally known whether there even exists a critical constant α_c such that as L grows, almost all 3-SAT formulas with $\alpha < \alpha_c$ are satisfiable and almost all 3-SAT formulas with $\alpha > \alpha_c$ are unsatisfiable. Over this observation take place the work of Martin et al. [MMZ01]. They presents the tools and concepts designed by physicists to deal with optimization or decision problems in an accessible language for computer scientists and mathematicians covering in detail the Random Graph, SAT and the Traveling Salesman problems. They observed that the potential connections between discrete mathematics, theoretical computer science and statistical physics become particularly obvious when one considers the typical properties of random systems. The way physicists and mathematicians proceed is quite different. Theoretical physicists generally do not prove theorems, rather they attempt to understand problems by obtaining exact and approximate results based on reasonable hypotheses. In this sense modern computer scientist, when studying phase transition, has approached it in a "physicist" way and we will do so with our observation later on in Section 5.3.

Chapter 3

Problem Formulation

*“No nit not.
Nit no not.
Nit nit folly bololey.
Alife my larder.”*

Robert Wyatt, Alifib

3.1 Notation and Preliminaries

We use capital boldface letters such as \mathbf{X} to denote matrices, while lower boldface letters \mathbf{x} are reserved for vectors/sets. For a matrix \mathbf{X} , \mathbf{X}_i represents the i -th row of \mathbf{X} while for a vector/set \mathbf{x} , x_i represents the i -th element of \mathbf{x} and $\mathbf{X}_{i,j}$ represents the element of the i -th row and j -th column. We use the notion of a sparse vector.

Definition 3.1.1. A Boolean vector $\mathbf{x} \in \{0, 1\}^n$ with dimension n is called a *sparse* vector if the number of non-zero elements $\sum_{i=1}^n x_i \ll n$.

3.1.1 Boolean Logic and CNF

Definition 3.1.2. A propositional formula F in Conjunctive Normal Form (CNF) with n boolean variables $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ is defined as a conjunction of clauses, where each clause C_i is a disjunction of literals. A literal can be either a variable x_i or its complement $\neg x_i$.

Definition 3.1.3. We define σ an assignment to variables and $x_i \in \mathbf{x}$. $\sigma(x_i)$ denote the value assigned to x_i in σ . The propositional satisfiability (SAT) problem finds a *satisfying assignment* or *witness* σ^* to variables in \mathbf{x} that makes F evaluate to 1 (*True*). Given a CNF formula $F = \bigwedge_i C_i$,

$$\sigma^* \models F \iff \forall i, \sigma^* \models C_i$$

wherein

$$\sigma^* \models C_i \iff \exists x \in C_i, \sigma^*(x) = 1$$

In this work, we focus on the weighted variant of CNF wherein a weight function is defined over clauses.

Definition 3.1.4. For a clause C_i and weight function $\text{wt}(\cdot)$. $\text{wt}(C_i)$ denote the weight of clause C_i . A clause C_i is *hard* if $\text{wt}(C_i) = \infty$; otherwise, $\text{wt}(C_i) < \infty$, C_i is called a *soft* clause.

To avoid notational clutter, we overload $\text{wt}(\cdot)$ to denote the weight of an assignment or clause, depending on the context.

Definition 3.1.5. We define the weight of an assignment σ as the sum of weights of the clauses that σ satisfies. Let $\mathbf{1}\{true\} = 1$ and $\mathbf{1}\{false\} = 0$. Formally,

$$\text{wt}(\sigma) = \sum_i \text{wt}(C_i) \cdot \mathbf{1}\{\sigma \models C_i\}$$

3.1.2 MaxSAT

Definition 3.1.6 (MaxSAT). Given a formula F and weight function $\text{wt}(\cdot)$, the problem of MaxSAT is to find an assignment σ^* that has the maximum weight, i.e.,

$$\sigma^* = \text{MaxSAT}(F, \text{wt}(\cdot)) \iff \forall \sigma \neq \sigma^*, \text{wt}(\sigma^*) \geq \text{wt}(\sigma)$$

We use additional lighter notation to denote an optimal assignment using

$$\sigma^* \underset{\text{opt}}{\models} F \iff \sigma^* = \text{MaxSAT}(F, \text{wt}(\cdot))$$

to denote an optimal assignment, so a MaxSAT solution, to a formula F when the context allow to imply function $\text{wt}(\cdot)$.

Our formulation will have positive clause weights, hence MaxSAT corresponds to satisfying as many clauses as possible, and picking the strongest clauses among the unsatisfied ones. Borrowing terminology from the community focused on developing MaxSAT solvers, we are solving a partial weighted MaxSAT instance wherein we mark all the clauses with ∞ weight as hard, and clauses with other positive value less than ∞ weight as soft, and asking for a solution that optimizes the partial weighted MaxSAT formula. Knowledge of the inner workings of MaxSAT solvers and the encoding of representations into weighted MaxSAT instances are not required for this thesis.

3.1.3 Group Testing

Combinatorial Group Testing

Definition 3.1.7 (Items Vector). Let $\mathbf{x} \in \{0, 1\}^n$ be a vector of n items, where $x_i = 1$ denotes a defective item and $x_i = 0$ denotes a non-defective item. Let $k(\mathbf{x}) = \sum_{i=1}^n x_i$ be the number of defective items in \mathbf{x} . We will use k instead of $k(\mathbf{x})$ to denote the number of defective items when the input \mathbf{x} is clear from the context.

Definition 3.1.8 (Pooling Matrix). We define a matrix of pooled measurements $\mathbf{A} \in \{0, 1\}^{m \times n}$ with m tests, where \mathbf{A}_{ij} denotes the j -th item of the i -th row (or test) of \mathbf{A} . Specifically, $\mathbf{A}_{ij} = 1$ if the j -th item belongs to the i -th test, and $\mathbf{A}_{ij} = 0$ otherwise.

Definition 3.1.9 (Outcomes Vector). We define the Boolean vector $\mathbf{y} \in \{0, 1\}^m$, where y_i represents the outcome of the i -th test:

$$y_i = \begin{cases} 1 & \text{if } \exists j \in \{1, \dots, n\}, \mathbf{A}_{ij} \wedge x_j = 1 \\ 0 & \text{otherwise} \end{cases}$$

In order to model a noisy setting, we allow tests to return inverted/flipped outcomes.

Definition 3.1.10 (Boolean Noise Vector). We define a boolean vector $\boldsymbol{\xi} \in \{0, 1\}^m$ such that the i -th test gives an inverted outcome iff $\xi_i = 1$ and $\xi_i = 0$ otherwise. In the noisy setting, we modify the definition of \mathbf{y} as follows:

$$y_i = \begin{cases} 1 & \text{if } \exists j \in \{1, \dots, n\}, (\mathbf{A}_{ij} \wedge x_j) \oplus \xi_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

Here \oplus represents the logical XOR operator.

Group Testing Set Definition

Definition 3.1.11 (Defective Set). We define a defective set \mathcal{K} a subset $\mathcal{K} \subseteq \mathcal{N}$ where \mathcal{K} contains all the defective items ($x_j = 1$). $\mathcal{N} = \{1, n\}$, where $n = |\mathbf{x}|$.

$$\mathcal{K} = \{j : x_j = 1\}$$

Definition 3.1.12 (Satisfying Set). Given a pool design \mathbf{A} and outcomes \mathbf{y} , we shall call a set of items $\mathcal{L} \subseteq \mathcal{N}$ a satisfying set if group testing with defective set \mathcal{L} and test design \mathbf{A} would lead to the outcomes \mathbf{y} . Clearly the defective set \mathcal{K} itself is a satisfying set. If we express a decision problem with constraints generated from pool design \mathbf{A} and outcome \mathbf{y} a non-satisfying set is a set that does not satisfy such constraints.

Definition 3.1.13 (Decoding Algorithm). A decoding algorithm is a method to estimate the defective set from the test outcomes; that is, a function

$$\widehat{\mathcal{K}} : \{0, 1\}^m \rightarrow \mathcal{P}(\mathcal{N})$$

where we write $\widehat{\mathcal{K}}$, that associates to each outcome vector \mathbf{y} a subset $\widehat{\mathcal{K}} \subseteq \mathcal{N}$ of the items. We write $\mathcal{P}(\mathcal{N})$ for the power set of \mathcal{N} .

Definition 3.1.14. A item j is said to be and intruding non-defective if it is non-defective and it never appears in any negative tests.

$$(x_j = 0) \wedge \bar{\exists} i (\mathbf{A}_{ij} = 1 \wedge y_i = 0)$$

Definition 3.1.15 (Masked Defective). A item j is said to be a masked defective if it is defective and it never appears in a positive tests that does not contains any other positive item.

$$(x_j = 1) \wedge \bar{\exists} i (\mathbf{A}_{ij} = 1 \wedge y_i = 1 \wedge \bar{\exists} l \neq j ((\mathbf{A}_{il} = 1 \wedge x_l = 1)))$$

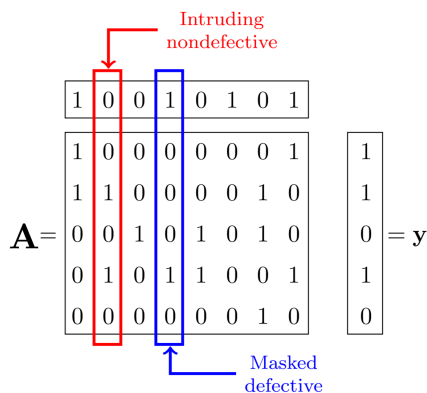


Figure 3.1.1: An example of a group testing problem, including a masked defective and an intruding non-defective, in the terminology we introduce here.

3.1.4 Stochastic Models

We consider Bernoulli trials [AJS19] to model a stochastic group testing instance. In particular, we consider three i.i.d. Bernoulli processes. A vector of item \mathbf{x} is a bernoulli process $\sim \text{Bern}(p)$ such that

$$\begin{cases} x_j = 1, & \text{with probability } p \\ x_j = 0, & \text{with probability } 1 - p \end{cases} \quad (3.1.1)$$

an item is defective independently with probability $p < 0.5$. Then we use a random model to design matrix \mathbf{A} , in particular we use bernoulli process as a Random Incidence Design (see Section 2.2.1) such that each item has a probability item i belongs to test j independently with probability q , and in the noisy setting, a test outcome is inverted independently with probability $d < 0.5$.

3.1.5 Group Testing Bounds

Definition 3.1.16 (Success Probability). Under the exact recovery criterion, the (average) error probability for noiseless group testing with a combinatorial prior is

$$\mathbb{P}(err) := \frac{1}{\binom{n}{k}} \sum_{\mathcal{K}:|\mathcal{K}|\leq n} \mathbb{P}(\widehat{\mathcal{K}}(\mathbf{A}, \mathbf{y}) \neq \mathcal{K})$$

where \mathbf{y} is related to \mathbf{A} and \mathcal{K} via the group testing model and the probability \mathbb{P} is over the randomness in the test design \mathbf{A} (if randomized), the group testing model (if random noise is present), and the decoding algorithm $\widehat{\mathcal{K}}$ (if randomized).

We call

$$\mathbb{P}(suc) := 1 - \mathbb{P}(err)$$

the success probability.

We note that this average error probability refers to an average over a uniformly distributed choice of defective set \mathcal{K} , where we can think of this randomness as being introduced by nature. Even in a setting where the true defective set \mathcal{K} is actually deterministic, this can be a useful way to think of randomness in the model. Since the outcomes of the tests only depend on the columns of the test matrix \mathbf{A} corresponding to \mathcal{K} , the same average error probability is achieved even for a fixed \mathcal{K} by any exchangeable matrix design (that is, one where the distribution of \mathbf{A} is invariant under uniformly-chosen column permutations). This includes Bernoulli, near-constant tests-per-item, and doubly regular designs, as well as any deterministic matrix construction acted on by uniformly random column permutations. We now introduce some bound on probability of success that will be used in the experimental section.

Theorem 3.1.1 (Counting Bound). *Any algorithm (adaptive or nonadaptive) for recovering the defective set with m tests has success probability satisfying*

$$\mathbb{P}(\text{suc}) \leq \frac{2^m}{\binom{n}{k}}$$

Corollary 3.1.1.1. *In particular, $\mathbb{P}(\text{suc}) \rightarrow 0$ as $n \rightarrow \infty$ whenever*

$$m \leq (1 - \eta) \log_2 \binom{n}{k} \quad (3.1.2)$$

for arbitrarily small $\eta > 0$.

Remark. From an information-theoretic viewpoint, this result essentially states that since the prior uncertainty is $\log_2 \binom{n}{k}$ for a uniformly random defective set, and each test is a yes/no answer revealing at most 1 bit of information, we require at least $\log_2 \binom{n}{k}$ tests. Because the result is based on counting the number of defective sets, we refer to it as the counting bound, we will mostly use the asymptotic version in the experimental evaluation.

Using Stirling approximation we can obtain a simpler bound that is interestingly similar with the number of tests required by the adaptive binary splitting algorithm.

Corollary 3.1.1.2. *$\mathbb{P}(\text{suc}) \rightarrow 0$ as $n \rightarrow \infty$ whenever*

$$m \leq k \log_2 \left(\frac{n}{k} \right) (1 + o(1)) \quad (3.1.3)$$

Following the reasoning of Counting Bound, it will be useful to think about how many bits of information we learn (on average) per test. Using an analogy with channel coding, we shall call this the rate of group testing. In general, if the defective set \mathcal{K} is chosen from some underlying random process with entropy H ,

then for a group testing strategy with m tests, Aldridge, Johnson and Scarlett [AJS19] define the rate to be H/m . In particular, under a combinatorial prior, where the defective set \mathcal{K} is chosen uniformly from the $\binom{n}{k}$ possible sets, the entropy is $H = \log_2 \binom{n}{k}$, leading to the following definitions.

Definition 3.1.17 (Group Testing Rate). Given a group testing strategy under a combinatorial prior with n items, k defective items, and m tests, we define the rate [AJS19] Δ :

$$\Delta = \frac{\log_2 \binom{n}{k}}{m} \quad (3.1.4)$$

Definition 3.1.18 (Achievable Rate). Consider a group testing problem, possibly with some aspects fixed (for example, the random test design or the decoding algorithm), in a setting where the number of defectives scales as $k = k(n)$ according to some function (e.g., $k(n) = \Theta(n^\gamma)$ with $\gamma \in (0, 1)$). We say [AJS19] a rate Δ' is achievable if, for any $\delta, \epsilon > 0$, for n sufficiently large there exists a group testing strategies with a number of tests $m = m(n)$ such that the rate satisfies

$$\Delta = \frac{\log_2 \binom{n}{k}}{m} > \Delta' - \delta \quad (3.1.5)$$

and the error probability $\mathbb{P}(err)$ is at most ϵ .

Remark. The Counting Bound also implies the so-called strong converse: The error probability $\mathbb{P}(err)$ tends to 1 when $m \leq (1 - \eta) \log_2 \binom{n}{k}$ for arbitrarily small $\eta > 0$, which corresponds to a rate $\Delta' \geq \frac{1}{1-\eta} > 1$.

For the noisy settings, recalling the symmetric noise model defined in 2.2.3 where the noise is a Bernoulli process with probability d then (from the work of Johnson [Joh17]):

Theorem 3.1.2 (Noisy Group Testing Bound). *Any algorithm (adaptive or nonadaptive) for recovering the defective set with m tests, given a noise symmetric model with probability d , has success probability satisfying*

$$\mathbb{P}(\text{suc}) \leq \frac{m(1 - H(d))}{\log_2 \binom{n}{k}}$$

where $H(d)$ is the binary entropy function

$$H(d) = -d \log_2(d) - (1 - d) \log_2(1 - d)$$

Corollary 3.1.2.1. *In particular, $\mathbb{P}(\text{suc}) \rightarrow 0$ as $n \rightarrow \infty$ whenever*

$$m \leq \frac{k \log\left(\frac{n}{k}\right)}{\log 2 - H(d)}(1 - \eta) \quad (3.1.6)$$

for arbitrarily small $\eta > 0$.

3.2 Non-Adaptive Group Testing Decoding Phase

In this thesis we focus the effort on study an optimal and general way to decode an instance of non-adaptive group testing. In particular we study and analyze the optimality of the so-called model SSS [BJA13] from which we derive a novel encoding and propose a MaxSAT framework. This section aim to introduce the formulation of the problem in both noiseless and noisy (error tolerant) settings. In order to do that we refer to the most important related work and cite the algorithm which has lead the improvement and motivated us to adopt the proposed model.

3.2.1 Compressed Sensing and $\widehat{\mathcal{K}}_{SSS}$

An important body of work on non-adaptive group testing has underlined the relation and the parallel between group testing and compressed sensing. Group testing is related in spirit to compressed sensing (CS). In CS we are given an n -dimensional sparse signal with support size k . Random projections of the sparse signal are obtained. The goal is to identify the support set while minimizing the number of projections. Group testing can be viewed as a boolean version of CS where we apply a measurement matrix (\mathbf{A}) to a sparse vector corresponding to the defective set with the goal of reconstructing the support, i.e. identify the defective items. Compressed sensing is set in the context of real vector spaces with additive noise, while group testing studies the same problem in the boolean setting and with Bernoulli noise. Recent works on group testing drew parallels with Compressed Sensing, in particular using the assumption on sparsity of signal to recover the set \mathcal{K} computing the Smallest Satisfying Set $\widehat{\mathcal{K}}_{SSS}$, which is the satisfying set (from definition 3.1.12) with the smallest cardinality.

3.2.2 Related Decoding Algorithms

We now present some algorithm which concepts lead to $\widehat{\mathcal{K}}_{SSS}$ adoption and that have been implemented in order to benchmark the proposed solution. Their investigation has the purpose to make the reader understanding why is smart to find the solution with less faulty items and which hindrance undergoes the process to decode an instance of non-adaptive group testing.

COMP

The simpler inference algorithm, in the noiseless settings is constructed on the following reasoning: if an item appears in a negative test, then it cannot be defective.

Definition 3.2.1. We consider the guaranteed not defective (\mathcal{ND}) set

$$\mathcal{ND} := \{j : \exists t | \mathbf{A}_{tj} = 1 \wedge y_t = 0\}$$

and write

$$\mathcal{PD} = \mathcal{ND}^{\cup \mathcal{N}}$$

for the set of possible defectives (\mathcal{PD}).

The algorithm called Baseline algorithm and is also known as **Combinatorial Orthogonal Matching Pursuit (COMP)** [CCJS11]. This algorithm compute the set of possible defectives (\mathcal{PD}) deduced from testing matrix \mathbf{A} and outcome vector \mathbf{y} and return it as an estimate of \mathcal{K} . Note that $\widehat{\mathcal{K}}_{\text{COMP}}$ is a satisfying set (by Definition 3.1.12), in fact it is the largest satisfying set. Thus if the true defective set \mathcal{K} is the unique satisfying set then the COMP algorithm certainly finds it. If it is not unique then the COMP algorithm can only make false positive errors (declaring non-defective items to be defective), and never makes false-negative errors (declaring defective items to be non-defective).

Lemma 3.2.1. *The estimate $\widehat{\mathcal{K}}_{\text{COMP}}$ generated by the COMP algorithm is a satisfying set (in the sense of Definition 3.1.12) and contains no false negatives. Every satisfying set is a subset of $\widehat{\mathcal{K}}_{\text{COMP}}$, so $\widehat{\mathcal{K}}_{\text{COMP}}$ is the unique largest satisfying set. Hence:*

$$\mathcal{K} \subseteq \widehat{\mathcal{K}}_{\text{COMP}}$$

Such algorithm has the guarantee to find the exact solution in case the recovery matrix \mathbf{A} is k -disjunct. **COMP** can aim to give a solution also with design that does not have such properties, but guarantee of exact solution are lost. This is because k -disjunctness implies that every non-defective item appears in at least one negative test, hence there are no intruding non-defectives. However, notice that k -disjunctness is a very restrictive property, since it imposes restrictions on all sets \mathcal{S} of cardinality $\leq k$.

DD

The **Definitive Defective DD** algorithm, proposed by Aldridge, Baldassini and Johnson [BJA13], assumes that each item is non-defective unless there is a certain simple proof that it is defective. It is conceptually the opposite of **COMP**. The motivation behind such algorithm is that once the possible defective (\mathcal{PD}) items have been identified, some other elements can be identified as being definitely defective (\mathcal{DD}). The key idea is that if a positive test contains exactly one possible defective item, then we can in fact be certain that such item is defective. This algorithm uses the possible defectives \mathcal{PD} found in the **COMP** algorithm as a starting point (step 1). Then, for each positive test which contains a single item from \mathcal{PD} , declare the corresponding item to be defective (step 2), and all remaining items are declared to be non-defective (step 3). Notice that steps 1 and 2 in the **DD** algorithm make no mistake; in particular step 1 just isolates all items that are non-defectives, which can then be ignored, thus allowing us to restrict our attention to the items in \mathcal{PD} . The set \mathcal{PD} contains the k true defectives, plus a (random) number g of intruding non-defectives (see 3.1.14), meaning we can analyse the $m \times (k + g)$

submatrix \mathbf{S} , corresponding to the items in \mathcal{PD} . Step 2, in turn, isolates the definitely defective items of \mathcal{PD} . After step 2 we are then left with g intruding non-defectives that haven't been discarded in step 1 and some masked defectives (see definition 3.1.15). Hence only step 3 can make a mistake, which occurs when there are masked defectives which are erroneously declared to be non-defective. One justification for DD is the observation that removing nondefective items from a test does not affect the outcome of the test, so the problem is the same as if we use the submatrix with columns in \mathcal{PD} . In addition, the principle to assume non-defective unless proved otherwise (used by DD) should be preferable to the rule 'assume defective unless proved otherwise' (used by COMP) under the natural assumption that defectivity is rare. Intuitively $\widehat{\mathcal{K}}_{DD}$ is a lower approximation of the Smallest Satisfying Set $\widehat{\mathcal{K}}_{SSS}$, in fact when $\widehat{\mathcal{K}}_{DD}$ is a satisfying set then it coincides with $\widehat{\mathcal{K}}_{SSS}$.

Lemma 3.2.2. *The estimate $\widehat{\mathcal{K}}_{DD}$ generated by the DD algorithm has no false positives.*

$$\widehat{\mathcal{K}}_{DD} \subseteq \mathcal{K}$$

Conversely, the COMP algorithm assumes that these unknown items are defective, thereby often making false positive errors. The main difference between the DD algorithm of Alridge et al. [BJA13] and the COMP algorithm of Chan et al. [CCJS11] is that COMP succeeds if and only if $g = 0$, whereas DD can succeed for positive g .

Sequential-COMP

SCOMP is an algorithm due to Aldridge, Baldassini, and Johnson [BJA13] that builds a satisfying set by starting from the set of definite defectives (\mathcal{DD}) and sequentially adding new items until a satisfying set is reached. The name comes from ‘Sequential COMP’, as it can be viewed as a sequential version of the COMP algorithm. This algorithm is an improvement over the DD algorithm and it uses $\widehat{\mathcal{K}}_{\text{DD}}$ to build another estimate $\widehat{\mathcal{K}}_{\text{SCOMP}}$ and improve such result.

Definition 3.2.2 (Unexplained Test). Given an estimate $\widehat{\mathcal{K}}$, we say that a positive test is unexplained by $\widehat{\mathcal{K}}$ if its pool contains no element from $\widehat{\mathcal{K}}$. Note that a set $\widehat{\mathcal{K}} \subseteq \mathcal{PD}$ of possible defectives being a satisfying set is equivalent to there being no unexplained positive tests.

Since each unexplained test must contain at least one of the masked defectives in $\mathcal{K} \setminus \widehat{\mathcal{K}}_{\text{DD}}$, we might consider items in \mathcal{PD} that appear in many unexplained tests as most likely to be defective. The SCOMP algorithm uses this principle to sequentially and greedily extend $\widehat{\mathcal{K}}_{\text{DD}}$ to a satisfying set, by seeking items which explain the most currently unexplained tests. This is an attempt to exploit all the information available at each step, which is updated every time an item in \mathcal{PD} is added to $\widehat{\mathcal{K}}$. Note also that the set of all possible defectives is satisfying, so the SCOMP algorithm does indeed terminate. Even if SCOMP is difficult to be analyzed from a theoretical perspective, Aldridge et al. [BJA13] give experimental evidence that it outperforms DD.

3.2.3 Optimal Solution: SSS

Under the assumption of sparsity of \mathbf{x} ($k \ll n$) DD and SCOMP try to approximate a solution which consider less faulty items as possible and justify the outcome \mathbf{y} . Such solution is called the Smallest Satisfying Set $\widehat{\mathcal{K}}_{SSS}$. In the work of Aldridge et al. [BJA13] such algorithm, called SSS algorithm, is considered unfeasible as the problem to solve is hard as an integer linear programming problem. In that work SSS is considered essentially optimal and in small ($n \in \{100, 500\}$) settings is used as a benchmark to the other proposed algorithms. The essential optimality of such model will be underlined in Section 4.3 and to avoid redundancy will not be presented here. It is interesting to notice the analogy with Chvatal's approximation algorithm to the set covering problem (or just 'set cover') – see [Vaz01] for a discussion. Given a set \mathcal{U} and a family of subsets $\mathcal{S} \subseteq \mathcal{P}(\mathcal{U})$, set cover requires us to find the smallest family of subsets in \mathcal{S} whose union contains all elements of \mathcal{U} . This optimisation problem is NP-hard, as for a putative solution optimality cannot be verified in polynomial time. In 1979, Chvatal [Chv79] proposed an approximate solution by choosing, at each stage, the set in \mathcal{S} that covers the most uncovered elements. The algorithm produces a solution which can be at most $H(|\mathcal{U}|)$ times larger than the optimal. Similarly, SCOMP chooses defective items in a greedy manner to 'cover' (or in our terminology, 'explain') as many tests as possible, until all tests are explained. So similarly, SCOMP guarantees to find a satisfying set with no more than $kH(k) \approx k \ln k$ items.

Theorem 3.2.3 (\mathcal{NP} -Hardness of $\widehat{\mathcal{K}}_{SSS}$). *Given the matrix of pooled measurements \mathbf{A} and the outcome vector \mathbf{y} , the problem*

of computing the Smallest Satisfying Set $\widehat{\mathcal{K}}_{SSS}$ is \mathcal{NP} -Hard.

Our goal is to find a way to compute the $\widehat{\mathcal{K}}_{SSS}$ efficiently enough, as polynomial time is not achievable, in order to make such solution feasible and scalable so that becomes applicable as the other proposed algorithms to real problems.

Combinatorial Optimization Formulation

Given the measurement matrix $\mathbf{A} \in \{0, 1\}^{m \times n}$ and test vector $\mathbf{y} \in \{0, 1\}^m$, we attempt to find the smallest set of defective items in \mathbf{x} , where each item in \mathbf{x} is defective with equal probability by a combinatorial prior. In the noiseless setting, we minimize the following function subject to the constraints (i.e., there are at least one defective item(s) in a positive test and no defective item in a negative test).

$$\begin{aligned} & \min \sum_{j=1}^n \widehat{x}_j \\ \text{subject to } & \sum_{j=1}^n \mathbf{A}_{tj} \widehat{x}_j \geq 1 \text{ when } y_t = 1 \\ & \sum_{j=1}^n \mathbf{A}_{tj} \widehat{x}_j = 0 \text{ when } y_t = 0 \\ & \widehat{x}_j \in \{0, 1\} \end{aligned}$$

which is an integer linear programming problem, known to be NP-Hard. We want that the optimal $\widehat{\mathbf{x}}$ will be close to the true input vector \mathbf{x} , since taking $\widehat{\mathbf{x}} = \mathbf{x}$ will satisfy the constraints. In general, we think of each vector $\widehat{\mathbf{x}} \in \{0, 1\}^n$ as the indicator function of some putative defective set \mathcal{L} , with $\mathcal{L} = \mathcal{L}(\widehat{\mathbf{x}}) := \{j : x_j = 1\}$.

The first two constraints ensure that \mathcal{L} is satisfying, by considering the positive and negative tests respectively. Hence, each $\widehat{\mathbf{x}}$ that achieves the minimal value of the linear program is the indicator function of a satisfying set of minimal size, i.e., $\mathcal{K} = \{j : \widehat{\mathbf{x}}_j = 1\}$ is a smallest satisfying set.

3.2.4 Noisy Extension

Each of the presented algorithms has extension in the noisy settings that allow to model a wrong test outcome, this is a natural feature of the non-adaptive approach. The presentation of each variant is beyond the scope of this thesis. We now present the extension to the ILP formulation of $\widehat{\mathcal{K}}_{SSS}$.

In the noisy setting, each test may be inverted/flipped in one of two ways: original test finds a defective item(s) but the noisy output indicates that all items are non-defective, and vice versa. In the noisy setting, we minimize noise variable ξ_i while preferring \mathbf{x} to be the sparsest [MM12] with i.i.d. prior on both \mathbf{x} and $\boldsymbol{\xi}$. In this setting, the constraints are similar to the noiseless setting except that now a test may be flipped.

$$\min \sum_{j=1}^n x_j + \lambda \sum_{i=1}^m \xi_i \quad (3.2.1)$$

The objective function is formulated as to obtain the *sparsest* and *less noisy* solution, where the parameter $\lambda \in \mathbb{R}^+$ balances the trade-off between the amount of noise and the sparsity of the solution. In Section 4.3, we discuss how to set the value of λ and the essential optimality of this model under sparsity assumption ($k \ll n$).

Linear Programming Relaxations:

Linear programming (LP) algorithms have been proposed as a way to approximate $\widehat{\mathcal{K}}_{SSS}$ with practical runtime, by solving a relaxed version of the smallest satisfying set problem. They can be paired with **SCOMP** in that sense, in fact has been shown that linear programming are the most efficient and accurate state of the art method to approximate $\widehat{\mathcal{K}}_{SSS}$ and has in general great performance in decoding a non-adaptive instance in the same setting [MM12]. Because of this LP-relax method will be the main benchmark to the extensive experimental analysis of our **MaxSAT** framework in terms of accuracy and runtime. The following formulation is the LP-relaxed version of 3.2.1, which is the noisy settings. The noiseless formulation can be trivially derived by removing the slack variables ξ_i so it is not included to avoid redundancy. This formulation underlines the parallel between group testing and compressed sensing, in fact the relaxation trivially consider the two set of variables $\widehat{\mathbf{x}}$ and

$$\begin{aligned} & \min \sum_{j=1}^n \widehat{x}_j + \lambda \sum_{i=1}^n \xi_i \\ \text{subject to } & \widehat{x}_j \geq 0 \\ & \xi_i \geq 0 \\ & \xi_i \leq 1 \\ & \sum_{j=1}^n \mathbf{A}_{tj} \widehat{x}_j + \xi_j \geq 1 \text{ when } y_t = 1 \\ & \sum_{j=1}^n \mathbf{A}_{tj} \widehat{x}_j = \xi_t \text{ when } y_t = 0 \end{aligned}$$

Chapter 4

MaxSAT Encoding and Optimality

"The purpose of computation is insight, not numbers."

Richard Hamming

This chapter describe the theory behind the primary contribution of this thesis, MGT, a MaxSAT-based framework for solving the decoding phase of non-adaptive group testing. We first discuss the MaxSAT encoding for both noiseless and noisy setting. Later we propose a compact MaxSAT encoding for the noisy setting and prove its soundness and equivalence with the trivial encoding. Finally, we discuss how to set the value of λ showing the essential optimality of the model under Bernoulli testing assumption.

4.1 MaxSAT Encoding

We first describe the MaxSAT encoding for the noiseless setting and later extend the formulation to noisy setting.

4.1.1 Noiseless Setting:

We consider a unit soft clause that tries to falsify each x_j in the objective function 3.2.3. The weight of the soft clause is 1, which is derived from the coefficient of x_j in the objective function in Eq. 3.2.3:

$$S_j := \neg x_j; \quad \text{wt}(S_j) = 1.$$

To encode the constraints associated with the tests, we construct hard clauses in the MaxSAT query. Recall that $\mathbf{A}_{i,j} = 1$ denotes the j -th item being included in the i -th test. When the outcome of the i -th test is positive ($y_i = 1$), there must be at least one defective item included in that test. Therefore, we construct the following hard clause when $y_i = 1$:

$$C_i := \bigvee_{j|\mathbf{A}_{i,j}=1} x_j; \quad \text{wt}(C_i) = \infty.$$

On the other hand, when $y_i = 0$, all items included in the i -th test must be non-defective. Therefore we construct the following hard clause when $y_i = 0$:

$$C_i := \neg\left(\bigvee_{j|\mathbf{A}_{i,j}=1} x_j\right); \quad \text{wt}(C_i) = \infty.$$

In this case, we apply de Morgan's law on clause C_i to convert it into CNF, $C_i = \bigwedge_{j|\mathbf{A}_{i,j}=1} \tilde{C}_{ij}$, where \tilde{C}_{ij} is defined as follows:

$$\tilde{C}_{ij} := \neg x_j; \quad \text{wt}(\tilde{C}_{ij}) = \infty.$$

Finally, the MaxSAT formula F is the conjunction of all clauses:

$$F := \bigwedge_{j=1}^n S_j \wedge \bigwedge_{i=1, y_i=1}^m C_i \wedge \bigwedge_{i=1, y_i=0, j | \mathbf{A}_{i,j}=1}^m \tilde{C}_{ij}. \quad (4.1.1)$$

Since all clauses S_j, C_i, \tilde{C}_{ij} are in CNF, no further translation is required. Once the MaxSAT formula F is formulated, an off-the-shelf MaxSAT solver takes formula F and weight $\text{wt}(\cdot)$ as inputs and returns an optimal assignment to the variable x_j . We find the defective items according to the optimal assignment as follows.

Construction 4.1.1 (Mapping of MaxSAT to $\widehat{\mathcal{K}}_{SSS}$). *Consider the assignment $\sigma^* = \text{MaxSAT}(F, \text{wt}(\cdot))$. Then item j is detected to be defective if $\sigma^*(x_j) = 1$. Hence*

$$x_j \in \widehat{\mathcal{K}}_{SSS} \iff \sigma^*(x_j) = 1$$

4.1.2 Noisy Setting:

In addition to S_j , we construct unit soft clauses that try to falsify each noise variable ξ_i , $\boldsymbol{\xi}$ represents the boolean vector of all the noise variable. The weight of each soft clause is λ ¹ which is the coefficient of ξ_i in Eq. 3.2.1:

$$N_i := \neg \xi_i; \quad \text{wt}(N_i) = \lambda.$$

In the noisy setting, test y_i is inverted when $\xi_i = 1$, and otherwise y_i remains same. Hence, we construct hard clauses C_i for positive tests and \tilde{C}_i for negative tests as follows:

¹As we consider Bernoulli noise with i.i.d. variables the weight is uniform

$$\begin{aligned}
 C_i &:= \left(\bigvee_{j|\mathbf{A}_{i,j}=1} x_j \right) \oplus \xi_i && \text{when } y_i = 1, \\
 \tilde{C}_i &:= \left(\bigwedge_{j|\mathbf{A}_{i,j}=1} \neg x_j \right) \oplus \xi_i && \text{when } y_i = 0.
 \end{aligned}$$

Here the hard clauses C_i and \tilde{C}_i have XOR operators in their definitions; that is, these clauses are not in CNF. We first translate these clauses into CNF and then call an off-the-shelf MaxSAT solver for the optimal solution. All translated CNF clauses have weight ∞ . Next, we discuss the translation in detail.

- For a positive test output ($y_i = 1$), if there are t_i literals in the clause $\bigvee_{j|\mathbf{A}_{i,j}=1} x_j$, standard CNF translation generates $(t_i + 1)$ CNF clauses while translating $(\bigvee_{j|\mathbf{A}_{i,j}=1} x_j) \oplus \xi_i$. For example, $(x_1 \vee x_2 \vee x_3) \oplus \xi$ is translated into 4-clause CNF as $(\neg \xi \vee \neg x_1) \wedge (\neg \xi \vee \neg x_2) \wedge (\neg \xi \vee \neg x_3) \wedge (w \vee x_1 \vee x_2 \vee x_3)$.
- For a negative test output ($y_i = 0$), if there are t_i literals in $\bigwedge_{j|\mathbf{A}_{i,j}=1} \neg x_j$, standard CNF translation generates $(t_i + 1)$ CNF clauses while translating $(\bigwedge_{j|\mathbf{A}_{i,j}=1} \neg x_j) \oplus \xi_i$. For example, $(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \oplus \xi$ is translated into 4-clause CNF as $(\neg x_1 \vee \xi) \wedge (\neg x_2 \vee \xi) \wedge (\neg x_3 \vee \xi) \wedge (\neg \xi \vee x_1 \vee x_2 \vee x_3)$.

Once all clauses are translated into CNF, we formulate F as follows.

$$F := \bigwedge_{j=1}^n S_j \wedge \bigwedge_{i=1}^m N_i \wedge \bigwedge_{i=1, y_i=1}^m C_i \wedge \bigwedge_{i=1, y_i=0}^m \tilde{C}_i \quad (4.1.2)$$

In the noisy setting, we find the defective items and noisy tests from the optimal assignment to the variables of F as follows.

Construction 4.1.2 (Mapping of MaxSAT to $\widehat{\mathcal{K}}_{SSS}$ in Noisy Settings). *Consider the assignment $\sigma^* = \text{MaxSAT}(F, \text{wt}(\cdot))$. Then item j is detected to be defective if $\sigma^*(x_j) = 1$, and test i is declared inverted if $\sigma^*(\xi_i) = 1$.*

4.2 A Compact Encoding for Noisy Setting

Since the MaxSAT query in the noisy setting has XOR operators in the definition, its translation to CNF generates additional clauses proportional to t number of variable tested in each test (on average $q \cdot n$ clauses, see Section 3.1.4). In general it's not true that less clauses lead to easier solution, there are several example in the SAT literature that prove the opposite. We propose both model then we experimentally verify that the compact encoding enhance dramatically the scalability of the proposed algorithm.

4.2.1 Intuition and Formulation

The improvement over the encoding is done by leveraging the soft clauses derived from objective 4.1.2. The intuition is that the hard clauses formulated in XOR has the two logical components (faultiness and noisiness) that are both minimized by the referenced objective function. XOR force the exclusive satisfiability, but we claim that the pairing of relaxed XOR(OR) with the objective function will lead to the same solution. The formulation of the objective function is pushing the solution to consider less

noise as possible and less faulty as possible and the solution has a *bonus* in form of weight λ to not have noise, then it will never be optimal to satisfy both condition in OR. This observation combined with the fact that ξ_i variable is constrained only by clauses generated by test i suggest us that the XOR constraints are implied by the relaxed formulation.

Formally, we propose a compact encoding, where we replace XOR with OR in both C_i and \tilde{C}_i and define relaxed clauses C'_i and \tilde{C}'_i respectively. We first define the relaxed clauses C'_i and \tilde{C}'_i , and then provide the theoretical guarantee of the optimal solution of the compact encoding:

$$C'_i := \left(\bigvee_{j|\mathbf{A}_{i,j}=1} x_j \right) \vee \xi_i \quad \text{when } y_i = 1,$$

$$\tilde{C}'_i := \left(\bigwedge_{j|\mathbf{A}_{i,j}=1} \neg x_j \right) \vee \xi_i \quad \text{when } y_i = 0.$$

4.2.2 Equivalence of Encodings

Let $\sigma = \text{MaxSAT}(F, \text{wt}(\cdot))$ be the optimal assignment to the variables in the MaxSAT formula F . Here we slightly abuse notation and define $\sigma \models_{\text{opt}} F$ to represent that σ is the optimal solution of a MaxSAT formula F .

Lemma 4.2.1. *Being σ an assignment, if*

$$\sigma \models \left(\bigwedge_{i=1, y_i=1}^m C_i \wedge \bigwedge_{i=1, y_i=0}^m \tilde{C}_i \wedge \bigwedge_{i=1, y_i=1}^m C'_i \wedge \bigwedge_{i=1, y_i=0}^m \tilde{C}'_i \right)$$

then:

$$\sigma \models_{\text{opt}} F \Leftrightarrow \sigma \models_{\text{opt}} F' \quad (4.2.1)$$

Remark. Lemma 4.2.1 is trivially provable as both encoding share the same soft clauses, so if an assignment σ satisfy hard clauses for both encoding, then it is absurd that it is optimal exclusively for one of the two.

Theorem 4.2.2 (Equivalence of XOR and Compact Encoding). *Let F be a MaxSAT formula in the noisy setting, and F' be the compact encoded formula with the above relaxation of XOR. Then σ is an optimal solution to F iff σ is an optimal solution to F' , i.e., although $F \neq F'$, it holds that*

$$\sigma \underset{\text{opt}}{\models} F \iff \sigma \underset{\text{opt}}{\models} F' \quad (4.2.2)$$

Remark. The intuition behind Theorem 4.2.2 is that an optimal solution to F minimizes a weighted sum of the sparsity and the number of tests flipped, and an optimal solution to F' minimizes a weighted sum of the sparsity and the number of tests whose constraints are ‘ignored’. By taking the tests marked ‘flipped’ in F and marking them as ‘ignored’ in F' , we find that any solution to F has a matching solution to F' . Moreover, the optimal solution to F' does not label ‘ignored’ for any test that is already consistent, since any solution doing so could be improved by marking that test as ‘not ignored’. Hence, in any optimal solution to F' , replacing ‘ignored’ by ‘flipped’ gets us to a matching solution to F . We proceed by formalizing this intuition.

Proof. Since F and F' differ only in the hard clauses C_i and C'_i (similarly \tilde{C}_i and \tilde{C}'_i), and an optimal assignment of a MaxSAT formula always satisfies all hard clauses, by Lemma 4.2.1

$$\sigma \underset{\text{opt}}{\models} F' \iff \sigma \underset{\text{opt}}{\models} F$$

can be proved if we prove that

$$\begin{aligned} \sigma \models_{\text{opt}} F' &\Rightarrow \sigma \models C_i \\ \sigma \models_{\text{opt}} F &\Rightarrow \sigma \models C'_i \end{aligned} \tag{4.2.3}$$

and

$$\begin{aligned} \sigma \models_{\text{opt}} F' &\Rightarrow \sigma \models \tilde{C}_i \\ \sigma \models_{\text{opt}} F &\Rightarrow \sigma \models \tilde{C}'_i \end{aligned} \tag{4.2.4}$$

We can prove 4.2.3 using a proof by contradiction. Let be σ an assignment such that $\sigma \models_{\text{opt}} F'$ and $\sigma \not\models C_i$;

we use logical equivalence

$$A \oplus B \Leftrightarrow (A \vee B) \wedge (\neg A \vee \neg B)$$

on C_i to attain

$$C_i := \left(\left(\bigvee_{j|\mathbf{A}_{i,j}=1} x_j \right) \vee \xi_i \right) \wedge \left(\neg \left(\bigvee_{j|\mathbf{A}_{i,j}=1} x_j \right) \vee \neg \xi_i \right).$$

Expanding the definition of

$$C'_i := \left(\bigvee_{j|\mathbf{A}_{i,j}=1} x_j \right) \vee \xi_i$$

we consider two distinct case, when ξ_i is assigned by σ to true and when ξ_i is assigned to false.

When $\sigma \models \neg \xi_i$, then, as $\sigma \models C'_i$, σ assign at least one x_j constrained in C'_i to true, hence $\sigma \models C_i$ as both part of the disjunction are satisfied, **contradiction**.

Now recall that F' has the soft clause $S_j := \neg x_j$ and $N_i := \neg \xi_i$ and that ξ_i is constrained only by C'_i .

When $\sigma \models \xi_i$, as σ is optimal to F' , it is an absurd that σ

satisfy both ξ_i and $\bigvee_{j|\mathbf{A}_{i,j}=1} x_j$, because it will exist a trivial assignment $\widehat{\sigma}$ such that $\text{wt}(\widehat{\sigma}) > \text{wt}(\sigma)$, and σ will not be optimal, **contradiction**.

Hence $\sigma \models \neg(\bigvee_{j|\mathbf{A}_{i,j}=1} x_j)$ and $\sigma \models C_i$, which is a contradiction for either assignment of ξ_i . Therefore,

$$\sigma \underset{\text{opt}}{\models} F' \Rightarrow \sigma \models C_i$$

Note that the above holds as we are considering separately each ξ_i , for both positive and negative assignment, which are constrained only by one and exclusive Hard constraint (C'_i).

We now prove $\sigma \underset{\text{opt}}{\models} F \Rightarrow \sigma \models C'_i$ using a proof by contradiction.

Let $\sigma \underset{\text{opt}}{\models} F$ and $\sigma \not\models C'_i$. When $\sigma \models C_i$, either $\sigma \models \xi_i$ or $\sigma \models (\bigvee_{j|\mathbf{A}_{i,j}=1} x_j)$. Hence $\sigma \models C'_i$, which is a **contradiction**. Therefore, $\sigma \underset{\text{opt}}{\models} F \Rightarrow \sigma \models C'_i$, which ends the proof of claim

4.2.3.

We can prove 4.2.4 with similar reasoning and to avoid redundancy we don't include such proof in this section. Therefore, recalling Lemma 4.2.1

$$\sigma \underset{\text{opt}}{\models} F \iff \sigma \underset{\text{opt}}{\models} F'$$

QED

4.3 Optimality of $\widehat{\mathcal{K}}_{SSS}$ and λ Trade-Off

$\widehat{\mathcal{K}}_{SSS}$ is an estimate to the set \mathcal{K} given a group testing problem and is important to prove the essential optimality of such estimate in the Bernoulli testing. In the objective function for the noisy setting in Eq. 3.2.1, the parameter λ decides the trade-off between the number of defective items and the number of inverted tests. In

our MaxSAT encoding λ is the weight associated with soft clauses describing noise, as depicted in section 4.1.2. This tradeoff lead the solution between sparsity and noisiness of tests and the value of this parameter must be optimal for the considered setting.

4.3.1 Overview

We consider the decoding in the Bernoulli random design (see section) 3.1.4, this random design does not require the experimenter to understand and accurately implement tricky combinatorial designs, as it does not necessarily require accurate knowledge of the number of defectives, or how many tests will be performed. Furthermore, recent work [AS12] has shown that the $\text{Be}(q)$ design is asymptotically close to optimal when $k \ll n$. Since we model both defectivity and noise using i.i.d. Bernoulli trials in the stochastic model, we can set the appropriate value of λ from the associated probability p and d .

4.3.2 Background

Suppose each item is defective independently with probability $p < 0.5$ and each test gets inverted independently with probability $d < 0.5$ (symmetric Bernoulli noise, see Definition 2.2.3). Then for a candidate defective set \mathbf{x} and recovered set $\hat{\mathbf{x}}$, finding the optimal solution requires minimizing $\mathbb{P}[\hat{\mathbf{x}} \neq \mathbf{x}]$, which is equivalent to maximizing the posterior probability $\mathbb{P}[\mathbf{x}|\mathbf{A}, \mathbf{y}]$ [ABJ14], in the noisy settings we denote the posterior as $\mathbb{P}[\mathbf{x}, \boldsymbol{\xi}|\mathbf{A}, \mathbf{y}]$.

4.3.3 Derivation

By definition of $\mathbb{P}[\mathbf{x}, \boldsymbol{\xi} | \mathbf{A}, \mathbf{y}]$:

$$\mathbb{P}[\mathbf{x}, \boldsymbol{\xi} | \mathbf{A}, \mathbf{y}] = \frac{\mathbb{P}[\mathbf{x}, \boldsymbol{\xi}, \mathbf{A}, \mathbf{y}]}{\mathbb{P}[\mathbf{A}, \mathbf{y}]} \quad (4.3.1)$$

In the decoding phase \mathbf{A} and \mathbf{y} are known and fixed, so being \mathbf{x} and $\boldsymbol{\xi}$ vectors of independent variables we know the proportionality of this posterior:

$$\frac{\mathbb{P}[\mathbf{x}, \boldsymbol{\xi}, \mathbf{A}, \mathbf{y}]}{\mathbb{P}[\mathbf{A}, \mathbf{y}]} \propto \mathbb{P}[\mathbf{x}] \mathbb{P}[\boldsymbol{\xi}] \quad (4.3.2)$$

Remember that $k(\mathbf{x})$ is the number of defectives. Then being $\tau(\mathbf{x}, \mathbf{A}, \mathbf{y})$ the Hamming distance between the output \mathbf{y} and the expected output that one would get if there were no noise, that is, the number of inverted tests. In order to get a tighter notation we can write $\tau(\mathbf{x}, \mathbf{A}, \mathbf{y}) = \rho(\boldsymbol{\xi}) = \sum_{i=1}^m \xi_i$. Then by the definition of $\mathbb{P}[\mathbf{x}]$ and $\mathbb{P}[\boldsymbol{\xi}]$:

$$\begin{aligned} \mathbb{P}[\mathbf{x}] &= p^{k(\mathbf{x})} (1-p)^{n-k(\mathbf{x})} \\ \mathbb{P}[\boldsymbol{\xi}] &= d^{\rho(\boldsymbol{\xi})} (1-d)^{m-\rho(\boldsymbol{\xi})} \end{aligned}$$

and then

$$\begin{aligned} \mathbb{P}[\mathbf{x}] \mathbb{P}[\mathbf{y}] &= \\ p^{k(\mathbf{x})} (1-p)^{n-k(\mathbf{x})} d^{\rho(\boldsymbol{\xi})} (1-d)^{m-\rho(\boldsymbol{\xi})} &= \\ = (1-p)^n \left(\frac{p}{1-p} \right)^{k(\mathbf{x})} (1-d)^m \left(\frac{d}{1-d} \right)^{\rho(\boldsymbol{\xi})} \end{aligned}$$

hence, being term $(1-p)^n$ and $(1-d)^m$ constant with respect to \mathbf{x} and $\boldsymbol{\xi}$,

$$\mathbb{P}[\mathbf{x}, \boldsymbol{\xi} | \mathbf{A}, \mathbf{y}] \propto \left(\frac{p}{1-p} \right)^{k(\mathbf{x})} \left(\frac{d}{1-d} \right)^{\rho(\boldsymbol{\xi})} \quad (4.3.3)$$

Remember that we are looking for the Maximum a-posteriori Probability,

$$\max_{\mathbf{x}, \boldsymbol{\xi}} \left(\frac{p}{1-p} \right)^{k(\mathbf{x})} \left(\frac{d}{1-d} \right)^{\rho(\boldsymbol{\xi})} \quad (4.3.4)$$

so applying the log we get:

$$\log \left(\left(\frac{p}{1-p} \right)^{k(\mathbf{x})} \left(\frac{d}{1-d} \right)^{\rho(\boldsymbol{\xi})} \right) =$$

$$k(\mathbf{x}) \log \left(\frac{p}{1-p} \right) + \rho(\boldsymbol{\xi}) \log \left(\frac{d}{1-d} \right)$$

taking the negative of the log we get a minimization:

$$\min_{\mathbf{x}, \boldsymbol{\xi}} \log \left(\frac{1-p}{p} \right)^{k(\mathbf{x})} + \log \left(\frac{1-d}{d} \right)^{\rho(\boldsymbol{\xi})}$$

normalizing:

$$\min_{\mathbf{x}, \boldsymbol{\xi}} k(\mathbf{x}) + \frac{\log \left(\frac{1-d}{d} \right)}{\log \left(\frac{1-p}{p} \right)} \rho(\boldsymbol{\xi}) \quad (4.3.5)$$

Recall Eq. 3.2.1:

$$\min \sum_{j=1}^n x_j + \lambda \sum_{i=1}^m \xi_i$$

by the definition of $k(\mathbf{x})$ and $\rho(\boldsymbol{\xi})$ we can argue that solve this problem is optimal to minimize probability $\mathbb{P}[\hat{\mathbf{x}} \neq \mathbf{x}]$ and we have now the optimal λ , maximizing the posterior probability $\mathbb{P}[\mathbf{x}, \boldsymbol{\xi} | \mathbf{A}, \mathbf{y}]$.

$$\lambda = \frac{\log \left(\frac{1-d}{d} \right)}{\log \left(\frac{1-p}{p} \right)} \quad (4.3.6)$$

In that sense compute the $\hat{\mathcal{K}}_{SSS}$ is optimal in the noisy settings under a combinatorial bernoulli prior and with symmetric bernoulli noise with the above λ . The extension to noiseless settings is trivial and is not reported to avoid redundancy. By the previously mentioned work [BJA13], under the assumption that $k \ll n$ our model is asymptotically optimal.

Chapter 5

Experimental Results and Evaluation

*“Through the day
As if on an ocean
Waiting here,
Always failing to remember
why we came, came, came
I wonder why we came”*

Brian Eno - By This River

5.1 Experiment Preliminaries

We now discuss preliminaries of our experimental evaluation; in particular we illustrate the implementation, the setup of instances,

the evaluation metrics and the question that we aim to answer.

5.1.1 About the Implementation of Algorithms

We have developed a prototype implementation of MGT to solve the decoding phase of non-adaptive group testing in both the noiseless and noisy settings. All the algorithms has been developed in the same Python 3.6 framework. In the implementation of our novel algorithm (MGT), we employ MaxHS [DB11] as the underlying MaxSAT solver. Not that MaxHS is not the best performing state of the art solver, so in theory the performance of our model can be even higher, we used this solver as it allow us to give a timeout to the solver and output the best solution found at the time.¹ We compare MGT with a state-of-the-art approach namely, the approximated linear programming relaxation approach (LP) that also solves the decoding phase of non-adaptive group testing [MM12]. We compare to the LP relaxation approach, implemented using CPLEX by IBM² as the underlying solver. The formulation of the LP relax is as defined in section 3.2.4 and when a non-integer solution is found we assign $\hat{x} = 1$ when $\hat{x} \geq 0.5$, 0 otherwise. We set the cut-off time of both LP and MaxHS solvers to be 100 seconds. If an optimal solution is not found within the cut-off time, MaxHS returns the current best solution. We've also implemented COMP, DD and SCOMP (see section 3.2), which can solve the noiseless case but not the noisy, and we have done few experiment with these compared to our

¹In the noiseless settings such feature was never triggered as the time complexity was far less than the set timeout (100 seconds).

²This makes the comparison even more relevant as also MaxHS uses CPLEX library to solve sub-problems in the solving

algorithm. LP is the main rival of our implementation as it's the best in approximate $\widehat{\mathcal{K}}_{\mathcal{SS}}$ [MM12]. First round of experiments has been conducted on machine in concession by the National Supercomputer Center of Singapore. The definitive round of experiment has been conducted on a local machine equipped with Intel core i7(3.4 GHz) and 8 GB of RAM. Every instances to be solved has a memory limit for each algorithm of 2 GB and the instances has been run sequentially. The results reported on this work are from the definitive round of experiments.

5.1.2 Setup of Instances

We uses stochastic model as defined in Section 3.1.4 with a combinatorial prior to group testing candidate set, symmetric bernoulli noise and bernoulli model for the testing matrix \mathbf{A} . To model a group testing instance in general we proceed as follows.

- choose the number of defective items k for a fixed number of items n defining a ratio variable $\beta = \frac{k}{n}$ ($\beta \in [0.01, 0.1]$)
- Then the item vector \mathbf{x} is generated with $k = \beta n$ defective items.
- consider a Bernoulli process with probability $q = (\log 2)/k$ to construct measurement matrix \mathbf{A} ³
- in the noisy setting, we consider another Bernoulli process

³This is known to be the best probability to be set for Bernoulli testing, even if this probability consider k to be known in general our model doesn't need such information. This value is used as the best to benchmark/comparison purposes.

with probability $d < 0.5$ to generate the noise vector ξ ; in the reported experiment $d = 0.05$

- recover a solution and compute metrics
- repeat the experiment for $l = 100$ trials to ensure statistical consistency.

5.1.3 Evaluation Metrics

We evaluate accuracy of algorithm (MGT, LP, COMP, DD, SCOMP) based on the Hamming distance and probability of success, which are defined below.

- **Hamming distance:** Given an item vector \mathbf{x} and the recovered solution $\hat{\mathbf{x}}$, the Hamming distance $h(\mathbf{x}, \hat{\mathbf{x}})$ is the number of items that are wrongly detected in either of the two ways: a non-defective item detected as defective, or a defective item detected as non-defective. Formally

$$h(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{j=1}^n \mathbb{1}(x_j \neq \hat{x}_j)$$

- **Probability of success $\hat{\mathbb{P}}_s$:** The probability of success is defined as the probability of attaining zero Hamming distance in the recovered solution. Formally,

$$\hat{\mathbb{P}}_s = \frac{\sum_{i=1}^l \mathbb{1}\{h(\mathbf{x}_i, \hat{\mathbf{x}}_i) = 0\}}{l}$$

where \mathbf{x}_i is the randomly generated item vector at the i -th trial and $\hat{\mathbf{x}}_i$ is the associated recovered solution. $\hat{\mathbb{P}}_s$ is an unbiased estimator of \mathbb{P}_s , see definition 3.1.16.

5.1.4 Question Answered

The main purpose of the experimental evaluation is to answer the following questions during the evaluation of both MGT and other algorithms.

1. How does MGT scale with respect to the number of items n , the number of defective items k , and the number of tests m ?
2. How does the accuracy and runtime performance of MGT compare to existing state-of-the-art approaches?
3. Do we observe any interesting behavior of the runtime of MGT?
4. Does MGT follow known bounds on the number of tests for recovery?
5. How efficient is the compact encoding compared to the naive encoding in the noisy setting?

5.1.5 Adoption of Bounds

The black line in each graph denotes the bound on m that allows non-zero \mathbb{P}_s (see Definition 3.1.16). For the noiseless settings the adopted bound is the asymptotic version of the Counting Bound 3.1.1.1, so the threshold on m in the noiseless setting [BJA13, AJS19]:

$$\tilde{m} = \log_2 \binom{n}{k} \quad (5.1.1)$$

For the noisy settings the adopted bound is the asymptotic version of the bound proposed by [Joh17] (see Theorem 3.1.2.1) where d

is the probability on noisy tests, and

$$H(d) = -d \log_2 d - (1 - d) \log_2(1 - d)$$

is the binary entropy. So the threshold on m in the noiseless setting

$$\tilde{m}_n = \frac{k \log\left(\frac{n}{k}\right)}{\log 2 - H(d)} \quad (5.1.2)$$

5.2 Empirical Results on Accuracy

The first outcome on which we focus is the accuracy in terms of \mathbb{P}_s . We compare the proposed algorithm MGT with the implementation of COMP, DD, SCOMP and in particular LP. Figure 5.2.1 anticipates the results about accuracy. MGT outperform all the algorithm and LP is the closest one in term of $\hat{\mathbb{P}}_s$. For an extended comparison LP is chosen as a test-bed for MGT.

5.2.1 Varying the Number of Items:

To start we observe whether MGT follows the theoretical bound on the number of tests m for non-zero probability of success ($\hat{\mathbb{P}}_s > 0$) in both noiseless and noisy settings. In Figure 5.2.2, we show graphs where we vary m and plot the corresponding $\hat{\mathbb{P}}_s$ for different choices of the number of items n . For each choice of n , we set the number of defective items with $\beta = 0.03$. In all graphs, we find that empirically $\hat{\mathbb{P}}_s$ becomes non-zero after m exceeds the theoretical bound in both MGT and LP. Moreover, $\hat{\mathbb{P}}_s$ increases and becomes closer to 1 as m increases and becomes closer to n . As we observe more closely, we find that in the noiseless setting, $\hat{\mathbb{P}}_s$ quickly becomes 1, whereas more tests are required to reach the same level of $\hat{\mathbb{P}}_s$ when we consider the noisy setting. We

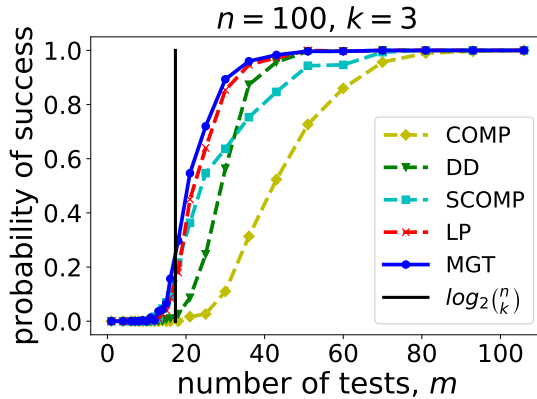


Figure 5.2.1: Accuracy trend, with respect to m , for all the implemented algorithms with a really simple, in terms of scalability, test case.

note that being the ratio β fixed in this experiment the shape of the curve remain qualitatively invariant and the proximity with the bounds remain proportionally the same. This observation will be extended in combination with other experiments later. In terms of accuracy, MGT and LP show a similar performance in the noiseless setting. However, MGT outperforms LP in the noisy setting, and the difference is more for higher values of n . This result suggests the effectiveness of MGT over LP in terms of the quality of the recovered solutions and so the calculation of non-approximate solution of $\hat{\mathcal{K}}_{SSS}$. Being $\beta = 0.03$ a sparse regimen we also experimentally verify that our model is asymptotically optimal with sparsity assumption, as probability $\hat{\mathbb{P}}_s$ becomes non-zero with $m = \tilde{m} + \epsilon$ and $\epsilon \approx 0$.

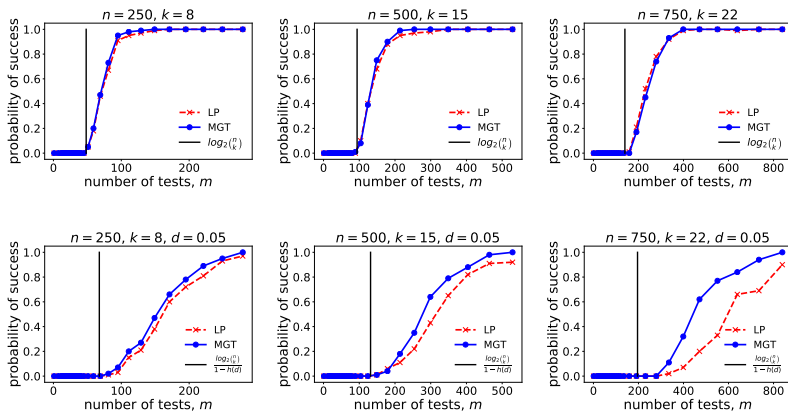


Figure 5.2.2: Effect of the number of items n on the probability of success in both the noiseless (figures in the top row) and noisy (figures in the bottom row) settings. In each graph $k = 0.03n$, and for the noisy setting, $d = 5\%$. For $n = 1000$ in the noisy settings LP can not recover solutions due to a memory-limit error, anyway with the set timeout recovery success of MGT was not relevant.

5.2.2 Varying the Number of Defective Items:

In Figure 5.2.3, we vary the number of defective items with respect to $\beta \in [0.01, 0.1]$ and show its effect on the probability of success in both the noiseless and the noisy settings⁴.

Empirically we find that for fixed number of items n , as we increase the number of defective items k , more tests are required to reach the same level of the probability of success in both MGT and LP. In addition, when β is much higher (≈ 0.1), 100% recovery often becomes unachievable for random Bernoulli tests even if $m = n$ in both the noiseless and noisy settings. Remember that in section 5.2.1 we observed that with a fixed β the function $\widehat{\mathbb{P}}_s(m)$ remain qualitatively the same with respect to n . Recalling also the definition of the considered lower bound (see theorem 3.1.1, 3.1.2) we can say that in the Bernoulli setting the ratio β characterize the possibility to extend the lower bound to be a precise threshold which trigger $\mathbb{P}_s > 0$. In other word, being β a measure of the sparsity of \mathbf{x} , we empirically show that the near-optimality of Bernoulli design is determined by β , such near optimality dramatically disappears with $\beta \rightarrow 0.1$.

5.2.3 Scalability Analysis:

We experiment with higher values of the number of items n to observe the scalability performance of both MGT and LP in the noiseless setting, and show the result in Figure 5.2.4 with 3% defective items. Both LP and MaxSAT solvers are given equal computation resources, but LP reached the shared memory

⁴We show here the plot for the two extreme value, we put additional plots in the Appendix

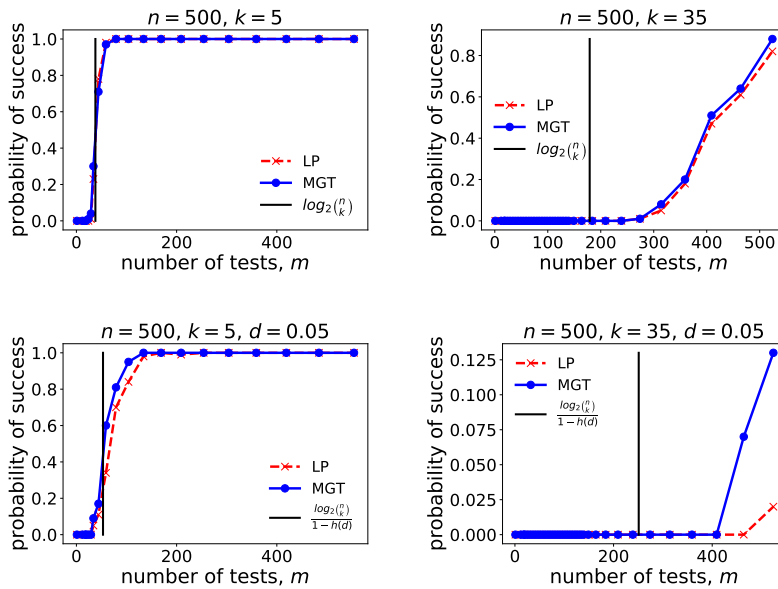


Figure 5.2.3: Effect of the number of defective items k on the probability of success in both the noiseless (figures in the top row) and noisy (figures in the bottom row) settings. In each graph, $n = 500$ and for the noisy setting, $d = 5\%$.

limit when $n > 2000$. Interestingly, we find that MGT shows an impressive scalability for higher values of n . In particular, we run MGT for $n = 16000$ and $\beta = 0.01$ successfully, and it can potentially go further as the timeout was never triggered in the noiseless settings. We have also experimented with more defective items $\beta > 0.05$ for up to $n = 10000$, where the accuracy of MGT starts deteriorating even if we allow more tests. To conclude, this scalability result shows the promise of applying MaxSAT solvers in various practical combinatorial optimization problems.

5.2.4 Efficiency of the Compact Encoding:

In Figure 5.2.5, we show the runtime performance of both the compact encoding and the naive encoding (encoding with XORs) on the similar choice of parameters. We empirically find that the runtime of the naive encoding is higher than that of the compact encoding as we consider more tests. Moreover, for higher values of n and k , the naive encoding often becomes intractable. This result suggests that the compact encoding not only provides a theoretical guarantee of optimality (see Section 4.2), but also is efficient in practice than the naive encoding.

5.3 Runtime Analysis and Phase Transition

We compute the average runtime of all trials, and in Figure 5.3.1, we present it while varying the number of tests m for both the noiseless and the noisy settings (see also Figure B.0.6). In all graphs, we find that both MGT and LP require more runtime in the noisy setting than in the noiseless setting, which points

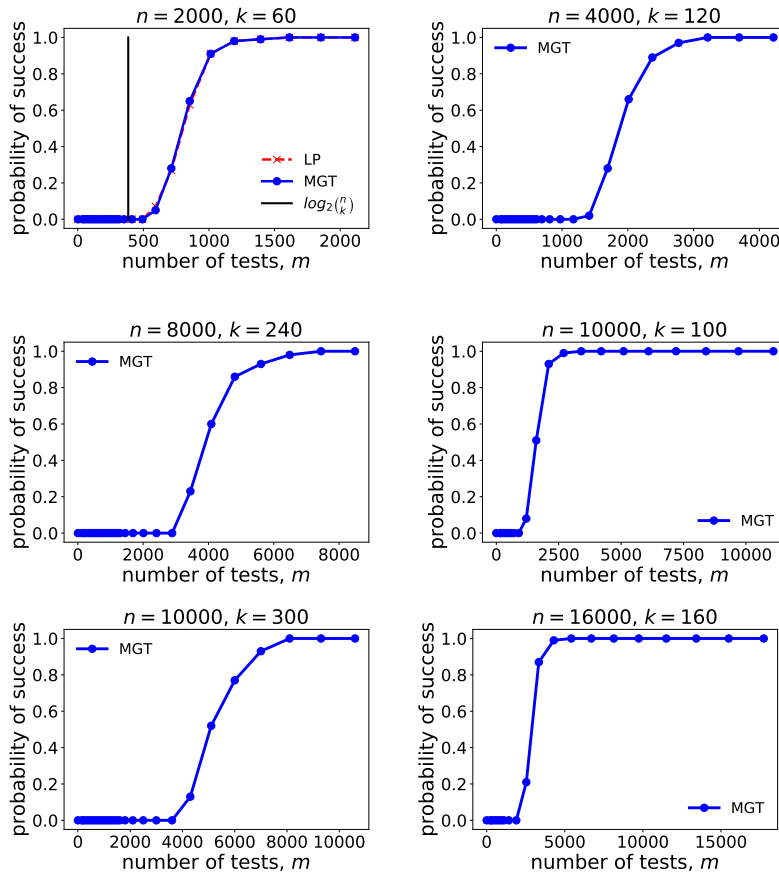


Figure 5.2.4: Probability of success for increasing values of n in the noiseless setting. In each graph, $k = 0.03n$. Due to memory limit of 2 Gigabyte⁵ LP fail to compute solution for $n > 2000$. Such memory limitation is the same for both algorithm.

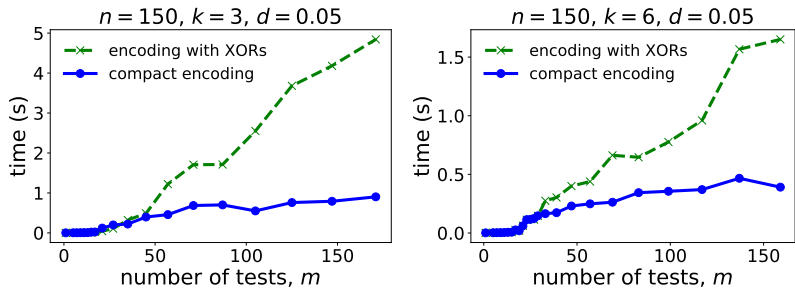


Figure 5.2.5: Runtime performance of the compact encoding and the naive encoding for the noisy setting of non-adaptive group testing.

out the effect of noise on the computation process. Moreover, LP takes less time than MGT as m increases in both settings, and the difference is more in the noisy setting. MGT reveals an exponential behaviour with respect to m in the noisy settings. In this context, LP applies relaxation in the integer programs to achieve performance boost in terms of runtime and finds an approximate solution. However, in this MaxSAT-based formulation, the MaxSAT solver finds an optimal solution while costing relatively higher runtime. The noisy settings runtime reveal the limit of MGT, in fact as we have noise the increasing of number of tests m , to which follows an increasement of boolean variable in the encoding, determine an exponential time behavior which is not observed in LP, even if LP undergoes memory limit error⁶. Compact encoding (see Section 4.2) relieve such problem, as in the XOR encoding with $n > 200$ solutions are never feasible, but we are still not able to compute solution with $n > 2000$ with noise

⁶Such limit is shared for both MGT and LP

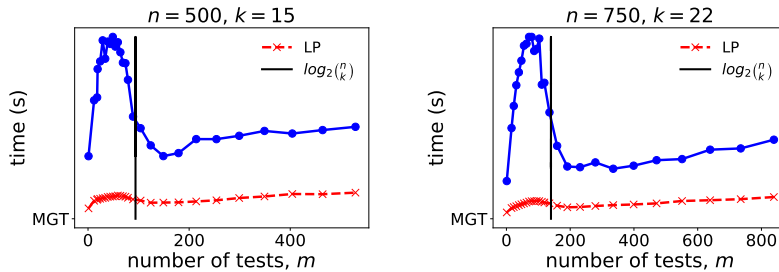


Figure 5.3.1: Computation time in the decoding phase of group testing in both the noiseless settings. In each graph, $k = 0.03n$.

considering a timeout of 100 seconds as done in the experiment. We hope to overcome this problem with the adoption of more powerful MaxSAT solver.

5.3.1 Noiseless Phase Transition

In the noiseless setting, we observed a phase transition of accuracy in both MGT and LP with easy-hard-easy behaviour of runtime. Specifically, the runtime increases until reaching a peak near the theoretical bound on m , then decreases quickly, and again increases as m increases, but never with exponential behavior. In fact the peak reached around \tilde{m} is never approached again. This observation reveals the easy-hard-easy nature, at least in its interpretation as a combinatorial optimization problem, of group testing, where the difficulty of a problem instance changes due to changing the number of tests m . The nearness with theoretical bound on tests denotes that the probability of success also goes from zero to non-zero during the easy-hard-easy transition. This

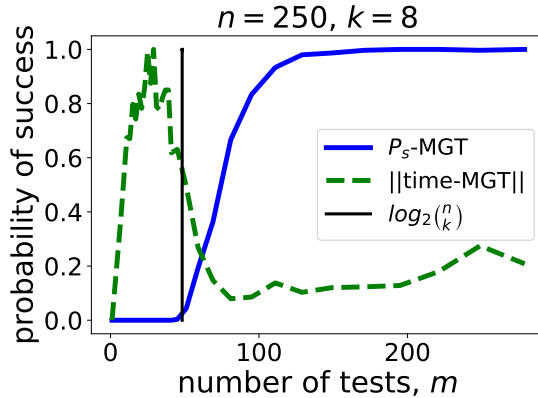


Figure 5.3.2: Noiseless Group Testing phase transition, in blue the estimate of \mathbb{P}_s and in dotted green runtime.

behaviour is similar to the phase transition observed in the SAT problem of random k -CNF formulas [COP13, CR92, DSS15] previously presented in Section 2.5. A random k -CNF formula (each clause in the formula has at most k literals) goes from SAT to UNSAT as the number of clauses increases, and the computation time near the transition gives a similar trend to what we have observed here (remember figure 2.5.1). Although phase transition is known in SAT instances, in our study we find that the decoding phase of noiseless non-adaptive group testing shows similar characteristics. It is evident that we can connect \mathbb{P}_s to $\mathbb{P}(SAT|X)$ probability of SAT (note that $\mathbb{P}(SAT|X)$ goes from 1 to 0 while \mathbb{P}_s goes from 0 to 1), m (number of tests) to \mathcal{L} number of clauses and so on for all the interested parameter. The really interesting thing is that time is observed for SAT-solving and MaxSAT-solving, so we are comparing time for two strongly

related problem, while probability of SAT and probability of success of decoding of Non-Adaptive Group Testing belongs to really different domain and the connection between them is really fascinating. The first is an intrinsic property of a formula that is always decidable, while the second is the effective capability of reconstructing information after a compression is performed. All the noiseless study reveal that the bell that determine the transition Easy-Hard and Hard-Easy stands on the "area" before \tilde{m} , so before $\widehat{P}_s > 0$ as reported in Figure 5.3.2 and Figure B.0.5. This phase transition behavior should be deeply investigate, better understood as it could build a strong bridge between optimization problem, SAT and compression theory helping on developing the understanding of them.

Table 5.3.1 report the number of tests which correspond to the peak of runtime varying n and the defective/item ratio β , we denote this specific number of tests with m_c , which can be paired with the α_c of random k -SAT problem (see Section 2.5.1). Surprisingly with fixed β the threshold m_c remain constant, and starts to depend from n with higher value of the ratio β . This changing of behaviour is similar to the validity of the lower bound as a threshold to trigger probability of success $P_s > 0$ observed in Section 5.2.2 and also the constancy with respect to n starts to disappear with the lessening of sparsity assumption.

5.3.2 Understanding Runtime Behaviour and \widehat{C} Estimate

In order to better understand this behaviour we ask ourself the following question: *Is this behaviour a consequence of the solving technique or is an intrinsic characteristic of the problem?* From

	$\beta = 0.01$	$\beta = 0.02$	$\beta = 0.03$	$\beta = 0.04$	$\beta = 0.05$
$n = 100$	$0.16n$	$0.27n$	$0.38n$	$0.49n$	$0.54n$
$n = 200$	$0.17n$	$0.27n$	$0.39n$	$0.5n$	$0.55n$
$n = 300$	$0.173n$	$0.273n$	$0.393n$	$0.503n$	$0.553n$
$n = 400$	$0.17n$	$0.28n$	$0.39n$	$0.51n$	$0.63n$
$n = 500$	$0.168n$	$0.278n$	$0.388n$	$0.508n$	$0.638n$
$n = 600$	$0.167n$	$0.277n$	$0.397n$	$0.507n$	$0.637n$
$n = 700$	$0.171n$	$0.281n$	$0.391n$	$0.511n$	$0.631n$
$n = 800$	$0.171n$	$0.281n$	$0.391n$	$0.511n$	$0.641n$
$n = 900$	$0.171n$	$0.281n$	$0.391n$	$0.511n$	$0.641n$
$n = 1000$	$0.171n$	$0.281n$	$0.391n$	$0.511n$	$0.641n$

Table 5.3.1: Number of tests, expressed as a fraction of n , which corresponds to the peak of runtime for different (n, β) . Data are collected from MGT.

Figure 5.3.3 and Figure B.0.4 we learn that also LP and SCOMP undergo the easy-hard-easy behaviour and such transition follow MGT runtime. The fact that COMP and DD does not manifest such trend can be easily motivated observing that both does not exploit all the information available and, when the information is not trivial, they simply ignore it and output a largely approximate solution. Recalling the difference between SCOMP and DD (see Section 3.2.2) it's easy to motivate the fact that SCOMP, after the phase transition behaviour, has the same runtime of DD and this indicate that the region with the more *unexplained test* by DD (see Section 3.2.2 for a discussion over their differences) is the one under the bell and around m_c . Since the main cause of unexplained test is the presence of masked defectives (see

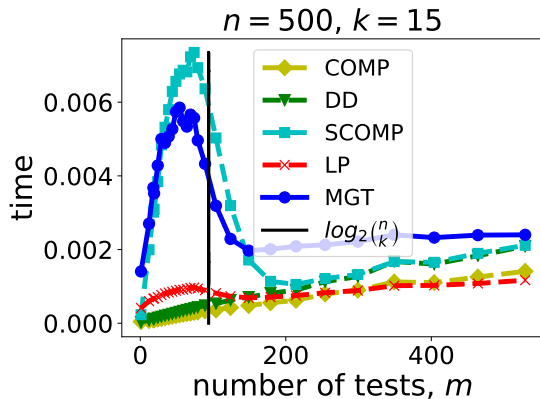


Figure 5.3.3: Noiseless runtime trend in seconds reported for all the algorithm

Definition 3.1.15), we derive that Bernoulli random design provide an high number of masked defectives in the range of m underlying the bell of the Easy-Hard-Easy transition. We observe then that any algorithm that tries to learn the most information available will find it harder around the threshold m_c , where the intrinsic nature of information becomes Hard to decompress, so we make the following conjecture.

Conjecture 5.3.1 (Hardness Threshold). *Given a group testing instance with n items, $k = \beta n$ defectives and m tests, there exists a m_c such that, being $m_p > \tilde{m}$ the minimum number of tests such that $\mathbb{P}(\text{succ}) > 0$ and \mathcal{A} the algorithm that achieves the highest possible rate $\Delta'(n, k, m_p)$, then*

$$c_{\mathcal{A}}(\beta, m_c) \geq c_{\mathcal{A}}(\beta, m) \quad \forall m \in \{1, n\} \quad (5.3.1)$$

where $c_{\mathcal{A}}(\beta, m)$ is the time required by algorithm \mathcal{A} to output a

solution for such instance.

In particular we claim that, under bernoulli testing assumption being fixed p , q and n

$$m_c = \operatorname{argmax}_{m \in \{1, n\}} (m\beta(1 - q(1 - pq)^{n-1})^m) \quad (5.3.2)$$

The last part of the conjecture derives from the C -estimate definition 5.3.2.

Remark. This conjecture tries to conceptualize all the observations over the runtimes of the implemented algorithms. The concept that we draw is that, given the whole range of possible number of test $r = \{1, n\}$, under some sparsity regimen β , there is a sub-interval $r' \subset r$ in which the problem becomes harder because of the nature of the information. In particular any accurate enough algorithm which tries to obtain the most from the available information, so tries to achieve the highest possible rate Δ' (see Definition 3.1.17) where $\mathbb{P}_s > 0$ is possible, will manifest an Easy-Hard-Easy transition with a peak of Hardness corresponding to the critical m_c .

The presence of masked defectives under the hardness bell, observed, suggest us to derive the probability of an item to be a masked defective.

Definition 5.3.1 (Masked Defective Probability). Given a group testing instance with \mathcal{K} sampled from a Bernoulli process $Be(\beta, n)$. Being the pooling matrix \mathbf{A} of a Bernoulli design $Be(q, m)$, then the probability \mathbb{P}_{md} to pick up an item which is a masked defective (see Definition 3.1.15) is:

$$\mathbb{P}_{md} = \beta((1 - q(1 - pq)^{n-1})^m) \quad (5.3.3)$$

We can interpret this probability as the hardness to decode each single test. From such probability we present an intuitive estimate of the complexity of decoding information of a non-adaptive group testing instances, which is the normalization of \mathbb{P}_{md} weighted by number of tests m . The main purpose of this estimate is to give an interpretation to the phase transition behaviour.

Definition 5.3.2 (C-Estimate). Given a noiseless decoding instance of non-adaptive group testing, with \mathcal{K} sampled from a Bernoulli process $Be(\beta, n)$ and pooling matrix \mathbf{A} designed with a Bernoulli design $Be(q, m)$. Being \mathcal{A} an algorithm which achieves the best achievable rate for $m_p > \tilde{m}$, then we define the C-Estimate:

$$\widehat{C}_{\mathcal{A}} = \left\| m\beta(1 - q(1 - pq)^{n-1})^m \right\|_m \quad (5.3.4)$$

where

$$\|f(x)\|_x = \frac{f(x) - f_{min}}{f_{Max} - f_{min}}$$

Such estimate is normalized to 1, where 1 represents the maximum hardness to decode an instance and output an estimate $\widehat{\mathcal{K}}_{\mathcal{A}}$ with respect to m .

We propose a supplementary discussion over C-Estimate in Section A.2. In Figure 5.3.4 we present some empirical results which sustain the conceptual correctness of the intuition behind $\widehat{C}_{\mathcal{A}}$. For each instance We set $p = \beta$ and $q = \frac{\log 2}{k}$ obtaining

$$\widehat{C}_{\mathcal{A}} = \left\| m\beta \left(1 - \frac{\log 2}{k} \left(1 - \frac{\log 2}{n} \right)^{(n-1)} \right)^m \right\|_m$$

In fact $\widehat{C}_{\mathcal{A}}$ qualitatively predict the trend of runtime for both SCOMP and MGT, while LP manifest a slightly different behaviour

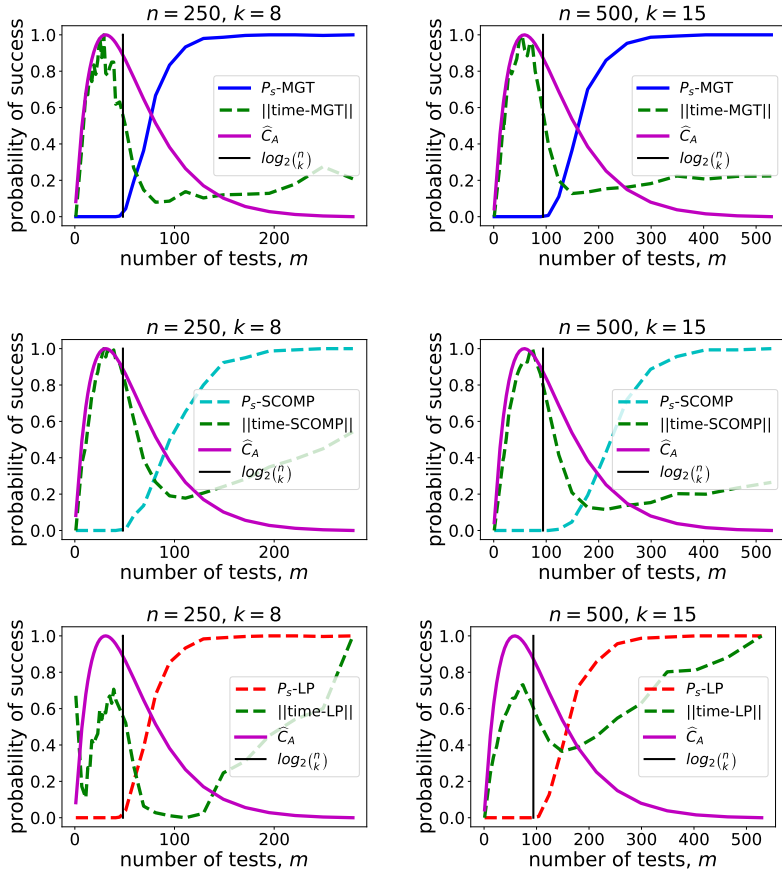


Figure 5.3.4: \hat{C}_A estimation compared with real time complexity of algorithm: from top to bottom MGT, SCOMP, LP

and reach its maximum hardness after the threshold m_c even if it also manifest easy-hard-easy behaviour around such threshold. Recall that we define $\widehat{C}_{\mathcal{A}}$ to estimate the trend of hardness for the best algorithm \mathcal{A} (i.e. the algorithm that achieves the best rate Δ'), in this sense it is correct that MGT is better characterized by $\widehat{C}_{\mathcal{A}}$. To compare the estimate $\widehat{C}_{\mathcal{A}}$ to the empirical results we normalize the time experiment to 1, where 1 is the maximum. Picturing $\widehat{C}_{\mathcal{A}}$ we have given an intuitive meaning to the Easy-Hard-Easy phase transition and we believe that a deeper understanding of such behaviour and its theoretical study promise to build bridge between computer science problems and unveil some aspect of the nature of computation.

Chapter 6

Conclusions and Future Research Direction

“Last and least is me. Mingus. I wrote the music for dancing and listening. It is true music with much and many of my meanings. It is my living epitaph from birth til the day I first heard of Bird and Diz. Now it is me again.”

Charles Mingus

In this work, we presented MGT, a novel MaxSAT-based formulation for solving the decoding phase of non-adaptive group testing in both the noiseless and the noisy settings. For the noisy setting, we proposed a compact encoding with a proof of soundness, claiming to decrease runtime following a reduction of clauses by leveraging the objective function. We have derived the optimality of our model and found a theoretically optimal value of the trade-off between noisiness and faultiness (parameter λ)

for Bernoulli Testing. We largely investigate the behaviour of our algorithm and compare it with other state of the art solutions. To summarize our empirical study, MGT, following known bound on the number of tests for recovery (see Section 3.1.5), shows an impressive scalability and can solve group testing instances with up to $n = 16000$ and $\beta = 0.03$ in the noiseless setting, which no prior work can handle to the best of our knowledge and in fact none of the comparing algorithms achieve. The experiment over noisy settings prove our claim and show that the compact encoding outperforms the naive encoding w.r.t. runtime by a large margin, which highlights the practical applicability of the compact encoding with added optimality guarantee. In the noisy settings MGT largely outperforms LP in terms of accuracy, however this setting denote a flaw, partially solved with the compact encoding, as with increasing m the instances becomes exponentially harder in terms of time. We hope in future works to experiment with different MaxSAT solver than MaxHS and to achieve larger scalability in noisy settings. Anyway the spatial complexity support the usage of MaxSAT as LP fails with $n > 1000$ in the noisy settings due to memory limit. In general the accuracy outcomes of MGT underline the optimality of $\hat{\mathcal{K}}_{SSS}$.

In addition, the runtime behavior of MGT in noiseless settings indicates a phase transition revealing the easy-hard-easy nature of group testing and its deep empirical study lead us to conjecture 5.3.1. An intuitive meaning of such behaviour has been identified with the derivation of $\hat{\mathcal{C}}_{\mathcal{A}}$, now we will look for a theoretical way to argue and formally prove 5.3.1 since the empirical results largely sustain our claim. We are opening a new problem so various future directions can be pursued. For a better understanding of phase

transition we suggest and aim to build a parallel with physics and to use model from such field to describe phase transition with good accuracy, following the approach in works such [KS99, MMZ01](see Section 2.5.1 for a discussion). Those directions are promising in order to unveil the nature of combinatorial optimization problem, decoding information problem and their relations.

Bibliography

- [AAES08] G. Atia, S. Aeron, E. Ermiş, and V. Saligrama. On throughput maximization and interference avoidance in cognitive radios. In *2008 5th IEEE Consumer Communications and Networking Conference*, pages 963–967, Jan 2008.
- [AAS16] Cem Aksoylar, George K Atia, and Venkatesh Saligrama. Sparse signal processing with linear and nonlinear observations: A unified shannon-theoretic approach. *IEEE Transactions on Information Theory*, 63(2):749–776, 2016.
- [ABJ14] Matthew Aldridge, Leonardo Baldassini, and Oliver Johnson. Group testing algorithms: Bounds and simulations. *IEEE Transactions on Information Theory*, 60(6):3671–3687, 2014.
- [ABL13] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196:77 – 105, 2013.

- [ABRW15] Andris Ambainis, Aleksandrs Belovs, Oded Regev, and Ronald de Wolf. Efficient quantum algorithms for (gapped) group testing and junta testing. *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, Dec 2015.
- [ACO08] Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, Oct 2008.
- [AJS19] Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett. Group testing: an information theory perspective. *arXiv preprint arXiv:1902.06002*, 2019.
- [AL97] Sanjeev Arora and Carsten Lund. Approximation algorithms for NP-hard problems. In Dorit S. Hochbaum, editor, *Hardness of Approximations*, pages 399–446. PWS Publishing Co., Boston, MA, USA, 1997.
- [Ald17] Matthew Aldridge. On the optimality of some group testing algorithms. *CoRR*, abs/1705.02708, 2017.
- [AMP03] Teresa Alsinet, Felip Manyà, and Jordi Planes. Improved branch and bound algorithms for MaxSAT. In *Sixth International Conference on Theory and Applications of Satisfiability Testing*, 2003.

- [AS07] Alp Atıcı and Rocco A. Servedio. Quantum algorithms for learning and testing juntas. *Quantum Information Processing*, 6(5):323–348, Sep 2007.
- [AS12] G. K. Atia and V. Saligrama. Boolean compressed sensing and noisy group testing. *IEEE Transactions on Information Theory*, 58(3):1880–1901, March 2012.
- [BBHH⁺17] Nader H. Bshouty, Vivian E. Bshouty-Hurani, George Haddad, Thomas Hashem, Fadi Khoury, and Omar Sharafy. Adaptive group testing algorithms to estimate the number of defectives, 2017.
- [BBTK96] D. J. Balding, W. J. Bruno, D.C Torney, and E. Knill. A comparative survey of non-adaptive pooling designs. In Terry Speed and Michael S. Waterman, editors, *Genetic Mapping and DNA Sequencing*, pages 133–154, New York, NY, 1996. Springer New York.
- [BC16] Nader H. Bshouty and Areej Costa. Exact learning of juntas from membership queries. *Algorithmic Learning Theory*, page 115–129, 2016.
- [BHJ15] Jeremias Berg, Antti Hyttinen, and Matti Järvisalo. Applications of MaxSAT in data analysis. *Pragmatics of SAT*, 2015.
- [BHKK92] A. Bar-Noy, F. K. Hwang, I. Kessler, and S. Kutten. A new competitive algorithm for group testing. In *Proc. IEEE INFOCOM '92: The Conference on*

- Computer Communications*, pages 786–793 vol.2, May 1992.
- [BJA13] Leonardo Baldassini, Oliver Johnson, and Matthew Aldridge. The capacity of adaptive group testing. In *2013 IEEE International Symposium on Information Theory*, pages 2676–2680. IEEE, 2013.
- [Bla10] Eric Blais. Testing juntas: A brief survey. pages 32–40, 01 2010.
- [BP98] Roberto Battiti and Marco Protasi. Approximate algorithms and heuristics for MaxSAT. In *Handbook of combinatorial optimization*, pages 77–148. Springer, 1998.
- [BT97] David J. Balding and David C. Torney. The design of pooling experiments for screening a clone map. *Fungal Genetics and Biology*, 21(3):302 – 307, 1997.
- [CCJS11] Chun Lam Chan, Pak Hou Che, Sidharth Jaggi, and Venkatesh Saligrama. Non-adaptive probabilistic group testing with noisy measurements: Near-optimal bounds with efficient algorithms. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1832–1839. IEEE, 2011.
- [CEPR10] Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. Pattern matching with don’t

- cares and few errors. *Journal of Computer and System Sciences*, 76(2):115 – 124, 2010.
- [CH08] Hong-Bin Chen and Frank K Hwang. A survey on nonadaptive group testing algorithms through the angle of decoding. *Journal of Combinatorial Optimization*, 15(1):49–59, 2008.
- [Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
- [CJSA14] Chun Lam Chan, Sidharth Jaggi, Venkatesh Saligrama, and Samar Agnihotri. Non-adaptive group testing: Explicit bounds and novel algorithms. *IEEE Transactions on Information Theory*, 60(5):3019–3035, 2014.
- [CM03] Graham Cormode and S. Muthukrishnan. What’s hot and what’s not: tracking most frequent items dynamically. In *PODS*, 2003.
- [COGHKL19] Amin Coja-Oghlan, Oliver Gebhard, Max Hahn-Klimroth, and Philipp Loick. Information-theoretic and algorithmic thresholds for group testing. *arXiv preprint arXiv:1902.02202*, 2019.
- [COP13] Amin Coja-Oghlan and Konstantinos Panagiotou. Going after the k-SAT threshold. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 705–714. ACM, 2013.

- [CR92] Vašek Chvátal and Bruce Reed. Mick gets some (the odds are on his side)(satisfiability). In *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627. IEEE, 1992.
- [CS90] Chao L. Chen and William H. Swallow. Using group testing to estimate a proportion, and to test the binomial model. *Biometrics*, 46(4):1035–1046, 1990.
- [DB11] Jessica Davies and Fahiem Bacchus. Solving MaxSAT by solving a sequence of simpler SAT instances. In *International conference on principles and practice of constraint programming*, pages 225–239. Springer, 2011.
- [DBV02] Annalisa De Bonis and Ugo Vaccaro. Efficient constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple access channels. In Rolf Möhring and Rajeev Raman, editors, *Algorithms — ESA 2002*, pages 335–347, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
- [DH06] Ding-Zhu Du and Frank K Hwang. *Pooling Designs and Nonadaptive Group Testing*. WORLD SCIENTIFIC, 2006.
- [DHH00] Dingzhu Du, Frank K Hwang, and Frank Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.

- [Dor43] Robert Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14:436–440, 1943.
- [DSS15] Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k . In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 59–68. ACM, 2015.
- [EGB⁺10] Yaniv Erlich, Assaf Gordon, Michael Brand, Gregory J. Hannon, and Partha P. Mitra. Compressed genotyping. *IEEE Transactions on Information Theory*, 56(2):706–723, Feb 2010.
- [EGN⁺15] Yaniv Erlich, Anna Gilbert, Hung Ngo, Atri Rudra, Nicolas Thierry-Mieg, Mary Wootters, Dina Zielinski, and Or Zuk. Biological screens from linear codes: theory and tools. *bioRxiv*, 2015.
- [EM14] A. Emad and O. Milenkovic. Poisson group testing: A probabilistic model for nonadaptive streaming boolean compressed sensing. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3335–3339, May 2014.
- [Erd] Paul Erdős. On two problems of information theory by.
- [EVM15] Amin Emad, Kush R. Varshney, and Dmitry M. Malioutov. A semiquantitative group testing ap-

- proach for learning interpretable clinical prediction rules. 2015.
- [Gal85] R. Gallager. A perspective on multiaccess channels. *IEEE Transactions on Information Theory*, 31(2):124–142, March 1985.
- [GAT05] Michael T. Goodrich, Mikhail J. Atallah, and Roberto Tamassia. Indexing information for data forensics. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 206–221, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [GGC91] Christoph Gille, Klaus Grade, and Charles Coutelle. A pooling strategy for heterozygote screening of the $\delta f508$ cystic fibrosis mutation. *Human Genetics*, 86(3):289–291, Jan 1991.
- [GH89] Joseph L. Gastwirth and Patricia A. Hammick. Estimation of the prevalence of a rare disease, preserving the anonymity of the subjects by group testing: application to estimating the prevalence of aids antibodies in blood donors. *Journal of Statistical Planning and Inference*, 22(1):15 – 27, 1989.
- [GH07] Michael T. Goodrich and Daniel S. Hirschberg. Improved adaptive group testing algorithms with applications to multiple access channels and dead sensor diagnosis. *Journal of Combinatorial Optimization*, 15(1):95–121, Aug 2007.

- [GIS08] A. C. Gilbert, M. A. Iwen, and M. J. Strauss. Group testing and sparse signal recovery. In *2008 42nd Asilomar Conference on Signals, Systems and Computers*, pages 1059–1063, Oct 2008.
- [GJS17] A. Ganesan, S. Jaggi, and V. Saligrama. Learning immune-defectives graph through group tests. *IEEE Transactions on Information Theory*, 63(5):3010–3028, May 2017.
- [GM19] Bishwamittra Ghosh and Kuldeep S. Meel. IMLI: An incremental framework for MaxSAT-based learning of interpretable classification rules. In *Proceedings of AAAI/ACM Conference on AI, Ethics, and Society(AIES)*, 1 2019.
- [Hay78] J. Hayes. An adaptive technique for local distribution. *IEEE Transactions on Communications*, 26(8):1178–1186, August 1978.
- [HD73] F. K. Hwang and D. N. Deutsch. A class of merging algorithms. *J. ACM*, 20(1):148–159, January 1973.
- [HHW81] M. Hu, F. Hwang, and J. Wang. A boundary problem for group testing. *SIAM Journal on Algebraic Discrete Methods*, 2(2):81–87, 1981.
- [HL02] E. S. Hong and R. E. Ladner. Group testing for image compression. *IEEE Transactions on Image Processing*, 11(8):901–911, Aug 2002.
- [Hwa72] F. K. Hwang. A method for detecting all defective members in a population by group testing.

Journal of the American Statistical Association, 67(339):605–608, 1972.

- [Ind97] P. Indyk. Deterministic superimposed coding with applications to pattern matching. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 127–136, Oct 1997.
- [JD08] Jun Luo and Dongning Guo. Neighbor discovery in wireless ad hoc networks based on group testing. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 791–797, Sep. 2008.
- [Joh17] Oliver Johnson. Strong converses for group testing from finite blocklength results. *IEEE Transactions on Information Theory*, 63(9):5923–5933, 2017.
- [Kar72] Richard M. Karp. Reducibility among combinatorial problems. 1972.
- [KG85] J. Komlos and A. Greenberg. An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Transactions on Information Theory*, 31(2):302–306, March 1985.
- [KR06] A. B. Kahng and S. Reda. New and improved bist diagnosis methods from combinatorial group testing theory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(3):533–543, March 2006.

- [KS64] W. Kautz and R. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4):363–377, October 1964.
- [KS99] Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random boolean formulae. 09 1999.
- [KW09] Raghunandan M. Kainkaryam and Peter J. Woolf. Pooling in high-throughput drug screening. *Current opinion in drug discovery development*, 12 3:339–50, 2009.
- [Li62] Chou Hsiung Li. A sequential method for screening experimental variables. *Journal of the American Statistical Association*, 57(298):455–477, 1962.
- [LK12] Pey-Chang Kent Lin and Sunil P Khatri. Application of MaxSAT-based ATPG to optimal cancer therapy design. *BMC genomics*, 13(6):S5, 2012.
- [LW66] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.
- [MA14] Yaakov Malinovsky and Paul S. Albert. A note on the minimax solution for the two-stage group testing problem. *The American statistician*, 69 1:45–52, 2014.
- [Mad89] T. Madej. An application of group testing to the file comparison problem. In *[1989] Proceedings*.

- The 9th International Conference on Distributed Computing Systems*, pages 237–243, June 1989.
- [Mal13a] Mikhail Malyutov. Search for sparse active inputs: A review. In *Information Theory, Combinatorics, and Search Theory*, pages 609–647. Springer, 2013.
- [Mal13b] Mikhail Malyutov. *Search for Sparse Active Inputs: A Review*, pages 609–647. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [MDM13] R. Mourad, Z. Dawy, and F. Morcos. Designing pooling systems for noisy high-throughput protein-protein interaction experiments using boolean compressed sensing. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(6):1478–1490, Nov 2013.
- [MM12] Dmitry Malioutov and Mikhail Malyutov. Boolean compressed sensing: Lp relaxation for group testing. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3305–3308. IEEE, 2012.
- [MM18] Dmitry Malioutov and Kuldeep S Meel. MLIC: A MaxSAT-based framework for learning interpretable classification rules. In *International Conference on Principles and Practice of Constraint Programming*, pages 312–327. Springer, 2018.
- [MMZ01] Olivier C. Martin, Rémi Monasson, and Riccardo Zecchina. Statistical mechanics methods and phase

- transitions in optimization problems. *Theoretical Computer Science*, 265(1-2):3–67, Aug 2001.
- [MOS04] Elchanan Mossel, Ryan O’Donnell, and Rocco A. Servedio. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421 – 434, 2004. Special Issue on STOC 2003.
- [MP04] Anthony J. Macula and Leonard J. Popyack. A group testing method for finding patterns in data. *Discrete Applied Mathematics*, 144(1):149 – 157, 2004. Discrete Mathematics and Data Mining.
- [MS09] MB Malyutov and H Sadaka. Capacity of screening under linear programming analysis. In *Proceedings of the 6th Simulation International Workshop on Simulation, St. Petersburg, Russia*, pages 1041–1045, 2009.
- [MSL92] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *AAAI*, volume 92, pages 459–465, 1992.
- [MSP07] Joao Marques-Silva and Jordi Planes. On using unsatisfiability for solving maximum satisfiability. *arXiv preprint arXiv:0712.1097*, 2007.
- [MV13] Dmitry Malioutov and Kush Varshney. Exact rule learning via boolean compressed sensing. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference*

- on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 765–773, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.
- [MVED17] Dmitry M. Malioutov, Kush R. Varshney, Amin Emad, and Sanjeeb Dash. *Learning Interpretable Classification Rules with Boolean Compressed Sensing*, pages 95–121. Springer International Publishing, Cham, 2017.
- [RGPS10] Nathan Robinson, Charles Gretton, Duc Nghia Pham, and Abdul Sattar. Partial weighted MaxSAT for optimal planning. In *Pacific rim international conference on artificial intelligence*, pages 231–243. Springer, 2010.
- [SAZ09] Noam Shental, Amnon Amir, and Or Zuk. Rare-allele detection using compressed sequencing, 2009.
- [SBC⁺02] Pak Sham, Joel Bader, Ian Craig, Michael O’Donovan, and Michael Owen. Dna pooling: A tool for large-scale association studies. *Nature reviews. Genetics*, 3:862–71, 12 2002.
- [SC16] Jonathan Scarlett and Volkan Cevher. Phase transitions in group testing. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 40–53. Society for Industrial and Applied Mathematics, 2016.

- [SFG⁺03] M. R. Stan, P. D. Franzon, S. C. Goldstein, J. C. Lach, and M. M. Ziegler. Molecular electronics: from devices and interconnect to circuits and architecture. *Proceedings of the IEEE*, 91(11):1940–1957, Nov 2003.
- [SFJ14] Miaoqing Shi, Teddy Furon, and Hervé Jégou. A group testing framework for similarity search in high-dimensional spaces. 11 2014.
- [SG59] M. Sobel and P. A. Groll. Group testing to eliminate efficiently all defectives in a binomial sample. *The Bell System Technical Journal*, 38(5):1179–1252, Sep. 1959.
- [Sha48] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 7 1948.
- [Swa85] William H. Swallow. Group testing for estimating infection rates and probabilities of disease transmission. 1985.
- [Tho62] Keith H. Thompson. Estimation of the proportion of vectors in a natural population of insects. *Biometrics*, 18(4):568–578, 1962.
- [Ula76] Stanislaw M. Ulam. *Adventures of a mathematician / S. M. Ulam*. Scribner New York, 1976.
- [Var95] M. K. Varanasi. Group detection for synchronous gaussian code-division multiple-access

- channels. *IEEE Transactions on Information Theory*, 41(4):1083–1096, July 1995.
- [Vaz01] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2001.
- [WHB80] Stephen D. Walter, Stephen W. Hildreth, and Barry J. Beaty. Estimation of infections rates in populations of organism using pool of variable size. *American Journal of Epidemiology*, 112(1):124–128, 07 1980.
- [WLY13] J. Wang, E. Lo, and M. L. Yiu. Identifying the most connected vertices in hidden bipartite graphs using group testing. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2245–2256, Oct 2013.
- [WZC18] C. Wang, Q. Zhao, and C. Chuah. Optimal nested test plan for combinatorial quantitative group testing. *IEEE Transactions on Signal Processing*, 66(4):992–1006, Feb 2018.
- [WZK13] Rouven Walter, Christoph Zengler, and Wolfgang Küchlin. Applications of MaxSAT in automotive configuration. In *Configuration Workshop*, volume 1, page 21. Citeseer, 2013.
- [XSTZ10] Y. Xuan, I. Shin, M. T. Thai, and T. Znati. Detecting application denial-of-service attacks: A group-testing-based approach. *IEEE Transactions*

- on Parallel and Distributed Systems*, 21(8):1203–1216, Aug 2010.
- [YS04] Yao-Win Hong and A. Scaglione. Group testing for sensor networks: the value of asking the right questions. In *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, volume 2, pages 1297–1301 Vol.2, Nov 2004.
- [ZB12] Lei Zhang and Fahiem Bacchus. MaxSAT heuristics for cost optimal planning. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
- [ZH17] W. Zhang and L. Huang. On or many-access channels. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2638–2642, June 2017.
- [ZKMZ13] Pan Zhang, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Non-adaptive pooling strategies for detection of rare faulty items. In *Proc. IEEE International Conference on Communications Workshops (ICC)*, pages 1409–1414. IEEE, 2013.

Appendix A

Proof and Derivations

In this appendix we report the proofs that were not crucial to the understanding of the work, but are important for the formalism.

A.1 NP-Hardness of $\widehat{\mathcal{K}}_{SSS}$

We reduced our problem, which was the decoding of results from test in a non-adaptive fashion of Group Testing, in particular compute the so called $\widehat{\mathcal{K}}_{SSS}$, to a Maximum-Satisfiability problem which is a known *NP-HARD* problem. To make stronger our argument is useful to prove that our problem is at least *HARD* as the one we are reducing it to. The following proves Theorem 3.2.3. In the way to prove that our problem is a *NP-HARD* problem I've identified a known *NP-HARD* problem that is suitable to be reduced to our problem in *polynomial time*. For this purpose we refer to the 1972 Karp article [Kar72] which show the *NP-Completeness* of some combinatorial decision problem. In our proof we will refer to the NP-Hardness of the **Minimum Vertex**

Cover.

A.1.1 What is Vertex Cover:

Definition A.1.1. A vertex cover \mathcal{V}' of an undirected graph $\mathcal{C} = (\mathcal{V}, \mathcal{E})$ is a subset of \mathcal{V} such that, being $\mathcal{E} = \{(u, v) : u, v \in \mathcal{V}\}$

$$\forall e \in \mathcal{E} \Rightarrow (u \in \mathcal{V}') \vee (v \in \mathcal{V}')$$

that is to say it is a set of vertices \mathcal{V}' where every edge has at least one endpoint in the vertex cover \mathcal{V}' . Such a set is said to cover the edges of \mathcal{C} . It is one of Karp's 21 NP-complete problems [Kar72]. The optimization problem (**Minimum Vertex Cover**) is known to be NP-Hard.

A.1.2 Proof

Reduction:

First of all we must reduce the optimization version of **Vertex Cover** into a decoding Non-Adaptive Group Testing instance with items vector \mathbf{x} , outcome vector \mathbf{y} and pooling matrix \mathbf{A} . The reduction is as follows:

- to each vertex $v \in \mathcal{V}$ is assigned a variable (an item) to be recovered x_j , so $|\mathbf{x}| = |\mathcal{V}| = n$, as above v_j is associated to x_j
- the matrix $m \times n$ of pooled measurements \mathbf{A} is defined as $\mathbf{A}_{ij} = 1 \iff e_i = (v_j, v) \vee e_i = (v, v_j) \in \mathcal{C}$ ¹

¹There's a 1:1 mapping between edges and tests, vertex and items; an item belongs to a test if and only if its associated vertex is part of the edges associated to the test

- to each edge $e \in \mathcal{E}$ is assigned a test $y_i = 1$, so $|\mathbf{y}| = |\mathcal{E}| = m$, to simplify we will enumerate the associated test and edges in the order in which they are associated such that y_i is associated to e_i

A decoding instance has been identified, now, once the solution $\widehat{\mathcal{K}}_{SSS}$ is computed we must specify which vertex cover is associated: an item (variable x_j) is recovered as faulty, so $x_j \in \widehat{\mathcal{K}}_{SSS}$, if and only if the vertex associated to x_j , $v_j \in \mathcal{V}'$.

So far we have defined \mathcal{V}' associated to an estimate $\widehat{\mathbf{x}}$. By the definition of a vertex cover \mathcal{V}' we claim that \mathcal{V}' is a vertex cover and in particular is the **Minimum Vertex Cover**.

Lemma A.1.1. *Let be $\widehat{\mathcal{K}}_{SSS}$ the estimate associated to (\mathbf{A}, \mathbf{y}) and let be \mathcal{V}' the set of vertex associated to $\widehat{\mathcal{K}}_{SSS}$, then \mathcal{V}' is the minimum vertex cover of \mathcal{C} .*

Proof. of A.1.1: Let's prove that \mathcal{V}' is a vertex cover.

Let's assume that \mathcal{V}' is not a vertex cover, that is there exist edge $e_i = (v_j, v_k) \in \mathcal{E}$ such that $v_j, v_k \notin \mathcal{V}'$.

By the definition of our reduction there must exist a test y_i , associated to e_i , such that $\mathbf{A}_{ij} = \mathbf{A}_{ik} = 1$. Hence a $\widehat{\mathcal{K}}_{SSS}$ must contain either x_k or x_j , as it is a Satisfying Set, that is a **contradiction**.

Let's now prove that \mathcal{V}' is a **Minimum Vertex Cover**.

Assume that there exist a \mathcal{V}'' such that $|\mathcal{V}''| < |\mathcal{V}'|$ and \mathcal{V}'' is a vertex cover of \mathcal{C} .

Then for each $e = (v_j, v_k) \in \mathcal{E}$ v_j or $v_k \in \mathcal{V}''$.

The tests are generated only by the edges, so the constraints on a satisfying set $\widehat{\mathcal{K}}$ to (\mathbf{A}, \mathbf{y}) are generated only by edges definition. By the definition of the reduction, each e generate a constraints

over a satisfying set $\widehat{\mathcal{K}}$ that is $(x_j \in \widehat{\mathcal{K}}) \vee (x_k \in \widehat{\mathcal{K}})$.

Hence \mathcal{V}'' is associated to a satisfying set $\widehat{\mathcal{K}}'$ to (\mathbf{A}, \mathbf{y}) which have $|\mathcal{V}''|$ faulty items. $|\widehat{\mathcal{K}}_{SSS}| > |\widehat{\mathcal{K}}''|$ is a **contradiction** as $\widehat{\mathcal{K}}_{SSS}$ is the smallest satisfying set to (\mathbf{A}, \mathbf{y}) by definition. QED

A.2 Concepts and Derivation of C-Estimate

In Section 5.3.2 we presented an estimate of the normalized time complexity of an accurate enough algorithm \mathcal{A} , $\widehat{C}_{\mathcal{A}}$. Such estimate comes from the empirical study of the observed easy-hard-easy behaviour. In Figure 5.3.3 we observed that such behaviour is shown by MGT, SCOMP and LP, while DD and COMP runtimes follow linearly number of tests m . In section 3.2.2 we presented the concepts behind COMP, DD and SCOMP. Recalling such concepts we try to interpret the behaviour focusing the attention on the differences between DD and SCOMP. The first gives a lower approximation of $\widehat{\mathcal{K}}_{SSS}$ (which is exactly computed by MGT) and consider an item to be defective when an easy and trivial proof of its defectiveness is available. In particular DD "marks" an item when it is defective and appears in at least one pool without any other defectives. We can express the same concept by defining the defectives in $\widehat{\mathcal{K}}_{DD}$ as the defectives that are not masked ones (see Masked Defective Definition 3.1.15). SCOMP computation take place over the $\widehat{\mathcal{K}}_{DD}$, after such set is obtained this algorithm consider all the tests \mathbf{y} and verify if there is any test which is not explained by the considered estimate. In particular the i -th test is unexplained by an estimate set $\widehat{\mathcal{K}}$ (hence $\widehat{\mathcal{K}}$ is not a satisfying set by Definition 3.1.12) when it has a positive outcome and there is no item of its pool \mathbf{A}_i which is contained by $\widehat{\mathcal{K}}$. SCOMP does not

terminate until there is no unexplained test remaining and marks items as defectives (i.e. add items to $\hat{\mathcal{K}}$) in a greedy manner², so the estimate $\hat{\mathcal{K}}_{\text{SCOMP}}$ refine the solution $\hat{\mathcal{K}}_{\text{DD}}$. Since MGT (by empirical evidence and theoretical optimality under Bernoulli design) is the algorithm that achieves the best rate, while LP and SCOMP compute solution which are an approximation of $\hat{\mathcal{K}}_{\text{SSS}}$ we can argue with such empirical results in support of our Conjecture 5.3.1. But why COMP and DD do not manifest the same complexity behaviour? Both algorithms chooses defectives with simple and approximate constraint over their defectiveness, the first compute an over estimate of \mathcal{K} making false positives errors, in particular it computes the largest satisfying set (see Lemma 3.2.1), and the latter computes an under estimate of \mathcal{K} making false negatives errors. In particular DD, as stated before, under approximate $\hat{\mathcal{K}}_{\text{SSS}}$ and is the starting point for the computation of SCOMP. We focus on the difference between those two algorithm (DD and SCOMP) to give a meaning of the easy-hard-easy phase transition.

A.2.1 Probability of Picking a Masked Defective

The presence of unexplained test under the hardness bell (see Figure 5.3.3) underline also the presence of the highest number of masked defectives under such bell. A masked defective is an item, which is a defective but never appears without any other defectives in a pool of a test. We can see the presence of this particular defectives in a designed set of pools as a measure of the difficulty of computation over the encoded information. In Figure

²At each step SCOMP chooses the items which is contained by the highest number of unexplained tests

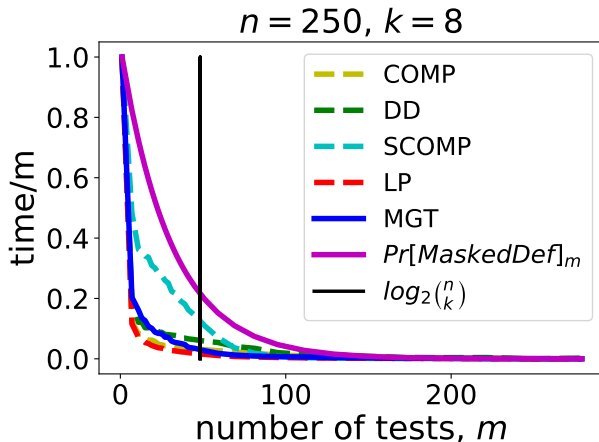


Figure A.2.1: Plot of the \mathbb{P}_{md} with the time per m ratio normalized to 1 for each algorithm.

A.2.1 we plot all the time complexity (over the range $m \in (1, n)$) divided by the number of tests m^3 in order to visualize "how much" an individual tests add difficulty of the computation. We also plot the derived probability of a Masked Defectives and we noticed that this probability is highly related to the other function plotted, it expresses qualitatively the "hardness per test" concept and bounds the empirical values.

Derivation of \mathbb{P}_{md}

We now recall Definition 5.3.1 of \mathbb{P}_{md} and we express how this probability has been derived.

We want to express the probability that picking up at random

³In order to compare the complexity all the runtimes are normalized to 1

an item from a group testing instance, such item is a defective and in particular is a masked one by Definition 3.1.15. Since the two process of generating a set with subset of defective \mathcal{K} and design of matrix \mathbf{A} are independent and being all the items sampled as i.i.d

$$\mathbb{P}_{md} = \mathbb{P}[\text{ItemDefectivity}]\mathbb{P}[\text{MaskedIfDefective}]^4$$

Recalling that the subset \mathcal{K} is generated by a bernoulli process with probability $p < 0.5$ then $\mathbb{P}[\text{Defectivity}] = p$. Consider each pool independetly we can decompose $\mathbb{P}[\text{MaskedIfDefective}]$ and obtain

$$\begin{aligned} \mathbb{P}[\text{MaskedIfDefective}] &= \\ \mathbb{P}[\text{BelongToPoolsWithNoDefectives}]^{\complement} &= \\ (\mathbb{P}[\text{NotBelongToPool}]\mathbb{P}[\text{BelongWithDefectives}])^m & \end{aligned}$$

So we are considering the joint probability of each independent pool for an item, in particular for each pool the item satisfy the condition if it does not belong to pool or, in case in it does belong, at least another defectives is in that pool.

Furthermore we can decompose $\mathbb{P}[\text{BelongWithDefectives}]$ as :

$$\begin{aligned} \mathbb{P}[\text{BelongWithDefectives}] &= \\ \mathbb{P}[\text{BelongToPool}]\mathbb{P}[\text{PoolWithNoDefectives}]^{\complement} &= \\ \mathbb{P}[\text{BelongToPool}](1 - (\mathbb{P}[\text{NoItemDefectivity}] + & \\ \mathbb{P}[\text{ItemDefectivity}]\mathbb{P}[\text{NotBelongToPool}])^{n-1}) & \end{aligned}$$

Explaining the above decomposition we are decomposing the "Belong with Defectives" to: "if an item belong to the pool" then

⁴This probability is the probability of appearing in a pool only with at least one defective

"there must be at least one defectives in such pool", the latter is the complementary of saying "there is no defective in such pool". Furthermore the probability inside the complementary can be expressed as: "if an item is not defective then can be either belong or not belongs to such pool" joint with "if an item is defectives then it must not belongs to such pool". With $1 - \mathbb{P}$ we obtain the complementary probability.

Since the pools \mathbf{A} are generated with Bernoulli process with probability q and being the other independent elements exactly $n - 1$ we obtain:

$$\mathbb{P}[\text{MaskedIfDefective}] = (1 - q + q(1 - (1 - p + p(1 - q))^{n-1}))^m$$

and putting together

$$\mathbb{P}_{md} = p(1 - q + q(1 - (1 - p + p(1 - q))^{n-1}))^m$$

With some with some trivial passage we finally obtain

$$\mathbb{P}_{md} = p(1 - q(1 - pq)^{n-1})^m$$

Note that $1 - q(1 - pq)^{n-1}$ can be directly derived as q is the probability of belonging to a pool and $(1 - pq)^{n-1}$ is the geometric series with argument $pq < 1$, so it is the summation of each independent probability of the other items to not belong to such pool if they are defectives.

A.2.2 From \mathbb{P}_{md} to \widehat{C}_A

\mathbb{P}_{md} describes, given the parameter of a group testing instances, the probability that picking up an item such item is a masked

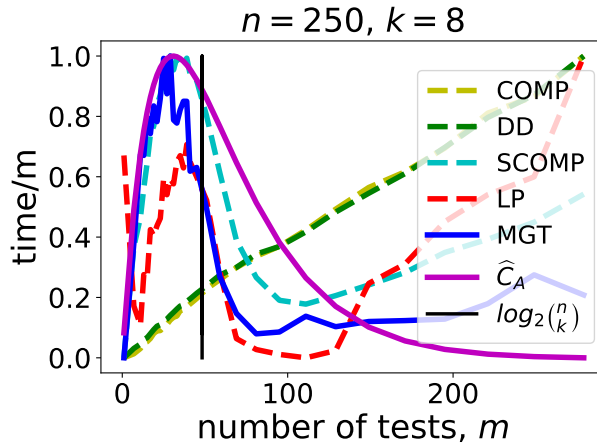


Figure A.2.2: Plot of the \hat{C}_A with the runtime normalized to 1 for each algorithm.

defective (Definition 3.1.15). We suggested that this probability can estimate the time complexity over number of tests ratio, so the "hardness per test" and verified it empirically(Figure A.2.1). We are seeking a meaning of the easy-hard-easy behaviour and proving empirically the relation between \mathbb{P}_{md} with the hardness of an instance can delineate the concept we are looking for. In particular being \mathbb{P}_{md} the estimate of the "hardness per test" we can simply multiply it per m and obtaining so an estimator of the "general hardness" of an instance. In figure A.2.2, where we plot all the runtimes normalized to 1, we show how qualitatively \hat{C}_A estimate the hardness of each instances and we can observed that the "trivial reasoner" does not follow in any way this trend, indeed the shape and the dependency from m is qualitatively the

same. Instead, the more accurate algorithms manifest a pretty similar, in some range identical, correlation to $\hat{C}_{\mathcal{A}}$ with respect to m and this correlation is higher considering the algorithm that achieves the highest rate, MGT.

Appendix B

Additional Plot

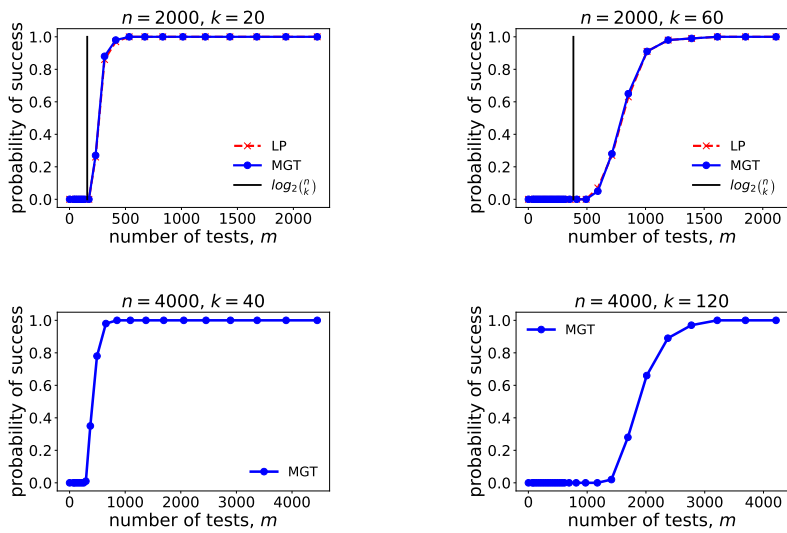


Figure B.0.1: Scalability analysis in noiseless settings.

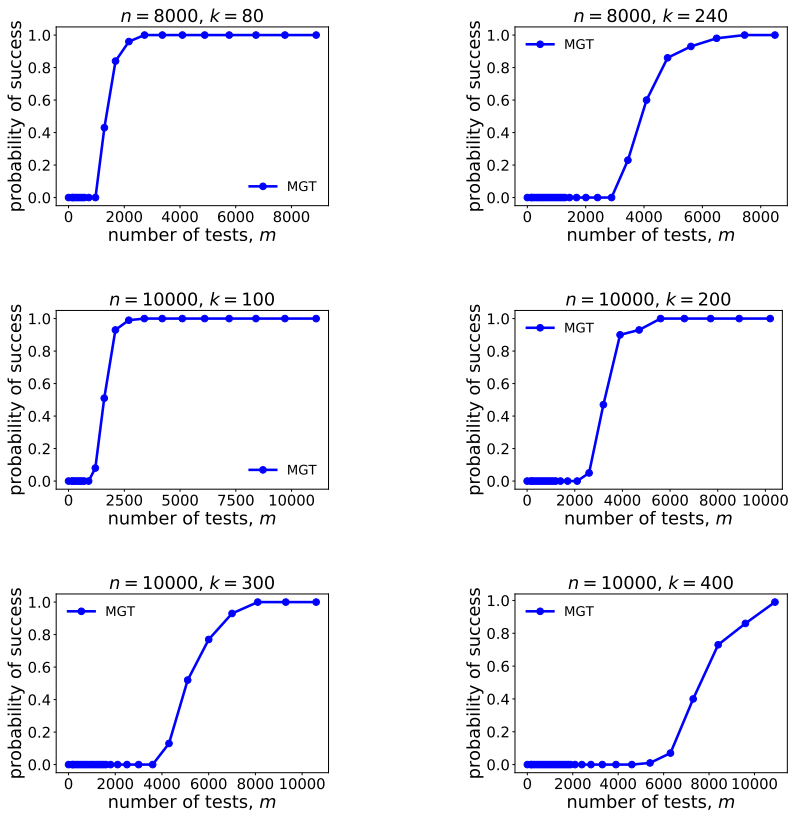


Figure B.0.2: Scalability analysis in noiseless settings.

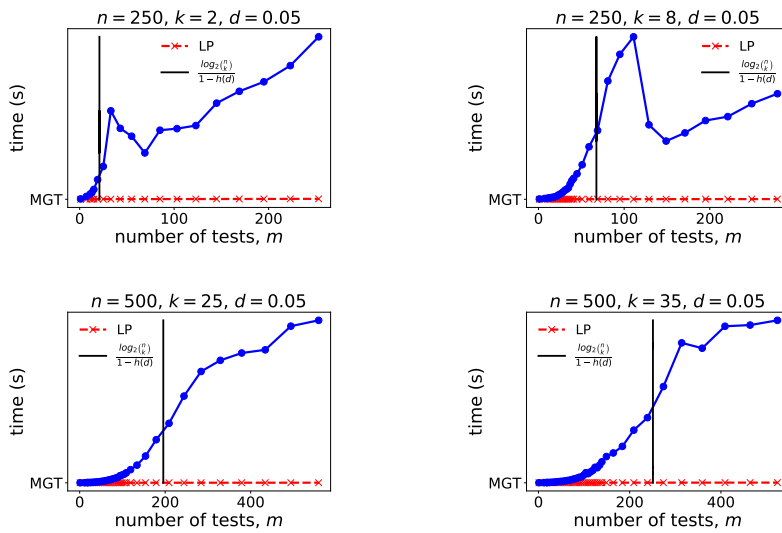


Figure B.0.3: Runtime Trend in Noisy Settings.

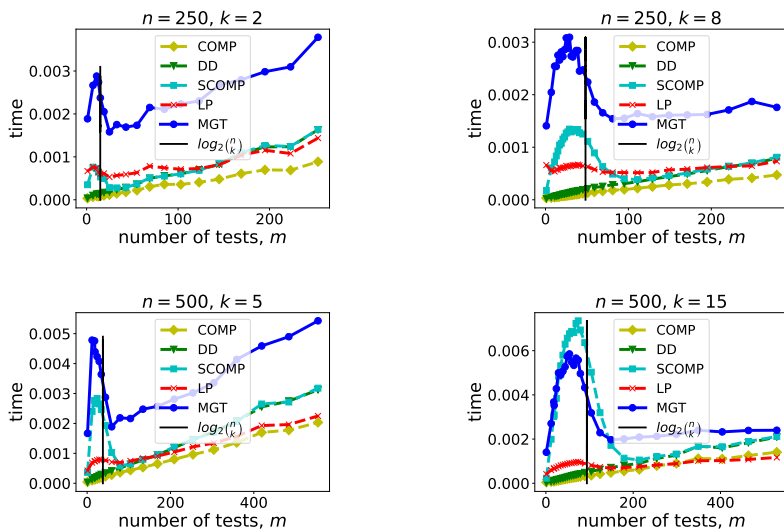


Figure B.0.4: Easy-Hard-Easy Phase Transition in Noiseless Settings for MGT, LP, COMP, DD, SCOMP

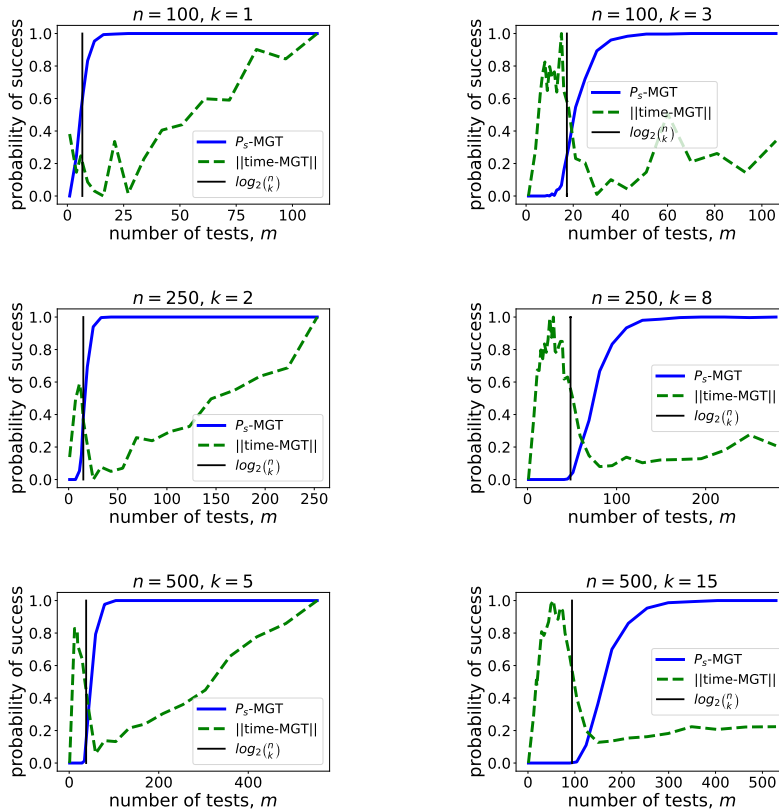


Figure B.0.5: Easy-Hard-Easy Phase Transition in Noiseless Settings for MGT and LP

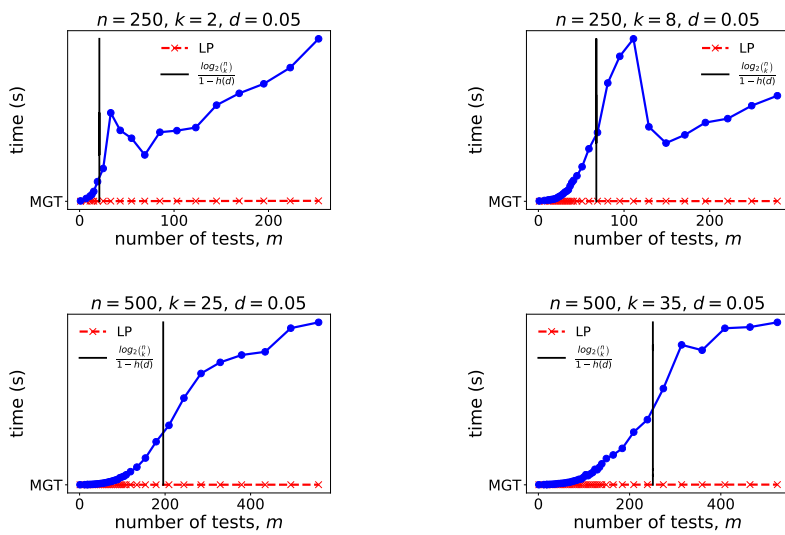


Figure B.0.6: Runtime Trend in Noisy Settings.