5.4 SMALL POLYGON COMPRESSION FOR INTEGER COORDINATES

Abhinav Jauhri, Martin Griss & Hakan Erdogmus*
Carnegie Mellon University, Moffett Field, California

ABSTRACT

We describe several polygon compression techniques to enable efficient transmission of polygons representing geographical targets. The main application is to embed compressed polygons to emergency alert messages that have strict length restrictions, as in the case of Wireless Emergency Alert messages. We are able to compress polygons to between 9.7% and 23.6% of original length, depending on characteristics of the specific polygons, reducing original polygon lengths from 43-331 characters to 8-55 characters. The best techniques apply several heuristics to perform initial compression, and then other algorithmic techniques, including higher base encoding. Further, these methods are respectful of computation and storage constraints typical of cell phones. Two of the best techniques include a "bignum" quadratic combination of integer coordinates and a variable length encoding, which takes advantage of a strongly skewed polygon coordinate distribution. Both techniques applied to one of two "delta" representations of polygons are on average able to reduce the size of polygons by some 80%. A repeated substring dictionary can provide further compression, and a merger of these techniques into a "polyalgorithm" can also provide additional improvements.

1. INTRODUCTION

Geo-targeting is widely used on the Internet to better target users with advertisements, multimedia content, and essentially to improve the user experience. Such information helps in marketing brands and increasing user engagement. Scenarios also exist where geo-targeting at a given time becomes imperative for specific sets of people for information exchange, thereby contributing to problems of network congestion and effectiveness. Emergency scenarios are a quintessential example where people in the affected area need to be informed and guided throughout the duration of an emergency. To address this, Wireless Emergency Alerts (WEA) is a nation-wide system for broadcasting short messages[1] (currently 90 characters, similar to SMS messages) to all phones in a designated geographic area via activation of appropriate cell towers. The area is typically identified by a polygon, though currently many operators use rather coarse-grained targeting (such as to a whole county).

Our research group has developed and evaluated improved geo-targeting technology. For testing purposes, we ran trials of the new system, called by us WEA+. We are using SMS (and WiFi) to simulate cell broadcast, and also have on campus an experimental cell system, CROSSMobile [10], that supports true cell broadcast on an unused GSM frequency to cell phones with a special SIM card.

In order to do this, we included some compressed polygon representation as part of the short message text, expected to be feasible in both current 90 WEA character messages and even more effective in future implementations of WEA that allow longer messages, or use multiple messages. We have available to us a corpus of 11,370 WEA messages sent out by the National Weather Service (NWS) [13][2]. The polygons in the NWS corpus range from 4-24 points, with a size ranging from 43-331 characters. Since WEA messages are broadcast to thousands of people, and it is believed that adding a polygon to more precisely define the target area is critical, it is essential to be able to compress typical WEA polygons to fit within the current or anticipated future WEA message length, leaving room for meaning text as well. In this paper, we explore several ways of substantially compressing such polygons using heuristics and standard algorithms. Our techniques provide better compression for almost all polygons in the corpus than standard algorithms.

The compression problem we are tackling here is quite different from that described in most other published research on polygon compression [1,6]. They typically are dealing with a large number of inter-connected polygons in a 2D or 3D representation of a surface or solid, and thus are compressing a large number of polygons at the same time. Many of these polygons share common points and edges, which can be exploited in the compression; in our case, we have a single, relatively small polygon to compress, and so can not amortize items such as a dictionary of common points.

In designing our techniques, we have looked at the typical distribution of coordinate values, polygon sizes and character string length of original uncompressed polygons, shown in Figures 2a and 2b. Many of these charac-

*Corresponding author address: Abhinav Jauhri, Carnegie Mellon University, ECE Department, Moffett Field, CA 94035; email: ajauhri@cmu.edu

[1] `https://www.fema.gov/wireless-emergency-alerts`

[2] These messages were sent out by the NWS in 2012, 2013, and through December, 2014

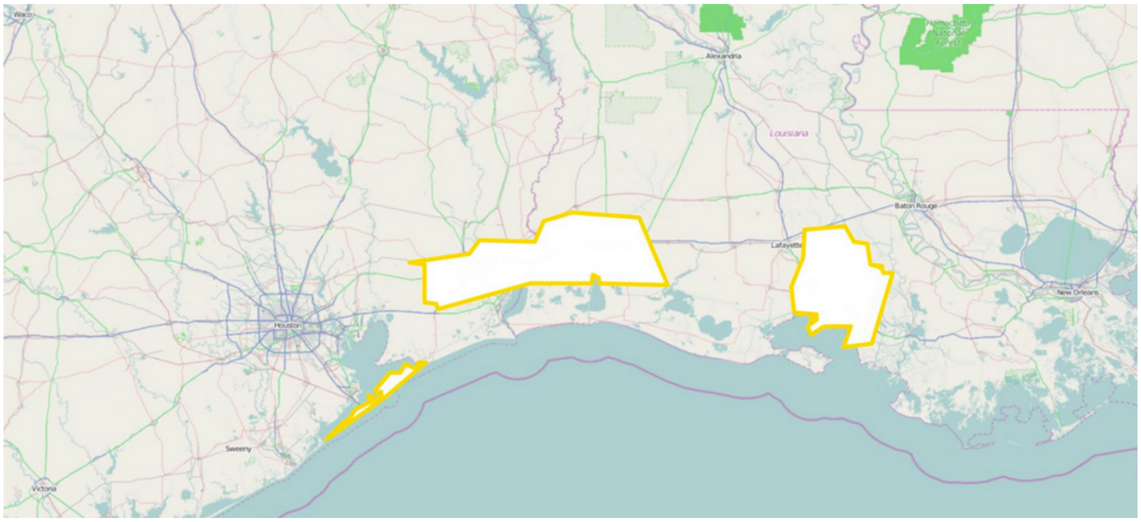


Figure 1: A map showing 3 polygons (yellow border). The NWS dataset has both convex and non-convex polygons

teristics have motivated our transformations of the original polygon.

In the NWS corpus the observed range of GPS coordinates, covering a significant portion of the USA, is:

$$(X, Y) = (17.67, -159.32) \text{ to } (48.84, -64.56)$$

We use three types of transformations on the numeric strings representing a polygon. The first group exploits heuristics, redundancies and patterns in the GPS coordinates, substantially simplifying and reducing the number of numeric characters in most polygon coordinates. The next set of transformations encodes the now simplified coordinates in a higher base, B, such as all alpha-numeric characters, and then applies arithmetic and character string operations to further compress each polygon. The final set of transformations uses the statistical nature of the entire set of polygons to further compress some polygons.

We evaluated our corpus of polygons with combinations of the following compression techniques:

1. Purely heuristic, using deltas and fixed or variable length fields

2. Encoding in a higher base

3. Using a repeated substring dictionary to replace most frequent occurring substrings

4. Arithmetic operations to combine values

5. Arithmetic encoding, an entropy based compression technique

6. Other standard algorithms - LZW [12], *7zip*, *gzip*, *Huffman* [9] and *Golomb* [8]

A key constraint is to compress the polygon into a string of characters that are acceptable via SMS or cell broadcast and specifically to the gateways we use to send an SMS via email for our pilot trials. To compress the set of decimals (or bits) representing GPS coordinates, we use a base B representation, avoiding characters that are questionable. This is discussed further in the practical considerations section. Base $B = 62$ is a convenient choice since it uses only alphanumeric characters [0-9a-zA-Z]. Using a higher base such as $B = 70$ or $B = 90$ uses more characters and will improve the overall compression, and changes the trade-off between techniques. Briefly, our paper explores numerous techniques of compression on different transformations, or manipulations to the original polygon.

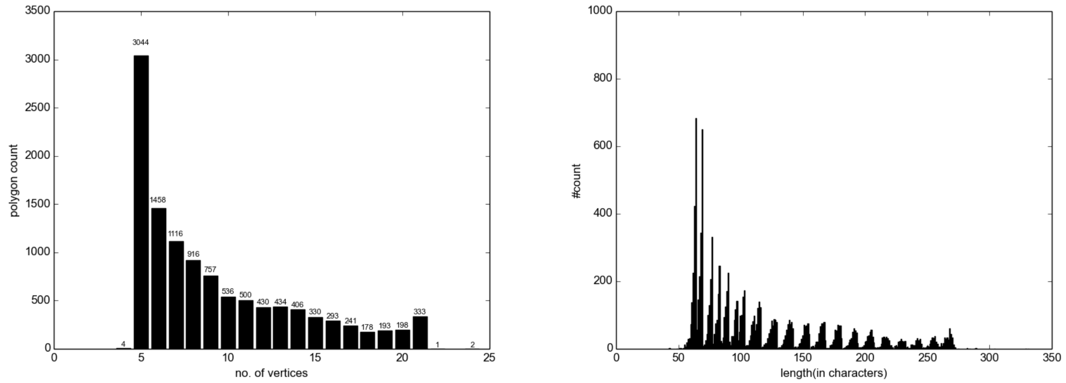
2. HEURISTIC APPROACHES

We have combined several heuristics, motivated by analysis and discovered experimentally to work well. The following describes our current techniques. We start with an original N point polygon, given as an ordered finite sequence in $\mathbb{R}^2$:

$$O = [X_1, Y_1, \ldots, X_N, Y_N]$$

where X_i, Y_i are GPS coordinates in decimal degrees.

In the NWS corpus, $N \in [4, 24]$, even though the NWS standard allows polygons of up to 100 points. The original uncompressed polygon length of 43 to 331 characters includes $2N - 1$ separating commas, $2N$ periods and N minus signs. Since X_i is typically *dd.dd* and Y_i is typically $-$*dd.dd* or $-$*ddd.dd*, the total length of the original polygon string O is:



(a) Distribution of polygon vertices

(b) Distribution of original polygon lengths

Figure 2: Distribution of NWS data

$$\begin{aligned} len(O) &<= \underbrace{4N}_{X_i} + \underbrace{5N}_{Y_i} + \underbrace{2N-1}_{\text{commas}} + \underbrace{2N}_{\text{periods}} + \underbrace{N}_{\text{minuses}} \\ &<= 14N - 1 \end{aligned} \tag{1}$$

There are several steps we take to successively compress the polygon. Initially we perform three simplifications and transformations to the original set of coordinates O. The first transformation converts all coordinates to positive integers to yield, O'. The second finds the minimum x-coordinate and y-coordinate and takes the difference with every vertex ($T^{\Delta min}$). The third transformation considers the difference of consecutive coordinates (T^{Δ}).

Step 1: Starting with polygon O, round all numbers to 2 (or 3) decimals precision, convert to integers to drop the decimal point, and switch sign of Y_i in USA, so both X_i and Y_i are positive integers, to produce O' [3]:

$$X_i = int(100 * X_i); Y_i = -int(100 * Y_i)$$

Outputting these with $2N-1$ separating commas gives a length of at most $11N-1$ characters as opposed to the original $14N-1$ (refer Eq. 1).

Step 2: Compute $X_{min} = \inf_i X_i$, $Y_{min} = \inf_i Y_i$.

Step 3: Compute deltas for all coordinates:

$$dX_i = X_i - X_{min}$$

$$dY_i = Y_i - Y_{min}$$

where dX_i and dY_i are non-negative integers.

Step 4: Compute deltas for X_{min} and Y_{min} from a chosen "origin", origin (X_0, Y_0):

$$dX_{min} = X_{min} - X_0$$

$$dY_{min} = Y_{min} - Y_0$$

We found $(1600, 6000)$ most effective (see Figures 4a and 4b).

Step 5: Since these are closed polygons, drop the last point (X_N, Y_N) which is a duplicate of the first point, producing a shorter set of coordinates:

$$\begin{aligned} T^{\Delta min} &= [dX_{min}, dY_{min}, dX_1, dY_1, \ldots, \\ &\ldots, dX_{N-1}, dY_{N-1}] \end{aligned}$$

Steps for the second transformation T^{Δ} are very similar:

Step 1: Round all numbers to 2 (or 3) decimals precision, convert to integers to drop the decimal point, and switch signs for Y_i:

$$X_i = int(100 * X_i); Y_i = -int(100 * Y_i)$$

Step 2: Compute deltas for all coordinates:

$$\Delta X_{i+1} = X_{i+1} - X_i$$

$$\Delta Y_{i+1} = Y_{i+1} - Y_i$$

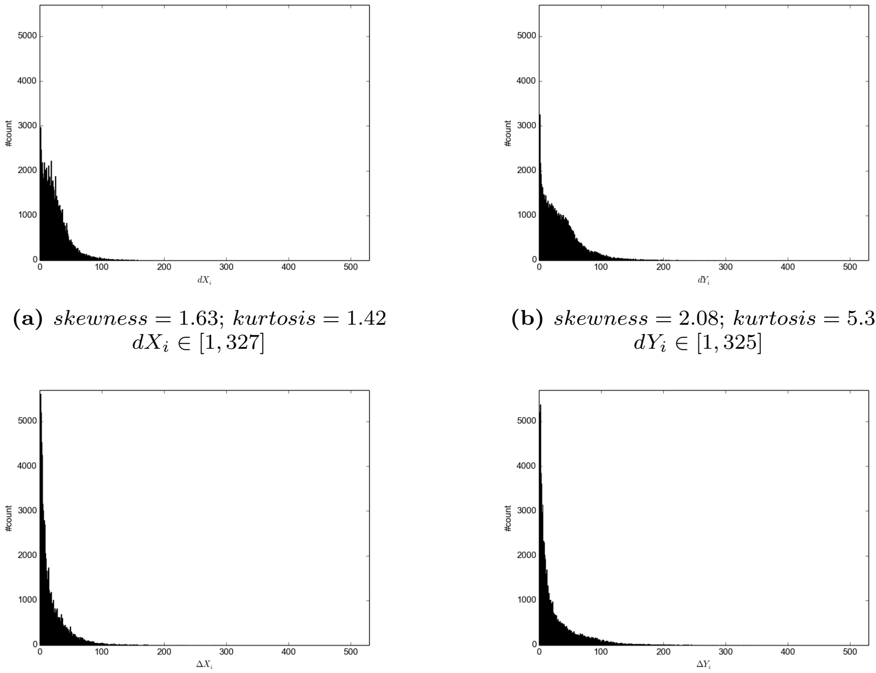
Step 3: Compute deltas for X_1 and Y_1 from a the chosen "origin", (X_0, Y_0) :

$$\delta X_1 = X_1 - X_0$$

$$\delta Y_1 = Y_1 - Y_0$$

Here again we used $(1600, 6000)$ as "origin".

[3] Following Mike Gerber of the NWS [7]



(a) *skewness* $= 1.63$; *kurtosis* $= 1.42$ $dX_i \in [1, 327]$

(b) *skewness* $= 2.08$; *kurtosis* $= 5.34$ $dY_i \in [1, 325]$

(c) *skewness* $= 4.34$; *kurtosis* $= 21.17$ $\Delta X_i \in [1, 361]$

(d) *skewness* $= 5.2$; *kurtosis* $= 31.56$ $\Delta Y_i \in [1, 524]$

Figure 3: Positive skewness in the deltas dX_i(3a) and dY_i(3b). Heavy tails and peakedness define positive kurtosis [3]. *kurtosis* $= 85.25$ and 139.76 for dX_i and dY_i respectively if 14354 cases where $dX_i = 0$ and 13838 cases where $dY_i = 0$ are included. Kurtosis not very high with inclusion of $\Delta X_i = 0$ and $\Delta Y_i = 0$ since their distributions have uniform fall off.

Step 4: Many of the Δs are negative integers which causes problems for the compression techniques discussed below. Therefore, every ΔX_i or ΔY_i element e will be converted as follows:

$$e = \begin{cases} 2e, & \text{if } e >= 0 \\ -2e - 1, & \text{if } e < 0 \end{cases}$$

Step 5: Drop the last point (X_N, Y_N) which is a duplicate of the first point, producing a shorter set of coordinates:

$$T^{\Delta} = [\delta X_1, \delta Y_1, \Delta X_2, \Delta Y_2, \ldots, \Delta X_{N-1}, \Delta Y_{N-1}]$$

.

Note that in addition to dropping the last point, we also save an additional point, since we start from δX_1 and δY_1 rather than the additional X_{min}, Y_{min} in the $T^{\Delta min}$ transformation. This form is particularly interesting, since the distribution of its ΔX_i has a skewed but less peaked shape to that of the dX_i in $T^{\Delta min}$, but with a longer tail (See Figure 3).

These heuristics already produce a substantial compression because of the limited ranges and skewed distribution of the delta polygon coordinates (dX_i, dY_i), and (ΔX_i, ΔY_i) and of the starting points (dX_{min}, dY_{min}), and (δX_1, δY_1), shown in Figures 3, 4. While the effectiveness of the various compression techniques work well because of these range and skew characteristics, many of the techniques are not strongly dependent on the specifics, and thus many would work well even for a somewhat different set of polygons.

It is important to note that in both forms of the delta transformations T we have two different sets of integers to compress, with distinctly different ranges and distributions: the single starting point pair of dX_{min} and dY_{min} for $T^{\Delta min}$ (or δX_1 and δY_1 for T^{Δ}) and the $N-1$ pairs dX_i, dY_i for $T^{\Delta min}$ (or $N-2$ pairs ΔX_i and ΔY_i for T^{Δ}). We will thus treat them separately to get the best results.

As we shall see, for the NWS corpus two of the techniques are best, but we describe several others and the sub-transformations since some of these might perform better for other sets of polygons.

As a very first step to compress polygon, $T^{\Delta min}$ or T^{Δ} could be directly encoded as a comma-delimited string:

$$T_c^{\Delta min} = dX_{min}\bullet,\bullet dY_{min}\bullet,\bullet dX_1\bullet, \bullet\, dY_1\bullet,\bullet\ldots\bullet\, dX_{N-1}\bullet,\bullet dY_{N-1} \quad (2)$$

$$T_c^{\Delta} = \delta X_1\bullet,\bullet\delta Y_1\bullet,\bullet\Delta X_2\bullet, \bullet\,\Delta Y_2\bullet,\bullet\ldots\bullet\,\Delta X_{N-2}\bullet,\bullet\Delta Y_{N-2} \quad (3)$$

Symbol $\bullet$ is used to denote string concatenation. Figure 5 shows the distribution of lengths using this transformation. Since all deltas, dX_i, dY_i are less than 350, (and ΔX_i, ΔY_i are less than 550) each of these deltas can be encoded in at most three decimal digits, *ddd*, while dX_{min} or δX_1 will take at most four digits, *dddd*, and dY_{min} or δY_1 at most five digits, *ddddd*. The lower bound on the length of $T_c^{\Delta min}$ is thus:

$$\begin{aligned} len(T_c^{\Delta min}) &>= \underbrace{1}_{dX_{min}} + \underbrace{1}_{dY_{min}} + \underbrace{(N-1)}_{dX_i} + \underbrace{(N-1)}_{dY_i} + \underbrace{2N-1}_{\text{commas}} \\ &>= 4N-1 \end{aligned} \quad (4)$$

and $4N-5$ for T_c^{Δ}, though it is very rare in the NWS corpus for dX_{min} and dX_{min} to be encodable in a single digit.

The upper bound on the length of $T_c^{\Delta min}$ is:

$$\begin{aligned} len(T_c^{\Delta min}) &<= \underbrace{4}_{dX_{min}} + \underbrace{5}_{dY_{min}} + \underbrace{3(N-1)}_{dX_i} + \underbrace{3(N-1)}_{dY_i} \\ &\quad + \underbrace{2N-1}_{\text{commas}} \\ &<= 8N+2 \end{aligned} \quad (5)$$

and $8N-6$ for T_c^{Δ}.

We can eliminate all commas by using fixed three digit fields for each delta, padded with zero, four digits for dX_{min} and δX_1 and five digits for dY_{min} and δY_1 to get a significant improvement (Figure 5b) over $len(T_c^{\Delta min})$ or $len(T_c^{\Delta})$, called $T_f^{\Delta min}$ or T_f^{Δ}:

$$\begin{aligned} len(T_f^{\Delta min}) &= \underbrace{4}_{dX_{min}} + \underbrace{5}_{dY_{min}} + \underbrace{3(N-1)}_{dX_i} + \underbrace{3(N-1)}_{dY_i} \\ &= 6N+3 \end{aligned} \quad (6)$$

and $6N-3$ for T_f^{Δ}, which is in between the upper and lower bounds on $T_c^{\Delta min}$ or T_c^{Δ}. The skew in the delta values results in $T_c^{\Delta min}$ being usually better than $T_f^{\Delta min}$.

3. HIGHER BASE ENCODING

For the next set of compression techniques we use a convenient base (B) transformation, $H_B(\cdot)$, to represent the encoded polygon. Encoding a large number in base 62 using alphanumeric characters [0-9A-Za-z] rather than just numeric digits [0-9] significantly reduces overall polygon string lengths. For example, one base 62 "bigit" *b* can represent an integer up to 61, two base-62 "bigits", *bb*, can represent an integer up to 3843, while three "bigits" *bbb* can represent an integer up to 238327, and so on. So each delta can be represented in one or two base ($B >= 62$) characters. A higher base, such as base 70, can represent even larger integers in fewer characters: a single base-70 "bigit" *b* can represent an integer up to 69, two base-70 bigits *bb* can represent an integer up to 4899, and three base-70 bigits *bbb* can represent integers up to 342999. Likewise, dX_{min} can be encoded in two base 70 bigits and dY_{min} can be encoded in two or three base 70 bigits. Because of the choice of values for X_0 and Y_0, the restricted ranges and skew of dX_{min} and dY_{min} allow most to pack within two base-70 characters. For our analysis and experiments of compression techniques, we will use $B = 70$ with all alphanumeric characters and some allowable special characters used in SMS.

String $H_{70}(T_c^{\Delta min})$ is the transformation of the comma-delimited values in base-70. For example, if $T_c^{\Delta min} = [9, 60, 70]$, then $H_{70}(T_c^{\Delta min}) =$"9,y,10". The upper bound on the length is then:

$$\begin{aligned} len(H_{70}(T_c^{\Delta min})) &<= \underbrace{2}_{dX_{min}} + \underbrace{3}_{dY_{min}} + \underbrace{2(N-1)}_{dX_i} \\ &\quad + \underbrace{2(N-1)}_{dY_i} + \underbrace{2N-1}_{\text{commas}} \\ &<= 6N \end{aligned} \quad (7)$$

and the lower bound will be:

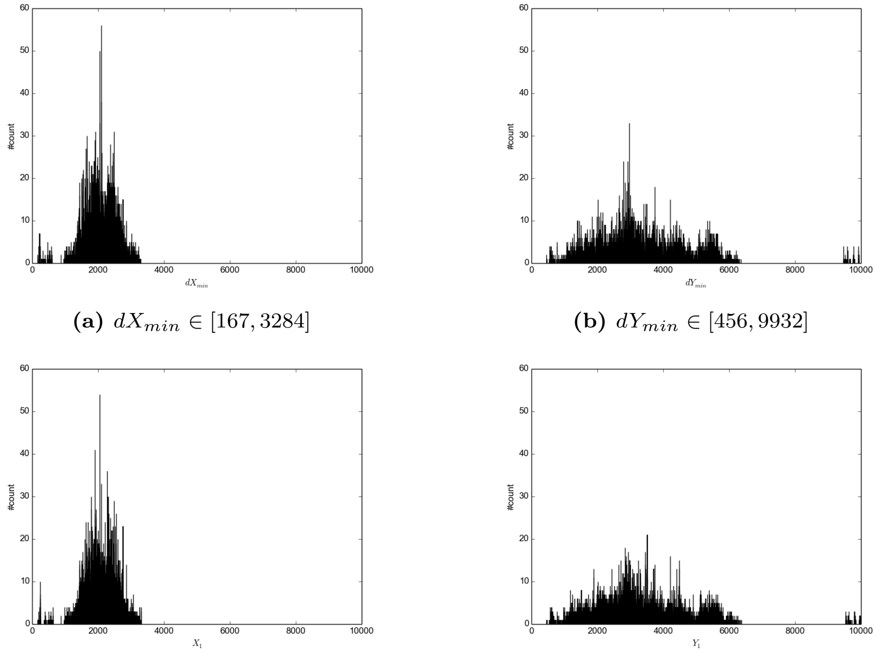
$$\begin{aligned} len(H_{70}(T_c^{\Delta min})) &>= \underbrace{1}_{dX_{min}} + \underbrace{1}_{dY_{min}} + \underbrace{(N-1)}_{dX_i} \\ &\quad + \underbrace{(N-1)}_{dY_i} + \underbrace{2N-1}_{\text{commas}} \\ &>= 4N-1 \end{aligned} \quad (8)$$

Similarly, fixed length coding $T_f^{\Delta min}$ in base-70 will have length strictly $4N+1$:

$$\begin{aligned} len(H_{70}(T_f^{\Delta min})) &= \underbrace{2}_{dX_{min}} + \underbrace{3}_{dY_{min}} + \underbrace{2(N-1)}_{dX_i} + \underbrace{2(N-1)}_{dY_i} \\ &= 4N+1 \end{aligned} \quad (9)$$

4. VARIABLE LENGTH ENCODING

Using the skewed distribution of the delta lengths (Figure 3), we can significantly improve compression compared to $T_f^{\Delta min}$, while still omitting the commas that appear in $T_c^{\Delta min}$. Similar to the concept of Golomb encoding [8], a simple variable-length encoding of the deltas can use a single base B bigit b for most deltas, those below B, and three characters $-bb$ for the rest. For deltas greater than B, we use the indicator character "-" (minus) followed by two characters, $-bb$. Note that reserving "-" for this pupose means there is one less character for the base encoding.



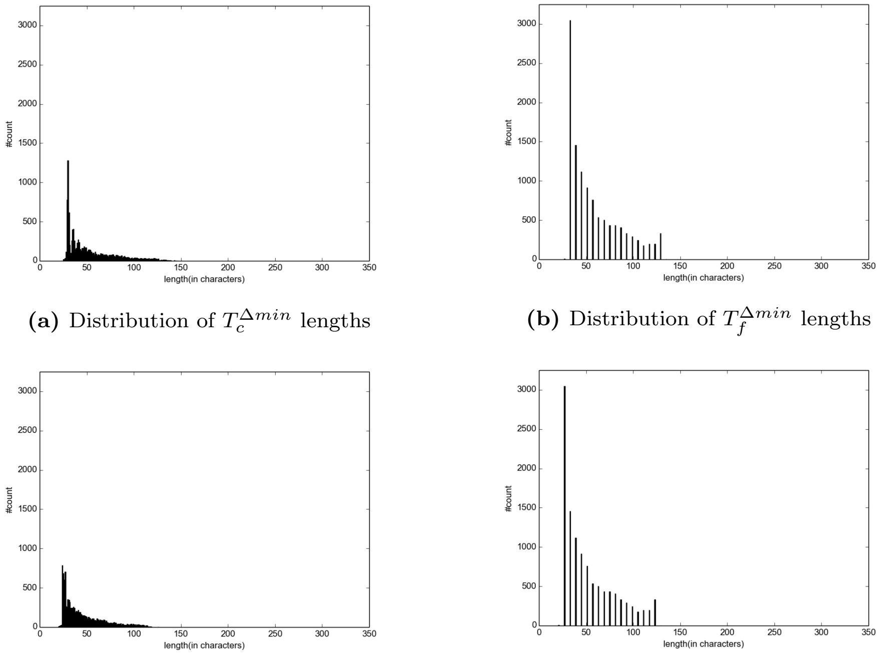
(a) $dX_{min} \in [167, 3284]$

(b) $dY_{min} \in [456, 9932]$

(c) $\delta X_1 \in [169, 3301]$

(d) $\delta Y_1 \in [458, 9988]$

Figure 4: Ranges for large values in both transformations. $T^{\Delta min}$ has (dX_{min}, dY_{min}), and T^{Δ} has $(\delta X_1$, $\delta Y_1)$



(a) Distribution of $T_c^{\Delta min}$ lengths

(b) Distribution of $T_f^{\Delta min}$ lengths

(c) Distribution of T_c^{Δ} lengths

(d) Distribution of T_f^{Δ} lengths

Figure 5: Distribution of base 70 compressed lengths for both transformations

Similarly, dX_{min} and dY_{min} could be encoded in one, three or four characters, using b, $-bb$, or $+bbb$, using $-$ or $+$ as the indicator. However, better compression occurs if we split dX_{min} and dY_{min}, or equivalently δX_1, and δY_1 for T^{Δ}, each into two smaller parts using an agreed factor. Using the base B is a particularly appropriate choice:

$$dX_{min} = B * dX'_{min} + dX''_{min} \text{ or } \delta X_1 = B * \delta X'_1 + \delta X''_1 \quad (10)$$

$$dY_{min} = B * dY'_{min} + dY''_{min} \text{ or } \delta Y_1 = B * \delta Y'_1 + \delta Y''_1 \quad (11)$$

dX'_{min} and dY'_{min} encoding will each require at most three characters due to their distribution shown in figures 4a and 4b. dX''_{min} and dY''_{min} are guaranteed to be encoded using a single bigit each. If B is 70, dX'_{min} will also be encoded using a single bigit. If $T^{\Delta min} = [9, 60, 70, 73]$, then the base-70 encoding would be “9y-10-13”.

It is important to note that we do not need commas or a fixed field to differentiate between the coordinates during decoding, since the indicator character, $-$, will suffice. The variable length encoding will be upper bounded by $6N$ for $T^{\Delta min}$ transformation, and $6N - 6$ for T^{Δ} transformation. While $H_{70}(T_f^{\Delta min})$ length is better than these bounds, the skewed distributions of the NWS corpus ensure that the variable length encoding will be most often better (see Figure 5b).

Leveraging both Skew and Limited Range of Deltas: We can do even better by further exploiting the skewed distributions. To improve on variable length encoding, we notice that since all deltas dX_i, dY_i are less than 350 (and $\Delta X_i, \Delta Y_i$ are less than 550), those that are greater than B, and would normally use $-bb$ do not use the full range allowed by bb; instead we will only see $1b$, $2b$, $\ldots 6b$ for $VAR^{\Delta min}$ (and $1b$, $2b$, $\ldots 8b$) for VAR^{Δ} which allows us to replace the three character $-bb$ with a two character xb, where x is one of several unused special characters such as $[+*/()\% \ldots]$. Likewise, the split dX_{min} uses one char b for each part. dY''_{min}, or $\delta Y''_1$ also uses one character. Only dY'_{min} or $\delta Y'_1$ might use $-bb$, but their range is also restricted (no more than 167), so instead of $-bb$, we will also use xb. The upper bound on the length of $T^{\Delta min}$ after applying variable length encoding (VAR):

$$\begin{aligned} len(VAR_{64}^{\Delta min}) <= & \underbrace{2}_{dX'_{min} \,\&\, dX''_{min}} + \underbrace{3}_{dY'_{min} \,\&\, dY''_{min}} \\ & + \underbrace{2(N-1)}_{dX_i} + \underbrace{2(N-1)}_{dY_i} \qquad (12) \\ <= & \ 4N + 1 \end{aligned}$$

Note again that the base 64 reserves an additional six special characters for the allowed 70. Transformation T^{Δ} needs two more special characters due to the bounds on $(\Delta X_i, \Delta Y_i)$, $len(VAR_{62}^{\Delta})$ is upper bounded by $4N - 3$. The lower bound on the length of $VAR_{64}^{\Delta min}$ is achieved when all values are less than $B = 64$ (and less than $B = 62$ for VAR_{62}^{Δ})[4]:

$$\begin{aligned} len(VAR_{64}^{\Delta min}) >= & \underbrace{2}_{dX'_{min} \,\&\, dX''_{min}} + \underbrace{2}_{dY'_{min} \,\&\, dY''_{min}} \\ & + \underbrace{(N-1)}_{dX_i} + \underbrace{(N-1)}_{dY_i} \qquad (13) \\ >= & \ 2N + 2 \end{aligned}$$

For example, $VAR_{64}^{\Delta min}$ =“9y+11”, if: $T^{\Delta min} = [9, 60, 62, 65]$. The lower bound for VAR_{62}^{Δ} remains same as for $VAR_{64}^{\Delta min}$.

5. BIGNUM COMPRESSION

Bignum compression further improves on $T^{\Delta min}$ or T^{Δ}, by combine each delta pair (dX_i, dY_i) or $(\Delta X_i, \Delta Y_i)$ into a larger single number:

$$dXY_i = dX_i * XX + dY_i, \quad (14)$$

where XX can be a fixed choice or chosen based on the range of dY_i to make sure there is “space” for dY_i. For instance based on our corpus $XX = 350$ will be an appropriate value to “make space” for dY_i because dY_i is less than 350 for the NWS corpus; similarly ΔY_i is always less than 550.

Likewise, we can combine (dX_{min}, dY_{min}) or $(\delta X_1, \delta Y_1)$ using a larger factor:

$$dXY_{min} = dX_{min} * Y_{factor} + dY_{min} \ , \quad (15)$$

where Y_{factor} can be chosen based on the range of dY_{min}. An appropriate value based on the corpus is 10,000 to make space for the dY_{min}.

Expanding on this pair of deltas idea, we can aggregate all deltas into a single large integer, using a simplified form of arbitrary precision integer arithmetic. The large integer is computed by successive pairing of elements of $T^{\Delta min}$:

$$BIG_{i+1}^{\Delta min} = (BIG_i^{\Delta min} * XX^2) + ((dX_i + 1) * XX) + (dY_i + 1), \quad (16)$$

where $i \in [1, N-1]$, and $BIG_i^{\Delta min}$ is 0. We add one to all dX_i and dY_i values to avoid the pathological case when delta values are zero.[5] For T^{Δ} we use essentially the same equation:

$$BIG_{i+1}^{\Delta} = (BIG_i^{\Delta} * XX^2) + ((\Delta X_i + 1) * XX) + (\Delta Y_i + 1), \quad (17)$$

where $i \in [1, N-2]$. The value XX is chosen for each polygon using a "indicator" character S defined by:

$$S = \frac{max(\sup_i dX_i, \sup_i dY_i)}{d} + 1 \quad (18)$$

$$XX = d * S + 1 \quad (19)$$

[4] One could choose to allocate less than 6 or 8 characters to the xb, but then the "-" would be needed in a few cases

[5] Strictly speaking, only the first value, dX_1 could cause a problem.

The value for d can be chosen based on the distribution of the deltas. For the NWS corpus, $d = 6$ ensures XX to be large enough to encode any (dX_i, dY_i), while $d = 9$ ensures XX will be large enough for $(\Delta X_i, \Delta Y_i)$. Also, the above selection of XX guarantees its encoding via S using a single character in base-70 for S. [6]

Larger values for the starting points in $T^{\Delta min}$ and T^{Δ} could also be included in BIG in several ways. Firstly by choosing an appropriate set of factors based on the distribution, such as $X_{factor} = 3500$ and $Y_{factor} = 10000$ for NWS corpus, and then apply the following:

$$BIG_N^{\Delta min} = (BIG_{N-1}^{\Delta min} * X_{factor} + dX_{min}) * Y_{factor} + dY_{min} \tag{20}$$

$$BIG_{N-1}^{\Delta} = (BIG_{N-2}^{\Delta} * X_{factor} + \delta X_1) * Y_{factor} + \delta Y_1 \tag{21}$$

The X_{factor} and Y_{factor} make space for their $(dX_{min}$, $dY_{min})$ or $(\delta X_1, \delta Y_1)$. The encoded string in base-70 representation is a concatenation:

$$\overline{BIG}_{70}^{\Delta min} = S \bullet BIG_N^{\Delta min}$$

$$\overline{BIG}_{70}^{\Delta} = S \bullet BIG_{N-1}^{\Delta}$$

An estimate of the bounds on BIG can be obtained by noticing that we are essentially placing each dX_i and dY_i, or ΔX_i and ΔY_i into an XX sized space, essentially "shifting" by $log_2(XX)$ bits, concatenating into a big number, and then chopping into B sized characters (each $log_2(B)$ bits). Thus allowing one character for S and about four characters for the dX_{min}, δX_1, etc. shifted by X_{factor} and Y_{factor}, we get approximately:

$$\begin{aligned} len(\overline{BIG}_B^{\Delta min}) &\approx 1 + \frac{(log_2(X_{factor}) + log_2(Y_{factor})}{log_2(B)} \\ &\quad + \frac{2(N-1)log_2(XX)}{log_2(B)} \end{aligned} \tag{22}$$

$$len(\overline{BIG}_{70}^{\Delta min}) \approx 2(N-1)\frac{log_2(XX)}{log_2(70)} + 5.09$$

and

$$\begin{aligned} len(\overline{BIG}_B^{\Delta}) &\approx 1 + \frac{(log_2(X_{factor}) + log_2(Y_{factor})}{log_2(B)} \\ &\quad + \frac{2(N-2)log_2(XX)}{log_2(B)} \end{aligned} \tag{23}$$

$$len(\overline{BIG}_{70}^{\Delta}) \approx 2(N-2)\frac{log_2(XX)}{log_2(70)} + 5.09$$

for $B = 70$, $X_{factor} = 3500$ and $Y_{factor} = 10000$.

Thus when $XX = B$, this is essentially one more than the bound on VAR, and gets increasingly better as XX decreases below B. Furthermore, at larger XX, BIG will be better than VAR for certain polygons when a significant number of the delta coordinates in the polygon are larger than B, thus requiring the longer two-character xb representation. Note that actual encoding base B is smaller for VAR than for BIG for the same set of available characters.

The best case compression in the BIG technique for a polygon would be to pick the compression parameters adaptively as follows:

$$XX = max(\sup_i dX_i, \sup_i dY_i) + 2^{[7]}$$

$$X_{factor} = dX_{min} + 1$$

$$Y_{factor} = dY_{min} + 1$$

However, outputting these exact choices as part of the string would add too many characters. Indeed, we could then directly encode dX_{min} and dY_{min} in other ways but experiments suggest that will not be any better than encoding using the X_{factor} and Y_{factor} approach. [8]

6. VARIABLE LENGTH ENCODING WITH REPEATED SUBSTRING DICTIONARY (RSD)

Here we extend the idea of variable length encoding further by usage of a dictionary[9]. The input string to this technique is the transformed list of coordinates represented by $VAR_B^{\Delta min}$ or VAR_B^{Δ}. As indicated above, each delta will either be a single b or two character xb.

Inspired by LZW, we exploit the statistical redundancy in polygons across the corpus to generate a static dictionary for the entire NWS corpus, and provide that to the encoding and decoding systems. This dictionary is essentially a set of most frequently repeated three character sub-strings. We are using the same base B as for variable length encoding (VAR) in the dictionary for keys and values[10]. A sub-string of size two would not have performed any extra compression since we need to prefix a dictionary value with an indicator character to make the distinction of dictionary value in the encoding, and non-dictionary value. The size of the dictionary is constrained by the available set of characters, and any

[6] A slightly better result is obtained if we use a piecewise linear approximation of $XX(S)$ to $max(\sup_i dX_i, \sup_i dY_i)$, whereby we exactly match for $XX = 0 : 30$, and then more granular from $XX = 31 : 330$ (or $31 : 540$ for T^{Δ}).

[7] Since we add one to all deltas in Eq.16, XX has to be strictly greater than all $dX_i + 1$ and $dY_i + 1$ such that we can decode correctly. Strictly, we only have to deal with the first point specially.

[8] We can also apply the same VAR splitting approach, treating the parts of dX_{min} and dY_{min} as four additional deltas. Thereafter, using a potentially larger XX' in place of XX in Eq.16, and also in Eq.20 in place of X_{factor} and Y_{factor}. This adaptively picking parameters holds true for (X'_1, Y'_1). However, in the NWS corpus it is not as good as the X_{factor}, Y_{factor} approach.

[9] A dictionary contains a set of mapping objects (key, value)

[10] Actually, we can use the full allocated 70 character size for the table

Transformation	Variable	Value
$T^{\Delta min}$	O	[31.3,-97.4,31.51,-97.55,31.8,-96.99,31.58,-96.84,31.3,-97.4]
	O'	[3130,9740,3151,9755,3180,9699,3158,9684]
	$T^{\Delta min}$	[1530,3684,0,56,21,71,50,15,28,0]
	$T_c^{\Delta min}$	“1530,3684,0,56,21,71,50,15,28,0”
	$T_f^{\Delta min}$	“153003684000056021071050015028000”
	$\overline{BIG}^{\Delta min}$	“14818307150871153 03684”
	$\overline{BIG}_{70}^{\Delta min}$	“Z7YfAH*‘vmYi4”
T^{Δ}	O	[31.3,-97.4,31.51,-97.55,31.8,-96.99,31.58,-96.84,31.3,-97.4]
	O'	[3130,9740,3151,9755,3180,9699,3158,9684]
	T^{Δ}	[1530,3740,42,30,58,111,43,29]
	T_c^{Δ}	“1530,3740,42,30,58,111,43,29”
	T_f^{Δ}	“15300374004203005811104302 9”
	$\overline{BIG}^{\Delta}$	“33202964332840303740”
	$\overline{BIG}_{70}^{\Delta}$	“ZBqu20DM8m*y”

Table 1: Example of compression with different transformations. Notice that T^{Δ} is always less in length in comparison to $T^{\Delta min}$, primarily because it has one less point. Base 70 is considered for convenience and easy comparison with VAR.

encoding using the dictionary will be of length two. We tried two approaches to construct the dictionary:

- Fixed field matching: Chop the character string into disjoint three character substrings, and keep a count of each unique sub-string.
- Sliding window: Slide a three character window across the string and keep a count of each substring in the dictionary. Again, the size is restricted by the base.

The fixed field case is easier to implement, and faster to execute, but the sliding window gives better results.

There may be cases when none of the repeated substrings occur in the input, and no extra compression is achieved, but there is no penalty, other than a linear order of cost of creating and storing the dictionary, and the finding any matching substrings. For any variable length encoding using base B, VAR_B, the encoding with RSD will be referred as VAR_{B-1_RSD}. $B-1$ due to the extra special character for encoding substrings found in the dictionary.

Table 2 is an example of a static dictionary storing 70 three character substrings for $VAR_{63}^{\Delta min}$ encodings of NWS corpus. 000 occurred 1414 times, whereas $00N$ was the last entry in the table which occurred 96 times:

key	base-70 value
000	0
100	1
...	...
C00	N
...	...
00N	_

Table 2: Repeated Substring Dictionary for variable length encoding

7. ARITHMETIC ENCODING

Arithmetic encoding (AE) is a variable length and lossless encoding technique. For compression and decompression AE relies on a probabilistic model. The algorithm is recursive for each character i.e. it operates upon and encodes (decodes) one data symbol per iteration [11].

The probability model over the possible characters to perform the encoding and decoding steps is essential for optimal compression. Specifically, the compression ratio depends on how well the probability model represents the string of characters to be encoded. For our experiments with polygons the probability of occurrence of any character is based on the entire corpus of polygons.

For the purpose of the polygons, we will define the character sequence $\Sigma = (0123456789)$. Before applying AE, all polygons were transformed to deltas by the same heuristics used to get the $T^{\Delta min}$ string.

Arithmetic encoding is applied to this delta string. The basic algorithm is described below.

1: Begin with the current interval $[l_0, h_0)$ initialized to $[0, 1)$.

2: Sub divide the current interval $[l_0, h_0)$ proportional to

Transformation	Variable	Value
$T^{\Delta min}$	O	[30.97,-92.28 30.89,-92.04 30.61,-92.22 30.65,-92.34 30.97,-92.28]
	O'	[3197,9228,3089,9204,3061,9222,3065,9234]
	$T^{\Delta min}$	[1461,3204,36,24,28,0,0,18,4,30]
	$VAR_{64}^{\Delta min}$	"Mro4aOS00I4U"
	$VAR_{63\_RSD}^{\Delta min}$	"NCosaOS@v4U"
T^{Δ}	O	[30.97,-92.28 30.89,-92.04 30.61,-92.22 30.65,-92.34 30.97,-92.28]
	O'	[3197,9228,3089,9204,3061,9222,3065,9234]
	T^{Δ}	[1497,3228,15,47,55,36,8,24]
	VAR_{62}^{Δ}	"O9q4Flta8O"
	$VAR_{61\_RSD}^{\Delta}$	"OXquFlta8O"

Table 3: Example of compression with different transformations. Notice that @ in $VAR_{63\_RSD}^{\Delta min}$ is the indicator character to distinguish between a dictionary value and non-dictionary value. Although, VAR_{61}^{Δ} had no keys in its RSD dictionary, and therefore $VAR_{61\_RSD}^{\Delta} \approx VAR_{62}^{\Delta}$

the probability of each character in Σ.

3: For each character c_i of the polygon string, we perform two steps: Consider the probability interval for c_i, call it $[l_i, u_i)$ and make it the current interval. Subdivide the current interval into subintervals, one for each possible character, and defined by probabilities over Σ.

4: We output enough bits representing the final interval $[l_n, u_n)$, where n is the length of the polygon string.

The output from step 3 of the algorithm is a binary representation of any real value in the interval $[l_n, u_n)$. A real value in the final interval uniquely identifies a string of characters provided the length of the string to be decoded is known by the decoder. In other words, each input string generates a unique probability interval due to the recursive approach of dividing the probability intervals.

As stated before, we need to embed the length of the input string for the decoder to retrieve the original polygon but here we embed the number of coordinates of the polygon in the compressed string which is sufficient for decompression.

8. STANDARD METHODS-LZW, GOLOMB, HUFFMAN, 7ZIP, GZIP

LZ78 [15] is a variant of the LZW (Lempel-Ziv-Welch) algorithm, implemented for example in the well-known GZIP. The basic idea of the LZW algorithm is to take advantage of repetition of substrings in the data [2] and use a smaller length encoding for such repetitions using data structure like a dictionary with one-to-one mapping of substrings to encodings. We tried the LZW algorithm with both $T_c^{\Delta min}$ and T_c^{Δ} strings.

With T^{Δ} as an input, we also tried other standard string compression algorithms available [12] like *7zip*, and *gzip*, but their compressed lengths were not as good as our BIG or VAR encoding techniques.

We compared our results to Golomb coding [8], which is well studied technique for input values following a geometric distribution, essentially where most values are small, like our delta distributions. Golomb is essentially a concatenation of a variable length unary coded prefix (a string of 1 followed by a 0) and a fixed bit length remainder. We then encoded this bit string in base B characters for both transformations.

We also compared our results to Huffman encoding [9] using probabilities similar to the AE method. Huffman builds a coding tree using these probabilities, but due to its size (4835 leaf nodes for Δmin and 4715 leaf nodes for Δ transformation) this compression technique will not be space efficient on a mobile phone.

9. PRACTICAL CONSIDERATIONS

In order to embed a compressed polygon string in a WEA message, or a simulated WEA message for the trials, we need to signal the start and stop of the polygon string, or the start and length, as appropriate. In most cases, we could prefix, # or #p and a postfix, # or #]. In order to embed the polygon in longer text messages that might contain a # character, we encode any other # as a ##. Furthermore, as indicated, we use base 70 (or higher) to compress numeric strings. We can use a larger base, such as 90, however we need to limit to only use characters that can be included in an SMS or broadcast message, and exclude any characters that are also used by the compression scheme (such as the # sentinel, the -, + and other characters used for signs for variable length VAR substrings, and the @ indicator in the RSD approach). Thus in most cases, the embedded polygon will be 2-4 characters longer than the numbers indicated above.

Also for Arithmetic Encoding (AE) although we need

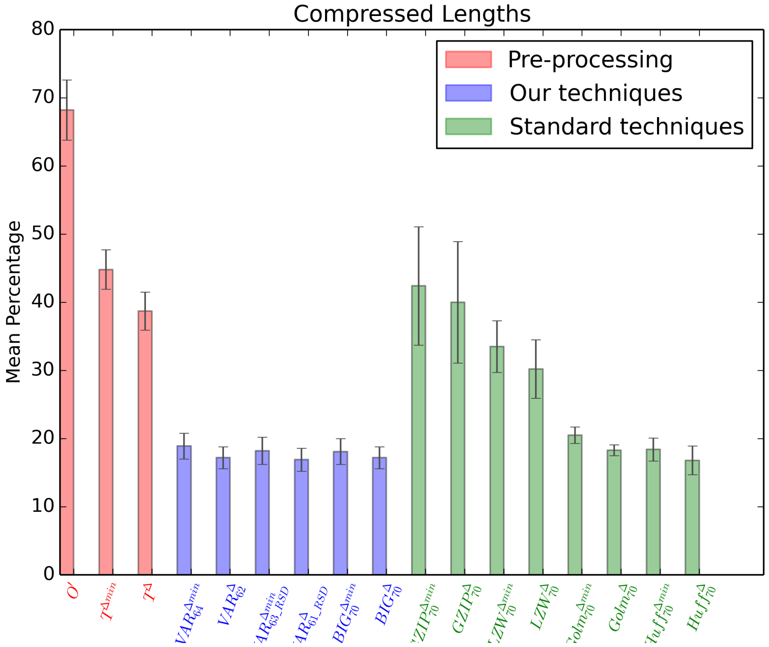


Figure 6: Summary of Base 70 results showing mean percentage of compressed length. Our compression techniques (shown in blue) have mean compressed length less than 20 with low error (standard deviation) in comparison to some standard techniques (shown in green).

to include the number of characters to be retrieved by the decoder, we will need a prefix and a postfix as sentinels. For some values of S in BIG, we can save one character by using a different sentinel, #q, #r, #s instead of #pS.

See [5] for discussion of character sets. Sometimes, operator gateways used in our experiments would not transmit some characters, and so we used base 70, even though a higher base would somewhat improve the compression percentages.

10. POLYALGORITHM

Because of the variability in the results for each technique, each has some polygons for which it is the best method, leading to consideration of combinations of techniques and adaptive technique selection.

As indicated below, the two best techniques $BIG^{\Delta}(S)$ and VAR^{Δ}_{RSD} are close in output character lengths. We can create a combined technique, $POLY^{\Delta}(S)$, that uses $BIG^{\Delta}(S)$ where it is best and VAR^{Δ}_{RSD} where it is best, adding an extra character $S = 0$ to signal VAR^{Δ}_{RSD} and other values of S to signal and control $BIG^{\Delta}(S)$. Since BIG^{Δ} and VAR^{Δ}_{RSD} are so close (with BIG better for small XX), adding this extra character can swamp the benefit. This extra character can be saved in the practical case by using a different sentinel #q, instead of #p0. Note that if we decide that only 70 characters are available, we use the full $B = 70$ as encoding base for BIG, but since we are reserving 9 characters for indicators in VAR^{Δ}_{RSD}, we actually use only $B = 61$ as encoding base for the corresponding $VAR^{\Delta}_{61,RSD}$. Thus VAR, and VAR_{RSD} do "waste" some of the available characters.

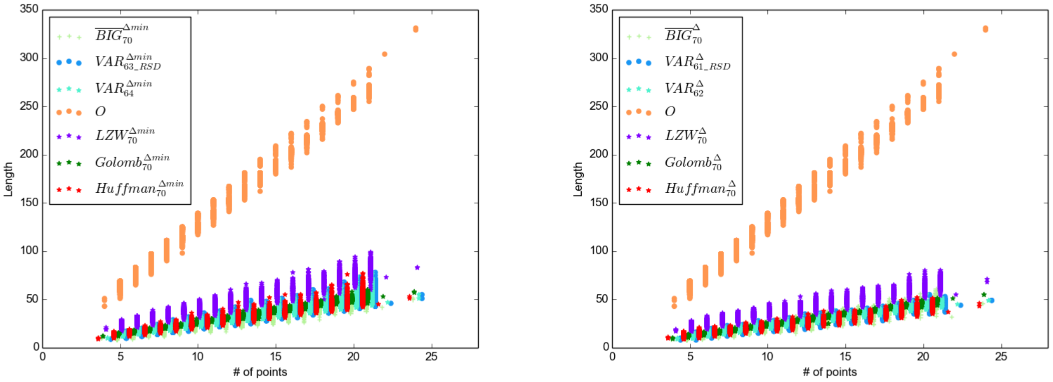
It is important to note that BIG should always be better than VAR when $dXY_{max} = \sup_i(dX_i, dY_i) < B$[11], which, as can be seen from Figure 3, occurs more than 65% of the time for $B = 62$ and more than 80% for $B = 90$ for $T^{\Delta min}$, and when $\Delta XY_{max} = \sup_i(\Delta X_i, \Delta dY_i) < B$ which occurs 34% for $B = 62$ and 62% for $B = 90$ for $T^{\Delta min}$. Typically the shorter length of BIG occurs when XX is quite a bit less than B.

So in setting $XX(S)$ effectively for the polyalgorithm, its important to have XX close to the dXY_{max} or ΔXY_{max}. In many cases BIG will be better than VAR, and will be substantially better for smaller XX and for those polygons when many of their deltas require two base B characters for larger XX.

11. COMPARISON, DISCUSSION & CONCLUSIONS

Figure 6 summarizes the 70 character (usually base 70,

[11] This is the B used in the comparible VAR encoding



(a) Compression lengths with $T^{\Delta min}$ as input transformation

(b) Compression lengths with T^{Δ} as input transformation

Figure 7: Comparison of compressed polygon lengths with original lengths (O) of polygon strings

except for the VAR techniques) compression results of the 11370 polygons collected from the NWS online portal for 2012, 2013, and through December, 2014. Base 70 was chosen to incorporate the reserved characters used by VAR and VAR_{RSD} apart from the alphanumeric characters of base 62. We noticed small improvements at a higher value 90, particularly in the maximum values of lengths, and the reduced standard deviation.

In general, VAR can be characterized as *the best* and simplest direct technique, with the best compression ratio and least variability. BIG is very close, and as indicated above is better when $XX < B$ and when a significant number of deltas require two characters in VAR for larger XX. However VAR_{RSD} is slightly better overall. Because the results are so close, which method is ultimately deemed best depends strongly on the specific polygon and the overall distribution of the polygons.

Figures 7a, 7b shows that all of the methods yield substantial compressions. As seen from the figures and from the formulas displayed earlier, the results are essentially linear in N.

BIG^{Δ} and VAR^{Δ} are the best direct techniques, each leading in about 50% of the cases. We observed from the detailed results for each polygon that $VAR^{\Delta}_{61\_RSD}$ is better than VAR^{Δ}_{62}, and for more than 50% of the time $VAR^{\Delta}_{61\_RSD}$ is better than BIG_{70}. Thus we introduce the polyalgorithm, $POLY_{70}$ which is slightly better than $VAR^{\Delta}_{61\_RSD}$ overall.

Figures 8a and 8b compare the best methods using a set of 70 available characters, with corresponding encoding base $B = 61, 62, 63, 64$, or 70 for the different techniques, but we would get similar results with a larger base.

Compression of polygons using kd-trees [4] is known to have good compression ratios only when the polygon mesh is sparse with few edges [1].

As we explored various techniques, we experimented with several small optimizations that would occasionally save a character. These involved adjusting some of the parameters such as X_0, Y_0, X_{factor}, and Y_{factor}, changing the piece-wise linear representation of $XX(S)$ and so forth, but overall the parameters presented in this paper seemed the best compromise.

By Shannon's coding theorem [14] optimal compression for a set Σ of symbols in which each symbol c has the probability of occurrence p_c, then the entropy is given by:

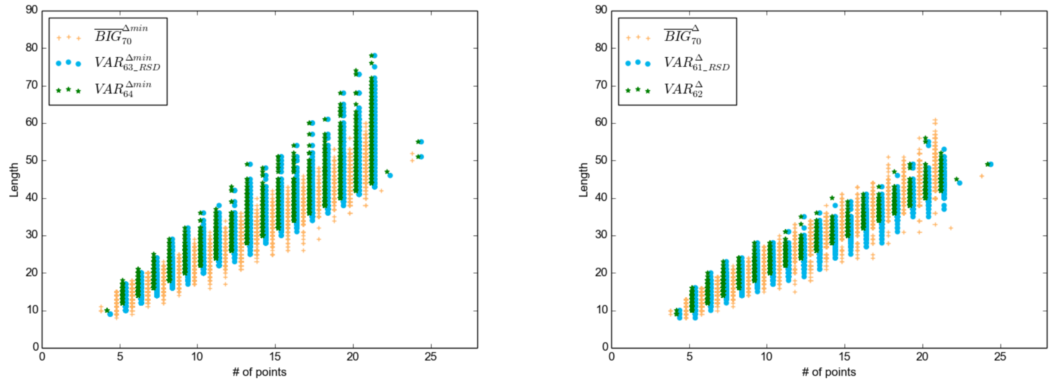
$$E(\Sigma) = \sum_{c \in \Sigma} -p_c \cdot \lg_2(p_c) \tag{24}$$

bits from encoding each symbol. Huffman encoding is an optimal prefix encoding technique, such that the Huffman encoding length is never longer than the entropy of the given distribution. In figure 9, we see that BIG and VAR are close to the almost optimal Huffman encoding. None of the other techniques like *7zip*, LZW, and *gzip* were as good on the NWS corpus as these two.

12. FUTURE WORK

Future work includes exploring even more use of statistical skew, such as an extended form of variable length encoding. This approach will allow us to take the variable-length encoding strategy further by starting from a base-2 (binary) representation, and using one or two bits to determine the length of the following field. We will further explore this more Golomb-like option.

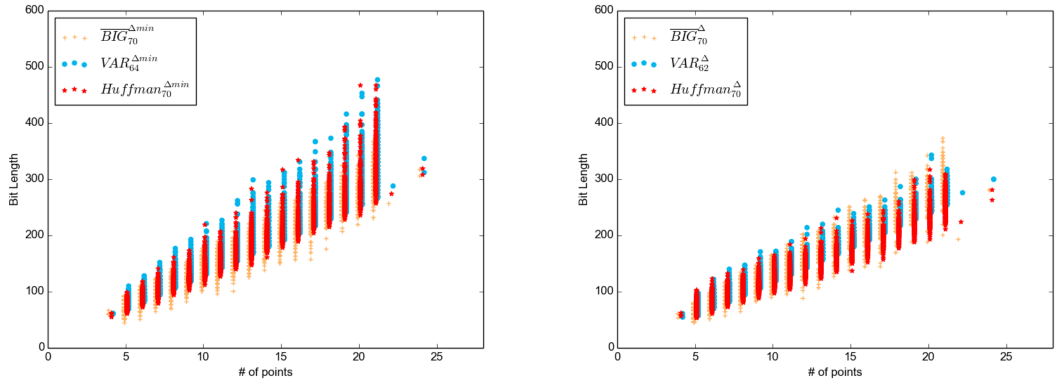
We also plan to use integer programing to find the near optimal set of compression parameters such that we maximize the leverage of any skew in the delta values.



(a) Compression lengths with $T^{\Delta min}$ as input transformation

(b) Compression lengths with T^{Δ} as input transformation

Figure 8: Comparison of best techniques



(a) Compression lengths in bits with $T^{\Delta min}$ as input transformation

(b) Compression lengths in bits with T^{Δ} as input transformation

Figure 9: Comparison of compressed polygon lengths with Shannon bound

13. ACKNOWLEDGEMENTS

We gratefully appreciate support from the Department of Homeland Security, Science and Technology Directorate. We also appreciate the conversations, advice and weather polygon data supplied by Mike Gerber of the National Oceanic and Atmospheric Administration's National Weather Service.

14. REFERENCES

[1] P. Alliez and C. Gotsman. Recent advances in compression of 3d meshes. In *Advances in Multiresolution for Geometric Modelling*, pages 3–26. Springer, 2005.

[2] S. Blackstock. LZW and GIF explained. `http://www.cs.cmu.edu/~cil/lzw.and.gif.txt`, 1989. [Online; accessed 26-January-2015].

[3] L. T. DeCarlo. On the meaning and use of kurtosis. *Psychological methods*, 2(3):292, 1997.

[4] O. Devillers and P.-M. Gandoin. Geometric compression for interactive transmission. In *Visualization 2000. Proceedings*, pages 319–326. IEEE, 2000.

[5] ETSI. Digital cellular telecommunications system (Phase 2+); Alphabets and language-specific information. `http://www.etsi.org/deliver/etsi_ts/100900_100999/100900/07.02.00_60/ts_100900v070200p.pdf`, 1998. [Online; accessed 19-October-2014].

[6] P.-M. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 372–379. ACM, 2002.

[7] M. Gerber. NOAA Public WEA Dataset, 2013.

[8] S. Golomb. Run-length encodings (corresp.). *Information Theory, IEEE Transactions on*, 12(3):399–401, Jul 1966.

[9] D. A. Huffman et al. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[10] B. Iannucci, P. Tague, O. J. Mengshoel, and J. Lohn. Crossmobile: A cross-layer architecture for next-generation wireless systems. 2014.

[11] G. G. Langdon Jr. An introduction to arithmetic coding. *IBM Journal of Research and Development*, 28(2):135–149, 1984.

[12] M. Mahoney. Data Compression Programs. `http://www.mattmahoney.net/dc/`, 2015. [Online; accessed 26-January-2015].

[13] NOAA. NOAA Public WEA Dataset. `http://weather.noaa.gov/pub/logs/heapstats/`, 2014. [Online; accessed 19-October-2014].

[14] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[15] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.