Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 45 (2006)

Approximability of Minimum AND-Circuits*

Jan Arpe†,1 Bodo Manthey‡,2

1 Universität zu Lübeck
Institut für Theoretische Informatik
Ratzeburger Allee 160
23538 Lübeck, Germany
arpe@tcs.uni-luebeck.de

2 Universität des Saarlandes
Informatik
Postfach 151150
66041 Saarbrücken, Germany
manthey@cs.uni-sb.de

Abstract

Given a set of monomials, the Minimum AND-Circuit problem asks for a circuit that computes these monomials using AND-gates of fan-in two and being of minimum size. We prove that the problem is not polynomial time approximable within a factor of less than 1.0051 unless $\mathsf{P} = \mathsf{NP}$, even if the monomials are restricted to be of degree at most three. For the latter case, we devise several efficient approximation algorithms, yielding an approximation ratio of 1.278. For the general problem, we achieve an approximation ratio of $d - 3/2$, where d is the degree of the largest monomial. In addition, we prove that the problem is fixed parameter tractable with the number of monomials as parameter. Finally, we reveal connections between the Minimum AND-Circuit problem and several problems from different areas.

1 Introduction

Given a set of Boolean monomials, the Minimum-AND-Circuit problem asks for a circuit that consists solely of logical AND-gates with fan-in two and that computes these monomials. The monomials may for example arise in the DNF-representation of a Boolean function or in some decomposed or factored form. Thus, the Minimum-AND-Circuit problem is of fundamental interest for automated circuit design, see Charikar et al. [4, Sect. VII.B] and references therein. In this paper, we assume that all variables always occur positively; no negations are permitted. The investigation of minimum AND-circuits from a complexity theoretic standpoint was proposed by Charikar et al. [4]. According to them, no approximation guarantees have been proved at all yet.

We give the first positive and negative approximability results for the Minimum-AND-Circuit problem. Specifically, we show that the problem is not approximable

*A preliminary version will appear in the Proc. of the 10th Scandinavian Workshop on Algorithm Theory (SWAT 2006), Lecture Notes in Computer Science, Springer.

†Supported by DFG research grant RE 672/4.

‡Work done as a member of the Institut für Theoretische Informatik of the Universität zu Lübeck and supported by DFG research grant RE 672/3.

ISSN 1433-8092

within a factor of less than $\frac{983}{978}$ unless $\mathsf{P} = \mathsf{NP}$, even if the monomials are restricted to be of maximum degree three (Sect. 3). For the latter variant, we present several algorithms and prove an upper bound of 1.278 on its approximation ratio (Sect. 4). If the number of occurrences of each submonomial of size two in the input instance, called the *multiplicity*, is bounded from above by a constant $\mu \geq 3$, similar hardness results are achieved (Sect. 3) and the upper bounds are slightly improved (Sect. 4.4). For $\mu = 2$, the problem is even in P (Sect. 4.2). However, if we allow the monomials to be of degree four, it remains open whether the case $\mu = 2$ is solvable in polynomial time. We prove that the general problem with multiplicity bounded by μ is approximable within a factor of μ (Sect. 6.2).

In general, restricting the monomials to be of degree at most d admits a straightforward approximation within a factor of $d - 1$, which we improve to $d - 3/2$ (Sect. 6.1). If the degrees are required to be exactly d and in addition, the multiplicity is bounded by μ, we prove an upper bound on the approximation ratio of $\mu(d-1)/(\mu + d - 2)$ (Sect. 6.2).

Besides from fixing the maximum degree or the multiplicity of the input monomials, we consider fixing the number of monomials (Sect. 5). We show that Minimum-AND-Circuit instances have small problem kernels, yielding a fixed parameter tractable algorithm (for terminology, see Downey and Fellows [8]). In other words, the Minimum-AND-Circuit problem restricted to instances with a fixed number of monomials is in P.

There are two evident generalizations of AND-circuits. The first one is to ask for a minimum *Boolean* circuit (with AND-, OR-, and NOT-gates) that computes a given function. This problem has, for example, been investigated by Kabanets and Cai [11]; its complexity is still open. Even if the functions to be computed consist solely of positive monomials, allowing the circuit to contain AND- and OR-gates can reduce the circuit size, as has been shown by Tarjan [15] (see also Wegener [17]).

The second one is to consider monomials over other structures such as the additive group of integers or the monoid of finite words over some alphabet (see also Sect. 6.3). While the former structure leads to *addition chains* [13, Sect. 4.6.3], the latter yields the *smallest grammar problem* which has attracted much attention in the past few years; a summary of recent results has been provided by Charikar et al. [4, Sect. I and II]. In fact, Charikar et al.'s suggestion to investigate minimum AND-circuits was motivated by the lack of understanding the hierarchical structure of grammar-based compression. In particular, there is a bunch of so-called *global algorithms* for the smallest grammar problem which are believed to achieve quite good approximation ratios, but no one has yet managed to prove this.

2 Preliminaries

For $i \in \mathbb{N}$, let $[i] = \{1, \ldots, i\}$.

2.1 Monomials and Circuits

We study the design of small circuits that simultaneously compute given monomials $M_1, \ldots, M_k$ over a set of Boolean variables $X = \{x_1, \ldots, x_n\}$. More precisely, a

(Boolean) monomial is an AND-product of variables of a subset of X, and by an *AND-circuit*, we mean a circuit consisting solely of AND-gates with fan-in two. We identify a monomial $M = x_{i_1} \wedge \ldots \wedge x_{i_d}$ with the subset $\{x_{i_1}, \ldots, x_{i_d}\}$, which we denote by M again. Since we only use one type of operation, we often omit the $\wedge$ signs and simply write $x_{i_1} \ldots x_{i_d}$. The *degree* of M is $|M|$.

An *(AND-)circuit* $\mathcal{C}$ over X is a directed acyclic graph with node set $G(\mathcal{C})$ (*gates*) and edge set $W(\mathcal{C})$ (*wires*) satisfying the following properties:

1. To each input variable $x \in X$ is associated exactly one *input gate* $g_x \in G(\mathcal{C})$ that has indegree zero and arbitrary outdegree.

2. All nodes that are not input nodes have indegree exactly two and arbitrary outdegree. These nodes are called *computation gates*.

We denote the set of computation gates of $\mathcal{C}$ by $G^*(\mathcal{C})$, i.e., $G^*(\mathcal{C}) = G(\mathcal{C}) \setminus \{g_x \mid x \in X\}$. The *circuit size* of $\mathcal{C}$ is equal to the number of computation gates of $\mathcal{C}$, i.e., $\text{size}(\mathcal{C}) = |G^*(\mathcal{C})|$. A gate g *computes* the monomial $\text{val}(g)$, which is defined as follows:

1. $\text{val}(g_x) = x$.

2. For a computation gate g with predecessors g_1 and g_2, $\text{val}(g) = \text{val}(g_1) \wedge \text{val}(g_2)$.

The circuit $\mathcal{C}$ *computes* a Boolean monomial M if some gate in $\mathcal{C}$ computes M. It computes a set $\mathcal{M}$ of monomials if it computes all monomials in $\mathcal{M}$. Such a circuit is called *a circuit for* $\mathcal{M}$. The gates that compute the monomials in $\mathcal{M}$ are referred to as the *output gates*. Note that output gates, unless they are input gates at the same time, are computation gates, too, and hence contribute to the circuit size. This makes sense since in a physical realization of the circuit, such gates have to perform an AND-operation—in the same way as all non-output computation gates.

A *subcircuit* $\mathcal{C}'$ of a circuit $\mathcal{C}$ is a subgraph of $\mathcal{C}$ that is again a circuit. In particular, $\mathcal{C}'$ contains all "induced" input gates. For $g \in G(\mathcal{C})$, let $\mathcal{C}_g$ be the minimal subcircuit of $\mathcal{C}$ containing g. Since $\mathcal{C}_g$ is a circuit, it contains all input gates g_x with $x \in \text{val}(g)$. Moreover, $\mathcal{C}_g$ contains at least $|\text{val}(g)| - 1$ computation gates. Let $\mathcal{M}$ be a set of monomials and $\mathcal{C}$ be a circuit for $\mathcal{M}$. For each $M \in \mathcal{M}$, denote the gate that computes M by g_M and write $\mathcal{C}_M$ for $\mathcal{C}_{g_M}$. The *frequency* of a computation gate $g \in G^*(\mathcal{C})$ (with respect to $\mathcal{M}$) is the number of monomials that g is used for, i.e.,

$$\text{freq}_{\mathcal{M}}(g) = |\{M \in \mathcal{M} \mid g \in G(\mathcal{C}_M)\}| \ .$$

A gate g with $\text{freq}_{\mathcal{M}}(g) = 0$ is called *useless*. The following straightforward equation proves very useful:

$$\sum_{g \in G^*(\mathcal{C})} \text{freq}_{\mathcal{M}}(g) = \sum_{M \in \mathcal{M}} \text{size}(\mathcal{C}_M) \ . \tag{1}$$

A gate is called *strict* if its predecessors compute disjoint monomials. A circuit is called *strict* if all of its gates are strict. It is not hard to see that any non-strict circuit for a Min-AC instance $\mathcal{M}$ of maximum degree at most four can be turned into a strict circuit for $\mathcal{M}$ of the same size. As we will show in the proof of Lemma 6.1, this is not true if the monomials are allowed to be of degree five or more.

Let $S \subseteq X$. The *multiplicity of* S in $\mathcal{M}$ is the number of occurrences of S in $\mathcal{M}$ as a submonomial, i.e.,

$$\operatorname{mult}_{\mathcal{M}}(S) = |\{M \in \mathcal{M} \mid S \subseteq M\}| \ .$$

The *maximum multiplicity of* $\mathcal{M}$ is defined by

$$\operatorname{mult}(\mathcal{M}) = \max_{|S| \geq 2} \operatorname{mult}_{\mathcal{M}}(S) \ .$$

It is equal to the number of occurrences of the most frequent pair of variables in $\mathcal{M}$. For all computation gates g of a circuit $\mathcal{C}$ for $\mathcal{M}$, we have

$$\operatorname{freq}_{\mathcal{M}}(g) \leq \operatorname{mult}_{\mathcal{M}}(\operatorname{val}(g)) \leq \operatorname{mult}(\mathcal{M}) \ . \qquad (2)$$

2.2 Optimization Problems

For an introduction to the approximation theory of combinatorial optimization problems, we refer to Ausiello et al. [3]. For an optimization problem P and an instance I for P, we write $\operatorname{opt}_P(I)$ for the measure of an optimum solution for I.

Let $\mathcal{A}$ be an approximation algorithm for P, i.e., an algorithm, that on an instance I of P, outputs an admissible solution $\mathcal{A}(I)$. The *approximation ratio* $\rho_{\mathcal{A}}(I)$ *of* $\mathcal{A}$ *at* I is the ratio between the measure $m(\mathcal{A}(I))$ of a solution $\mathcal{A}(I)$ output by $\mathcal{A}$ and the size of an optimal solution, i.e., $\rho_{\mathcal{A}}(I) = \frac{m(\mathcal{A}(I))}{\operatorname{opt}_P(I)}$. The *approximation ratio* $\rho_{\mathcal{A}}$ *of* $\mathcal{A}$ is the worst-case ratio of all ratios $\rho_{\mathcal{A}}(I)$, i.e., $\rho_{\mathcal{A}} = \max_I \rho_{\mathcal{A}}(I)$.

The Minimum-AND-Circuit problem, abbreviated Min-AC, is defined as follows:

> Given a set of monomials $\mathcal{M} = \{M_1, \ldots, M_k\}$ over a set of Boolean input variables $X = \{x_1, \ldots, x_n\}$, find a circuit $\mathcal{C}$ of minimum size that computes $\mathcal{M}$.

Throughout the paper, k denotes the number of monomials, n denotes the number of input variables, and $N = \sum_{M \in \mathcal{M}} |M|$ denotes the total input size. In addition, we always assume that $X = \bigcup_{M \in \mathcal{M}} M$.

We denote by Min-d-AC the Minimum-AND-Circuit problem with instances restricted to monomials of degree *at most d*. The problem where the degrees are required to be *exactly d* is denoted by Min-Ed-AC.

A *vertex cover* of a graph G is a subset $\tilde{V} \subseteq V$ such that every edge has at least one endpoint in $\tilde{V}$. This definition also applies to hypergraphs. Aside from Min-AC, we will encounter the following optimization problems: The vertex cover problem, denoted by Min-VC, is defined as follows:

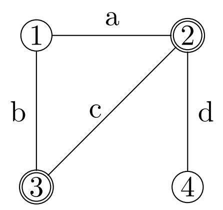
> Given an undirected graph G, find a vertex cover of G of minimum size.

The restriction of Min-VC to graphs of maximum degree d is denoted by Min-d-VC. A hypergraph is called *r-uniform* if all of its edges have size exactly r. The vertex cover problem for r-uniform hypergraphs, denoted by Min-r-UVC, is:

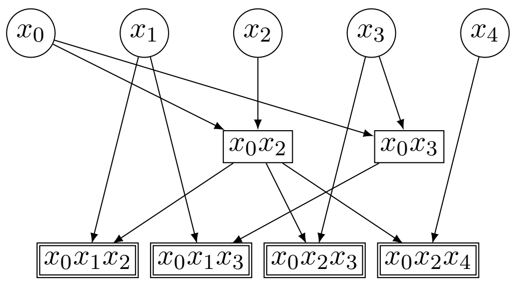
> Given an r-uniform hypergraph G, find a vertex cover of G of minimum size.

Finally, Maximum-Coverage is the following optimization problem:

> Given a hypergraph G and a number $r \in \mathbb{N}$, find r edges $e_1, \ldots, e_r \in E$ such that $\bigcup_{i=1}^{r} e_i$ is of maximum cardinality.



(a) Graph with vertex cover $\{2, 3\}$.



(b) AND-circuit for the Min-3-AC instance $\mathcal{M} = \{M_a, M_b, M_c, M_d\}$ with $M_a = x_0x_1x_2$, $M_b = x_0x_1x_3$, $M_c = x_0x_2x_3$, and $M_d = x_0x_2x_4$. Input gates are represented as circle nodes, whereas computation gates are boxed. In addition, output gates have a double box.

Figure 1: A graph with a vertex cover and the corresponding AND-circuit as constructed in the proof of Lemma 3.1.

3 Hardness

In this section, we prove that the Minimum-AND-Circuit problem is NP-complete and that there is no polynomial-time approximation algorithm that achieves an approximation ratio of less than $\frac{983}{978}$ unless P = NP. To do this, we reduce Min-VC to Min-AC.

Let $G = (V, E)$ be an undirected graph with $n = |V|$ vertices and $m = |E|$ edges. We construct an instance of Min-AC as follows. For each node $v \in V$, we have a variable x_v. In addition, there is an extra variable x_0. For each edge $e = \{v, w\} \in E$, we construct the monomial $M_e = x_0x_vx_w$. Our instance of Min-AC is then $\mathcal{M}_G = \{M_e \mid e \in E\}$. Note that $|M| = 3$ for all $M \in \mathcal{M}_G$. Moreover, if G has maximum degree Δ, then $\mathcal{M}_G$ has maximum multiplicity Δ. Clearly, $\mathcal{M}_G$ can be constructed in polynomial time. An example is shown in Figure 1.

Lemma 3.1. *Let G and $\mathcal{M}_G$ be as described above. Then $\mathrm{opt}_{\textsf{Min-AC}}(\mathcal{M}_G) = |E| + \ell$, where $\ell = \mathrm{opt}_{\textsf{Min-VC}}(G)$. Furthermore, given a circuit $\mathcal{C}$ of size $|E| + \ell'$ for $\mathcal{M}_G$, we can compute a vertex cover $\tilde{V}$ of G with $|\tilde{V}| \leq \ell'$ in polynomial time.*

Proof. We prove the above lemma by showing that every vertex cover of size ℓ' yields a circuit of size $|E| + \ell'$ and vice versa.

Suppose we are given a vertex cover $\tilde{V} \subseteq V$ of G of size $|\tilde{V}| = \ell'$. We construct an AND-circuit for $\mathcal{M}_G$ as follows. The circuit consists of two layers. In the first layer, there is one gate g_v for each node $v \in \tilde{V}$. The gate g_v computes x_0x_v. In the second layer, the monomials in $\mathcal{M}_G$ are computed: for each edge $e \in E$, there is a gate g_e. If $e = \{v, w\}$ with $v \in \tilde{V}$, then g_e has computation gate g_v and input gate g_{x_w} as predecessors, thus computes M_e. The described circuit computes $\mathcal{M}_G$ and uses $\ell + |E|$ gates.

Now suppose that there is a circuit $\mathcal{C}$ of size $\ell' + |E|$ that computes $\mathcal{M}_G$. Since each $M \in \mathcal{M}_G$ is of degree 3, we can assume that $\mathcal{C}$ has exactly two layers, the second one containing the $|E|$ output gates that compute the monomials M_e. Let F denote the set of the remaining ℓ' gates in the first layer. For a gate $g \in F$, let $v(g)$ be a node such that $g_{x_{v(g)}}$ is an input of g. Such a node exists since g has two predecessors and at least one of them is different from x_0. If both predecessors are different from x_0, then we choose one of them arbitrarily. We claim that $\tilde{V} = \{v(g) \mid g \in F\}$ forms a vertex cover of G. To prove this, let $e = \{v, w\} \in E$. The gate that computes M_e must be connected to at least one gate $g \in F$. This gate in turn has an incoming edge from either g_{x_v} or g_{x_w} (or both). Thus $v \in \tilde{V}$ or $w \in \tilde{V}$. Given the circuit, the vertex cover can clearly be constructed in polynomial time. □

Theorem 3.2. Min-AC *is* NP*-complete,* APX*-hard and cannot be approximated in polynomial time within a factor of less than* $\frac{983}{978} > 1.0051$ *unless* P = NP*.*

This holds even for Min-3-AC *restricted to instances with maximum multiplicity six.*

Proof. The NP-completeness and APX-completeness follows from Theorem 3.4 below. For the inapproximability, we exploit a result of Chlebík and Chlebíková.

Theorem 3.3 (Chlebík and Chlebíková [5]). *Given an instance G of* Min-6-VC *with n vertices, it is, for every sufficiently small $\epsilon > 0$,* NP*-hard to decide whether the size of a minimum vertex cover of G is at most $(\frac{474}{494}+\epsilon)\cdot n$ or at least $(\frac{484}{494}-\epsilon)\cdot n$.*

Thus, it is NP-hard to decide whether the instance of Min-AC corresponding to the graph can be computed by a circuit of size at most $|E| + (\frac{474}{494} + \epsilon)|V|$ or if every circuit for this instance has a size of at least $|E| + (\frac{484}{494} - \epsilon)|V|$ for sufficiently small $\epsilon > 0$. The inapproximability bound follows by plugging in the inequality $|E| \leq 3|V|$. □

Theorem 3.4. Min-3-AC *restricted to instances of maximum multiplicity three is* NP*-complete,* APX*-hard, and cannot be approximated in polynomial time within a factor of less than* $\frac{269}{268} > 1.0037$ *unless* P = NP*.*

Proof. The NP-completeness and APX-hardness follow from the NP-completeness and APX-completeness of Min-3-VC [2, 9].

What remains to be proved is the inapproximability bound. Again, we exploit a result of Chlebík and Chlebíková.

Theorem 3.5 (Chlebík and Chlebíková [5]). *Given an instance G of* Min-3-VC *with n vertices, it is, for every sufficiently small $\epsilon > 0$,* NP*-hard to decide whether the size of a minimum vertex cover of G is at most $(\frac{494}{564}+\epsilon)\cdot n$ or at least $(\frac{499}{564}-\epsilon)\cdot n$.*

Analogously to the proof of Theorem 3.2, we obtain the inapproximability result for Min-3-AC by plugging in the inequality $|E| \leq (3/2) \cdot |V|$. □

Since for fixed d, Min-d-AC can be approximated within a constant factor (see Section 6.1), the problem is in APX and thus APX-complete.

4 Approximation Algorithms for Min-3-AC

In this section, we provide several polynomial-time approximation algorithms for Min-3-AC, the problem of computing minimum AND-circuits for monomials of degree at most three. Note that the lower bounds proved in Section 3 hold already for Min-E3-AC.

Without loss of generality, we may assume that all monomials have degree exactly three for the following reasons. Firstly, we do not need any computation gates to compute monomials of degree one, so we can delete such monomials from the input. Secondly, for each input monomial of size two, we are forced to construct an output gate. On the other hand, we should use this gate wherever we can for other input monomials, so we can delete all monomials of degree two from the input and substitute all occurrences of such monomials in the other monomials by extra variables. We repeat this process until no more monomials of size two are in the input. As we have already mentioned in Section 2, we can assume without loss of generality that a circuit for a Min-3-AC instance is strict. Moreover, if all monomials are exactly of degree three, then a circuit can be assumed to consist of two layers of computation gates. The gates of the first layer compute monomials of size two, and the gates of the second layer are the output gates.

Since each monomial M of degree at most three can be computed by a circuit of size two, we can construct a *trivial circuit* $\mathcal{C}_{\text{triv}}$ for a Min-3-AC instance $\mathcal{M}$ of size $2k$, where k is the number of monomials. On the other hand, the computation of k monomials obviously requires at least k gates. Thus, we obtain an upper bound of 2 on the polynomial-time approximation ratio for Min-3-AC. In the following, we show how to improve this bound.

4.1 Algorithm "Cover"

We first reduce Min-3-AC to Min-3-UVC, the problem of finding a vertex cover in three-uniform hypergraphs. Subsequently, we will present our algorithms.

Let $\mathcal{M}$ be a Min-3-AC instance. We introduce some notation that will be used throughout this paper. For $M \in \mathcal{M}$, let

$$\text{pairs}(M) = \{S \subseteq X \mid |S| = 2 \wedge S \subseteq M\}$$

be the set of pairs contained in M. Note that $|\text{pairs}(M)| = 3$. Furthermore, let $\text{pairs}(\mathcal{M}) = \bigcup_{M \in \mathcal{M}} \text{pairs}(M)$ be the set of all pairs of variables appearing in $\mathcal{M}$.

Let $\mathcal{C}$ be a circuit for $\mathcal{M}$. Then $\mathcal{C}$ consists of two layers, the second one containing the $k = |\mathcal{M}|$ output gates. In the first layer, certain monomials of size two are computed: for each monomial $M \in \mathcal{M}$, one of the pairs $S \in \text{pairs}(M)$ has to be computed at the first level of $\mathcal{C}$. The task is thus to find a minimum set of pairs $S \in \text{pairs}(\mathcal{M})$ such that each monomial $M \in \mathcal{M}$ contains one such pair. This corresponds to finding a minimum vertex cover of the three-uniform hypergraph $H(\mathcal{M}) = (V, E)$ described in the following. The node set is the set of pairs appearing in $\mathcal{M}$, i.e., $V = \text{pairs}(\mathcal{M})$, and for each monomial $M \in \mathcal{M}$, there is a hyperedge containing the pairs that appear in M, i.e., $E = \{\text{pairs}(M) \mid M \in \mathcal{M}\}$. A circuit $\mathcal{C}$ for $\mathcal{M}$ with gates computing the pairs $S_1, \ldots, S_\ell$ at its first level corresponds to the vertex cover of $H(\mathcal{M})$ given by $\{S_i \mid 1 \leq i \leq \ell\}$ and vice versa. We denote the circuit corresponding to a vertex cover $\tilde{V}$ by $\mathcal{C}_{\tilde{V}}$. By the preceding discussion, we have shown

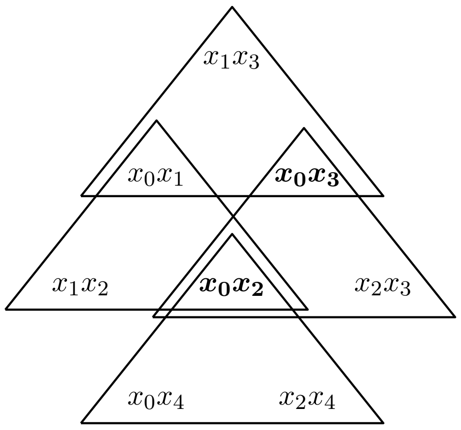


Figure 2: The hypergraph $H(\mathcal{M})$ associated with the Min-AC instance $\mathcal{M}$ introduced in Figure 1. Each triangle represents a hyperedge. The two bold monomials constitute a vertex cover.

Cover

```
1: Input $\mathcal{M} = \{M_1, \ldots, M_k\}$.
2: Compute the hypergraph $H = H(\mathcal{M})$.
3: Compute greedily an inclusion-maximal matching $\tilde{E}$ in $H$, i.e., a
   collection of disjoint hyperedges that cannot be enlarged.
4: Let $\tilde{V} = \bigcup_{e \in \tilde{E}} e$.
5: Compute $\mathcal{C} = \mathcal{C}_{\tilde{V}}$.
6: Output $\mathcal{C}$.
```

Figure 3: Algorithm Cover for Min-3-AC.

Lemma 4.1. *Let $\tilde{V}$ be a vertex cover of $H(\mathcal{M})$. Then* $\operatorname{size}(\mathcal{C}_{\tilde{V}}) = k + |\tilde{V}|$. *In particular,*

$$\operatorname{opt}_{\mathsf{Min\text{-}3\text{-}AC}}(\mathcal{M}) = k + \operatorname{opt}_{\mathsf{Min\text{-}3\text{-}UVC}}(H(\mathcal{M})) .$$

Our first polynomial-time approximation algorithm for Min-3-AC, which is presented in Figure 3, is based on the reduction we have just presented. The set $\tilde{V}$ consists of all nodes that are incident with the matching $\tilde{E}$. Thus the size of $\tilde{V}$ equals $3 \cdot |\tilde{E}|$. $\tilde{V}$ is a vertex cover since $\tilde{E}$ cannot be enlarged. On the other hand, any vertex cover of $H(\mathcal{M})$ must include at least one vertex from each hyperedge of the maximum matching $\tilde{E}$, so any vertex cover of $IG(\mathcal{M})$ must be of size at least $|\tilde{E}|$. In conclusion, we have $|\tilde{V}| \leq 3 \cdot \operatorname{opt}_{\mathsf{Min\text{-}3\text{-}UVC}}(H(\mathcal{M}))$. Together with Lemma 4.1 this proves

Lemma 4.2. *Let* $\operatorname{opt}_{\mathsf{Min\text{-}3\text{-}AC}}(\mathcal{M}) = k + \ell$. *Then* Cover *outputs a circuit $\mathcal{C}_{\text{Cover}}$ for $\mathcal{M}$ of size at most $k + 3 \cdot \ell$.*

E.g., for instances $\mathcal{M}$ that consist of pairwise disjoint monomials of degree three, $\operatorname{size}(\mathcal{C}_{\text{Cover}}) = k + 3\ell$ is indeed achieved (with $\ell = k$).

In case that $\ell \geq \frac{1}{3}k$, Cover outputs a circuit that is larger than the trivial one. Choosing to output the trivial circuit instead, yields an algorithm with an

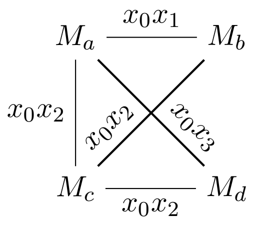


Figure 4: Intersection graph $IG(\mathcal{M})$ associated with the Min-AC instance $\mathcal{M}$ introduced in Figure 1. The edges are labeled by the pairs that their endpoints have in common. The bold edges constitute a maximal matching.

MATCH

```
1: Input $\mathcal{M} = \{M_1, \ldots, M_k\}$.
2: Compute $G = IG(\mathcal{M})$.
3: Compute a matching $\tilde{E}$ of $G$ of maximum cardinality.
4: For each $\{M, M'\} \in \tilde{E}$:
5:   Add a gate computing $M \cap M'$ to $\mathcal{C}$.
6:   Add subcircuits computing $M$ and $M'$ to $\mathcal{C}$, using two additional
     gates.
7: For each $M \in \mathcal{M} \setminus \bigcup_{e \in \tilde{E}} e$ (not incident with $\tilde{E}$):
8:   Add a subcircuit computing $M$, using $|M| - 1$ gates.
9: Output $\mathcal{C}$.
```

Figure 5: Algorithm MATCH for Min-3-AC.

approximation ratio of

$$\frac{\max\{k + 3\ell, 2k\}}{k + \ell} \leq \frac{2}{4/3} = \frac{3}{2} \,.$$

Thus, we have already found an algorithm that achieves a non-trivial approximation ratio. In the course of this paper, we will improve this ratio to below 1.3.

4.2 Algorithm "Match"

Before we present our next algorithm, we introduce another technical utility. Associate with $\mathcal{M}$ the *intersection graph* $IG(\mathcal{M})$ defined as follows: the nodes of $IG(\mathcal{M})$ are the monomials of $\mathcal{M}$, and two monomials $M, M' \in \mathcal{M}$ are connected by an edge iff $|M \cap M'| = 2$. An example is shown in Figure 4.

Algorithm MATCH, which is presented in Figure 5, is a polynomial-time algorithm; in particular, a maximum matching in $IG(\mathcal{M})$ can be computed in time $O(n^{2.5})$ [1]. To bound the approximation ratio of MATCH, we need the following

Lemma 4.3. *Let* $\mathrm{opt}_{\text{Min-3-AC}}(\mathcal{M}) = k + \ell$ *and* $\tilde{E}$ *be the matching computed in step 3 of* MATCH *on input* $\mathcal{M}$*. Then* $|\tilde{E}| \geq \frac{1}{2}(k - \ell)$*.*

Proof. Let $\mathcal{C}$ be a minimum circuit for $\mathcal{M}$. Let $\mathcal{S} = \{S_1, \ldots, S_\ell\}$ be the set of pairs computed by the gates at the first level of $\mathcal{C}$. We construct a matching of $IG(\mathcal{M})$

of size $\frac{1}{2}(k-\ell)$. For each $M \in \mathcal{M}$, we select an $S_M \in \mathcal{S} \cap \mathrm{pairs}(M)$. This partitions the monomials into sets $E_S = \{M \in \mathcal{M} \mid S_M = S\}$, $S \in \mathcal{S}$. Since all monomials in E_S have the pair S in common, each set E_S forms a clique of the intersection graph $IG(M)$. Hence we can choose $\lfloor \frac{|E_S|}{2} \rfloor \geq \frac{|E_S|-1}{2}$ disjoint edges of $IG(\mathcal{M})$ with endpoints in E_S. In total, this yields a matching of size at least

$$\sum_{S \in \mathcal{S}} \frac{|E_S|-1}{2} = \frac{1}{2}(|\mathcal{M}| - |\mathcal{S}|) = \frac{1}{2}(k-\ell)\ .$$

□

Lemma 4.4. *Let* $\mathrm{opt}_{\mathsf{Min\text{-}3\text{-}AC}}(\mathcal{M}) = k + \ell$. *Then* Match *outputs a circuit* $\mathcal{C}_{\text{Match}}$ *for* $\mathcal{M}$ *of size at most* $\frac{3}{2} \cdot k + \frac{1}{2} \cdot \ell$.

Proof. Each edge of the matching $\tilde{E}$ saves us at least one gate compared with the trivial solution (since we compute two monomials with only three gates). Hence $\mathrm{size}(\mathcal{C}_{\text{Match}}) = 2k - |\tilde{E}|$. By Lemma 4.3, $|\tilde{E}| \geq \frac{1}{2}(k-\ell)$. Consequently,

$$\mathrm{size}(\mathcal{C}_{\text{Match}}) = 2k - |\tilde{E}| \leq \frac{3}{2}k + \frac{1}{2}\ell\ .$$

□

It is not hard to construct instances for which the upper bound on $\mathrm{size}(\mathcal{C}_{\text{Match}})$ stated in Lemma 4.4 is indeed achieved.

Although the analysis of Match is not needed for our best upper bound result for Min-3-AC, the algorithm is the only one for which we can prove a non-trivial approximation ratio for Min-d-AC in case that $d \geq 4$. We will discuss this issue in Section 6.1.

For Min-3-AC with instances restricted to a multiplicity of at most two, Match computes an optimum solution.

Lemma 4.5. *Let* $\mathcal{M}$ *be a* Min-3-AC *instance with multiplicity at most two. Then* Match *outputs a circuit* $\mathcal{C}_{\text{Match}}$ *of minimum size for* $\mathcal{M}$.

Proof. Since every edge of the matching $\tilde{E}$ computed by Match in step 3 saves exactly one gate, we have $\mathrm{size}(\mathcal{C}_{\text{Match}}) = 2k - |\tilde{E}|$.

Claim 4.6. *An arbitrary circuit* $\mathcal{C}$ *for* $\mathcal{M}$ *yields a matching* F *of the intersection graph* $IG(M)$ *of size* $2k - \mathrm{size}(\mathcal{C})$.

Proof of Claim 4.6. Without loss of generality, assume that $\mathcal{C}$ is strict. Let $\ell = 2k - \mathrm{size}(\mathcal{C})$. Since $\mathcal{M}$ has multiplicity at most two, each gate is used for at most two monomials and thus saves at most one gate. But then there must be exactly ℓ gates that are used in two monomials. Let these gates be $g_1, \ldots, g_\ell$ with g_i used for M_i and M'_i, $i \in [\ell]$. We claim that $F = \{\{M_i, M'_i\} \mid i \in [\ell]\}$ is a matching in $IG(\mathcal{M})$. Clearly, $\{M_i, M'_i\} \in E$. Moreover, the edges are disjoint since otherwise two different gates g_i and g_j would be used for the same monomial, which would contradict the strictness of $\mathcal{C}$. Consequently, $\tilde{F}$ is indeed a matching of size ℓ. □

Greedy

```
1: Input $\mathcal{M} = \{M_1, \ldots, M_k\}$.
2: While there exists an $S \in \binom{X}{2}$ such that $|\{M \in \mathcal{M} \mid S \subseteq M\}| \geq 3$:
3:   Arbitrarily select $S \in \binom{X}{2}$ with maximum $|\{M \in \mathcal{M} \mid S \subseteq M\}|$.
4:   Add a gate computing $S$ to $\mathcal{C}$.
5:   For each $M \in \mathcal{M}$ with $S \subseteq M$:
6:     Add subcircuit computing $M$ to $\mathcal{C}$, using at most $|M| - 2$
       additional gates.
7:     $\mathcal{M} \leftarrow \mathcal{M} \setminus \{M\}$.
8: $\mathcal{C}' \leftarrow \textsc{Match}(\mathcal{M})$.
9: $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$.
10: Output $\mathcal{C}$.
```

Figure 6: Algorithm Greedy for Min-3-AC.

Now let $\mathcal{C}_{\text{opt}}$ be a circuit for $\mathcal{M}$ of minimum size. By the above claim, the intersection graph $IG(M)$ has a matching of size $2k - \text{size}(\mathcal{C}_{\text{opt}})$. Since $\tilde{E}$ is a matching of maximum size, $|\tilde{E}| \geq 2k - \text{size}(\mathcal{C}_{\text{opt}})$. Hence $\text{size}(\mathcal{C}_{\textsc{Match}}) = 2k - |\tilde{E}| \geq \text{size}(\mathcal{C}_{\text{opt}})$. □

Corollary 4.7. Min-3-AC *with instances restricted to a maximum multiplicity of at most two can be solved in polynomial time.*

4.3 Algorithm "Greedy"

Our last algorithm Greedy is presented in Figure 6. It greedily constructs gates for pairs that occur most frequently in the input instance $\mathcal{M}$ until each remaining pair is shared by at most two monomials. At that point, instead of proceeding in an arbitrary order, an optimal solution is computed for the remaining monomials. The latter task is achieved by Match, as we have shown in Lemma 4.5.

Lemma 4.8. *Let $\mathcal{M} = \{M_1, M_2, \ldots, M_k\}$ be an instance for* Min-3-AC *such that $\text{opt}_{\mathsf{Min\text{-}3\text{-}AC}}(\mathcal{M}) = k + \ell$. Then* Greedy *outputs a circuit $\mathcal{C}_{\textsc{Greedy}}$ for $\mathcal{M}$ of size at most*

$$\min\left\{\frac{4}{3} \cdot k + \ell, \left(1 + \frac{1}{e^2}\right) k + 2\ell\right\} .$$

Proof. Clearly, for every $M \in \mathcal{M}$, Greedy eventually adds M to $\mathcal{C} = \mathcal{C}_{\textsc{Greedy}}$, hence $\mathcal{C}$ computes $\mathcal{M}$. Let k_1 denote the number of monomials in $\mathcal{M}$ that are computed by $\mathcal{C}$ after steps 1–8 and k_2 denote the size of $\mathcal{C}'$ computed in step 9 of Greedy. Since the sets S selected in step 4 are all shared by at least three monomials each, at most $k_1/3$ gates are added to $\mathcal{C}$ in step 5. In addition, k_1 gates are added to $\mathcal{C}$ in step 7. We denote by $\mathcal{M}'$ the set of monomials that remain in $\mathcal{M}$ after the `while` loop is exited. By Lemma 4.5, the circuit $\mathcal{C}'$ constructed in step 9 is of minimum size for $\mathcal{M}'$. Let $\mathcal{C}_{\text{opt}}$ be a circuit for $\mathcal{M}$ of minimum size $k + \ell$. Clearly, we can construct an alternative circuit for $\mathcal{M}'$ by only using the ℓ non-output gates of $\mathcal{C}$

and k_2 output gates for the monomials $M \in \mathcal{M}'$, i.e., size$(\mathcal{C}') \le k_2 + \ell$. In total, size$(\mathcal{C}) \le \frac{4}{3} \cdot k + \ell$.

Next we show that also size$(\mathcal{C}) \le \left(1 + \frac{1}{e^2}\right) k + 2\ell$. Although not necessary for our investigations, we start by showing that size$(\mathcal{C}) \le \left(1 + \frac{1}{e}\right) k + \ell$ since the proof of this latter bound is easily understandable and the proof of the former bound follows the same line. Let $H(\mathcal{M}) = (V, E)$ be the hypergraph associated with the Min-3-AC instance $\mathcal{M}$. The greedy algorithm for Maximum-Coverage achieves an approximation ratio of $(1 - \frac{1}{e})$ [6]. In particular, if $\mathrm{opt}_{\textsf{Min-3-AC}} = k + \ell$, then all k elements of $\mathcal{M}$ can be covered by ℓ pairs by Lemma 4.1, and so the greedy algorithm covers at least $(1 - 1/e)k$ monomials. To cover the remaining $(1/e)k$ monomials, the greedy algorithm will clearly need to select at most $(1/e)k$ additional nodes. Thus, size$(\mathcal{C}) \le k + \ell + \frac{1}{e}k$. However, this bound is worse than size$(\mathcal{C}) \le \frac{4}{3}k + \ell$. But let us take the analysis one step further, for it may happen that $\frac{1}{e}k$ is still quite large compared to ℓ. Let k_1 denote the number of monomials covered by the first ℓ nodes selected by the greedy algorithm. By the preceding argument, $k_1 \ge (1 - 1/e)k$. The remaining $k - k_1$ can easily be covered by ℓ nodes again since this is even possible for the entire set of monomials. Consequently, the greedy algorithm covers at least $(1 - 1/e)(k - k_1)$ out of these monomials, and we remain with at most $k - k_1 - (1 - 1/e)(k - k_1) = \frac{1}{e}(k - k_1) \le \frac{1}{e^2}k$ uncovered monomials. Again, this number is an upper bound on the number of nodes picked by the greedy algorithm after having chosen 2ℓ nodes. In total, we obtain the desired bound: size$(\mathcal{C}) \le \left(1 + \frac{1}{e^2}\right) k + 2\ell$. □

It does not make much sense to reiterate the last step of the analysis since this would give us a circuit of size larger than $k + 3\ell$, the size achieved by Cover.

Corollary 4.9. *The approximation ratio achieved by* Greedy *for* Min-3-AC *is at most* $\frac{5e^2-3}{4e^2-3} \approx 1.278$.

Proof. Let $\mathcal{M}$ be a Min-3-AC instance with $\mathrm{opt}_{\textsf{Min-3-AC}} = k + \ell$, $k = |\mathcal{M}|$. By Lemma 4.8, the approximation ratio of Greedy is at most

$$\min\{\rho_1(\ell), \rho_2(\ell)\} \;, \tag{3}$$

where $\rho_1(\ell) = \frac{\frac{4}{3}k+\ell}{k+\ell}$ and $\rho_2(\ell) = \frac{(1+\frac{1}{e^2})k+2\ell}{k+\ell}$. We have

$$\rho_1(\ell) \ge \rho_2(\ell) \Leftrightarrow \frac{4}{3}k+\ell \le \left(1 + \frac{1}{e^2}\right) k+2\ell \Leftrightarrow \ell \ge \left(\frac{1}{3} - \frac{1}{e^2}\right) k = \frac{e^2-3}{3e^2} \approx 0.1980 \cdot k \;.$$

Since ρ_1 is monotone increasing and ρ_2 is monotone decreasing in ℓ, the minimum in (3) is attained for $\ell = (e^2 - 3)k/(3e^2)$. It is

$$\rho_1\left(\frac{e^2-3}{3e^2}k\right) = \frac{\frac{4}{3} + \frac{e^2-3}{3e^2}}{1 + \frac{e^2-3}{3e^2}} = \frac{5e^2-3}{4e^2-3} \approx 1.278 \;.$$

□

The best lower bound that we are able to show for the approximation ratio of Greedy is $10/9 = 1.111\ldots$. It is obtained by the reduction from vertex cover presented in Lemma 3.1. The corresponding vertex cover instance is shown in Figure 7.

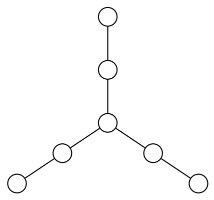
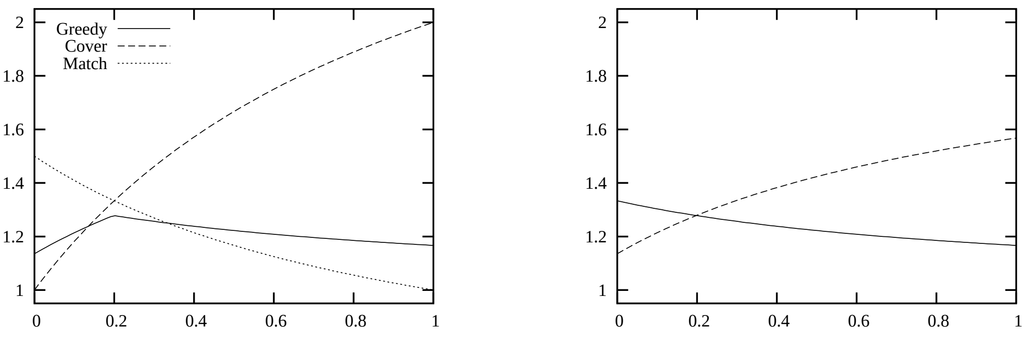


Figure 7: Graph with vertex cover of size 3, but for which the greedy algorithm outputs a cover of size 4.



(a) Upper bounds for Greedy, Cover, and Match.

(b) Upper bounds for Greedy given by Lemma 4.8.

Figure 8: Upper bounds on the approximation ratios of the Min-3-AC algorithms dependent on the ratio ℓ/k.

4.4 Summary of Approximation Ratios

In this subsection, we summarize the approximation ratios of the algorithms presented in the preceding subsections and present some improvements for Min-3-AC instances with bounded multiplicity. So far, we have found the following bounds for the approximation ratios of the Min-3-AC algorithms:

$$\begin{array}{lll}
\rho_{\text{Cover}} & \le \frac{k+3\ell}{k+\ell} & \text{increasing in } \ell\ , \\
\rho_{\text{Greedy}} & \le \frac{(1+e^{-2})k+2\ell}{k+\ell} & \text{increasing in } \ell\ , \\
\rho_{\text{Greedy}} & \le \frac{\frac{4}{3}k+\ell}{k+\ell} & \text{decreasing in } \ell\ , \\
\rho_{\text{Match}} & \le \frac{\frac{3}{2}k+\frac{1}{2}\ell}{k+\ell} & \text{decreasing in } \ell\ .
\end{array}$$

These approximation ratios are presented in Figure 8. Concerning restricted multiplicity, we prove the following result.

Theorem 4.10. *The* Min-3-AC *problem restricted to instances of maximum multiplicity μ, $\mu \in \{3,4,5\}$, is approximable with a factor of*

(a) $5/4 = 1.25$ *if* $\mu = 3$,

(b) $19/15 = 1.2\overline{6}$ *if* $\mu = 4$, *and*

(c) $23/18 = 1.2\overline{7}$ *if* $\mu = 5$.

Proof. Let $\mathcal{M}$ be a Min-3-AC instance with $\mathrm{mult}(\mathcal{M}) = \mu$ and $\mathcal{C}$ be a circuit for $\mathcal{M}$. Then each of the ℓ gates at the first layer of $\mathcal{C}$ can be used for at most μ monomials in $\mathcal{M}$. Consequently, $\ell \geq k/\mu$. For $\mu = 3$, Match has approximation ratio at most $\max_{\ell \geq \frac{1}{3}k} \frac{\frac{3}{2}k+\frac{1}{2}\ell}{k+\ell} = \frac{5}{4}$. For $\mu = 4$ and $\mu = 5$, Greedy achieves ratio at most $\max_{\ell \geq \frac{1}{\mu}k} \frac{\frac{4}{3}k+\ell}{k+\ell}$, which evaluates to 19/15 and 23/18 for $\ell = k/4$ and $\ell = k/5$, respectively. $\square$

5 Fixing the Number of Monomials

In this section we show that Min-AC is fixed parameter tractable with respect to the number k of monomials in the input instance. For more details on fixed parameter tractability, we refer to Downey and Fellows [8].

Theorem 5.1. Min-AC*, parameterized by the number of input monomials, is fixed parameter tractable. This means that there are a function* $f : \mathbb{N} \to \mathbb{N}$ *and a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *such that* Min-AC *can be solved deterministically in time* $f(k)+p(N)$.

Proof. To prove the theorem, we show that the instances of Min-AC have problem kernels that can be computed in polynomial time and the size of which depends only on the number k of monomials.

Let $\mathcal{M} = \{M_1, \ldots, M_k\}$ be a Min-AC instance and $X = \bigcup_{i \in [k]} M_i = \{x_1, \ldots, x_n\}$. We describe a problem kernel of $\mathcal{M}$ of size $f(k)$, i.e., a Min-AC instance $\widetilde{\mathcal{M}}$ such that we can compute a minimum AND-circuit for $\mathcal{M}$ from a minimum AND-circuit for $\widetilde{\mathcal{M}}$ (i.e., we present a self-reduction of Min-AC such that an instance with k monomials is mapped to an instance with total size depending only on k).

Define the equivalence relation $\sim$ on X by $x_i \sim x_j$ if and only if for all $\mathcal{M} \in M$, $x_i \in M$ iff $x_j \in M$. Denote the equivalence classes of $\sim$ by $X_1, \ldots, X_r$, so that X is partitioned into $X_1 \cup \ldots \cup X_r$. For each $j \in [r]$, let $i_j \in [n]$ be the minimal variable index in X_j. We call the variables $x_{i_1}, \ldots, x_{i_r}$ *primary* variables. In contrast, all other variables are referred to as *secondary* variables. Thus, each primary variable is a unique representative of its equivalence class.

Now $\widetilde{\mathcal{M}}$ is obtained from $\mathcal{M}$ by identifying equivalent variables: the variables of $\widetilde{\mathcal{M}}$ are the equivalence classes $X_1, \ldots, X_r$, and the monomials are $\tilde{M}_1, \ldots, \tilde{M}_k$, where for a monomial $M = x_{j_1} \ldots x_{j_d}$, the *reduced* monomial $\tilde{M}$ is defined by

$$\tilde{M} = X_{\xi_1} \ldots X_{\xi_d}$$

with ξ_t chosen such that $x_{j_t} \in X_{\xi_t}$ for $t \in [d]$. Since there are only k monomials in $\mathcal{M}$, there can be at most 2^k different equivalence classes. Thus, the number of different monomials over the possible 2^k variables for $\widetilde{\mathcal{M}}$ is 2^{2^k}. Consequently, the size of $\widetilde{\mathcal{M}}$ can be bounded by a function that only depends on k.

Given a circuit $\tilde{\mathcal{C}}$ for $\widetilde{\mathcal{M}}$, we now describe how to construct a circuit $\mathcal{C}$ for $\mathcal{M}$ such that

$$\mathrm{size}(\mathcal{C}) = \mathrm{size}(\tilde{\mathcal{C}}) + \sum_{i=1}^{r}(|X_i| - 1) . \tag{4}$$

Reduce

```
1: Input circuit $\mathcal{C}$ for $\mathcal{M} = \{M_1, \dots, M_k\}$.
2: For each secondary variable $x_i$:
3:   For each successor gate $g$ of $g_{x_i}$ in $\mathcal{C}$:
4:     Let $g_{x_i}$ and $g'$ be the predecessors of $g$.
5:     For all successors $g''$ of $g$:
6:       Replace the wire $(g', g)$ with $(g', g'')$.
7:     Delete $g$ from $\mathcal{C}$.
8: Delete all connected components of $\mathcal{C}$ that do not contain any
   input gates.
9: Replace each input gate $g_{x_i}$ of $\mathcal{C}$ with the input gate $g_{X_i}$.
10: Output $\tilde{\mathcal{C}} = \mathcal{C}$.
```

Figure 9: Procedure Reduce turns a circuit for $\mathcal{M}$ into a circuit for $\widetilde{\mathcal{M}}$ satisfying (5), see proof of Theorem 5.1.

For each equivalence class X_i, we build a trivial circuit $\mathcal{C}_i$ of size $|X_i| - 1$ that computes the monomial that consists of all variables in X_i. Subsequently, we merge the circuits $\tilde{\mathcal{C}}$ and $\mathcal{C}_1, \dots, \mathcal{C}_r$ by replacing each input gate g_{X_i} of $\tilde{\mathcal{C}}$ with the output gate of $\mathcal{C}_i$. The resulting circuit $\mathcal{C}$ clearly computes $\mathcal{M}$ and satisfies (4).

What remains to be proved is that if $\tilde{\mathcal{C}}$ is of minimum size for $\tilde{\mathcal{C}}$, then $\mathcal{C}$ is of minimum size for $\mathcal{M}$. We do this by showing how to turn an arbitrary circuit $\mathcal{C}$ without useless gates for $\mathcal{M}$ into a circuit $\tilde{\mathcal{C}}$ for $\widetilde{\mathcal{M}}$ with

$$\text{size}(\tilde{\mathcal{C}}) \leq \text{size}(\mathcal{C}) - \sum_{i=1}^{r}(|X_i| - 1) \ . \tag{5}$$

The corresponding Procedure Reduce is presented in Figure 9.

Let $T = \{x_{i_1}, \dots, x_{i_d}\}$ be the transversal of primary variables and let $g_1, \dots, g_{|\mathcal{C}|}$ be the original computation gates of $\mathcal{C}$.

Claim 5.2. *After step 8 of* Reduce, *$\mathcal{C}$ computes exactly the monomials* $\text{val}(g_i) \cap T$, $i \in [|\mathcal{C}|]$.

Proof of Claim 5.2. If a gate g is deleted from $\mathcal{C}$, then its predecessors are g_{x_i} and g' for some secondary variable x_i and some other gate g'. Thus, using set notation, $\text{val}(g') = \text{val}(g) \setminus \{x_i\}$. Consequently, for each gate g of $\mathcal{C}$, after each iteration of the `for`-loop in steps 2–7 processing the secondary variable x_i, the monomial $\text{val}(g) \setminus \{x_i\}$ is computed by some gate (either by g itself or by one of its predecessors). Since every secondary variable is eventually processed, the claim follows. □

In step 9 of Reduce, for each $i \in [k]$, the value of the gate that computes $M_i \cap T$ changes to $\tilde{M}_i$. Thus, $\tilde{\mathcal{C}}$ computes $\mathcal{C}$. Finally, we show that for every secondary variable x_i, at least one gate is deleted and hence (5) is satisfied. Since by assumption, x_i appears in some monomial M_j, the input gate g_{x_i} has at least one successor in $\mathcal{C}$ when starting Procedure Reduce. Moreover, before variable x_i is processed in the `for`-loop in steps 2–7 of REDUCE, g_{x_i} keeps at least one successor. This is because if a successor of g_{x_i} is deleted in step 7 while processing another variable $x_{i'}$, then the wire (g_{x_i}, g) is redirected to (g_{x_i}, g'') in step 6. If no successor g'' of g exists, then g computes $x_i x_{i'}$ without being used for any input monomial M_j and is thus useless, contradicting the assumption that $\mathcal{C}$ does not contain any useless gates. Consequently, for each secondary variable, at least one gate is deleted and $\tilde{\mathcal{C}}$ satisfies (5). □

6 Concluding Remarks and Future Research

6.1 Approximation Algorithms for Min-d-AC, $d \geq 4$

Obviously, the approximation ratio of Min-d-AC is at most $d - 1$ since on the one hand, every monomial of degree at most d can be computed by at most $d - 1$ separate gates and on the other hand, any circuit contains at least one gate per monomial of the input instance. It is easy to see that MATCH achieves the slightly better approximation ratio $d - \frac{3}{2}$ (which is tight); the proof is almost identical to the proofs of Lemmas 4.3 and 4.4. Unfortunately, neither do we see how to generalize algorithm COVER to $d \geq 4$, nor is it clear how to analyze greedy algorithms in that case. The problem is that once one has decided to substitute all occurrences of a pair of variables in all monomials, it may happen that an optimal circuit for the remaining monomials is *strictly larger* than an optimal circuit for the original instance. This makes it difficult to apply standard techniques as in the proof of the classical $1 + \ln n$ approximation bound for the greedy set cover algorithm [10].

We are particularly curious about whether Min-d-AC is approximable within a factor of $o(d)$ or whether it is possible to show an $\Omega(d)$ hardness result.

For $d \geq 4$, there are several possibilities of generalizing the greedy algorithm, some of which are presented in the following.

- GREEDY PAIRING: Select a most frequent pair, build a gate for it, and substitute the pair by a new variable wherever possible. Repeat until all monomials have size one.

- GREEDY SAVING: Select a monomial to be computed by a gate such that its usage "saves" as many gates as possible compared to a trivial completion of the circuit. Substitute the monomial by a new variable wherever possible and add the monomial to the input instance. Repeat until all monomials are computed by the circuit.

- GREEDY CUTTING: Select a longest submonomial appearing in multiple places and build a gate for it. Substitute the monomial by a new variable wherever possible and add the monomial to the input instance. Repeat until all monomials have size one.

For $d = 3$, all three variants coincide.

The algorithms GREEDY PAIRING, GREEDY CUTTING, and MATCH produce strict circuits. Already for $d = 5$, we can construct Min-AC instances $\mathcal{M}$ of maximum degree d such that any strict circuit for $\mathcal{M}$ is roughly 4/3 times larger than a minimum non-strict circuit:

Lemma 6.1. *There are* Min-5-AC *instances $\mathcal{M}$ such that every circuit $\mathcal{C}$ for $\mathcal{M}$ of minimum size is non-strict. Moreover, for arbitrarily small $\epsilon > 0$, there are instances $\mathcal{M}$ such that the ratio between a minimum strict circuit for $\mathcal{M}$ and a minimum non-strict circuit is $4/3 - \epsilon$.*

Proof. Let $\mathcal{M} = \{xy, yz\} \cup \{xya_i, yzb_i, xyza_ib_i \mid i \in [t]\}$, $t \geq 1$. It is easy to construct a minimum AND-circuit $\mathcal{C}$ for $\mathcal{M}$ such that every computation gate of $\mathcal{C}$ is also an output gate, i.e., $\operatorname{size}(\mathcal{C}) = |\mathcal{M}| = 3t + 2$. On the other hand, it is impossible to *strictly* build the monomial $xyza_ib_i$ from other monomials of $\mathcal{M}$. Thus, in a strict circuit $\mathcal{C}'$ for $\mathcal{M}$, we must include an additional non-output gate, say to compute the monomial a_ib_i for each $i \in [t]$. Consequently, $\operatorname{size}(\mathcal{C}') = 4t + 2$, and hence

$$\operatorname{size}(\mathcal{C}')/\operatorname{size}(\mathcal{C}) = (4t+2)/(3t+2) \xrightarrow[t\to\infty]{} 4/3 .$$

□

Corollary 6.2. *Any approximation algorithm for* Min-AC *(or even* Min-5-AC*) that produces only strict circuits does not achieve an approximation ratio better than 4/3.*

6.2 Approximation of Instances with Bounded Multiplicity

In Section 4.2, we showed that Min-3-AC instances with maximum multiplicity two are optimally solvable in polynomial time. In contrast, Min-3-AC instances with maximum multiplicity three are hard to solve, as we saw in Section 3. We leave it as an open problem whether Min-d-AC instances with $d \geq 4$ are polynomial time solvable. Nonetheless we can provide a positive approximability result for general Min-AC restricted to instances with bounded multiplicity:

Theorem 6.3. *The* Min-AC *problem with instances restricted to be of maximum multiplicity μ is polynomial-time approximable within a factor of μ.*

Proof. Let $\mathcal{M}$ be a Min-AC instance with $\operatorname{mult}(\mathcal{M}) = \mu$ and let $\mathcal{C}$ be a circuit for $\mathcal{M}$ of minimum size. As $\operatorname{size}(\mathcal{C}_M) \geq |M| - 1$ for every $M \in \mathcal{M}$, equation (1) yields

$$\sum_{g \in G^*(\mathcal{C})} \operatorname{freq}_{\mathcal{M}}(g) = \sum_{M \in \mathcal{M}} \operatorname{size}(\mathcal{C}_M) \geq \sum_{M \in \mathcal{M}} (|M| - 1) .$$

By equation (2), $\sum_{g \in G^*(\mathcal{C})} \operatorname{freq}_{\mathcal{M}}(g) \leq \mu \cdot \operatorname{size}(\mathcal{C})$. We denote by $\mathcal{C}_{\mathrm{triv}}$ the trivial circuit of size $\sum_{M \in \mathcal{M}}(|M| - 1)$ in which every monomial is computed by a separate subcircuit. Then, by the preceding arguments, we have $\operatorname{size}(\mathcal{C}) \geq \operatorname{size}(\mathcal{C}_{\mathrm{triv}})/\mu$ and thus $\frac{\operatorname{size}(\mathcal{C}_{\mathrm{triv}})}{\operatorname{opt}_{\mathsf{Min\text{-}AC}}(\mathcal{M})} = \frac{\operatorname{size}(\mathcal{C}_{\mathrm{triv}})}{\operatorname{size}(\mathcal{C})} \leq \mu$, which means that $\mathcal{C}_{\mathrm{triv}}$ is a μ-approximation for $\mathcal{M}$. □

Theorem 6.3 also follows from a more general result by Wegener [17, Section 6.6] about *Boolean sums*, which are collections of disjunctions of (positive) Boolean variables, and thus are dual to collections of monomials. Wegener [17, Definition

6.1] defines such a collection to be (h,k)-disjoint if $h+1$ disjunctions have at most k common summands. In particular, sets of monomials of multiplicity μ correspond to $(\mu,1)$-disjoint collections. The claim then follows from [17, Lemma 6.1] by plugging in $h=\mu$ and $k=1$. Although the lemma is only stated for collections in which the number of input variables equals the number of disjunctions, it also holds if these numbers differ.

We can improve the result of Theorem 6.3 for Min-Ed-AC restricted to instances with bounded multiplicity using the fact that for these instances, all output gates have frequency one.

Theorem 6.4. *The* Min-Ed-AC *problem with instances restricted to be of maximum multiplicity μ is polynomial-time approximable within a factor of* $\frac{\mu(d-1)}{\mu+d-2}$.

Proof. Let $\mathcal{M}$ be an Min-Ed-AC instance with k monomials of maximum multiplicity μ and $\mathcal{C}$ be a circuit for $\mathcal{M}$ of minimum size. Since $|M|=d$ for all $M\in\mathcal{M}$, $\text{size}(\mathcal{C}_{\mathcal{M}})\geq d-1$ (with equality guaranteed if $\mathcal{C}$ is strict). Furthermore, all output gates have frequency 1, and all other computation gates have a frequency of at most μ. Putting these things together and using (1), we obtain

$$(d-1)k \;\leq\; \sum_{M\in\mathcal{M}} \text{size}(\mathcal{C}_{\mathcal{M}}) \;=\; \sum_{g\in G^*(\mathcal{C})} \text{freq}_{\mathcal{M}}(g) \;\leq\; \mu(\text{size}(\mathcal{C})-k)+k\ .$$

Consequently, $\text{size}(\mathcal{C})\geq\frac{d-2+\mu}{\mu}\cdot k$. Let $\mathcal{C}_{\text{triv}}$ be the trivial circuit for $\mathcal{M}$ of size $(d-1)k$. Then

$$\frac{\text{size}(\mathcal{C}_{\text{triv}})}{\text{opt}_{\textsf{Min-AC}}(\mathcal{M})}) = \frac{\text{size}(\mathcal{C}_{\text{triv}})}{\text{size}(\mathcal{C})} \leq \frac{\mu(d-1)}{d-2+\mu}\ .$$

□

Corollary 6.5. Min-E4-AC *with maximum multiplicity two is polynomial-time approximable within a factor of* 3/2.

The approximation ratio of 3/2 is much lower than the ratio of 5/2 achieved by Match for general Min-4-AC instances.

6.3 Generalizations and Related Problems

Let us first mention some applications that arise as alternative interpretations of the problem in this paper. Viewing monomials M over X as subsets of X (see also Section 2), an AND-gate computes the *union* of the sets computed by its predecessors. Thus, AND-circuits may be interpreted as compact representations of set systems. Since each gate has to be evaluated only once, the circuit may be considered as a straight-line program that generates the set system. Furthermore, in a Boolean matrix-vector product, each entry of the result is a disjunction (or a parity, depending on which type of "sum" is considered) of the vector entries corresponding to the positions of 1s in the matrix rows. Thus, if many vectors have to be multiplied by the same matrix, it may be useful to preprocess the matrix by constructing a circuit that computes all disjunctions (with indeterminates) first.

S	$\circ$	k	n	Description	Remark
$\{0,1\}$	$\wedge$	arb.	arb.	Boolean monomials, Min-AC	
$\mathbb{Z}$	$+$	1	1	Addition chains [13, 16]	complexity unknown
$\mathbb{Z}$	$+$	arb.	1	Extended addition chains	NP-complete [7]
Σ^*	concat.	1	arb.	Grammar-based compression [12] of strings over alphabets of size n	NP-complete for $n \geq 3$ [14], complexity unknown for $n \leq 2$

Table 1: The circuit problem for several semigroup structures and parameters.

Beside Boolean variables and monomials, it is natural to consider monomials over other structures. In general, the variables $x \in X$ take values from some semigroup $(S, \circ)$ (note that we assume the structure to be associative since otherwise it makes no sense to design small circuits). In case that S is non-commutative, the predecessors of a gate have to be ordered. Table 1 shows several examples of semigroups and other parameters with their corresponding circuit problem. As one can see, many seemingly different problems turn out to be instantiations of a general *semigroup circuit problem.*

The greedy algorithms proposed in Section 6.1 are closely related to the so-called *global algorithms* Re-Pair, Greedy, and Longest Match for the smallest grammar problem [4], which deals with the compression of a given string by a context-free grammar that generates exactly that string. Global algorithms are of particular interest for this problem since they are believed to have low approximation ratios. However, despite their simplicity, only very weak upper bounds are known. We hope that techniques for proving upper and lower bounds for global algorithms may be transferred between the smallest grammar problem and the minimum AND-circuit problem.

6.4 Some More Open Problems

For the approximation ratio of Min-3-AC, we believe that a more concise analysis of Greedy or similar algorithms may yield an upper bound below 5/4.

Since we still lack good approximation algorithms for any $d \geq 4$, it would already be interesting to have approximation algorithms with ratio less than 2.5 for Min-4-AC, which may be achieved by an algorithm that is similar to Cover, tailored to the case $d = 4$.

Finally, as we have determined the complexity of the decision problem associated with Min-d-AC with multiplicity bounded by μ for several choices of d and μ, it would be nice to complete these results by studying the case $d \geq 4$ and $\mu = 2$.

Acknowledgments

We are grateful to Ingo Wegener for pointing to known facts about Boolean sums [17, Sect. 6.6], including a proof of Theorem 6.3. We also thank Jehoshua Bruck, Rüdiger Reischuk, Thomas Zeugmann, and an anonymous referee for helpful suggestions.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, 1993.

[2] Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1–2):123–134, 2000.

[3] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer, 1999.

[4] Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abbi Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005.

[5] Miroslav Chlebík and Janka Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theoretical Computer Science*, 354(3):320–338, 2006.

[6] Gérard P. Cornuéjols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23:789–810, 1977.

[7] Peter J. Downey, Benton L. Leong, and Ravi Sethi. Computing sequences with addition chains. *SIAM Journal on Computing*, 10(3):638–646, 1981.

[8] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999.

[9] Michael R. Garey, David S. Johnson, and Larry K. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[10] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[11] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problems. In *Proc. of the 32nd Ann. ACM Symp. on Theory of Computing (STOC)*, pages 73–79. ACM Press, 2000.

[12] John C. Kieffer and En-hui Yang. Grammar based codes: A new class of universal lossless source codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000.

[13] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming.* Addison-Wesley, 2nd edition, 1981.

[14] James A. Storer and Thomas G. Szymanski. The macro model for data compression. In *Proc. of the 10th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 30–39. ACM Press, 1978.

[15] Robert E. Tarjan. Complexity of monotone networks for computing conjunctions. *Annals of Discrete Mathematics*, 2:121–133, 1978.

[16] Edward G. Thurber. Efficient generation of minimal length addition chains. *SIAM Journal on Computing*, 28(4):1247–1263, 1999.

[17] Ingo Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/