Small Polygon Compression

Abhinav Jauhri, Martin Griss, Hakan Erdogmus

Carnegie Mellon University, ECE Department

1. Introduction & Motivation

We examine the question of representing streams of planar coordinates (polygon) using printable alphabets. The main application is to embed compressed polygons in emergency alert messages that have strict length restrictions, as in the case of Wireless Emergency Alert messages. We transform polygon coordinates to sets of integers and are able to compress them to between 10.4% and 25.6% of original length reducing original polygon lengths from 43-331 characters to 9-61 characters. We also show that our technique is similar to and at times better than prior published state of the art integer compression techniques in terms of bits per integer.

About WEA

Wireless Emergency Alert (WEA) is a nation-wide service for broadcasting short messages to all phones in a designated area via activation of appropriate cell towers. The area is typically identified by a polygon as shown below.

Figure 1: A map showing 3 polygons (outlined by a yellow border). Actual broadcast of any alert covers a bigger area than shown by a polygon above.

2. Proposed Compression Technique *(Bignum)*

Use two standard techniques to normalize the structure of numbers describing the polygon. Original polygon is given by ordered sequence *O*:

$$O = \{X_1, Y_1, \ldots, X_N, Y_N\}$$

Steps common to $T^{\Delta min}$ and 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, so both X_i and Y_i are positive integers, to produce O':

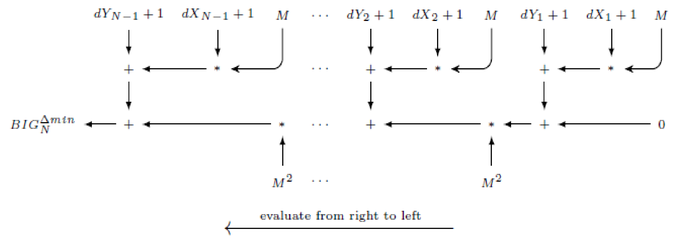
$$X_i = int(100 * X_i); Y_i = -int(100 * Y_i)$$

Step 2: Drop the last point N which is a duplicate of the first point since these are closed polygons

Steps specific to $T^{\Delta min}$	Steps specific to T^{Δ}
Step 3: Compute $X_{min} = \inf_i X_i$, $Y_{min} = \inf_i Y_i$.	**Step 3:** Compute deltas for coordinates where $i \in [1, N-2]$: $\Delta X_{i+1} = X_{i+1} - X_i$ $\Delta Y_{i+1} = Y_{i+1} - Y_i$
Step 4: Compute deltas for coordinates such that $i \in [1, N-1]$: $dX_i = X_i - X_{min}$ $dY_i = Y_i - Y_{min}$, where dX_i and dY_i are non-negative integers.	**Step 4:** Compute deltas for X_1 and Y_1 from a the chosen "origin", $(X_0, Y_0) = (1600, 6000)$: $\delta X_1 = X_1 - X_0$ $\delta Y_1 = Y_1 - Y_0$
Step 5: Compute deltas for X_{min} and Y_{min} from a chosen "origin", origin (X_0, Y_0): $dX_{min} = X_{min} - X_0$ $dY_{min} = Y_{min} - Y_0$.	**Step 5:** 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: $e = \begin{cases} 2e, & \text{if } e >= 0 \\ -2e-1, & \text{if } e < 0 \end{cases}$
Resultant set: $T^{\Delta min} = \{dX_{min}, dY_{min}, dX_1, dY_1, \ldots, dX_{N-1}, dY_{N-1}\}$	**Resultant set:** $T^{\Delta} = \{\delta X_1, \delta Y_1, \Delta X_2, \Delta Y_2, \ldots, \Delta X_{N-1}, \Delta Y_{N-1}\}$

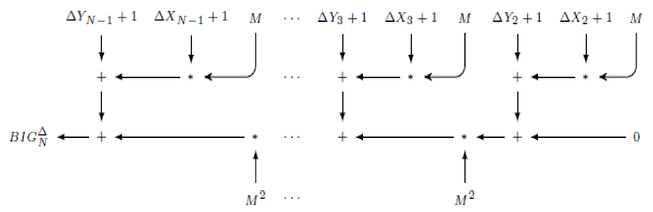
For each of the resultant sets, find the corresponding parameter and perform radix based repacking:

$$M = max(\sup_i dX_i, \sup_i dY_i) + 2$$



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

$$\overline{BIG}_B^{\Delta min} = M \bullet BIG_{N+1}^{\Delta min}$$



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

$$\overline{BIG}_B^{\Delta} = M \bullet BIG_{N+1}^{\Delta}$$

Using two characters to encode *M,* the approximate length in base B characters can be given by:

$$len(\overline{BIG}_B^{\Delta min}) \approx 2 + \frac{(log_2(X_{factor}) + log_2(Y_{factor}) + 2(N-1)log_2(M)}{log_2(B)}$$

$$len(\overline{BIG}_B^{\Delta}) \approx 2 + \frac{(log_2(X_{factor}) + log_2(Y_{factor}) + 2(N-2)log_2(M)}{log_2(B)}$$

3. Results

	Compressed length					Compression ratio (%)				
	min.	mean	max.	σ	95^{th} percentile	min.	mean	max.	σ	95^{th} percentile
$\overline{BIG}_{70}^{\Delta min}$	9	21.9	61	10	44	11.7	19.0	27.3	2.1	22.4
$\overline{BIG}_{70}^{\Delta}$	9	21	61	10	43	10.4	18.0	25.6	1.8	20.8

Table 1: Results for *Bignum* with B = 70

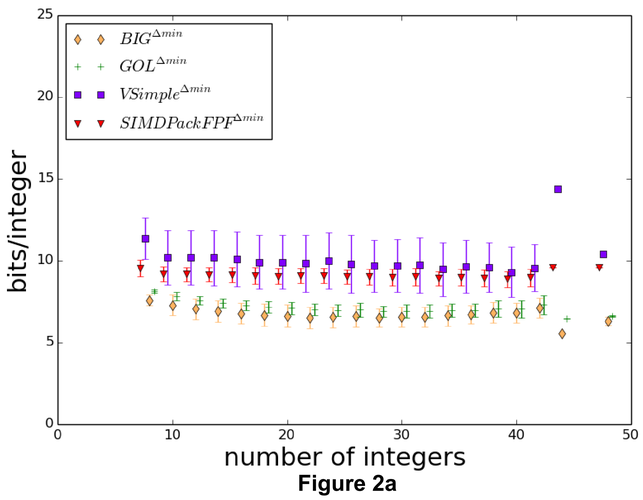


Figure 2a

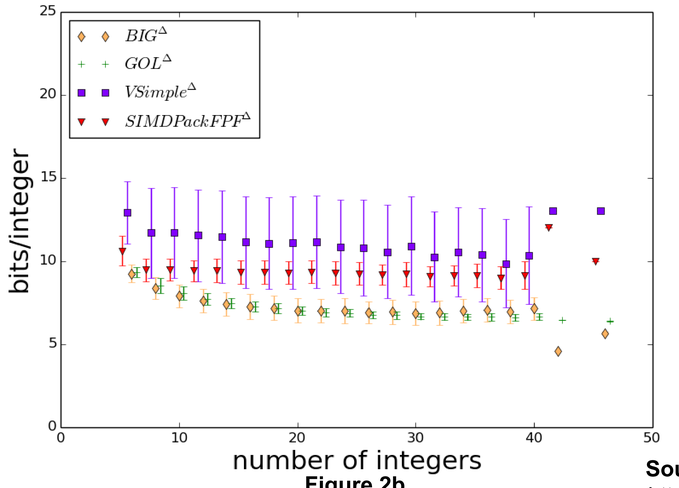


Figure 2b

Figure 2: Experimental comparison of compression techniques for both transformations as input. The number of integers on x-axis is equal to *(#vertices)* * *2* for fig. 2a & *(#vertices – 1)*2* for fig. 2b.

Golomb & Rice uses a fixed parameter *b* to compress a positive integer *v* via the quotient *v/b.* We found $b=2^5$ to give best results.

For vectorized schemes, we used open source library *TurboPFor (https://github.com/powturbo/TurboPFor)*

Source code:
https://github.com/ajauhri/bignum_compression

4. Example

Transformation	Variable	Value
$T^{\Delta min}$	O	{31.35,-85.42 31.27,-85.82 31.43,-85.85 31.6,-85.42 31.35,-85.42}
	O'	{3135,8542,3127,8582,3143,8585,3160,8542,3135,8542}
	$T^{\Delta min}$	{1527,2542,8,0,0,40,16,43,33,0}
	$BIG_{N+1}^{\Delta min}$	11800230453586527254 2
	$\overline{BIG}_{70}^{\Delta min}$	"hfsEYx0N5(xC"
T^{Δ}	O	{31.35,-85.42 31.27,-85.82 31.43,-85.85 31.6,-85.42 31.35,-85.42}
	O'	{3135,8542,3127,8582,3143,8585,3160,8542,3135,8542}
	T^{Δ}	{1535,2542,15,80,32,6,34,85}
	BIG_N^{Δ}	2954312847725352542
	$\overline{BIG}_{70}^{\Delta}$	"1F13Eq4y'g*g2"

Table 2: Example of compression using *Bignum*