MATHEMATICS OF COMPUTATION
Volume 86, Number 307, September 2017, Pages 2429–2447
http://dx.doi.org/10.1090/mcom/3160
Article electronically published on December 27, 2016

ON DYNAMIC ALGORITHMS FOR FACTORIZATION INVARIANTS IN NUMERICAL MONOIDS

THOMAS BARRON, CHRISTOPHER O'NEILL, AND ROBERTO PELAYO

Abstract. Studying the factorization theory of numerical monoids relies on understanding several important factorization invariants, including length sets, delta sets, and ω-primality. While progress in this field has been accelerated by the use of computer algebra systems, many existing algorithms are computationally infeasible for numerical monoids with several irreducible elements. In this paper, we present dynamic algorithms for the factorization set, length set, delta set, and ω-primality in numerical monoids and demonstrate that these algorithms give significant improvements in runtime and memory usage. In describing our dynamic approach to computing ω-primality, we extend the usual definition of this invariant to the quotient group of the monoid and show that several useful results naturally extend to this broader setting.

1. Introduction

Numerical monoids (co-finite, additive submonoids of $\mathbb{N}$) and their non-unique factorization theory have emerged as an active area of study in recent years [1, 2, 4, 7, 15, 16]. The primary objects in these investigations are factorization invariants, which measure in different ways the plural factorizations of an element n into irreducibles in the numerical monoid S. The set $\mathsf{Z}(n)$ of factorizations of a single element $n \in S$ can uniquely determine the monoid structure of S. As such, $\mathsf{Z}(n)$ is often too cumbersome to compute for large values of n in numerical monoids with several irreducible elements. Most of the major factorization invariants of interest, including the length set $\mathsf{L}(n)$, delta set $\Delta(n)$, ω-primality $\omega(n)$, and catenary degree $\mathsf{c}(n)$, are typically only computed after first computing $\mathsf{Z}(n)$.

Discovering many of the results regarding these factorization invariants has relied on effectively utilizing computer algebra packages, such as the GAP package `numericalsgps` [8]. Of particular interest is finding closed forms for and analyzing the asymptotic behavior of these invariants, which frequently requires computing the invariant for numerous elements of S. As each of these individual computations generally requires first computing $\mathsf{Z}(n)$ and then passing to the invariant of interest, finding patterns for numerical monoids with large numbers of irreducible elements is often computationally infeasible.

Recent investigations in numerical monoids show that their factorization invariants tend to have eventual quasi-polynomial behavior. For example, ω-primality [15] and maximal and minimal elements of length sets [2] have quasi-linear behavior, while delta sets [7] and catenary degree [5] have periodic (i.e., quasi-constant) behavior. The inductive nature of these results motivates the major results for

Received by the editor July 28, 2015 and, in revised form, February 20, 2016 and March 15, 2016.

2010 *Mathematics Subject Classification.* Primary 05C70, 11Y11.

this paper: several factorization invariants, including $\mathsf{Z}(n), \mathsf{L}(n), \Delta(n)$, and $\omega(n)$, can be computed dynamically. Moreover, the dynamic algorithms we present for $\mathsf{L}(n), \Delta(n)$, and $\omega(n)$ (Algorithms 3.7, 3.9, and 5.6, respectively) do not require computing $\mathsf{Z}(n)$, which significantly improves runtimes and memory usage. Additionally, as closed forms and asymptotic behavior is of interest, a dynamic algorithm, which computes several values quickly in succession, is a fruitful approach.

In Section 2, we provide all necessary definitions, including those for numerical monoids (Definition 2.3) and their length set and delta set invariants (Definition 2.6), and state a recent periodicity result of interest. In Section 3, we give a dynamic algorithm to compute the delta set of a numerical monoid (Algorithm 3.9), and compare this algorithm to existing algorithms [10]. In Section 5, we give a dynamic algorithm to compute ω-primality in numerical monoids (Algorithm 5.6), after developing the necessary theory in Section 4. In Section 6, we prove Theorem 6.2, which relates factorizations and bullets (Definition 4.6), and derive an improved bound on the start of quasi-linear behavior of the ω-primality function (Theorem 6.12). Finally, Section 7 states several open questions.

At the time of writing, Algorithms 3.9, 3.3, and 5.6 are currently implemented in the GAP package `numericalsgps` [8]. All included benchmarks use this software, including comparisons to existing algorithms.

2. Background

We begin by defining numerical monoids (Definition 2.3) and introducing three of the main factorization invariants of interest: factorization sets, length sets, and delta sets (Definitions 2.5 and 2.6). In this section, and in the majority of this paper, monoids are written additively, and $\mathbb{N}$ denotes the set of non-negative integers.

Definition 2.1. A commutative monoid M (written additively) is *cancellative* if for any $a, b, c \in M$, $a + b = a + c$ implies that $b = c$. An element $u \in M$ is said to be *irreducible* (or an *atom*) if whenever $u = a + b$ for $a, b \in M$, then either a or b is a unit in M. The monoid M is said to be *atomic* if for every $m \in M$, there exist irreducible elements $u_1, u_2, \ldots, u_r \in M$ such that $m = u_1 + u_2 + \cdots + u_r$.

Remark 2.2. Unless otherwise stated, all monoids in this paper are assumed to be cancellative, commutative, and atomic.

This paper will focus on the factorization theory of additive submonoids of $\mathbb{N}$.

Definition 2.3. For coprime positive integers $n_1, n_2, \ldots, n_k \in \mathbb{N}$, the *numerical monoid S generated by* $\{n_1, n_2, \ldots, n_k\}$ is the additive submonoid of $\mathbb{N}$ given by

$$S = \langle n_1, n_2, \ldots, n_k \rangle = \{c_1 n_1 + c_2 n_2 + \cdots + c_k n_k : c_i \in \mathbb{N}\} \subset \mathbb{N}.$$

The *Frobenius number* $F(S) = \max(\mathbb{N} \setminus S)$ is the largest integer lying outside of S, and an element $n \in \mathbb{Z} \setminus S$ is a *pseudo-Frobenius number* if $n + n_i \in S$ for all n_i.

Remark 2.4. Every co-finite, additive submonoid of $\mathbb{N}$ is a numerical monoid generated by a set of relatively prime generators $\{n_1, n_2, \ldots, n_k\}$. In fact, there exists a unique collection of generators that is minimal with respect to set-theoretic inclusion. For this generating set, the irreducible elements of the numerical monoid coincide precisely with the generators. Thus, every numerical monoid is cancellative, reduced, and finitely generated. Unless otherwise stated, when we write $S = \langle n_1, n_2, \ldots, n_k \rangle$, we assume that the n_i constitute a minimal generating set and that $n_1 < n_2 < \cdots < n_k$.

Of particular interest is the non-unique factorization theory of numerical monoids. We establish here notation for factorizations of numerical monoid elements.

Definition 2.5. Let $S = \langle n_1, n_2, \ldots, n_k \rangle$ be a numerical monoid. A *factorization* of $x \in S$ is a k-tuple $\mathbf{a} = (a_1, a_2, \ldots, a_k) \in \mathbb{N}^k$ such that $x = a_1 n_1 + a_2 n_2 + \cdots + a_k n_k$, and the *set of factorizations of* x *in* S is given by

$$\mathsf{Z}_S(x) = \{\mathbf{a} \in \mathbb{N}^k \,:\, a_1 n_1 + a_2 n_2 + \cdots + a_k n_k = x\}.$$

When the monoid is understood, this is often denoted $\mathsf{Z}(x)$. The *length* of a factorization $\mathbf{a}$ is given by $|\mathbf{a}| = \sum_{i=1}^k a_i$.

In a numerical monoid S, the set of factorizations $\mathsf{Z}(x)$ is usually cumbersome, especially for larger values of $x \in S$. Therefore, many factorization invariants are computed using only the lengths of factorizations. This leads us to the concepts of the length set and delta set.

Definition 2.6. Given a numerical monoid $S = \langle n_1, n_2, \ldots, n_k \rangle$ and $x \in S$, the *length set of* x is given by

$$\mathsf{L}(x) = \{|\mathbf{a}| \,:\, \mathbf{a} \in \mathsf{Z}(x)\}.$$

If we order the elements of $\mathsf{L}(x)$ in increasing order $\mathsf{L}(x) = \{l_1 < l_2 < \cdots < l_r\}$, the *delta set of* x is given by

$$\Delta(x) = \{l_i - l_{i-1} \,:\, 2 \leq i \leq r\}.$$

The *delta set* of S is given as the union of delta sets of all non-identity elements:

$$\Delta(S) = \bigcup_{x \in S \setminus \{0\}} \Delta(x).$$

Example 2.7. Consider the numerical monoid $M = \langle 6, 9, 20 \rangle$, which has Frobenius number $F(M) = 43$. The element $60 \in M$ has factorization set

$$\mathsf{Z}(60) = \{(0,0,3), (1,6,0), (4,4,0), (7,2,0), (10,0,0)\}.$$

This produces the length set $\mathsf{L}(60) = \{3, 7, 8, 9, 10\}$ and delta set $\Delta(60) = \{1, 4\}$.

While the delta set of a numerical monoid S is the union of delta sets of all of its (infinitely many) non-identity elements, one needs only to compute $\Delta(x)$ for finitely many $x \in S$. A weaker version of Theorem 2.8, which appeared in [7], states that delta sets in a numerical monoid are eventually periodic and that $\Delta(S)$ is obtained by taking the union of $\Delta(n)$ over every $n \leq 2kn_2n_k^2 + n_1n_k$. More recently, the authors of [10] give a greatly improved bound N_S (significantly smaller than the bound from [7]) for the start of this periodic behavior, and show that $\Delta(S)$ can be computed by taking the union of $\Delta(n)$ over every $n \leq N_S + n_k - 1$.

Theorem 2.8 ([10, Corollary 18])**.** *Fix a numerical monoid* $S = \langle n_1, \ldots, n_k \rangle$. *There exists an integer* N_S *such that* $n \geq N_S$ *implies* $\Delta(n) = \Delta(n + \mathrm{lcm}(n_1, n_k))$.

Example 2.9. Figure 1 plots the delta sets of elements in $M = \langle 6, 9, 20 \rangle$. The bound for the start of periodic behavior [10] is $N_S = 144$, and we can see from the plot that this behavior actually starts at 91. Additionally, the eventual period is 20, which by [10] is guaranteed to divide (but clearly need not equal) $\mathrm{lcm}(6, 20) = 60$. Using this, we can compute the delta set of M to be $\Delta(M) = \{1, 2, 3, 4\}$.

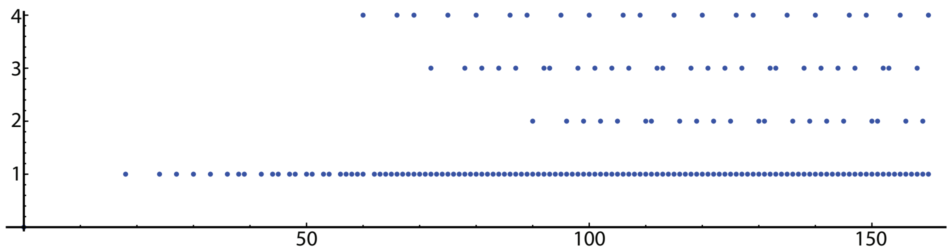

FIGURE 1. A plot showing the delta sets of elements in the numerical monoid $S = \langle 6, 9, 20\rangle$ from Example 4.5. Here, a dot is placed at the point (n, d) whenever $d \in \Delta(n)$.

3. Factorization sets and delta sets in numerical monoids

This section provides dynamic algorithms to compute factorization sets, length sets, and delta sets for numerical monoids (Algorithms 3.3, 3.7, and 3.9, respectively), along with corresponding proofs of correctness. Furthermore, we demonstrate the significant runtime advantages of these dynamic algorithms when compared to existing (non-dynamic) algorithms.

We begin with a lemma describing an inductive relationship between the factorization sets of elements. As this result applies to a significantly larger class of monoids than numerical monoids, it is stated in full generality. In what follows, $\mathbf{e}_i$ denotes the ith unit vector in $\mathbb{N}^k$.

Lemma 3.1. *Fix a reduced, finitely generated monoid M (written additively) with irreducible elements $m_1, m_2, \ldots, m_k$. For each non-zero $x \in M$, we have*

$$\begin{aligned} \mathsf{Z}(x) &= \textstyle\bigcup_{i=1}^{k} \{\mathbf{a} + \mathbf{e}_i : \mathbf{a} \in \mathsf{Z}(x - m_i)\} \\ &= \textstyle\bigsqcup_{i=1}^{k} \{\mathbf{a} + \mathbf{e}_i : \mathbf{a} \in \mathsf{Z}(x - m_i), a_j = 0 \text{ for each } j < i\}. \end{aligned}$$

Proof. Fix $\mathbf{a} \in \mathsf{Z}(x)$, and let $i = \min\{j : a_j \neq 0\}$. Then $\mathbf{a} - \mathbf{e}_i \in \mathsf{Z}(x - m_i)$. Moreover, if $\mathbf{a} - \mathbf{e}_j \in \mathsf{Z}(x - m_j)$ for some j, then $j \geq i$ by minimality of i. This proves the second equality, from which the first equality follows. $\square$

Example 3.2. Let $M = \langle 6, 9, 20\rangle$. Lemma 3.1 allows us to compute $\mathsf{Z}(60)$ in terms of $\mathsf{Z}(40)$, $\mathsf{Z}(51)$, and $\mathsf{Z}(54)$. In particular, each factorization of 54 yields a factorization of 60 with one additional copy of the irreducible 6. Similarly, each factorization of 51 or 40 yields a factorization for 60 with one additional copy of 9 or 20, respectively. We give the full computation below.

$$\begin{aligned} \mathsf{Z}(40) &= \{(0,0,2)\}, \\ \mathsf{Z}(51) &= \{(1,5,0),(4,3,0),(7,1,0)\}, \\ \mathsf{Z}(54) &= \{(0,6,0),(3,4,0),(6,2,0),(9,0,0)\}, \\ \mathsf{Z}(60) &= \{(0,0,3)\} \cup \{(1,6,0),(4,4,0),(7,2,0)\} \\ &\quad \cup \{(1,6,0),(4,4,0),(7,2,0),(10,0,0)\} \\ &= \{(0,0,3),(1,6,0),(4,4,0),(7,2,0),(10,0,0)\}. \end{aligned}$$

Since each factorization of 60 has at least one irreducible, the above computation produces the entire set $\mathsf{Z}(60)$. Given $n \geq 0$, Algorithm 3.3 uses this method inductively to compute $\mathsf{Z}(i)$ for all $i \in \{0, 1, \ldots, n\}$, starting with $\mathsf{Z}(0) = \{\mathbf{0}\}$.

TABLE 1. Runtimes for computing $\mathsf{Z}(n)$ and $\{(m, \mathsf{Z}(m)) : m \le n\}$ for $n \in S$. The last column gives runtimes for Algorithm 3.3 implemented in GAP, and the two previous columns use the implementation of $\mathsf{Z}(n)$ found in the GAP package `numericalsgps` [8].

S	n	$\lvert\mathsf{Z}(n)\rvert$	$\mathsf{Z}(n)$	$\{(m, \mathsf{Z}(m)) : m \le n\}$	Alg. 3.3
$\langle 10, 17, 19, 25, 31\rangle$	1000	20293	641 ms	55342 ms	20120 ms
$\langle 51, 53, 55, 117\rangle$	5000	1299	54 ms	61643 ms	3874 ms
$\langle 7, 15, 17, 18, 20\rangle$	1000	75375	1028 ms	234339 ms	102857 ms
$\langle 100, 121, 142, 163, 284\rangle$	30000	16569	2437 ms	13266033 ms	660320 ms

Algorithm 3.3. Given $n \in S = \langle n_1, \ldots, n_k\rangle$, computes $\mathsf{Z}(m)$ for all $m \in [0, n] \cap S$.

```
function FactorizationsUpToElement(S, n)
    F[0] ← {0}
    for all m ∈ [0, n] ∩ S do
        Z ← {}
        for all i = 1, 2, ..., k with m − n_i ∈ S do
            Z ← Z ∪ {a + e_i : a ∈ F[m − n_i]}
        end for
        F[m] ← Z
    end for
    return F
end function
```

Remark 3.4. Table 1 gives a runtime comparison for Algorithm 3.3 with the existing implementation of $\mathsf{Z}(n)$ in the GAP package `numericalsgps`. Notice that Algorithm 3.3 is slower than the current GAP implementition for computing a single factorization set $\mathsf{Z}(n)$, but faster for computing a large collection of factorization sets, such as $\{(m, \mathsf{Z}(m)) : m \le n\}$. The same holds true for memory usage: Algorithm 3.3 consumes more memory than the GAP implementation when computing a single factorization set, since it must store several factorization sets, but when computing $\{(m, \mathsf{Z}(m)) : m \le n\}$, it does not consume any more memory than is necessary for the returned list.

Typically, a length set for an element is computed only once its factorization set is known. Lemma 3.5 provides a dynamic algorithm that allows the computation of the length set of an element simply by knowing the lengths sets of smaller elements (and not their factorization sets).

Lemma 3.5. *Fix a reduced, finitely generated monoid M with irreducible elements $m_1, m_2, \ldots, m_k$. We have*

$$\mathsf{L}(x) = \bigcup_{i=1}^{k} (\mathsf{L}(x - m_i) + 1) = \bigcup_{i=1}^{k} \{l + 1 : l \in \mathsf{L}(x - m_i)\}$$

for each non-zero $x \in M$.

Proof. Apply the length function to both sides of the equality in Lemma 3.1. $\square$

Example 3.6. We resume notation from Example 3.2. Lemma 3.5 specializes the computation of $\mathsf{Z}(n)$ in Lemma 3.1 to length sets, allowing us to compute $\mathsf{L}(60)$ from $\mathsf{L}(40)$, $\mathsf{L}(51)$, and $\mathsf{L}(54)$. The key observation is that producing a factorization for 60 from a factorization of 40, 51 or 54 using Lemma 3.1 always increases length by exactly one. Given below is the full computation.

$$\begin{aligned} \mathsf{L}(60) &= (\mathsf{L}(40)+1) \cup (\mathsf{L}(51)+1) \cup (\mathsf{L}(54)+1) \\ &= \{3\} \cup \{7,8,9\} \cup \{7,8,9,10\} \\ &= \{3,7,8,9,10\}. \end{aligned}$$

Given $n \geq 0$, Algorithm 3.7 uses this method to dynamically compute $\mathsf{L}(m)$ for all $m \in [0,n] \cap S$, which Algorithm 3.9 then uses to compute $\bigcup_{m=1}^{n} \Delta(m)$. Notice that this does not require computing any factorizations; see Remark 3.10.

Algorithm 3.7. Given $n \in S = \langle n_1, \ldots, n_k \rangle$, computes $\mathsf{L}(m)$ for all $m \in [0,n] \cap S$.

function LengthSetsUpToElement(S, n)
  $\mathcal{L}[0] \leftarrow \{\mathbf{0}\}$
  for all $m \in [0,n] \cap S$ **do**
    $\mathsf{L} \leftarrow \{\}$
    for all $i = 1, 2, \ldots, k$ with $m - n_i \in S$ **do**
      $\mathsf{L} \leftarrow \mathsf{L} \cup \{l+1 : l \in \mathcal{L}[m-n_i]\}$
    end for
    $\mathcal{L}[m] \leftarrow L$
  end for
  return $\mathcal{L}$
end function

Remark 3.8. Since Algorithm 3.7 computes $\{\mathsf{L}(m) : m \in [0,n] \cap S\}$, it can also be used to compute $\bigcup_{m=1}^{n} \Delta(m)$ by first computing $\{\Delta(m) : m \in [0,n] \cap S\}$. As stated in Theorem 2.8, $\Delta(m)$ is periodic for m greater than an integer N_S described in [10]. Together with Algorithm 3.7, this yields Algorithm 3.9 for computing $\Delta(S)$.

Algorithm 3.9. Given a numerical monoid S, computes $\Delta(S)$.

function DeltaSet(S)
  Compute $\mathcal{L}$ using Algorithm 3.7
  Compute N_S using [10, Section 3]
  Compute $\Delta = \bigcup_{m \in (0, N_S + \mathrm{lcm}(n_1, n_k)] \cap S} \Delta(m)$
  return Δ
end function

Remark 3.10. In addition to giving an improved bound N_S on the start of periodic behavior of Δ_S for any numerical monoid $S = \langle n_1, \ldots, n_k \rangle$, the authors also give [10, Algorithm 21] to find $\Delta(S)$ by first computing $\mathsf{Z}(N_S + n_k - n_1), \ldots, \mathsf{Z}(N_S + n_k - 1)$. Table 2 gives a runtime comparison between Algorithm 3.9 and [10, Algorithm 21], demonstrating a marked improvement in computation time.

Our method of computing $\Delta(S)$ has several advantages over [10, Algorithm 21]. First and foremost is memory comsumption. For large n, factorization sets grow large very quickly [13], whereas $|\mathsf{L}(n)|$ grows linearly in n [2, Theorem 4.3]. Since [10, Algorithm 21] requires computing $\mathsf{Z}(n)$ for several large n, it is often memory intensive. Algorithm 3.9, on the other hand, avoids the computation of factorization sets altogether by only computing length sets.

Due in part to the low memory footprint, our algorithm is significantly faster than [10, Algorithm 21]. Some runtimes for [10, Algorithm 21] had to be omitted from Table 2, as the high memory requirements left us unable to complete the computation. However, [10, Table 2] gives a runtime for the computation of

TABLE 2. Runtime comparison for computing $\Delta(S)$. Several computations for [10, Algorithm 21] could not be completed due to insufficient available memory and hence have been omitted. All computations used GAP and the package numericalsgps [8].

S	N_S [10]	Dis.	$\Delta(S)$	[10, Alg. 21]	Alg. 3.9
$\langle 10, 17, 19, 25, 31\rangle$	1180	76	$\{1, 2, 3\}$	3254 ms	45 ms
$\langle 51, 53, 55, 117\rangle$	9699	1006	$\{2, 4, 6\}$	23565 ms	250 ms
$\langle 7, 15, 17, 18, 20\rangle$	1935	46	$\{1, 2, 3\}$	88831 ms	146 ms
$\langle 7, 19, 20, 25, 29\rangle$	3894	76	$\{1, 2, 3, 5\}$	——— ms	624 ms
$\langle 11, 53, 73, 87\rangle$	14381	873	$\{2, 4, 6, 8, 10, 22\}$	49418 ms	2588 ms
$\langle 31, 73, 77, 87, 91\rangle$	31364	558	$\{2, 4, 6\}$	——— ms	4274 ms
$\langle 100, 121, 142, 163, 284\rangle$	24850	5499	$\{21\}$	——— ms	3697 ms
$\langle 1001, 1211, 1421, 1631, 2841\rangle$	2063141	114535	$\{10, 20, 30\}$	——— ms	116371 ms

$\Delta(\langle 31, 73, 77, 87, 91\rangle)$ on the order of 24,000 seconds, a stark contrast to the 4.2 seconds required for Algorithm 3.9.

Remark 3.11. Although Algorithm 3.9 as stated requires the computation of $\mathsf{L}(i)$ for all $i \le N_S$, Algorithm 3.7 computes each $\mathsf{L}(i)$ using values at least $i - n_k$. As such, in implementing Algorithm 3.9 to compute $\Delta(S)$, one only needs to store n_k length sets at any given time. The GAP implemenation of Algorithm 3.9 stores length sets in a ring buffer of length n_k, cutting memory requirements even further.

Remark 3.12. Since the implementation of Algorithm 3.9 in the numericalsgps package [8], other promising delta set algorithms have been developed. Recent results of the second author of this manuscript [14] have produced an algorithm that computes the delta set of any affine monoid (a strictly more general setting) and appears to run faster than Algorithm 3.9. This algorithm is currently being implemented in GAP and will likely be included in a future version of the numericalsgps package. Additionally, a log-time algorithm to compute $\Delta(S)$ is given in [12] for the special case where $S = \langle n_1, n_2, n_3\rangle$ and S is non-symmetric.

4. ω-PRIMALITY IN THE QUOTIENT GROUP $\mathsf{q}(M)$

In the remaining sections, we turn our attention from factorization sets and delta sets to ω-primality (Definition 4.2), a factorization invariant that has recently received much attention [1, 11, 15, 16]. The results presented in these sections are motivated by those in Section 3, namely that dynamic computations can yield significant performance improvements when running a large collection of computations.

In this section, we provide definitions and motivation related to the ω-primality invariant. We begin with a definition of the quotient group of a monoid.

Remark 4.1. Let $\mathsf{q}(M)$ denote the quotient group of a monoid M. The map $M \to \mathsf{q}(M)$ given by $m \mapsto (m - 0)$ is an injective monoid homomorphism, and we often identify elements of M with their image in $\mathsf{q}(M)$ and view $M \subset \mathsf{q}(M)$. Under this convention, $x \mid y$ for elements $x, y \in \mathsf{q}(M)$ if $y = a + x$ for some $a \in M$.

The extension of the ω-function on a monoid to elements of its quotient group is one of the key insights of this paper. In this section, we develop the theory behind this extension, with minimal assumptions on the underlying monoid.

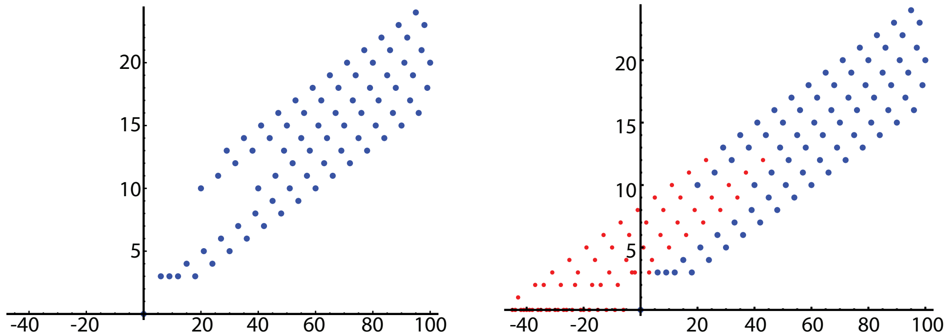

FIGURE 2. A plot of the ω-values of elements in the numerical monoid $S = \langle 6, 9, 20\rangle$ (left) and its quotient group (right) discussed in Example 4.5. The smaller red dots in the right-hand plot mark ω-values of elements lying outside of S.

Definition 4.2. Fix a monoid M. The *ω-primality function* $\omega_M : \mathsf{q}(M) \to \mathbb{N}\cup\{\infty\}$ is given by $\omega_M(x) = m$ if m is the smallest positive integer with the property that whenever $\sum_{i=1}^{r} a_i - x \in M$ for $r > m$ and $a_i \in M$, there exists a subset $T \subset \{1, \ldots, r\}$ with $|T| \leq m$ such that $\sum_{i\in T} a_i - x \in M$. If no such m exists, define $\omega_M(x) = \infty$. When M is clear from context, we simply write $\omega(x)$.

Remark 4.3. It is easy to check that the function ω_M defined above coincides with that defined in [16, Definitions 2.3 and 3.4] for elements of M, so Definition 4.2 simply extends the domain of ω_M from M to $\mathsf{q}(M)$. Example 4.5 demonstrates that this is a natural extension, and the results that follow show that many of the properties of the usual ω-function still hold in this new setting.

Remark 4.4. Written in a multiplicative setting, Definition 4.2 becomes more transparent; see [16, Definition 2.3] for more details. In particular, it is clear that prime monoids elements $x \in M$ characteristically satisfy $\omega(x) = 1$. In fact, much of the seminal work on ω-primality focused on computing ω-values for irreducible elements, as non-unique factorizations arise from non-prime irreducible elements.

Example 4.5. Let $S = \langle 6, 9, 20\rangle$ denote the numerical monoid from Example 3.2. Since S has finite complement in $\mathbb{N}$, $\mathsf{q}(S)$ is naturally isomorphic to $\mathbb{Z}$, with the obvious inclusion map. Figure 2 plots side-by-side the ω-values of S (as defined in [16]) and those of $\mathsf{q}(S)$ (from Definition 4.2). Notice the plotted values coincide for each $n \in S$, and the ω_S-values in the right-hand plot of elements lying in the complement of S seem to "fill in" the missing values in the left-hand plot.

One of the key ideas used to study ω-primality in recent years is its characterization in terms of bullet lengths [16, Proposition 2.10]. We now extend the definition of bullets to elements of the quotient group (Definition 4.6) and recover the usual characterization of ω-primality in terms of their length (Proposition 4.9). Note that Proposition 4.9 also implies that we may assume the elements a_i in Defintion 4.2 are irreducible.

Definition 4.6. Fix a monoid M. A *bullet* for $x \in \mathsf{q}(M)$ is an expression $u_1 + \cdots + u_r$ of irreducible elements $u_1, \ldots, u_r \in M$ such that (i) $u_1 + \cdots + u_r - x \in M$, and (ii) $u_1 + \cdots + u_r - x - u_i \notin M$ for each $i \le r$. The *value* of a bullet $u_1 + \cdots + u_r$ is the element $u_1 + \cdots + u_r \in M$, and its *length* is r. The set of bullets of x is denoted $\mathrm{bul}(x)$.

Remark 4.7. For a monoid element $m \in M$, the set $\mathrm{bul}(m)$ in Definition 4.6 is identical to the usual definition ([16, Definition 2.8]), since any element of $\mathsf{q}(M)$ that m divides also must lie in M. Both of these definitions differ slightly from the classical definition of a bullet; see [3].

Remark 4.8. Fix a monoid M and an element $x \in M$. If, additionally, M is both reduced (that is, M has no non-identity units) and finitely generated, then M has only finitely many irreducible elements $u_1, \ldots, u_k$. In this setting, we can denote a bullet $b_1u_1 + \cdots + b_ku_k \in \mathrm{bul}(x)$ by $\mathbf{b} = (b_1, \ldots, b_k) \in \mathbb{N}^k$. This is of particular use in Sections 5 and 6 (as well as Example 4.11), where ω-primality for numerical monoids is examined in more detail.

Proposition 4.9. *Given any monoid M,*

$$\omega(x) = \sup\{r : u_1 + \cdots + u_r \in \mathrm{bul}(x), u_i \textit{ irreducible}\}$$

for each element $x \in \mathsf{q}(M)$.

Proof. This is identical to the proof of [16, Proposition 2.10]. □

The notion of cover maps between bullet sets, first introduced in [15, Definition 3.4], will be critical in the dynamic algorithm for computing ω-primality (Algorithm 5.6). In the remainder of this section, we extend the original definition of cover maps to elements of the quotient group (Definition 4.13) and provide stronger results than the prior setting would allow (see Remark 4.14). First, we give Lemma 4.10, which was stated in [1] for numerical monoids and played a crucial role in developing the first algorithm to compute ω-primality.

Lemma 4.10. *Fix a monoid M, an element $x \in \mathsf{q}(M)$, a bullet $u_1 + \cdots + u_r \in \mathrm{bul}(x)$, and an irreducible element $u \in M$. Then $u + u_1 + \cdots + u_r \notin \mathrm{bul}(x)$.*

Proof. Omitting u yields the expression $u_1 + \cdots + u_r$, which x divides. □

Much like Lemma 3.1 for factorizations, the bullet set of an element x is determined by the bullet sets of its divisors. Definition 4.13 specifies how to determine the image of each bullet as demonstrated in Example 4.11, and Theorem 4.12 ensures the resulting map is well defined. First, we give an example.

Example 4.11. Let $S = \langle 6, 9, 20 \rangle$ denote the numerical monoid from Example 4.5. Consider $n = 60$ and the following bullet sets:

$$\begin{aligned}
\mathrm{bul}(40) &= \{(0,0,2), (4,4,0), (7,2,0), (10,0,0), (1,6,0), (0,8,0)\}, \\
\mathrm{bul}(51) &= \{(0,7,0), (10,0,0), (4,3,0), (1,5,0), (0,0,3), (7,1,0)\}, \\
\mathrm{bul}(54) &= \{(9,0,0), (6,2,0), (0,6,0), (3,4,0), (0,0,3)\}, \\
\mathrm{bul}(60) &= \{(4,4,0), (7,2,0), (10,0,0), (1,6,0), (0,8,0), (0,0,3)\}.
\end{aligned}$$

We see that for each bullet $\mathbf{b} \in \mathrm{bul}(54)$, either $\mathbf{b} \in \mathrm{bul}(60)$ or $\mathbf{b} + \mathbf{e}_1 \in \mathrm{bul}(60)$. Notice that it is impossible for both of these to lie in $\mathrm{bul}(60)$ by Lemma 4.10. Similarly, for each bullet $\mathbf{b} \in \mathrm{bul}(51)$, either $\mathbf{b} \in \mathrm{bul}(60)$ or $\mathbf{b} + \mathbf{e}_2 \in \mathrm{bul}(60)$, and for each $\mathbf{b} \in \mathrm{bul}(40)$, either $\mathbf{b} \in \mathrm{bul}(60)$ or $\mathbf{b} + \mathbf{e}_3 \in \mathrm{bul}(60)$. Moreover, each bullet for 60 is the "image" of a bullet for 54, 51 or 40 in this way. The resemblance to Lemma 3.1 here is not a coincidence; Theorem 6.2 makes this similarity precise.

Theorem 4.12. *Let M be a monoid. Fix an element $x \in \mathsf{q}(M)$, an irreducible $u \in M$, and a bullet $u_1 + \cdots + u_r \in \mathrm{bul}(x)$.*

(i) *If $u_1 + \cdots + u_r - (u + x) \in M$, then $u_1 + \cdots + u_r \in \mathrm{bul}(u + x)$.*
(ii) *If $u_1 + \cdots + u_r - (u + x) \notin M$, then $u + u_1 + \cdots + u_r \in \mathrm{bul}(u + x)$.*

Proof. Suppose $u_1 + \cdots + u_r - (u + x) \in M$. For each $i \le r$, we also have

$$u_1 + \cdots + u_r - (u_i + u + x) = (u_1 + \cdots + u_r - x - u_i) - u \notin M$$

since $u_1 + \cdots + u_r - (u_i + x) \notin M$. This means $u_1 + \cdots + u_r \in \mathrm{bul}(u + x)$.

Next, suppose $u_1 + \cdots + u_r - (u + x) \notin M$. We now verify the necessary conditions.

(i) Since $u_1 + \cdots + u_r \in \mathrm{bul}(x)$, we have $u + u_1 + \cdots + u_r - (u + x) = u_1 + \cdots + u_r - x \in M$.
(ii) For each $i \le r$, $u + u_1 + \cdots + u_r - (u_i + u + x) = u_1 + \cdots + u_r - (u_i - x) \notin M$.

Thus, $u + u_1 + \cdots + u_r \in \mathrm{bul}(u + x)$. □

Definition 4.13. Fix a monoid M, and an irreducible $u \in M$. For $x \in \mathsf{q}(M)$, the *u-cover map* $\psi_M^u : \mathrm{bul}(x) \to \mathrm{bul}(u + x)$ is given by

$$\psi_M^u(u_1 + \cdots + u_r) = \begin{cases} u_1 + \cdots + u_r & \text{if } u_1 + \cdots + u_r - (u + x) \in M, \\ u + u_1 + \cdots + u_r & \text{otherwise,} \end{cases}$$

for $u_1 + \cdots + u_r \in \mathrm{bul}(x)$. We often omit the subscript when there is no confusion.

Remark 4.14. The cover map ϕ_M^u was also defined in [15] in the context of numerical monoids, though its domain was restricted to bullets in which u appears. Both the extended domain of ψ_M^u in Definition 4.13 and the extended domain of ω_M in Definition 4.2 are crucial for the results that follow. In fact, Theorem 4.15 and Corollary 4.16 only held for "sufficiently large" elements of M prior to these extentions. This improved strength is our primary motivation for extending the domain of ω_M to $\mathsf{q}(M)$, as they ensure the correctness of Algorithm 5.6 for all (numerical) monoid elements.

Theorem 4.15. *Fix a monoid M, $x \in \mathsf{q}(M)$, and a bullet $u_1 + \cdots + u_r \in \mathrm{bul}(x)$. For each $j \le r$, $u_1 + \cdots + u_r - u_j \in \mathrm{bul}(x - u_j)$.*

Proof. Since $u_1 + \cdots + u_r \in \mathrm{bul}(x)$, we have

(i) $(u_1 + \cdots + u_r - u_j) - (x - u_j) = u_1 + \cdots + u_r - x \in M$, and
(ii) $(u_1 + \cdots + u_r - u_j) - (u_i + x - u_i) = u_1 + \cdots + u_r - (u_i + x) \notin M$

for each $i \ne j$. This means $u_1 + \cdots + u_r - u_j \in \mathrm{bul}(x - u_j)$. □

Corollary 4.16 follows directly from Theorems 4.12 and 4.15, and will serve as the inductive step of Algorithm 5.6. It also justifies use of the term "cover map".

Corollary 4.16. *If M is a monoid, $x \in \mathsf{q}(M)$, and $u_1 + \cdots + u_r \in \mathrm{bul}(x)$, then*

$$\mathrm{bul}(x) = \bigcup_{i \le r} \psi_M^{u_i}(\mathrm{bul}(x - u_i)).$$

We conclude the section with Proposition 4.17, which drastically simplifies the base case for Algorithm 5.6 (see Remark 5.1 for more detail).

Proposition 4.17. *If M is a monoid and $x \in \mathsf{q}(M)$, the following are equivalent:*

(i) $\omega(x) = 0$.
(ii) $\mathrm{bul}(x) = \{\mathbf{0}\}$.
(iii) $\mathbf{0} \in \mathrm{bul}(x)$.
(iv) $-x \in M$.

Proof. We show (i) $\implies$ (ii) $\implies$ (iii) $\implies$ (i) and (iii) $\Leftrightarrow$ (iv).

(i) $\implies$ (ii): If $\omega(x) = 0$, then any bullet for x has length at most 0, so $\mathrm{bul}(x) = \{\mathbf{0}\}$.

(ii) $\implies$ (iii): This is clear.

(iii) $\implies$ (i): This follows from Lemma 4.10.

(iii) $\Leftrightarrow$ (iv): This follows directly from Definition 4.2. □

Example 4.18. Let $S = \langle 6, 9, 20\rangle$ denote the numerical monoid in Example 4.5. Proposition 4.17 implies $\omega_S(n) = 0$ for every $n < -43 = -F(S)$; see Figure 2.

5. A dynamic programming algorithm for ω-primality

In this section, we apply the results of Section 4 in the setting of numerical monoids to obtain Algorithm 5.6 for dynamically computing ω-values, with significant runtime improvements over existing algorithms. Since any numerical monoid S has finite complement in $\mathbb{N}$, we have $\mathsf{q}(S) \cong \mathbb{Z}$ and often write $\mathbb{Z}$ in place of $\mathsf{q}(S)$.

Remark 5.1. One of the main difficulties in developing Algorithm 5.6 was determining where to start the inductive process. While Algorithms 3.3 and 3.7 naturally begin at the irreducible elements of S, the ω-value and bullet set of an irreducible element x are only trivial when x is prime, and this is never the case for non-trivial numerical monoids (see Proposition 5.2). One possible (though inelegant) solution is to use existing algorithms to compute the bullet sets of every element below some threshold, and then use a dynamic algorithm for all further values.

Thankfully, the extended notion of ω-primality presented in Section 4 allows us to avoid this caveat entirely. Proposition 5.5(a) states that for any numerical monoid S, the smallest element of $\mathsf{q}(S) = \mathbb{Z}$ on which ω_S takes a non-zero value is $-F(S)$, providing a natural base case for Algorithm 5.6.

As a non-trivial numerical monoid S contains no prime elements, ω_S only takes on values strictly greater than 1 on elements in S. Proposition 5.2, however, characterizes which elements of $\mathsf{q}(S)$ have an ω_S-value of 1. See [9, 17, 18] for more background on pseudo-Frobenius numbers.

Proposition 5.2. *Fix a numerical monoid $S = \langle n_1, \ldots, n_k\rangle$. For each $n \in \mathbb{Z}$, the following are equivalent.*

(i) $\omega(n) = 1$.
(ii) $\mathrm{bul}(n) = \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$.
(iii) *$-n$ is a pseudo-Frobenius number of S.*

Proof. We show (i) $\Leftrightarrow$ (ii) and (ii) $\Leftrightarrow$ (iii).

(i) $\implies$ (ii): Suppose $\omega(n) = 1$. For each $i \in \{1, \ldots, k\}$, we have $b_i\mathbf{e}_i \in \mathrm{bul}(n)$ for some $b_i \geq 0$ since numerical monoids are Archimedian, and by Proposition 4.17, $b_i > 0$. Since $\omega(n) = 1$, each $b_i = 1$.

(ii) $\implies$ (i): This follows from Proposition 4.9.

(ii) $\implies$ (iii): Suppose $\text{bul}(n) = \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$. For $i \in \{1, \ldots, k\}$, we have $n_i - n \in S$ and $n_i - n - n_i = -n \notin S$, so $-n$ is a pseudo-Frobenius number of S.

(iii) $\implies$ (ii): If $-n$ is a psuedo-Frobenius number of S, then $-n \notin S$ and $n_i - n \in S$ for each $i \in \{1, \ldots, k\}$, so $\mathbf{e}_i \in \text{bul}(n)$. Lemma 4.10 ensures no other bullets exist. $\square$

Remark 5.3. Bullets in numerical monoids are usually denoted either as tuples or as a list of irreducibles, the former of which is well-suited for storage when implementing algorithms. However, for the purpose of implementing Algorithm 5.6, each bullet $\mathbf{a} = (a_1, \ldots, a_k) \in \text{bul}(n)$ can be represented using only its value $v = a_1 n_1 + \cdots + a_k n_k$ and length $\ell = a_1 + \cdots + a_k$ (a *dynamic bullet*, see Definition 5.4). Indeed, v is sufficient for determining the image of $\mathbf{a}$ under cover maps, and ℓ is sufficient for computing $\omega(n)$ once all bullets have been found. This is the content of Proposition 5.5.

Since several distinct bullets can be represented by the same dynamic bullet, the cardinality of the computed bullet set with this representation is significantly reduced. Much like dynamically computing length sets instead of factorization sets greatly improves efficiency when computating delta sets (Remark 3.10), using this compact bullet representation significantly reduces the runtime and memory footprint when dynamically computing ω-values.

Proposition 5.5 is the analog of Lemmas 3.1 and 3.5 for Algorithm 5.6. First and foremost, parts (b) and (c) prove the correctness of Algorithm 5.6 by ensuring that dynamic bullets (Definition 5.4) are sufficient for computing ω-values. Moreover, part (a) provides a natural starting place for the inductive procedure in Algorithm 5.6. Lastly, part (d) proves that we may further restrict our attention to maximal dynamic bullets, and Corollary 6.6 demonstrates the benefit of this reduction (see Remark 6.5).

Definition 5.4. Fix a numerical monoid $S = \langle n_1, \ldots, n_k \rangle$ and an element $n \in \mathsf{q}(S)$. A *dynamic bullet for n* is an ordered pair $(v, \ell) \in \mathbb{N}^2$ such that $v = \sum_{i=1}^k b_i n_i$ and $\ell = \sum_{i=1}^k b_i$ are the value and length of some $\mathbf{b} \in \text{bul}(n)$, respectively. The set of dynamic bullets of n is denoted $\text{bul}^*(n)$. The *value* and *length* of a dynamic bullet $(v, \ell) \in \text{bul}^*(n)$ are the values v and ℓ, respectively, and (v, ℓ) is *maximal* if ℓ is maximal among dynamic bullets with value v.

Proposition 5.5. *Fix a numerical monoid $S = \langle n_1, \ldots, n_k \rangle$, and fix $i \in \{1, \ldots, k\}$ and $n \in \mathsf{q}(S)$. Let $\psi : \text{bul}(n) \to \text{bul}(n+n_i)$ denote the n_i-cover map, $\phi_n : \text{bul}(n) \to \text{bul}^*(n)$ denote the map given by*

$$\mathbf{b} \in \text{bul}(n) \longmapsto \left(\textstyle\sum_{i=1}^k b_i n_i, \sum_{i=1}^k b_i\right) \in \text{bul}^*(n),$$

and $\psi^* : \text{bul}^*(n) \to \text{bul}^*(n + n_i)$ denote the map given by

$$(v, \ell) \in \text{bul}^*(n) \mapsto \begin{cases} (v, \ell) & \text{if } v - (n_i + n) \in S, \\ (v + n_i, \ell + 1) & \text{if } v - (n_i + n) \notin S. \end{cases}$$

(a) *If $n < -F(S)$, then* $\text{bul}(n) = \{\mathbf{0}\}$.
(b) *For all $n \in \mathsf{q}(S)$, $\omega(n) = \max\{w : (v, w) \in \text{bul}^*(n)\}$.*

(c) *For all $n \in \mathsf{q}(S)$ and $i \in \{1, \ldots, k\}$, the following diagram commutes:*

$$\begin{array}{ccc} \mathrm{bul}(n) & \xrightarrow{\psi} & \mathrm{bul}(n+n_i) \\ \Big\downarrow \phi_n & & \Big\downarrow \phi_{n+n_i} \\ \mathrm{bul}^*(n) & \xrightarrow{\psi^*} & \mathrm{bul}^*(n+n_i) \end{array}$$

(d) *A dynamic bullet (v, ℓ) is maximal if and only if $\psi^*(v, \ell)$ is maximal.*

Proof. First, Theorem 4.12 ensures that the map ψ^* is well defined upon observing that the value and length of the image of a bullet $\mathbf{b}$ under ψ depend only on the value and length of $\mathbf{b}$. Now, parts (a) and (b) follow directly from Propositions 4.17 and 4.9, respectively, and part (c) follows immediately from Definitions 4.13 and 5.4. Finally, part (d) follows from part (c) and the observation that applying ψ^* to dynamic bullets $(v, \ell_1), (v, \ell_2) \in \mathrm{bul}^*(n)$ preserves any ordering on the second component. □

Algorithm 5.6. Given $n \in S = \langle n_1, \ldots, n_k \rangle$, computes $\omega(m)$ for all $m \in [0, n] \cap S$.

function OmegaPrimalityUpToElement(S, n)
  $B_m \leftarrow \{(0,0)\}$ for all $m < -F(S)$
  for all $m \in \{-F(S), \ldots, n\}$ **do**
    $B_m \leftarrow \{\}$
    for all $i \in \{1, 2, \ldots, k\}$ **do**
      for all $(v, \ell) \in B_{m-n_i}$ **do**
        $B_m \leftarrow B_m \cup \begin{cases} \{(v, \ell)\} & \text{if } v - (n_i + m) \in S \\ \{(v + n_i, \ell + 1)\} & \text{otherwise} \end{cases}$
      end for
    end for
    $B_m \leftarrow \{(v, \ell) \in B_m : (v, \ell) \text{ is maximal}\}$
    if $m \in S$ **then**
      $\omega[m] \leftarrow \max\{\ell : (v, \ell) \in B_m\}$
    end if
  end for
  return ω
end function

Remark 5.7. Table 3 gives a runtime comparison between Algorithm 5.6 and [3, Proposition 3.3] the algorithm previously implemented in the `numericalsgps` package. Not only is Algorithm 5.6 significantly faster, it also computes an extensive list of ω-values, rather than just a single value.

6. Computing ω-primality from factorizations

There are many similarities between Algorithms 3.3 and 5.6 beyond their dynamic nature. In non-precise terms, bullet sets and factorization sets seem to behave in a very similar manner. In this section, we make this connection explicit for numerical monoids S by characterizing the bullet set $\mathrm{bul}(n)$ of any element $n \in S$ in terms of factorization sets of certain submonoids of S. As an application, we greatly improve the existing bound [15] on the start of quasi-linear behavior (Theorem 6.8) of the ω-function on S; see Theorem 6.12.

6.1. **Runtime of Algorithm 5.6.** We first define an Apéry set, which will be utilized in Theorem 6.2 below. See [18] for a full treatment of Apéry sets.

Definition 6.1. Let M be a monoid. The *Apéry set* of $x \in M$ is defined as

$$\mathrm{Ap}(M, x) = \{m \in M : m - x \in \mathsf{q}(M) \setminus M\}.$$

Theorem 6.2. *Fix a reduced finitely generated monoid M, and let $G = \{u_1, \ldots, u_k\}$ denote the set of irreducible elements of M. For $\emptyset \neq A \subset G$ and $x \in M$, write*

$$\mathsf{Z}_A(x) = \{\mathbf{a} \in \mathsf{Z}(x) : a_i = 0 \textit{ whenever } u_i \notin A\} \subset \mathsf{Z}(x),$$

that is, the factorizations in $\mathsf{Z}_M(x)$ corresponding to factorizations in $\mathsf{Z}_{\langle A \rangle}(x)$. Then

$$\mathrm{bul}(x) = \left\{\mathbf{b} \in \mathsf{Z}_A(y + x) : \emptyset \neq A \subset G \textit{ and } y \in \textstyle\bigcap_{u_i \in A} \mathrm{Ap}(M; u_i)\right\}$$

for all $x \in \mathsf{q}(M)$ with $\omega(x) > 0$.

Proof. First, fix $A \subset G$ non-empty, $y \in \bigcap_{u_i \in A} \mathrm{Ap}(M; u_i)$, and $\mathbf{b} \in \mathsf{Z}_A(y + x)$. It follows that $\sum_{j=1}^k b_j u_j - x = y \in M$, and

$$\sum_{j=1}^{k} b_j u_j - (u_i + x) = y - u_i \notin M$$

for each $u_i \in A$, so $\mathbf{b} \in \mathrm{bul}(x)$.

Now, fix $\mathbf{b} \in \mathrm{bul}(x)$. Let $A = \{u_i : b_i \neq 0\} \subset G$ and $y = \sum_{j=1}^k b_j u_j - x$. Since $\omega(x) > 0$, A is non-empty by Proposition 4.17. Rearranging the equation for y gives $y + x = \sum_{j=1}^k b_j u_j$, meaning $\mathbf{b} \in \mathsf{Z}_A(y + x)$, and since $y \in M$ and $y - u_i \notin M$ for $b_i \neq 0$, we have $y \in \bigcap_{u_i \in A} \mathrm{Ap}(M; u_i)$. $\square$

Example 6.3. Let $S = \langle 6, 9, 20\rangle$ and $n = 60 \in S$. Our goal is to compute bul(60) using Theorem 6.2. For any subset $A \subset \{6, 9, 20\}$ such that $\gcd(A) > 1$ and $60 \in \langle A\rangle$, each $m \in \bigcap_{a \in A} \mathrm{Ap}(S; a)$ is relatively prime to $\gcd(A)$, so $\mathsf{Z}_A(60 + m) = \emptyset$. The only subsets not satisfying both of these conditions are $A = \{9\}$ and $A = \{6, 9, 20\}$. Since $\mathrm{Ap}(S; 6) \cap \mathrm{Ap}(S; 9) \cap \mathrm{Ap}(S; 20) = \{0\}$ and the only non-zero $m \in \mathrm{Ap}(S; 9)$ with $\mathsf{Z}_{\{9\}}(60 + m)$ non-empty is $m = 12$, we compute bul(60) as follows:

$$\begin{aligned} \mathrm{bul}(60) &= \mathsf{Z}_{\{6,9,20\}}(60 + 0) \cup \mathsf{Z}_{\{9\}}(60 + 12) \\ &= \{(4, 4, 0), (7, 2, 0), (10, 0, 0), (1, 6, 0), (0, 0, 3)\} \cup \{(0, 8, 0)\}. \end{aligned}$$

Remark 6.4. For each element n in a numerical monoid S, Theorem 6.2 gives a way to compute the set $\mathrm{bul}(n)$ (and thus $\omega(n)$) by computing factorizations of elements in certain submonoids of S. Since factorization sets in numerical monoids can be computed rather quickly in GAP [8], this yields an algorithm for computing ω-primality. This turns out to be faster than [3, Proposition 3.3], the algorithm widely used to compute ω-primality in numerical monoids prior to Algorithm 5.6, but not as fast as Algorithm 5.6, as it still relies on computing $\mathrm{bul}(n)$, rather than the (substantially smaller) set $\mathrm{bul}^*(n)$. Additionally, algorithmic use of Theorem 6.2 only produces a single ω-value, whereas Algorithm 5.6 produces an exhaustive list of ω-values.

TABLE 3. Runtime comparison for computing the ω-value of $n \in \mathsf{q}(S)$. All computations performed using the GAP package `numericalsgps` [8] and do not make use of eventual quasi-linearity.

S	n	$\omega(n)$	[3, Prop. 3.3]	Algorithm 5.6
$\langle 6, 9, 20\rangle$	1000	170	61319 ms	6 ms
$\langle 11, 13, 15\rangle$	1000	97	10732 ms	5 ms
$\langle 11, 13, 15\rangle$	3000	279	874799 ms	15 ms
$\langle 11, 13, 15\rangle$	10000	915	———	42 ms
$\langle 15, 27, 32, 35\rangle$	1000	69	234738 ms	9 ms
$\langle 10, 12, 15, 16, 17\rangle$	500	52	14338290 ms	40 ms
$\langle 10, 12, 15, 16, 17\rangle$	50000	5002	———	4084 ms
$\langle 100, 121, 142, 163, 284\rangle$	25715	308	———	27449 ms
$\langle 1001, 1211, 1421, 1631, 2841\rangle$	357362	405	———	3441508 ms

Remark 6.5. One immediate consequence of Theorem 6.2 is that the number of distinct values occurring for dynamic bullets in $\mathrm{bul}^*(n)$ is at most $|\bigcup_{i=1}^{k} \mathrm{Ap}(S; n_i)|$. In particular, this bound does not depend on n. Since Algorithm 5.6 only stores a single dynamic bullet with each value (Proposition 5.5(d)), the inductive step runs in constant time in n (for fixed S). This yields Corollary 6.6.

Corollary 6.6. *For fixed S, Algorithm* 5.6 *runs in linear time in n.*

6.2. **Bounding the eventual quasilinearity of ω-primality.** For numerical monoids, the ω-function admits a predictable behavior for sufficiently large values, a result that appeared as [15, Theorem 3.6] and independently as [11, Corollary 20]. Before stating this result here as Theorem 6.8, we give a definition.

Definition 6.7. A map $f : \mathbb{N} \to \mathbb{N}$ is *quasilinear* if $f(n) = a_1(n)n + a_0(n)$, where $a_1, a_2 : \mathbb{N} \to \mathbb{Q}$ are each periodic and $a_1(n)$ is not identically zero.

Theorem 6.8. [15, Theorem 3.6] *Fix a numerical monoid $S = \langle n_1, \ldots, n_k\rangle$. The ω-primality function is eventually quasilinear. In particular, there is a periodic function $a(n)$ with period n_1 and integer N_0 such that for $n > N_0$,*

$$\omega_S(n) = \frac{1}{n_1}n + a(n).$$

Remark 6.9. The value of N_0 described in Theorem 6.8, which gives an upper bound for the start of the quasi-linear behavior of w, is explicitly given in [15]. The actual start of the quasi-linear behavior, which is called the *dissonance*, is very often significantly lower. Theorem 6.12 will greatly improve the bound for the dissonance.

Example 6.10. Figure 3 plots the ω-values for the numerical monoids $\langle 4, 13, 19\rangle$ and $\langle 6, 9, 20\rangle$. The eventual quasi-linearity ensured by Theorem 6.8 manifests graphically as a collection of discrete lines with identical slopes, and is readily visible in these plots. The proof of Theorem 6.8 given in [15] yields a precise (but computationally impractical) bound on the start of this quasi-linear behavior, and Theorem 6.12 drastically improves this bound; see Table 4 for a detailed comparison.

We now apply Theorem 6.2 to improve the bound on the dissonance point of the ω-function. As with the original bound on the dissonance point given in [15],

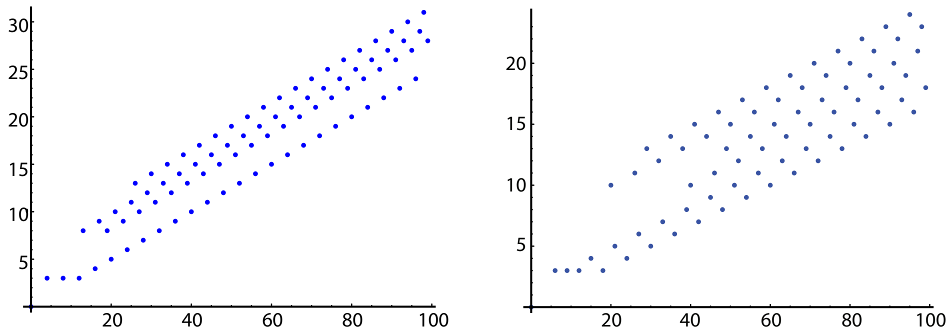

FIGURE 3. A plot of ω_S for the numerical monoids $S = \langle 4, 13, 19\rangle$ (left) and $\langle 6, 9, 20\rangle$ (right) discussed in Example 6.10.

this result is heavily motivated by computational evidence, in this case provided by Algorithm 5.6.

Lemma 6.11. *Fix a submonoid $S = \langle n_1, n_2, \ldots, n_k\rangle \subset \mathbb{N}$. For $n \in S$, we have*

$$M(n) \le n/n_1 \le \omega(n),$$

where $M(n)$ is the maximum length of a factorization for n.

Proof. Fix $b_1 > 0$ such that $\mathbf{b} = b_1\mathbf{e}_1 \in \mathrm{bul}(n)$ (note that b_1 exists since numerical monoids are Archimedian). This gives $\omega(n) \ge |\mathbf{b}| \ge n/n_1$. Additionally, we have $n = \sum_{j=1}^{k} a_j n_j \ge a_1 n_1$ for each factorization $\mathbf{a} \in \mathsf{Z}(n)$, meaning $M(n) \le n/n_1$. □

Theorem 6.12. *Fix a numerical monoid $S = \langle n_1, \ldots, n_k\rangle$ and $n > N_0$, where*

$$N_0 = \frac{F(S) + n_2}{n_2/n_1 - 1}.$$

Any maximal bullet $\mathbf{b} \in \mathrm{bul}(n)$ satisfies $b_1 > 0$, and $\omega(n) = \omega(n - n_1) + 1$.

Proof. Fix a bullet $\mathbf{b} \in \mathrm{bul}(n)$, and let $A = \{n_i : b_i > 0\}$. First, suppose $b_1 = 0$. By Theorem 6.2, we have $\mathbf{b} \in \mathsf{Z}(n+s)$ for some $s \in \bigcap_{n_i \in A} \mathrm{Ap}(S; n_i)$. In particular, $s - \min(A) \notin S$, so $s - \min(A) \le F(S)$. Since $n > N_0$, we have

$$|\mathbf{b}| \le M_{\langle A\rangle}(n+s) \le \frac{n+s}{\min(A)} \le \frac{n+F(S)}{\min(A)} + 1 \le \frac{n+F(S)}{n_2} + 1 < n/n_1 \le \omega_S(n)$$

by Lemma 6.11, meaning $\mathbf{b}$ is not maximal. This proves the first statement.

Now, suppose $\mathbf{b}$ is maximal, and let $\mathbf{a} = \mathbf{b} - \mathbf{e}_1$. The above argument implies $b_1 > 0$, so by Theorem 4.15, we have $\mathbf{a} \in \mathrm{bul}(n - n_1)$. This gives

$$\omega(n) - 1 \ge \omega(n - n_1) \ge |\mathbf{a}| = |\mathbf{b}| - 1 = \omega(n) - 1,$$

where the first inequality follows from Theorem 4.12(ii). □

Remark 6.13. The bound given in [15] for the dissonance point of the ω-function of a numerical monoid S, while explicit, is far too large to be feasibly reached in computation. In contrast, the new bound N_0 given in Theorem 6.12 can actually be reached computationally; that is, it is often possible to compute every ω-value below N_0 (using Algorithm 5.6, for instance). See Table 4 for a detailed comparison

TABLE 4. Comparison for the start of quasi-linear behavior of the ω-function. The dissonance points were computed using Algorithm 5.6, available in the next release of the GAP package `numericalsgps` [8].

S	Dissonance	Theorem 6.12	n_0 [15, Theorem 3.6]
$\langle 6, 9, 20\rangle$	12	104	37,800
$\langle 10, 12, 15\rangle$	190	325	66,600
$\langle 10, 12, 15, 16, 17\rangle$	10	175	34,272,000
$\langle 10, 12, 13, 14, 15, 16, 17, 18, 19, 21\rangle$	10	115	450,087,321,600
$\langle 100, 121, 142, 163, 284\rangle$	100	25715	64,426,520,664,000
$\langle 1001, 1211, 1421, 1631, 2841\rangle$	1001	357362	$\approx 6 \cdot 10^{19}$

of these bounds. The ω-value of any element above N_0 is determined by the ω-values of the elements just below N_0, and can be obtained by simply evaluating a linear function. This makes it possible to compute the ω-value of any element of S.

7. Future work

It is natural to ask whether the dynamic algorithms given above can be extended to settings outside the realm of numerical monoids. In fact, Algorithms 3.3 and 3.7 extend immediately to any finitely generated monoid M, once a suitably "bounded" subset of M is chosen (for instance, one could use the set of divisors of some element $m \in M$). In order to generalize Algorithm 3.9, the eventual behavior of $\Delta : M \to 2^{\mathbb{Z}}$ must be examined.

Problem 7.1. Characterize the eventual behavior of $\Delta : M \to 2^{\mathbb{Z}}$ for M a finitely generated monoid.

It is worth noting that the generalization of Algorithm 3.7 to finitely generated monoids does allow the use of computation packages for investigating Problem 7.1.

Generalizing Algorithm 5.6 presents a slightly more subtle issue. Section 4 extends the ω-function on a finitely generated monoid M to its quotient group $\mathsf{q}(M)$, along with both the "base case" and the "inductive step" for Algorithm 5.6. However, given an element $m \in M$, the set of divisors of m with non-zero ω-value need not be finite; see Example 7.2. Thus, more care must be taken in order to dynamically compute ω-values in M, and it is as yet unclear how to make this distinction.

Example 7.2. Consider the monoid $M \subset \mathbb{N}^2$ consisting of all points not lying above the ray generated by $(2, 1)$. Definition 4.2 extends the domain of ω_M to all of $\mathbb{Z}^2$, and Proposition 4.17 states that $\omega_M(m) = 0$ precisely when $m \in -M$. However, the set of divisors of any non-zero element $m \in M$ contains infinitely many elements of $\mathbb{Z}^2$ which necessarily have both non-zero ω-value and non-trivial bullet sets; see Figure 4.

Problem 7.3. Develop a dynamic programming algorithm to compute ω-values for finitely generated monoids.

Given an element $n \in S$, its *catenary degree* $c(n)$ can be computed from a graph with vertex set $\mathsf{Z}(n)$ [6]. Like delta sets, the catenary degree function $c : S \to \mathbb{N}$ is

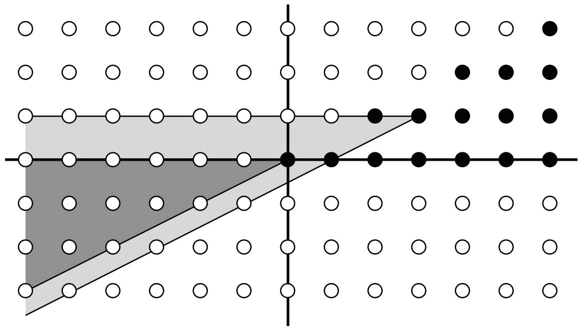

FIGURE 4. The monoid $M \subset \mathbb{N}^2$ given in Example 7.2. The elements of M are marked by filled dots, the lightly shaded region marks the divisors of $m = (3, 1)$, and the dark region marks $-M$.

periodic for large input values [5], but no bound on the start of this periodic behavior is known, and the period is only known to divide the product of the generators of S. The iterative nature of the periodicity proof given in [5] hints at the existence of a dynamic algorithm for computing catenary degrees in numerical monoids, allowing for the use of computation when investigating the eventual periodic behavior.

Problem 7.4. Find a dynamic programming algorithm to compute the catenary degrees of numerical monoid elements.

Acknowledgements

The authors would like to thank Pedro García-Sánchez for numerous insightful conversations.

References

[1] D. F. Anderson, S. T. Chapman, N. Kaplan, and D. Torkornoo, *An algorithm to compute ω-primality in a numerical monoid*, Semigroup Forum **82** (2011), no. 1, 96–108, DOI 10.1007/s00233-010-9259-5. MR2753835

[2] T. Barron, C. O'Neill, and R. Pelayo, *On the set of elasticities in numerical monoids*, preprint, available at arXiv:1409.3425. To appear in *Semigroup Forum.*

[3] V. Blanco, P. A. García-Sánchez, and A. Geroldinger, *Semigroup-theoretical characterizations of arithmetical invariants with applications to numerical monoids and Krull monoids*, Illinois J. Math. **55** (2011), no. 4, 1385–1414 (2013). MR3082874

[4] C. Bowles, S. T. Chapman, N. Kaplan, and D. Reiser, *On delta sets of numerical monoids*, J. Algebra Appl. **5** (2006), no. 5, 695–718, DOI 10.1142/S0219498806001958. MR2269412

[5] S. T. Chapman, M. Corrales, A. Miller, C. Miller, and D. Patel, *The catenary and tame degrees on a numerical monoid are eventually periodic*, J. Aust. Math. Soc. **97** (2014), no. 3, 289–300, DOI 10.1017/S1446788714000330. MR3270769

[6] S. T. Chapman, P. A. García-Sánchez, and D. Llena, *The catenary and tame degree of numerical monoids*, Forum Math. **21** (2009), no. 1, 117–129, DOI 10.1515/FORUM.2009.006. MR2494887

[7] S. T. Chapman, R. Hoyer, and N. Kaplan, *Delta sets of numerical monoids are eventually periodic*, Aequationes Math. **77** (2009), no. 3, 273–279, DOI 10.1007/s00010-008-2948-4. MR2520501

[8] M. Delgado, P. García-Sánchez, and J. Morais, *GAP Numerical Semigroups Package*, http://www.gap-system.org/Manuals/pkg/numericalsgps/doc/manual.pdf.

[9] P. Freyd, *Redei's finiteness theorem for commutative semigroups*, Proc. Amer. Math. Soc. **19** (1968), 1003. MR0227290

[10] J. I. García-García, M. A. Moreno-Frías, and A. Vigneron-Tenorio, *Computation of delta sets of numerical monoids*, Monatsh. Math. **178** (2015), no. 3, 457–472, DOI 10.1007/s00605-015-0785-9. MR3411252

[11] J. I. García-García, M. A. Moreno-Frías, and A. Vigneron-Tenorio, *Computation of the ω-primality and asymptotic ω-primality with applications to numerical semigroups*, Israel J. Math. **206** (2015), no. 1, 395–411, DOI 10.1007/s11856-014-1144-6. MR3319645

[12] P. García-Sánchez, D. Llena, and A. Moscariello, *Delta sets for numerical semigroups with embedding dimension three*, preprint. Available at arXiv: math.AC/1504.02116

[13] F. Halter-Koch, *On the asymptotic behaviour of the number of distinct factorizations into irreducibles*, Ark. Mat. **31** (1993), no. 2, 297–305, DOI 10.1007/BF02559488. MR1263556

[14] C. O'Neill, *On factorization invariants and Hilbert functions*, preprint. Available at arXiv: math.AC/1503.08351.

[15] C. O'Neill and R. Pelayo, *On the linearity of ω-primality in numerical monoids*, J. Pure Appl. Algebra **218** (2014), no. 9, 1620–1627, DOI 10.1016/j.jpaa.2014.01.002. MR3188860

[16] C. O'Neill and R. Pelayo, *How do you measure primality?*, Amer. Math. Monthly **122** (2015), no. 2, 121–137, DOI 10.4169/amer.math.monthly.122.02.121. MR3324682

[17] J. C. Rosales and M. B. Branco, *Decomposition of a numerical semigroup as an intersection of irreducible numerical semigroups*, Bull. Belg. Math. Soc. Simon Stevin **9** (2002), no. 3, 373–381. MR2016576

[18] J. C. Rosales and P. A. García-Sánchez, *Numerical semigroups*, Developments in Mathematics, vol. 20, Springer, New York, 2009. MR2549780

Department of Mathematics, University of Kentucky, Lexington, Kentucky 40506
E-mail address: thomas.barron@uky.edu

Department of Mathematics, Texas A&M University, College Station, Texas 77843
Current address: Department of Mathematics, University of California Davis, One Shields Avenue, Davis, California 95616
E-mail address: coneill@math.ucdavis.edu

Department of Mathematics, University of Hawai'i at Hilo, Hilo, Hawaii 96720
E-mail address: robertop@hawaii.edu