Introduction:

* This project studies optimizing radar waveforms for signal processing
* Typically, to maximize SNR, a matched filter is applied to the signal received
* This creates unwanted sidelobes.
* Fundamental problem: how to minimize size (peak sidelobe level) or integrated power (integrated sidelobe level) in these sidelobes?
* One possible solution: design a waveform that has these properties for its *autocorrelation* or *ambiguity function*.

AC(c)\_{p} = , AM(c)\_{n, p} =

* Rigorously:

Problem: given a code a = {a\_i} with a\_i \in C and a function f, minimize f(AC(a)) or f(AM(a)).

* Frequently: f outputs the second highest value (peak sidelobe level) or integrated power in all but the peak (integrated sidelobe level).

Types of codes:

* In this project, we considered only coding using bauds, in which the pulse is split into equal segments, encoded by the phase of that segment.
* Binary codes: a = {a\_i}, a\_i \in {-1, 1}, simplest to generate.
* Polyphase codes with fixed generator: a = {a\_i}, a\_i \in {e^{2\pi ij/m}, j \in {0, … m-1}}, fixed m.
* Arbitrary polyphase codes: a = {a\_i}, a\_i = e^{I \* phi\_i} for some phase phi\_i.
* These are all amplitude one codes; can also consider amplitude modulated codes, but these are usually very difficult to generate.

Exhaustive search:

* Simplest way to generate optimal code: search through all possible codes
* Naively, this takes exponential time
* Number of ways to optimize this process
  + Exploit PSL-preserving operations: reversal, negation, alternate negation
  + Branch-and-bound algorithms
* Wrote my own efficient algorithm, graph below compares the runtime of naïve and efficient algorithms, written in Python

Heuristic optimization algorithms

* For lengths > 40, exhaustive search takes far too long
* More problematically, runtime still grows too fast
* Enter optimization: exist many ways to estimate the minimum of a function
* Many techniques require differentiation, however, or heavily depend in performance on continuity
* Solution: heuristic optimization
  + Hill climbing
  + Simulated annealing
  + Tabu search
  + Evolutionary, genetic algorithms
  + Particle swarm
  + Great deluge approach
* Two classes:
  + Population-based approach. Start with population of codes, evolve them over time based on their PSL values.
  + Single code approach. Randomly generate a single code, and move it around based on an update step that moves towards minima but can escape local minima.
* Most powerful algorithms: global heuristic with a local search subroutine.

Binary codes:

* Extensive literature applying optimization algorithms to minimizing PSL or ISL of binary code (citations here)
* To compare existing techniques, we wrote pseudocode based off of some of these approaches, as well as some of our own
* Our algorithms were able to, even with short time constraints, find codes close to optimal for lengths 2-55.
* Best algorithms: basic hill climbing, evolutionary algorithms, memetic tabu search.

Arbitrary polyphase codes

* Can also consider codes that are not binary, but have elements that are arbitrary roots of unity
* These can be optimized similarly to binary codes.
* More generally, can consider codes with {c\_n = e^{i \* phi\_n}} for arbitrary phases phi\_n.
* Some literature applying optimization algorithms to this problem; we test an existing technique, the great deluge algorithm, against a simulated annealing and particle swarm approach.
* All work better than a random approach

Doppler ambiguity functions

* This project’s main addition not well studied in the literature is minimizing over Doppler space
* Our algorithms can be extended naturally to minimize *any* function of the ambiguity function of a code, in any Doppler width
* Used these algorithms to generate a number of novel, near-optimal polyphase codes in Doppler widths 2-6.
* These codes, for Barker lengths, significantly outperform these codes in Doppler space, reflecting the poor Doppler tolerance of Barker codes.

Alternating codes

* Applied our algorithm to the generation of alternating codes
* Exist a deterministic way to generate alternating codes with lengths a power of 2.
* Apply our algorithms to find (near)-alternating binary and polyphase codes with lengths other than a power of 2

Future work and acknowledgments:

* Most of these algorithms were tuned for parameters, but a number of improvements could be made
  + Parallelization
  + Addition of C or C++ wrapper
  + Adaptive parameter selection, or tests to optimize for the best parameters
  + Ensemble algorithms, or additions of relevant local search routine to some of the global algorithms
  + Use of characteristics of the autocorrelation function to better inform search, crossover.
* General waveforms: briefly studied linear chirps and optimized unsuccessfully over their parameters.
  + Particle swarm and associated algorithms (gravitational search) seem well suited to this problem
* Mismatched filters: designing optimal decoding filters
* Perfect codes: limiting amplitude modulation, other techniques for finding them

Acknowledgments:

* Ryan Volz
* MIT Haystack Observatory
* NSF