Reducing Data Transfer for Charts on Adaptive Web Sites

Giuseppe Burtini
University of British Columbia
g.burtini@alumni.ubc.ca

Scott Fazackerley
University of British Columbia
scott.fazackerley@ieee.org

Ramon Lawrence
University of British Columbia
ramon.lawrence@ubc.ca

ABSTRACT

Web content is consumed on devices with a significant variation in display resolution. Visualizing data is typically performed by extracting data from a database for transmission to the client and then visualizing it with a client-side Javascript library. A major challenge is to retrieve only the required data for visualization. Current approaches require programmers to manually modify their data extraction queries and do not adapt to client display characteristics. The contribution in this work is a configurable data compression method that automatically adapts the amount of data transmitted for client-side visualization based on device characteristics. We evaluate several different techniques for time series summarization and show the amount of data transmitted can be reduced by between 40% and 80% on standard data sets while preserving pixel-perfect visualization.

Categories and Subject Descriptors

E.4 [**Coding and Information Theory**]: Data compaction and compression; H.2.4 [**Systems**]: Query Processing

Keywords

visualization, data compression, time series

1. INTRODUCTION

With the proliferation of web devices, designers are challenged to construct web sites with content that is adaptive to device display characteristics. Client-side libraries [7] allow content to be scaled and modified for best display. However, there has been less work on how to adapt the amount of data provided to the client. Displaying data is a common task for applications involving time series data such as those in economics, environmental monitoring, and sensor networks. In these domains, the amount of data to be visualized is often considerably larger than can be displayed, and transmitting all data to the client is costly. Retrieving too much or too little data for the client device is costly in bandwidth and/or visualization accuracy.

This work demonstrates a system that takes time series data and generates compressed summaries to only transmit the necessary data to the client for visualization. The focus of this work is not on the particular compression technique chosen, but rather demonstrating how these techniques can be used in conjunction with knowledge of device characteristics to reduce the amount of data transmitted for visualization. Using standard data sets [5], we reduce data transmitted by an average of 42%. For smart phones, the reduction is 77%.

2. BACKGROUND

A standard solution to the problem of presenting large data is aggregation. For example, if the data consists of sensor readings sampled every 5 seconds, then a chart displaying the data over a year will typically aggregate the samples on a daily basis. The issue is that this aggregation is usually pre-defined and does not adapt to the display. A larger display can present more data and should not be limited *a priori* on the data that it displays. Conversely, a smart phone display that attempts to display a chart with data better suited to a larger display will end up discarding data that does not fit on its screen or overplotting the data making the visualization difficult to understand. In a web environment, the transmission of the data to the client is the most costly and time-consuming factor.

In this work, we focus primarily on presenting time series data. Time series summarization techniques produce a summary of the data set that is significantly smaller than the original. There are numerous techniques used for time series compression.In this paper, we implement a piecewise polynomial approximation approach similar to [10]. The closest related work defines perceptually important points (PIPs) [4] to only transmit the most visually important points. PIPs are an implementation of the well-known line simplification algorithm first proposed in [2]. Although [4] mentions adapting the number of PIPs to display resolution, it does not define a practical stopping condition (ϵ).

2.1 Definitions

Consider a time series T of n points $t_1, t_2, ..., t_n$. Assume this time series will be visualized on a line chart with height h pixels and width w pixels. If $n \leq w$, the chart can display the time series with at least one pixel per data point. If $n > w$, then there are more data points than can be displayed and summarization is required.

Aggregation reduces data by averaging in intervals, $d_i = \frac{1}{x}\sum_{j=(i-1)*x+1}^{i*x} t_j$ for $i = 1..\lceil \frac{n}{x} \rceil$. By controlling x, the time series can be reduced in size by any factor. For a display of width w pixels and time series of n points, the minimum $x = \lceil n/w \rceil$.

Consider an example data set consisting of $n = 10$ points

$\{5,7,9,7,5,5,1,9,3,5\}$. If $w=5$, then aggregation will display $\{6,8,5,5,4\}$. As shown in Figure 1, aggregation loses detail information. Without any aggregation, displaying the 10 data points in 5 pixels results in an overplotted figure as there is more data points than pixels to display them.

The vertical resolution V is defined as the size of each pixel in the vertical dimension in terms of the data such that

$$V = \frac{\max(t_i) - \min(t_i)}{h} \qquad \text{for } i = 1..n. \tag{1}$$

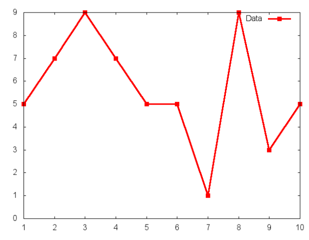
A model m is a representation of a time series with values $m_1, m_2, ..., m_n$. The absolute error is defined as the sum of absolute differences between the data and the model with $AE = \sum_{i=1}^{n} |m_i - t_i|$.

Visual error for model m is defined as the sum of the number of differing pixels between the resulting visualization and that achieved by aggregation with $x = \lceil n/w \rceil$. Visually relevant error is defined as

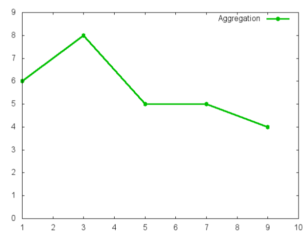
$$VRE = \sum_{i=1}^{n} \lfloor \frac{|m_i - d_{\lfloor (i-1)/x \rfloor + 1}|}{V} \rfloor. \tag{2}$$

With our example, absolute error on aggregation with $x=2$ is 12. Visual error is defined to be 0 as aggregation is the baseline model. We are interested in algorithms that perform lossless summarization in terms of visual error.

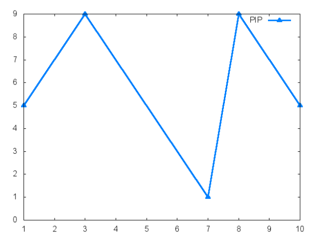
Assume the example 10 data point set is part of a larger set such that for visualization only 5 pixels are available to display the data ($w=5$). Sending the entire data set to the client for visualization may result in sampling, aggregation, or overplotting. Developers perform aggregation on the data to ensure it can be visualized properly. The minimum amount of aggregation is averaging two points as shown in Figure 1(b). Aggregation loses information (e.g. change between 7 and 8) which is measurable by absolute error. In Figure 1(c) is the output for PIP which models the base data series with no aggregation but may still be subject to overplotting (e.g. points 7 to 8 will occupy the same pixel). In Figure 1(d) is an example of overplotting where excess data is displayed.



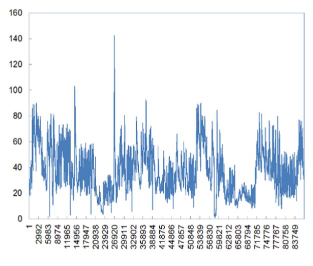
(a) Base Data



(b) Aggregation



(c) PIP



(d) Overplotting

Figure 1: Time Series Summarization

3. OVERVIEW OF THE APPROACH

When a user requests a web page containing a visualization,the browser provides information on the client environment and display resolution. Prior to returning the data, it is passed through a compression module which outputs a compressed data stream specific to the client request. Thus, a request will be handled according to the different visualization properties of each device. When the client receives the compressed data, a Javascript routine transparently decompresses the data before passing it to the visualization library. The server transmits significantly less data to the client to achieve an equivalent visualization compared to the whole data set. This saves bandwidth and time and cost for the user and the service provider.

3.1 Basic Algorithms

No compression and run-length encoding (RLE) provide the full, lossless data to the client. This leaves the presentation details up to the library which typically results in overplotting and unreadable charts. The simplest visually aware algorithm is dynamic, visual aggregation. Given the display width w and time series of length n, the time series is aggregated by averaging $\lceil n/w \rceil$ adjacent points, guaranteeing one point per pixel in the visualization (maximizing the information displayed). A natural extension to lossless RLE is to perform dynamic RLE, considering points equivalent if they are within an error bound ϵ. By setting $\epsilon = V$, considerable compression may be achieved as similar points may be deemed equivalent in terms of visual representation. Visual RLE produces an output that is visually equivalent to visual aggregation.

3.2 Perceptually Important Points (PIPs)

The work on PIPs [4] is one of the only existing works to consider summarizing a time series data by its visual characteristics. As explained in Section 2, PIPs are determined iteratively by calculating the next point that is furthest from the current representation. Although [4] indicates that PIPs can be used to adapt to display characteristics, the authors do not state specifically how this is achieved. The major drawback is that there is no stopping criteria defined for PIPs. Our implementation provides a parameter n denoting how many PIPs to use for data summary. The algorithm can also continually calculate PIPs until a given error bound ϵ is hit.

3.3 Piecewise Polynomial Approximation

Our implementation combines piecewise linear and polynomial approximation. The algorithm models the data using a series of linear and polynomial functions as shown in Algorithm 1. A fitting function performs a linear scan through the data with the goal of fitting a piecewise function of a functional form F. Fitting continues until the stopping condition where the error exceeds ϵ. The algorithm then stores the last good fit as a piecewise section for the model and continues to construct a new piecewise segment starting from the point where the previous segment failed to meet the stopping condition. During the fitting process for a single segment, if it is not possible to fit F with a given degree n, it attempts to fit a degree $n-1$ function. This process continues until the base case where F is the data point itself to ensure that no point is left unfitted.

4. EXPERIMENTAL RESULTS

It is well-known that algorithms have widely different performance on different data sets. The data sets tested are listed in

Algorithm 1 PPA(Series T, Form F, Threshold ϵ)

```
data = [], pieces = [], model = null
for $i = 0$; $i <$ T.length; $i++$ do
    data.append($t_i$)
    previousModel = model
    model = fit(data, F)
    error = calculateError(model, data)
    if error $\geq \epsilon$ then
        pieces[pieceCounter++] = model
        $i$ -= 1, data = []
    end if
end for
return pieces
```

Data Set	Size	Source
CPI Seasonal Data	787	[1]
Minutely Soil Moisture Readings	136 503	[3]
Variable Star Magnitude	600	[9]
Artificial AR(1) Process	500	[8]
Dow Jones Industrial Avg. Closing	641	[8]
Central England Temp. (1659-1989)	3 972	[6]
Keogh: fortune500.dat	128 499	[5]
Keogh: slips.dat (subset)	2 500	[5]
Keogh: katafu.dat (subset)	5 000	[5]
Keogh: stocks-n6480.dat (subset-1)	10 000	[5]
Keogh: stocks-n6480.dat (subset-2)	10 000	[5]
Child EEG Readings	49 999	[5]

Table 1: Experimental Data Sets

Algorithm	Avg. Rel. Data	Avg. AE	Avg. VRE
RLE	1.01	0	0%
V. Agg.	0.448	27 366	0%
PIP ($\epsilon = 0$)	1.808	6 373	0%
PIP ($\epsilon = best$)	1.185	26 767	3.77%
VRLE	0.925	3 785	0%
PPA	**0.379**	**3 169**	**0%**

Table 2: Avg. Results for all Data and Resolutions

Table 1. Common resolutions starting from 100x100 (thumbnail charts) to 1440x960 (HD laptop screen) were tested. The algorithms are evaluated based on their absolute error, visually relevant error, and size of data transferred.

The absolute error is the difference between the original data and the compressed representation. Uncompressed and RLE have no absolute error as they send the entire data set. PIP has a high absolute error until it sends all the data points. Functional compression has considerably less error, both absolute and visually relevant as shown in Table 2. Visually relevant error occurs due to inaccurate data caused by summarization. We have modified the selected algorithms to exit when there is no visually relevant error, so the interesting variable to compare is the relative data size to achieve no visually relevant error with a given algorithm.

Figure 2 shows the fraction transferred relative to the data set. PIP transfers more bytes than the data set itself. Although it is possible to send fewer PIP than the horizontal resolution as recommended by [4], the amount of absolute and visual error is extreme which renders the visualization useless. Functional compression sends, on average, 62% less data than the entire data set and adapts to the screen resolution. As expected there is significant savings in data transmitted for lower resolutions.

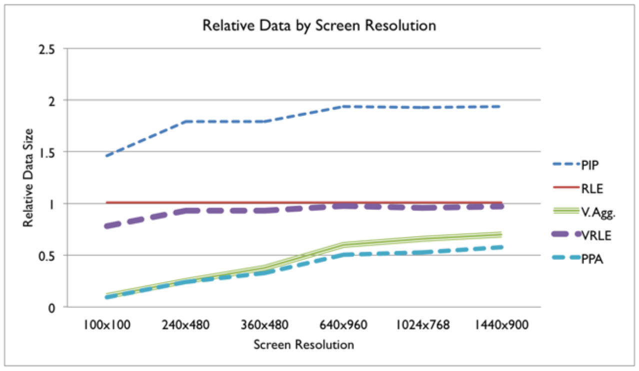


Figure 2: Relative Data Size by Screen Resolution

5. CONCLUSIONS

Visualization of time series data is very common in web applications. Performance of the visualization is primarily based on the amount of data sent from the server to the client. In this work, we demonstrated a technique that uses knowledge of the device characteristics as input for time series compression algorithms, in order to reduce the amount of data transferred while at the same time achieving pixel-perfect visualizations. Across a wide variety of different time series and screen resolutions, the size-weighted average savings is 42%. Future work will examine low/high-pass style filtering to extract structural components, find appropriate functional forms for specific data series, and apply a PPA-style technique to higher dimensional data.

6. REFERENCES

[1] Bureau of Labour Statistics. CPI Seasonally Adjusted, Monthly, 1913-2012. `http://www.bls.gov/cpi/`, 2012.

[2] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[3] S. Fazackerley and R. Lawrence. Reducing turfgrass water consumption using sensor nodes and an adaptive irrigation controller. *IEEE Sensors Applications Symposium*, pages 90–94, 2010.

[4] T.-C. Fu, K. F.-L. Chung, C.-F. Lam, R. W. P. Luk, and C. man Ng. Adaptive data delivery framework for financial time series visualization. In *ICMB*, pages 267–273, 2005.

[5] E. Keogh, Q. Zhu, B. Hu, H. Y., X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage: `www.cs.ucr.edu/~eamonn/time_series_data/`, 2011.

[6] G. Manley. Central england temperatures: monthly means 1659 to 1973. *Quarterly Journal of the Royal Meteorological Society*, pages 389–405, 1974.

[7] V. Ogievetsky and J. Heer. D^3 data-driven documents. *IEEE TVCG*, 17(12):2301–2309, 2011.

[8] The University of York, Math Department. `http://www.york.ac.uk/depts/maths/data/ts/`, 2004.

[9] E. Whitaker and G. Robinson. *The Calculus of Observations*, page 341. Blackie and Son Ltd., 1941.

[10] Z. Xu, R. Zhang, K. Ramamohanarao, and U. Parampalli. An adaptive algorithm for online time series segmentation with error bound guarantee. In *EDBT*, pages 192–203, 2012.