Slope packings and coverings, and generic algorithms for the discrete logarithm problem

M. Chateauneuf
Department of Combinatorics and Optimization
University of Waterloo
Waterloo ON, N2L 3G1
Canada

A.C.H. Ling
Department of Computer Science
University of Vermont
Burlington VT, 05405
USA

D.R. Stinson
Department of Combinatorics and Optimization
University of Waterloo
Waterloo ON, N2L 3G1
Canada

November 5, 2001

Abstract

We consider the set of slopes of lines formed by joining all pairs of points in some subset S of a Desarguesian affine plane of prime order p. If all the slopes are distinct and non-infinite, we have a *slope packing*; if every possible non-infinite slope occurs, then we have a *slope covering*. We review and unify some results on these problems that can be derived from the study of Sidon sets and sum covers. Then we report some computational results we have obtained for small values of p. Finally, we point out some connections between slope packings and coverings and generic algorithms for the discrete logarithm problem in prime order (sub)groups. Our results provide a combinatorial characterization of such algorithms, in the sense that any generic algorithm implies the existence of a certain slope packing or covering, and conversely.

1 Introduction and Definitions

We are interested in the set of slopes defined by a set of points in the desarguesian affine plane $\mathsf{AG}(2,p)$, where p is prime. We begin with some notation: Let $p \geq 2$ be prime. Define $\Delta : \mathbb{Z}_p \times \mathbb{Z}_p \to (\mathbb{Z}_p \cup \{\infty\})$ as follows:

$$\Delta((x_1,y_1),(x_2,y_2)) = \begin{cases} (y_2-y_1)(x_2-x_1)^{-1} \bmod p & \text{if } x_2 \neq x_1 \\ \infty & \text{otherwise.} \end{cases}$$

$\Delta((x_1,y_1),(x_2,y_2))$ is the slope of the line in $\mathsf{AG}(2,p)$ that joins the two points (x_1,y_1) and (x_2,y_2). Observe that $\Delta((x_1,y_1),(x_2,y_2)) = \Delta((x_2,y_2),(x_1,y_1))$.

For $S \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$, let $\Delta(S)$ denote the set of slopes which is defined as follows:

$$\Delta(S) = \{\Delta((x_1,y_1),(x_2,y_2)) : (x_1,y_1),(x_2,y_2) \in S, (x_1,y_1) \neq (x_2,y_2)\}.$$

Definition 1.1. Suppose that $S \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$, where p is prime. We say that S is a *slope packing* if $\Delta(S) \subseteq \mathbb{Z}_p$ and $|\Delta(S)| = \binom{|S|}{2}$; and S is a *slope cover* if $\mathbb{Z}_p \subseteq \Delta(S)$. We will denote a slope packing (covering, resp.) of size w in $\mathbb{Z}_p$ by the notation $\mathsf{SP}(p,w)$ ($\mathsf{SC}(p,w)$, resp.).

Let

$$N(p) = \min\{w : \text{there exists a } \mathsf{SC}(p,w)\}$$

and let

$$D(p) = \mathsf{max}\{w : \text{there exists a } \mathsf{SP}(p, w)\}.$$

As well, denote

$$N^*(p) = \mathsf{min}\left\{w \in \mathbb{Z}^+ : \binom{w}{2} \geq p\right\}$$

and

$$D^*(p) = \mathsf{max}\left\{w \in \mathbb{Z}^+ : \binom{w}{2} \leq p\right\}.$$

The following combinatorial bounds are obvious.

Proposition 1.1. *$N(p) \geq N^*(p)$ and $D(p) \leq D^*(p)$.*

Here is an easy upper bound on the size of a minimum slope cover (i.e., an existence result). It is derived from the Shanks baby-step giant-step algorithm for the discrete logarithm problem.

Proposition 1.2. *For any prime p, $N(p) \leq 2\lceil\sqrt{p}\rceil$.*

Proof. Let $m = \lceil\sqrt{p}\rceil$. Clearly, the collection

$$\{(1, mi \bmod p) : 0 \leq i \leq m-1\} \cup \{(0, -j \bmod p) : 0 \leq j \leq m-1\}$$

is a $\mathsf{SC}(p, 2m)$. □

The rest of this paper is organized as follows. In Section 2, discuss Sidon sets and sum covers, which give rise to certain special types of slope covers and packings. We review some known results on Sidon sets and sum covers, and provide a unified approach to the derivation of bounds on these objects. In Section 3, we present some computational results for small p. These results suggest that general slope coverings and packings often exist that are better than those obtained from Sidon sets and sum covers. These computational results also provide some evidence that slope coverings and packings always exist that are reasonably close to the trivial counting bounds for such objects. In Section 4, we turn to the Discrete Logarithm problem, and discuss the connections between generic algorithms for the Discrete Logarithm problem in (sub)groups of prime order and slope packings and coverings. We provide a fairly tight combinatorial characterization linking these concepts, and derive the well-known Nechaev-Shoup lower bound on the complexity of a generic algorithm for the Discrete Logarithm problem as an immediate corollary of one of our results. Section 5 is a summary and brief discussion of future research directions.

2 Sidon Sets and Sum Covers

For any $i \in \mathbb{Z}_p$, define $f(i) = (i, i^2 \bmod p)$. For any $T \subseteq \mathbb{Z}_p$, define $f(T) = \{f(i) : i \in T\}$. We have the following easy result.

Proposition 2.1. *Let p be a prime and let $T \subseteq \mathbb{Z}_p$. Then $f(T)$ is a slope packing if and only if there do not exist two different unordered pairs of elements in T having the same sum modulo p. Also, $f(T)$ is a slope cover if and only if for every $g \in \mathbb{Z}_p$, there is a pair of distinct elements $a, b \in T$, such that $g \equiv a + b \pmod{p}$.*

Proof. It is easy to see that $\Delta((i, i^2 \bmod p), (j, j^2 \bmod p)) = i + j \bmod p$ if $i \neq j$. The results follow. □

Let $T \subseteq \mathbb{Z}_n$. T is called a *weak Sidon set* modulo n provided that, for any four distinct elements, $w, x, y, z \in T$, it holds that $w + x \not\equiv y + z \pmod{p}$. Such sets have applications, including constructions of constant-weight codes (see [2]).

Define $D_{\text{sum}}(n)$ to be the maximum size of a weak Sidon set modulo n. From Proposition 2.1, it is clear that $D_{\text{sum}}(p) \leq D(p)$ if p is prime.

Denote

$$D^*_{\text{sum}}(n) = \mathsf{max}\left\{w \in \mathbb{Z}^+ : w(w-3) + 1 \leq n\right\}.$$

The following upper bound on $D_{\text{sum}}(n)$ is proved in [7] and [3].

Proposition 2.2. *Suppose n is odd. Then $D_{\text{sum}}(n) \le D^*_{\text{sum}}(n)$.*

One focus of the paper [3] is computer searches for maximum cardinality weak Sidon sets modulo n. The following weak Sidon sets, appearing in both papers [7] and [3], are optimal: $\{0,1,2,4\} \subseteq \mathbb{Z}_7$, and $\{0,1,2,4,7\} \subseteq \mathbb{Z}_{11}$.

Suppose $T \subseteq \mathbb{Z}_n$ has the property that for any $g \in \mathbb{Z}_n$ there are distinct elements $x, y \in T$ such that $g \equiv x + y \pmod{n}$. Such a set T will be called a *sum cover* modulo n. Define $N_{\text{sum}}(n)$ to be the minimum size of a sum cover modulo n. From Proposition 2.1, it is clear that $N_{\text{sum}}(p) \ge N(p)$ if p is prime.

We present a simple construction that gives an easily stated upper bound on $N_{\text{sum}}(n)$.

Proposition 2.3.

$$N_{\text{sum}}(n) \le \begin{cases} 2r-1 & \text{if } r^2-2 \le n \le r^2+r-3 \\ 2r & \text{if } r^2+r-2 \le n \le r^2+2r-2. \end{cases}$$

Proof. First, observe that every positive integer n falls into exactly one of the two cases enumerated above. For positive integers r and k, define

$$T(r,k) = \{0,1,\dots,r-1\} \cup \{2r-2, 3r-2, \dots, kr-2\}.$$

It is easy to show that $T(r,k)$ is a sum cover of $\mathbb{Z}_n$ if $n \le (k+1)r - 3$. Also, $|T(r,k)| = r + k - 1$. The desired result is obtained by taking $T(r,r)$ if $r^2 - 2 \le n \le r^2 + r - 3$, and by taking $T(r, r-1)$ if $r^2 + r - 2 \le n \le r^2 + 2r - 2$. $\square$

Denote

$$N^*_{\text{sum}}(n) = \min\left\{ w \in \mathbb{Z}^+ : \frac{(w-1)^2+2}{2} \ge n \right\}.$$

The following lower bound on $N_{\text{sum}}(n)$ is proved in [1].

Proposition 2.4. *Suppose n is odd. Then $N_{\text{sum}}(n) \ge N^*_{\text{sum}}(n)$.*

We now briefly point out that Propositions 2.2 and 2.4 can be proven in a unified manner using a linear programming appraoch. Let $T \subseteq \mathbb{Z}_n$, where n is odd, and denote $|T| = w$. For $g \in \mathbb{Z}_n$, define

$$s(g) = |\{(t,t') \in T \times T : t + t' \equiv g \pmod{n}\}|$$

and

$$d(g) = |\{(t,t') \in T \times T : t - t' \equiv g \pmod{n}\}|.$$

It is clear that $0 \le s(g) \le w$ and $0 \le d(g) \le w$ for all g. For $0 \le i \le w$, define

$$s_i = |\{g \in \mathbb{Z}_n : s(g) = i\}|$$

and

$$d_i = |\{g \in \mathbb{Z}_n : d(g) = i\}|.$$

The following equations are easily proven (see [1]):

$$\sum_{i=0}^{w} s_i = \sum_{i=0}^{w} d_i = n \tag{1}$$

$$\sum_{i=0}^{w} i s_i = \sum_{i=0}^{w} i d_i = w^2 \tag{2}$$

$$\sum_{i=0}^{w} i^2 s_i = \sum_{i=0}^{w} i^2 d_i. \tag{3}$$

It is also clear that $d(0) \ge w$, which implies that $d_w \ge 1$.

It is now a simple matter to show, for any integer ℓ, that the following inequality holds:

$$\sum_{i=0}^{w}(i-\ell)(i-\ell-1)s_i \geq (w-\ell)(w-\ell-1). \tag{4}$$

The proof of (4) follows from (1) – (3):

$$\begin{aligned}\sum_{i=0}^{w}(i-\ell)(i-\ell-1)s_i &= \sum_{i=0}^{w}(i-\ell)(i-\ell-1)d_i \\ &\geq (w-\ell)(w-\ell-1)d_w \\ &\geq (w-\ell)(w-\ell-1).\end{aligned}$$

Because n is odd, the mapping $t \mapsto 2t \bmod n$ is injective. Considering points of the form (t,t), it is clear that the following equation holds:

$$\sum_{\substack{1 \leq i \leq w \\ i \text{ odd}}} s_i = w. \tag{5}$$

Observe that (1) – (5) hold for any set T. By making some appropriate assumptions on T, we can bound n in terms of w.

First, suppose that T is a weak Sidon set modulo n. Then $s_i = 0$ if $i \geq 4$. In this situation, equations (5), (2) and (4) (with $\ell = 1$) imply the following:

$$s_1 + s_3 = w \tag{6}$$

$$s_1 + 2s_2 + 3s_3 = w^2 \tag{7}$$

$$2s_0 + 2s_3 \geq (w-1)(w-2). \tag{8}$$

We now have a linear programming problem: find the maximum value of $n = s_0 + s_1 + s_2 + s_3$, subject to the constraints $s_i \geq 0$ for all i, and the constraints (6) – (8). It can be shown that the optimal solution to this LP is $n = w(w-3)+1$, and it is achieved when

$$\begin{aligned}s_0 &= \frac{w^2-5w+2}{2} \\ s_1 &= 0 \\ s_2 &= \frac{w^2-3w}{2} \\ s_3 &= w.\end{aligned}$$

In the case of sum covers, we have $s_0 = s_1 = 0$. Suppose w is odd. Then equations (5), (2) and (4) (with $\ell = 2$) imply the following:

$$s_3 + s_5 + \cdots + s_w = w \tag{9}$$

$$2s_2 + 3s_3 + \cdots + ws_w = w^2 \tag{10}$$

$$2s_4 + 6s_5 + 12s_6 + \cdots + (w-2)(w-3)s_w \geq (w-2)(w-3). \tag{11}$$

The case when w is even is similar, except that (9) is replaced by

$$s_1 + s_3 + \cdots + s_{w-1} = w \tag{12}$$

It can be shown that the maximum value of $n = s_3 + s_5 + \cdots + s_w$, subject to the constraints $s_i \geq 0$ for all i, and the constraints (9) – (11) (when w is odd) or (10) – (12) (when w is even), is $(w^2-2w+2)/2$. Further, the optimal solution is obtained when

$$\begin{aligned}s_2 &= \frac{w^2-4w}{2} \\ s_3 &= w \\ s_w &= 1.\end{aligned}$$

3 Computational Results

The $\mathsf{SC}(p,w)$ and $\mathsf{SP}(p,w)$ in Tables 1 and 2 were found via computer searches for primes $p < 100$. Without loss of generality, each search begins at level $\ell = 0$ with the set $X_0 = \{(0,0)\}$. Then at level ℓ, a set of *choices* $C_\ell \subseteq (\mathbb{Z}_p \times \mathbb{Z}_p)\backslash X_{\ell-1}$, for the next point to be added to $X_{\ell-1}$, is computed. For each point $(x,y) \in C_\ell$, $X_\ell \leftarrow X_{\ell-1} \cup \{(x,y)\}$, and the search moves to level $\ell+1$.

The values $\mathsf{SC}(p,w)$ with $w > N^*(p)$ in Table 1 were found via non-exhaustive search. At level ℓ, for each point $(x,y) \notin X_{\ell-1}$, $|\Delta(X_{\ell-1} \cup \{(x,y)\})|$ is computed, and the maximum such number is stored. Then C_ℓ is defined to be the set of all points which provide this maximum coverage.

The $\mathsf{SC}(p,w)$ with $w = N^*(p)$ were found via backtrack search [4]. For each p, we searched for an optimal cover, S. Denote by E the number of "Extra" quotient differences occurring in S, that is, the number of repeated quotient differences plus the number of undefined quotient differences. Clearly $E = \binom{w}{2} - p$ for any cover, S. At level ℓ, for each point $(x,y) \notin X_{\ell-1}$, the number of extra quotient differences in $X_{\ell-1} \cup \{(x,y)\}$ is counted. If this number is at most E, then $(x,y) \in C_\ell$.

Table 1 compares the minimum possible size of a cover $(N^*(p))$ with the size of the cover found by computer search (w); the smallest size of a sum cover in a cyclic group $(N_{\text{sum}}(p))$, found by a computer search similar to the one used to find slope covers; and the bound reported in [1] $(N^*_{\text{sum}}(p))$. All the covers have size less than $2\lceil\sqrt{p}\rceil$.

The optimal packings in Table 2 were found using a backtrack search. The non-optimal packings were found via non-exhaustive search. At level ℓ, for each point $(x,y) \notin X_{\ell-1}$, if $X_{\ell-1} \cup \{(x,y)\}$ is a packing which contains no undefined difference quotients, then $(x,y) \in C_\ell$.

Table 2 compares the maximum possible size of a packing $(D^*(p))$ with the size of the packing found by computer search (w); the maximum size of a weak Sidon set reported in [3] $(D_{\text{sum}}(p))$; and the maximum possible size of a weak Sidon set $(D^*_{\text{sum}}(p))$.

4 Generic Algorithms for the Discrete Logarithm Problem

In this section, we investigate some interesting connections between slope packings and coverings and generic algorithms for the $\mathsf{Discrete\ Logarithm}$ problem defined in a finite abelian group $(G, \cdot)$. Let $\alpha \in G$ have order n, and define

$$\langle\alpha\rangle = \{\alpha^i : 0 \le i \le n-1\}.$$

$\langle\alpha\rangle$ is a cyclic subgroup of G.

Problem 1:	Discrete Logarithm
Instance:	A finite abelian group $(G,\cdot)$, an element $\alpha \in G$ having order n, and an element $\beta \in \langle\alpha\rangle$.
Question:	Find the unique integer a, $0 \le a \le n-1$, such that $\alpha^a = \beta$. We will denote this integer a by $\log_\alpha \beta$.

Odlyzko [6] is a recent survey on the Discrete Logarithm problem. We will be considering generic algorithms for the Discrete Logarithm problem. Roughly speaking, any algorithm that does not depend on the representation of the group G is called a *generic algorithm*. Nechaev and Shoup [5, 9] have proven that the complexity of a generic algorithm for the Discrete Logarithm problem in a (sub)group of order n is $\Omega(\sqrt{n})$. Schnorr [8] studies the complexity of generic algorithms when the unknown discrete logarithm is restricted to a subset (of a group) having specified cardinality. Teske [10] is survey of discrete logarithm algorithms having complexity $O(\sqrt{n})$.

We will restrict our attention to instances of the Discrete Logarithm problem in which α has prime order, say p. In this situation, we show the easy result that any $\mathsf{SC}(p,w)$ yields a deterministic generic algorithm. We will measure the (time) complexity of the algorithm by the number of exponentiations performed by the algorithm.

Theorem 4.1. *Suppose that $(G,\cdot)$ is a finite group, $\alpha \in G$ has prime order p, and $\beta \in \langle\alpha\rangle$. Suppose that $S \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$ is an $\mathsf{SC}(p,w)$. Then there exists a deterministic algorithm for the* Discrete Logarithm *problem in $\langle\alpha\rangle$ that has complexity $O(w)$.*

Table 1: $\mathsf{SC}(p, w)$.

p	$N_{\mathrm{sum}}(p)$	$N^*_{\mathrm{sum}}(p)$	w	$N^*(p)$	cover
2	3	3	3	3	$(0,0),(0,1),(1,0)$
3	3	3	3	3	$(0,0),(1,0),(2,1)$
5	4	4	4	4	$(0,0),(0,1),(1,0),(2,2)$
7	5	5	5	5	$(0,0),(0,1),(0,2),(1,0),(6,5)$
11	6	6	6	6	$(0,0),(0,1),(0,2),(1,0),(1,3),(5,9)$
13	6	6	6	6	$(0,0),(0,1),(1,0),(1,2),(2,8),(8,11)$
17	7	7	7	7	$(0,0),(0,1),(1,2),(2,6),(3,5),(1,9),(3,2)$
19	8	8	7	7	$(0,0),(0,1),(1,2),(2,6),(3,5),(17,8),(9,4)$
23	9	8	8	8	$(0,0),(0,1),(0,2),(1,0),(1,3),(2,9),(15,5),(21,10)$
29	9	9	9	9	$(0,0),(0,1),(0,2),(0,3),(1,0),(2,5),(13,25),(19,6),$ $(28,23)$
31	10	9	9	9	$(0,0),(0,1),(0,2),(1,0),(1,2),(5,15),(16,7),(22,8),$ $(29,18)$
37	11	10	10	10	$(0,0),(0,1),(0,2),(0,3),(1,0),(1,5),(13,9),(16,19),$ $(35,26),(36,20)$
41	11	11	10	10	$(0,0),(0,1),(0,2),(1,0),(1,6),(18,34),(21,9),(25,36),$ $(34,30),(36,37)$
43	12	11	10	10	$(0,0),(0,1),(1,0),(1,2),(2,13),(12,26),(14,27),(27,32),$ $(31,4),(37,28)$
47	12	11	11	11	$(0,0),(0,1),(0,2),(0,3),(1,0),(1,5),(16,38),(23,10),$ $(38,26),(41,45),(44,18)$
53	12	12	11	11	$(0,0),(1,1),(6,0),(14,2),(23,5),(39,15),(51,38),(48,29),$ $(52,21),(9,7),(8,32)$
59	13	12	12	12	$(0,0),(1,1),(36,0),(15,2),(54,5),(40,14),(52,49),(32,16),$ $(7,8),(50,23),(31,34),(27,31)$
61	13	13	12	12	$(0,0),(1,1),(34,0),(9,2),(46,5),(28,14),(50,42),(33,52),$ $(59,37),(17,7),(10,39),(39,13)$
67		13	13	13	$(0,0),(0,1),(1,0),(2,2),(3,5),(4,14),(5,18),(6,44),$ $(11,47),(14,55),(41,19),(15,66),(46,65)$
71		13	13	13	$(0,0),(1,1),(10,0),(22,2),(35,5),(54,14),(68,18),(63,54),$ $(6,49),(56,29),(50,64),(37,12),(59,36)$
73		14	13	13	$(0,0),(1,1),(59,0),(47,2),(36,5),(31,14),(24,21),(55,15),$ $(7,20),(28,10),(49,42),(22,9),(54,7)$
79		14	14	14	$(0,0),(1,1),(32,0),(66,2),(22,5),(63,14),(20,18),(21,66),$ $(60,56),(65,10),(77,40),(55,63),(73,37),(39,9)$
83		14	15	14	$(0,0),(1,1),(42,0),(3,2),(48,5),(16,14),(62,18),(34,31),$ $(10,48),(58,4),(24,33),(46,7),(6,1),(17,53),(37,63)$
89		15	15	14	$(0,0),(1,1),(73,0),(59,2),(46,5),(39,14),(27,18),(36,43),$ $(19,74),(75,87),(18,85),(24,12),(65,83),(41,74),(56,32)$
97		15	16	15	$(0,0),(1,1),(69,0),(43,2),(18,5),(96,14),(72,18),(57,31),$ $(44,46),(24,69),(61,58),(16,60),(20,10),(30,62),(75,48),(82,83)$

Table 2: $\mathsf{SP}(p, w)$.

p	$D_{\mathrm{sum}}(p)$	$D^*_{\mathrm{sum}}(p)$	w	$D^*(p)$	packing
2	2	2	2	2	$(0,0),(1,0)$
3	3	3	3	3	$(0,0),(1,0),(2,1)$
5	3	4	3	3	$(0,0),(1,0),(2,1)$
7	4	4	4	4	$(0,0),(1,0),(2,1),(3,4)$
11	5	5	5	5	$(0,0),(1,0),(2,1),(3,4),(8,6)$
13	5	5	5	5	$(0,0),(1,0),(2,1),(3,4),(4,10)$
17	5	5	6	6	$(0,0),(1,0),(2,1),(3,4),(4,14),(6,2)$
19	6	6	6	6	$(0,0),(1,0),(2,1),(3,4),(4,13),(12,7)$
23	6	6	7	7	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,17),(9,5)$
29	7	7	8	8	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,3),(12,19),(27,9)$
31	7	7	8	8	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,14),(12,8),(14,5)$
37	7	7	9	9	$(0,0),(1,0),(2,1),(3,4),(4,10),(14,32),(16,21),(24,36),(28,13)$
41	7	8	9	9	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,14),(7,40),(21,11),(23,35)$
43	8	8	9	9	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,3),(6,2),(23,40),(29,6)$
47	8	8	10	10	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,17),(9,39),(12,27),(19,16),(21,31)$
53	8	8	10	10	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,3),(6,28),(8,52),(12,27),(38,30)$
59	9	9	11	11	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,3),(10,36),(45,40),(53,14),(54,24),(56,15)$
61	9	9	11	11	$(0,0),(1,0),(2,1),(3,4),(4,10),(5,3),(6,59),(24,5),(26,42),(28,50),(58,54)$
67	9	9	11	12	$(0,0),(0,1),(1,2),(2,6),(3,5),(4,3),(5,17),(7,33),(38,55),(50,56),(54,10)$
71	9	10	11	12	$(0,2),(0,3),(1,0),(2,1),(3,4),(4,8),(5,15),(6,28),(8,68),(42,59),(58,70)$
73	10	10	12	12	$(0,0),(1,1),(17,0),(36,2),(56,5),(9,14),(37,25),(34,59),(55,27),(53,40),(24,33),(32,7)$
79	10	10	12	13	$(0,0),(1,1),(7,0),(16,2),(26,5),(42,14),(53,18),(76,34),(62,15),(57,33),(10,38),(39,46)$
83	10	10	12	13	$(0,0),(0,1),(1,2),(2,6),(3,5),(4,3),(5,17),(6,27),(7,44),(16,10),(65,16),(66,74)$
89	10	11	13	13	$(0,0),(1,1),(5,0),(12,2),(20,5),(34,14),(43,18),(73,43),(30,74),(24,87),(67,85),(75,12),(33,83)$
97	11	11	13	14	$(0,0),(1,1),(4,0),(10,2),(17,5),(30,14),(38,18),(55,31),(74,46),(20,69),(16,58),(94,60),(64,10)$

Proof. For each $(x, y) \in S$, compute $\gamma(x, y) = \alpha^y\beta^{-x}$. If possible, find $\gamma(x, y) = \gamma(x', y')$ where (x, y), $(x', y') \in S$ and $(x, y) \neq (x', y')$. If this happens, then $x \neq x'$, $\log_\alpha \beta = (y - y')(x - x')^{-1} \bmod p$ and the algorithm succeeds.

We prove that this must occur, as follows: denote $a = \log_\alpha \beta$. Since S is a SC, there exist (x, y), $(x', y') \in S$ such that $a = (y - y')(x - x')^{-1} \bmod p$. Clearly this implies that $x \neq x'$, and we are done.

Finally, observe that we compute $2w$ exponentiations in G, so the complexity of the algorithm is $O(w)$. □

Remark. If we apply Theorem 4.1 with the slope cover from Proposition 1.2, then we obtain the well known baby step-giant step algorithm of Shanks.

In the next theorem, we provide a small extension to Theorem 4.1, in which $\Delta(S)$ misses only one element of $\mathbb{Z}_p$.

Theorem 4.2. *Suppose that $(G, \cdot)$ is a finite group, $\alpha \in G$ has prime order p, and $\beta \in \langle\alpha\rangle$. Suppose that $S \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$ and $|\Delta(S) \cap \mathbb{Z}_p| = p - 1$. Then there exists a deterministic algorithm for the* Discrete Logarithm *problem in $\langle\alpha\rangle$ that has complexity $O(w)$.*

Proof. For each $(x, y) \in S$, compute $\gamma(x, y) = \alpha^y\beta^{-x}$. If possible, find $\gamma(x, y) = \gamma(x', y')$ where (x, y), $(x', y') \in S$ and $(x, y) \neq (x', y')$. If this happens, then $x \neq x'$, $\log_\alpha \beta = (y - y')(x - x')^{-1} \bmod p$ and the algorithm succeeds. (So far, this is the same as in the previous algorithm, which was presented in the proof of Theorem 4.1). If there do not exist (x, y), $(x', y') \in S$ such that $\gamma(x, y) = \gamma(x', y')$ and $(x, y) \neq (x', y')$, then it is easy to see that $\log_\alpha \beta = a_0$, where $\{a_0\} = \mathbb{Z}_p \backslash \Delta(S)$. □

We next consider Las Vegas type algorithms, which are defined as follows.

Definition 4.1. Let $\epsilon \in \mathbb{R}$, $0 < \epsilon \leq 1$. A *Las Vegas* algorithm having success probability ϵ for a certain problem is a randomized algorithm, say A, which, given any instance I of the problem, will correctly solve the instance I with probability at least ϵ. Furthermore, if A does not correctly solve the instance I, then it will return the output "no answer".

We now show how any $S \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$ gives rise to a Las Vegas algorithm for the Discrete Logarithm problem having a specified success probability. Our main result depends on the following simple lemma.

Lemma 4.3. *Suppose that p is prime and $S \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$. Let $h \in \mathbb{Z}_p$, and define*

$$m_h(S) = \{(x, y + hx \bmod p) : (x, y) \in S\}.$$

Then $\infty \in \Delta(m_h(S))$ if and only if $\infty \in \Delta(S)$; and for any $g \in \mathbb{Z}_p$, $g + h \in \Delta(m_h(S))$ if and only if $g \in \Delta(S)$.

Theorem 4.4. *Suppose that $(G, \cdot)$ is a finite group, $\alpha \in G$ has prime order p, and $\beta \in \langle\alpha\rangle$. Suppose that $S \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$, $|S| = w$. Then there exists a Las Vegas algorithm for the* Discrete Logarithm *problem in $\langle\alpha\rangle$ that has success probability*

$$\epsilon = \frac{|\Delta(S) \cap \mathbb{Z}_p|}{p}$$

and complexity $O(w)$.

Proof. First, choose $h \in \mathbb{Z}_p$ at random. Then, for each $(x, y) \in m_h(S)$, compute $\gamma(x, y) = \alpha^y\beta^{-x}$. If possible, find $\gamma(x, y) = \gamma(x', y')$ where (x, y), $(x', y') \in S$ and $(x, y) \neq (x', y')$. If this happens, then $x \neq x'$, $\log_\alpha \beta = (y - y')(x - x')^{-1} \bmod p$ and the algorithm succeeds.

We now compute the success probability of the algorithm. Denote $a = \log_\alpha \beta$. Clearly, the algorithm succeeds if and only if there exist (x, y), $(x', y') \in m_h(S)$ such that $a = (y - y')(x - x')^{-1} \bmod p$, i.e., if and only if $a \in \Delta(m_h(S))$. By Lemma 4.3, this happens if and only if $a - h \in \Delta(S)$. Since h is chosen at random, it is easily seen that the probability that $a - h \in \Delta(S)$ is exactly $|\Delta(S) \cap \mathbb{Z}_p|/p$.

Finally, observe that we compute $2w$ exponentiations in G, so the complexity of the algorithm is $O(w)$. □

Remarks. 1. The success probability of the algorithm constructed in the proof of Theorem 4.4 is maximized (as a function of $|S|$) precisely when S is a slope packing. In this situation, $\epsilon = \binom{w}{2}/p$.

2. Theorem 4.4 is true when $|\Delta(S) \cap \mathbb{Z}_p| = p - 1$, but it is better instead to use Theorem 4.2 in this situation.

We now consider converse results, which show that generic algorithms for the Discrete Logarithm problem imply the existence of sets S where lower bounds on $|\Delta(S) \cap \mathbb{Z}_p|$ can be proven. We begin by giving a precise description of what we mean by a generic algorithm. We consider a cyclic group or subgroup of prime order p, which is therefore isomorphic to $(\mathbb{Z}_p, +)$. We will study generic algorithms for the Discrete Logarithm problem in $(\mathbb{Z}_p, +)$. (The particular group that is used is irrelevant in the context of generic algorithms; the choice of $(\mathbb{Z}_p, +)$ is arbitrary.)

An *encoding* of $(\mathbb{Z}_p, +)$ is any injective mapping $\sigma : \mathbb{Z}_p \to X$, where X is a finite set. The encoding function specifies how group elements are represented. Any discrete logarithm problem in a (sub)group of cardinality p of an arbitrary group G can be specified by defining a suitable encoding function. As mentioned previously, a generic algorithm is one that works for any encoding. In particular, a generic algorithm must work correctly when the encoding function σ is a random injective function; for example, when $X = \mathbb{Z}_p$ and σ is a random permutation of $\mathbb{Z}_p$.

We suppose that we have a random encoding, σ, for the group $(\mathbb{Z}_p, +)$. In this group, the discrete logarithm of any element a to the base 1 is just a, of course. Given the encoding function σ, the encoding $\sigma(1)$ of the generator, and an encoding of an arbitrary group element $\sigma(a)$, a generic algorithm is trying to compute the value of a. In order to perform operations in this group when group elements are encoded using the function σ, we hypothesize the existence of an oracle (or subroutine) to perform this task.

Given encodings of two group elements, say $\sigma(i)$ and $\sigma(j)$, it must be possible to compute the encodings $\sigma((i+j) \bmod p)$ and $\sigma((i-j) \bmod p)$. This is necessary if we are going to add and subtract group elements, and we assume that our oracle will do this for us. By combining operations of the above type, it is possible to compute linear combinations of the form $\sigma((di \pm cj) \bmod p)$, where $c, d \in \mathbb{Z}_p$. However, using the fact that $-j \equiv p - j \pmod{p}$, we observe that we only need to be able to compute linear combinations of the form $\sigma((di + cj) \bmod p)$. We will assume that the oracle can directly compute linear combinations of this form in $O(1)$ time. (This is a slight extension of the model used by Shoup, and it is similar to the model used in Schnorr [8]. Note that we will obtain lower bounds similar to those of Nechaev and Shoup in this stronger model.)

Group operations of the type described above are the only ones allowed in a generic algorithm. That is, we assume that we have some method of performing group operations on encoded elements, but we cannot do any more than that. Now let us consider how a generic Las Vegas algorithm, say GenLog, can go about trying to compute a discrete logarithm. The input to the algorithm GenLog consists of $\sigma_1 = \sigma(1)$ and $\sigma_2 = \sigma(a)$, where $a \in \mathbb{Z}_p$. GenLog will be successful if and only if it outputs the value a.

GenLog will use the oracle to generate a sequence of w, say, encodings of linear combinations of 1 and a. The execution of GenLog can be specified by a list of ordered pairs $(c_i, d_i) \in \mathbb{Z}_p \times \mathbb{Z}_p$, $1 \leq i \leq w$. (We can assume that these w ordered pairs are distinct.) For each ordered pair (c_i, d_i), the oracle computes the encoding $\sigma_i = \sigma((d_i + c_i a) \bmod p)$. Note that we can define $(c_1, d_1) = (0, 1)$ and $(c_2, d_2) = (1, 0)$ so our notation is consistent with the input to the algorithm.

In this way, the algorithm GenLog obtains a list of encoded group elements, $(\sigma_1, \ldots, \sigma_w)$. Because the encoding function σ is injective, it follows immediately that $d_i + c_i a \equiv d_j + c_j a \pmod{p}$ if and only if $\sigma_i = \sigma_j$. This provides a method to possibly compute the unknown value a: Suppose that $\sigma_i = \sigma_j$ for two integers $i \neq j$. If $c_i = c_j$, then $d_i = d_j$ and the two ordered pairs (c_i, d_i) and (c_j, d_j) are the same. Since we are assuming the ordered pairs are distinct, it follows that $c_i \neq c_j$. Because p is prime, we can compute a as follows:

$$a = (d_i - d_j)(c_j - c_i)^{-1} \bmod p.$$

Suppose first that the algorithm GenLog chooses a set

$$S = \{(c_i, d_i) : 1 \leq i \leq w\} \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$$

of w distinct ordered pairs all at once, at the beginning of the algorithm. Such an algorithm is called a *non-adaptive algorithm*. Then the list of w corresponding encodings is obtained from the oracle. Define $\mathsf{Good}(S)$ to consist of all elements $a \in \mathbb{Z}_p$ that are the solution of an equation $a = (d_i - d_j)(c_j - c_i)^{-1} \bmod p$ with $i \neq j$, $i, j \in \{1, \ldots, w\}$. Observe that

$$\mathsf{Good}(S) = \{-x \bmod p : x \in \Delta(S) \cap \mathbb{Z}_p\}.$$

By what we have said above, we know that the value of a can be computed by GENLOG if $a \in \mathsf{Good}(S)$. If $|\mathsf{Good}(S)| = p-1$, then it must be the case that a is the unique element in $\mathbb{Z}_p \backslash \mathsf{Good}(S)$ (as in the proof of Theorem 4.2). If $|\mathsf{Good}(S)| \leq p-2$, then GENLOG must return "no answer", since it cannot determine a unique value for a given the information that it has. Then the success probability of GENLOG is $|\mathsf{Good}(S)|/p = |\Delta(S) \cap \mathbb{Z}_p|/p$.

A generic discrete logarithm algorithm is not required to choose all the ordered pairs in S at the beginning of the algorithm, of course. It can choose later pairs after seeing what encodings of previous linear combinations look like (i.e., we allow the algorithm to be an *adaptive algorithm*). However, we will show, using an inductive argument, that this does not improve the success probabilty of the algorithm.

Let GENLOG be an adaptive generic algorithm for the discrete logarithm problem. A random encoding σ can be fixed. The problem instance is specified by the encodings $\sigma_1 = \sigma(1)$ and $\sigma_2 = \sigma(a)$. For $i = 1, 2, \ldots$, let S_i consist of the first i ordered pairs in a possible execution of GENLOG, for which the oracle computes the corresponding encodings $\sigma_1, \ldots, \sigma_i$. The set S_i and the list $\sigma_1, \ldots, \sigma_i$ represent all the information available to GENLOG at time i of its execution.

We claim that the value of a can be computed at time i if $a \in \mathsf{Good}(S_i)$; and if $a \notin \mathsf{Good}(S_i)$, then the probability that a has any given value in the set $\mathbb{Z}_p \backslash \mathsf{Good}(S_i)$ is exactly $1/(p - |\mathsf{Good}(S_i)|)$. (This probability is the conditional probability of a value of a at time i, assuming that a was chosen randomly from $\mathbb{Z}_p$.)

This claim is true for $i = 1$ (which forms the base case for an inductive proof) because $S_1 = \{(0,1)\}$ by definition, and $\mathsf{Good}(S_1) = \emptyset$. If the claim is true for $i = j-1$, then it is easy to see that it is also true for $i = j$. Then the claimed result follows by induction.

We can now prove the following result.

Theorem 4.5. *Suppose that* GENLOG *is any generic Las Vegas algorithm for the* Discrete Logarithm *problem in a (sub)group of order p, where p is prime. Let ϵ denote the success probability of* GENLOG. *Then there exists a set $S \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$ such that*

$$|\Delta(S) \cap \mathbb{Z}_p| \geq \begin{cases} p-1 & \text{if } \epsilon > \frac{p-2}{p} \\ \lceil \epsilon p \rceil & \text{if } \epsilon \leq \frac{p-2}{p}. \end{cases}$$

Proof. Let $\mathcal{S}$ denote the set of all maximal sets S that could be constructed during the execution of GENLOG, given a fixed random encoding σ. For each $S \in \mathcal{S}$, let $\Pr(S)$ denote the probability that S is the set of ordered pairs generated by GENLOG (this probability is computed over all possible values of the discrete logarithm a, and over all possible random choices made by GENLOG). Let $q(S)$ denote the conditional probability that GENLOG is successful, given that S is the set of ordered pairs it generates. By the discussion above, we have that

$$|\Delta(S) \cap \mathbb{Z}_p| \leq p-2 \quad \Rightarrow \quad q(S) \leq \frac{|\Delta(S) \cap \mathbb{Z}_p|}{p}. \tag{13}$$

Also, because the probability that GENLOG is successful is at least ϵ, we have that

$$\epsilon \leq \sum_{S \in \mathcal{S}} q(S)\Pr(S). \tag{14}$$

It follows immediately from (14) that there exists a set $S \in \mathcal{S}$ such that $q(S) \geq \epsilon$. First, suppose that $\epsilon > (p-2)/p$. Then $q(S) > (p-2)/p$ and it follows from (13) that $|\Delta(S) \cap \mathbb{Z}_p| \geq p-1$.

Now suppose that $\epsilon \leq (p-2)/p$. If $|\Delta(S) \cap \mathbb{Z}_p| \geq p-1$, then we are done, so assume $|\Delta(S) \cap \mathbb{Z}_p| \leq p-2$. In this case, (13) establishes that $q(S) \leq |\Delta(S) \cap \mathbb{Z}_p|/p$, so we have that $|\Delta(S) \cap \mathbb{Z}_p| \geq \epsilon p$. Since $|\Delta(S) \cap \mathbb{Z}_p|$ is an integer, we have that $|\Delta(S) \cap \mathbb{Z}_p| \geq \lceil \epsilon p \rceil$. □

We obtain the Nechaev-Shoup lower bound [5, 9] for the generic Discrete Logarithm problem as an immediate corollary of the previous theorem.

Corollary 4.6. *Suppose that* GENLOG *is any generic algorithm that correctly solves any instance of the* Discrete Logarithm *problem in a (sub)group of order p, where p is prime. Then the complexity of* GENLOG *is $\Omega(\sqrt{p})$.*

Proof. Set $\epsilon = 1$ and apply Theorem 4.5, observing that $|\Delta(S) \cap \mathbb{Z}_p| \leq \binom{|S|}{2}$. □

5 Comments and Summary

The construction of slope coverings and packings is an interesting problem in finite geometry that has connections with generic algorithms for the Discrete Logarithm problem. Slope packings also arise naturally in the context of the location-correcting codes studied by Roth and Seroussi in [7]. We define these now: A code over $\mathbb{Z}_p$ is a *t-location-correcting code* if up to t errors can be corrected whenever their values (but not their locations) are known. A 2-LCC$[n,k]$ is a k-dimensional t-location-correcting code of length n.

Suppose $S = \{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$ is an $\mathsf{SP}(p,n)$. Define a $2 \times n$ matrix, H, as follows:

$$H = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{pmatrix}.$$

Using the argument from Example 1 of [7], it is easy to show that H is the parity-check matrix of a (linear) 2-LCC$[n,n-2]$ over $\mathbb{Z}_p$. Hence, any slope packing gives rise to a 2-LCC.

Conversely, a linear 2-LCC$[n,n-2]$ over $\mathbb{Z}_p$ (p prime), defined by a parity-check matrix H of the above form, yields an $\mathsf{SP}(p,n)$ provided that the values $y_1,\ldots,y_n$ are all distinct. Suppose $q \neq 2,5,9,27$ is a prime power. It is proven in [7] that the length n of a 2-LCC$[n,n-2]$ over $\mathbb{F}_q$ satisfies the inequality $n(n-1) \leq 2q$. For an odd prime $p \neq 5$, it is possible to show that any linear 2-LCC$[n,n-2]$ can be transformed into one in which the parity-check matrix H has the property that the values $y_1,\ldots,y_n$ are distinct. Hence, a linear 2-LCC$[n,n-2]$ over $\mathbb{Z}_p$ (p prime, $p \neq 5$) implies the existence of an $\mathsf{SP}(p,n)$.

Computational evidence suggests that slope coverings and packings may exist that are "close to" optimal, and better than the ones obtainable from Sidin sets and sum covers. However, no construction seems to be known that will provide a proof of these empirical observations. This is an open problem worthy of further study.

Aside from its intrinsic mathematical interest, the construction of near-optimal slope packings and/or coverings has potential application to practical algorithms for the Discrete Logarithm problem. Suppose it were possible to give an efficient uniform construction of near-optimal slope packings or coverings. That is, suppose there exists an efficient algorithm which, when given any prime p as input, constructs a near-optimal slope packing or covering in the Desarguesian affine plane of order p. Then this could possibly yield a more efficent generic algorithm for the Discrete Logarithm problem than is currently known. However, the existence of the Shanks baby-step giant-step algorithm, together with the Nechaev-Shoup lower bound, shows that the potential improvement is limited to a constant-factor speed-up.

Acknowledgements

D.R. Stinson's research was supported by NSERC grants IRC #216431-96 and RGPIN #203114-98.

The authors would like to thank the following people for comments and for providing pointers to relevant literature: Aart Blokhuis, Charlie Colbourn, James Hirschfeld and Leo Storme.

References

[1] A. Blokhuis, H.A. Wilbrink and A. Sali. Perfect sumsets in finite abelian groups, *Linear Algebra and its Applications* **226–228** (1995), 47–56.

[2] A.E. Brouwer, J.B. Shearer, N.J.A. Sloane and W.D. Smith. A new table of constant weight codes, *IEEE Trans. Inform. Theory* **36** (1990), 1334–1380.

[3] H. Haanpää, A. Huima and P.R.J. Östergård. Sets in $\mathbb{Z}_n$ with distinct sums of pairs, Preprint.

[4] D.L. Kreher and D.R. Stinson. *Combinatorial Algorithms: Generation, Enumeration and Search*, CRC Press, 1999.

[5] V.I. Nechaev. On the complexity of a deterministic algorithm for a discrete logarithm, *Math. Notes* **55** (1994), 165–172.

[6] A. Odlyzko. Discrete logarithms: the past and the future, *Designs, Codes and Cryptogrpahy* **19** (2000), 129–145.

[7] R.M. Roth and G. Seroussi. Location-correcting codes, *IEEE Trans. Inform. Theory* **42** (1996), 554–565.

[8] C.P. Schnorr. Small generic hardcore subsets for the discrete logarithm problem: short secret DL-keys, *Inform. Process. Letters* **79** (2001), 93–98.

[9] V. Shoup. Lower bounds for discrete logarithms and related problems, *Lecture Notes in Computer Science* **1223** (1997), 256–266 (EUROCRYPT '97 Proceedings).

[10] E. Teske. Square-root algorithms for the discrete logarithm problem (a survey). In *Public Key Cryptography and Computational Number Theory*, Walter de Gruyter, 2001, pp. 283–301.