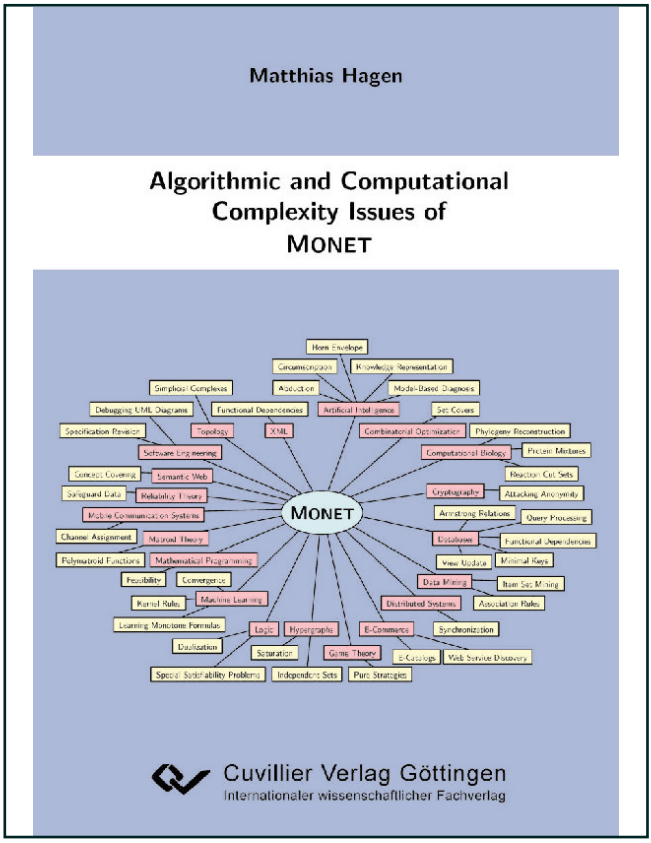
Matthias Hagen (Autor)

"Algorithmic and Computational Complexity Issues of MONET



https://cuvillier.de/de/shop/publications/1237

Chapter 1

Introduction

1.1 What means and to what end do we study MONET?[1]

Warning: What we do not(!) do. The first association when reading MONET probably is the famous French painter Claude Monet (1840–1926). His painting "Impression soleil levant" from 1872—a morning scene from Le Havre harbor—coined the term *Impressionism* that later on was used to describe a whole field of art. But as this thesis is on Theoretical Computer Science (TCS), this first impression on MONET is very misleading. We neither reveal any ancient TCS results due to Claude Monet (as we do not even know whether he was interested in efficient computation at all), nor do we study algorithmics and computational complexity with an "impressionistic" view. Hence, a big apology to all who started reading with the intention of discovering a not yet documented relation between the fine arts and TCS: Claude Monet is not(!) the topic of this work.

So, what is MONET? In this thesis we consider the problem MONET—an acronym for MO(notone) N(ormal form) E(quivalence) T(est)—that asks to decide equivalence of a monotone DNF (disjunctive normal form) and a monotone CNF (conjunctive normal form) (for formal definitions of any term we refer to Chapter 2). MONET is equivalent to TRANSHYP—given two hypergraphs, decide whether one is the transversal hypergraph of the other. TRANSHYP is a variant of the well-known problem VERTEXCOVER, in its hypergraph version also known as HITTINGSET. But instead of just asking for one vertex set of minimum cardinality that has a non-empty intersection with all edges, we are interested in the set of *all* minimal sets that "cover" resp. "hit" all edges.

Practical relevance. Consequently, MONET is a covering problem and can be interpreted as the enumeration of all (in some sense) minimal solutions of some system. Similar questions for enumeration of minimal solutions are ubiquitous in many problems from diverse applications. In fact, covering and enumeration problems that are MONET-equivalent or strongly related can be found in fields such as

[1] Title inspired by Friedrich Schiller's inaugural lecture (1789) at this thesis' author's university.

artificial intelligence, computational biology, databases, data mining, distributed systems, graph theory, logic, machine learning, mathematical programming, matroid theory, and mobile communication systems (cf. Chapter 3 for more details). The consequence is that any approach solving MONET can be easily transformed to solve a wide range of problems in very different fields. Thus, on the one hand, research on algorithms solving MONET and even any technique improving known procedures is very important from the point of view of practical applications.

And in theory: Unsettled complexity. On the other hand, MONET research is faced with some very interesting theoretical issues that we will describe in the following. The equivalence test for arbitrary monotone formulas (not necessarily in normal form) is coNP-complete [Rei03]. The same bound can be easily proven for arbitrary Boolean formulas in normal form. The coNP-completeness in this context means that these problems are "hard" since it is very unlikely that they are solvable by "fast" algorithms in the sense of deterministic polynomial running time (running time is usually measured with respect to the input size and polynomial running time is the usual notion of efficient solutions in complexity theory).

But note that in the MONET setting we require the input formulas to be monotone *and* in normal form, which together represent stronger "structural" restrictions and thus may ease the solution process. Whether these restrictions really can be exploited to develop polynomial MONET algorithms is an exciting open question for more than 25 years now [DT87, EG95, Joh91, LLK80, Lov92, Man02, Pap97]. The best currently known MONET algorithm has quasi-polynomial running time $n^{o(\log n)}$ [FK96], which still, after all, can be seen as an indication that MONET is probably not coNP-complete. Otherwise, all coNP-complete problems would be solvable in quasi-polynomial time—a result that hardly any expert expects. But as it is *quasi*-polynomial this algorithm is not "fast" as well.

Another indication that MONET probably is not coNP-complete is a recent result that shows MONET to be solvable using only $O(\log^2 n)$ guessed bits [EGM03, KS03a, KS03b]. Again, no expert expects any coNP-complete problem to be solvable with $O(\log^2 n)$ guessed bits as usually a polynomial number is assumed to be required.

MONET and the P-vs.-NP question. The currently unsolved complexity of MONET places it in the group of a handful—and thus really rare—problems that yet cannot be classified as "easy" (polynomial time solvable) or "hard" (NP- or coNP-complete). Hence, MONET, along with the prominent GRAPH ISOMORPHISM problem—given two graphs, decide if they are isomorphic—, may play some role in research on the popular P-vs.-NP question (one of the seven major problems in mathematics whose solution is worth a reward of $1,000,000\,\$$ denoted by the Clay Mathematics Institute). The question is whether the class of deterministic polynomial time solvable problems (the class P) and the class of nondeterministic polynomial time solvable problems (the classes NP, resp. coNP) coincide. If

MONET is not solvable in polynomial time (formally written as MONET $\notin$ P), then this immediately implies a separation of P and NP. But as the P-vs.-NP question is open since the field of complexity theory emerged, we expect showing MONET $\notin$ P a really tough problem. On the other hand, there are no known complexity theoretic consequences that would follow from MONET $\in$ P, although this question is also open for many years now and, hence, seems to be tough as well. But note that also PRIMES—given an integer, decide whether it is prime—had an analogue unsettled complexity status for a long time until it was actually shown to be solvable in polynomial time only a few years ago [AKS04]. Thus, there is still a possibility of a breakthrough showing MONET $\in$ P. (Unfortunately, this breaktrough is not contained in this thesis.) Nevertheless, we expect new techniques to be necessary as we conjecture that none of the known MONET algorithms (for which not always upper and lower bounds on the running time are known) is polynomial.

Hence, the problems between "easy" or "hard" (resp., "fast" / no "fast" solution) constitute two very exciting challenges to algorithmicists and complexity theorists as well. One is to actually find a fast algorithm and the other is to find some kind of complexity theoretic arguments for or against a fast solution. As for PRIMES these challenges finally led to a fast solution [AKS04]. In the case of GRAPH ISOMORPHISM we have a complexity theoretic classification that would yield a solution to an unsolved question very strongly related to the P-vs.-NP question [KST93] in case that GRAPH ISOMORPHISM is not(!) polynomial. More precisely, GRAPH ISOMORPHISM cannot be NP-hard unless the so-called polynomial hierarchy collapses to its second level—an event that is supposed to be rather unlikely. As for MONET the situation is comparable to that of GRAPH ISOMORPHISM. There are several indications that MONET is more likely to be polynomial than to be hard—like the quasi-polynomial algorithms by Fredman and Khachiyan [FK96] and the solvability with bounded nondeterminism [EGM03, KS03a, KS03b]. Furthermore, the consequences of being not polynomial even seem to be a little stronger for MONET than for GRAPH ISOMORPHISM.

However, all that is known for the fast solvability yet is that MONET is polynomial if and only if its computational variant MONET′—given an irredundant, monotone DNF, compute the equivalent irredundant, monotone CNF—is output-polynomial [BI95a]. Here, output-polynomiality is an appropriate notion of fast solvability for computation problems [JPY88]. Note in this context, that a slight generalization of MONET′ is very unlikely to have an output-polynomial solution. Namely, finding an algorithm that, given a monotone formula (not necessarily in DNF), computes the irredundant, monotone CNF in output-polynomial time is the same as showing P = NP and thus as hard as solving the P-vs.-NP question [GHM05].

Easy classes as a way out?! As there is no known fast algorithm for MONET yet, one branch of research focuses on identifying restrictions of the inputs that sufficiently simplify the problem to allow for polynomial time solutions. Such restrictions then yield "easy" classes of the problem. One example might be to examine instances where the DNF only contains monomials of constantly bounded size. And in fact, MONET with constantly size bounded monomials has polynomial time algorithms [BEGK00, EG95]. Many other easy classes are known (cf. Chapter 4 for more details). Due to the successes in searching for more and more easy classes, many practically interesting input instances can be solved in polynomial time although there is no known polynomial algorithm for MONET itself. This is maybe the major reason that makes worthwhile every effort invested in research on easy classes of MONET. But there are two other, possibly not less important reasons. One is that the knowledge of easy classes reveals new information about the really hard problems apparent when trying to attack polynomial solvability of the general problem MONET itself. And the other is that the easy classes might prove useful when looking for lower bound or hardness results for MONET.

Known MONET algorithms. There are many known algorithms solving MONET or MONET′ (cf. Chapter 5 for more details). They are inspired by very diverse techniques from different fields—which is not that surprising having in mind the broad range of equivalent problems. We do not really have to distinguish between algorithms for MONET or MONET′ as they can be easily transformed to solve the respective other problem version.

One of the earliest approaches was the Berge-multiplication algorithm (cf., e. g., [Ber89]). So far, it is the only algorithm having a known lower bound (that states what resources (e. g., running time) are minimally needed using the algorithm to solve arbitrary MONET instances). Takata's lower bound shows that Berge-multiplication is not fast [Tak07]. But as Berge-multiplication can be easily implemented, there have been several improvements of it [BMR03, DL05, KS05, Uno02, US03]. Analyses of the running times of the improved versions have been pending.

Other algorithms do not follow a term based approach as Berge-multiplication does but use a variable based decomposition technique [BMR03, EGM03, Eit91, MR94, Rym92, Rym94a]. Maybe the most famous variable based algorithms are the FK-algorithms A and B by Fredman and Khachiyan [FK96]. Algorithm B is the MONET algorithm with the currently best upper bound on the running time of $n^{o(\log n)}$. This upper bound is a guarantee on the amount of running time that will never be exceeded on any instance by FK-algorithm B.

Inspired by model-based diagnosis techniques are Reiter's algorithm [Rei87] (and its improvements [GSW89, Wot01]) and several other recently published algorithms, e. g., [LJ03, TT02]. There are MONET algorithms based on techniques that were useful in data mining settings [GKM$^+$03, HBC07], as well as genetic algorithms [LJ02, Vin99a, VØ00b] or parallel approaches [Elb08, KBEG07a].

Although there are many very diverse algorithms, all approaches have in common that there are only very few proven non-trivial upper or lower bounds on their running times. Hence, theoretical knowledge of the algorithms' behavior is not really well developed. Having in mind the wide applicability of MONET results, this situation is not satisfying at all.

Algorithm Engineering. One possibility—besides thorough theoretical analyses—to get some impression of (practical) performance of an algorithm is to implement it and run it on a lot of well-chosen inputs. This is one branch of the emerging field of Algorithm Engineering (cf. Chapter 6 for more details). Though, not that many experimental studies on MONET algorithms have been published and all have some lack of coverage [BMR03, DL05, KBEG06, KS05, LJ03, TT02, US03]. Some of the studies only show the potential of a single approach and do not really compare it to others. In addition, none of them includes the theoretically best algorithm, FK-algorithm B [FK96]. Hence, it is not clear at all, which of the currently known algorithms is the best choice on which kind of instances.

This thesis' contributions. In this thesis, we try to shed some light on the open questions discussed in the previous paragraphs.

As for the easy classes, there are two main questions. First, how easy are the easy classes? Can we do better than polynomial running time? The second one is to find new easy classes. In this thesis we work on both questions. We analyze the known easy classes with the intention of tightening the known resource bounds. Thereby, we show some of the easy classes to be solvable with logarithmic space, which improves the known polynomial time bounds. And we discover some new easy classes.

As for the known MONET algorithms, our conjecture is that none of the currently known approaches is fast in the sense of polynomial (resp., output-polynomial) running time. We start working on proving our conjecture, by giving non-trivial lower bounds on the running times of several algorithms. From the point of view of practical applications there is also another interesting open question, namely that for the performance of the theoretically fastest MONET algorithm, FK-algorithm B, in algorithmic experiments. Using the Algorithm Engineering methodology we compare FK-algorithm B to FK-algorithm A and show it to be competitive on our testbed. This result somehow adjusts the folklore assumption that FK-algorithm B should be practically much slower than its theoretically worse relative, FK-algorithm A.

As for computational complexity issues, we examine the fixed-parameter tractability of MONET, a subject that has not been addressed in the literature so far. Informally, a problem is said to be fixed-parameter tractable if there is an algorithm whose running time is arbitrary (exponential or even worse) in a given parameter but only includes polynomial terms for the input size (cf. Chapter 7 for more details). The idea then is that for small values of the parameter the running

time of the algorithm behaves like being polynomial. We show MONET to be in the class FPT of fixed-parameter tractable problems with respect to several parameters.

1.2 Legend to this thesis

This thesis is organized as follows. In Chapter 2, we introduce basic notations and concepts as well as the problem MONET. Chapter 3 then gives some further motivation for studying MONET in the sense that we collect problems equivalent or strongly related to MONET from very different fields. Afterwards, in Chapter 4, we introduce easy classes of MONET that are restrictions of the input formulas that allow for polynomial time solutions. In contrast, Chapter 5 contains algorithms solving the general problem MONET without any restrictions on the input. Some experimental results for important MONET algorithms follow in Chapter 6. In Chapter 7, we then turn to the emerging field of parameterized complexity and analyze MONET in that setting. Finally, some concluding remarks follow in Chapter 8.

Parts of this thesis have already appeared or are accepted for publication in refereed journals, in refereed conference and workshop proceedings, as technical reports, or as manuscripts. In particular, Chapter 4 contains results from [GHM05, GHM08], and [Hag06]; Chapter 5 contains results from [Hag07a] and [EHR08a]; Chapter 6 contains results from [HHM09], whereas the choice of test instances is based on observations contained in [BHHM07]; and Chapter 7 contains results from [Hag07b] and [EHR08b].