Parallel Optimization of Signal Detection in Active Magnetospheric Signal Injection Experiments

Michael Gowanlock[1,2], Justin D. Li[2], Cody M. Rude[2], Victor Pankratius[2]

[1]*School of Informatics, Computing & Cyber Systems, Northern Arizona University, Flagstaff, AZ, U.S.A.*
michael.gowanlock@nau.edu

[2]*Massachusetts Institute of Technology, Haystack Observatory, Westford, MA, U.S.A.*
[jdli, cmrude, pankrat]@mit.edu

Abstract

Signal detection and extraction requires substantial manual parameter tuning at different stages in the processing pipeline. Time-series data depends on domain-specific signal properties, necessitating unique parameter selection for a given problem. The large potential search space makes this parameter selection process time-consuming and subject to variability. We introduce a technique to search and prune such parameter search spaces in parallel and select parameters for time series filters using breadth- and depth-first search strategies to increase the likelihood of detecting signals of interest in the field of magnetospheric physics. We focus on studying geomagnetic activity in the extremely and very low frequency ranges (ELF/VLF) using ELF/VLF transmissions from Siple Station, Antarctica, received at Québec, Canada. Our technique successfully detects amplified transmissions and achieves substantial speedup performance gains as compared to an exhaustive parameter search. We present examples where our algorithmic approach reduces the search from hundreds of seconds down to less than one second, with a ranked signal detection in the top 99th percentile, thus making it valuable for real-time monitoring. We also present empirical performance models quantifying the trade-off between the quality of signal recovered and the algorithm response time required for signal extraction. In the future, improved signal extraction in scenarios like the Siple experiment will enable better real-time diagnostics of conditions of the Earth's magnetosphere for monitoring space weather activity.

Keywords: Parallel Parameter Search, Signal Processing, Siple Station Experiment, Time Series Analysis.

1. Introduction

We introduce new parallel optimization techniques for the detection of signals injected into the Earth's magnetosphere. The application addresses a variety of interesting questions, such as improving an understanding of radiation-belt dynamics, monitoring space weather conditions, and advancing scientific insight into the coupling dynamics between the Earth and the Sun.

Signal detection and extraction is challenging for applications where signal transmission occurs in noisy channels with interference. This is further complicated in this application by interactions of interest between energetic particles and whistler-mode waves in the radiation belts that modify wave behavior and plasma particle populations [1, 2, 3, 4]. These waves are generated on Earth naturally, for instance by lightning, or artificially, such as by power lines. We examine data from a controlled wave-particle interaction experiment at Siple Station, Antarctica, where a dedicated transmitter injected signals in the extremely and very low frequency ranges (ELF/VLF; 0.3 - 30 kHz) into the magnetosphere. Such signals travel along field-aligned paths through the magnetosphere, undergoing modifications through the interactions with energetic electrons in the radiation belts, and are received at the geomagnetic conjugate point in Québec, Canada [5, 6]. This experimental setup is shown in Figure 1.

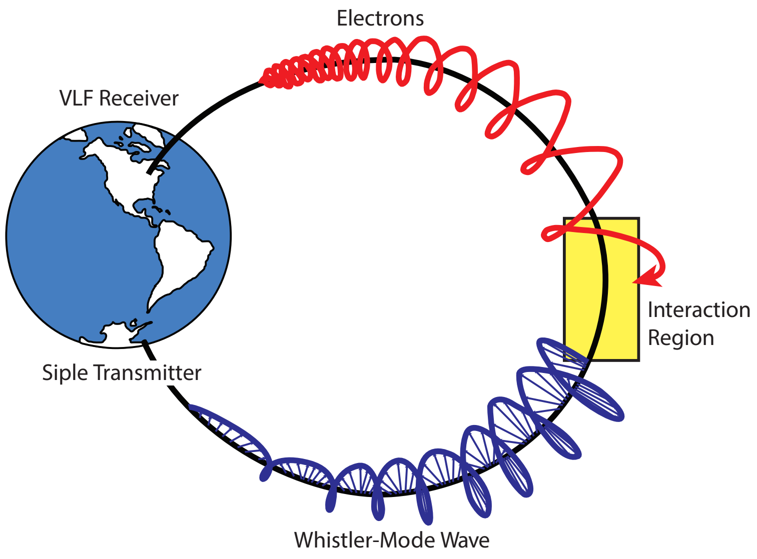


Figure 1: The transmitter at Siple Station, Antarctica, injects whistler-mode waves (blue) into the magnetosphere, which undergo interactions with energetic electrons (red) in the interaction region (yellow) near the equator. The modified waves then continue along the field-aligned ducts and are received at the receiver at the geomagnetic conjugate point in Québec, Canada. This figure is adapted from [7].

The received signal is broadband in nature, which requires applying a configurable signal processing chain in order to extract the signal of interest, including for example a lightning filter, a demodulation stage, a smoothing filter, and a signal thresholding filter. The challenge is: how to select the respective parameters in each stage in order to separate the noise and enhance the detection of the transmitted signal? This problem can lead to large search spaces which can make an exhaustive search intractable.

While the typical signal detection approach uses a manually tuned signal processing chain, alternate parameters from the broader permissible range could provide other insights and emphasize different salient features in the signal. In this work, we automate this search and pruning process while leveraging parallel computing to improve algorithm response time. These performance gains can lead to better scientific insights by exploring more of the parameter space, or enable real-time signal detection. We develop a parallel, multithreaded implementation that achieves good scalability on a multicore platform which improves search throughput and present the results from our evaluation on data from the Siple Station Experiment. Our methods show possible applications for improving the detection of events exhibiting magnetospheric amplification and for highlighting different salient features based on the searched parameter space.

2. The Siple Station Magnetospheric Signal Injection Experiment

2.1. Overview

Siple Station, Antarctica, operated a powerful ELF/VLF transmitter for controlled wave-injection experiments from 1973 to 1988. It enabled a unique experiment in magnetospheric physics, whose observations remain unmatched by any other instrument in providing long-running observations of wave-particle interactions in the magnetosphere [7]. These data have improved our understanding of various magnetospheric phenomena and processes, but due to their historicity and format, remain an underutilized dataset [5].

We examine data from the 1986 portion of the dataset, which was previously preprocessed to allow for digital analysis [5]. An example of an MDIAG (magnetospheric diagnostic format) transmission from 6/23/1986 7:01:00 UT is shown in spectrogram format in Figure 2. The spectrogram shows several features of interest that must be considered in the signal processing analysis. The narrow, vertical features indicate lightning strikes, while the long, horizontal features correspond with power line harmonics. Both such features interfere with detecting the transmitted signal. The 2 s pulse of interest is the higher amplitude signal at 3480 Hz, which can be seen starting around 3 s.

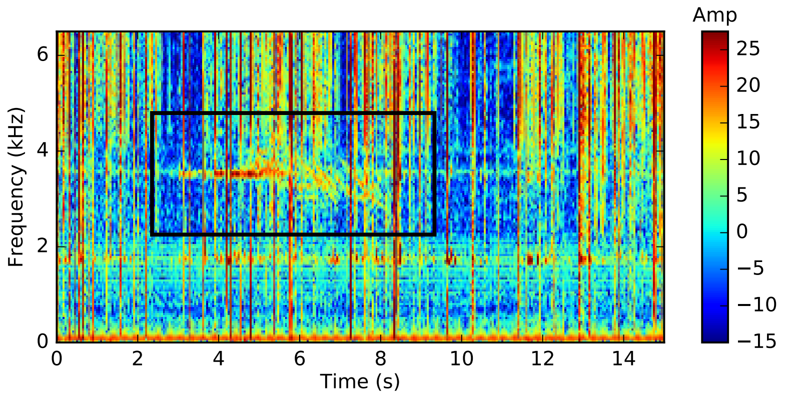


Figure 2: Example of an MDIAG transmission from 6/23/1986 7:01:00 UT shown in a time-frequency amplitude representation using a spectrogram. The signal component is the narrowband transmission at 3480 Hz, as part of the magnetospheric diagnostic format, MDIAG [5] which is enclosed within the black box.

2.2. *Siple Experiment Signal Processing Pipeline*

The canonical signal analysis in this field is exemplified by the process in [5, 4]. It consists of removing the lightning-generated noise (called sferics) in the data, extracting the narrowband signal amplitude based on transmission characteristics, smoothing to reduce noise, and thresholding to generate a simplified signal for computing the fitness function. Sferics are strongly impulsive and can easily dominate the received broadband signal. For simplicity, sferics are primarily removed by zeroing portions of the signal that exceed some number of standard deviations of the amplitude. Then, the signal is mixed down to baseband by shifting the given transmission frequency to 0 Hz and low-pass filtering to extract the narrowband amplitude. A median filter is applied to smooth residual impulsive noise from the sferics as well as artifacts from the low-pass filter. Finally, a regression tree classifier [8] thresholds the signal and approximates the detectable signal amplitudes.

The quality of the signal extraction is evaluated by cross-correlating the thresholded signal with a defined approximation of the signal, which can be obtained based on the transmission characteristics. The result is normalized to range between -1 and 2, which will serve as a fitness function score in our approach (Section 2.2.3).

2.2.1. *Signal Processing Pipeline Stages*

- Stage 1: Takes the entire dataset and thresholds by a single parameter, the number of standard deviations, to remove the impulsive sferics and produce a filtered dataset of the same length. The number of standard deviations ranges between 0.5 and 2.0 to encompass the range of coincident sferics [9].

- Stage 2: Takes the sferic-filtered data and calculates the narrowband amplitude at the known transmission frequency. This stage involves 4 steps: (a) mixing the signal down to baseband using a complex exponential frequency shift; (b) generating a low-pass filter using three parameters, the number of filter taps, the passband frequency, and the roll-off frequency; (c) applying the low-pass filter; and (d) downsampling the data based on passband frequency. The output of this stage is the downsampled narrowband signal amplitude. Only the low-pass filter, defined using the Remez algorithm [10], requires parameters. The number of taps ranges from 100 to 300, and the cut-off and roll-off frequencies each range from 100 to 200 Hz. These parameter ranges are reasonable for the Siple signal processing chain, similar to the values used in [5].

- Stage 3: Takes the downsampled data and applies a naïve median filter [11] with a parameter-specified window size to smooth the narrowband amplitude. The downsampling typically shrinks the data by $10-100$x. The output of this stage is a smoothed amplitude, where the data length is unchanged. Window sizes range from 7 to 23, encompassing the value used in [5].

- Stage 4: Fits a regression tree [8] of depth specified by the tree-depth parameter to estimate the time characteristics of the signal. The regression tree output of this stage is used to compute the fitness function score that determines the quality of the parameters used to detect the signal. The tree depth ranges from 1 to 3, as determined empirically in this study.

The computational complexity of stages 1, 2, 4 is $O(n)$ and $O(n(k\log k))$ for stage 3, where n is the number of data elements in the signal and k is the window size (as k is small and n decreases due to downsampling, the median filter is not computationally prohibitive). Figure 3 shows example output of stages 2-4 applied to the signal from Figure 2.

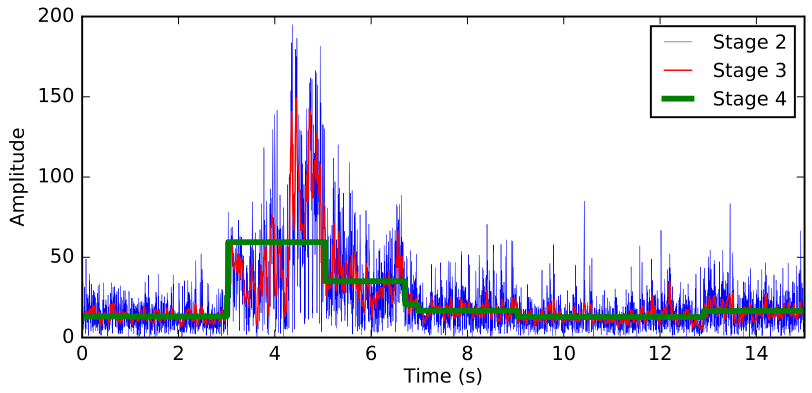


Figure 3: Visualization of the results of the filtering process for the transmission shown in Figure 2, with the transient signal readily apparent after filtering. The narrowband signal amplitude time series after applying filter stages 2,3,4 are shown in blue, red, and green respectively. This particular filter parameter configuration produced the maximum fitness function score score for detecting the signal..

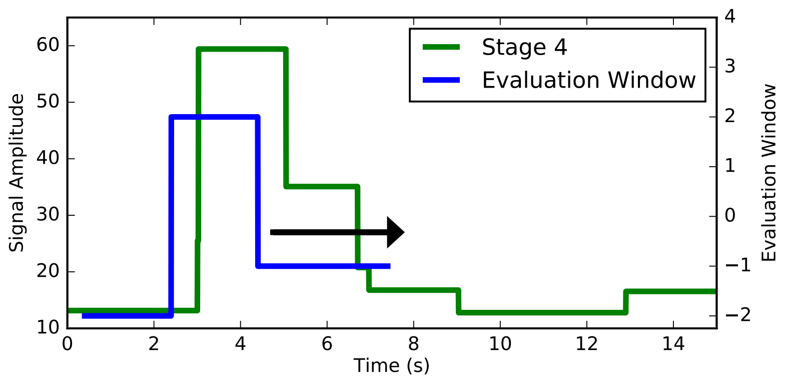


Figure 4: Visualization of the sliding window used to compute the fitness function score. The window (blue) is cross-correlated with the output of Stage 4 in Figure 3. This example achieves a metric score of 0.999.

2.2.2. *Fitness Function*

The fitness function score is computed by cross-correlating the ideal received signal (based on the characteristics of the transmission) with the processed data and normalizing to determine the highest score for a given parameter configuration. An example of this evaluation process is shown in Figure 4, where the evaluation window specified by a domain expert (in blue) slides over the regression tree output (in green) from Stage 4 (from Figure 3). For this application domain, the evaluation window is a square pulse with -2 weight for the first 2 s, $+2$ weight for the next 2 s, and -1 for the next 3 s, in order to emphasize a signal pulse surrounded by noise. For this particular example, the fitness function score is 0.999, and this parameter configuration led to the highest ranked detection. The maximum metric score of 2 would only occur for an ideal signal, i.e. a signal without noise and with idealized amplification characteristics, which is not achievable in practice.

2.2.3. *Generating the Evaluation Window*

The evaluation window in this particular application is selected to capture the occurrence of the 2 s long main tone of interest. For the purpose of detecting the MDIAG transmission, this tone is the strongest component to detect. Scientifically, the tone is useful for characterizing the conditions and degree of nonlinear amplification [5, 7]. Furthermore, the evaluation window design draws on knowledge of the overall MDIAG format, where the period before and after the tone are expected to have no transmitted signal. The following duration is weighted to account for potential multipath and signal interference from later transmitted tones.

3. Problem Statement

Prior work [5, 4] has manually selected parameters for processing data like the one used here in the Siple Station Experiment and then visually detected the transmitted signal. The above mentioned manual process has significant drawbacks and is thus the motivation for this work. We advance methods for the efficient automation of signal detection for the magnetospheric signal injection experiment.

Due to the above mentioned drawbacks of manual signal processing and parameter selection, this work takes a set of parameters *given as input before all signal processing*, and selects the subset of these parameters that yield the best signal detection based on the evaluation score (Section 2.2.2) defined by the evaluation window (Section 2.2.3). Thus, the signal processing pipeline is supervised, as permissible parameter ranges need to be given as input.

The signal processing pipeline consists of a series of stages; the output of one stage is used as input for the following stage. Let P be a pipeline with n stages. Each stage is denoted as S_i where $i = 1, \ldots, n$, and $|S_i|$ denotes the number of parameters for each stage. Each S_i has a list of parameters, denoted as p_i, where $p_i = (p_i^j, p_i^{j+1}, \ldots, p_i^{|S_i|})$, where the j^{th} value in this list defines the value of the j^{th} parameter. We use the notation S_i^j to denote the value of the parameter as applied to a specific stage. Note that S_i^j may be an ordered list of parameters, if a given stage takes multiple parameters as input. However, since we enumerate all of the possible parameters for each stage up to $|S_i|$, a list of parameters applied to a single stage is considered a single parameter.

We aim to find a list of parameters that discovers a signal in the data. Figure 5 shows possible enumerations in a simple pipeline that uses 1 parameter in S_1, $|S_1|= 1$, and 2 parameters in the other stages, i.e., $|S_2|= |S_3|= |S_4|= 2$. The output of S_4 is used to compute the fitness function score that evaluates the quality of the signal.

Figure 5 illustrates our search space. The hierarchy implies that after S_1^1 has been processed, its results can be used in all subsequent stages of S_2, and the output of S_2^1 and S_2^2 can be reused in S_3 and so on. Assuming hierarchical reuse, there are different trade-offs to how we search for the best set of parameters. A breadth-first search (BFS) [12] or depth-first search (DFS) [13] may be more appropriate depending on the characteristics of the signal and the quality of the subsequent parameter values in finding that signal. An exhaustive search (evaluating all of S_4 by enumerating the parameter values for each stage) will yield the best possible result for the selection of input parameters at each stage. Greater detail is given in upcoming sections.

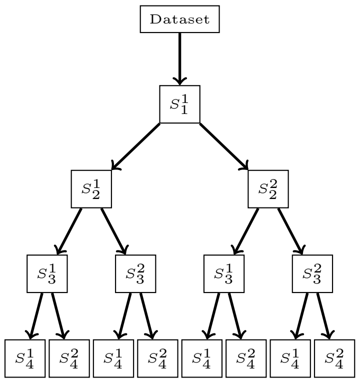


Figure 5: Example 4-stage pipeline with $|S_1|= 1$, $|S_2|= |S_3|= |S_4|= 2$.

4. Search & Pruning Techniques

4.1. Breadth First Search and Pruning

A useful property of time-series data is that its correlation with a reference signal can be incrementally evaluated after a pipeline stage has been processed. This information can be leveraged to make pruning decisions, i.e., whether to continue applying additional steps in a particular pipeline variant or if the result so far does not warrant further parameter exploration.

In Figure 5, the output of the stages S_1, S_2, or S_3 can be correlated with the reference signal described in Section 2.2.3. A fitness score is obtained, and can be compared to the the ideal score (Section 2.2.2), or the scores of other variants that have been evaluated at a given stage. Thus, a pruning decision can be made. For instance, not continuing the execution of a variant after S_2 because its fitness score is low in comparison to the other variants evaluated at S_2.

In the context of the Siple signal processing chain, Figure 6 shows pruning points (red horizontal lines) in the example pipeline from Figure 5. After S_2 the signal can be correlated with the metric pulse and evaluated (Sections 2.2.2 and 2.2.3). If no pruning occurs, intermediate data between stages (e.g., the output of S_2^1 as input into S_3^1 and S_3^2) may remain in cache as input into successive stages. This is a performance benefit of the BFS, and exploits locality in the memory hierarchy.

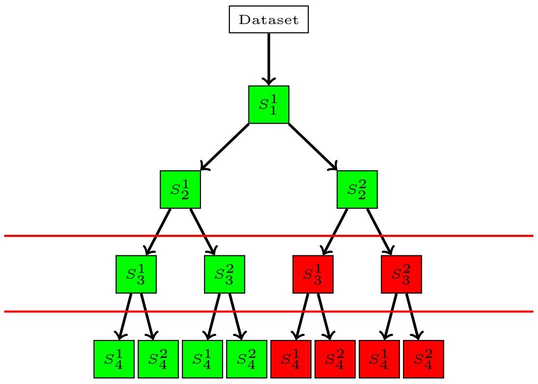


Figure 6: Example of a BFS pipeline. The red horizontal lines indicate pruning points, which occur after the amplitude outputs of S_2.

4.1.1. Stage Pruning Technique:

This approach evaluates the fitness of a signal as computed using the evaluation window (Section 2.2.3) after the execution of a stage in the variant tree, excepting leaf nodes, as illustrated in Figure 6. The results are ranked by fitness value, and we retain a fraction of the top ranking variants. This fraction, denoted as $f_{S_iS_j}$, where $0 < f_{S_iS_j} \leq 1$ for each pair of stages S_i and S_j with $j = i + 1$, can be set by a user to control trade-offs in performance and accuracy in this algorithm. For the Siple signal processing chain, we prune after S_2, which generates the signal amplitudes required for computing the fitness score. Pruning too aggressively may reduce the chances of finding the parameters that best recover the signal; however, insufficient pruning may not adequately reduce the search space.

4.2. Depth First Search and Pruning

The DFS and pruning approach executes using the following steps:

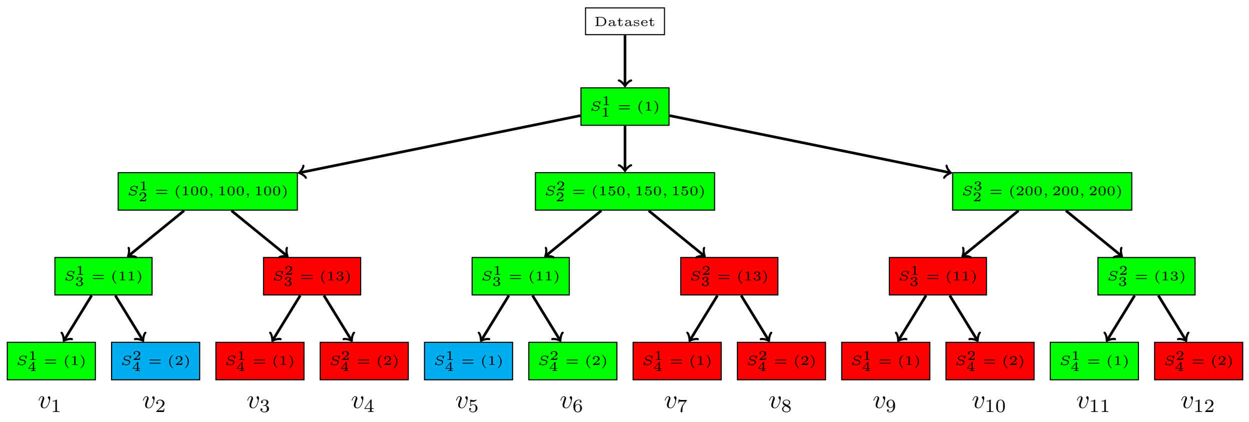


Figure 7: Example of a DFS pipeline, where parameter values are shown. See text for description.

1. This approach begins by taking a sample of variants defined by the pipeline configuration, out of all possible pipeline variants. The number variants sampled is denoted by n_t. There are initially $|S_1|\times|S_2|\times|S_3|\times|S_4|$ variants in total. Each of these sample variants are executed.
2. A list L captures the parameter configuration for each pipeline variant and its fitness value obtained at the last stage, for each of the n_t variants.
3. The fraction f_{DFS} (where $0 < f_{DFS} \leq 1$) of L with the highest rank/fitness score and determines the empirical range of values of pipeline parameters for these selected variants. The number of top ranking variants becomes $n_{top} = \lceil n_t \times f_{DFS} \rceil$; the ceiling is taken as an integral number is required.
4. Unprocessed variants whose parameters lie outside of the accepted parameter ranges (defined by n_{top}) are pruned from the search and are not executed.
5. The search begins again at Step 1, where the total number of variants left to process has decreased because some of the variants have been either processed (Step 1) or pruned (Step 4). Note that the number of variants that have been pruned depends upon the characteristics of the sample variants that have been executed. The choice of the n_t parameter determines how many variants are executed before pruning, and f_{DFS} controls how many top ranking variants are considered to construct the parameter ranges for pruning the unprocessed variants. The search ends when all variants have been either executed or pruned.

Consider Figure 7, which differs from Figure 5 with $|S_2|= 3$. We execute the variants (denoted as v_1, v_6, v_{11}) in green. Selecting $n_t = 3$ (we prune after 3 variants have been executed), and using $f_{DFS} = 2/3$, then $n_{top} = \lceil 3(2/3) \rceil = 2$. Assuming v_1 and v_6 have metric values greater than v_{11}, we use the minimum/maximum ranges bound by v_1 and v_6 to prune the variants that have not been executed. We use the notation $[minvalue, maxvalue]$ to denote the range of values which encompasses the best set of parameters.

The resulting parameter ranges from the n_{top} variants are: $S_1 = \{[1, 1]\}$, $S_2 = \{[100, 150], [100, 150], [100, 150]\}$, $S_3 = \{[11, 11]\}$, and $S_4 = \{[1, 2]\}$. Thus, those variants in red will be pruned and not executed. Those in blue are within the parameter ranges and may be executed in the next iteration. Note that from Figure 7, executing either of the blue variants (v_2, v_5) only requires processing S_4, as the previous stages (in green) have already been executed.

There are benefits to using either breadth first or depth first searches and associated pruning procedures. In terms of performance, for the Siple Station Experiment signal chain, BFS can only filter after S_2 and S_3; however, it has the benefit of being able to take advantage of locality better than the DFS. Pruning using DFS can filter the execution of up to all of the stages of entire variants. Thus, DFS is less sensitive to the execution time of the individual stages than BFS (which must execute all variants up to S_2) for the Siple Station Experiment signal chain.

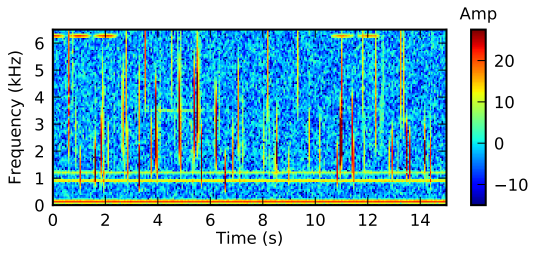
5. Evaluation

5.1. Datasets

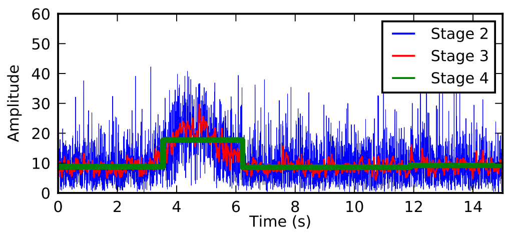
The evaluation is conducted using the transmissions from the Siple Station Experiment. The *Siple-* class consists of two datasets, one containing a signal with an injected transmission (*SipleSignal*) and the other only containing noise (*SipleNoise*). To demonstrate that our signal detection method holds across different transmission conditions, we also conduct an evaluation using the *Synthetic-* class, which consists of a synthetic signal (*SyntheticSignal*) and a noise (*SyntheticNoise*) dataset, which have similar characteristics to the *Siple-* datasets. This is seen in spectrogram format in Figure 8(a) and from the results of the pruning process in Figure 8(b). The synthetic data has a two second pulse, preceded and followed by a short amplitude ramp to resemble the 2 s main tone, and low frequency and pulsed high frequency background signal interference, Gaussian background noise, and Daubechies wavelets simulating the occurrence of sferics. Amplitude and noise properties are scaled to match the observed, real data, with signal amplitudes around 30 dB, signal interference between 40 dB and 150 dB, and the standard deviation for the gaussian noise at 100. We utilize these two datasets to demonstrate that our fitness metric discerns between signal and noise. All datasets consist of 375,000 data elements, corresponding to 15 seconds of data.

5.2. Experimental Methodology

We implemented multithreaded implementations using OpenMP in C++ and compiled using the O3 compiler optimization flag. The executions occur on up to 16 cores of dedicated 2.40 GHz Intel Xeon E5-2630 v3 processors with 20 MB L3 cache. Response times are averaged over 3 trials unless otherwise noted.



(a)



(b)

Figure 8: (a) Example of the synthetic data used in this study shown using a spectrogram (time-frequency amplitude) representation, created to exhibit similar noise and signal characteristics to the real Siple data. (b) Example visualization of the synthetic data after applying the pruning process. The narrowband signal amplitude time series after applying filter stages 2,3,4 are shown in blue, red, and green respectively. This particular filter parameter configuration produced the maximum fitness score for detecting the signal.

5.2.1. Evaluation Scenario 1 (E1)

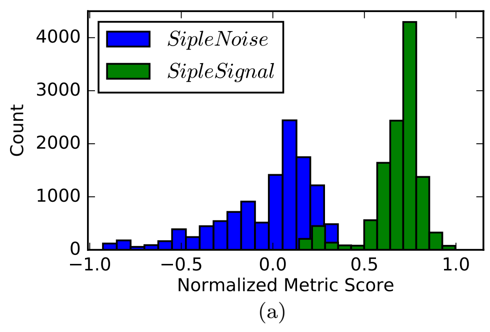
The selection of the parameter values depends on domain-specific knowledge, and we use ranges found in the literature [9, 5, 4]:

- Stage 1: Lightning filter, $\{0.5, 0.6, \ldots, 2.0\}$, $|S_1| = 16$.
- Stage 2: Remez filter, Number of Taps: $\{100, 200, 300\}$, Cut-off and roll-off frequency (both take the same values): $\{100, 150, 200\}$. Permuting all combinations of these 3 parameters gives $|S_2| = 3^3 = 27$.
- Stage 3: Median filter, $\{7, 9, \ldots, 23\}$, $|S_3| = 9$.
- Stage 4: Regression tree, $\{1, 2, 3\}$, $|S_4| = 3$.

Stage 4 generates a fitness function score. If we evaluate all possible combinations of parameters without pruning the search, we obtain a total of $|S_1||S_2||S_3||S_4| = 11,664$ parameter combinations, or pipeline variants. Note that the output dataset size after downsampling in S_2 is 6,049 (from the original size of 375,000).

5.2.2. Evaluation Scenario 2 (E2)

As mentioned in Section 2.2, the performance of the stages is a function of the input size. We vary the degree of downsampling to assess how input size may degrade performance by downsampling less aggressively than that outlined in $E1$ at S_2, where the dataset size becomes 31,250 after downsampling. This is 417% larger than $E1$. Furthermore, the S_3 window size is adjusted linearly and becomes: $\{35, 45, \ldots, 115\}$, $|S_3| = 9$. This adjustment to the window size is done to preserve the desired smoothing characteristics in the data.



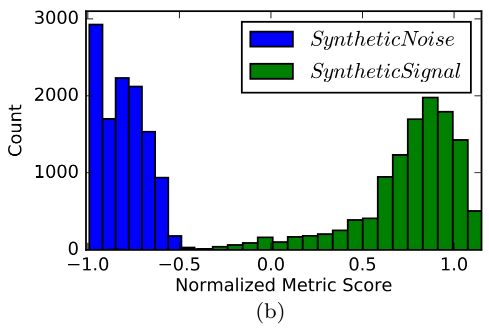


Figure 9: Histograms of normalized fitness scores for filter parameter searches in evaluation scenario 1 of (a) *SipleNoise* in blue and *SipleSignal* in green, and of (b) *SyntheticNoise* in blue and *SyntheticSignal* in green.

5.3. Evaluation Metric Validity

Figure 9 (a) plots the histograms of the metric scores (fitness) of the real-world *SipleSignal* and *SipleNoise* datasets in the context of exhaustively executing all of the variants in scenario $E1$. The bimodal distribution indicates that the metric separates the signal and noise. Figure 9 (b) shows a similar plot, but for the *Synthetic-* datasets, which have a more pronounced separation between the signal and noise cases in the bimodal distribution. Furthermore, as the metric values span a significant range in the signal and noise datasets, we conclude that the range of parameters selected for the experimental evaluation is not biased towards finding good metric values when searching for the best parameters.

5.4. Multicore Scalability Analysis

We execute all of the variants outlined in $E1$ (an exhaustive search), using up to 16 threads. In the breadth-first case, the threads are assigned to an individual variant at each stage. In the depth-first case, threads are assigned entire variants (all 4 stages) to process (e.g., v_1, in green, in Figure 7). In all that follows, both searches use the OpenMP guided schedule to assign loop iterations to threads as it was experimentally found to achieve the best load balancing. Figures 10 and 11 show the response times for the breadth-first and depth-first searches, respectively, on both *SipleSignal* and *SyntheticSignal* datasets. We observe that the response times for the real and synthetic datasets are nearly identical, as the performance of the pipeline is independent of the data characteristics. Furthermore, from the two figures, we see that the response time of breadth-first is slightly less than depth-first across all threads, yielding a speedup of $\sim 12\times$ and $\sim 9.6\times$ using 16 cores, respectively. The breadth-first approach better preserves locality over the depth-first approach.

Figure 12 shows the response time by stage for the breadth-first search for 1 and 16 threads and corresponding speedup by stage. S_1 has 16 parameters and takes negligible time (thus has minimal speedup). S_2 requires the majority of the execution time. This is partially because S_2 takes the entire input data

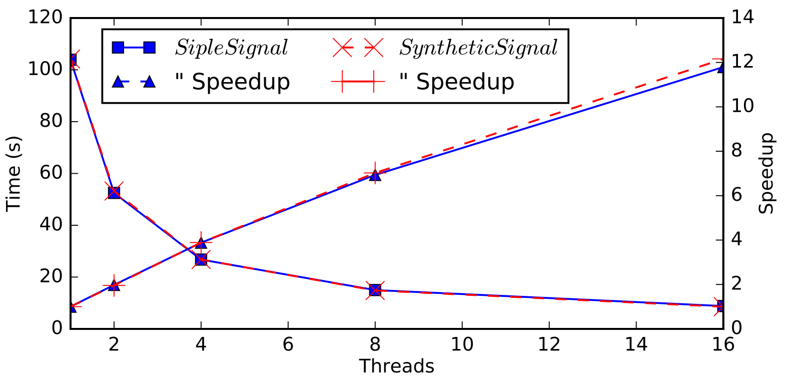


Figure 10: Response time vs. number of threads using exhaustive BFS for *SipleSignal* and *SyntheticSignal* datasets in scenario $E1$. The right vertical axis shows the speedup relative to the single-threaded implementation.

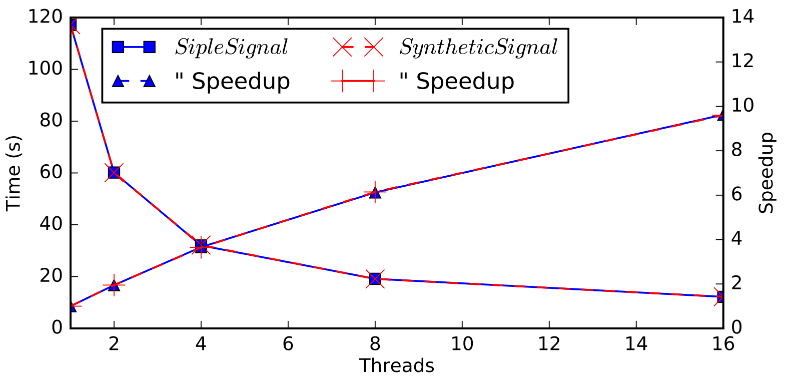


Figure 11: Same as Figure 10, but for DFS.

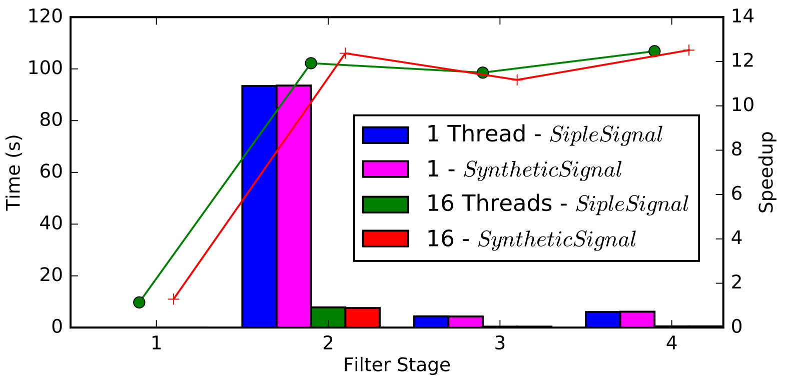


Figure 12: Response time profile breakdown of the breadth-first approach based on the filter stage and the number of threads, and the resulting speedup for both *SipleSignal* and *SyntheticSignal* datasets in scenario $E1$. The bar plot shows the response time for the given stage for 1 and 16 threaded implementations, while the lines show the resulting speedup for each stage (*SipleSignal* in green and *SyntheticSignal* in red).

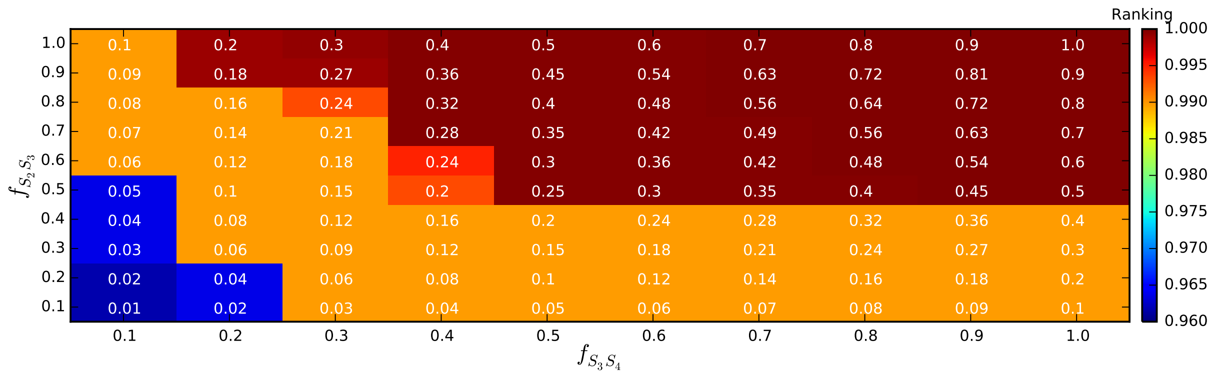


Figure 13: A heatmap displaying the fraction of variations retained between Stage 2 and 3 (y-axis), and between Stage 3 and 4 (x-axis) for breadth-first pruning using *SipleSignal* on scenario $E1$. The cell color indicates the ranking of the best fitness score found for the given ratio of retained data (a ranking of 1 indicates that the best variant was found). The white inset text describes the resulting total fraction of S_4 evaluations for each pair of retained ratios.

($n = 375,000$) and downsamples it for the following stages (Section 2.2.1), thereby reducing the size of the input data in S_3 to $n = 6,049$. Despite large variation in processing times between stages, there is a $\sim 12\times$ speedup across S_2–S_4, indicating that the overall speedup is not diminished by a particular stage.

5.5. Breadth-First Pruning

The breadth-first strategy prunes the pipeline between stages S_2 and S_3, and S_3 and S_4. Figure 13 shows a heatmap of the fraction of data retained between stages and the ranking of the best variant found for $E1$. If we only retain 10% of the data between both filtering points in the pipeline (blue, bottom left corner), we can still obtain the parameter set that has a metric value within the top 96% of all variants. Aggressively filtering may significantly improve responses times. Although, given the response time discrepancy between stages (Figure 12), the large degree of downsampling decreases the ability of breadth-first to reduce large computational loads in $E1$ as filtering occurs after S_2. Pipelines with less downsampling or even computational loads across stages would significantly decrease response times with the breadth-first strategy.

5.6. Depth-First Pruning

Unlike the breadth-first pruning, the depth-first pruning narrows the search space by examining the parameters across all stages, thus it can prune executing S_2, which is a drawback of the breadth-first approach for the Siple Station Experiment signal processing chain. Figure 14 (a) plots the response time vs. the metric ranking (a value of 1 corresponds to the best metric value obtained through an exhaustive search) with 16 threads for *SipleSignal* on $E1$. Because the depth-first search heuristic filters as a function of the variants that have already been executed, it is sensitive to the order that the variants are processed.

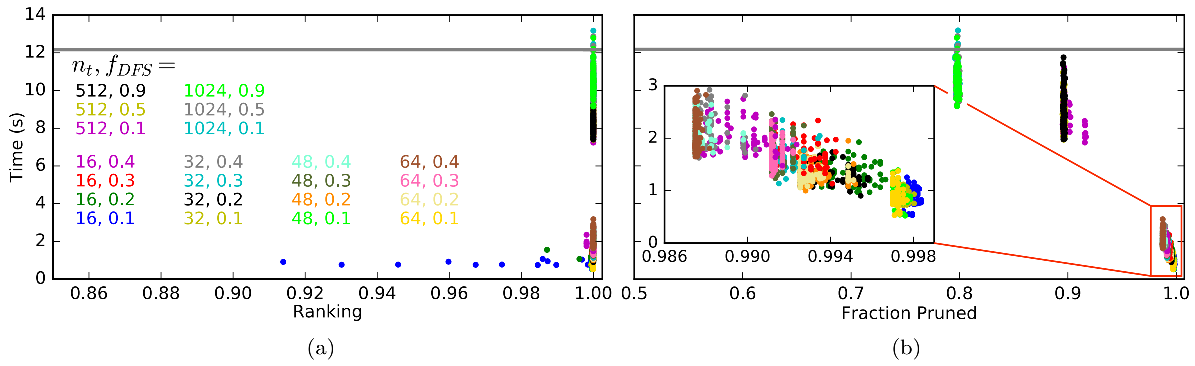


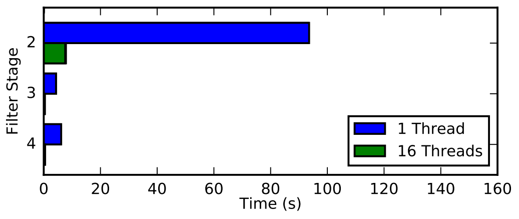
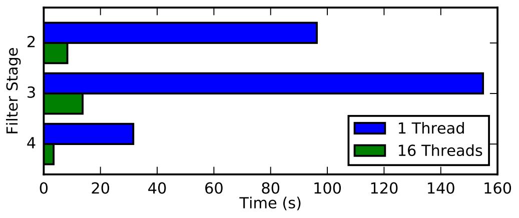
Figure 14: A comparison of response time versus (a) the metric ranking of a search and (b) the fraction of S_4 evaluations that are pruned for 24 different DFS parameter combinations on $E1$. The inset plot shows the faster run time and higher fraction discarded region in greater detail. Colors are indicated in the corresponding text that describe that parameters n_t, the number of variants executed before filtering, and f_{DFS}, the fraction of n_t kept that define the parameter ranges used by the heuristic. The horizontal gray line corresponds to the the exhaustive search time (see text for details).

Therefore, we generate a schedule that randomly assigns variants to threads. In Figure 14 (a), we execute 100 randomly generated schedules (1 time measurement per point) and show their distribution as a function of response time and metric ranking for the two depth-first filter parameters (n_t and f_{DFS}). n_t controls how often the depth-first technique prunes after evaluating a given number of variants. The points converge to a ranking of 1, and thus the 100 measurements are plotted on top of each other in Figure 14 (a). We see a trade-off between the best metric value obtained and the response time. Interestingly, examining the blue dots, we can obtain the parameters that yield a metric value in the top 91% in $<$ 2s. Figure 14 (b) plots the response time vs. the fraction of variants that are pruned. The plot clearly shows that aggressive pruning (those with $n_t \leq 64$) that prune over $\gtrsim$ 98% of variants significantly reduces response time (Figure 14 (b), inset). The horizontal lines in Figure 14 show the response time of the exhaustive search. The exhaustive search outperforms some of the configurations as thread load imbalance may result from randomly assigning variants to threads (which does not occur in the exhaustive search). The random assignment of threads to variants is used to demonstrate the robustness of the heuristic. In practice, a mapping of threads to variants that considers the overall parameter distribution will improve filtering efficacy and overcome the load imbalance for certain depth-first configurations. We omit results for other datasets as they yield similar results.

5.7. Comparison of Workload Sizes After Stage 2

The performance of the breadth-first pruning is more sensitive to the computational load at each stage, in comparison to depth-first pruning. We compare scenarios $E1$ and $E2$ where the latter does not aggressively downsample (Section 5.2.2). We examine the performance of the breadth-first and depth-first strategies as follows: 1) an exhaustive search; and 2) aggressively prune with the following configurations for the

Table 1: Comparison of $E1$ (upper) and $E2$ (lower) scenarios on *SipleSignal*. $E2$ downsamples less than $E1$ at S_2. The response time profiles of the BFS exhaustive search are shown (the breakdown for DFS by stage is similar, as illustrated in the comparison of exhaustive response times). S_1 has been omitted in the figures due to negligible response time. The rank of the best metric found is shown for the aggressive searches, where the DFS response time and metric are the mean of 100 randomized trials.

	BFS Time (s) (Best Ranking)	DFS Time (s) (Best Ranking)	Stage Response Time Distribution
Scenario $E1$			
Exhaustive (1 thread)	103.91	117.16	
Exhaustive (16 threads)	8.81	12.17	
Aggressive (16 threads) BFS: $f_{S_2S_3} = f_{S_3S_4} = 0.1$ DFS: $n_t = 16$, $f_{DFS} = 0.1$	7.76 (0.962)	0.846 (0.993)	
Scenario $E2$			
Exhaustive (1 thread)	283.0	298.17	
Exhaustive (16 threads)	25.83	27.10	
Aggressive (16 threads) BFS: $f_{S_2S_3} = f_{S_3S_4} = 0.1$ DFS: $n_t = 16$, $f_{DFS} = 0.1$	9.73 (0.957)	0.925 (0.994)	

breadth-first and depth-first approach (shown in Figure 13, blue, bottom left, and Figure 14, blue dots). Breadth-first parameters: $f_{S_1S_2} = 1$, $f_{S_2S_3} = f_{S_3S_4} = 0.1$; and depth-first parameters: $n_t = 16$, $f_{DFS} = 0.1$.

The bar plots in Table 1 compares these scenarios on $E1$ and $E2$ on *SipleSignal*. The output of S_2 generates a larger dataset in $E2$ than $E1$; therefore, the response time is greater in S_3 in $E2$. The exhaustive search executes all variants, thus finding the best set of parameter values. In the exhaustive search, BFS slightly outperforms DFS in both $E1$ and $E2$. Comparing the parallel exhaustive (16 threads) and parallel aggressive searches on the breadth-first approach, the response time decreases by 13.5% ($E1$), and 165% ($E2$), and for the depth-first, these values are 1338% ($E1$) and 2839% ($E2$). Both searches obtain greater performance gains over the parallel exhaustive search when there are larger computational workloads at S_3 and S_4. However, the depth-first outperforms the breadth-first technique, both in performance and metric value obtained (unless using an exhaustive search).

6. Performance Modeling

We advance performance models for the breadth- and depth-first searches in the supplementary material. We accurately predict the response time for different pipeline filtering configurations. For instance, when aggressively filtering, the breadth-first performance model is only 2.6 s less than the measured time of 97.07 s.

7. Related Work and Discussion

We have used BFS and DFS to optimize the set of parameters for detecting a signal. We used a tree-based representation of the search space that allows for a parallel search strategy, and for data reuse between stages. In this work, using the enumerated search across parameter sets typically yields good parameter values in low-dimensionality search spaces. However, global search strategies could be employed [14, 15].

Other parameter searches are applicable as well. A randomized parameter search generally outperforms a grid search for high (32) dimensions and was applied to neural networks [14]. Our work is effectively a parameter search on 4 dimensions, and thus may not benefit from the randomized search. Furthermore, highly iterative approaches such as genetic algorithms [16] or gradient descent methods [17] may find a good set of parameters. From our experiments it is not clear that gradient descent would quickly converge to a unique global maxima using the fitness function, which is a common pitfall of that class of methods [18]. Furthermore, iterative algorithms are less conducive to the high throughput parallel processing technique utilized in this work.

8. Siple Detection Results

We focused on a single MDIAG transmission at 6/23/1986 7:01:00 UT, *SipleSignal*, and a single noise dataset at 7:01:15 UT, *SipleNoise*. We apply the parallel processing parameter search to the 50-minute experiment with 100 transmissions of the MDIAG format, from 7:00:00 UT to 7:50:00 UT. Each transmission (at 0 s and 30 s) and noise interval (at 15 s and 45 s) are datasets similar to *SipleSignal* and *SipleNoise*. Figures 10 and 11 show that the performance of our pruning approaches is independent of the data properties across this set of MDIAG transmissions, which is why we do not show the performance for each individual transmission in the 50-minute experiment. We show a histogram of the best metric score using the parameters outlined in $E1$ for 200 signal and no signal datasets in Figure 15. A metric score of around 0.8 clearly separates the signal and noise, successfully enabling detection of MDIAG transmissions.

While 9 signal datasets score below 0.8 and place within the same range as the noise datasets, a closer examination of each of these datasets show that no transmission was received. For 4 of these signals, the experiment logs note that the scheduled transmissions could not be conducted, which matches well with our detection result. However, the other 5 signals occur at regular intervals, potentially indicating transmitter or experiment troubles that are not in the logs. These results demonstrate how a properly designed fitness function clearly separates signal transmissions and noise. This can be extended to detect other transmission

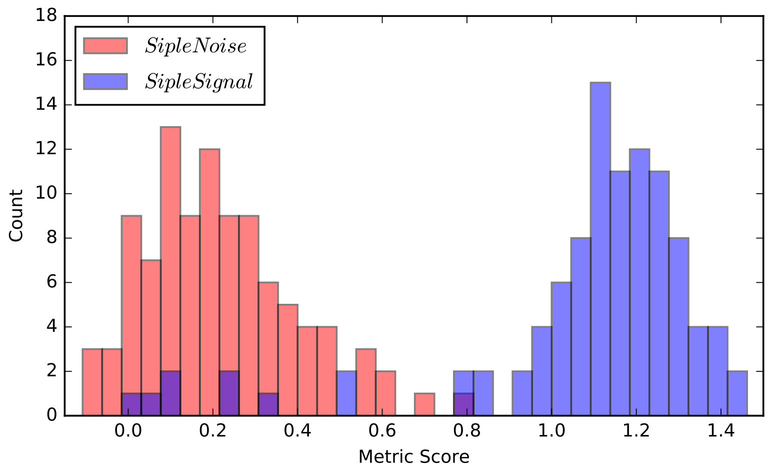


Figure 15: Histogram of the fitness function scores for 100 *SipleSignal*-like signal datasets (blue) and 100 *SipleNoise*-like noise datasets (red) from the experiment at 6/23/1986 7:00:00 − 7:50:00 UT.

formats for analysis, by altering the evaluation window, and also be used as a diagnostic for evaluating the magnetospheric amplification of naturally generated signals, such as banded chorus or hiss.

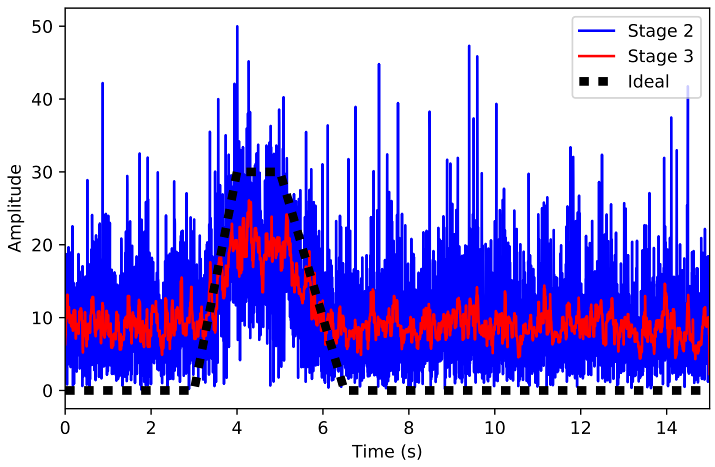


Figure 16: Comparison of the results of the pruning process for the synthetic data with the ideal, noiseless signal. The narrowband signal amplitudes resulting from filter stages 2 and 3 are the same as in Figure 8. The ideal narrowband signal without noise is plotted in black.

We evaluate how well the detected result recovers the original signal. Examining the results from our *SyntheticSignal* dataset, we compare the outputs of Stage 3 with the ideal, noiseless signal, as shown in Figure 16. We observe that the detected pulse and its timing matches well with the ideal signal. The amplitude of the pulse is attenuated, which is expected due to the presence of noise and the filtering operations, and the noise floor is also higher.

9. Conclusion

This work advances novel automated detection of narrowband signals in the Siple Station Experiment. We improve the detection of signals transmitted through a noisy environment with complex physical interactions by attaching an automated search process to the respective parameterized signal processing workflows.

In addition to improving the effectiveness and scalability of this search, our approach enables reuse of intermediate evaluations among computations exploring alternative parameter values in parallel. Using the heuristics and signal evaluation criteria, pruning the search space determines filter parameters that do not yield a good signal detection. Our pruning procedure allows domain scientists to control the balance of signal recovery quality and algorithm response time. Respectable speedups are achieved, with execution times being reduced from > 100 s to < 1 s. This progress offers new prospects for real-time monitoring of transient signals in the space environment. Furthermore, the parameter values obtained in our optimization could be mapped to the properties of the magnetospheric plasma in future studies to characterize longer term changes in magnetospheric conditions.

This work directly applies to "Siple-type" signals, where the signal of interest is narrowband and with well-defined temporal durations. However, as the pipeline search process is independent of the particular pipeline, modifications to pipeline by adjusting either the pipeline parameter ranges or by altering the stages in the pipeline may extend the applicability of this work. Naturally generated structured magnetospheric noise such as banded chorus or hiss could be explored in this manner. Similarly, careful consideration of the appropriate pipeline, pipeline parameters, and evaluation window could extend its potential utility in other geoscience domains as well, such as in volcanic inflation event detection [19], groundwater storage fluctuations [20], monsoon prediction [21], and air temperature variability studies [22], which similarly face the challenge of extracting signals of interest from time series data and the associated large parameter search space problem in multi-stage processing workflows.

Acknowledgment

We acknowledge support from NSF ACI-1442997. We thank Frank Lind for useful comments and suggestions on the manuscript.

[1] V. Harid, M. Gokowski, T. Bell, U. S. Inan, Theoretical and numerical analysis of radiation belt electron precipitation by coherent whistler mode waves, Journal of Geophysical Research: Space Physics 119 (6) (2014) 4370–4388. `doi:10.1002/2014JA019809`.

[2] V. Harid, M. Gokowski, T. Bell, J. D. Li, U. S. Inan, Finite difference modeling of coherent wave amplification in the Earth's radiation belts, Geophysical Research Letters 41 (23) (2014) 8193–8200. `doi:10.1002/2014GL061787`.

[3] A. V. Streltsov, M. Golkowski, U. S. Inan, K. D. Papadopoulos, Propagation of whistler mode waves with a modulated frequency in the magnetosphere, Journal of Geophysical Research 115. `doi:10.1029/2009JA015155`.

[4] J. D. Li, V. Harid, M. Spasojevic, M. Gokowski, U. S. Inan, Preferential amplification of rising versus falling frequency whistler mode signals, Geophysical Research Letters 42 (2) (2015) 207–214.

[5] J. D. Li, et al., Analysis of magnetospheric ELF/VLF wave amplification from the Siple Transmitter experiment, Journal of Geophysical Research: Space Physics 119 (2014) 1837 – 1850. doi:10.1002/2013JA019513.

[6] R. A. Helliwell, J. P. Katsufrakis, VLF Wave Injection Into the Magnetosphere From Siple Station, Antarctica, Journal of Geophysical Research 79 (16).

[7] J. D. Li, Nonlinear Amplification and Generation of Very Low Frequency Waves in the Near-Earth Space Environment, Ph.D. thesis, Stanford University (2015).

[8] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, Classification and regression trees, CRC press, 1984.

[9] R. K. Said, Accurate and Efficient Long-Range Lightning Geo-Location Using a VLF Radio Atmospheric Waveform Bank, Ph.D. thesis, Stanford University (2009).

[10] J. McClellan, T. Parks, A unified approach to the design of optimum FIR linear-phase digital filters, IEEE Transactions on Circuit Theory 20 (6) (1973) 697–701. doi:10.1109/TCT.1973.1083764.

[11] J. A. Robinson, Software design for engineers and scientists, Elsevier, 2004.

[12] C. E. Leiserson, T. B. Schardl, A Work-efficient Parallel Breadth-first Search Algorithm (or How to Cope with the Nondeterminism of Reducers), SPAA, 2010, pp. 303–314.

[13] R. Sedgewick, Algorithms in C++, Part 5: Graph Algorithms-3/E.

[14] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research 13 (Feb) (2012) 281–305.

[15] M. J. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in: Advances in optimization and numerical analysis, Springer, 1994, pp. 51–67.

[16] M. Mitchell, An introduction to genetic algorithms, MIT press, 1998.

[17] K. C. Kiwiel, Convergence and efficiency of subgradient methods for quasiconvex minimization, Mathematical programming 90 (1) (2001) 1–25.

[18] G. Bekey, M. Gran, A. Sabroff, A. Wong, Parameter optimization by random search using hybrid computer techniques, in: Proceedings of the November 7-10, 1966, fall joint computer conference, ACM, 1966, pp. 191–200.

[19] J. D. Li, C. M. Rude, D. M. Blair, M. G. Gowanlock, T. A. Herring, V. Pankratius, Computer aided detection of transient inflation events at Alaskan volcanoes using GPS measurements from 2005-2015, Journal of Volcanology and Geothermal Research 327 (2016) 634 – 642. doi:http://dx.doi.org/10.1016/j.jvolgeores.2016.10.003.

[20] F. Silverii, N. D'Agostino, M. Mtois, F. Fiorillo, G. Ventafridda, Transient deformation of karst aquifers due to seasonal and multiyear groundwater variations observed by GPS in southern Apennines (Italy), Journal of Geophysical Research: Solid Earth 121 (11) (2016) 8315–8337. doi:10.1002/2016JB013361.

[21] V. Stolbova, E. Surovyatkina, B. Bookhagen, J. Kurths, Tipping elements of the Indian monsoon: Prediction of onset and withdrawal, Geophysical Research Letters 43 (8) (2016) 3982–3990. doi:10.1002/2016GL068392.

[22] N. Jajcay, J. Hlinka, S. Kravtsov, A. A. Tsonis, M. Palu, Time scales of the European surface air temperature variability: The role of the 78 year cycle, Geophysical Research Letters 43 (2) (2016) 902–909. doi:10.1002/2015GL067325.