# Methods for Solving Extremal Problems in Practice

Michael Frank

**Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel**
frankm@cs.bgu.ac.il

## Abstract

During the 20th century there has been an incredible progress in solving theoretically hard problems in practice. One of the most prominent examples is the DPLL algorithm and its derivatives to solve the Boolean satisfiability problem, which can handle instances with millions of variables and clauses in reasonable time, notwithstanding the theoretical difficulty of solving the problem.

Despite this progress, there are classes of problems that contain especially hard instances, which have remained open for decades despite their relative small size. One such class is the class of extremal problems, which typically involve finding a combinatorial object under some constraints (e.g, the search for Ramsey numbers). In recent years, a number of specialized methods have emerged to tackle extremal problems. Most of these methods are applied to a specific problem, despite the fact there is a great deal in common between different problems.

Following a meticulous examination of these methods, we would like to extend them to handle general extremal problems. Further more, we would like to offer ways to exploit the general structure of extremal problems in order to develop constraints and symmetry breaking techniques which will, hopefully, improve existing tools. The latter point is of immense importance in the context of extremal problems, which often hamper existing tools when there is a great deal of symmetry in the search space, or when not enough is known of the problem structure. For example, if a graph is a solution to a problem instance, in many cases any isomorphic graph will also be a solution. In such cases, existing methods can usually be applied only if the model excludes symmetries.

**1998 ACM Subject Classification** D.3.2 Language Classifications

**Keywords and phrases** Extremal Problems, Constraints, SAT Solving, Logic Programming, Parallelism

**Digital Object Identifier** 10.4230/OASIcs.ICLP.2016.21

## 1 Introduction

A Fundamental research topic in Computer Science is that of combinatorics. Specifically that of finite combinatorial objects, such as finite graphs, finite groups, and circuits. Many of the problem instances which arise in these fields tend to be theoretically intractable, though they are often solvable in practice.

However, among such instances, there are many small, yet notoriously difficult structures to “crack”: objects the understanding of which still eludes us in both theory and practice. Several prominent examples include: characterizing and finding Ramsey numbers [21], finding optimal size/depth sorting networks [16, 8], determining the complexity of XOR-AND circuits [1, 2, 23, 4], graph enumeration under constraints [17, 18] (e.g, limited girth, cuts, coloring, etc.) and forbidden-graph characteristics.

Broadly speaking: extremal combinatorics and extremal graph theory are the fields of research which examine finite combinatorial and graph problems the solutions of which usually have to satisfy some restrictions (such as the problems presented above). The problems associated with these fields are collectively referred to as extremal problems.

Historically, there are many techniques, which facilitate solving intractable (usually NP-Hard problem instances), in reasonable time. In recent years, two such techniques came to focus: constraint based techniques and iterative techniques. These techniques led to a plethora of constraints solvers [19, 14], SAT solvers [9, 22], graph iterators [17] and additional applications that can solve an abundance of theoretically hard problem instances in practical scenarios.

While these techniques can be extremely powerful, many times they are not enough to solve extremal problem instances on their own, evident by the lackluster progress with extremal problems. A prominent example comes from the search for Ramsey numbers – where only a handful of exact values are known for small instances [21]. Mathematician Paul Erdős was quoted as saying about the calculation of the Ramsey number $R(5,5)$:

“Suppose aliens invade the earth and threaten to obliterate it in a year’s time unless human beings can find the Ramsey number for red five and blue five. We could marshal the world’s best minds and fastest computers, and within a year we could probably calculate the value. If the aliens demanded the Ramsey number for red six and blue six, however, we would have no choice but to launch a preemptive attack.”

*“Ramsey Theory” by Ronald L. Graham and Joel H. Spencer, Scientific American (July 1990), pp. 112–117*

Since the introduction of Ramsey numbers in 1930, and twenty five years after the quotation above, the precise value of $R(5,5)$ remains an open problem.

During the past few years, however, we are witnessing the rise of new methodologies, which enable us to better understand and solve extremal problems. Indeed, in the last two decades several prominent extremal problems, many of which have been open for over 50 years – were closed. Such problems include e.g, the computation of Ramsey numbers $R(4,3,3)$ [6], $R(4,5)$ [18], an improved lower bound for $R(4,8)$ [13], size optimal sorting networks for 9 and 10 inputs [5], depth optimal sorting networks for 17 inputs [3], improved lower bounds for circuit complexity of XOR-AND circuits for 5 and 7 inputs [4].

These new methodologies include on the one hand – improvements to existing techniques and theory of extremal problems, and on the other hand – the development of new, more sophisticated, albeit specific techniques aimed towards particular extremal problems. These techniques include e.g, SAT solving [9, 22], symmetry breaking [7, 15], abstraction [6], and parallelism [5]. Note that in many cases, extremal problems are NP-Hard, or $\Sigma_2^P$-Hard, which in part explains their difficulty.

During the past two years we have made contributions in the area of extremal combinatorial and graph problems. In particular in exploring extremal circuit problems (e.g, sorting networks and AND-XOR circuit complexity), and in the computation of multi-color Ramsey numbers. We propose to expand and generalize domain specific methods in order to solve general problems in the fields of extremal combinatorics and graph theory. Our initial goal is to expand on the techniques discussed in e.g [4, 15, 6, 5, 7, 20, 11, 10] in order to solve more difficult instances, and eventually develop generalized techniques which can be applied across extremal problems. Further more, we propose to exploit problem structure in order to employ optimized solving algorithms such as those discussed in [20].

## 2 Research Progress

Seminal to our work is the integration of two methods: (1) The *Generate & Test* method and (2) The *Constrain & Generate* method. With the *generate and test* method, one explicitly enumerates over all solutions, pruning undesired results, and checking each for a given property. Whereas with the *constrain and generate* approach, one typically encodes the problem for some general-purpose discrete satisfiability solver (i.e. SAT, integer programming, constraint programming), which does the enumeration implicitly, and outputs a result. One of the keys to our approach, is to combine these two methods. Using a generate and test algorithm to produce partial solutions, which are then encoded individually, and solved independently (and in parallel).

Moreover, in both the generate and test, and constrain and generate methods, structural knowledge of the problem as well as symmetry breaking techniques have been employed (e.g, [7, 15, 6]) to facilitate the search and limit the search space.

The following subsections present a short summary of work based on these methods. The first three present previously unknown results in the field of extremal problems, which rely on these methods, and the fourth subsection present a tool implemented to aid in the use of these methods.

### 2.1 Problem 1: Optimal Size Sorting Networks

In [5], we present a computer-assisted non-existence proof of 9-input sorting networks consisting of 24 comparators, thus showing that the 25-comparator sorting network found by Floyd in 1964 is optimal. As a corollary, the 29-comparator network found by Waksman in 1969 is also shown to be optimal for sorting 10 inputs. This proof employs three primary techniques that, although specific to sorting networks, also appear in some form in the problems discussed further in Sections 2.2 and 2.3.

### 2.2 Problem 2: Multi-Color Ramsey Number $R(4, 3, 3) = 30$

In [6], we present a computer-assisted non-existence proof of the multi-color Ramsey graph $(4, 3, 3)$ with 30 vertices, thus establishing that $R(4, 3, 3) = 30$. The problem of finding $R(4, 3, 3)$ has been characterized as the one with the best chances of being found "soon". Nevertheless, the precise value of $R(4, 3, 3)$ has remained unknown for nearly 50 years. The proof employs two main techniques: *abstraction* and *symmetry breaking*, in order to first prune the search space and then split it into manageable pieces. Both techniques have a great deal in common with techniques explained in Section 2.1 and the ones discussed in Section 2.3. We believe that these techniques can either be generalized or integrated, as discussed in Section 3.

### 2.3 Problem 3: AND-XOR Circuit Complexity

In [4] we present a computer-assisted proof that a Boolean function on 7 inputs with multiplicative complexity of at least 7 exists. The multiplicative complexity of a function is a measure of its non-linearity, and is of particular interest in the fields of cryptographic cipher analysis, the study of hash functions, and the study of communication complexity of multiparty computations. The results presented in this chapter rely on examining the topologies of XOR-AND circuits, which are equivalent to Boolean functions, and eventually applying the pigeonhole principle to show that there must exist a function with multiplicative complexity of 7. Three primary techniques are described, that we believe may be explored further as discussed in Chapter 3.

### 2.4 `pl-nauty` & `pl-gtools`

In [12] we introduce the `pl-nauty` & the `pl-gtools` libraries, which integrate the `nauty` graph automorphism tool with Prolog, thereby allowing Logic Programming to interface with `nauty`. Adding the capabilities of `nauty` to Prolog combines the strength of the "generate and prune" approach that is commonly used in logic programming and constraint solving, with the ability to reduce symmetries while reasoning over graph objects. Moreover, it enables the integration of `nauty` in existing tool-chains, such as SAT-solvers or finite domain constraints compilers which exist for Prolog.

## 3 Future Work

We are currently looking into two generalizations of the problems presented in sections 2.1, 2.2, and 2.3.

### 3.1 The Subsumption Problem

The subsumption problem is to determine whether given two sets $A, B \subseteq \{0,1\}^n$, there exists a permutation $\pi\colon [n] \to [n]$ such that $\pi(A) \subseteq B$, where $\pi(A) = \{\pi(x) : x \in A\}$.

The subsumption problem arises when solving the sorting network problem mentioned in section 2.1, and it has close ties to the sub-graph isomorphism problem. A better understanding of this problem will hopefully lead to a better algorithm for solving it. A generalized algorithm for subsumption may be used to generate arbitrary monotone Boolean functions, as well as allow the methods in [5] to be generalized for larger sizes of sorting networks.

We are currently exploring the structural information that can be obtained from $A$ and $B$ in order to perform a quicker subsumption test, much in the vein of `nauty`.

### 3.2 Abstraction & Concretization for Coloring Problems

Many graph coloring problems are often given as a predicate $\varphi$ such that the free variables of $\varphi$ correspond to an adjacency matrix $A$ with domain $0 \cup [k]$, and a satisfying assignment to $\varphi(A)$ implies the sought after coloring. Such problems are often notoriously difficult to solve, such as the case with e.g, the Ramsey coloring problem, variations of the Latin square and magic square problems, multi-color n-queens and more.

▶ **Definition 1** ((weak) isomorphism of graph colorings)**.** Let $(G, \kappa_1)$ and $(H, \kappa_2)$ be $k$-color graph colorings with $G = ([n], E_1)$ and $H = ([n], E_2)$. We say that $(G, \kappa_1)$ and $(H, \kappa_2)$ are weakly isomorphic, denoted $(G, \kappa_1) \approx (H, \kappa_2)$ if there exist permutations $\pi\colon [n] \to [n]$ and $\sigma\colon [k] \to [k]$ such that $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$ and $\kappa_1((u, v)) = \sigma(\kappa_2((\pi(u), \pi(v))))$. We extend the notation for the adjacency matrices of colorings and denote $A \approx B$ for the adjacency matrices $A$, $B$ of $(G, \kappa_1) \approx (H, \kappa_2)$.

A graph coloring problem is said to be $\approx$-closed (i.e, closed under $\approx$ relation) if the following definition hold:

▶ **Definition 2** (≈-closed graph coloring problem)**.** Let $\varphi$ a graph coloring problem. $\varphi$ is said to be ≈-closed if for all $(G_1, \kappa_1) \approx (G_2, \kappa_2)$ with adjacency matrices $A_1$ and $A_2$ respectively it holds that $\varphi(A_1) \iff \varphi(A_2)$. Alternatively:

$$(G_1, \kappa_1) \approx (G_2, \kappa_2) \iff (\varphi(A_1) \iff \varphi(A_2)) .$$

In [6] we present the method of *abstraction* and *concretization* for graph coloring problems. Although this method was developed specifically to solve Ramsey instances, it may be applied to any graph coloring problem closed under the weak isomorphism property. The degree matrix of coloring $A$ is a matrix $M$ such that $M_{i,j}$ is the number of $j$ colored neighbour of node $i$. The *abstraction* of an adjacency matrix $A$ is the lexicographically sorted degree matrix $M$ of $A$, and denoted $\alpha(A)$, and that the *concretization* of $M$ are all the adjacency matrices whose abstraction is $M$, denoted $\gamma(M)$. Notice also that:

▶ **Lemma 3.** *$A \approx A'$ if and only if $\alpha(A) = \alpha(A')$.*

Now, using observation 3, the search space of any graph coloring problem may be described in terms of the abstraction of degree matrices. The method then computes an over-approximation of possible solutions and uses that to guide the search for an actual solution (should one exists).

Notice that many classic coloring problems are closed under this relation e.g, Latin squares, Ramsey colorings, multi-colored n-queens. Therefore, it may be conceivable that the abstraction and concretization of graphs presented in [6], may be generalized for coloring problems which are ≈-closed.

## References

1 Joan Boyar and René Peralta. Tight bounds for the multiplicative complexity of symmetric functions. *TCS*, 396(1–3):223–246, 2008.

2 Joan Boyar, René Peralta, and Denis Pochuev. On the multiplicative complexity of Boolean functions over the basis (∧, +, 1). *TCS*, 235(1):43–57, 2000.

3 Daniel Bundala and Jakub Zavodny. Optimal sorting networks. In Adrian Horia Dediu, Carlos Martín-Vide, José Luis Sierra-Rodríguez, and Bianca Truthe, editors, *Language and Automata Theory and Applications – 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, volume 8370 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 2014. doi:10.1007/978-3-319-04921-2_19.

4 Michael Codish, Luís Cruz-Filipe, Michael Frank, and Peter Schneider-Kamp. When six gates are not enough. *CoRR*, abs/1508.05737, 2015. URL: http://arxiv.org/abs/1508.05737.

5 Michael Codish, Luís Cruz-Filipe, Michael Frank, and Peter Schneider-Kamp. Sorting nine inputs requires twenty-five comparisons. *Journal of Computer and System Sciences*, 82(3):551–563, 2016. doi:10.1016/j.jcss.2005.06.002.

6 Michael Codish, Michael Frank, Avraham Itzhakov, and Alice Miller. Computing the ramsey number r(4, 3, 3) using abstraction and symmetry breaking. *CoRR*, abs/1510.08266, 2015. URL: http://arxiv.org/abs/1510.08266.

7 Michael Codish, Alice Miller, Patrick Prosser, and Peter James Stuckey. Breaking symmetries in graph representation. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China*. IJCAI/AAAI, 2013. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6480.

8 Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

9 Niklas Eén and Niklas Sörensson. Minisat sat solver. URL: `http://minisat.se/Main.html`.

10 Thorsten Ehlers and Mike Müller. Faster sorting networks for 17, 19 and 20 inputs. *CoRR*, abs/1410.2736, 2014. URL: `http://arxiv.org/abs/1410.2736`.

11 Thorsten Ehlers and Mike Müller. New bounds on optimal sorting networks. *CoRR*, abs/1501.06946, 2015. URL: `http://arxiv.org/abs/1501.06946`.

12 M. Frank and M. Codish. Logic programming with graph automorphism: Integrating nauty with prolog (a tool paper). Technical report, Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel, 2016. URL: `https://www.cs.bgu.ac.il/~frankm/plnauty/`.

13 Hiroshi Fujita. A new lower bound for the ramsey number r(4, 8). *CoRR*, abs/1212.1328, 2012. URL: `http://arxiv.org/abs/1212.1328`.

14 M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam answer set solving collection. *AI Communications*, 24(2):107–124, 2011.

15 Avraham Itzhakov and Michael Codish. Breaking symmetries in graph search with canonizing sets. *CoRR*, abs/1511.08205, 2015. URL: `http://arxiv.org/abs/1511.08205`.

16 Donald E. Knuth. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

17 B. McKay. *nauty* user's guide (version 1.5). Technical Report TR-CS-90-02, Australian National University, Computer Science Department, 1990.

18 Brendan D. McKay and Stanislaw P. Radziszowski. $R(4, 5) = 25$. *Journal of Graph Theory*, 19(3):309–322, 1995. `doi:10.1002/jgt.3190190304`.

19 Amit Metodi and Michael Codish. Compiling finite domain constraints to sat with bee. *Theory and Practice of Logic Programming*, 12(4-5):465–483, 2012.

20 Amit Metodi, Michael Codish, and Peter J. Stuckey. Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. *J. Artif. Intell. Res. (JAIR)*, 46:303–341, 2013. `doi:10.1613/jair.3809`.

21 Stanislaw P. Radziszowski. Small Ramsey numbers. *Electronic Journal of Combinatorics*, 1994. Revision #14: January, 2014. URL: `http://www.combinatorics.org/`.

22 Mate Soos. CryptoMiniSAT, v2.5.1, 2010. URL: `http://www.msoos.org/cryptominisat2`.

23 Meltem Sönmez Turan and René Peralta. The multiplicative complexity of Boolean functions on four and five variables. In Thomas Eisenbarth and Erdinç Öztürk, editors, *LightSec 2014*, volume 8898 of *LNCS*, pages 21–33. Springer, 2015.