May, 2008

An Improved Multi-Set Algorithm for the Dense Subset Sum Problem

Andrew Shallue, *Illinois Wesleyan University*

An Improved Multi-Set Algorithm for the Dense Subset Sum Problem

Andrew Shallue

University of Calgary
Calgary AB T2N 1N4 Canada
ashallue@math.ucalgary.ca

Abstract. Given sets $L_1, \ldots, L_k$ of elements from $\mathbb{Z}/m\mathbb{Z}$, the k-set birthday problem is to find an element from each list such that their sum is 0 modulo m. We give a new analysis of the algorithm in [16], proving that it returns a solution with high probability. By the work of Lyubashevsky [10], we get as an immediate corollary an improved algorithm for the random modular subset sum problem. Assuming the modulus $m = 2^{n^\epsilon}$ for $\epsilon < 1$, this problem is now solvable using time and space

$$\widetilde{O}(2^{\frac{n^\epsilon}{(1-\epsilon)\log n}}).$$

Let $a_1, a_2, \ldots, a_n, t \in \mathbb{Z}/m\mathbb{Z}$ be given. The modular subset sum problem is to find a subset of the a_i that sum to t in $\mathbb{Z}/m\mathbb{Z}$, i.e. to find $x_i \in \{0, 1\}$ such that

$$\sum_{i=1}^{n} a_i x_i = t \mod m \ . \tag{1}$$

The corresponding decision problem is to determine whether or not there exists $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ that satisfies (1).

A subset sum problem is called random if n, m, and t are all fixed parameters but the a_i are drawn uniformly at random from $\mathbb{Z}/m\mathbb{Z}$. In addition to being interesting in their own right, random subset sum problems accurately model problems that arise naturally in number theory and combinatorics. We will use the shorthand MSS for modular subset sum and RMSS for random modular subset sum.

A useful way of classifying subset sum problems is by density.

Definition 1. *The* density *of a MSS instance is* $\frac{n}{\log_2 m}$. *Problems with density less than one are called* sparse, *while those with density greater than one are called* dense.

Now let sets $L_1, \ldots, L_k$ of elements of $\mathbb{Z}/m\mathbb{Z}$ be given. The k-set birthday problem is to find $b_i \in L_i$ such that $b_1 + \cdots + b_k = 0 \mod m$. We will assume that the elements of the L_i are uniformly generated and independent.

In this paper all logarithms will have base 2. Since the main algorithm has exponential complexity, we will often use "Soft-Oh" notation (see [5] for a definition) to highlight the main term and will assume "grade-school" arithmetic for simplicity.

This work was part of the author's dissertation research. Contact the author for further details or for proofs omitted from this paper.

Thanks to Eric Bach, Matt Darnall, Tom Kurtz, and Dieter van Melkebeek for thoughtful discussions that proved essential, to NSF award CCF-8635355 and the William F. Vilas Trust Estate for monetary support, and to the referees for helpful comments.

1 Previous Work and Results

The subset sum problem is of great practical and theoretical interest. Its decision version was proven NP-complete by R. Karp in his seminal 1972 paper on reductions among combinatorial problems [8]. It has seen application in the creation of public key cryptosystems [3], and is a vital tool in discovering Carmichael numbers [6].

Trivial algorithms for MSS include brute force enumeration at $O(2^n)$ time and constant space, basic time-space tradeoff at $O(2^{n/2})$ time and space, and dynamic programming at $O(n \cdot m)$ time and space. Schroeppel and Shamir [14] were first to discover a nontrivial method for solving the subset sum problem. Their algorithm takes time $O(2^{n/2})$ and space $O(2^{n/4})$, using a technique of decomposition that is reflected in this paper.

Despite its status as an NP-complete problem, many cases are quite tractable. If m is polynomial in n (giving a very dense instance), the problem is solvable in polynomial time using dynamic programming. More sophisticated methods can improve the running time, for example [1] achieved a running time of $O(n^{7/4}/\log^{3/4} n)$ for instances with $m \log m = \Theta(n^2)$. In [4], the range of problems solvable in polynomial time was extended to cases with $m = 2^{O(\log n)^2}$.

For sparse instances, the current favored technique is that of lattice basis reduction. If we have density $d < 0.64$, then Lagarias and Odlyzko [9] proved that almost all (as n goes to infinity) subset sum problems reduce to the shortest vector problem in polynomial time. The density bound was improved to 0.98 in [2]. Note that this work was on the integer subset sum problem, and so the definition of density used was different from the one in this paper. It was also proven in [9] that if $m = \Omega(2^{n^2})$, almost all subset sum problems are solvable in polynomial time using lattice basis reduction.

The inspiration for the present paper is the work of Lyubashevsky [10], who gave a rigorous analysis of Wagner's algorithm [16] for the k-set birthday problem over $\mathbb{Z}/m\mathbb{Z}$, proving that a solution is output with high probability. However, in order to preserve the independence and uniformity of set elements at all levels of the algorithm, the complexity of the algorithm was weakened to $\widetilde{O}(km^{2/\log k})$ time and space from Wagner's proposed complexity of $\widetilde{O}(km^{1/\log k})$ time and

space. Lyubashevsky also leveraged the k-set birthday problem into a new algorithm for the random subset sum problem. This algorithm uses $\widetilde{O}(km^{2/\log k})$ time and space, though by assuming $m = 2^{n^\epsilon}$, $\epsilon < 1$ and choosing $k = \frac{1}{2}n^{1-\epsilon}$ this becomes $\widetilde{O}(2^{\frac{2n^\epsilon}{(1-\epsilon)\log n}})$.

This paper extends this research by providing a rigorous analysis of Wagner's original algorithm.

Theorem 2. *Let sets $L_1, \ldots, L_k$ each contain $\alpha m^{1/\log k}$ independent and uniformly generated elements from $\mathbb{Z}/m\mathbb{Z}$. We make the technical assumptions that $\alpha > \max\{1024, k\}$ and that $\log m > 7(\log \alpha)(\log k)$. Then Wagner's algorithm for the k-set birthday problem has complexity $\widetilde{O}(k\alpha \cdot m^{1/\log k})$ time and space and finds a solution with probability greater than $1 - m^{1/\log k}e^{-\Omega(\alpha)}$.*

The most novel part of this result is that an exponentially small failure probability is achieved despite the fact that the elements at higher levels of the algorithm are neither independent nor uniform. The key tool that makes this possible is the theory of martingales.

This theorem has profound implications for cryptographic applications that use Wagner's k-set birthday algorithm. Though this analysis has only been done for the case of $\mathbb{Z}/m\mathbb{Z}$, it is anticipated that the techniques developed will work for other algebraic objects where the k-set birthday algorithm is applied, most notably the case of bit strings with the bitwise exclusive-or operation. This will provide justification for the use of the k-set birthday algorithm in cryptography.

Following [10], we get the following new result for RMSS as a corollary. This gives the fastest known algorithm for dense problems of asymptotic density smaller than $n/(\log n)^2$.

Theorem 3. *Let $m = 2^{n^\epsilon}$, $\epsilon < 1$, and assume that $n^\epsilon = \Omega((\log n)^2)$. Then there is a randomized algorithm that runs using time and space $\widetilde{O}(2^{\frac{n^\epsilon}{(1-\epsilon)\log n}})$ and finds a solution to RMSS with probability greater than $1 - 2^{-\Omega(n^\epsilon)}$.*

Here the probability of success is over the random bits of the algorithm and also over the random choice of inputs.

Note that by choosing $n^\epsilon = O((\log n)^2)$ the running time becomes polynomial, though not as small of a polynomial as that in [4].

The algorithm works just as well on problems of large enough constant density. Let $m = 2^{cn/k}$ for $c < \log k/(\log k + 4)$, giving problems of density greater than $k(1 + \frac{4}{\log k})$. Then the randomized algorithm runs using time and space $\widetilde{O}(m^{1/\log k})$ and finds a solution to RMSS with probability greater than $1 - 2^{-\Omega(n)}$. The constant in the exponent of the success probability depends on c and on k in such a way that the probability of success increases with increasing density.

2 Outline

The outline of this paper is as follows. In Section 3 we present Wagner's algorithm for the k-set birthday problem. In Section 4 we discuss what probability

distribution the elements of the lists in the algorithm have, and show that it is close to uniform. We show that the elements are close to independent in Section 5 and then give our new analysis of the k-set birthday algorithm in Section 6. The final section applies the k-set brithday algorithm to the RMSS problem.

3 The k-set Birthday Algorithm

We next provide a description of Wagner's k-set algorithm from [16]. A key subroutine is Algorithm ListMerge, which takes as input two lists L_1 and L_2 of integers in the interval $[-\frac{R}{2}, \frac{R}{2})$ and outputs elements $b + c \in [-\frac{Rp}{2}, \frac{Rp}{2})$ where $b \in L_1$, $c \in L_2$. Here $p < 1$ is a parameter set at the beginning. This subroutine is implemented by sorting L_1, L_2 and then for all $b \in L_1$, searching for $c \in L_2$ in the interval $[-b - \frac{Rp}{2}, -b + \frac{Rp}{2})$. Assuming that $|L_1| = |L_2| = N$, the cost of sorting is $O(N \log N)$ and the cost of N searches is $O(N \log N)$, giving a resource usage of $O(N \log N)$ time and space.

Algorithm 1 (ListMerge).
Input: two lists L_1, L_2 of integers in the interval $[-\frac{R}{2}, \frac{R}{2})$, parameter $p < 1$
Output: list L_{12} of integers $b + c \in [-\frac{Rp}{2}, \frac{Rp}{2})$ where $b \in L_1$, $c \in L_2$

1. sort L_1, L_2
2. **for** $b \in L_1$ **do:**
3. pick random $c \in L_2$ from those in interval $[-b - \frac{Rp}{2}, -b + \frac{Rp}{2})$
4. $L_{12} \leftarrow L_{12} \cup \{b + c\}$
5. output L_{12}

Note that at most one $b + c$ is taken as output for each $b \in L_1$, so the output list again has at most N elements.

For the k-set birthday problem we will choose $p = m^{-1/\log k}$, and assume that the initial k sets are populated with α/p elements of $\mathbb{Z}/m\mathbb{Z}$ chosen uniformly and independently at random. Treating the elements of the lists as integers in the interval $[-\frac{m}{2}, \frac{m}{2})$, we apply ListMerge to pairs of lists in a binary tree fashion, so that after $\log k$ levels we are left with a single list of integers in the interval $[-\frac{mp^{\log k}}{2}, \frac{mp^{\log k}}{2}) = [-\frac{1}{2}, \frac{1}{2})$. Having kept track of how each element is composed of elements from level 0, we have solved the problem (assuming the final list is nonempty) since we have found $s_1, \ldots, s_k$ such that $s_1 + \cdots + s_k = 0 \mod m$.

The resource usage of the algorithm is dominated by that of ListMerge applied to $2k$ lists of size at most $\alpha/p = \alpha \cdot m^{1/\log k}$, and so the k-set birthday problem is solvable using $\widetilde{O}(k\alpha \cdot m^{1/\log k})$ time and space. This proves the complexity claim of Theorem 2. However, proving that the algorithm outputs a solution with reasonable probability is much more difficult. The elements of the lists at levels greater than 0 are not uniformly generated over $\mathbb{Z}/m\mathbb{Z}$, nor are they independent. In the sections that follow we will analyze the distributions that arise, finishing the proof of Theorem 2.

4 Symmetric Unimodal Distributions

We choose parameters $p = m^{-1/\log k}$ and $\alpha > \max\{1024, k\}$. We also make the weak technical assumption from Theorem 2 that $\log m > 7(\log \alpha)(\log k)$. In most of the lemmas that follow, simplification requires an assumption that p is small. Note that the condition on $\log m$ implies $p < \frac{1}{128\alpha k^5}$. This is sufficient for the results that follow, though each has a weaker condition on p if allowed.

Our strategy over the next several sections is to prove by induction that the output list of Algorithm 1 has at least α/p elements. At level 0, the initial lists have α/p elements which are independent and uniformly generated. Our inductive hypothesis is that at level $\lambda - 1$ the remaining lists have α/p elements which are close to uniform and close to independent (to be defined precisely later). In this section we analyze the distributions of the elements at level λ. For this we need the following definitions.

Definition 4. *Let independent random variables X and Y have distributions F and G with probability mass functions f and g.*

1. *The distribution F is* symmetric *about the origin if $f(-x) = f(x)$ for all x.*
2. *The distribution F is* unimodal *at a if f is nondecreasing on $(-\infty, a]$ and nonincreasing on $[a, \infty)$.*
3. *The* convolution *of F and G, denoted $F * G$, is defined by*

$$f * g(s) = \sum_x f(x)g(s - x)$$

where the sum is over the probability space (for ease of notation, this is extended to $(-\infty, \infty)$).

From now on we take symmetric, unimodal to mean symmetric and unimodal about the origin. We will also use symmetric, unimodal to refer to the corresponding mass function of a distribution. The following are standard facts from probability theory.

Proposition 5. *Let X and Y be independent random variables with discrete distributions F and G.*

1. *The random variable $S = X + Y$ has distribution $F * G$.*
2. *If X and Y are symmetric, $F * G$ is symmetric.*
3. *If X and Y are symmetric and unimodal, $F * G$ is unimodal.*

Proof. 1. and 2. follow directly from the definitions. The proof of 3. is more technical. For a continuous version that is nicely written, see [13]. □

Fix the following notation. At level λ of the algorithm (note that $\lambda < \log k$), we have lists L_1 and L_2 of integers in the interval $[-\frac{mp^\lambda}{2}, \frac{mp^\lambda}{2})$ with $|L_1| = |L_2| = N = \alpha/p$. Let b_i be the elements of L_1 and c_i the elements of L_2. Let I be the interval $[-\frac{mp^{\lambda+1}}{2}, \frac{mp^{\lambda+1}}{2})$ and I_b the interval $[-\frac{mp^{\lambda+1}}{2} - b, \frac{mp^{\lambda+1}}{2} - b)$ for $b \in L_1$.

Let D_λ be the distribution of the elements of L_1 and L_2 with probability mass function f_λ. The support of f_λ is $[-\frac{mp^\lambda}{2}, \frac{mp^\lambda}{2})$, so that in particular $f_{\log k}$ is supported on $[-\frac{1}{2}, \frac{1}{2})$. Since elements at level λ are sums of elements from level $\lambda - 1$ that fall in the restricted interval $[-\frac{mp^\lambda}{2}, \frac{mp^\lambda}{2})$, we see that D_λ is the convolution of two copies of $D_{\lambda-1}$ with the tails thrown out and the remainder normalized to make a new probability distribution. Symbolically this looks like

$$f_\lambda(x) = \frac{1}{\sum_{a=-mp^\lambda/2}^{mp^\lambda/2} f_{\lambda-1} * f_{\lambda-1}(a)} \cdot f_{\lambda-1} * f_{\lambda-1}(x)$$

where x ranges over the support of f_λ. Here summing over an interval will always mean summing over the integers in the interval.

Elements from different lists are independent, so we conclude from Proposition 5 that at all levels, the distributions D_λ are symmetric unimodal.

A very surprising and useful fact is that f_λ is always close to a uniform distribution . The following lemma supports this claim by bounding the largest difference between f_λ and the uniform distribution on the support of f_λ. Intuitively, $6^\lambda p$ is small since $\lambda < \log k$ and p is small. Note that $\log m > 7(\log \alpha)(\log k)$ implies the required condition that $p \leq 1/(24k^3)$.

Lemma 6. *Let U be the uniform distribution on $[-\frac{mp^\lambda}{2}, \frac{mp^\lambda}{2})$, and assume that $p \leq 1/(24k^3)$. Then for all $x \in [-\frac{mp^\lambda}{2}, \frac{mp^\lambda}{2})$,*

$$|f_\lambda(x) - U(x)| \leq \frac{6^\lambda p}{mp^\lambda} \ .$$

Consider that if two uniform distributions are convolved the result is a triangle distribution. While far from uniform, if we only consider part of the distribution above a small interval centered at the origin the result is much closer to uniform. Carefully bounding the highest and lowest points while using induction on λ gives the proof, the details of which may be found in [15, Sect. 5.1].

The next result allows us to bound the expected number of elements in the output of Algorithm 1.

Proposition 7. *Assume notation for level λ, and that $p \leq 1/(2k^3)$. Let $b \in L_1$ be a random variable and let $\ell_i = b + c_i$ for $c_i \in L_2$. Assume that $|L_2| = \alpha/p$. Then the expected number of ℓ_i in I is at least $\alpha/8$.*

Proof. The work is in bounding $\Pr[\ell_i \in I] = \sum_b \Pr[b] \Pr[c_i \in I_b]$. Assume first that the level is not $\log k - 1$.

The number of integers in I_b is at least $\lfloor \frac{mp^{\lambda+1}}{2} \rfloor$, this lower bound corresponding to the case when $b = \pm\frac{mp^\lambda}{2}$. Using Lemma 6, we conclude that

$$\Pr[c_i \in I_b] \geq \frac{1 - 6^\lambda p}{mp^\lambda} \left(\frac{mp^{\lambda+1}}{2} - 1 \right) \geq \frac{1}{2mp^\lambda} \left(\frac{mp^{\lambda+1}}{2} - 1 \right) \tag{2}$$

$$= \frac{p}{4} - \frac{1}{2mp^\lambda} \geq \frac{p}{8} \tag{3}$$

where for (2), $p \leq 1/(2k^3) \leq 1/(2 \cdot 6^\lambda)$ by assumption and for (3) we assume $mp^{\lambda+1} \geq 4$ (satisfied since $\lambda + 1 < \log k$).

We conclude that $\Pr[\ell_i \in I] \geq p/8$ for all i and hence that the expected number of ℓ_i in I is at least $\alpha/8$.

If the level is $\log k - 1$, then $mp^\lambda = 1/p$ and $mp^{\lambda+1} = 1$. Thus I_b contains exactly one integer unless $b = \pm mp^\lambda/2$. Since the distribution is symmetric unimodal, these values of b are the least likely and so

$$\Pr[\ell_i \in I] \geq \sum_{b \neq \pm 1/(2p)} \Pr[b] \left(\frac{1 - 6^\lambda p}{mp^\lambda} \right) \geq \frac{1}{2} \cdot \frac{p}{2}$$

again using the assumption that $p \leq 1/(2 \cdot 6^\lambda)$. □

We conclude from this lemma that if L_1 and L_2 have α/p elements, the output of Algorithm ListMerge is expected to again have at least α/p elements. Our task in the next two sections is to prove that the number of elements is close to the expected value with high probability.

5 Bounding Dependency

Since we have uniform bounds for most distributions, we often suppress the value a random variable takes in expressing a probability. For example, $\Pr[\ell]$ means the probability that a random variable ℓ takes some unspecified value in its interval of support.

In the last section we showed that the distributions which arise in the k-set birthday algorithm are close to uniform, which allowed us to bound the expected size of the output of Algorithm 1. In this section we analyze what dependencies arise among list elements.

The first observation is that they are not independent. Consider the following example using the notation for combining lists L_1 and L_2 at level λ, where X_i is a Bernoulli random variable taking value 1 if $\ell_i \in I$ and 0 otherwise. If $\ell_1 = b_1 + c_1$, $\ell_2 = b_2 + c_1$, $\ell_3 = b_1 + c_2$, and $\ell_4 = b_2 + c_2$, then $\ell_4 = \ell_2 + \ell_3 - \ell_1$. Thus the random variable X_4 is functionally dependent upon X_1, X_2, X_3. Avoiding similar examples is the inspiration for the following definition.

Definition 8. *Organize the elements of $L_1 + L_2$ at level λ into a table, where if $\ell = b + c$ it appears in the row corresponding to b and column corresponding to c. Then $\ell_1, \ldots, \ell_j$ are called* row distinct *if they each appear in a distinct row.*

To motivate the next lemma, suppose that the distributions of the elements of L_1 and L_2 (at level 0) are uniform over $\mathbb{Z}/m\mathbb{Z}$, and that sums are taken over $\mathbb{Z}/m\mathbb{Z}$. Then if ℓ_1 shares column c with ℓ_2,

$$\Pr[\ell_1, \ell_2] = \sum_{z \in \mathbb{Z}/m\mathbb{Z}} \Pr[c = z] \Pr[b_1 = \ell_1 - z] \Pr[b_2 = \ell_2 - z] = \frac{1}{m^2}$$

while if L_1 and L_2 share neither row nor column they are also independent.

This extends easily to larger numbers of ℓ_i, proving that in this simple situation row distinct implies independent. At higher levels the sums start dropping terms due to exceeding interval bounds. However, since we are interested in the dependence only among those ℓ_i in the restricted interval, the number of terms lost is small. Combining these ideas along with Lemma 6 and induction on λ yields the following technical result. The proof may be found in [15, Sect. 5.5].

Lemma 9. *Let the current level of the algorithm be λ, and let X' be the event $X_1 = 1 \wedge \cdots \wedge X_{r-1} = 1$. Assume that $\ell_1, \ldots, \ell_r$ are row distinct. Then*

$$\frac{(1-p)^{4^\lambda}(1-3\cdot 6^\lambda p)^{4^{\lambda-1}}}{(1+4\cdot 6^\lambda p)^{4^{\lambda-1}}} \le \frac{\Pr[\ell_1,\ldots,\ell_r \mid X_r = 1, X']}{\Pr[\ell_r \mid X_r = 1]\Pr[\ell_1,\ldots,\ell_{r-1} \mid X']} \quad \textit{and}$$

$$\frac{\Pr[\ell_1,\ldots,\ell_r \mid X_r = 1, X']}{\Pr[\ell_r \mid X_r = 1]\Pr[\ell_1,\ldots,\ell_{r-1} \mid X']} \le \frac{(1+4\cdot 6^\lambda p)^{4^{\lambda-1}}}{(1-p)^{4^\lambda}(1-3\cdot 6^\lambda p)^{4^{\lambda-1}}}$$

unless both numerator and denominator are 0.

Using power series and the assumption that $p \le \frac{1}{864k^5} \le \frac{1}{864\cdot 24^\lambda}$ gives conceptually simpler bounds of $1 \pm 2\cdot 24^\lambda p$. Also note that the distribution of b or c from level $\lambda+1$ is the same as that of ℓ_i given $X_i = 1$ from level λ, so we can rewrite the statement of Lemma 9 for level $\lambda+1$ as

$$1 - 2\cdot 24^\lambda p \le \frac{\Pr[c_1,\ldots,c_r]}{\Pr[c_1]\Pr[c_2,\ldots,c_r]} \le 1 + 2\cdot 24^\lambda p \ . \tag{4}$$

This bound on the dependence will allow us to prove in the next section that Algorithm 1 outputs $N = \frac{\alpha}{p}$ row distinct elements with high probability.

6 Correctness Proof

Recall our induction hypothesis that the lists at level $\lambda - 1$ have α/p elements (one from each row, making them row distinct), and that these elements are close to uniform in the sense of Lemma 6 and close to independent in the sense of Lemma 9. Lemmas 6 and 9 are true at all levels, so to finish the induction it is enough to prove that every row contains an element in the restricted interval I. In fact we apply a tail bound to show that the probability is low that the number of row elements in the restricted interval strays too far from the expected value of $\alpha/8$.

Recall our previous notation for level λ of the k-set birthday algorithm. We are interested in proving that at least one ℓ_j per row is in the restricted interval $I = [-\frac{mp^{\lambda+1}}{2}, \frac{mp^{\lambda+1}}{2})$. Towards this end we fix $b \in L_1$ and relabel indices so that $\ell_i = b + c_i$ for $1 \le i \le N$. Let $I_b = [-\frac{mp^{\lambda+1}}{2} - b, \frac{mp^{\lambda+1}}{2} - b)$, and redefine the random variable X_i to take value 1 if $c_i \in I_b$ and 0 otherwise.

The next set of notation follows a survey paper by McDiarmid [12, Sect. 3.2] that covers numerous concentration inequalities and their applications to problems in combinatorics and computer science.

Let $f(\boldsymbol{X})$ be a bounded real valued function on $X_1, \ldots, X_N$, which for our purposes will be $S = \sum_{i=1}^{N} X_i$. Let B denote the event that $X_i = x_i$ for $i = 1, \ldots, j-1$ where x_i is either 0 or 1. For $x = 0, 1$ let

$$K_j(x) = \mathrm{E}[f(\boldsymbol{X}) \mid B, X_j = x] - \mathrm{E}[f(\boldsymbol{X}) \mid B] \ .$$

Define $dev(x_1, \ldots, x_{j-1})$ to be $\sup\{|K_j(0)|, |K_j(1)|\}$, while $ran(x_1, \ldots, x_{j-1})$ is defined to be $|K_j(0) - K_j(1)|$.

Let the *sum of squared ranges* be $R^2(\boldsymbol{x}) = \sum_{j=1}^{N} ran(x_1, \ldots, x_{j-1})^2$ and let $\hat{r}^2$, the *maximum sum of squared ranges*, be the supremum of $R^2(\boldsymbol{x})$ over all choices of $\boldsymbol{x} = (x_1, \ldots, x_N)$. Let *maxdev* be the maximum of $dev(x_1, \ldots, x_{j-1})$ over all choices of j and all choices of x_i.

The context of all this notation is the theory of martingales. By the Doob construction, $Y_j = \mathrm{E}[f(\boldsymbol{X}) \mid X_1, \ldots, X_j]$, $1 \le j \le N$ forms a martingale sequence. The standard tail bound for martingales is the Azuma-Hoeffding theorem, but in our case this is not tight enough to be meaningful. The theorem we will use instead is the following martingale version of Bernstein's inequality, proven by McDiarmid [12, Sect. 3.2].

Theorem 10. *Let $X_1, \ldots, X_N$ be a family of random variables with X_i taking values in $\{0, 1\}$, and let f be a bounded real-valued function defined on $\{0, 1\}^N$. Let μ denote the mean of $f(\boldsymbol{X})$, let b denote the maximum deviation maxdev, and let $\hat{r}^2$ denote the maximum sum of squared ranges. Suppose that X_i takes two values with the smaller probability being $p < \frac{1}{2}$. Then for any $t \ge 0$,*

$$\Pr[|f(\boldsymbol{X}) - \mu| \ge t] \le 2 \exp\left(-\frac{t^2}{2p\hat{r}^2(1 + bt/(3p\hat{r}^2))}\right) \ .$$

In our application, μ is $\alpha/8$ and thus we choose $t = \alpha/16$. We will prove that $bt/(3p\hat{r}^2)$ is small, which means that $\hat{r}^2$ needs to be not much bigger than α/p (the value it would take if the X_i are independent) in order for the bound to be meaningful. The next lemma is crucial for finding good bounds on $\hat{r}^2$ and $maxdev$.

Lemma 11. *Use the notation for level λ, with X_i being the indicator event for $c_i \in I_b$. Assume that $c_1, \ldots, c_N$ are row distinct when treated as ℓ_i from level $\lambda - 1$, and independent if $\lambda = 0$. Then for any $i > j$,*

$$|\mathrm{E}[X_i \mid X_1, \ldots, X_j] - \mathrm{E}[X_i]| \le 4 \cdot 24^{\lambda} p^2$$

assuming that $p \le 1/(864k^5)$.

Proof. We will reduce the result to finding a uniform bound for $|\Pr[c_i \mid X_1, \ldots, X_j] - \Pr[c_i]|$. If the level λ is 0, then $c_1, \ldots, c_j$, c_i are all fully

independent and hence $|\Pr[c_i \mid X_1, \ldots, X_j] - \Pr[c_i]| = 0$. In the general case

$$\begin{aligned}\Pr[c_i \mid X_1, \ldots, X_j] &= \frac{\Pr[c_i \wedge X_1 \wedge \cdots \wedge X_j]}{\Pr[X_1 \wedge \cdots \wedge X_j]} \\ &= \frac{\sum_{d_1} \cdots \sum_{d_j} \Pr[c_i \wedge c_1 = d_1 \wedge \cdots \wedge c_j = d_j]}{\sum_{d_1} \cdots \sum_{d_j} \Pr[c_1 = d_1 \wedge \cdots \wedge c_j = d_j]} \\ &\leq (1 + 2 \cdot 24^{\lambda-1} p) \Pr[c_i]\end{aligned}$$

where each sum in the numerator and denominator ranges over $d \in I_b$ if $X = 1$ and $d \notin I_b$ if $X = 0$. For the last step we have used (4) to break off the $\Pr[c_i]$ term, after which the rest of the terms cancel. A similar argument using the other inequality in (4) gives a lower bound for $\Pr[c_i|X_1, \ldots, X_j]$ of $(1-2\cdot 24^{\lambda-1}p)\Pr[c_i]$.

We now have

$$\begin{aligned}|\Pr[c_i \mid X_1, \ldots, X_j] - \Pr[c_i]| &\leq |\Pr[c_i](1 \pm 2 \cdot 24^{\lambda} p) - \Pr[c_i]| \\ &= \Pr[c_i] \cdot 2 \cdot 24^{\lambda} p \\ &\leq 2 \cdot 24^{\lambda} p\Big(\frac{1}{mp^{\lambda}} + \frac{6^{\lambda} p}{mp^{\lambda}}\Big)\end{aligned}$$

using Lemma 6 to bound $\Pr[c_i]$, and conclude that

$$\begin{aligned}|\mathrm{E}[X_i \mid X_1, \ldots, X_j] - \mathrm{E}[X_i]| &\leq \sum_{c_i \in I_b} |\Pr[c_i \mid X_1, \ldots, X_j] - \Pr[c_i]| \\ &\leq mp^{\lambda+1} \cdot 2 \cdot 24^{\lambda} p\Big(\frac{1}{mp^{\lambda}} + \frac{6^{\lambda} p}{mp^{\lambda}}\Big) \\ &\leq 4 \cdot 24^{\lambda} p^2\end{aligned}$$

by our assumption on p. □

With ingredients in hand, we next present the correctness proof for Algorithm 1. Note that the result requires and preserves the property that each list contains a row distinct sublist. This is prevented from circular reasoning by the fact that list elements at level 0 are independent.

Lemma 12 (k-set ListMerge). *Use the notation for level λ. Let A be the following event: for every $b \in L_1$, there exists $c \in L_2$ such that $b+c \in [-\frac{mp^{\lambda+1}}{2}, \frac{mp^{\lambda+1}}{2})$. Then*

$$\Pr[A] \geq 1 - (\alpha/p)e^{-\alpha/1024}$$

assuming that $p \leq 1/(128\alpha k^5)$.

Proof. First consider one row of the table. Fix $b \in L_1$, and let X_i be indicator variables for $c_i \in I_b$, $1 \leq i \leq N$. Then by Proposition 7 we have $\mathrm{E}[X_i] \geq p/8$ and hence that $\mathrm{E}[S] \geq \alpha/8$.

Our main goal now is to find upper bounds for *maxdev* and $\hat{r}^2$.

Consider $dev(x_1, \ldots, x_{j-1})$ for any $j \leq N$, any choice of $x_1, \ldots, x_{j-1}$, and $x_j = 0$ or 1. Note that

$$\begin{aligned} |K_j(x_j)| &= |\mathrm{E}[S \mid X_1, \ldots, X_j] - \mathrm{E}[S \mid X_1, \ldots, X_{j-1}]| \\ &\leq \sum_{i=1}^{N} |\mathrm{E}[X_i \mid X_1, \ldots, X_j] - \mathrm{E}[X_i \mid X_1, \ldots, X_{j-1}]| \ . \end{aligned}$$

If $i < j$ the corresponding term is 0 since its value has already been fixed. If $i = j$ the term can be at most 1 since that is the range of X_i. If $i > j$ then we apply Lemma 11 to see that

$$\begin{aligned} &|\mathrm{E}[X_i \mid X_1, \ldots, X_j] - \mathrm{E}[X_i \mid X_1, \ldots, X_{j-1}]| \\ &= |\mathrm{E}[X_i \mid X_1, \ldots, X_j] - \mathrm{E}[X_i] + \mathrm{E}[X_i] - \mathrm{E}[X_i \mid X_1, \ldots, X_{j-1}]| \\ &\leq 8 \cdot 24^{\lambda} p^2 \end{aligned}$$

Thus *maxdev* is no more than $1 + \frac{\alpha}{p} \cdot 8 \cdot 24^{\lambda} p^2 = 1 + 8 \cdot 24^{\lambda} \alpha p$.

By definition

$$\begin{aligned} ran(x_1, \ldots, x_{j-1}) &= |K_j(0) - K_j(1)| \\ &= |\mathrm{E}[S \mid B, X_j = 1] - \mathrm{E}[S \mid B, X_j = 0]| \\ &\leq \sum_{i=1}^{N} |\mathrm{E}[X_i \mid B, X_j = 1] - \mathrm{E}[X_i \mid B, X_j = 0]| \ . \end{aligned}$$

Following the same reasoning as we did for *maxdev*, if $i < j$ the corresponding term is 0, if $i = j$ the corresponding term is at most 1 since X_i is an indicator variable, while if $i > j$ the term is at most $8 \cdot 24^{\lambda} p^2$ by Lemma 11.

So $ran(x_1, \ldots, x_{j-1})$ has a uniform upper bound of $1 + 8 \cdot 24^{\lambda} \alpha p$ and thus $\hat{r}^2 \leq \frac{\alpha}{p}(1 + 8 \cdot 24^{\lambda} \alpha p)^2 \leq \frac{\alpha}{p} + 32 \cdot 24^{\lambda} \alpha^2$, assuming that $p \leq \frac{1}{4\alpha 24^{\lambda}}$.

We now conclude from Theorem 10 that

$$\begin{aligned} \Pr\left[S \leq \frac{\alpha}{8} - \frac{\alpha}{16}\right] &\leq \Pr\left[S \leq \mu - \frac{\alpha}{16}\right] \\ &\leq \exp\left(-\frac{\alpha^2/256}{2(\alpha + 32 \cdot 24^{\lambda} \alpha^2 p) + \frac{2}{3}\frac{\alpha}{16}(1 + 8 \cdot 24^{\lambda} \alpha p)}\right) \\ &\leq \exp\left(-\frac{\alpha^2/256}{2\alpha + \frac{\alpha}{2} + \frac{2}{3}\alpha + \frac{\alpha}{2}}\right) \leq e^{-\alpha/1024} \end{aligned}$$

assuming that $p \leq 1/(128\alpha 24^{\lambda})$.

Finally, by using the union bound the probability that some row fails to have at least $\alpha/16$ elements fall in I_b is smaller than $(\alpha/p)e^{-\alpha/1024}$ and the bound on the probability of event A follows. □

The proof of Theorem 2 now follows quite easily.

Proof (Theorem 2).

Lemma 12 completed our proof by induction on the level that all lists have α/p elements with high probability. An application of Algorithm Listmerge on lists of size α/p will again result in a list of size α/p with probability at least $1-(\alpha/p)e^{-\alpha/1024}$.

The k-set birthday algorithm successfully finds a solution as long as this occurs for all $2k$ applications of Algorithm Listmerge. Thus the algorithm succeeds with probability at least

$$(1-(\alpha/p)e^{-\alpha/1024})^{2k} > 1-2k(\alpha/p)e^{-\alpha/1024} = 1-m^{1/\log k}e^{-\Omega(\alpha)} \ .$$

As for the complexity, it is dominated by storing, sorting, and searching through $2k$ lists of size $\alpha m^{1/\log k}$, giving a time and space bound of $\widetilde{O}(k\alpha \cdot m^{1/\log k})$. □

7 Application to RMSS

In this section we show how to use the multi-set birthday problem to solve dense instances of RMSS. This work can be found in [10] and [11]; we include it here for completeness.

Consider the following random variable $Z_{\boldsymbol{a}}$ taking values on $\mathbb{Z}/m\mathbb{Z}$, where $\boldsymbol{a} = (a_1, \ldots, a_n)$ with $a_i \in \mathbb{Z}/m\mathbb{Z}$. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be an n-bit vector, where each element is drawn uniformly and independently from $\{0,1\}$. Then we define

$$Z_{\boldsymbol{a}} := \sum_{i=1}^{n} x_i a_i \mod m \ . \tag{5}$$

In the case where $\boldsymbol{a}$ is fixed and understood from context (say where it is the input of an RMSS instance) we will suppress the $\boldsymbol{a}$ in the notation. Note that for fixed $\boldsymbol{a}$ and varying $\boldsymbol{x}$, the collection $\{Z_{\boldsymbol{a}}(\boldsymbol{x})\}$ is a collection of independent random variables .

Our goal is to show that the distribution of $Z_{\boldsymbol{a}}$ is close to uniform, thus it is vital that we formalize what we mean by "close."

Definition 13. *Let X and Y be random variables taking values in a probability space A. The* statistical distance *between X and Y, denoted $\Delta(X,Y)$, is*

$$\Delta(X,Y) = \frac{1}{2}\sum_{a\in A} |Pr[X=a] - Pr[Y=a]| \ .$$

The next proposition states that for most choices of $\boldsymbol{a}$, $Z_{\boldsymbol{a}}$ is exponentially close to uniform. The proof involves showing that $\{Z_{\boldsymbol{a}} : \{0,1\}^n \to \mathbb{Z}/m\mathbb{Z}\}_{\boldsymbol{a}\in(\mathbb{Z}/m\mathbb{Z})^n}$ is a universal (and hence almost universal) family of hash functions, and then applying the leftover hash lemma. Here we encode elements of $\mathbb{Z}/m\mathbb{Z}$ as bit strings of length cn, $c<1$.

Proposition 14 (Impagliazzo, Naor [7]). *Let $m = 2^{cn}$ with $c < 1$. Then the probability over all choices of input vector $\boldsymbol{a} = (a_1, \ldots, a_n)$ that $\Delta(Z_{\boldsymbol{a}}, U) < 2^{-\frac{(1-c)n}{4}}$ is greater than $1 - 2^{-\frac{(1-c)n}{4}}$.*

Definition 15. *We call the multiset $\boldsymbol{a} = (a_1, \ldots, a_n)$* well-distributed *if it is one of the good choices from Proposition 14, i.e.*

$$\Delta(Z_{\boldsymbol{a}}, U) < 2^{-\frac{(1-c)n}{4}} \ .$$

So if the a_i are chosen uniformly at random, we lose very little by assuming that $Z_{\boldsymbol{a}}$ is uniform. This is the only place we use the fact that our subset sum problem is random, and it is possible to apply the k-set algorithm to MSS instances with the additional assumption that $\boldsymbol{a}$ is well-distributed. This might be preferable if a constructive criterion could be found for $\boldsymbol{a}$ being well-distributed, but for now that remains an open problem. Note that a necessary condition for being well-distributed is that the a_i contain no common factor. If $\gcd(a_1, \ldots, a_n, m) \neq 1$ then $Z_{\boldsymbol{a}}$ is only nonzero on a subgroup of $\mathbb{Z}/m\mathbb{Z}$, and thus is far from uniform.

Just as in [10], our subset sum algorithm is as follows. Choose parameters k and α. Break up the a_i into k sets, and generate k lists where each list contains $\alpha m^{1/\log k}$ random subset sums of that portion of the a_i. Apply the k-set birthday algorithm to find a solution.

Proof (Theorem 3).

For the analysis we make parameter choices of $\alpha = n$ and $k = \frac{1}{2}n^{1-\epsilon}$. Our assumption that $n^\epsilon = \Omega((\log n)^2)$ satisfies the requirement of Theorem 2 that $\log m > 7(\log \alpha)(\log k)$.

The probability of success is greater than the probability that all k subsets of $\boldsymbol{a}$ are well-distributed, times the probability that the algorithm succeeds given that all subsets are well-distributed. By applying Proposition 14, the probability that one of the subsets is well-distributed is greater than $1 - 2^{-\frac{(1-c)n}{4k}}$, where $c = kn^{\epsilon-1}$ since $2^{n^\epsilon} = 2^{cn/k}$. Thus the probability that all are well-distributed is greater than

$$(1 - 2^{-\frac{(1-.5)n}{4k}})^k > 1 - \frac{1}{2}n^{1-\epsilon} \cdot 2^{-\frac{n^\epsilon}{4}} \geq 1 - 2^{-\Omega(n^\epsilon)} \ .$$

In addition, the distance between these distributions and uniform ones is less than $2^{-\frac{n^\epsilon}{4}}$.

Now, assume that all elements from all initial lists are drawn independently from uniform distributions. Then the probability that the k-set birthday algorithm succeeds is at least

$$1 - n^{1-\epsilon} \cdot n2^{\frac{n^\epsilon}{(1-\epsilon)\log n}} e^{-n/1024} \geq 1 - n^{2-\epsilon} 2^{-n(\frac{1}{1024} - \frac{1}{(1-\epsilon)n^{1-\epsilon}\log n})} \geq 1 - 2^{-\Omega(n)} \ .$$

Accounting for the fact that the elements of the initial lists are only close to uniform, the probability of the birthday algorithm succeeding is reduced by $2^{-\Omega(n^\epsilon)}$ (see [11]).

Thus the probability of success of the multi-set subset sum algorithm is greater than

$$\left(1 - 2^{-\Omega(n^\epsilon)}\right)\left(1 - 2^{-\Omega(n)} - 2^{-\Omega(n^\epsilon)}\right) \geq 1 - 2^{-\Omega(n^\epsilon)} \ .$$

The complexity is dominated by the complexity of the k-set birthday algorithm. Thus the algorithm takes $\widetilde{O}(k\alpha \cdot m^{1/\log k}) = \widetilde{O}(2^{\frac{n^\epsilon}{(1-\epsilon)\log n}})$ time and space. □

References

1. Chaimovich, M.: New algorithm for dense subset-sum problem. Astérisque 258 (1999) 363 – 373
2. Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: Improved low–density subset sum algorithms. Comput. Complexity 2(2) (1992) 111 – 128
3. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Information Theory IT-22(6) (1976) 644 – 654
4. Flaxman, A., Przydatek, B.: Solving medium-density subset sum problems in expected polynomial time. In: STACS 2005. LNCS vol. 3404, pp. 305 – 314. Springer, Berlin (2005)
5. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. second edn. Cambridge University Press, Cambridge (2003)
6. Howe, E.W.: Higher-order Carmichael numbers. Math. Comp. 69(232) (2000) 1711 – 1719
7. Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. J. of Cryptology 9(4) (1996) 199 – 216
8. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations pp. 85 – 103. Plenum Press, NY (1972)
9. Lagarias, J., Odlyzko, A.: Solving low-density subset sum problems. JACM: Journal of the ACM 32(1) (1985) 229 – 246
10. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: APPROX-RANDOM. LNCS vol. 3624, pp. 378–389. Springer (2005)
11. Lyubashevsky, V.: On random high density subset sums. Electronic Colloquium on Computational Complexity (ECCC) 12 (2005) http://eccc.hpi-web.de/eccc-reports/2005/TR05-007/index.html
12. McDiarmid, C.: Concentration. In: Probabilistic Methods for Algorithmic Discrete Mathematics. Algorithms Combin. vol. 16 pp. 195 – 248. Springer, Berlin (1998)
13. Purkayastha, S.: Simple proofs of two results on convolutions of unimodal distributions. Statist. Prob. Lett. 39(2) (1998) 97 – 100
14. Schroeppel, R., Shamir, A.: A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. SIAM J. Comput. 10(3) (1981) 456 – 464
15. Shallue, A.: Two Number-Theoretic Problems that Illustrate the Power and Limitations of Randomness. PhD thesis, University of Wisconsin–Madison (2007)
16. Wagner, D.: A generalized birthday problem (extended abstract). In: Advances in Cryptology – CRYPTO 2002. LNCS vol. 2442 pp. 288 – 303. Springer, Berlin (2002)