Large-Scale GNSS Spreading Code Optimization

Alan Yang[1], Tara Mina[2], Stephen Boyd[1], Grace Gao[1]
[1]*Stanford University*, [2]*Georgia Institute of Technology*

BIOGRAPHY

Alan Yang is a Ph.D. candidate in the Department of Electrical Engineering at Stanford University. He received his M.S. degree in Electrical and Computer Engineering in 2021 from the University of Illinois at Urbana-Champaign, where he also received his B.S. in 2018. His research interests include convex optimization, control, and applications.

Tara Mina is a research engineer at Georgia Tech in the Space Exploration Analysis Laboratory. She received her Ph.D. in the Electrical Engineering Department from Stanford University in 2023. Prior to that, she obtained her B.S. in Electrical Engineering from Iowa State University in 2017 and her M.S. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 2019. Her research interests include secure positioning, navigation, and timing (PNT), GNSS signal design, lunar PNT, and optical navigation for space exploration.

Stephen Boyd is the Samsung Professor of Engineering and Professor of Electrical Engineering at Stanford University. He received an AB degree in Mathematics, summa cum laude, from Harvard University in 1980, and a PhD in EECS from U. C. Berkeley in 1985. He is a fellow of IEEE, SIAM, INFORMS, and IFAC, a member of the US National Academy of Engineering, a foreign member of the Chinese Academy of Engineering, and a foreign member of the National Academy of Engineering of Korea.

Grace Gao is an associate professor in the Department of Aeronautics and Astronautics at Stanford University. Before joining Stanford University, she was an assistant professor at University of Illinois at Urbana-Champaign. She obtained her Ph.D. degree at Stanford University. Her research is on robust and secure positioning, navigation, and timing with applications to manned and unmanned aerial vehicles, autonomous driving cars, as well as space robotics.

ABSTRACT

We propose a bit-flip descent method for optimizing binary spreading codes with large family sizes and long lengths, addressing the challenges of large-scale code design in GNSS and emerging PNT applications. The method iteratively flips code bits to improve the codes' auto- and cross-correlation properties. In our proposed method, bits are selected by sampling a small set of candidate bits and choosing the one that offers the best improvement in performance. The method leverages the fact that incremental impact of a bit flip on the auto- and cross-correlation may be efficiently computed without recalculating the entire function. We apply this method to two code design problems modeled after the GPS L1 C/A and Galileo E1 codes, demonstrating rapid convergence to low-correlation codes. The proposed approach offers a powerful tool for developing spreading codes that meet the demanding requirements of modern and future satellite navigation systems.

I. INTRODUCTION

A global navigation satellite system (GNSS) comprises a constellation of satellites that transmit signals to Earth. Each satellite's signal contains a *spreading code*, which is combined with navigation data and modulated onto a carrier wave at a specific frequency for transmission. Because all signals are transmitted at the same frequency, the spreading codes are designed to have low autocorrelation and low cross-correlation. This enables the receiver to distinguish signals from individual satellites and accurately determine their timing (Misra and Enge, 2012).

Traditionally, pseudorandom binary sequences generated using linear shift registers have been employed as spreading codes. For instance, the Global Positioning System (GPS) utilizes Gold codes (Gold, 1967) and Weil codes (Legendre, 1808; Rushanan, 2007). While these codes exhibit favorable autocorrelation and cross-correlation properties, they are only available in specific code lengths. Additionally, they are suboptimal because they cannot be tailored to a given code family size, *i.e.*, the number of codes required for a particular application.

There has been recent significant interest in developing new spreading codes (Wallner et al., 2007; Winkel, 2011; Yang et al., 2023b), as well as codes for wireless communications (Jiang et al., 2020) and radar (Alaee-Kerahroodi et al., 2019) using optimization techniques. These methods enable the design of spreading codes with good autocorrelation and cross-correlation properties, tailored to specific applications and not restricted by predefined code lengths. For example, the European Union's Galileo constellation employs spreading codes designed using a genetic algorithm (Wallner et al., 2007, 2008). Additionally, the upcoming Navigation Technology Satellite-3 (NTS-3) will serve as a platform for testing future GPS technologies, marking the first time a GPS satellite signal can be fully reprogrammed while in orbit (Chapman et al., 2020; AFRL US Air Force, 2023). Moreover, emerging Positioning, Navigation, and Timing (PNT) applications, such as those in low Earth orbit (LEO) and Lunar PNT, stand to benefit from custom-designed spreading codes (Soualle et al., 2005; Reid et al., 2020; Israel and Gramling, 2023).

Spreading code design is a challenging combinatorial optimization problem. Although numerous heuristic methods have been proposed to identify codes with favorable correlation properties, few are well-suited for the large-scale code design challenges encountered in GNSS applications (Avila-Rodriguez et al., 2006; Alaee-Kerahroodi et al., 2019; Mina and Gao, 2022; Yang et al., 2023a). The number of required codes can be substantial, often in the hundreds, particularly for LEO PNT applications. Additionally, code lengths can be considerable, sometimes reaching tens of thousands, since longer codes generally offer better performance in low signal-to-noise ratio environments.

In this work, we propose a bit-flip descent method suitable for optimizing spreading codes with large code family sizes and long code lengths. In each iteration of the method, a single bit, or code entry, is selected and flipped if it improves the objective function. To ensure rapid convergence, the method samples a small set of candidate bits and selects the one that offers the greatest improvement in the objective. This approach is particularly effective for large-scale code design problems, since the impact of flipping a single bit on the objective function can be efficiently calculated without recomputing the entire objective function (Winkel, 2011). Moreover, bit-flip descent methods have been shown to perform as well as, if not better than, more sophisticated methods for code design problems (Yang et al., 2023b). The method is guaranteed to converge to a solution where no further improvement is possible through single-bit flips. For an example of this type of algorithm applied in a different context, see Angeris et al. (2021).

In our experiments, we use our method to perform code optimization for two code design problems. The first is modeled after the GPS L1 C/A codes, which consist of 63 length-1023 codes. The second is based on the Galileo E1 codes, which consist of 100 length-4092 codes. We demonstrate that our method converges to low-correlation codes quickly. Our implementation and experiments have been made available at `https://github.com/Stanford-NavLab/decor`.

The remainder of this paper is structured as follows: §II introduces the spreading code design problem. In §III, we present efficient methods for evaluating the impact of altering a single code entry on the correlation matrices and objective function. §IV details the proposed bit-flip descent method. §V provides numerical examples, and finally, we conclude in §VI.

II. SPREADING CODE DESIGN

A family of n spreading codes, each of length T, is represented by a binary matrix $X \in \{-1, 1\}^{n \times T}$. Here, $X_{i,t}$ is the (i, t)-th bit, or t-th element of the i-th spreading code, for $i = 1, \ldots, n$ and $t = 0, \ldots, T-1$. The i-th spreading code, denoted by X_i, corresponds to the i-th row of the matrix X.

1. Correlation matrices

The auto- and cross-correlation properties of the spreading code family X are described by T correlation matrices, $\Sigma_0, \ldots, \Sigma_{T-1}$, where each $\Sigma_t \in \mathbf{R}^{n \times n}$ represents the shift-t correlation matrix:

$$(\Sigma_t)_{i,j} = \frac{1}{T} \sum_{\tau=0}^{T-1} X_{i,\tau} X_{j,(\tau - t) \bmod T}, \quad t = 0, \ldots, T-1, \quad i, j = 1, \ldots, n. \tag{1}$$

The diagonal entries $(\Sigma_t)_{i,i}$ represent the shift-t autocorrelation of the i-th code, while the off-diagonal entries $(\Sigma_t)_{i,j}$ capture the shift-t cross-correlation between the i-th and j-th codes, for $i \neq j$. Note that for any X, the shift-zero autocorrelation values are always 1, *i.e.*, the diagonal entries satisfy $(\Sigma_0)_{i,i} = 1$ for all i.

2. Spreading code design problem

The goal of spreading code design is to find a code family X with small auto- and cross-correlation values. This may be formulated as an optimization problem in which we minimize the objective function

$$f(X) = \sum_{(t,i,j) \in \mathcal{I}} |(\Sigma_t)_{i,j}|^p, \tag{2}$$

where $p \geq 1$ is a parameter and

$$\mathcal{I} \subseteq \{0, \ldots, T-1\} \times \{1, \ldots, n\} \times \{1, \ldots, n\}$$

is an index set. Increasing p increases the penalty on large correlation values, which can reduce outliers with large magnitude, at the cost of a higher variance in correlation values.

Index set. Since the cross-correlation is symmetric, we restrict the index set $\mathcal{I}$ to the indices $j \geq i$. We also exclude the zero-shift autocorrelation values $(\Sigma_0)_{i,i}$ for $i = 1, \ldots, n$, since they do not depend on the value of X. Therefore, we take $\mathcal{I}$ to be

$$\mathcal{I} = \{(t, i, j) \mid j > i\} \cup \{(t, i, i) \mid t > 0\},$$

which contains $T(n^2 + n)/2 - n$ indices.

Computing the objective. The objective function (2) may be directly computed in $O(n^2 T \log T)$ FLOPS (floating-point operations) using the fast Fourier transform (FFT). Let $\mathcal{F}_T$ and $\mathcal{F}_T^{-1}$ denote the forward and inverse FFT operators, respectively. Then, we have

$$(\Sigma_t)_{i,j} = \mathcal{F}_T^{-1}(\mathcal{F}_T(X_i) \circ \mathcal{F}_T(X_j)^*)_t, \quad t = 0, \ldots, T-1, \quad i, j = 1, \ldots, n, \tag{3}$$

where $\circ$ denotes the element-wise product and $*$ denotes the complex conjugate. Since $\mathcal{F}_T$ and $\mathcal{F}_T^{-1}$ are computable in $O(T \log T)$ FLOPS using the fast Fourier transform (FFT), $\Sigma_0, \ldots, \Sigma_{T-1}$ may be computed in $O(n^2 T \log T)$ FLOPS. It then follows that the objective (4) may also be computed in $O(n^2 T \log T)$ FLOPS.

Spreading code design problem. The spreading code design problem is given by the optimization problem

$$\begin{array}{ll} \text{minimize} & f(X) \\ \text{subject to} & X_{i,t} \in \{-1, 1\}, \quad i = 1, \ldots, n, \quad t = 0, \ldots, T-1. \end{array} \tag{4}$$

The code design problem (4) is a combinatorial optimization problem that is difficult to solve in general. In §IV, we propose an efficient bit-flip descent method that can quickly generate good solutions to (4). First, we present methods for efficiently updating the values of the correlation matrices and objective function, when a single code entry is changed.

In the following section, we show how the correlation matrices may be updated in just $O(nT)$ FLOPS after flipping a single bit, without needing to fully recompute the correlation matrices using (3).

III. EFFICIENT COMPUTATION

In this section, we describe efficient methods for evaluating the effect of flipping the sign of any bit $X_{a,b}$ on the correlation matrices (Σ_t) and the objective function $f(X)$, for $a = 1, \ldots, n$ and $b = 0, \ldots, T-1$. These methods form the foundation of the bit-flip descent algorithm introduced in §IV.

1. Updating the correlation matrices

Suppose we have a code matrix X with correlation matrices (Σ_t), which have already been computed, *e.g.*, using (3). After flipping the bit $X_{a,b}$, the updated code matrix X' is

$$X'_{i,t} = \begin{cases} -X_{i,t} & i = a, t = b, \\ X_{i,t} & \text{else.} \end{cases} \tag{5}$$

Let (Σ'_t) be the correlation matrices of the updated code X'. Although (Σ'_t) may be computed in $O(n^2 T \log T)$ FLOPS using (3), we show how it may be computed in just $O(nT)$ FLOPS by directly updating (Σ_t).

When $X_{a,b}$ is flipped, at most one term in the sum (1) changes. Therefore, the updated correlation matrices (Σ'_t) may be computed from (Σ_t) using the update rule $\Sigma'_t = g(\Sigma_t, a, b)$, where

$$g(\Sigma_t, a, b)_{i,j} = \begin{cases} (\Sigma_t)_{i,j} - (2/T) X_{a,b} X_{j,(b+t) \bmod T} & i = a,\ j < a \\ (\Sigma_t)_{i,j} - (2/T) X_{a,b} X_{j,(b-t) \bmod T} & j = a,\ j > a \\ (\Sigma_t)_{i,j} - (2/T) X_{a,b} (X_{a,(b+t) \bmod T} + X_{a,(b-t) \bmod T}) & i = a,\ j = a,\ t \neq 0 \\ (\Sigma_t)_{i,j} & \text{else.} \end{cases} \tag{6}$$

The correlation matrices may therefore be computed in $O(nT)$ FLOPS, since nT entries need to be updated, and each update takes $O(1)$ FLOPS. The updates may also be performed in parallel, since the updates are independent of each other.

2. Computing objective function deltas

We now show how to efficiently compute the *objective function delta*, or the change in the objective function value, when $X_{a,b}$ is flipped. Let X' be the updated code matrix (5) with $X_{a,b}$ flipped. Then, the objective function delta corresponding to this bit flip is

$$\Delta_{a,b} = f(X') - f(X). \tag{7}$$

Since the updated correlation matrices (Σ'_t) can be computed in $O(nT)$ FLOPS using (6), $\Delta_{a,b}$ may also be computed in $O(nT)$ FLOPS.

Given the current correlation matrices (Σ_t), the objective function delta $\Delta_{a,b}$ is

$$\Delta_{a,b} = \sum_{(t,i,j)\in\mathcal{J}(a)} \left(|g(\Sigma_t, a, b)_{i,j}|^p - |(\Sigma_t)_{i,j}|^p\right), \tag{8}$$

where

$$\mathcal{J}(a) = \{(t,i,j) \in \mathcal{I} \mid i = a\} \cup \{(t,i,j) \in \mathcal{I} \mid j = a\}$$

is the set of indices (t,i,j) for which $i = a$ or $j = a$. This involves a sum over only $nT-1$ terms, since the terms $(t,i,j) \notin \mathcal{J}(a)$ do not change when $X_{a,b}$ is flipped. Like the correlation matrix updates, the objective function delta may be computed in parallel, since the terms in the sum are independent of each other.

3. Updating the objective function delta matrix

Given a code matrix X, the delta matrix $\Delta \in \mathbf{R}^{n\times T}$ may be computed for all nT possible bit flips in $O(n^2T^2)$ FLOPS using (8). In this subsection, we show how the delta matrix may be updated in only $O(nT^2)$ FLOPS after a single bit flip.

Let X' be the updated code matrix (5) after flipping $X_{a,b}$. Then X' has correlation matrices (Σ'_t), which may be computed using the update rule (6): $\Sigma'_t = g(\Sigma_t, a, b)$, for $t = 0, \dots T-1$. Let $\Delta' \in \mathbf{R}^{n\times T}$ be the delta matrix for X', *i.e.*, $\Delta'_{a',b'}$ is the change in the objective function value when $X'_{a',b'}$ is flipped. Then, we may write $\Delta'_{a',b'}$ as

$$\begin{aligned}
\Delta'_{a',b'} &= \sum_{(t,i,j)\in\mathcal{J}(a')} \left(|g(\Sigma'_t, a', b')_{i,j}|^p - |(\Sigma'_t)_{i,j}|^p\right) \\
&= \Delta_{a',b'} + \sum_{(t,i,j)\in\mathcal{J}(a')} \left(|g(\Sigma'_t, a', b')_{i,j}|^p - |(\Sigma'_t)_{i,j}|^p - |g(\Sigma_t, a', b')_{i,j}|^p + |(\Sigma_t)_{i,j}|^p\right),
\end{aligned}$$

for $a' = 1, \dots, n$ and $b' = 0, \dots, T-1$.

When $a' \neq a$, the sum has only T nonzero terms, corresponding to the cases where $i = a$ and $j = a'$ or $i = a'$ and $j = a$. Therefore, $\Delta'_{a',b'}$ may be computed in $O(T)$ FLOPS when $a' \neq a$. When $a' = a$, the sum has no nonzero terms, and is equivalent to computing directly computing $\Delta'_{a',b'}$ using (8) in $O(nT)$ FLOPS.

This leads to the following update rules for $\Delta'_{a',b'}$. If $a' = a$,

$$\Delta'_{a',b'} = \sum_{(t,i,j)\in\mathcal{J}(a')} \left(|g(\Sigma'_t, a', b')_{i,j}|^p - |(\Sigma'_t)_{i,j}|^p\right). \tag{9}$$

If $a' \neq a$,

$$\Delta'_{a',b'} = \Delta_{a',b'} + \sum_{(t,i,j)\in\mathcal{K}(a,a')} \left(|g(\Sigma'_t, a', b')_{i,j}|^p - |(\Sigma'_t)_{i,j}|^p - |g(\Sigma_t, a', b')_{i,j}|^p + |(\Sigma_t)_{i,j}|^p\right), \tag{10}$$

where

$$\mathcal{K}(a,a') = \{0, \dots, T-1\} \times \{(a,a'), (a',a)\}.$$

Since (9) costs $O(nT)$ FLOPS and (10) costs $O(T)$ FLOPS, the total cost of updating Δ' is $O(nT^2)$ FLOPS. This is a factor of n fewer than recomputing Δ' from scratch using (8). The delta matrix update may be done in parallel, since the updates are independent of each other.

In §IV, we present a fast bit-flip descent method that leverages the efficient updates for the correlation matrices, objective function deltas, and delta matrix derived in this section.

Algorithm 1 Bit-flip descent for spreading code design

```
Initialize: Code matrix X
Compute correlation matrices (Σ_t) using (3)
for k = 1, ..., K do
    Choose a set of M_k distinct indices S_k = {(a_i, b_i)}_{i=1}^{M_k} at random
    Select best index (a, b) = argmin_{(a',b')∈S^k} Δ_{a',b'}
    if Δ_{a,b} < 0 then
        Σ_t ← g(Σ_t, a, b) for t = 0, ..., T − 1
        X_{a,b} ← −X_{a,b}
return X
```

IV. BIT-FLIP DESCENT

1. Algorithm

In this section, we propose a fast bit-flip descent method that leverages the efficient methods for updating the correlation matrices, objective function deltas, and delta matrix described in §III. This enables the method to find good solutions to the spreading code design problem (4) in a reasonable amount of time, even when the number of bits nT is large, on the order of a million.

Bit-flip descent. Starting from an initial code matrix, the bit-flip descent method iteratively searches for bits that, when flipped, reduce the objective function value. In the k-th iteration, the method randomly selects a set of distinct candidate bit indices $S_k = \{(a_i, b_i)\}_{i=1}^{M_k}$, where $1 \leq M_k \leq nT$ is the search size in the k-th iteration. The method then selects the index $(a, b) \in S_k$ that offers the greatest improvement in the objective function value, *i.e.*, the index that has the smallest delta $\Delta_{a,b}$. If $\Delta_{a,b} < 0$, the bit $X_{a,b}$ is flipped. The method is summarized in Algorithm 1.

Convergence. The method is guaranteed to converge to a local optimum, where no single bit flip can further reduce the objective function value, although it may not achieve a global optimum. Nonetheless, the method is highly efficient and, in practice, consistently generates codes with low auto- and cross-correlation values. It can be run for a fixed number of iterations K, until convergence, or until a stopping criterion is reached, such as when the objective function fails to improve after a specified number of iterations. The method can be carried out multiple times from different random initial code designs, after which we take the best one as our design.

Initialization. Algorithm 1 requires an initial code matrix X. In this work, we initialize X at random, where each entry $X_{i,t}$ takes value -1 or 1 with equal probability. Alternatively, the method may be initialized with codes that have been designed using a different method, or with codes that are known to have good correlation properties, such as the Gold codes (Gold, 1967) or the Weil codes (Legendre, 1808; Rushanan, 2007).

2. Choice of search size

The choice of search size M_k plays a crucial role in the method's performance. A larger search size M_k can yield greater improvements in the objective function with fewer iterations, but comes at the cost of higher per-iteration computational cost. Conversely, a smaller search size reduces the cost per iteration but may require more iterations to converge, since it is more likely to spend time evaluating non-promising bit flips.

We begin by presenting an efficient implementation of the greedy search strategy, which always selects the bit flip with the best delta and corresponds to the case where $M_k = nT$. Next, we explore fixed search sizes and introduce an adaptive strategy that gradually increases the search size M_k over time.

Greedy search. When $M_k = nT$, the method is referred to as a *greedy search*, since the method always selects the bit flip that leads to the greatest improvement in the objective function. Greedy search tends to converge in a relatively small number of iterations, but may be computationally intractable. This is because the entire delta matrix needs to be computed in each iteration. It costs $O(n^2T^2)$ FLOPS to compute the delta matrix using (8), since each delta costs $O(nT)$ FLOPS. However, the per-iteration cost can be reduced by a factor of n to $O(nT^2)$ FLOPS using the delta matrix update rules (9) – (10). Therefore, it is only necessary to directly compute the delta matrix in $O(n^2T^2)$ FLOPS once, for the initial code matrix. In subsequent iterations, the $O(nT^2)$ update rules may be applied after each bit flip. Since a bit is always flipped in each iteration, the per-iteration cost is $O(nT^2)$ FLOPS.

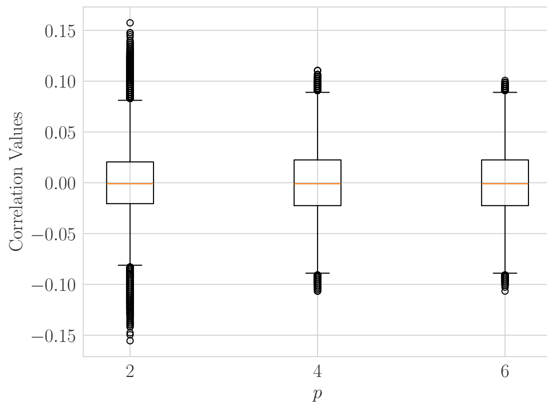


Figure 1: Correlation values for varying values of p, for the GPS L1 C/A test case with $n = 63$ and $T = 1023$.

Fixed search size. Since greedy search can be computationally expensive, an alternative is to use a fixed search size $M_k = M$, where $M < nT$. When $M = 1$, the method accepts the first bit flip that improves the objective function. For $M > 1$, the method instead selects the first bit flip that outperforms $M - 1$ other randomly chosen bit flips. In our experiments, a modest search size of $M = 100$ was found to achieve the best tradeoff between objective improvement and per-iteration cost, maximizing convergence speed in terms of overall running time.

Adaptive search size. The fixed search size strategy requires selecting a suitable value for M, which may vary depending on n and T, and may require extensive tuning to identify. We propose an adaptive strategy that gradually increases the search size M_k, which can work well for a range of code design problems without tuning the search size. The adaptive strategy is initialized with $M_1 = 1$, and M_k is increased by 1 when the selected bit flip fails to improve the objective function for two consecutive iterations, *i.e.*, when $\Delta_{a,b} \geq 0$ in line 6 of Algorithm 1 twice in a row. Since the k-th iteration costs $O(M_k nT)$ FLOPS, we switch to the greedy strategy when the per-iteration cost exceeds the greedy strategy's per-iteration cost, which is $O(nT^2)$ FLOPS. In our experiments, we switch to the greedy strategy when $M_k = 10T$.

3. Extensions and variations

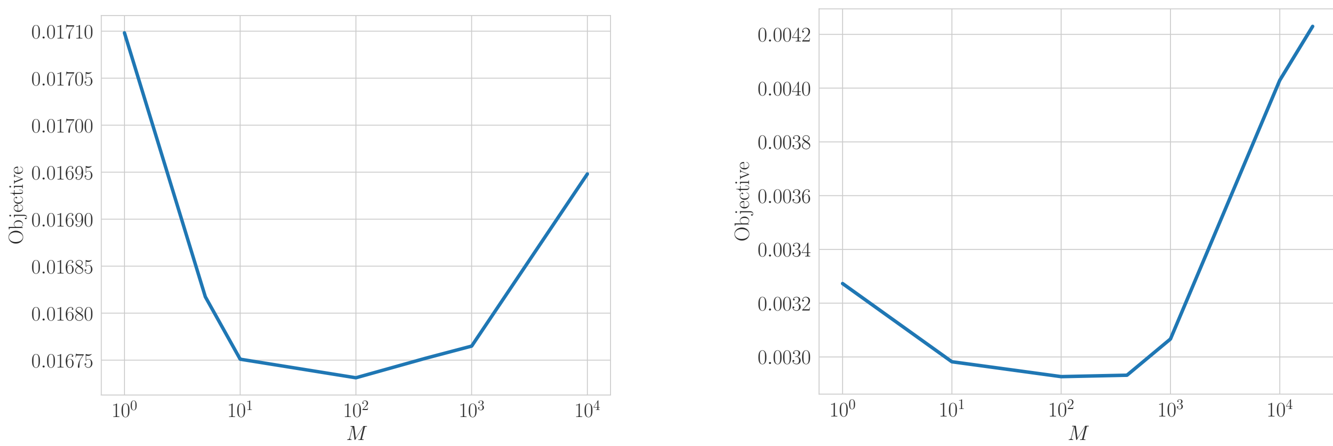
In this subsection, we consider a few possible variations on Algorithm 1.

Randomization. Algorithm 1 only changes the sign of a code entry if doing so strictly improves the objective. This can lead to the algorithm being trapped at suboptimal local minima, where the objective value cannot be improved by changing a single code entry. Randomization may be introduced to escape those suboptimal local minima. For example, we may run Algorithm 1 several times, each time with a different initial code matrix, and select the best code matrix found. Another approach is simulated annealing (Bertsimas and Tsitsiklis, 1993), which is a randomized version of Algorithm 1 that flips $X_{a,b}$ with some positive probability when $\Delta_{a,b} \geq 0$. Simulated annealing can find solutions with lower objective function values since it allows for objective-worsening bit flips to explore the search space, but tends to require careful tuning and be slow to converge.

Constraints. The method may be extended to handle constraints on the code matrix X. For example, we may require each code to have the same number of -1 and 1 entries (Soualle et al., 2005). Another constraint is the autocorrelation sidelobe zero property, which is useful for improving signal tracking performance (Wallner et al., 2007). Codes that satisfy the autocorrelation sidelobe zero property have shift-1 autocorrelation values $(\Sigma_1)_{i,i}$ that are zero (or ± 1, when T is odd). A simple approach for incorporating these constraints is to modify Algorithm 1 to only consider bit flips that do not violate the constraints. That is, if flipping $X_{a,b}$ violates the constraints, then $\Delta_{a,b}$ is taken to be infinite (Yang et al., 2024). This approach requires the initial codes to satisfy the constraints. An appropriate initialization can be found by modifying the objective function to penalize only the constraint violations, and then running Algorithm 1.

V. NUMERICAL EXAMPLES

We consider three code design problems, summarized in Table 1. The first is based on the GPS L1 C/A codes, which consist of 63 length-1023 codes (MilComm & PNT Directorate, Space Systems Command, 2022a). The second is based on the Galileo



(a) GPS L1 C/A: $n = 63, T = 1023$

(b) Galileo E1: $n = 100, T = 4092$

Figure 2: Objective values achieved in one hour of runtime for different choices of M.

E1 codes, which consist of 100 length-4092 codes (European Union, 2021). The third and largest code family is based on the GPS L1C signal, and has 210 codes, each of length 10230 (MilComm & PNT Directorate, Space Systems Command, 2022b). The total number of bits ranges from 64,000 for the GPS L1 C/A example to 2 million for the GPS L1C example.

In the following experiments, we initialize the code matrices X at random, and all methods being compared are initialized with the same code matrix. Unless otherwise specified, we take $p = 6$ in the objective function (2). Each method is run until convergence, or until a time limit has been reached. The computation of the correlation matrices, objective function, objective function deltas, and delta matrix were all parallelized. The GPS L1 C/A and Galileo E1 experiments were run on a machine with a 32-core CPU, while the GPS L1C experiments were run on a machine with a 128-core CPU. Our code is publicly available at:

`https://github.com/Stanford-NavLab/decor`

In §V.1, we evaluate the effect of the parameter p in the objective function and the search size M in the fixed-size strategy on the performance of Algorithm 1. In §V.2, we compare the performance of Algorithm 1 for different search strategies.

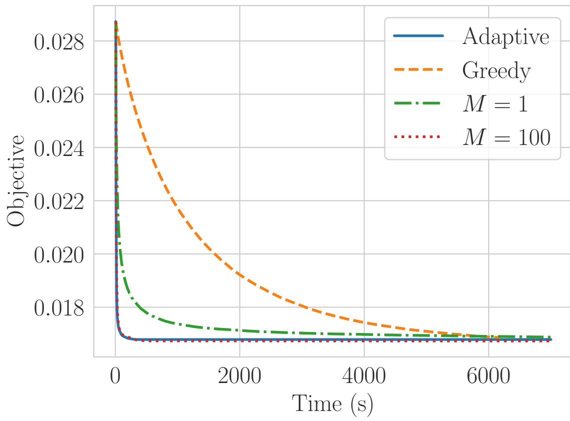
1. Choice of parameters

Choice of p. In this example, we run Algorithm 1 with the adaptive search strategy and three values of p, for the GPS L1 C/A test case with $n = 63$ and $T = 1023$. Figure 1 shows box plots of the $T(n^2 + n)/2 - n$ correlation values $(\Sigma_t)_{i,j}$ of the optimized codes, for $p = 2, 4,$ and 6. Increasing p from 2 to 6 reduced the maximum correlation value from 0.157 to 0.107. However, the standard deviation of the correlation values increased slightly, from 0.0308 to 0.0310.

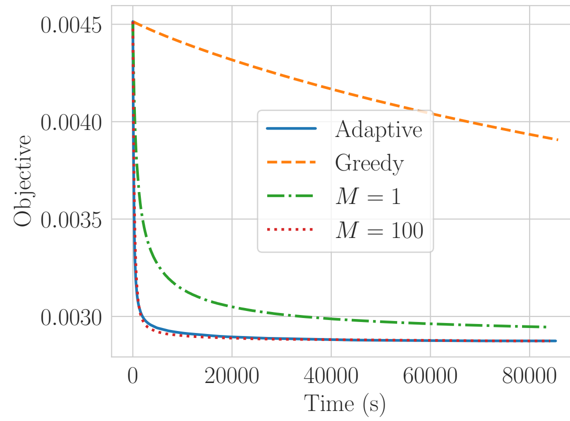
Choice of M. Next, we evaluate the effect of the parameter M on the performance of the fixed search size strategy with $M_k = M$. A good choice of M minimizes the convergence speed as measured by running time. Increasing M decreases the number of iterations needed to converge, but also increases the per-iteration cost. On the other hand, decreasing M reduces the per-iteration cost, but also increases the number of iterations needed to converge. Figure 2 shows the objective value achieved by Algorithm 1 for the GPS L1 C/A and Galileo E1 test cases after one hour of running time, for different values of M. For both test cases, choosing $M = 100$ achieves the best convergence speed.

	Code length (T)	Family Size (n)	Number of bits (nT)
GPS L1 C/A	1,023	63	64,449
Galileo E1	4,092	100	409,200
GPS L1C	10,230	210	2,148,300

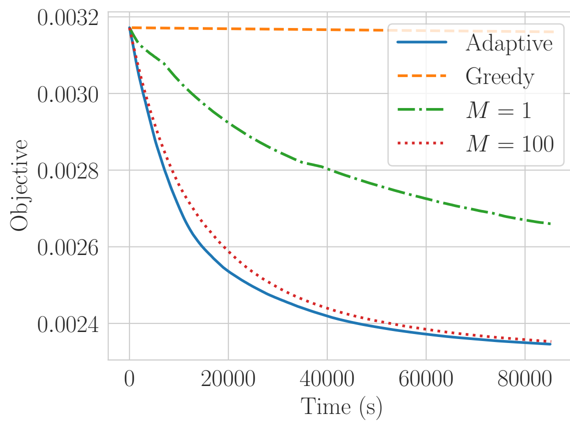
Table 1: Summary of code design problems.



(a) GPS L1 C/A: $n = 63, T = 1023$



(b) Galileo E1: $n = 100, T = 4092$



(c) GPS L1C: $n = 210, T = 10230$

Figure 3: Objective value vs. runtime of Algorithm 1 with four different search strategies, for the three test cases.

2. Performance comparison

In this subsection, we compare the performance of Algorithm 1 for four search strategies: adaptive, greedy search, and fixed search size with $M = 1$ and $M = 100$. We ran each method for a maximum of 24 hours. For the GPS L1 C/A and Galileo E1 test cases, we ran each method ten times with different initial codes. For the GPS L1C test case, we ran each method only once starting from the same random initialization, due to the high computational cost. Table 2 shows the achieved objective value and percent improvement over the initial codes for each method and test case. For the GPS L1 C/A and Galileo E1 test cases, best result out of the ten runs are reported. Figure 3 shows the objective value vs. runtime for each method and test case, for one of the ten runs. Finally, Table 3 shows the minimum and maximum objective values achieved by each method over the ten runs.

Overall, the adaptive search strategy achieved the best performance across the three test cases. The fixed search size strategy with $M = 100$ also performed well, achieving similar performance to the adaptive strategy. While the greedy strategy performs reasonably for the GPS L1 C/A case, it makes little progress for the other two test cases, due to the high per-iteration cost. Finally, Table 3 suggests that the particular choice of initial code does not significantly affect the final objective value, since the minimum and maximum objective values achieved by each method are similar.

VI. CONCLUSIONS

In this paper, we introduced a bit-flip descent method for optimizing binary spreading codes, specifically designed for the large family sizes and extended lengths required by modern GNSS and emerging PNT applications. We developed efficient techniques to update correlation matrices, objective functions, and objective deltas after individual bit flips, allowing the method to scale effectively to large-scale code design problems involving millions of bits. Additionally, the method requires minimal tuning, making it a practical and flexible option for testing various code lengths, family sizes, and objective functions tailored to specific applications.

	GPS L1 C/A		Galileo E1		GPS L1C	
	Obj. ($\times 10^{-3}$)	% Impr.	Obj. ($\times 10^{-3}$)	% Impr.	Obj. ($\times 10^{-3}$)	% Impr.
Adaptive	16.7	42.2	2.9	36.6	2.3	26.0
Greedy	16.8	42.0	3.7	19.0	3.2	0.3
$M = 1$	16.7	42.1	2.9	35.1	2.7	16.2
$M = 100$	16.7	42.1	2.9	36.6	2.4	25.8

Table 2: Objective values and percent improvements over the initial code across the four search strategies and three test cases.

	GPS L1 C/A		Galileo E1	
	Min obj. ($\times 10^{-3}$)	Max obj. ($\times 10^{-3}$)	Min obj. ($\times 10^{-3}$)	Max obj. ($\times 10^{-3}$)
Adaptive	16.7	16.8	2.9	2.9
Greedy	16.8	17.0	3.7	3.9
$M = 1$	16.7	16.8	2.9	2.9
$M = 100$	16.7	16.8	2.9	2.9

Table 3: Minimum and maximum objective values achieved by each method over ten runs, for the GPS L1 C/A and Galileo E1 test cases.

ACKNOWLEDGEMENTS

This material is based upon work supported by the Air Force Research Lab (AFRL) under grant number FA9453-20-1-0002.

REFERENCES

AFRL US Air Force (2023). Navigation Technology Satellite–3 (NTS-3).

Alaee-Kerahroodi, M., Modarres-Hashemi, M., and Naghsh, M. M. (2019). Designing sets of binary sequences for MIMO radar systems. *IEEE Transactions on Signal Processing*, 67(13):3347–3360.

Angeris, G., Vučković, J., and Boyd, S. (2021). Heuristic methods and performance bounds for photonic design. *Optics Express*, 29(2):2827–2854.

Avila-Rodriguez, J.-A., Wallner, S., and Hein, G. (2006). How to optimize GNSS signals and codes for indoor positioning. In *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pages 2418–2426.

Bertsimas, D. and Tsitsiklis, J. (1993). Simulated annealing. *Statistical science*, 8(1):10–15.

Chapman, D., Hinks, J., and Anderson, J. (2020). Way, way out in front–Navigation Technology Satellite-3: The vanguard for space-based PNT. *Inside GNSS*, 15(4):32–35.

European Union (2021). European GNSS (Galileo) open service: Signal-in-space interface control document. Technical report, European Union.

Gold, R. (1967). Optimal binary sequences for spread spectrum multiplexing. *IEEE Transactions on information theory*, 13(4):619–621.

Israel, D. J. and Gramling, C. J. (2023). Lunanet signal-in-space recommended standard-augmented forward signal. Technical report.

Jiang, Y., Kim, H., Asnani, H., Kannan, S., Oh, S., and Viswanath, P. (2020). Joint channel coding and modulation via deep learning. In *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE.

Legendre, A.-M. (1808). Essai sur la théorie des nombres. Chez Courcier.

MilComm & PNT Directorate, Space Systems Command (2022a). NAVSTAR GPS space segment/navigation user segment interfaces (IS-GPS-200, Revision N). Technical report, Navstar GPS Directorate.

MilComm & PNT Directorate, Space Systems Command (2022b). NAVSTAR GPS space segment/user segment L1C interfaces (IS-GPS-800, Revision J). Technical report, Navstar GPS Directorate.

Mina, T. Y. and Gao, G. X. (2022). Designing low-correlation GPS spreading codes with a natural evolution strategy machine-learning algorithm. *NAVIGATION, Journal of the Institute of Navigation*, 69(1).

Misra, P. and Enge, P. (2012). *Global Positioning System: Signals, Measurements, and Performance - Revised Second Edition*. Ganga-Jamuna Press.

Reid, T., Walter, T., Enge, P. K., Lawrence, D., Cobb, H. S., Gutt, G., O'Connor, M., and Whelan, D. (2020). Navigation from low earth orbit: part 1: concept, current capability, and future promise. *Position, Navigation, and Timing Technologies in the 21st Century: Integrated Satellite Navigation, Sensor Systems, and Civil Applications*, 2:1359–1379.

Rushanan, J. J. (2007). The spreading and overlay codes for the L1C signal. *Navigation*, 54(1):43–51.

Soualle, F., Soellner, M., Wallner, S., Avila-Rodriguez, J.-A., Hein, G. W., Barnes, B., Pratt, T., Ries, L., Winkel, J., Lemenager, C., et al. (2005). Spreading code selection criteria for the future GNSS Galileo. In *Proceedings of the European navigation conference GNSS*, pages 19–22.

Wallner, S., Avila-Rodriguez, J.-A., Hein, G. W., and Rushanan, J. J. (2007). Galileo E1 OS and GPS L1C pseudo random noise codes-requirements, generation, optimization and comparison. In *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007)*, pages 1549–1563.

Wallner, S., Avila-Rodriguez, J.-A., Won, J.-H., Hein, G., and Issler, J.-L. (2008). Revised PRN code structures for Galileo E1 OS. In *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008)*, pages 887–899.

Winkel, J. O. (2011). Spreading codes for a satellite navigation system. Patent No.: US 8,035,555 B2.

Yang, A., Mina, T., and Gao, G. (2023a). Binary sequence set optimization for CDMA applications via mixed-integer quadratic programming. In *IEEE International Conference on Acoustics, Speech, & Signal Processing*.

Yang, A., Mina, T., and Gao, G. (2024). Spreading code optimization for low-Earth orbit satellites via mixed-integer convex programming. *EURASIP Journal on Advances in Signal Processing*, 2024(1):67.

Yang, A., Mina, T. Y., and Gao, G. X. (2023b). Spreading code sequence design via mixed-integer convex optimization. In *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*.