Approximability of Minimum AND-Circuits*

Jan Arpe[1],** and 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. [3, 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. [3]. 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 within a factor of less than $\frac{983}{978}$ unless $\mathsf{P} = \mathsf{NP}$, even if the monomials are

* A full version of this work with all proofs is available as Report 06-045 of the Electronic Colloquium on Computational Complexity (ECCC).
** 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.

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 [6]). 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 [7]; 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 [11] (see also Wegener [13]).

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* [9, 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. [3, 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

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.* 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 $\mathcal{C}_{g_M}$.

A gate is called *strict* if its predecessors compute disjoint monomials. A circuit is called *strict* if all of its gates are strict. 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. This is not true if the monomials are allowed to be of degree five or more (Sect. 6.1).

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.,

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

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 $\mathcal{M}$.

2.2 Optimization Problems

For an introduction to the approximation theory of combinatorial optimization problems, we refer to Ausiello et al. [2]. 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.

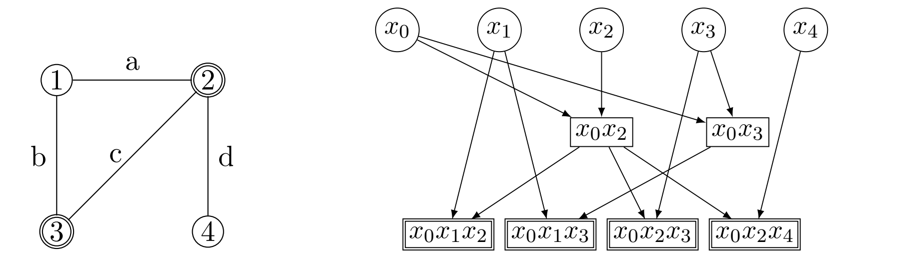
A *vertex cover* of a graph G is a subset $\tilde{V} \subseteq V$ such that every edge has at least one endpoint in $\tilde{V}$. This definition also applies to hypergraphs. Aside from Min-AC, we will encounter the following optimization problems: The vertex cover problem, denoted by Min-VC, is defined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

3 Hardness

We show that Minimum-AND-Circuit 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.



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

(b) Circuit for the Min-3-AC instance $\{M_a, M_b, M_c, M_d\}$ with $M_a = x_0x_1x_2$, $M_b = x_0x_1x_3$, $M_c = x_0x_2x_3$, and $M_d = x_0x_2x_4$.

Fig. 1. A graph with a vertex cover and the corresponding circuit as constructed in Section 3.

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.

There is a one-to-one correspondence between the sizes of the vertex cover and the circuit: We have $\mathrm{opt}_{\text{Min-AC}}(\mathcal{M}_G) = |E| + \ell$, where $\ell = \mathrm{opt}_{\text{Min-VC}}(G)$. 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 $|\tilde{V}| \le \ell'$ in polynomial time. This together with recent inapproximability results by Chlebík and Chlebíková [4] yields the following theorems.

Theorem 1. *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.*

Theorem 2. *Min-3-AC restricted to instances of maximum multiplicity three is NP-complete, APX-complete, and cannot be approximated in polynomial time within a factor of less than $\frac{269}{268} > 1.0037$ unless P = NP.*

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 circuits for Min-3-AC instances are strict. Moreover, if all monomials are of degree exactly 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 $\{S_i \mid 1 \le i \le \ell\}$ and vice versa. We denote the circuit corresponding to a vertex cover $\tilde{V}$ by $\mathcal{C}_{\tilde{V}}$.

Our first polynomial-time approximation algorithm for Min-3-AC is based on the reduction we have just presented (Algorithm 1). 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

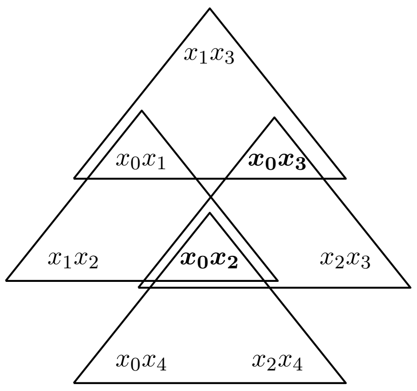


Fig. 2. The hypergraph $H(\mathcal{M})$ associated with the Min-AC instance $\mathcal{M}$ introduced in Figure 1. Each triangle represents a hyperedge. The two bold monomials constitute a vertex cover.

Algorithm 1 Algorithm Cover for Min-3-AC.

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

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 $|\tilde{V}| \leq 3 \cdot \mathrm{opt}_{\mathsf{Min\text{-}3\text{-}UVC}}(H(\mathcal{M}))$.

Overall, Cover achieves the following approximation performance.

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

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 approximation ratio of 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 3.

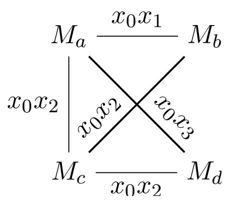


Fig. 3. Intersection graph $IG(\mathcal{M})$ associated with the Min-AC instance $\mathcal{M}$ introduced in Figure 1. The edges are labeled by the pairs that their endpoints have in common. The bold edges constitute a maximal matching.

Algorithm 2 Algorithm Match for Min-3-AC.

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

Match (Algorithm 2) is a polynomial-time algorithm; in particular, a maximum matching in $IG(\mathcal{M})$ can be computed in time $O(n^{2.5})$ [1]. The approximation performance of Match is stated in the following

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

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 $d \geq 4$.

For Min-3-AC with instances restricted to a multiplicity of at most two, Match computes an optimum solution. Thus, Min-3-AC restricted to instances with a maximum multiplicity of at most two can be solved in polynomial time.

4.3 Algorithm "Greedy"

Our last algorithm Greedy (Algorithm 3) 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 stated in Section 4.2.

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

Algorithm 3 Algorithm Greedy for Min-3-AC.

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

at most

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

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

The best lower bound that we are able to show for the approximation ratio of Greedy is 10/9.

4.4 Summary of Approximation Ratios

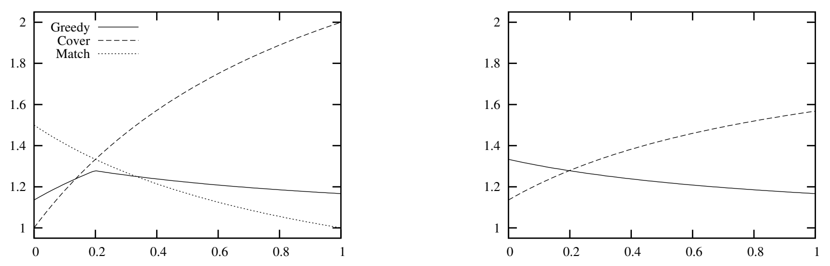
In this subsection, we summarize the approximation ratios of the algorithms presented in the preceding subsections and present some improvements for Min-3-AC instances with bounded multiplicity. So far, we have found the following bounds for the approximation ratios of the Min-3-AC algorithms:

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

These approximation ratios are presented in Figure 4. Concerning restricted multiplicity, we can show the following result.

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

- $5/4 = 1.25$ *if* $\mu = 3$,
- $19/15 = 1.2\overline{6}$ *if* $\mu = 4$, *and*
- $23/18 = 1.2\overline{7}$ *if* $\mu = 5$.



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

(b) Upper bounds for Greedy given by Lemma 3.

Fig. 4. Approximation ratios of the Min-3-AC algorithms dependent on the ratio ℓ/k.

5 Fixing the Number of Monomials

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 [6].

Theorem 4. 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* $f(k) + p(N)$.

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).

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, which coincide for $d = 3$.

The algorithms Greedy 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.

Corollary 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 with instances restricted to maximum multiplicity three is 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 instances with bounded multiplicity.

Theorem 5. Min-AC *with instances restricted to be of maximum multiplicity μ is polynomial-time approximable within a factor of μ.*

Theorem 5 also follows from a more general result by Wegener [13, Sect. 6.6] about *Boolean sums*, which are collections of disjunctions of (positive) Boolean variables, and thus are dual to collections of monomials. Wegener [13, Def. 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 [13, Lem. 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 5 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. *The* Min-Ed-AC *problem with instances restricted to be of maximum multiplicity μ is polynomial-time approximable within a factor of $\frac{\mu(d-1)}{\mu+d-2}$.*

This implies an improved approximation ratio of 3/2 compared to 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 Sect. 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.

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

S	$\circ$	k	n	Description	Remark
$\{0,1\}$	$\wedge$	arb.	arb.	Boolean monomials, Min-AC	
$\mathbb{Z}$	$+$	1	1	Addition chains [9,12]	complexity unknown
$\mathbb{Z}$	$+$	arb.	1	Extended addition chains	NP-complete [5]
Σ^*	concat.	1	arb.	Grammar-based compression [8] of strings over alphabets of size n	NP-complete for $n \geq 3$ [10], complexity unknown for $n \leq 2$

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

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.*

References

1. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, 1993.
2. 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.
3. 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.
4. Miroslav Chlebík and Janka Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theoretical Computer Science*, 354(3):320–338, 2006.
5. Peter J. Downey, Benton L. Leong, and Ravi Sethi. Computing sequences with addition chains. *SIAM Journal on Computing*, 10(3):638–646, 1981.
6. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999.
7. 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.
8. 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.
9. Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming.* Addison-Wesley, 2nd edition, 1981.
10. 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.
11. Robert E. Tarjan. Complexity of monotone networks for computing conjunctions. *Annals of Discrete Mathematics*, 2:121–133, 1978.
12. Edward G. Thurber. Efficient generation of minimal length addition chains. *SIAM Journal on Computing*, 28(4):1247–1263, 1999.
13. Ingo Wegener. *The Complexity of Boolean Functions.* Wiley-Teubner, 1987.