Handling Side-Information for Data Compression of Radar Pulse Trains

J. Andrew Johnson and Mark L. Fowler[†]

Department of Electrical and Computer Engineering
State University of New York at Binghamton
Binghamton, NY 13902

ABSTRACT

Previously a method has been proposed for a high-performance compression method designed expressly for compressing intercepted radar pulse trains for the purpose of locating the transmitter. The method relies on gating pulses, putting them into a pulse matrix and then using the singular value decomposition (SVD) to compress the signal data. The pulse gating step, however, generated a sequence of varying integers as side information and the previous study did not provide a means for efficiently coding this side information; although it did given an empirically generated estimate of the entropy of this side information.

This paper reformulates the pulse gating method to generate a different form of side information that is much easier to compress. It is shown that this new side information can be simply and efficiently coded at a rate that compares favorably with the empirically-estimated entropy of the original method.

It is shown that this new pulse gating method requires the solution of an integer linear programming problem and several standard methods are first considered. It is shown that the large number of constraints in the original formulation can be significantly reduced by replacing the constraint set by its convex hull; simple rules for identifying the convex hull are given. However, even with these reductions the execution time for these methods can be prohibitive at very large pulse counts; furthermore, these methods exhibited numerical precision and convergence problems as the number of pulses increased. Therefore, an efficient non-standard method for solving this integer optimization problem is developed by exploiting characteristics of the objective function. This method solves the pulse gating problem with short execution times that grow negligibly with increasing pulse counts.

Keywords: data compression, side information, integer optimization, singular value decomposition, SVD, emitter location, time-difference-of-arrival, TDOA, frequency-difference-of-arrival, FDOA

1. INTRODUCTION

A common way to locate electromagnetic emitters is to measure the time-difference-of-arrival (TDOA) and the frequency-difference-of-arrival (FDOA) between pairs of signals received at geographically separated platforms.[1,2,3] The measurement of TDOA/FDOA between these signals is done by coherently cross-correlating the signal pairs.[2,3] This requires that the signal samples of the two signals are available at a common platform, which is accomplished by compressing and then transferring the signal samples over a data link from one platform to the other (see Figure 1). Various general data compression approaches have been proposed,[4–9] although they have not been designed to fully exploit the characteristics of radar signals. For the radar case, the singular value decomposition (SVD) has been used[10] to exploit redundancy between pulses in a radar pulse train in order to achieve significant levels of compression ratio that exceed what is possible using the previously proposed general methods.[4–9]

Section 2 will briefly review the previously proposed compression method[10] and in particular will discuss the previous problems with handling the side information. Section 3 reformulates the side information processing to eliminate the difficulties previously encountered and in so doing it is seen that it requires the solution of an integer optimization problem. Section 4 proposes several methods to solve the integer optimization problem and shows that a simple method exists to effectively solve it. Section 5 provides conclusions.

[†] Correspondence: mfowler@binghamton.edu

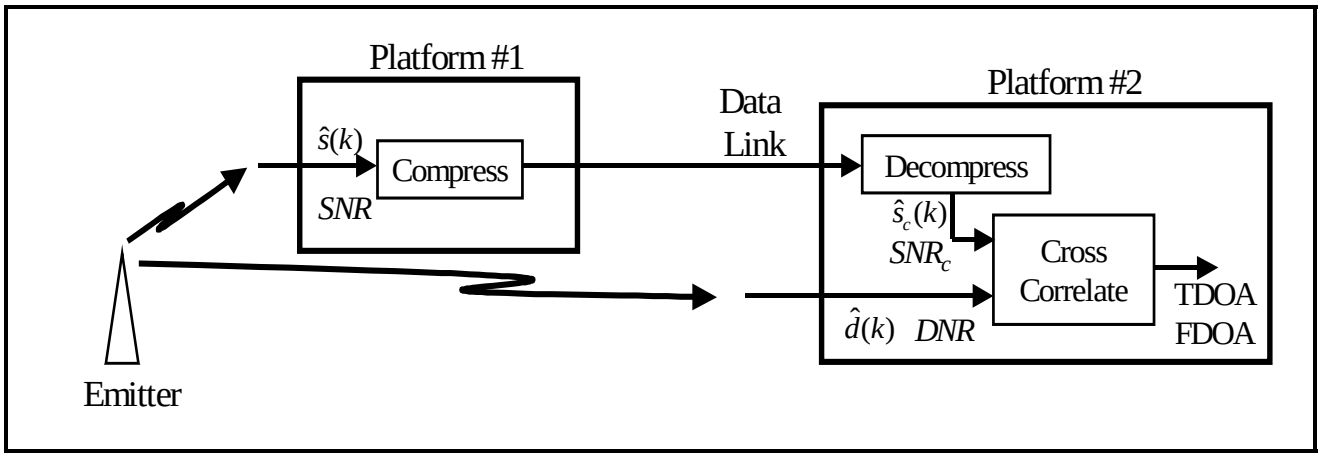

Figure 1: System Configuration for Compression

2. SVD-BASED DATA COMPRESSION FOR RADAR PULSE TRAINS

This section will give an overview of the previously proposed[10] SVD-based method and describe the previous results on handling the side information. Before compression processing, Platform #1 receives and digitizes the radar waveform. It is assumed that the radar waveform contains a sequence of nearly-evenly spaced pulses. Once the signal has been digitized, it was previously proposed to undergo data compression as described in Figure 2. It is clear that the pulses in the waveform, and their relative positions, contain the information required, and the "dead spaces" between the pulses contain no useful information. So, naturally, the first step in the data compression is to remove the unwanted samples between pulses by using pulse gating.

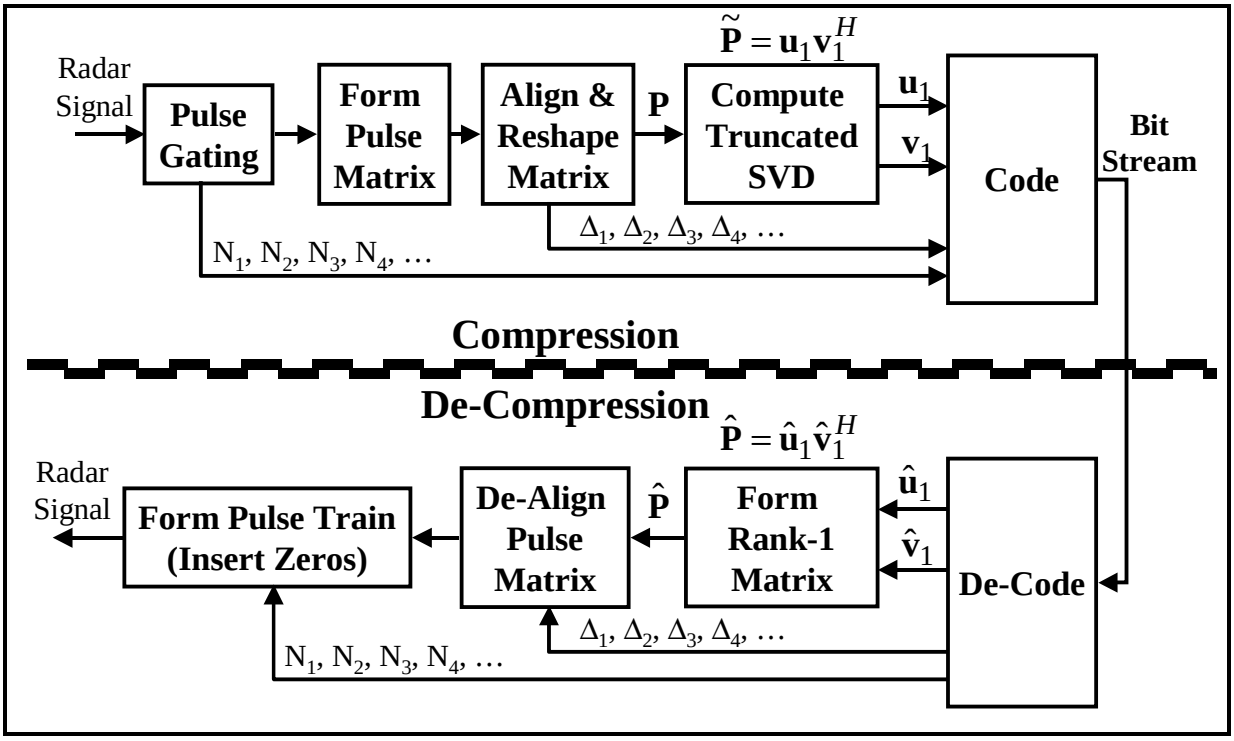

Figure 2: Overview of SVD-Based Compression

The pulse gating step, as previously proposed, is illustrated in Figure 3. By design, a typical radar-interception receiver detects the beginning and end of each pulse and measures the times at which these events occur. These time measurements will be called the measured leading edge (LE) and measured trailing edge (TE); for each detected pulse the number of samples that lie between TE and LE (inclusive) is the measured pulse width. The previously proposed pulse gating method used a constant gate width W that is chosen to be equal to the largest measured pulse width in the pulse train. The gate for each pulse starts at the pulse's measured leading edge, resulting in a varying gate interval G_k between gate starting edges. The samples falling inside each gate are put into the Pulse Matrix, with one pulse per row. Note that W and G_k are integers.

The pulse gating method also determines the number of samples excised after each pulse – this sequence of integers N_1, N_2, N_3, … are called the excision integers and must be sent as side information. It is the fact that the excision integers N_k vary that leads to the previously noted problems in coding the side information.

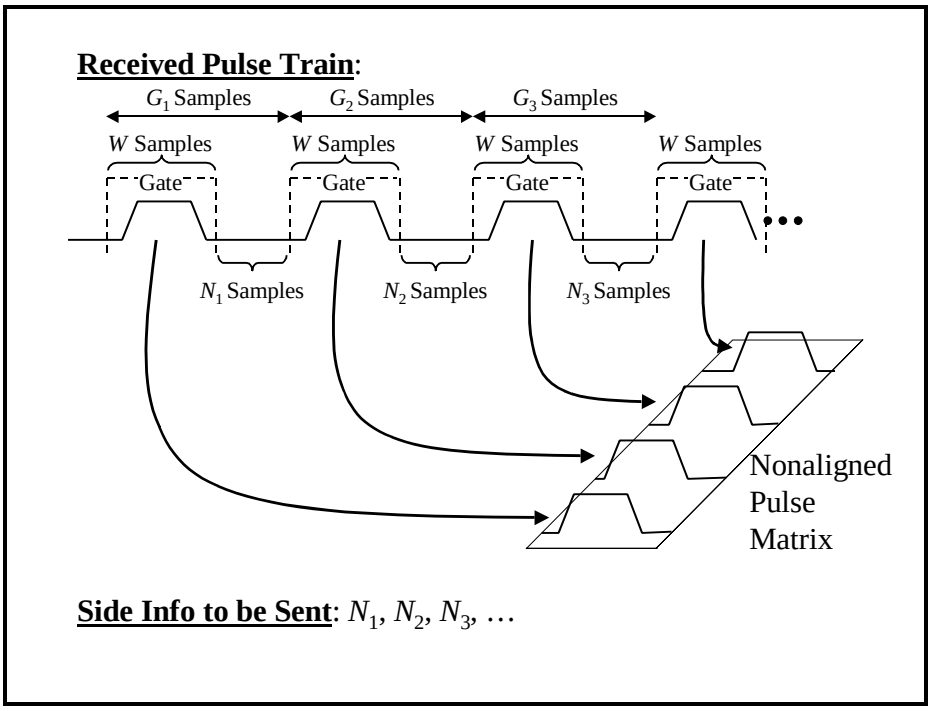

Figure 3: Previously Proposed Pulse Gating and Pulse Matrix Formation

After the Pulse Matrix is formed, the pulses in the rows are aligned using either fractional delay FIR filters or DFT-based processing in order to obtain an Aligned Pulse Matrix that has rank of nearly one[10]. The amount of alignment imparted to each pulse is sent as side information in the sequence Δ_1, Δ_2, Δ_3, … as shown in Figure 2. If desired, the Aligned Pulse Matrix can be reshaped by putting multiple pulses per row – proper reshaping has been shown to maximize the compression ratio[10]. After alignment and reshaping, the resulting rank-nearly-one matrix $\mathbf{P}$ is decomposed using the SVD, which is used create $\tilde{\mathbf{P}} = \mathbf{u}_1 \mathbf{v}_1^H$, an exactly-rank-one approximation to the Aligned Pulse Matrix, where $\mathbf{u}_1$ and $\mathbf{v}_1$ are the left and right singular vectors (respectively) of corresponding to the largest singular value.

Effective methods for coding the singular vectors $\mathbf{u}_1$, $\mathbf{v}_1$ and the alignments Δ_1, Δ_2, Δ_3, … have been determined[10]. However, no explicit method was given for coding the side information of the excision integers N_k coming from the pulse gating. In fact, only an estimate of the number of bits needed to code the excision integers was previously given as $2p+16$ bits, where p is the number of pulses processed. This was determined by empirically computing the entropy of differential forms of the sequence of excision integers N_k for some field-collected radar signals[10].

The goal of the present paper is to better address the issue of coding the side information coming from the pulse gating. Rather than deriving a scheme to compress the previously proposed excision integers N_k, we reformulate the pulse gating problem and derive a new pulse gating method that results in side information that is easier to compress efficiently than the originally proposed excision integers N_k.

3. PULSE GATING PROBLEM FORMULATION

The gating method developed here constrains the gate width W and the gate interval G to be constants – this yields a constant excision integer N to be used between each pair of pulses, which makes the coding of the pulse gating side information trivial. However, it also makes it harder to choose the proper W and G to yield a series of gates that completely encapsulates the series of pulses. The challenge addressed in this paper is to find an effective way to choose a minimal gate width W and the gate interval G; for maximum flexibility and performance we also include an initial gate offset T, thus the first gate starts at index T. We desire a *minimal* gate width to reduce the size of the resulting Pulse Matrix to achieve an effective level of compression.

To properly choose the pulse gating parameters W, G, T we formulate this mathematically. The k^{th} gate is defined according to

$$\begin{aligned} k^{\text{th}}\ \text{Gate Start}: &\quad kG+T \\ k^{\text{th}}\ \text{Gate Stop}: &\quad kG+T+(W-1) \end{aligned} \tag{1}$$

The following conditions must be met for any viable choice of gating parameters W, G, T:

C1. Pulses must not be truncated; all digital samples from every pulse will be placed in the matrix.

C2. The gate width, W, the gate interval, G, and the initial *gate offset*, T, are restricted to be integers (i.e., an integer number of samples).

As presented thus far, the goal is easily met by choosing W fairly large – on the order of the spacing between pulses. But, as mentioned before, to ensure a high compression ratio we must choose the gate width W to be as small as possible; thus, to ensure that condition C1 is satisfied requires explicit attention. Let t_k and w_k be the measured LE and pulse width, respectively, for pulse k. Then condition C1 is satisfied when

$$\left.\begin{aligned} kG+T &\le t_k \\ kG+T+(W-1) &\ge t_k+(w_k-1) \end{aligned}\right\} \quad \text{for } k=0,1,\ldots,p-1 \tag{2}$$

where p is the number of intercepted pulses. We are finally ready to present the problem in the classic optimization form:

$$\begin{aligned} &\text{minimize}: \\ &f(T,G,W)=W \\ &\text{such that for } k=\{0,1,\ldots p-1\} \\ &kG+T \le t_k \\ &kG+T+W \ge t_k+w_k \\ &G,W \ge 0 \\ &T,G,W \quad \text{integers} \end{aligned} \tag{3}$$

To get a better perspective of this minimization problem, Figure 4 shows t_k and (t_k+w_k) vs. k along with two lines of slope G that bound these two sets of points – the bottom line has an intersect of T and the upper line is W above the bottom line. Note that the bottom line touches at least one of the t_k points whereas the upper line touches at least one of the (t_k+w_k) points. On the left side of the figure the corresponding pulse gates are represented by shaded rectangles. Thus, the minimization problem to be solved is to pick integers T, G, W (with G and W non-negative) so that the two lines enclose the two sets of points while minimizing the vertical distance between the lines.

Now, what impact does this have on the coding of the side information? The coding of W is handled as before. The coding of T can be absorbed into the overhead that exists whether compression is used or not – namely that there is some minuscule amount of header information that describes the platform clock time of the first sample sent, which would be the time of the sample at index T. Thus, the only side information that must be sent is G, which would require no more than 32 bits to handle realistic pulse spacing (pulse repetition interval) at typical sampling rates; in fact, fewer than 32 bits could be used for most systems, but allocating an excessive number of bits to this one piece of side information has a negligible impact on the resulting compression ratio. An implicit assumption has been made in the above analysis: there are no missing pulses (i.e., undetected due to low SNR, for example). The impact of this on the specification of the above minimization problem is negligible, but the impact on the amount of side information must be addressed. While there are perhaps more efficient

ways, we propose here to use a bit mask to indicate where there are missing pulses: the bit mask would have a "1" to indicate a pulse is present and a "0" to indicate that a pulse is missing; the occurrence of a missing pulse can be recognized from irregular spacing between LE times. Thus, the length of this bit mask will depend on the probability of a missing pulse P_{mp}. If there are p intercepted pulses, then the expected length of the bit mask is p/P_{mp} bits. Thus, if we consider a rather low, some-what worst case scenario, with a value of $P_{mp} = 0.5$, then the expected number of bits in the bit mask would be $2p$ bits. Thus, we will use $2p + 32$ bits as a rough upper bound on the number of bits needed for the gating side-information, which is only slightly larger than the result conjectured for the original gating approach[10]. Thus, the compression ratio results derived earlier[10] are still valid.

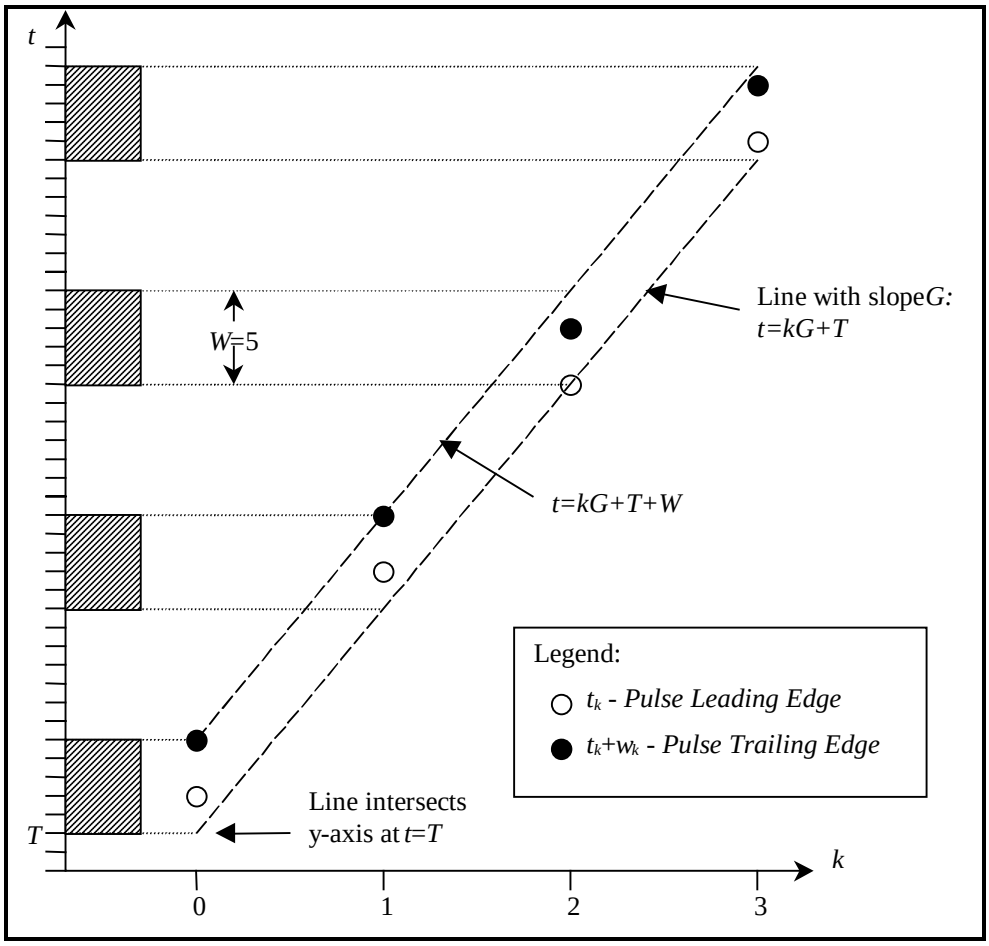

Figure 4: Pulse Leading and Trailing Edge Times vs. Pulse Index *k*

4. OPTIMIZATION METHODS FOR PULSE GATING

From the above problem formulation we see that we have reduced this problem to a traditional constrained linear optimization problem[11] with three exceptions:

E1. The objective function is not an explicit function of all the system variables. However, since W is implicitly a function of G and T by virtue of the constraints, standard methods of solving are still applicable.

E2. T is not restricted to being non-negative, but most standard methods for solving constrained linear optimization problems require all the variables to be non-negative.

E3. The objective variables are restricted to being integers. This makes the problem an integer programming problem[12], which complicates the solution.

In the remainder of this section we consider and compare several means to solve this optimization problem.

1. Constraint Relaxation

As a first step in understanding the path to a solution, we will consider use of the simplex method[11,12]. As mentioned above, exception E1 is safely ignored. Exception E2 will be dealt with by a mathematical device: replacing T by $T_p - T_m$ where $T_p, T_m \geq 0$. It is intended that the simplex method will select only one of T_p and T_m as a basis variable. Exception E3, the integer restriction, prohibits application of the simplex method; however, as an approximation, the integer restrictions can be ignored, and the problem solved in continuous space. The resulting solution can then be rounded to place them in the integer domain, and adjusted to ensure the constraints are met. This intuitive (but often inappropriate) approach can be used, if nothing else, to bound the possible solution space. The simplex method reliably solves this non-integer problem and has the nice feature of placing a lower bound on the value of W. With the integer constraint relaxed, the problem yields a form that can be solved by the simplex algorithm:

$$\begin{aligned} &\text{minimize}: \\ &f(T,G,W) = W \\ &\text{such that for } k = \{0,1,\ldots p-1\} \\ &kG + T \leq t_k \\ &kG + (T_p - T_m) + W \geq t_k + w_k \\ &T_p, T_m, G, W \geq 0 \end{aligned} \tag{4}$$

2. Integer Methods

It is well-known that adaptively adding additional constraints (known as cutting planes) will force standard solution methods to produce integer results[12]. The application of cutting planes can be used to solve the optimization problem with the integer constraints enforced. A cutting plane is simply an additional constraint that reduces the continuous feasible space without removing any discrete points from the feasible set. The dual simplex method is applicable to (4), because it is well-suited to the adaptive application of cutting planes (additional constraints) to solve the integer problem. (The Simplex method is not amenable to the addition of constraints during the algorithm execution.)

3. Convex Hull and Non-Binding Constraint Removal

In the context of the simplex method and the dual simplex method with cutting planes in the integer case, it is desired to have as few constraints as possible in order to reduce the computational complexity. We now show how some constraints can be removed without affecting the solution to the optimization problem.

Consider the lower line $t=kG+T$ in Figure 4. Notice that it passes directly through one of the points defining a pulse leading edge. If it did not, obviously W could be reduced by raising the line until it did pass though one the of the leading edge points. Likewise, it can be shown that the line $t=kG+T+W$ must also pass through at least one of the trailing edge points to be a candidate for optimality. We say that $t=kG+T$ is tangent to the set $\{t_k\}$, and $t=kG+T+W$ is tangent to the set $\{t_k+w_k\}$. Furthermore, it can be observed that there are some leading edge points (e.g. t_1 in Figure 4) for which it is impossible for the line $t=kG+T$ to pass though while adhering to the constraints imposed by their neighbors. In general, only those leading edge points that belong to the lower convex hull of the set $\{t_k\}$ need be considered in the constraint inequalities. Also, only those trailing edge points that belong to the upper convex hull of the set $\{t_k+w_k\}$ need be considered in the constraint inequalities.

To show how a particular point may be eliminated as a constraint, consider three (possibly non-sequential) leading edge points, t_a, t_b, and t_c, where $a < b < c$. Here we show the conditions under which, if the constraints for t_a and t_c are met, the constraint for t_b must also be met, and thus may no longer need be considered. The constraint inequalities for t_a, t_b, and t_c are, respectively:

$$\begin{aligned} aG+T &\leq t_a \\ bG+T &\leq t_b \\ cG+T &\leq t_c \end{aligned} \tag{5}$$

By assumption, the first and third constraints in (5) will be made as strict as possible, by making them equalities, which are then solved for G and T (integer restrictions relaxed):

$$\begin{aligned} aG+T=t_a \quad & G=\frac{t_c-t_a}{c-a} \\ \Rightarrow & \\ cG+T=t_c \quad & T=\frac{ct_a-at_c}{c-a} \end{aligned} \tag{6}$$

Using the right-hand side of (6) in the second constraint $bG+T \le t_b$ shows that the second constraint is guaranteed to be met when:

$$(c-b)t_a+(b-a)t_c \le t_b(c-a) \tag{7}$$

If (7) holds, the constraint associated with t_b is non-binding and may be ignored. A similar conjecture for the pulse trailing edges will show that the constraint associated with (t_b+w_b) is non-binding when:

$$(c-b)(t_a+w_a)+(b-a)(t_c+w_c) \ge (t_b+w_b)(c-a) \tag{8}$$

By repeatedly applying conditions (7) and (8) to the given data it is possible to remove all non-binding constraints. This constraint removal is, in effect, removing points which are not part of the convex hull and in most realistic cases results in a significant reduction in the number of constraints.

4. Unimodal Method

In this subsection we show that it is possible to develop a very efficient solution to this integer linear programming problem that is not based on standard, general purpose linear programming methods. We start by fixing *G* to some constant value, an integer. As stated above, the line $t = kG + T$ must pass though at least one of the leading edge points. If it did not, then *W* could be reduced by incrementing *T* until the line did pass through one of the leading edge points. From this, we find that if *G* is fixed, then *T* can be chosen such that this condition holds. In fact, it leads to this equation for selecting *T*, given *G*:

$$T=\min_k\left[t_k-kG\right] \tag{9}$$

A similar argument shows that for fixed *G* and *T*, *W* can be optimally chosen as:

$$W=\max_k\left[(t_k+w_k)-(kG+T)\right] \tag{10}$$

Note that in (9) and (10), if *G* is an integer, then so are *T* and *W* (since t_k and w_k are integers). Now it's a matter of finding the value of *G* that minimizes *W*. It may well be computationally possible to evaluate *W* for a large number of values of *G*: for each *G* value (9) is solved for *T* and then substituted into (10) which is then solved for *W*. However, it is possible to do even better than that.

It can easily be seen that *W* as a function of *G* is unimodal – where *T* is considered to take on its optimal value for each G as defined in (9). This arises from (4) as follows. Imagine the three dimensional space with axes *W*, *G*, and *T*. The first constraint in (4) specifies a series of vertical planes (i.e., parallel to the *W* axis) and Figure 5 shows the intersection of these constraint planes with the *G*-*T* plane; the shaded area identifies the region satisfying these constraints. From our previous considerations we know that the solution must lie on one of the lines forming the constraint region. The second constraint in (4) places a series of tilted planes above the plane shown in Figure 5, each of which has a slope of –1 with respect to the *T* axis and the k^{th} plane has a slope of –*k* with respect to the *G* axis. The conglomeration of these constraints plane creates a convex downward surface. Thus, *W* as a function of *G* is the values on this convex downward surface as traversed along the conglomeration of the constraint lines in Figure 5, which creates a unimodal function with a single minimum.

A property of a unimodal function is:

$$\text{if } f(x_1)>f(x_2)<f(x_3),\ x_1<x_2<x_3 \text{ then } x_1<x^*<x_3 \quad (x^* \text{ minimzes } f(x)) \tag{11}$$

Thus, if a local minimum is located, it will be the global minimum. Various well-known algorithms (such as bisection method, golden section search, etc.) can now be used to solve the optimization problem.

There are several different methods that can be used to select the initial trial value for *G*. The non-integer simplex method may be used. This tends to locate the optimal value for *G* within a fraction, but the method is complicated. (This method also places a lower bound on *W*). A least-square fit to the slope of the leading edge sample points may be used to estimate *G*. The method is simple, but may require more trials to reach the optimal *G*. A simple slope calculation between the leading edges of the first and last pulse may also be useful.

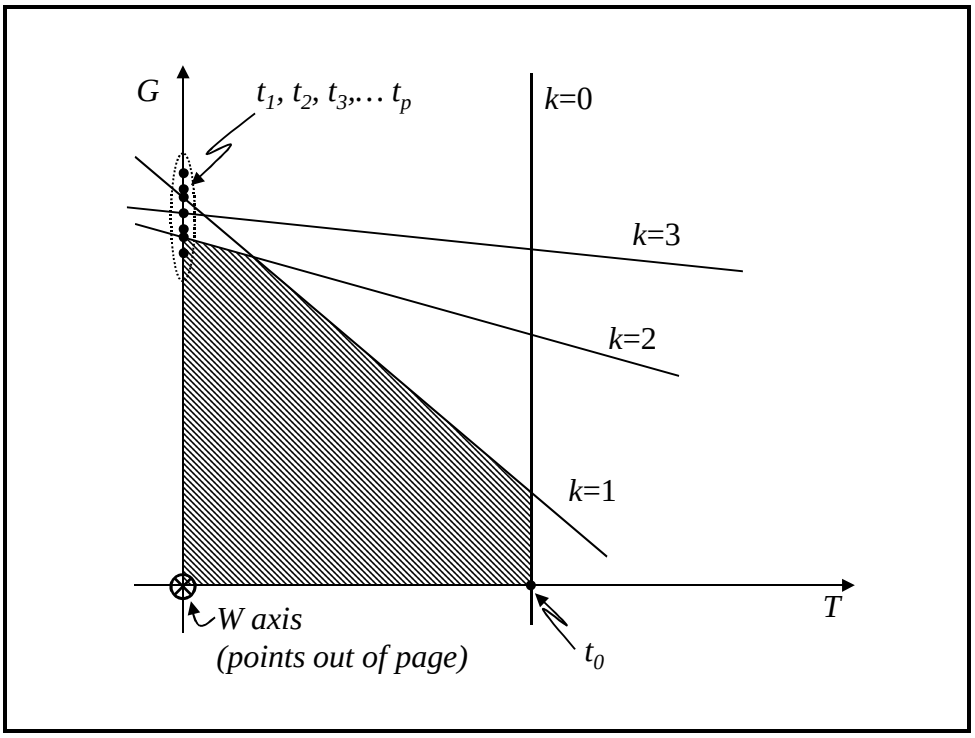

Figure 5: Example of constraints in the *T*-*G* plane

Our so-called unimodal method (called this because it relies on the fact that *W* is a unimodal function of *G*) thus consists of finding an initial estimate for *G* using any one of the methods mentioned and then iteratively refining it by using (9) and (10). Specifically, it consists of getting an initial estimate of *G* (call it G_1) , and then incrementing it by one to get another estimate $G_2 = G_1 + 1$, solving (9) and (10) for each of these *G* estimates to get W_1 and W_2, from this information the unimodality can be used to determine the direction to the minimum of $W(G)$; *G* is then incremented in unit steps until $W(G)$ – checked using (9) and (10) – starts to increase, at which point the optimal *G*, *T*, and *W* have been found. The efficiency of this method depends on the accuracy of the initial estimate of *G*.

5. Comparison of Methods

To evaluate the performance of the optimization approaches discussed above, each was implemented in Matlab. Several variants of the simplex and the dual simplex methods were tried, as well as the unimodal method. Except for cases where the integer constraints were relaxed, all produced the correct result, when a result was produced. (Some convergence problems were encountered with large matrices, which will be discussed below.) Here is a description of each of the methods tested:

Simplex: A standard implementation of the simplex method, with integer constraints relaxed. For "less than or equal to" inequalities, a slack variable was used, and for "greater than or equal to" inequalities, an excess and an artificial variable were used. All problem constraints (except the integer constraints) were explicitly implemented.
Simplex-Hull: As above, a standard implementation of the simplex method, with integer constraints relaxed. A convex hull algorithm was used to removed the non-binding constraints in order to simplify the computations.
Dual: This used the dual simplex method. All "greater than or equal to" inequality constraints were converted to "less than or equal to" inequalities, and the problem solved from outside the feasible solution space. Integer constraints were relaxed.
Dual-Hull - This used the dual simplex method also, but non-binding constraints were removed prior to setting up the constraint matrix. Integer constraints were relaxed.

Dual-Hull-Cut: This used the dual simplex method with non-binding constraints removed. Cutting planes were dynamically added to the constraint matrix as the problem was solved to enforce the integer constraints.
Unimodal Method: This implemented the min/max approach by first choosing a value of G and then finding the associated optimal values for T and W. A simple iterative line search method (constant unity step size) was used to locate the optimal value of G. The initial value for G was selected by computing the slope between the first and last leading edge points, and rounding.

Table 1 shows various performance metrics of each method described above. It should be noted that the execution times should be interpreted on a relative basis since MATLAB programs that contain loop are notorious for long run times compared to what is possible using other programming languages. For each method, tests were run at a variety of pulse counts, and the leading and trailing edge times determined by randomly perturbed uniformly spaced times (in order to simulate the effect of errors in estimating pulse edge times). The performance measures (except for "Fail Count") in each row of the table are obtained by averaging over the 1000 trials. Figure 6 shows the execution times plotted against the pulse count for the six methods; it is important to note the slow growth of execution time for the unimodal method developed here, making it the most applicable in real-time, large pulse-count applications.

Clearly, the superior performer is the unimodal method, which is not only fast, but simple to implement. However, there are some interesting insights provided by the other methods. It is an obvious advantage to remove as many non-binding constraints as possible. For a large number of pulses, solutions using the convex-hull constraint removal technique out-performed the otherwise equivalent methods by two orders of magnitude. Note that the non-cut convex-hull methods significantly reduce the number of constraints and that as the number of pulses increases the number of constraints increases only negligibly.

Comparing the simplex method and the dual-simplex method, the latter seems to have some advantages. For larger pulse counts, the execution times tended to be shorter than the simplex method. However, the iteration counts were far lower than for the simplex method. This seems to be an inconsistency; a higher correlation between the iteration count and execution time was expected. Perhaps there is significant overhead in constructing the initial tableau. In any case, the dual-simplex method is preferred over the simplex method for three reasons:

1. The dual-simplex method has reduced complexity, as artificial variables are not required (for this problem, since there are no equalities).
2. The dual-simplex method reaches a solution in fewer iterations than the simplex method (for this constraint-heavy application)
3. It is well-suited for the dynamic addition of constraints, to solve the integer problem using cutting planes.

However, both the simplex and dual-simplex methods are highly susceptible to limited numerical precision problems. It was originally the intent to perform additional tests using an even larger number of pulses, but both methods began breaking down. It should be noted, however, that no concerted attempt was made to address precision difficulties in the Matlab implementation. The addition of constraints during the integer solutions aggravated the precision problem. The unimodal method is not expected to exhibit problems at higher pulse counts.

In this study, advantages and limitations of several standard methods for solving linear programs were brought to light. Ironically, it was an ad hoc method that was found to be superior, showing that standard methods, while providing a good basis for theory and comparison, leave room for innovative solutions.

5. CONCLUSIONS

We have revisited the pulse gating issue that is embedded in a data compression scheme recently proposed for compressing intercepted radar pulse trains. The original formulation of the pulse gating problem led to a situation that required a large amount of side information to be coded but no clear method was available to efficiently code this side information. However, an empirical entropy calculation estimated that in theory it could be coded with $2p+16$ bits, where p is the number of pulses in the pulse train. The goal in this paper was not to find an efficient way to code this side information, but rather to reformulate the pulse gating problem to yield side information that yields to simple, yet efficient coding schemes. We reformulated the pulse gating problem to be constrained to have a fixed gate width W as well as a fixed gate interval G; an initial gate offset T was introduced to yield a more efficient gating scheme. By doing this we were able to show that the side information can be coded using no more than $2p+32$ bits, where the extra 16 bits over the previous estimated side information size is negligible. The problem then is to optimally choose the gating parameters W, G, and T.

Table 1: Performance Metrics for Tested Methods

Method	Pulse Count	Execution Time, avg. (seconds)	Number of Iterations[‡], avg.	Number of Constraints[§], avg.	Fail Count[**]
Simplex	5	0.004180	7.8	10.0	
	10	0.010710	13.8	20.0	
	25	0.046733	29.8	50.0	
	50	0.471090	55.3	100.0	
	100	3.435210	106.5	200.0	
Simplex-Hull	5	0.006700	5.8	6.1	
	10	0.008680	7.3	7.7	
	25	0.014000	9.3	9.6	
	50	0.021150	10.6	11.1	
	100	0.034050	12.2	12.6	
Dual	5	0.009340	5.9	10.0	
	10	0.018670	6.7	20.0	
	25	0.073600	7.2	50.0	
	50	0.256400	7.6	100.0	
	100	1.266250	7.7	200.0	
Dual-Hull	5	0.007190	5.9	6.2	
	10	0.009840	6.6	7.6	
	25	0.015100	7.3	9.6	
	50	0.021860	7.6	11.1	
	100	0.033340	7.8	12.6	
Dual-Hull-Cut	5	0.009560	7.0	7.2	
	10	0.015100	9.5	10.2	
	25	0.060910	20.1	19.6	
	50	0.347812	43.6	37.7	8
	100	6.781081	101.3	78.8	326
Unimodal	5	0.000390	8.7	*N/A*	
	10	0.000270	5.6	*N/A*	
	25	0.000270	3.6	*N/A*	
	50	0.000280	3.0	*N/A*	
	100	0.000330	3.0	*N/A*	

[‡] For simplex and dual simplex methods, the iteration count is the number of tableaus generated after the initial formulation. For the unimodal method, it is the number of values of *G* for which the objective function was evaluated during the line search.

[§] This is the final number of constraints defined in the tableaus. For the integer (cutting-plane) implementation, it includes the additional constraints added during the solving.

[**] The Fail count is the number of times (of the 1000 trails) in which the dual-simplex method failed to converge, due to numerical round-off; no entry listed when count was zero.

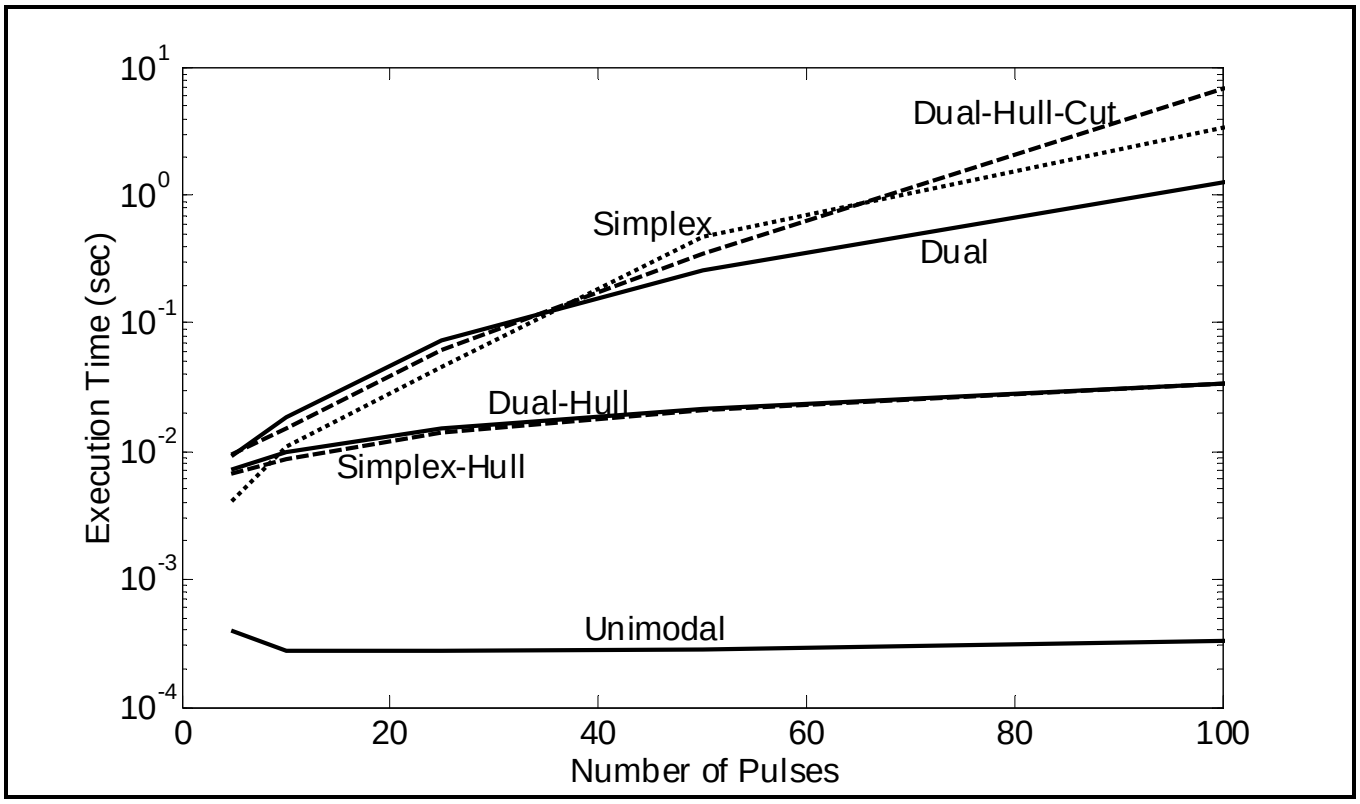

Figure 6: Plot of Execution Time vs. Number of Pulses for Various Methods

We have shown that the pulse gating selection problem can be expressed as an integer linear programming problem. If the integer constraints are relaxed, then the simplex method or dual simplex method can be used to solve it. Otherwise, the dual simplex method in combination with plane-cutting techniques can be used. Both these methods are relatively complex, and although many constraints can potentially be removed (the ones that don't lie in the convex hull), even simpler methods are desired. We then developed a non-standard method. We showed how to effectively reduce the problem to a univariate objective function. Although the new objective function was non-linear, it was unimodal, and solvable using simple, well-known minimization techniques. Some methods for choosing the starting point for the search were suggested. Computer-based tests showed that this unimodal method yielded a rapidly executing method whose execution time grew *very* slowly with increasing pulse count, making it viable choice for real-time applications having large pulse counts.

6. REFERENCES

1. P. C. Chestnut, "Emitter location accuracy using TDOA and differential doppler ," *IEEE Trans. Aero. and Electronic Systems*, vol. AES-18, pp. 214-218, March 1982.
2. S. Stein, "Differential delay/doppler ML estimation with unknown signals," *IEEE Trans. Sig. Proc.*, vol. 41, pp. 2717 - 2719, August 1993.
3. S. Stein, "Algorithms for ambiguity function processing," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-29, pp. 588 - 599, June 1981.
4. D. J. Matthiesen and G. D. Miller, "Data transfer minimization for coherent passive location systems," Report No. ESD-TR-81-129, Air Force Project No. 4110, June 1981.
5. G. Desjardins, "TDOA/FDOA technique for locating a transmitter," US Patent #5,570,099 issued Oct. 29, 1996, held by Lockheed Martin.
6. M. L. Fowler, "Coarse quantization for data compression in coherent location systems," *IEEE Trans. Aero. and Electr. Systems*, vol. 36, no. 4, pp. 1269 – 1278, Oct. 2000.

7. M. L. Fowler, “Data compression for TDOA/DD-based location system,” US Patent #5,991,454 issued Nov. 23, 1999, held by Lockheed Martin.
8. M. L. Fowler, “Data compression for emitter location systems,” Conference on Information Sciences and Systems, Princeton University, March 15-17, 2000, pp. WA7b-14 – WA7b-19.
9. M. L. Fowler, “Decimation vs. quantization for data compression in TDOA systems,” in *Mathematics and Applications of Data/Image Coding, Compression, and Encryption III*, Mark S. Schmalz, Editor, Proceedings of SPIE Vol. 4122, pp. 56 – 67, San Diego, CA, July 30 – August 4, 2000.
10. M. L. Fowler, “Data compression via pulse-to-pulse redundancy for radar emitter location,” in *Mathematics and Applications of Data/Image Coding, Compression, and Encryption IV*, Mark S. Schmalz, Editor, Proceedings of SPIE Vol. 4475, San Diego, CA, July 29 – August 3, 2001, pp. 1 –12.
11. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, San Diego, CA: Academic Press, 1981.
12. L. R. Foulds, *Combinatorial Optimization for Undergraduates*, New York: Springer-Verlag, 1984.