University Transportation Research Center - Region 2

Final Report

Techniques for Information Extraction from Compressed GPS Traces

Performing Organization: State University of New York (SUNY)

December 2015

Sponsor:
University Transportation Research Center - Region 2

University Transportation Research Center - Region 2

The Region 2 University Transportation Research Center (UTRC) is one of ten original University Transportation Centers established in 1987 by the U.S. Congress. These Centers were established with the recognition that transportation plays a key role in the nation's economy and the quality of life of its citizens. University faculty members provide a critical link in resolving our national and regional transportation problems while training the professionals who address our transportation systems and their customers on a daily basis.

The UTRC was established in order to support research, education and the transfer of technology in the field of transportation. The theme of the Center is "Planning and Managing Regional Transportation Systems in a Changing World." Presently, under the direction of Dr. Camille Kamga, the UTRC represents USDOT Region II, including New York, New Jersey, Puerto Rico and the U.S. Virgin Islands. Functioning as a consortium of twelve major Universities throughout the region, UTRC is located at the CUNY Institute for Transportation Systems at The City College of New York, the lead institution of the consortium. The Center, through its consortium, an Agency-Industry Council and its Director and Staff, supports research, education, and technology transfer under its theme. UTRC's three main goals are:

Research

The research program objectives are (1) to develop a theme based transportation research program that is responsive to the needs of regional transportation organizations and stakeholders, and (2) to conduct that program in cooperation with the partners. The program includes both studies that are identified with research partners of projects targeted to the theme, and targeted, short-term projects. The program develops competitive proposals, which are evaluated to insure the mostresponsive UTRC team conducts the work. The research program is responsive to the UTRC theme: "Planning and Managing Regional Transportation Systems in a Changing World." The complex transportation system of transit and infrastructure, and the rapidly changing environment impacts the nation's largest city and metropolitan area. The New York/New Jersey Metropolitan has over 19 million people, 600,000 businesses and 9 million workers. The Region's intermodal and multimodal systems must serve all customers and stakeholders within the region and globally.Under the current grant, the new research projects and the ongoing research projects concentrate the program efforts on the categories of Transportation Systems Performance and Information Infrastructure to provide needed services to the New Jersey Department of Transportation, New York City Department of Transportation, New York Metropolitan Transportation Council , New York State Department of Transportation, and the New York State Energy and Research Development Authorityand others, all while enhancing the center's theme.

Education and Workforce Development

The modern professional must combine the technical skills of engineering and planning with knowledge of economics, environmental science, management, finance, and law as well as negotiation skills, psychology and sociology. And, she/he must be computer literate, wired to the web, and knowledgeable about advances in information technology. UTRC's education and training efforts provide a multidisciplinary program of course work and experiential learning to train students and provide advanced training or retraining of practitioners to plan and manage regional transportation systems. UTRC must meet the need to educate the undergraduate and graduate student with a foundation of transportation fundamentals that allows for solving complex problems in a world much more dynamic than even a decade ago. Simultaneously, the demand for continuing education is growing – either because of professional license requirements or because the workplace demands it – and provides the opportunity to combine State of Practice education with tailored ways of delivering content.

Technology Transfer

UTRC's Technology Transfer Program goes beyond what might be considered "traditional" technology transfer activities. Its main objectives are (1) to increase the awareness and level of information concerning transportation issues facing Region 2; (2) to improve the knowledge base and approach to problem solving of the region's transportation workforce, from those operating the systems to those at the most senior level of managing the system; and by doing so, to improve the overall professional capability of the transportation workforce; (3) to stimulate discussion and debate concerning the integration of new technologies into our culture, our work and our transportation systems; (4) to provide the more traditional but extremely important job of disseminating research and project reports, studies, analysis and use of tools to the education, research and practicing community both nationally and internationally; and (5) to provide unbiased information and testimony to decision-makers concerning regional transportation issues consistent with the UTRC theme.

Project No(s):
UTRC/RF Grant No: 49997-39-25

Project Date: December 2015

Project Title: Techniques for Information Extraction from Compressed GPS Traces

Project's Website:
http://www.utrc2.org/research/projects/techniques-information-extractions

Principal Investigator(s):

Dr. Catherine T. Lawson
Associate Professor
Department of Geography and Planning
University at Albany
Albany, NY 12222
Tel: (518) 442-4775
Email: lawsonc@albany.edu

Co-author(s):
Feng Chen
Assistant Professor
Department of Computer Science
University at Albany
Albany, NY 12222
Email: fchen5@albany.edu

Jeong-Hyon Hwang
Associate Professor
Department of Computer Science
University at Albany
Albany, NY 12222
Email: jhh@cs.albany.edu

Sekharipuram S. Ravi
Distinguished Teaching Professor
Department of Computer Science
University at Albany
Albany, NY 12222
Email: sravi@albany.edu

To request a hard copy of our final reports, please send us an email at utrc@utrc2.org

Mailing Address:

University Transportation Reserch Center
The City College of New York
Marshak Hall, Suite 910
160 Convent Avenue
New York, NY 10031
Tel: 212-650-8051
Fax: 212-650-8374
Web: www.utrc2.org

Board of Directors

The UTRC Board of Directors consists of one or two members from each Consortium school (each school receives two votes regardless of the number of representatives on the board). The Center Director is an ex-officio member of the Board and The Center management team serves as staff to the Board.

City University of New York
Dr. Hongmian Gong - Geography/Hunter College
Dr. Neville A. Parker - Civil Engineering/CCNY

Clarkson University
Dr. Kerop D. Janoyan - Civil Engineering

Columbia University
Dr. Raimondo Betti - Civil Engineering
Dr. Elliott Sclar - Urban and Regional Planning

Cornell University
Dr. Huaizhu (Oliver) Gao - Civil Engineering

Hofstra University
Dr. Jean-Paul Rodrigue - Global Studies and Geography

Manhattan College
Dr. Anirban De - Civil & Environmental Engineering
Dr. Matthew Volovski - Civil & Environmental Engineering

New Jersey Institute of Technology
Dr. Steven I-Jy Chien - Civil Engineering
Dr. Joyoung Lee - Civil & Environmental Engineering

New York University
Dr. Mitchell L. Moss - Urban Policy and Planning
Dr. Rae Zimmerman - Planning and Public Administration

Polytechnic Institute of NYU
Dr. Kaan Ozbay - Civil Engineering
Dr. John C. Falcocchio - Civil Engineering
Dr. Elena Prassas - Civil Engineering

Rensselaer Polytechnic Institute
Dr. José Holguín-Veras - Civil Engineering
Dr. William "Al" Wallace - Systems Engineering

Rochester Institute of Technology
Dr. James Winebrake - Science, Technology and Society/Public Policy
Dr. J. Scott Hawker - Software Engineering

Rowan University
Dr. Yusuf Mehta - Civil Engineering
Dr. Beena Sukumaran - Civil Engineering

State University of New York
Michael M. Fancher - Nanoscience
Dr. Catherine T. Lawson - City & Regional Planning
Dr. Adel W. Sadek - Transportation Systems Engineering
Dr. Shmuel Yahalom - Economics

Stevens Institute of Technology
Dr. Sophia Hassiotis - Civil Engineering
Dr. Thomas H. Wakeman III - Civil Engineering

Syracuse University
Dr. Riyad S. Aboutaha - Civil Engineering
Dr. O. Sam Salem - Construction Engineering and Management

The College of New Jersey
Dr. Thomas M. Brennan Jr - Civil Engineering

University of Puerto Rico - Mayagüez
Dr. Ismael Pagán-Trinidad - Civil Engineering
Dr. Didier M. Valdés-Díaz - Civil Engineering

UTRC Consortium Universities

The following universities/colleges are members of the UTRC consortium.

City University of New York (CUNY)
Clarkson University (Clarkson)
Columbia University (Columbia)
Cornell University (Cornell)
Hofstra University (Hofstra)
Manhattan College (MC)
New Jersey Institute of Technology (NJIT)
New York Institute of Technology (NYIT)
New York University (NYU)
Rensselaer Polytechnic Institute (RPI)
Rochester Institute of Technology (RIT)
Rowan University (Rowan)
State University of New York (SUNY)
Stevens Institute of Technology (Stevens)
Syracuse University (SU)
The College of New Jersey (TCNJ)
University of Puerto Rico - Mayagüez (UPRM)

UTRC Key Staff

Dr. Camille Kamga: *Director, Assistant Professor of Civil Engineering*

Dr. Robert E. Paaswell: *Director Emeritus of UTRC and Distin*guished Professor of Civil Engineering, The City College of New York

Herbert Levinson: *UTRC Icon Mentor, Transportation Consultant and Professor Emeritus of Transportation*

Dr. Ellen Thorson: *Senior Research Fellow, University Transportation Research Center*

Penny Eickemeyer: *Associate Director for Research, UTRC*

Dr. Alison Conway: *Associate Director for Education*

Nadia Aslam: *Assistant Director for Technology Transfer*

Nathalie Martinez: *Research Associate/Budget Analyst*

Tierra Fisher*: Office Assistant*

Bahman Moghimi*: Research Assistant;*
Ph.D. Student, Transportation Program

Wei Hao*: Research Fellow*

Andriy Blagay: *Graphic Intern*

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2.Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Techniques for Information Extraction from Compressed GPS Traces		5. Report Date December 31, 2015 6. Performing Organization Code
7. Author(s) Catherine T. Lawson, University at Albany S. S. Ravi, University at Albany Jeong-Hyon Hwang, University at Albany Feng Chen, University at Albany		8. Performing Organization Report No.
9. Performing Organization Name and Address University at Albany 1400 Washington Avenue Albany, New York 12222		10. Work Unit No. 11. Contract or Grant No. 49997-39-25
12. Sponsoring Agency Name and Address University Transportation Research Center City College of New York 138th Street and Convent New York, NY 10031	Federal Highway Administration U. S. Department of Transportation Washington, D. C.	13. Type of Report and Period Covered Final Report 1/1/2014 – 12/31/2015 14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Developing techniques for extracting information requires a good understanding of methods used to compress the traces. Many techniques for compressing trace data consisting of position (i.e., latitude/longitude) and time values have been developed. Since current vehicles are equipped with many on-board instruments, traces generated by such vehicles contain many attributes in addition to position and time. The problem of compressing such multi--atribute traces is currently being studied by a number of researchers.

We consider the Multiple Attribute Trajectory Compression program with defined error bounds on attributes. Our focus is on solving this problem using attribute partition methods. Such methods partition attributes into groups with the aim of reducing the total storage cost after compression methods are applied to each group individually. We present a comprehensive overview of various trajectory compression algorithms, concentrating on the most recent works since SQUISH-E [20]. New accuracy metrics for measuring the difference between a trajectory and its compressed representation are also explained. Lastly, we present some preliminary experimental results on two real trajectory data sets with multiple attributes using two known compression algorithms, namely PROXIMUS and CompreX.

17. Key Words	18. Distribution Statement

19. Security Classif (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No of Pages 20	22. Price

Form DOT F 1700.7 (8-69)

Disclaimer
The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. The contents do not necessarily reflect the official views or policies of the UTRC or the Federal Highway Administration. This report does not constitute a standard, specification or regulation. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

Techniques for Information Extraction from Compressed GPS Traces

January 6, 2016

Abstract

Developing techniques for extracting information from compressed GPS traces requires a good understanding of methods used to compress the traces. Many techniques for compressing trace data consisting of position (i.e., latitude/longitude) and time values have been developed. Since current vehicles are equipped with many on-board instruments, traces generated by such vehicles contain many attributes in addition to position and time. The problem of compressing such multi-attribute traces is currently being studied by a number of researchers.

We consider the Multiple Attribute Trajectory Compression problem with defined error bounds on attributes. Our focus is on solving this problem using attribute partition methods. Such methods partition attributes into groups with the aim of reducing the total storage cost after compression methods are applied to each group individually. We present a comprehensive overview of various trajectory compression algorithms, concentrating on the most recent works since SQUISH-E [20]. New accuracy metrics for measuring the difference between a trajectory and its compressed representation are also explained. Lastly, we present some preliminary experimental results on two real trajectory data sets with multiple attributes using two known compression algorithms, namely PROXIMUS and CompreX.

1 Organization of the Report

This report explores the compression of trajectories with multiple attributes. Section 2 is a review on trajectory compression using point selection algorithms. Section 3 presents attribute partition methods used in traditional relational databases. Section 4 summarizes metrics used in previous work. Section 5 compares the performance of two known compression algorithms, namely CompreX and PROXIMUS. Section 6 presents some conclusions and directions for future research.

2 Trajectory Compression

2.1 Need for Trajectory Compression

Many trajectory compression techniques have been developed over the years to deal with the drastic growth of trajectory data. The recent interest in connected vehicle initiatives to increase auto safety will produce not only additional trajectory data, but also a large volume of multiple attributes associated with vehicle operations (e.g., data when brakes are applied). The overall purpose of compression is to minimize storage space while preserving the accuracy of the data sets to the maximum possible extent. Typical algorithms achieve compression by retaining a subset of the points from a given trajectory. Algorithms for trajectory compression can be classified along several different axes: offline or online, local or global, lossy or lossless, etc. In the literature, algorithms have been compared on the basis of their space complexity, running time as well as their ability to achieve certain levels of accuracy with respect to known metrics.

2.2 Trajectory Compression Methods

2.2.1 Position Preserving Framework

Position preserving error can be measured in terms of Euclidean distance or synchronized Euclidean distance. While Euclidean distance is considered a minimal requirement, the synchronized version has the advantage of incorporating temporal information and leads to better decisions in terms of choosing relatively significant points in a trajectory. Both metrics have been considered in the literature. In the following, we summarize some of the known methods which use these metrics.

Uniform Sampling Uniform sampling selects points from a trajectory at a constant interval determined by the space and rate of source data. This method can reduce storage requirements, but at the cost of the level of accuracy. This technique is particularly effective when dealing with quick and simple compressions with reasonable error rates. However, it can introduce large errors if data exhibits no particular pattern with respect to time domain. If the data does exhibit a regular pattern with respect to time, the sampled points provide a coarse resolution of the data set in time domain.

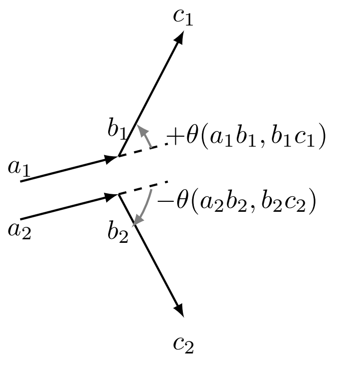
Top-down Split Top-down split methods split a trajectory into segments and recursively apply the same optimization criteria to the segments treated as new trajectories. Generally, top-down methods can only be carried out in an offline fashion. Recall that comparison based sorting algorithms have a running time of $\Theta(n \log n)$. Top down split algorithms achieve this time complexity by carefully choosing an appropriate data structure. The most famous top-down split approaches are the Douglas-Peucker method [3] and the modified Douglas-Peucker method [5] using a convex hull to reduce the time complexity. The Douglas-Peucker method is a greedy method which chooses the most deviated points with respect to the segments of trajectory which have already been considered. The method can also be adapted for online use with a fixed buffer as proposed by Liu [15]. While the original method using Euclidean distance metric can achieve the minimum running time bound by using the convex hull, the method using synchronized Euclidean distance metric is asymptotically slower. Another method which has been studied is the top-down time ratio method [18], which uses top down split to improve the compression rate/error balance. However, the focus of this algorithm is to showcase the importance of temporal information in the compression of trajectory data.

Sliding Window Merge Methods based on sliding windows are suitable for online trajectory compression. This type of algorithm can achieve linear running time with error bounds. The criterion used to decide whether to keep or discard incoming points differentiates the methods. To achieve better time complexity and better error bounds, an appropriate data structure is needed. Time complexity of algorithms without any enhanced data structures can be quadratic. For example, there is such an algorithm for online time series which achieves the worst running time [11]. However, with appropriate data structure and careful usage of data, Liu's method [15] achieves linear time complexity on average, though the more appropriate running time bound is $O(n \log(n))$. It sets up a coordinate bounded system and uses the structure of a convex hull to decide whether or not to calculate the Euclidean distance for a new point. Based on that calculation, it determines whether or not to keep or discard the incoming point. The Squish-E algorithm [20] uses a priority queue to achieve $O(n \log(n))$ time complexity. Dead Reckoning algorithm [24] achieve linear time complexity by using the speed information derived from previously chosen points. However, the performance of sliding window algorithms is drastically different with respect to error metrics [19].

2.2.2 Direction Preserving Framework

Direction-based angular error was first introduced by Long [16]. It claims that in some cases, it is more reasonable to use angular error as the error bound since it better reflects the simplified trajectory compared to the original trajectory and the clusters produced by trajectory clustering algorithm by Lee [13]. Position information can be derived from the direction information [16]. The problem of minimizing the direction-based angular error with a space constraint is proposed in [17]. An offline 2-factor approximation algorithm is introduced. Lee's angular framework is very useful in 3-dimensional space; however, there is more work to be done in order to extend it to four and higher dimensional spaces. Further, the algorithm's time complexity needs to be improved if one needs to use it in an online setting.

Line simplifications can be categorized as either online or batch model based. Batch model based methods focus on discarding locations with minimal error from the original trajectory. The uniform sampling algorithm is effectively a line simplification that uses only the ıth location and discarding the others.The Douglas-Peucker algorithm considers the trajectory as a line segment and recursively selects the point with the largest error as a split point, until the trajectory satisfies the error requirement. Bellman's algorithm uses dynamic programming to minimize the area between the original trajectory and the compressed one. While these methods are efficient, PRESS is much more efficient with respect



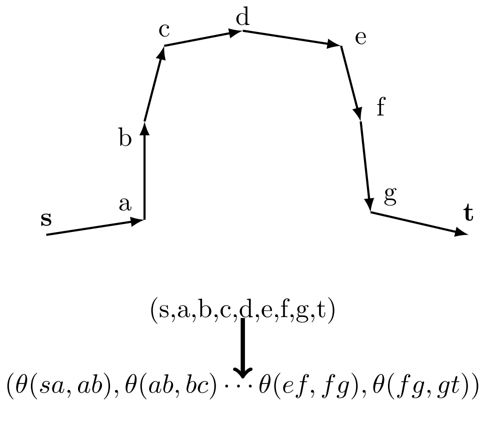

c
d
e
f
b
a
g
s
t
(s,a,b,c,d,e,f,g,t)
$(\theta(sa,ab),\theta(ab,bc)\cdots\theta(ef,fg),\theta(fg,gt))$
(b)

Figure 1: (a) Curvature as signed angle of turn, clockwise turn as negative curvature. (b) Transformation of trajectory to curvature function. Trajectory input is represented as a sequence of locations, then is processed according to the definition of curvature. Sequence beginning with **s**, ending at **t** is transformed into sequence of curvature values.

to time complexity compared to DP-variants and Bellman's algorithms. Other benefits of PRESS include spatial losslessness and temporal error-bound.

2.2.3 Topological Persistence Preserving Framework

Persistence Preserving Framework usually defines a score function for each point of a trajectory. The accuracy of an algorithm in this category is based on whether or not the algorithm can detect and preserve the points with high score values according to the definition of the score function used by the algorithm. A good example to demonstrate topological persistence preserving framework is from Katsikouli et al.'s work [10] discussed below.

Katsikouli et al. [10] proposed a trajectory simplification algorithm named β Persistence Algorithm which attempts to identify large scale sharp features of trajectory. The points closer to those with large features of trajectory should be preserved by the β Persistence Algorithm to represent the original trajectory. The entire compression method consists of two steps.

In the first step, it converts the trajectory to a sequence of curvature values. Fig.1(a) illustrates what the curvature means and Fig.1(b) shows the process of transformation from trajectory data points to sequence of curvature values. In order to better explain the sweep algorithm, we use terminology of graph theory such as vertex to represent curvature value of a location point, and connected component to represent a set of curvature values of consecutive points in a trajectory. Then, from the sequence of curvature values from the first step, it identifies the set of local maxima $\{V_{m_1}\cdots V_{m_i}\}$ and the set of local minima $\{V_{n_1}\cdots V_{n_j}\}$ of the curvature function values. Initially, for each local minimum vertex V_{n_j}, it creates a connected component with only one such vertex $C\{V_{n_j}\}$. Then, it iteratively expands each connected component by one neighboring vertex. Note that each vertex (other than the first and last) has two neighboring vertices V_a and V_b. It chooses V_b if $V_a > V_b$. Each connected component $C\{V_{n_j} \cdots V_r\}$ is annotated as a pair of vertices, one is the starting local minimum vertex of the connected component V_{n_j}, the other is the most recently added vertex of the connected component V_r. A connected component will stop expanding once one local maxima vertex $V_{m_i} \in \{V_{m_1} \cdots V_{m_i}\}$ is added to the connected component. When the other connected component $C\{V_{n_{\hat{j}}} \cdots V_{\hat{r}}\}$ adds the above local maxima vertex V_{m_i} and $V_{n_{\hat{j}}} < V_{n_j}$, the previous connected component $\{V_{m_1} \cdots V_{m_i}\}$ will be terminated and merged into the other connected component $C\{V_{n_{\hat{j}}} \cdots V_{\hat{r}}, V_{m_i}\}$.

Once the connected component $\{V_{m_1} \cdots V_{m_i}\}$ is complete, the lifespan of that component will be calculated as $V_{m_i} - V_{m_1}$. The higher the lifespan of a connected component, the sharper is the feature it defined by that component.

The algorithms of persistence based framework can achieve linear running time; however, their performance is not guaranteed when the error bound is tight; further, the sharp features used to process the incoming points must be defined by users.

3 Compression Using Attribute Partitioning

3.1 What is Attribute Partitioning?

Attribute partitioning [4] in relational databases splits a relation into a set of sub-relations, each containing a subset of attributes of the original relation [2]. Every new sub-relation thus obtained should contain attributes of the original relation and the union of the sub-relations should be equivalent to the original relation. However, several new sub-relations may contain the same attributes. We will use the phrase “attribute partitioning” to ensure consistency with existing literature. In general, attribute partitioning attempts to reduce transaction costs associated with databases. Previous work has shown that correlations between attributes can be exploited in compressing multi-attribute trajectories by placing such attributes in the same sub-relation. Solving the challenges associated with efficient compression of the multi-attribute data generated from connected vehicle operations is critical to transportation planners and researchers. This section will survey previously developed heuristic methods which aim to solve attribute partition effectively and efficiently.

3.2 Attribute Partition Methods

There are two perspectives to view the problem, and they lead to two different methodologies. The first approach is designing heuristic methods to overcome the intractable nature of the underlying mathematical optimization problem. The other is to study the problem using graph theoretic techniques. In the literature, graph theoretic methods chronologically follow those based on mathematical optimization.

Current attribute partition methods have three components in common. The first component is the Similarity Metric which measures the similarity or distance between attributes or usage among data items. This metric is the basis for clustering and any other possible learning methods which require distance metric. The second is the Cost Function which incorporates the information about the usage pattern and data characteristics while considering the access cost for different levels of the memory hierarchy (e.g. cache, main memory and disk) in a computer system.

The third component is the algorithm used to search a partition which optimizes the cost or utility functions. The number of partitions of a set is known as the Bell number, which grows faster than the factorial function. Due to this level of complexity, there are many heuristic algorithms in previous works that attempt to reduce the search space, leading to two classes of methods. One class is based on machine learning methods, such as the bond energy clustering and greedy hill-climbing. Under the hill-climbing method, there exists some subtlety with respect to the initial state. The initial choice is generally problem or context sensitive. The other class of methods address the attribute partition problem using graph theoretic techniques. The primary advantage with graph-related algorithms is the cost function need not be explicitly defined.

3.2.1 Clustering Methods

The problem of attribute partition is cast as the problem of clustering in machine learning [6]. Clustering methods group observations into different clusters, where differences within clusters are minimized and difference between clusters are maximized. Many comprehensive reviews on clustering methods are available in the literature [9][26]. Therefore, we focus on a detailed example to illustrate the application of clustering methods for the problem of attribute partition. Two major components of any clustering method are the definition of similarity metric and the algorithm for generating clusters. The example below will illustrate the two components in detail.

Hoffer et al. [7][4][21] discuss the process of formulating a similarity metric between attributes. The following is one example from their paper.

Given a set of attributes $A = \cup_{i=1}^{|A|} a_i$ and a set of records $R = \sum_{j=1}^{|R|} r_j$, a collection of n attribute partitions are represented as a collection of n pairs (A_i, R_i), where, $\cup_{i=1}^{n} A_i = A$ and $\cup_i R_i = R$. The quantity l_{a_i} denotes the encoding length of attribute a_i.

Given a set of retrieval requests $T = \sum_{k=1}^{k=|T|} t_k$, $p(t_k, a_i)$ denotes the probability of retrieving attribute a_i in request t_k, and $p(t_k, a_i, a_j)$ denotes the probability of retrieving attributes a_i a_j together in request t_k. Also w_{t_k} is defined as the weight of request t_k in T.

With the above definitions, for each combination of attributes a_i, a_j and request t_k, a metric $c(a_i, a_j, t_k)$ is defined as follows:

$$\frac{l_{a_i} * p_{t_k,a_i} + l_{a_j} * p_{t_k,a_j}}{(l_{a_i} + l_{a_j}) * \max(p_{t_k,a_i}, p_{t_k,a_j})}$$

and $s(a_i, a_j, \alpha)$ as an aggregate function of $c(a_i, a_j, t_k)$ is defined by

$$\frac{\sum_{k=1}^{k=|T|} w_{t_k} c(a_i, a_j, t_k)^\alpha}{\sum_{k=1}^{k=|T|} w_{t_k} c(a_i, a_j, t_k)^\alpha}$$

The function $s(a_i, a_j, \alpha)$ is defined to capture the intuition that attributes with similar retrieval patterns are closer in metric value. Also, it is context sensitive and can vary according to the context of the attribute partition problem. In general, the similarity function is monotonic in each parameter chosen by users.

After the calculation of similarity matrix M_α, where $M(i, j) = s(a_i, a_j, \alpha)$, there are a variety of clustering algorithms [9] to apply to the matrix depending on the application. We present the Bond Energy Algorithm [7].

The Bond Energy Algorithm permutes the columns and rows of M_α to get a new matrix $\hat{M}_\alpha$. The new $\hat{M}_\alpha$ maximizes the objective function

$$\sum_{\hat{i}=1}^{\hat{i}=|A|} \sum_{\hat{j}=1}^{\hat{j}=|A|} s(a_{\hat{i}}, a_{\hat{j}}, \alpha) * [s(a_{\hat{i}-1}, a_{\hat{j}}, \alpha) + s(a_{\hat{i}}, a_{\hat{j}-1}, \alpha) + s(a_{\hat{i}+1}, a_{\hat{j}}, \alpha) + s(a_{\hat{i}}, a_{\hat{j}+1}, \alpha)]$$

. An intuitive explanation of such an objective function is that it maximizes the nearest neighbor strengths. The procedure of partitioning the attributes is based on the new matrix $\hat{M}_\alpha$.

There are a variety of clustering algorithms other than the Bond Energy Algorithm. They differ in the formation of objective function and algorithms to find clusters. The appropriateness of each algorithm depends on the application.

In essence, clustering methods free users from the burden of defining objective functions. However, users must choose the number of clusters beforehand.

3.2.2 Hill-climbing Method

Hammer [4] applies Hill-climbing method to the attribute partition problem. The Hill-climbing algorithm consists of two components. The first component is informed search for a next partition derived from a previous partition. The second one is a heuristic function to examine the gain of a new partition over previous partition. It greedily chooses the best new partition as current partition according to the heuristic function and user-defined condition. It terminates when the current partition cannot be improved.

Given a set of attributes $A = \cup_{i=1}^{|A|} a_i$, and an evaluation function F, the beginning partition is initialized as a set of single attribute blocks $\{P_i^1\}$ where $P_i^1 = \{a_i\}$, $|\{P_i^1\}| = |A|$. One possible next partition with respect to the previous partition $\{P^j\}$ where $|\{P^j\}| = c > 0$ is created by merging any two attribute blocks P_x^j and P_y^j indexed by x, y respectively where $0 < x < c$, $0 < y < c$ and $x \neq y$ and keeping other attribute blocks P_l^j where $l \neq x$ and $l \neq y$ and $l \in \{1, \ldots, c\}$ intact.

For all the possible partitions with respect to $\{P^j\}$, choose one partition P as $\{P^{j+1}\}$ with the condition that $F(P) > F(\{P^{j+1}\})$, where $F(P)$ is the maximum of F among all possible next partitions. The Hill-climbing algorithm terminates once there exist no such P that satisfies the condition.

The Hill-climbing method is a greedy approach because it only explores the branch $\{P^{j+1}\}$ that currently scores the best with respect to an evaluation function F and ignores other possible partitions with respect to $\{P^j\}$. So, in each iteration, the set of possible next partitions shrinks as the algorithm continues.

The method of Navathe et al. [22] differs from Hammer's [4] approach. Instead of merging attribute blocks in a bottom-up fashion like Hammer, it splits an attribute block into two during the process starting from the coarsest partition $\{a_1, a_2, \cdots, a_{|A|}\}$ in a top-down fashion.

A straightforward implementation of hill-climbing is very inefficient in terms of time complexity. In order to reduce the number of attribute block evaluations, it memoizes the gain induced by merged attribute blocks, which is assumed to be consistent along the search process. The cost reduction between a new merged attribute block and other blocks will be calculated and memoized. The memoization leads to time complexity $O(|A|^3)$.

3.2.3 Graph-based Methods

Son et al. [8] views the problem from the perspective of graph theory and propose Fuzzy α-cut Algorithm. It defines a graph $G(V, E)$, where the vertex set V corresponds to the set of attributes and the edge set E corresponds to the affinity value between any two vertices. Then, it defines two variables $X(e)_{cohesive}, X(e)_{discohesive}$ for each edge $e \in E$ with the constraint that

$$X(e)_{cohesive} + X(e)_{discohesive} = 1.$$

Here, $X(e)_{cohesive}$ defines the degree of affinity between the two vertices connected by e, which is calculated through the user defined affinity score function of two vertices. Finally, a variable named certainty given by

$$C(e) = \max(X(e)_{cohesive}, X(e)_{discohesive})$$

is defined over each edge e.

The Fuzzy α-Cut Algorithm is based on the assumption that if the certainty $C(e)$ is closer to 1, then it is easier to decide whether the two vertices connected by e are better placed together or not; if $C(e)$ is close to 0.5, then it is harder to determine whether the two vertices should be together. The algorithm cuts any edge $e \in E$ if such e satisfies the condition $C(e) < \alpha$. The process creates several sub-graphs $G(\hat{V}, \hat{E})$, where $\hat{V} \in V$ and $\hat{E} \in E$. Also, a new sub-graph $G(\overline{V}, \overline{E})$, where $\overline{E}$ is made of any e deleted from the previous cutting steps.

The determination of an optimal α-cut is based on an aggregation function $f(\alpha)$ which is directly based on score values of $G(\hat{V}, \hat{E})$s and $G(\overline{V}, \overline{E})$. For any sub-graph $G(V, E)$, the score value is given by

$$\sum\nolimits_{e_i \in E \text{ and } e_j \in E} |C(e_i) - C(e_j)|,$$

which denotes the total uncertainty within the sub-graph $G(V, E)$. An optimal $\hat{\alpha}$-cut is denoted by $\min\limits_{\hat{\alpha}-cut} \{f(\alpha)\}$.

The running time of the Fuzzy α-Cut Algorithm is very sensitive to $|\{\alpha\}|$, the size of the set of distinct affinity values. The worst case is when every affinity value between any two attributes is distinct, then the time complexity is $O(n^2)$, where $n = |V|$.

4 Metric

4.1 Accuracy Metric

Accuracy metric measures the similarity between the compressed trajectory and the original trajectory. Accuracy metrics can be spatial, temporal-spatial, direction-based, speed-based, or direction-speed based, or temporal based. Each error metric is associated with the requirement of specific applications and moving objects traveling mode and geographic conditions. Applications with direction-based metrics can easily capture the swift direction transition in traveling mode. However, this representation of simplified trajectory may be too fine for applications which may require only the rough positional information. Speed error metric is used in many online compression algorithms with linear running times. Temporal-spatial error metrics are popular in $O(n \log(n))$ online and offline compression algorithms.

Accuracy metric is also highly related to the algorithm for recovering an uncompressed trajectory from a compressed one. It is of interest to explore the relationship between different combinations of accuracy metrics in compression and decompression parts of algorithms.

Accuracy of a single point in the compressed trajectory with respect to a point in the original trajectory can be readily defined. It is also easy to define an aggregation function on the accuracy of all points. Some use the maximum value [16] while others use geometric means [19]. Spatial and spatial-temporal metric have been discussed in many other survey papers [20]. Here we focus

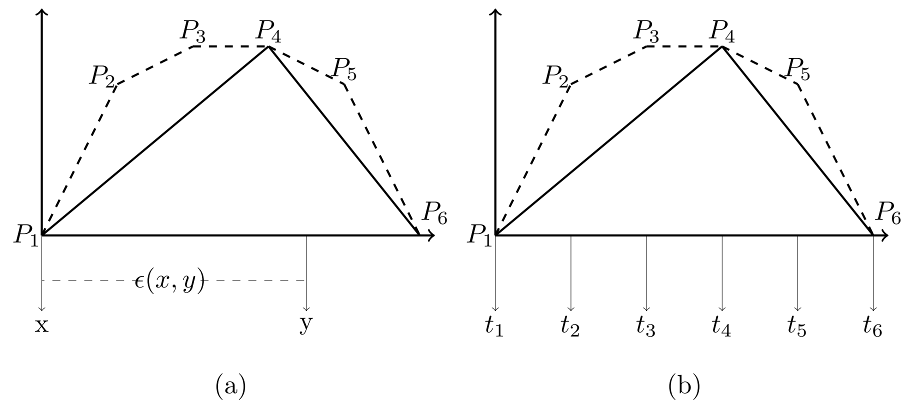


Figure 2: Compression Accuracy Metrics: Direction-based error metric and Speed-based error metric are illustrated in (a) and (b) respectively.

on the other metrics such as Direction-based metric, Speed-based metric and Semantics-based metric, none of which has been surveyed in other papers. In Fig.2, given a trajectory T which consists of $(P_1, P_2, P_3, P_4, P_5, P_6)$ and its compressed representation $\hat{T}$ which consists of (P_1, P_4, P_6), we will illustrate the following metrics on the trajectories T and $\hat{T}$.

4.1.1 Direction-Based Error Metric

Direction-based error metric has been studied in several research papers [10][16]. It can be used to detect important points in trajectory. In the paper by Long et al. [17], it also theoretically proven that Direction Persistence Simplification algorithm with error bound with respect to Direction-based error metric can achieve bounded approximation error defined by the spatial error metric.

In Fig.2(a), the direction-based error of $\hat{T}$ with respect to a specified period of time $\epsilon(x, y)$ in T is defined as the maximum angular difference between the direction of movement in T and the direction of the movement in $\hat{T}$. The distinct feature of the Direction-based metric is that it specifies the range of time. For the specified time $\epsilon(x, y)$, it identifies all the location points (P_2, P_3, P_4) generated during the same time span. For any location point P_i, and time of location point $t(P_i)$, it will be included if $x < (P_i) \leq y$. For each of (P_2, P_3, P_4), it identifies a pair of predecessor and successor points in $\hat{T}$ as compressed angles. This is stored as the edge list $E = \{(P_1, P_4), (P_1, P_4), (P_1, P_4), (P4, P_6)\}$. Then, for each of (P_2, P_3, P_4), it identifies a pair of predecessor and successor points in T as the original angles. This is stored as the edge list $\hat{E} = \{(P_1, P_2), (P_2, P_3), (P3, P_4), (P_4, P_5)\}$. Then a maximum absolute angle difference between corresponding edge lists represents the Direction-based error at$\epsilon(x, y)$. Fig 2(a) showcases the calculation of angular difference between two edges. The reason for using maximum angular difference instead of average angular difference is that the former can better preserve the shape of the original trajectory.

4.1.2 Speed-Based Error Metric

Speed-based error metric is used whenever it is important to preserve the speed information in the compressed form [25][23]. It is very similar to Direction-based error metric. Fig.2(b) illustrates the calculation of Speed-based error metric. Each point P_i occurs at time stamp t_i. Given the point P_3 at time t_3 and another point P_4 at time t_4, the speed at P_3 is computed as $(P_3, P_4)/t_4 - t_3$. Given a time span $\epsilon(t_1, t_6)$, the speed error during the time span can be an aggregate of speed error of each location point which happens in the same time span. The appropriate aggregate function depends on the specific application. For example, it can be the average, median or maximum of the list of speed errors of individual points.

4.1.3 Semantics-Based Metric

Semantics-based metric usually incorporates road network information or other information to measure the accuracy of compression. Such metrics have been used in many studies [27][14][23]. In Zheng et al. [27], two specific semantics-based metrics are proposed. One is Time Synchronized Network Distance and the other is Network Synchronized Time Distance. Both of these error metrics utilize the road network information. Given two Euclidean space trajectories with time stamps

$$T((P_1, t_1), (P_2, t_2), \cdots, (P_n, t_n))$$

and

$$\hat{T}((\hat{P_1}, \hat{t_1}), (\hat{P_2}, \hat{t_2}), \cdots, (\hat{P_n}, \hat{t_n})),$$

the approach in [27] converts them into an appropriate distance and time space $d - t$. The process starts with a matching step which transforms the sequence T into another sequence $((e_1, t_1), (e_2, t_2), \cdots, (e_k, t_k)$, where (e_i, t_i) denotes at the time t_i, the trajectory at the beginning of e_i. The transformation process must shift the time stamp to obtain a correct presentation of such a trajectory. In a road network, the length e of each edge is known, so the distance covered at the time t_j is just $\sum_{i=1}^{i=j} e_i$.

Given a trajectory T and its compressed version $\hat{T}$, TSND measures the maximum difference between the distance object travels via trajectory T and that via trajectory $\hat{T}$ during any time span. The concept is illustrated at Fig.3(a). The figure shows the TSND value which is the maximum vertical distance between T and $\hat{T}$.

NSTD defines the maximum time difference between a trajectory T and its compressed form $\hat{T}$ while traveling the same distance with $NSTD(T, \hat{T}) = Max_{d_x}(|Tim(T, d_x) - Tim(\hat{T}, d_x)|)$. This is illustrated at Fig.3(b). The figure indicates the NSTD value which is the maximum horizontal distance between T and $\hat{T}$.

4.2 Performance Metric

The performance of an algorithm is generally specified using space and time complexity measures. When there is a bound on the compression ratio or a

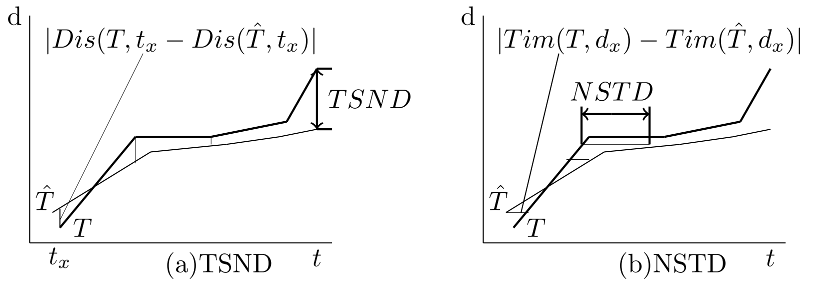

d
$|Dis(T, t_x - Dis(\hat{T}, t_x)|$
$TSND$
$\hat{T}$
T
t_x
(a)TSND
t
d
$|Tim(T, d_x) - Tim(\hat{T}, d_x)|$
$NSTD$
$\hat{T}$
T
(b)NSTD
t

Figure 3: $TSND$ and $NSTD$

Table 1 Compression Schemes Summary (m: number of attributes, n: length of trajectory, r: size of reduced set of original attributes, λ: target compression ratio, Ψ: spatial error bound, μ: SED error bound, σ: discretization error bound, ϵ: timing error bound, l: loss function bound).

Scheme	Method	Param.	Time Complexity	Space	SED(Same Compression Ratio)
Trajectory	Douglas Peucker	Ψ	$O(nm^2)$	$O(nm)$	large
	Squish-E(λ)	λ	$O(nm\log(n/\lambda))$	$O(m\log(nm\lambda))$	small
	Squish-E(μ)	μ	$O(nm\log(n))$	$O(nm)$	small
Partition	Hill Climb	λ	$O(m^3 + nm\log(n/\lambda))$	$O(m\log(n/\lambda))$	small
	Optimized HC	λ	$O(m^2 + nm\log(n/\lambda))$	$O(m\log(n/\lambda))$	small
	Clustering	λ	$O(m^2 + nm\log(n/\lambda))$	$O(m\log(n/\lambda))$	small
	Graph	λ	$O(m\log(m) + nm\log(n/\lambda))$	$O(m\log(n/\lambda))$	small
Reduction	Selection	λ,l	$O(m\log(m) + rn\log(n/\lambda))$	constant	large
	Extraction	λ,l	$O(m\log(m) + rn\log(n/\lambda))$	constant	large
TS Method	Multi-Pattern	σ, ϵ, l	$O(m\log(m)\log(n))$	constant	large
	Single-Pattern	σ, ϵ, l	$O(m\log(n))$	constant	large

strict accuracy requirement, it is usually necessary to choose algorithms with larger space or time complexity measures.

4.2.1 Space Complexity

Space complexity refers to the amount of internal memory for compression schemes. The linear time complexity implies linear space complexity. Normally, a time complexity of $O(n\log(n))$ usually means that the space complexity is also $O(n\log(n))$.

4.2.2 Time Complexity

Comparison based algorithms typically have a time complexity of $O(n\log(n))$. Examples of such algorithms include Squish-E [20], Douglas-Peucker (modified) [5] and Bounded Convex Hull [15]. These algorithm use efficient data structures to avoid the comparison between the incoming point and all the previous points. However, these algorithm still need at least $\Omega(\log(n))$ look up time in order to determine weather or not to keep the incoming point. The reason why some online algorithms have linear running times is that they discard each incoming point using a constant running time decision function which doesn't depend on the total number of points. However, these algorithms with linear running times may have significantly larger values for error metrics.

5 Comparisons

In this section we are going compare two known compression techniques, namely PROXIMUS and CompreX.

5.1 PROXIMUS

The first result tested through was PROXIMUS, which provides a technique for compressing larger data sets of representative patterns, on which traditional analysis algorithms can be applied. This framework is a non-orthogonal matrix transform and it relies on recursive partitioning of a data set. The dominant pattern is computed as a binary approximation vector of the matrix of relations. PROXIMUS provides several facilities to analyze data with discrete attributes. These include:

- discovering dominant and inconsistent patterns in the data in a hierarchical manner,
- clustering of data in an error-bounded and physically understandable form,
- finding accurate data representation and
- isolating signal from noise in a multi resolution framework [12].

Previous research has shown that PROXIMUS has the ability to accurately represent data after compression. By focusing on the binary nature of discrete data, the compression technique provides a framework and takes advantage of the discrete data in order to improve efficiency.

5.1.1 Computational Procedure

Unlike singular value decomposition (SVD)-based techniques, PROXIMUS uses a rank-one approximation of the input matrix for row decomposition. Given m binary vectors $a_1, a_2, \ldots, a_m$ in an n-dimensional space, we find binary approximation vectors $y_1,\ y_2,\ \ldots,\ y_k$, where each y_j is a $n \times 1$ vector $(1 \leq j \leq k)$, such that

$$\forall i\ (1 \leq i \leq m)\ \exists j\ \text{ such that }\ ||a_i - y_j||_2^2\ <\ \epsilon.$$

We also need to minimize k, when there is a prescribed error bound [12]. This algorithm proceeds by recursively computing discrete rank-one approximation to the matrix, locating appropriate patterns and organizing them. The purpose of finding a discrete rank-one approximation of binary matrices is find a low rank decomposition that approximates groups of rows with local patterns. The solution will be associated with a local pattern and we can apply additional methods to collect other local patterns in the matrix. The partition process continues until the set represents the entire matrix. Then the concept of Hamming radius is used to determine whether the pattern represent all rows of the corresponding submatrix. While finding the rank-one approximation, the initial pattern vector must have a magnitude which is *strictly larger than* zero. Possible methods for finding a correct initial pattern vector include the following:

- Partition: Select a separator column and identify the rows that have a nonzero at that column.

- Greedy Graph Growing: The initial pattern vector is set to the centroid of rows in this part. (This approach is also useful in association rule mining.)

First, we formulate the problem of error-bounded approximation of binary matrices. Our approach to solving this problem is based on recursively computing discrete rank-one approximation to the matrix. The rank-one approximation problem for matrix A with m columns and n rows is equivalent to finding two vectors, x and y, that maximize the number of zeros in the matrix. That is,

$$||A - xy^T||_F^k = |a_{ij} \in (A - xy^T) : \ a_{ij} = 1|.$$

The error for a rank-one approximation is the number of nonzero entries in the residual matrix. the main purpose here is to find a low-rank decomposition that approximates groups of rows with local patterns rather than a globally optimal rank-one approximation. Next, recursively decompose the binary matrices in order to get rows corresponding to 1's in the presence vector which are maximally connected submatrices of the main matrix. Let $1 \leq i \leq m$. Then

$$a_i \in \begin{cases} A_1 & \text{if } x_i = 1 \\ A_0 & \text{otherwise} \end{cases}$$

with a_i denoting the ith row of A. The partitioning-and-approximation process continues until the matrix cannot be further partitioned or the resulting approximation adequately represents the entire matrix. Finally, the concept of Hamming radius is used as the major stopping criterion for the algorithm to decide whether the underlying pattern can represent all rows of the corresponding submatrix adequately. Given a set of binary vectors $R = \{x_1, x_2, ..., x_n\}$ and a binary vector y, the Hamming radius of R directly centered within y is defined by

$$r(R, y) = \max\{h(x_i, y)\}$$

where $h(x, y) = ||x - y||_2^2$ is the Hamming distance between binary vectors x and y. If all the conditions hold, the pattern vector is considered a dominant pattern in matrix A_i and recorded with its presence vector within the approximation of A.

5.2 CompreX

This method identifies anomalies in large multi-dimensional databases using pattern-based recognition. It finds a collection of dictionaries that describe the norm of a database succinctly, and subsequently flags those points which differ significantly from the norm. Simply put, the plan of action is to use a set of dictionaries to encode a database. CompreX builds off of previous research from the KRIMP compressor. It employs the Minimum Description Length (MDL) principle to decide on the number of groups. This avoids the mining and filtering the collection of patterns.

We use the set of code tables to compress. According to the MDL principle we calculate the length of the codes and the tables to find the minimum possible size of the compressed representation.

To begin, one must compute the IG matrix for all pairs of the current feature sets; this is a non-negative and symmetric matrix. Let $|F_i|$ denote number of features in set F_i. Pairs of feature sets are placed in a decreasing order of IG-per-feature (normalized by total cardinality), and the outer iterations begin to revise the pairs as the candidate CT's to be merged (CT$_i$ and CT$_j$). The initial cost given by CT$_{init}$ is equivalent to the total cost with the initial set of CTs. Then we construct CT$_{i|j}$ using the existing patterns $p_{i,1}, ...p_{i,n_i}$ and $p_{j,1}, ...p_{j,n_j}$ from CT$_i$ and CT$_j$ into the new CT. They are first ordered by length and next by usage. Removing code table sets CT_i and CT_j along with the inclusion of the concatenated set $\hat{CT}_{i|j}$ from P builds our candidate tables $\hat{CT}$; the same process is used for candidate partitioning of F.

Afterwards, all the unique rows of the database induced on $F_{i|j}$ are collected. These are then organized in decreasing order of their occurrence in the database and directly inserted into the new CT. Let $p_{i|j,1}, ..., p_{i|j,n_{i|j}}$ denote these patterns of the combined feature set $F_{i|j}$ in their sorted order of frequency. In our inner iterations, we insert these one-by-one, update the usages of the existing overlapping patterns, remove those patterns whose usage drops to zero, recompute the code word lengths with updated usages and compute the total cost after each insertion. If total cost is reduced, we store the candidate partitioning $\hat{P}$ and associated set of coded tables $\hat{CT}$; otherwise, we continue insertions with the next candidate patterns for possible future cost reduction.

In the outer iterations, if total cost is reduced, the IG between the new feature set $F_{i|j}$ and the rest are computed. Otherwise, the merge is rejected and the candidates $\hat{P}$ and $\hat{CT}$ are discarded. Next, the algorithm continues to search for future merges, until no more pairs of feature sets can be merged to reduce the cost. The resulting set of feature sets and their corresponding set of code tables represents the solution [1].

5.3 Results

The datasets which were compressed using PROXIMUS and CompreX were donated by a member of the UTRC consortium. Each dataset is a trajectory with additional attributes produced by on-board truck operations equipment. Each data set had a total of 22 columns, each with its own unique set of rows. Data set 1 contained 926 rows, set 2 contained 645 rows and the third and final set contained 1004 rows.

Table 2 Results of compressing the same files using Proximus and CompreX

Datasets	Original Size(.txt file)	CompreX Output	Proximus Output
Dataset1	83kb (.csv file size-109kb)	139kb (input size-194kb)	5kb (I.Size 35kb, After binary conversion)
Dataset2	60kb (.csv file size-77kb)	78kb (input size-95kb)	3.6kb (I.Size-31kb, After binary conversion)
Dataset3	80kb (.csv file size-111kb)	117kb (input size-163kb)	3.4kb (I. Size-31kb, After binary conversion)

Figure 4 above represents the rankings of Proximus and CompreX based on three criteria, namely data recovery, compression ratio and compression time. The data is first run through the Hill-Climbing method for grouping before

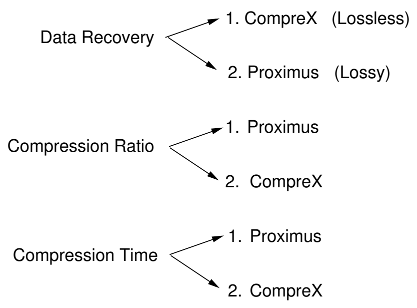


Figure 4: Ranking of compression methods Proximus and CompreX

applying the compression methods. The results show that both methods have their own strengths and weaknesses. PROXIMUS, being a more Lossy compression technique, is time efficient. On the other hand, CompreX, a lossless compression technique, takes more time to compress the data. Our results show that neither of these methods is superior to the other. The decision regarding which to choose depends on which compression criteria are more important to a user.

6 Summary, Conclusions and Future Research

We implemented some attribute partition algorithms for multiple attribute trajectory compression. We view the trajectory data as a big table. This enables us to apply previous work on big table compression on multi-attribute trajectory data sets. Different views of data can lead to different compression schemes. We used a point selection algorithm to compress both spatial and non-spatial parts of data sets. Non-spatial data sets can also be compressed using methods other than point selection. In general, highly correlated data attributes can be compressed together to reduce the size of the compressed representation. However, our results indicate that the attributes in our data sets are not correlated. Also, for most combinations of error bounds, the best partition for compression is the finest one (i.e., each attribute in a group by itself). In some cases, even though some attributes appear to be related, having them in the same group did not improve the compression ratio. We believe that further work should consider methods other than point selection to compress non-spatial attributes; instead, it may be best to view these non-spatial attributes as a time series and employ methods for compressing time-series data. A probabilistic view of error bounds may also achieve better compression ratios for non-spatial attributes. Such methods can also exploit ideas and results from the machine learning literature.

We chose Douglas-Peucker compression technique as the baseline method for trajectory compression. The reason is that it provides a high compression ratio. In addition, Douglas-Peucker method can be conveniently adapted for non-trajectory compression since it is a line simplification algorithm. Further, the Douglas-Peucker method does not rely on many assumptions about the data being compressed besides the definition of a distance. In contrast, newly developed trajectory compression methods such as Bounded Convex Hull rely on many geometric properties of a trajectory and cannot be used for non-spatial data compression. Even though the running time of Douglas-Peucker compression is quadratic, it achieves better compression ratios compared to other trajectory compression methods. For very large trajectory datasets, it is more appropriate to use an $O(n \log(n))$ compression algorithm such as Squish-E. When the allowed error bound is large, linear running time trajectory compression algorithms can be used.

For compressing non-spatial attributes in multi-attribute trajectory data, we chose attribute grouping. Attribute grouping method do not require any preprocessing steps. Methods based on attribute reduction would incur severe penalties with respect to accuracy. A naive method to achieve an optimal attribute partition is exhaustive search. This method, which tries all possible groupings, is computationally feasible only when the number of attributes is reasonably small. So we adopted the hill climbing heuristic method with compression size as the objective function to reduce the time complexity. Since naive hill climbing has a time complexity of $O(n^3)$, we implemented a hill climbing method with an internal data structure to reduce time complexity to $O(n^2)$. Clustering-based attribute grouping method also has a time complexity of $O(n^3)$. Graph-based attribute grouping has a time complexity of $O(n \log(n))$; however, in general, its compression ratio is worse than that of the hill-climbing method.

Further work should explore some new directions with respect to non-spatial attribute compression and multi-attribute compression. One direction is to consider appropriate modifications to attribute reduction methods for non-spatial attributes. When some attributes are highly correlated or can be predicted with high accuracy from the values of other attributes, attribute reduction methods may be appropriate. If error bounds can be relaxed, interdependence among attributes can be utilized in attribute selection. This approach has the potential to completely eliminate redundant attributes.

In our work, we found that point selection methods perform poorly on attributes which are essentially binary-valued. It may be appropriate to use methods other than point selection to compress such attributes. Also, it is of interest to explore representations of non-spatial attributes as time series and develop appropriate compression techniques. One disadvantage of such representations is that they may introduce other errors related to time. However, these representations provide several benefits with respect to space when there are relationships among the attributes along the time domain.

As mentioned earlier, a clear understanding of the techniques for compressing multi-attribute trajectory data is an essential first step for developing techniques for extracting information from compressed traces. In general, information extraction techniques will rely on both the method used to compress the data as well as the set of queries that will arise in applications. Also, one may choose an appropriate compression method given the set of queries that must be processed on the compressed traces. In such a case, the expected levels of accuracy in the responses to the queries may determine the error bounds on the attributes to used by a compression method. Thus, techniques for information extraction must address the complex interaction between compression techniques and the classes of queries that are of interest to transportation planners.

Acknowledgments

This work was supported by the Research and Innovative Technology Administration of the US Department of Transportation through the Region 2 – University Transportation Research Centers (UTRC) Program. The data used in this study was donated by a member of the UTRC consortium for the purposes of exploring multiple attribute compression. The data was produced from on-board truck operations equipment.

We also thank the following students from the University at Albany – SUNY who actively participated in this research as well as in the preparation of this report: Hang Lin, Xiaohu Nian, Patrick Piedra, Rahul Srivastava and Zeyang Wu.

References

[1] Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. Fast and reliable anomaly detection in categorical data. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 415–424. ACM, 2012.

[2] Gajanan S. Chinchwadkar and Angela Goh. An overview of vertical partitioning in object oriented databases. *The Computer Journal*, 42(1):39–50, 1999.

[3] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[4] Michael Hammer and Bahram Niamir. A heuristic approach to attribute partitioning. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, pages 93–101, New York, NY, USA, 1979. ACM.

[5] John E. Hershberger and Jack Snoeyink. *Speeding up the Douglas-Peucker line-simplification algorithm.* University of British Columbia, Department of Computer Science, 1992.

[6] Jeffrey A. Hoffer and Dennis G. Severance. The use of cluster analysis in physical data base design. In *Proceedings of the 1st International Conference on Very Large Data Bases*, pages 69–86. ACM, 1975.

[7] Jeffrey A. Hoffer and Dennis G. Severance. The use of cluster analysis in physical data base design. In *Proceedings of the 1st International Conference on Very Large Data Bases*, pages 69–86. ACM, 1975.

[8] Jin HyunSon and Myoung HoKim. -Partitioning algorithm: Vertical partitioning based on the fuzzy graph. In HeinrichC Mayr, Jiri Lazansky, Gerald Quirchmayr, and Pavel Vogel, editors, *Database and Expert Systems Applications*, volume 2113 of *Lecture Notes in Computer Science*, pages 537–546. Springer Berlin Heidelberg, 2001.

[9] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[10] Panagiota Katsikouli, Rik Sarkar, and Jie Gao. Persistence based online signal and trajectory simplification for mobile devices. 2014.

[11] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD Rec.*, 30(2):151–162, May 2001.

[12] Mehmet Koyutürk, Ananth Grama, and Naren Ramakrishnan. Compression, clustering, and pattern discovery in very high-dimensional discrete-attribute data sets. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):447–461, 2005.

[13] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[14] Hengfeng Li, Lars Kulik, and Kotagiri Ramamohanarao. Spatio-temporal trajectory simplification for inferring travel paths. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 63–72. ACM, 2014.

[15] Jiajun Liu, Kun Zhao, Philipp Sommer, Shuo Shang, Brano Kusy, and Raja Jurdak. Bounded quadrant system: Error-bounded trajectory compression on the go. *arXiv preprint arXiv:1412.0321*, 2014.

[16] Cheng Long, Raymond C. Wong, and H. V. Jagadish. Direction-preserving trajectory simplification. *Proceedings of the VLDB Endowment*, 6(10):949–960, 2013.

[17] Cheng Long, Raymond C. Wong, and H. V. Jagadish. Trajectory simplification: On minimizing the direction-based error. *Proceedings of the VLDB Endowment*, 8(1), 2014.

[18] Nirvana Meratnia and A. Rolf. Spatiotemporal compression techniques for moving point objects. In *Advances in Database Technology-EDBT 2004*, pages 765–782. Springer, 2004.

[19] Jonathan Muckell, Jeong-Hyon Hwang, Catherine T. Lawson, and S. S. Ravi. Algorithms for compressing GPS trajectory data: an empirical evaluation. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 402–405. ACM, 2010.

[20] Jonathan Muckell, Paul W. Olsen, Jeong-Hyon Hwang, Catherine T. Lawson, and S. S. Ravi. Compression of trajectory data: a comprehensive evaluation and new approach. *GeoInformatica*, 18(3):435–460, 2014.

[21] Shamkant Navathe, Stefano Ceri, Gio Wiederhold, and Jinglie Dou. Vertical partitioning algorithms for database design. *ACM Transactions on Database Systems (TODS)*, 9(4):680–710, 1984.

[22] Shamkant B. Navathe and Mingyoung Ra. Vertical partitioning for database design: A graphical algorithm. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, SIGMOD '89, pages 440–450, New York, NY, USA, 1989. ACM.

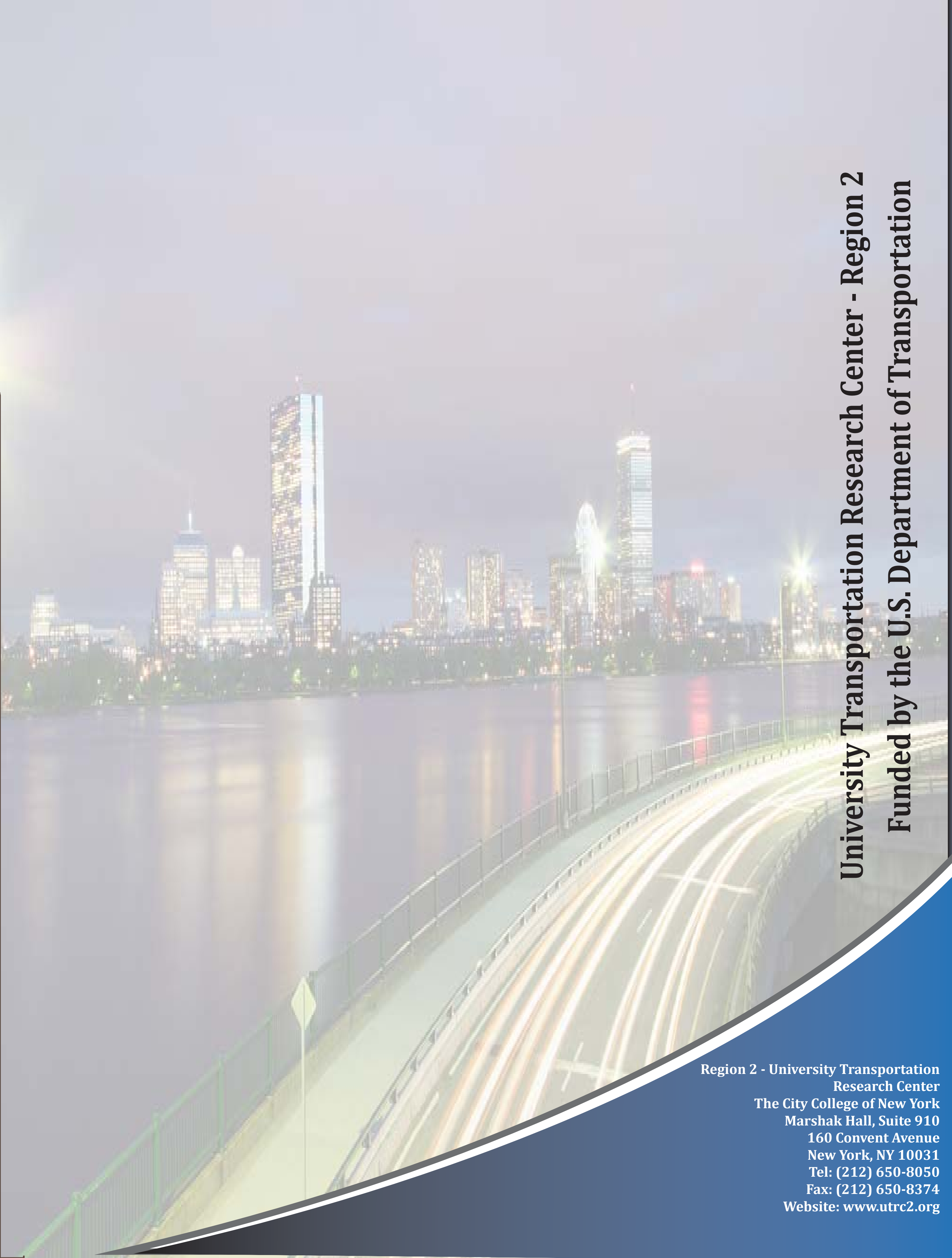
[23] Kai-Florian Richter, Falko Schmid, and Patrick Laube. Semantic trajectory compression: Representing urban movement in a nutshell. *Journal of Spatial Information Science*, (4):3–30, 2015.

[24] Goce Trajcevski, Hu Cao, Peter Scheuermanny, Ouri Wolfsonz, and Dennis Vaccaro. On-line data reduction and the quality of history in moving objects databases. In *Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access*, pages 19–26. ACM, 2006.

[25] Pan Wei, Yao Chunlong, Li Xu, and Shen Lan. An online compression algorithm for positioning data acquisition. *Informatica*, 38(4), 2014.

[26] Rui Xu, Donald Wunsch, and Others. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[27] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.



University Transportation Research Center - Region 2
Funded by the U.S. Department of Transportation

Region 2 - University Transportation
Research Center
The City College of New York
Marshak Hall, Suite 910
160 Convent Avenue
New York, NY 10031
Tel: (212) 650-8050
Fax: (212) 650-8374
Website: www.utrc2.org