

An Empirical Comparison of Outlier Detection Algorithms

Matthew Eric Otey, Srinivasan Parthasarathy, and Amol Ghoting
Department of Computer Science and Engineering
The Ohio State University
Contact: srini@cse.ohio-state.edu

Abstract

In recent years, researchers have proposed many different techniques for detecting outliers and other anomalies in data sets. In this paper we wish to examine a subset of these techniques, those that have been designed to discover outliers quickly. The algorithms in question are ORCA, LOADED, and RELOADED. We have performed an empirical evaluation of these algorithms, and here present our results as guide to their strengths and weaknesses.

1. Introduction

A common problem in data mining is that of automatically finding outliers in a database, since they can be indicative of bad data or malicious behavior. Examples of bad data include skewed data values resulting from measurement error, or erroneous values resulting from data entry mistakes. A common example of data indicating malicious behavior occurs in the field of network traffic analysis, where anomalous IP packets may indicate either a possible intrusion or attack, or a failure in the network [21]. Efficient detection of such outliers reduces the risk of making poor decisions based on erroneous data, and aids in identifying, preventing, and repairing the effects of malicious or faulty behavior. Additionally, many data mining and machine learning algorithms, and techniques for statistical analysis may not work well in the presence of outliers. Outliers may introduce skew or complexity into models of the data, which may make it difficult, if not impossible, to fit an accurate model to the data in a computationally feasible manner. Accurate, efficient removal of outliers may greatly enhance the performance of statistical and data mining algorithms and techniques [5]. As can be seen, different domains have different reasons for discovering outliers: They may be noise that we want to remove, since they obscure the true patterns we wish to discover, or they may be the very things in the data that we wish to discover. As has been said before, “One person’s noise is another person’s signal” [14].

Effective outlier detection requires the construction of a model that accurately represents the data. Over the years, a large number of techniques have been developed for building such models for outlier and anomaly detection. However, real-world data sets and environments present a range of difficulties that limit the effectiveness of these techniques. Among these

problems is the fact that the data sets may be dynamic, the fact that the data may be distributed over various sites, and the fact that the data sets may contain a mixture of attribute types (i.e. continuous and categorical attributes).

In this paper we empirically compare and contrast a set of outlier detection algorithms. This set includes ORCA and variations on the LOADED algorithm. These algorithms are designed to minimize execution times, by means of minimizing the number of passes of the data that must be made. Each addresses one or more of the problems mentioned above in a different manner, and each has their own strengths and weaknesses with respect to execution time, memory usage, and detection quality.

2. Related Work

There are several approaches to outlier detection. One approach is that of statistical model-based outlier detection, where the data is assumed to follow a parametric (typically univariate) distribution [1]. Such approaches do not work well in even moderately high-dimensional (multivariate) spaces, and finding the right model is often a difficult task in its own right. Simplified probabilistic models suffer from a high false positive rate [16, 17]. Also, methods based on computational geometry [10] do not scale well as the number of dimensions increase. To overcome these limitations, researchers have turned to various non-parametric approaches including distance-based approaches [2, 11], clustering-based approaches [7, 21], and density-based approaches [4, 20]. Here we consider these methods in more detail.

An approach for discovering outliers using distance metrics was first presented by Knorr *et al.* [12, 13, 11]. They define a point to be a *distance outlier* if at least a user-defined fraction of the points in the data set are further away than some user-defined minimum distance from that point. In their experiments, they primarily focus on data sets containing only continuous attributes. Related to distance-based methods are methods that cluster data and find outliers as part of the process of clustering [9]. Points that do not cluster well are labeled as outliers. This is the approach used by the ADMIT intrusion detection system [21].

Recently, density-based approaches to outlier detection have been proposed [4]. In this approach, a local outlier factor

(*LOF*) is computed for each point. The *LOF* of a point is based on the ratios of the local density of the area around the point and the local densities of its neighbors. The size of a neighborhood of a point is determined by the area containing a user-supplied minimum number of points (*MinPts*). A similar technique called LOCI (Local Correlation Integral) is presented in [20]. LOCI addresses the difficulty of choosing values for *MinPts* in the *LOF*-based technique by using statistical values derived from the data itself. Both the *LOF*- and LOCI-based approaches do not scale well with a large number of attributes and data points, and so are not considered in this evaluation.

A comparison of various anomaly detection schemes is presented in [15]. Its focus is on how well different schemes perform with respect to detecting network intrusions. The authors used the 1998 DARPA network connection data set to perform their evaluation, which is the basis of the KDDCup 1999 data set used in our experiments [8]. They found detection rates ranging from a low of 52.63% for a Mahalanobis distance-based approach, to a high of 84.2% for an approach using support vector machines.

3. Algorithms

In this section we give a brief overview of each of the algorithms we evaluate. For a more in-depth discussion, the reader is referred to the papers in which the algorithms were originally proposed [2, 6, 18, 19].

3.1 ORCA

Most distance-based methods for detecting outliers take time that is at least quadratic in the number of points in the data set, which may be unacceptable if the data set is very large or dynamic. Bay and Schwabacher [2] present a method called ORCA for discovering outliers in near linear time. The central idea is to perform pruning by keeping a monotonically decreasing score for each point in the data set. If the score falls below a certain threshold, then further processing on the data point is not necessary. In the worst case (when there are no outliers), the algorithm still takes quadratic time, but in practice the algorithm runs very close to linear time. Such an approach assumes that the data set is randomized, and randomization is performed on disk prior to running the algorithm. ORCA handles mixed-attribute data sets by using the Euclidean distance for the continuous attributes and the Hamming distance for the categorical attributes.

3.2 LOADED

The LOADED (Link-based Outlier and Anomaly Detection in Evolving Data sets) algorithm was first presented in [6]. It is designed explicitly for dynamic data with heterogeneous attributes. For dynamic data sets, it can process the data in one pass, and for static data sets it can make a second pass for increased accuracy.

The original version of LOADED presented in [6] is a centralized algorithm for detecting outliers in dynamic mixed-attribute

data. The central data structure used to model the data is an augmented lattice of all itemsets formed from the categorical attributes of the data. Each node in the lattice is augmented with the support count of the corresponding itemset, and the correlation matrix computed from the continuous attributes of all data points in the data set containing that itemset. Such a data structure ensures that the dependencies between all attributes, regardless of type, can be modeled. Each data point is assigned an anomaly score based on the support of all its itemsets and how well the continuous attributes agree with the relevant correlation matrices. The basic algorithm makes a single pass of the data, incrementally updating the lattice for each data point processed. The algorithm is also able to make a second pass of the data, which allows for better detection rates. Finally, it is also possible to constrain the size of the lattice to conserve memory, at a small cost to accuracy.

The LOADED algorithm has also been extended to handle distributed data sets [18]. The algorithm is essentially the same as the centralized version of LOADED presented above, but by default it performs two passes of the data set. In the first pass, each site constructs a local augmented lattice from its local portion of the data set. The lattices are then exchanged and combined to form a global augmented lattice which is used to detect outliers at each site during the second pass. There is also a variation that allows the global model to be computed incrementally, which allows for a single-pass approach. To avoid repeatedly exchanging lattices, only local outliers are exchanged between nodes. If all nodes agree that a given point is an outlier, then it is marked as a global outlier.

3.3 RELOADED

The RELOADED [19] algorithm is designed to address the memory usage problems of LOADED. As it stands, LOADED's space complexity is exponential in the number of categorical attributes, since it must maintain an augmented itemset lattice. RELOADED dispenses with the lattice and uses set of classifiers to model dependencies between the categorical attributes. Each classifier is trained to predict the value of a given categorical attribute based on the values of the remaining categorical and continuous attributes. Incorrect predictions of a categorical attribute for a data point increase the point's anomaly score. To further model dependencies between the categorical and continuous attributes, a covariance matrix is maintained for each unique value of each categorical attribute. Each covariance matrix is computed from those data points in which the given categorical attribute has the given value. A data point's anomaly score is also based on how well the data point's continuous attributes adhere to the relevant covariance matrices.

4. Evaluation

4.1 Setup

For evaluating the centralized algorithms, we use a machine with a 2.8 GHz Pentium IV processor and 1.5 GB of memory, running Mandrake Linux 10.1. Our implementations are in C++ and are compiled using gcc with O2 optimizations. We evaluate the distributed version of LOADED using an eight-node cluster, where each node has dual 1 GHz Pentium III

processors and 1 GB of memory, running Red Hat Linux 7.2. Our implementation uses MPI for message passing. Unless otherwise noted, we use the default values of the parameters for each algorithm (see [6, 18, 19, 2]). We use the following data sets.

4.1.1 KDDCup 1999 Intrusion Detection Data

The 1999 KDDCup data set [8] contains a set of records that represent connections to a military computer network where there have been multiple intrusions and attacks. This data set was obtained from the UCI KDD archive [3]. The training data set has 4,898,430 data instances with 32 continuous attributes and 9 categorical attributes. The testing data set is smaller and contains several new intrusions that were not present in the training data set. Since these data sets have an unrealistic number of attacks, we preprocess them such that intrusions constitute 2% of the data set, and the proportions of different attacks is maintained. Since packets tend to occur in bursts for some intrusions, intrusion instances are not randomly inserted into the data, but occur in bursts that are randomly distributed in the data set. The processed training data set contains 983,561 instances with 10,710 attack instances, while the processed testing data set contains 61,917 instances with 1,314 attack instances.

4.1.2 Adult Data

The Adult data set [3], contains 48,842 data instances with 6 continuous and 8 categorical attributes. Since the algorithms we test differ in their abilities to handle missing data, we removed all records containing missing data, leaving 32,561 records. The data was extracted from the US Census Bureau’s Income data set. Each record contains an individual’s demographic attributes together with a class label indicating whether person made more or less than 50,000 dollars per year.

4.1.3 Synthetic Data

Since there are very few publicly available large mixed-attribute data sets, we wrote a synthetic data set generator to produce data to compare performance with existing algorithms, and with varying data set characteristics. The generator can produce data sets with a user-supplied number of continuous attributes and categorical attributes. The data points are generated according to a user-supplied multi-modal distribution. The exact details can be found in [18]. To create actual data sets for our experiments, we first generate a set of normal points from one distribution, and then separately generate a much smaller set of outliers from another distribution. The two sets are then randomly mixed to produce the final data set. However, the synthetic data set is designed for benchmarking the memory and execution time scalability of the detection algorithms, and so it is not fair to use it to make detection quality comparisons. In our experiments, we consider a synthetic data set containing a 1% mixture of outliers.

4.2 Detection Quality

Our first set of experiments compares the detection rates of the various algorithms. In particular, we examine the performance of the two-pass versions of RELOADED and LOADED, and ORCA. Both the two-pass centralized and distributed versions of LOADED have the same accuracies, so we only present

the former here. Also, since ORCA is designed to find the top k outliers, we set k equal to the number of outliers in the data sets.

Detection rates for all the different algorithms are reported in Table 1 (Note that “n/a” indicates that the attack was not present in that particular data set). Since the intrusion packets tend to occur in bursts in our data set, we mark an intrusion as detected if at least one instance in a burst is flagged as an outlier. This is realistic, since a network administrator needs to be alerted only once that an intrusion is underway. Consequently, the detection (true positive) rates in the table are in terms of the number of intrusion bursts detected. We report false positive rates in terms of the number of normal packets marked as outliers (in the case of ORCA, this is the percentage of normal packets that are marked as top- k outliers). *Overall, the detection rates for LOADED are very good, better than those of RELOADED and much better than those of ORCA.* LOADED has a false positive rate of 0.35%, which is extremely good for anomaly detection schemes, especially considering its high detection rates for many of the intrusions. ORCA has a false positive rate of 0.43%, but this is not as significant considering its low detection rates. RELOADED has detection rates comparable to LOADED on many intrusions, and does very well on a handful of intrusions (e.g. IP sweep and smurf) on which both LOADED and ORCA do poorly. RELOADED has higher false positive rates of 1.5% for the testing data set and 3.6% for the training data set, which is to be expected since it builds a less intricate model in order to save on memory. Finally, we note that as the single-pass distributed version of LOADED scales to more nodes, its detection rate decreases slightly, as can be seen in the experimental results presented in [18]. This is can be attributed to data points that are flagged as local normals when they are in fact global outliers.

4.3 Memory Usage

Algorithm	KDDCup (Test)	KDDCup (Train)	Adult
RELOADED	623	852	291
LOADED	49,328	595,280	58,316
ORCA	599	n/a	390

Table 2. Peak heap usage in kilobytes.

We first compare the memory usage of ORCA, LOADED, and RELOADED when they are run on the KDDCup and Adult data sets. For RELOADED and LOADED we use single-pass approaches, as the amount of memory used does not vary with the number of passes. For LOADED, we use just 4 lattice levels, and we set ORCA to find the top 1,314 outliers in the KDDCup testing data set, and the top 30 outliers in the Adult data set. We also note that ORCA cannot finish processing the KDDCup training data set in a reasonable amount of time. We measure memory usage by looking at the peak heap usage measured in kilobytes. The results can be seen in Table 2. Both RELOADED and ORCA consume less than one megabyte of memory, while LOADED uses *two to three orders of magnitude more memory*, even when we constrain the lattice to 4 levels. RELOADED manages to keep its memory usage low by using a compact model of the data, while ORCA processes the data in blocks.

Unlike ORCA, LOADED and RELOADED have greater than

Attack	KDDCup Testing			KDDCup Training	
	RELOADED	LOADED	ORCA	RELOADED	LOADED
Apache2	100%	100%	0%	n/a	n/a
Back	n/a	n/a	n/a	0%	98%
Buffer Overflow	72%	90%	100%	0%	91%
FTP Write	n/a	n/a	n/a	0%	33%
Guess Password	50%	100%	0%	34%	100%
Imap	n/a	n/a	n/a	50%	100%
IP Sweep	100%	28%	0%	90%	37%
Land	n/a	n/a	n/a	100%	100%
Load Module	n/a	n/a	n/a	0%	100%
Multihop	63%	70%	75%	0%	94%
Named	67%	100%	40%	n/a	n/a
Neptune	n/a	n/a	n/a	100%	98%
Nmap	n/a	n/a	n/a	64%	91%
Perl	n/a	n/a	n/a	0%	100%
Phf	80%	20%	100%	0%	0%
Pod	96%	100%	18%	81%	54%
Port Sweep	100%	100%	3%	93%	100%
Root Kit	n/a	n/a	n/a	0%	33%
Saint	100%	100%	1%	n/a	n/a
Satan	n/a	n/a	n/a	80%	72%
Sendmail	17%	50%	50%	n/a	n/a
Smurf	98%	21%	0%	78%	22%
Snmptest	0%	52%	0%	n/a	n/a
Spy	n/a	n/a	n/a	0%	100%
Teardrop	n/a	n/a	n/a	40%	30%
Udpstorm	0%	0%	0%	n/a	n/a
Warez Client	n/a	n/a	n/a	4%	43%
Warez Master	n/a	n/a	n/a	0%	25%
Xlock	50%	50%	66%	n/a	n/a
Xsnoop	100%	100%	100%	n/a	n/a

Table 1. Detection rates for the KDDCup data sets.

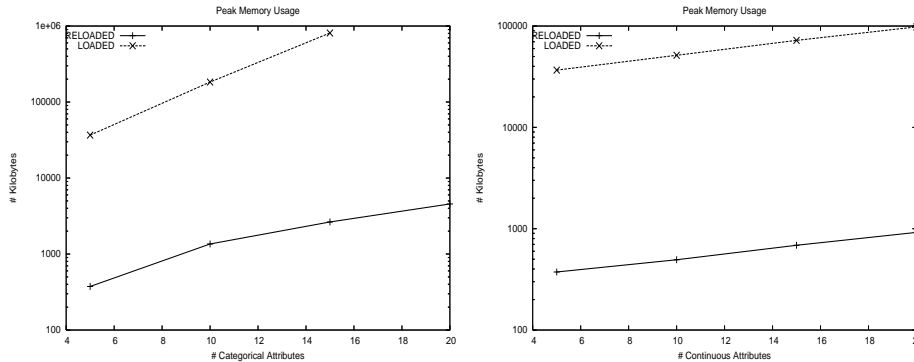


Figure 1. Log of peak memory usage with increasing numbers of (a) categorical and (b) continuous attributes.

linear space complexity with respect to the number of categorical and continuous attributes, and so we empirically test how they scale in terms of peak memory usage as the number and types of attributes vary. Again, in all cases, we set LOADED to use only 4 lattice levels. In Figure 1(a) we plot the log of peak memory usage versus increasing numbers of categorical attributes, while setting the number of continuous attributes equal to 5. As is evident from the graph, the memory requirements of LOADED are very large and grow exponentially as the number of categorical attributes increase, while those of RELOADED grow much more slowly. Note that we cannot run LOADED on data sets with more than 15 categorical attributes, as our machines do not have sufficient memory. In Figure 1(b) we set the number of categorical attributes equal to 5 and then vary the number of continuous attributes. The peak memory requirements of both RELOADED and LOADED increase at about the same rate, which is expected since they both use space that is quadratic in the number of continuous attributes. Note that even for 5 categorical attributes, LOADED requires significantly more memory to maintain the itemset lattice.

4.4 Execution Time

In our next set of experiments we compare the execution times of the ORCA, LOADED and RELOADED algorithms. Again, for LOADED we only use 4 lattice levels. Also, we use the single-pass versions of both RELOADED and LOADED. In our first experiment, we measure the execution times on the KDDCup testing data set. ORCA takes 303 seconds to complete, compared with 109 seconds for LOADED, and 47 seconds for RELOADED. In our next experiment, we examine how execution time scales with the number of data points processed. We use synthetic data with 10 categorical and 5 continuous attributes. The results can be seen in Figure 2(a). Note that we use a log scale on both axes. For small data sets, ORCA outperforms both LOADED and RELOADED, but since it does not scale linearly, this advantage is lost for larger data sets. As we expect, both RELOADED’s and LOADED’s execution times scale linearly with the number of points, though LOADED does not scale as well as RELOADED.

While ORCA’s time complexity is linear with respect to the number of categorical and continuous attributes, LOADED’s and RELOADED’s complexity is not, and so in our next two experiments we compare how the execution of times of both LOADED and RELOADED scale for data sets with varying numbers of categorical and continuous attributes. In our first experiment, we set the number of continuous attributes equal to 5 and vary the number of categorical attributes between 1 and 15. The results can be seen in Figure 2(b). Note that we only use a log scale on the execution time axis. Though we limit LOADED to using only 4 lattice levels, its execution time still increases exponentially with the number of categorical attributes, while RELOADED increases quadratically. In our second experiment we examine the scalability of both algorithms with respect to the number of continuous attributes. In this experiment we set the number of categorical attributes equal to 5 and vary the number of continuous attributes between 1 and 25. The results can be seen in Figure 2(c). For smaller numbers of continuous attributes (less than 15), LOADED is more efficient than

RELOADED, but since it appears that RELOADED scales near linearly while LOADED scales quadratically with respect to the number of continuous attributes, RELOADED is more efficient if there are larger numbers of continuous attributes.

As we noted above, LOADED does not scale well for large numbers of categorical attributes, since it maintains the entire itemset lattice in memory. However, LOADED uses an approximation scheme in which it only uses a partial lattice. The primary benefit of the approximation scheme is that LOADED achieves far better execution times if fewer lattices levels are maintained, as can be seen in Figure 3(a). Empirically, it appears from Figure 3(a) that execution time grows quadratically with the number of lattice levels. The detection rates decrease as the number of lattice levels decrease, as can be seen in Figure 3(b). The affect of the number of lattice levels on the false positive rates can be seen in Figure 3(c). Note that the false positive rate axis uses a log scale. The false positive rates are not affected significantly with changing lattice levels. On the other hand, detection rates seem to increase as the number of lattice levels increase to 3, after which they stabilize.

4.4.1 Distributed LOADED

In our last set of experiments, we explore the benefits gained from using the distributed versions of LOADED. We first explore the speedup obtained when running the two-pass distributed LOADED algorithm on two, four, and eight sites. The KDDCup 1999 training and synthetic data sets are split evenly between the sites for this experiment. Figure 4(a) shows the speedup obtained on the two data sets. As there is only one round of communication, the overall message passing overhead is minimal. Most of the time is spent in the two phases: 1) building the local model in the first pass; and 2) finding outliers in the second pass. Consequently, each node works independently, and we see up to 7.7-fold speedup on 8 sites. The slight reduction in speedup with increasing number of sites is due to increasing communication overhead associated the local model exchange. Next, we vary the link bandwidth between the sites in our controlled environment in order to simulate a network spanning larger distances. As shown in Figure 4(b), for a wide area setting consisting of eight nodes, efficiency varies from a low of 69% to a high of 96%, for link bandwidths equal to 1 MB/s and 100 MB/s respectively (note the log scale of the bandwidth axis). Even when link bandwidth is equal to 10 MB/s, LOADED achieves an efficiency of 95%, suggestive of good scalability for outlier detection within an organization.

Finally, we explore the speedup obtained when running the LOADED one-pass outlier detection algorithm on two, three, and four sites. The KDDCup 1999 and synthetic data sets are evenly split between the nodes for this experiment. Figure 5(a) shows speedup obtained on the two data sets. Since there are relatively few outliers in the data set, and we have a low false positive rate, there is very little communication overhead, resulting in minimal synchronization between the sites. Therefore each site is able to work independently. As the number of nodes increases, the communication overhead also increases, as more nodes are involved in the local outlier exchange. As a result we see a slight reduction from the ideal speedup, and efficiency falls to 95% on the two data set when using four

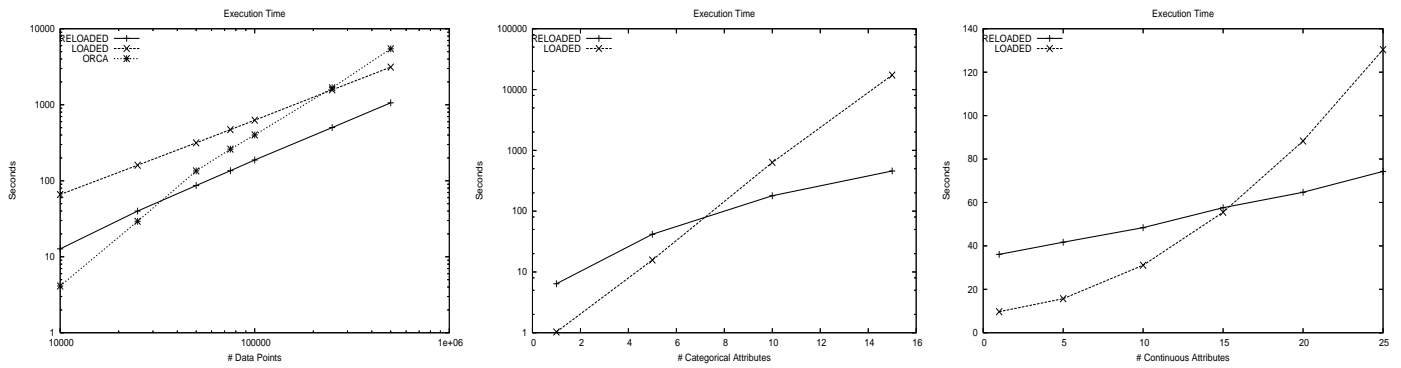


Figure 2. Plots of execution time versus (a) data set size; (b) increasing categorical attributes; (c) increasing continuous attributes.

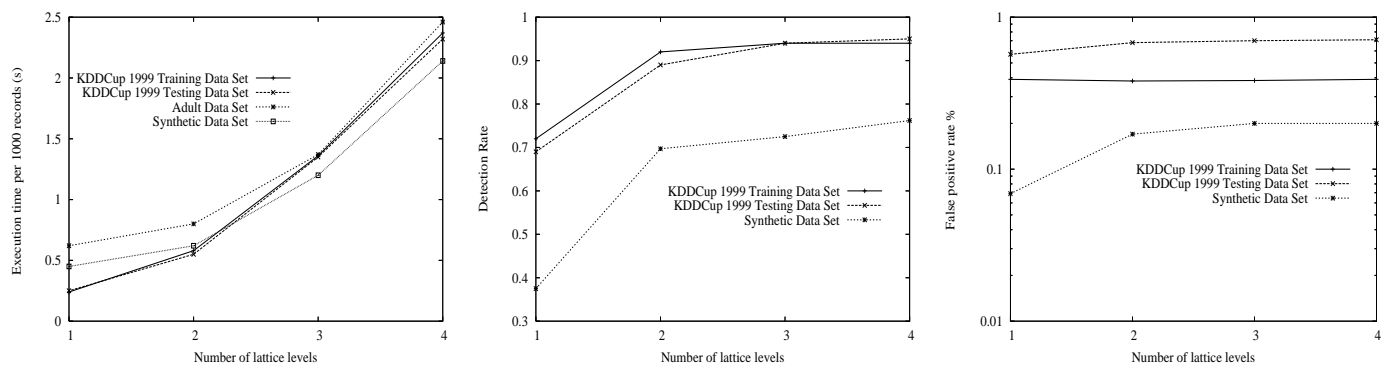


Figure 3. Effect of the number of lattice levels on (a) Execution time, (b) Detection rates, and (c) False positive rates.

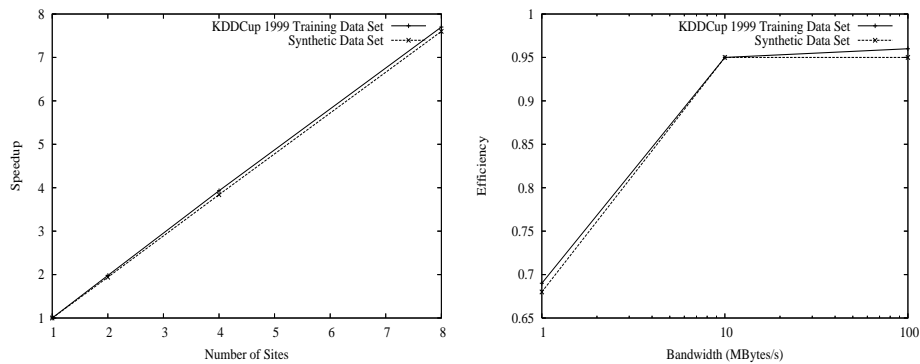


Figure 4. (a) Speedup for a varying number of sites, and (b) Expected efficiency in a wide area network.

sites.

As nodes primarily communicate by exchanging outliers, which are small messages, link latency will be the primary performance bottleneck in a wide area setting. We vary the average link latency between the nodes in our controlled environment to simulate a wide area network spanning larger distances. As shown in Figure 5(b), efficiency falls to 90% for the KDDCup 1999 data set when using 4 sites, with an average link latency of 25ms. This is representative of networks spanning across several states and excellent scalability.

5. Conclusions

It is clear from our evaluation that ORCA, LOADED, and RELOADED have different strengths and weaknesses. ORCA has relatively poor detection rates for mixed-attribute data sets such as the KDDCup data set, due to its use of an ad-hoc combination of the Euclidean and Hamming distance metrics to account for both the continuous and categorical attributes. Also, ORCA performs randomization and multiple passes of data sets, which make it unsuitable for incremental outlier detection. However, it shows excellent memory usage properties and good scalability with respect to the number of data points. LOADED, on the other hand, outperformed all the other algorithms with respect to detection rates, and is the only algorithm currently able to process distributed data sets. However, it scales very poorly with respect to memory usage, even if the number of lattice levels are restricted. RELOADED, like ORCA, scales very well with respect to memory usage, and achieves better detection rates than ORCA, at the cost of an increased false positive rate. Both LOADED and RELOADED are capable of detecting outliers in a single pass of the data, unlike ORCA. Though its detection rate is lower than LOADED's, RELOADED's small memory footprint and small execution times make it a good candidate for embedded outlier detection systems, such as might be found in network interface card-based intrusion detection [17], or sensor networks.

6. Acknowledgments

This work is supported in part by NSF grants (CAREER-IIS-0347662) and (NGS-CNS-0406386).

7. References

- [1] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley, 1994.
- [2] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proc. of 9th annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.
- [3] C. Blake and C. Merz. UCI machine learning repository, 1998.
- [4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jorg Sander. LOF: Identifying density-based local outliers. In *ACM SIGMOD Intl. Conf. Management of Data*, 2000.
- [5] D. Gamberger, N. Lavrač, and C. Grošelj. Experiments with noise filtering in a medical domain. In *ICML*, 1999.
- [6] Amol Ghoting, Matthew Eric Otey, and Srinivasan Parthasarathy. Loaded: Link-based outlier and anomaly detection in evolving data sets. In *Proceedings of the IEEE International Conference on Data Mining*, 2004.
- [7] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.
- [8] S. Hettich and S. Bay. KDDCUP 1999 dataset, UCI KDD archive, 1999.
- [9] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [10] Theodore Johnson, Ivy Kwok, and Raymond T. Ng. Fast computation of 2d depth contours. In *ACM SIGKDD*, pages 224–228, 1998.
- [11] E. Knorr and *et al.* Distance-based outliers: Algorithms and applications. *VLDB Journal*, 2000.
- [12] E. Knorr and R. Ng. A unified notion of outliers: Properties and computation. In *ACM SIGKDD*, 1997.
- [13] E. Knorr and R. Ng. Finding intentional knowledge of distance-based outliers. In *VLDB*, 1999.
- [14] E. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. Int'l Conf. on VLDB*, 1998.
- [15] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of outlier detection schemes for network intrusion detection. In *SIAM Data Mining*, 2003.
- [16] Matthew V. Mahoney and Philip K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *ACM SIGKDD*, 2002.
- [17] M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula, and D. Panda. Towards nic-based intrusion detection. In *Proceedings of 9th annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.
- [18] Matthew Eric Otey, Amol Ghoting, and Srinivasan Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. Technical Report OSU-CISRC-6/05-TR42, Department of Computer Science and Engineering, The Ohio State University, 2005.
- [19] Matthew Eric Otey, Srinivasan Parthasarathy, and Amol Ghoting. Fast lightweight outlier detection in mixed-attribute data. Technical Report OSU-CISRC-6/05-TR43, Department of Computer Science and Engineering, The Ohio State University, 2005.
- [20] Spiros Papadimitriou, Hiroyuki Kitawaga, Phillip B. Gibbons, and Christos Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *ICDE*, 2003.

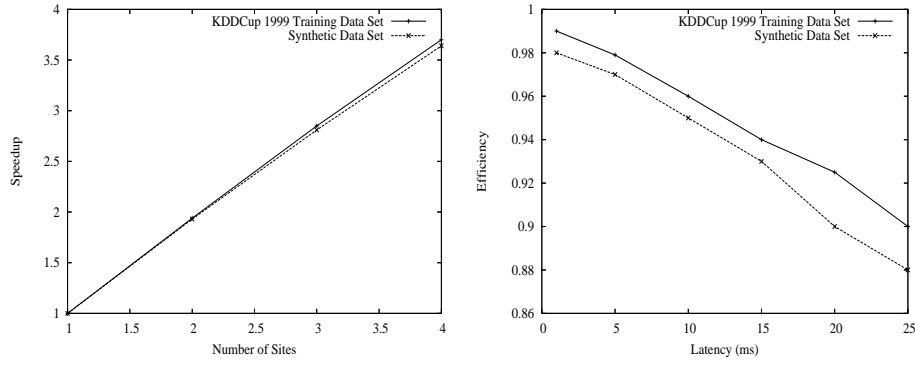


Figure 5. (a) Speedup for the single-pass approach, and (b) Expected efficiency in a wide area network.

- [21] Karlton Sequeira and Mohammed Zaki. Admit: Anomaly-based data mining for intrusions. In *ACM SIGKDD 02*, pages 386–395, 2002.