BŞEÜ Fen Bilimleri Dergisi
7(1), 129-140, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.649539

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Araştırma Makalesi - Research Article

Standart Dışı Değerlerin Tespitinde DBSCAN Algoritması ile Seri ve Paralel Programlama Performansının Karşılaştırılması

Hüseyin Yaşar[1*], Mehmet Albayrak[2]

Geliş / Received: 21/11/2019 ***Revize / Revised: 08/04/2020*** ***Kabul / Accepted: 26/04/2020***

ÖZ

Bilgisayarların hayatımıza girmesiyle beraber dijital verilerin boyutları giderek artmaktadır. Dijital dünyada üretilen bu verilerin içinde benzerlerinden farklı davranış sergileyen standart dışı değerler (aykırı değerler) bulunabilmektedir. Bu değerlerin özellikle büyük veri setleri içinde tespiti; güvenlik, sigortacılık, finans, tıp ve genetik gibi alanlarda büyük önem taşımaktadır. Büyük veri setlerinde standart dışı değerlerin tespitinde veri madenciliği yöntemlerinden kümeleme teknikleri sıklıkla kullanılmaktadır. Gürültülü ve aykırı değerlere karşı hassas olan kümeleme algoritmalarından, yoğunluk tabanlı DBSCAN (Density-based spatial clustering of applications with noise) algoritması standart dışı değerlerin tespitinde kullanılan en önemli yöntemlerdendir. Bu çalışmada standart dışı değerlerin tespiti için C# programlama dilinde DBSCAN algoritması kullanılarak bir uygulama geliştirilmiştir. Geliştirilen uygulamada; veri sayıları birbirinden farklı 2 adet veri seti ele alınmış ve analizleri yapılmıştır. Veri setleri analizinin en kısa süreye indirilebilmesi için seri ve paralel programlama teknikleri ayrı ayrı kullanılmıştır. Büyük veri setlerinin analiz süresini kısaltmak amacı ile. Net 4.0 ile gelen TPL (Task Parallel Library) içinde yer alan paralel sınıf üyelerinden yararlanılmıştır. Veri setlerinde yapılan analizlerde DBSCAN algoritmasının standart dışı değerlerin tespiti açısından seçilen diğer algoritmalara göre daha yüksek doğruluk oranında sonuç verdiği ve kullanılabilir olduğu görülmüştür. Hesaplama performansı açısından ele alındığında ise paralel programlamanın veri sayısı arttıkça daha verimli olabileceği sonucuna varılmıştır.

Anahtar Kelimeler- Standart dışı veri, Kümeleme, DBSCAN, Paralel programlama

[1]*Sorumlu yazar iletişim: huyasar@hotmail.com (https://orcid.org/0000-0003-2715-9313)
Department of Electronics and Computer Education, Süleyman Demirel University, Isparta
[2]İletişim: mehmetalbayrak@isparta.edu.tr (https://orcid.org/0000-0002-7089-122X)
Department of Computer Tech., Isparta University of Applied Sciences, Isparta

BŞEÜ Fen Bilimleri Dergisi
7(1), 129-140, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.649539

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Comparison of Serial and Parallel Programming Performance in Outlier Detection with DBSCAN Algorithm

ABSTRACT

With the introduction of computers into our lives, digital data sizes are increasing gradually. Non-standard values (outliers) which behave differently from the others can be found in these data produced in the digital world. Detection of these values, especially in big data sets; has great importance in fields such as security, insurance, finance, medicine and genetics. Clustering methods of data mining techniques are frequently used in outlier detection in big data sets. Density based DBSCAN (Density-based spatial clustering of applications with noise) algorithm from clustering algorithms which are sensitive to noisy and outlier values is one of the most important methods in outlier detection. In this study, an application was developed using DBSCAN algorithm in C# programming language for the detection of outliers. In the developed application, 2 data sets with different data numbers were examined and analyzed. For the shortest possible data analysis time, serial and parallel programming techniques were used separately. In order to shorten the analysis time of big data sets, parallel class members in TPL (Task Parallel Library) provided with .Net 4.0 were used. In series of analysis of data sets, it was seen that DBSCAN algorithm produces more accurate results and is more practicable than other selected algorithms in terms of outlier detection. When considered in terms of computing performance, parallel programming has become more efficient as the number of data increases.

Keywords- Outlier, Clustering, DBSCAN, Parallel Programming

I. INTRODUCTION

Every day in the world, 2.5 quintillion bytes of digital data are created. 90% of the data in the world today has been created in the last two years alone. These data are mainly; information produced by sensors used to gather climate information, posts on social media sites, digital pictures and videos, purchase transaction records and cell phone GPS signals [1]. More information is produced in many areas such as these. It takes a long time to find the outliers in this data because of the size of the data sets. Outliers are abnormal values which do not fit in overall data set and deviate from the overall distribution of the data. Inconsistencies in data sets may be erroneous or they may reflect the truth. The first thing to take into account at this point is to decide whether the data reflects the reality [2].

Outlier detection has important applications in the field of data mining, such as fraud detection, customer behavior analysis, and intrusion detection. Outlier detection is the process of detecting the data objects which are grossly different from or inconsistent with the remaining set of data. Outliers are traditionally considered as single points; however, there is a key observation that many abnormal events have both temporal and spatial locality, which might form small clusters that also need to be deemed as outliers. In other words, not only a single point but also a small cluster can probably be an outlier [3].

Computer hardware has difficulty to respond the needs of software. While the memory or bit depth in hardware components can be increased, the processor speed reaches almost physical limits. Hardware manufacturers increase the number of cores/processors used in computers instead of the processor speed which reaches physical limits. Computer software is required to be programmed parallelly in order that they benefit more efficiently from multiprocessor computers [4].

Performance of computer systems is directly proportional to processor clock frequency and memory capacity. Performance can be improved by using processors that can operate at higher frequencies since it will increase the number of commands processed at the unit of time. However, increasing processor frequencies is limited due to physical reasons. On the other side, performance can also be increased by extending the memory capacity using additional memory. However, this is also not economic after certain values. It is seen that the performance of single-processor computer systems is limited due to both physical and economic reasons. As a result, parallelism in various dimensions to enhance the performance is inevitable for economic solutions inside or outside the processor [5]. The emergence of multicore chips makes computer programming model facing huge pressure due to the shift from traditional serial programming mode to new parallel programming mode. The performance of serial applications can only be improved with parallelism, and programmers are on the way to parallel programming [6]. If current technological developments in the world does not accelerate certain studies at the desired level, it will be waited that the technology reaches the desired level which may take a long time in certain cases; there may be problems which require an increase at an impossible level or it is necessary to find solutions using the existing technology. In order to accelerate the solution of a problem with existing resources, to segment the problem and to solve each segment in different computers or processors will be the most optimal solution. In other words, the method that should be preferred is definitely parallel programming [7]. Parallel computation has become increasingly important in recent years, moving from the realm of scientific supercomputing into corporate database servers and into the cellphones and personal computers of users [8].

The main purpose of parallel programming of an algorithm is to benefit from the advantages of multi-core processors in the most effective and correct way. One of the most effective methods to shorten the computing time in the applications that uses very large data and requires a long computing time, is to distribute processes parallelly among multi cores. Thus, the calculation time is shortened and therefore the performance increase is aimed [9]. Multi-core processors and shared memory multiprocessor systems can speed up applications when using multiple threads and/or multiple processes. At this level, parallel programs can be written using multi-threaded programming using explicit threading supported by the operating system [10].

The current state of the information sector and the corresponding rapid increase of needs day by day brings a lot of problems with it. Information Technologies sector also occupies the first place by getting the maximum share of the cake. Now, in addition to the stable and faultless operation of a developed system, to achieve the best result in the shortest time possible is early in the list of demands. The subject of speed has vital importance for production industry, engineering calculations, military simulation projects and weather forecasts. [11].

Additionally, Multicore development support within the C# language is extremely powerful and versatile making it one of the best languages available for the rapid development and prototyping of parallel Symmetric Multiprocessing (SPM) applications. Although C# is primarily designed for the Windows operating system, parallel applications can be deployed on many other non-Windows systems via the World Wide Web or using an open source compiler such as Mono. Considering its reasonable execution performance and the very rapid development time lines afforded by the robust Visual Studio IDE and multicore development classes, C# is preferred for multicore development unless Asymmetric Multiprocessing or non-uniform memory access were key project requirements [12]. TPL (Task Parallel Library) auto scales the concurrency and LINQ queries to a multicore level. It handles the partitioning of the work and uses Thread Pool where required. It is easy to use and reduces the complexity of working with threads directly [13].

In this study, subjects of non-standard data analysis, DBSCAN algorithm and parallel programming with Microsoft. NET was emphasized. DBSCAN algorithm which is a clustering method has been studied in the study. Microsoft .Net parallel programming methods are used to improve the speed and performance of the developed program. Speed and performance of serial and parallel programming have been compared.

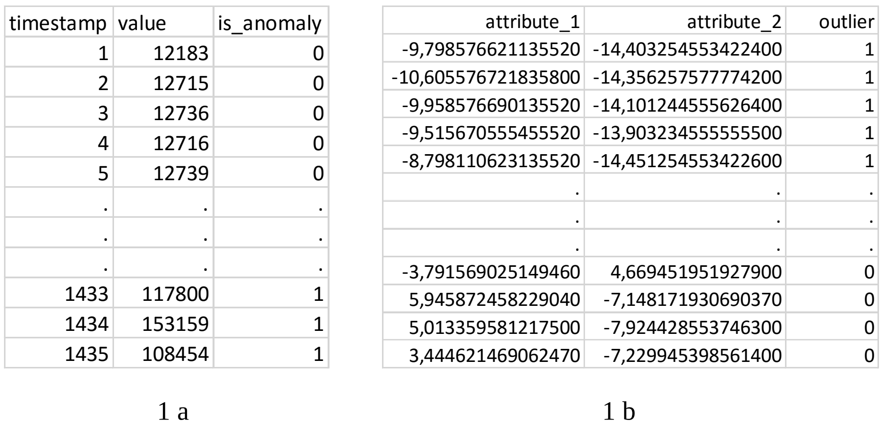
II. OUTLIER ANALYSIS

In order to be able to evaluate the data sets scientifically, certain preliminary operations are required. In addition to problems such as incomplete or inconsistent data, presence of excessive or extreme values in the data is a problem that must be taken into consideration, especially for statistical analysis [14]. Extreme values which do not fit into the data set when compared to the other values in the data set are called as outliers (non-standard data) [15]. Several methods have been developed to determine whether the values in the data sets are outliers. These are classic determinants, robust methods and data mining methods. Classic determinants experience difficulties in the presence of a large number of outliers, while they do not have problems in the presence of a single outlier. Multiple classic determinants are needed in case of multiple outliers, and these require complex operations [16]. These methods are not preferred due to computational complexity in big data sets. More distance and density-based methods are used in big data sets.

A. DBSCAN Algorithm

DBSCAN, a density-based clustering algorithm, accounts for the densities of objects while generating clusters. Clusters are defined by high-density data objects; while clusters with low density objects indicate outliers or noisy points. DBSCAN algorithm can be viewed as a prototypal method in which the extension of a cluster relies on some conditions that can be viewed as a predicate. DBSCAN is particularly useful for large databases and data sets containing noisy objects [17,18]. DBSCAN is an iterative algorithm which iterates over the objects in the dataset, analyzing their neighborhood. If there are more than minPts objects whose distance from the considered object is less than eps, then the object and its neighborhood originate a new cluster. DBSCAN is effective at finding clusters with arbitrary shape, and it is capable of identifying outliers as a low-density area in the data space. The effectiveness of the algorithm is strongly affected by the setting of parameters eps and minPts [19]. DBSCAN clustering algorithm needs two input parameters to define the notion of density: Eps and MinPts. The input parameter Eps is a radius value and it is based on a distance metric such as Manhattan, Euclidean etc. The second input parameter MinPts specifies the minimum number of points that should occur within Eps radius [20]. The terms of core point, eps, minPts, directly density reachable point, density reachable point and density connected point are main concepts for DBSCAN. The algorithm requires two parameters: eps and minPts. It controls all objects starting from any object in the database. If the controlled object was previously included in a cluster, it switches to the other object without performing the operation. If the object was not clustered previously, it finds the neighbors of the object within the eps neighborhood by making a region query. If the number of neighbors is more than minPts, it marks this object and its neighbors as a new cluster. Then, it finds new neighbors by making region query for each neighbor which has not been clustered before. If the number of neighbors of the region queried points is more than minPts, it is included into the cluster [21]. Objects outside of the cluster are marked as outliers.

In this study, real_2 data set is provided from Yahoo Webscope library. The values in the data set are derived from the values obtained from Yahoo services. The data set is named S5 - A Labeled Anomaly Detection Dataset, version 1.0 (16M) and is 1439 lines long. [22]. The other data set used in the study is provided German Artificial Intelligence and Research Center. The dataset is named dfki-artificial-3000-unsupervised-ad and is 3000 lines long [23]. Data sets specifically contain non-standard values (Fig. 1). The characteristics of the data sets are shown in Table 1.



timestamp	value	is_anomaly
1	12183	0
2	12715	0
3	12736	0
4	12716	0
5	12739	0
.	.	.
.	.	.
.	.	.
1433	117800	1
1434	153159	1
1435	108454	1

1 a

attribute_1	attribute_2	outlier
-9,798576621135520	-14,403254553422400	1
-10,605576721835800	-14,356257577774200	1
-9,958576690135520	-14,101244555626400	1
-9,515670555455520	-13,903234555555500	1
-8,798110623135520	-14,451254553422600	1
.	.	.
.	.	.
.	.	.
-3,791569025149460	4,669451951927900	0
5,945872458229040	-7,148171930690370	0
5,013359581217500	-7,924428553746300	0
3,444621469062470	-7,229945398561400	0

1 b

Figure 1. Sample dataset parts; **1a:** real_2 data set **1b:** dfkiartificial300 data set

Table 1. The characteristics of the data sets

Data Set	Lines Length	Number of Non-standard values	Place Provided
real_2	1439	16	Yahoo Webscope
dfkiartificial300	3000	37	German Research Center for Artificial Intelligence

The application is run on MS Windows 10 operating system, Intel (R) Core (TM) i7 - 3630QM CPU 2.40 GHZ processor and 16GB memory hardware. The serial and parallel codes of the program were written in C# programming language. The application was developed using the Visual Studio 2015 editor. The application was developed as a Windows Form Application in C# language and the analysis results are shown visually in figures. Operation of the application and objects on the form display; table selection, number of data, number of nonstandard values, distances of nonstandard values, operation with serial codes, operation with parallel codes, optimal eps and minPts values, graphic area operating with ZedGraph plugin, size of non-standard values and working time.

For measurement of run time (performance); Freeman [24] mentioned that the Stopwatch class is a right method for performance measurement.

In this study, the execution run times of serial and parallel methods were measured using this class and shown on the application in millisecond.

B. Finding Optimal Eps and minPts Values for Data Sets

DBSCAN algorithm requires the eps and minPts values to be entered by the user. The selection of the eps and minPts values is very important for the DBSCAN algorithm to give the best result. Elbatta and Ashour [25], specified that in order to determine eps and minPts values, it is required to look for the kth nearest neighbor distances of the points and named the distance of the points as kNN-dist. kNN-dist is calculated for all points and these k distances are transformed into an increasing graph. In order to calculate the appropriate eps and minPts values for each data set, the R project named statistical modeling and development tool was used to plot the nearest neighbor k graph.

For the Real_2 data set, the sharp change in the graph is taken as the value of eps since it is approximately 500, and the value of minPts is taken equal to the value of k (Fig. 2). For the dfkiartificial3000 data set, the sharp change in the graph is taken as the value of eps since it is approximately 0.9, and the value of minPts is taken equal to the value of k. (Fig. 3). In addition, the dashed line in Figures 1 and 2 is used to indicate approximately at which points the sharp change in the graphics.

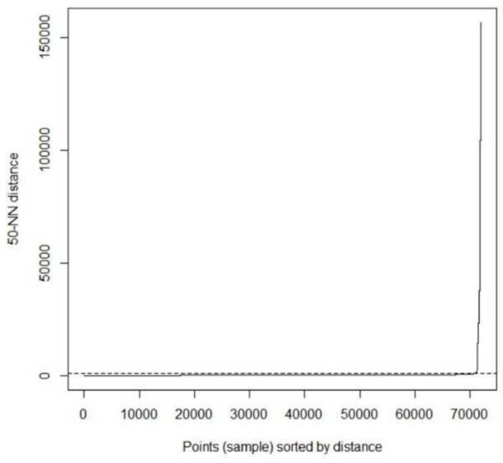


Figure 2. k-Dist graph for data sets

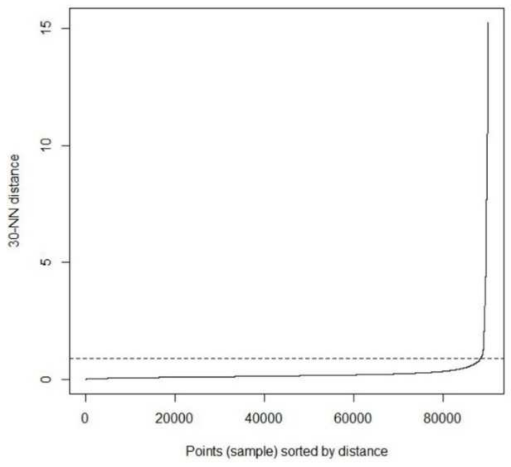


Figure 3. k-Dist graph for data sets

C. DBSCAN Application in Serial and Parallel Programming and the Results

The implemented application contains 3 methods (GetClusters, ExpandCluster, GetRegion) (Table 2). The code map generated by the Visual Studio editor of the serial and parallel application is shown (Fig. 4 and Fig. 5). Code maps help you see how the code fits together without reading through files and lines of code [26].

Table 2. Methods used in the application and their tasks.

Method Name	Task
GetClusters	It finds arrays containing clusters.
ExpandCluster	It expands clusters
GetRegion	Finds core points.

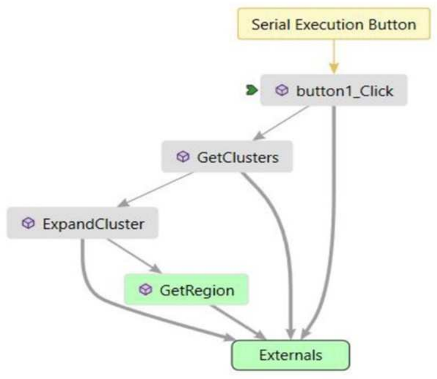


Figure 4. Code map of serial execution

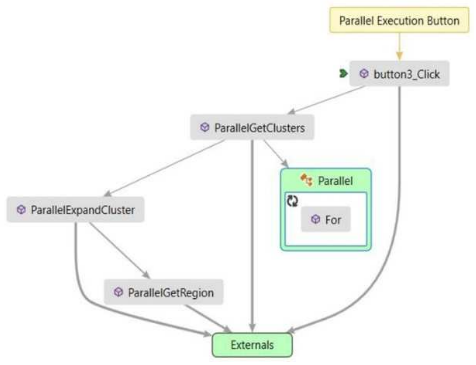
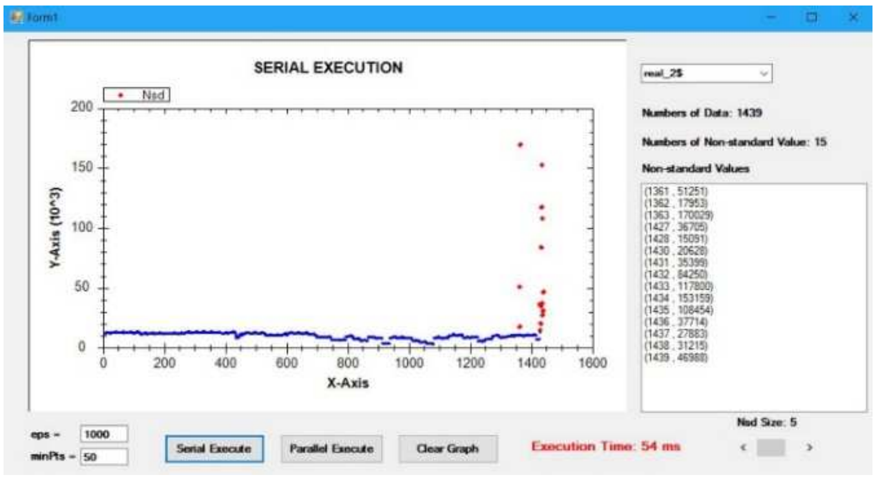


Figure 5. Code map of parallel execution

With the Microsoft .Net Framework 4.0 version, applications can be run simultaneously and more efficiently with multiple-channel methods on multiple cores. Instead of using "for" loops used in serial applications, we tried to improve performance by using the "parallel for" loop, which is a parallel class member in the task parallel library (TPL) that provided with .Net framework 4.0. “Parallel for” loop was used in “ParallelGetCluster” method. The Parallel for loop concurrently processes the points in the data set, through the automatically generated "threads". The analysis images for the Real_2 and DfkiArificial3000 data sets are shown in Fig. 6 and Fig. 7. In addition, the tasks of the objects on the application are given in Table 3.

Table 3. Tasks of objects in the application interface

Object	Task
Number of Data	The number of lines in the selected data set is shown.
Number of Non-Standard Value	The number of data detected by the application as non-standard data is shown.
Non-Standard Values	Identified non-standard values
Serial Execute	The application is provided to work with serial codes
Parallel Execute	The application is provided to work with parallel codes.
eps and minPts	Optimum eps and minPts values are shown according to the selected data set
Graphics Area	The analyzed data are shown in the graphics area. Normal values are shown in blue and non-standard values are shown in red.



real_2$

Numbers of Data: 1439

Numbers of Non-standard Value: 15

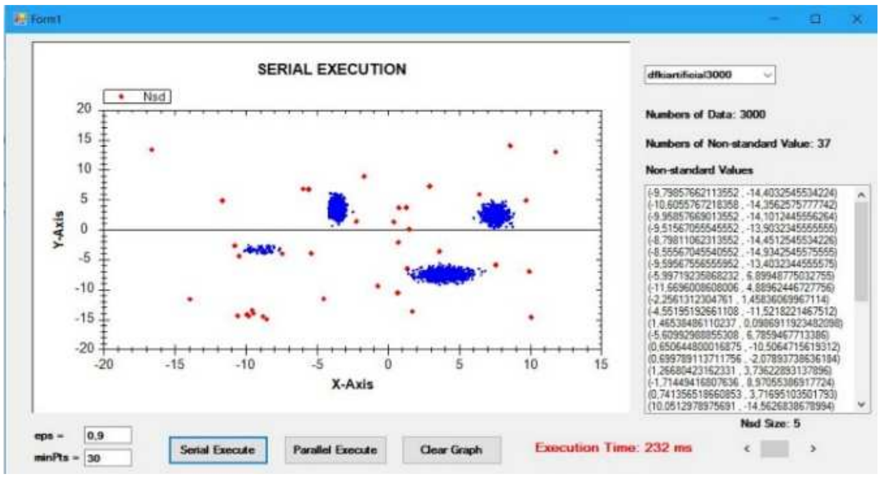
Non-standard Values

```
(1361 , 51251)
(1362 , 17953)
(1363 , 170029)
(1427 , 36705)
(1428 , 15091)
(1430 , 20628)
(1431 , 35399)
(1432 , 84250)
(1433 , 117800)
(1434 , 153159)
(1435 , 108454)
(1436 , 37714)
(1437 , 27883)
(1438 , 31215)
(1439 , 46988)
```

eps = 1000
minPts = 50

Execution Time: 54 ms

Figure 6. real_2 Data set analysis



dfkiartificial3000

Numbers of Data: 3000

Numbers of Non-standard Value: 37

Non-standard Values

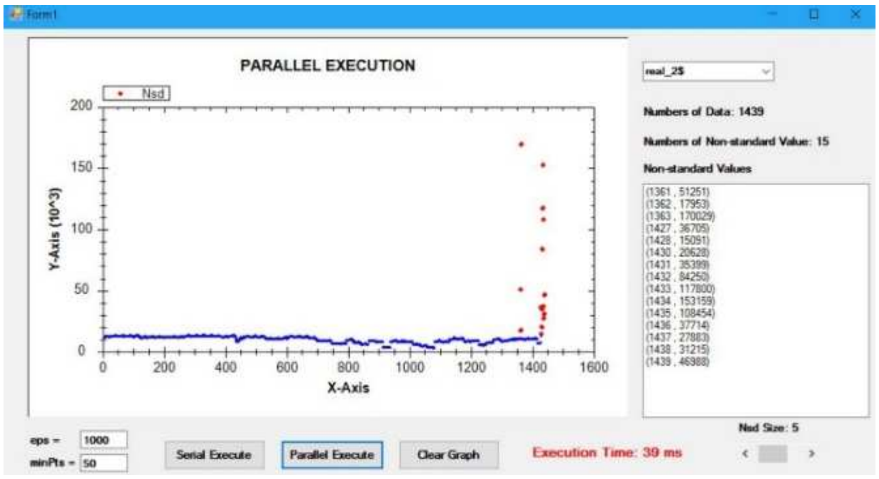
```
(-9,79857662113552 , -14,4032545534224)
(-10,6055767218358 , -14,3562575777742)
(-9,95857669013552 , -14,1012445556264)
(-9,51567055545552 , -13,9032345555555)
(-8,79811062313552 , -14,4512545534226)
(-8,55567045540552 , -14,9342545575555)
(-9,59567556555952 , -13,4032344555575)
(-5,99719235868232 , 6,89948775032755)
(-11,5696008608006 , 4,88962446727756)
(-2,2561312304761 , 1,45836069967114)
(-4,55195192661108 , -11,5218221467512)
(1,46538486110237 , 0,0986911923482098)
(-5,60992988855308 , 6,7859467713386)
(0,650644800016875 , -10,5064715619312)
(0,699789113711756 , -2,07893738636184)
(1,26680423162331 , 3,73622893137896)
(-1,71449416807636 , 0,97055386917724)
(0,741356519660853 , 3,71695103501793)
(10,0512978975691 , -14,5626838678994)
```

eps = 0,9
minPts = 30

Execution Time: 232 ms

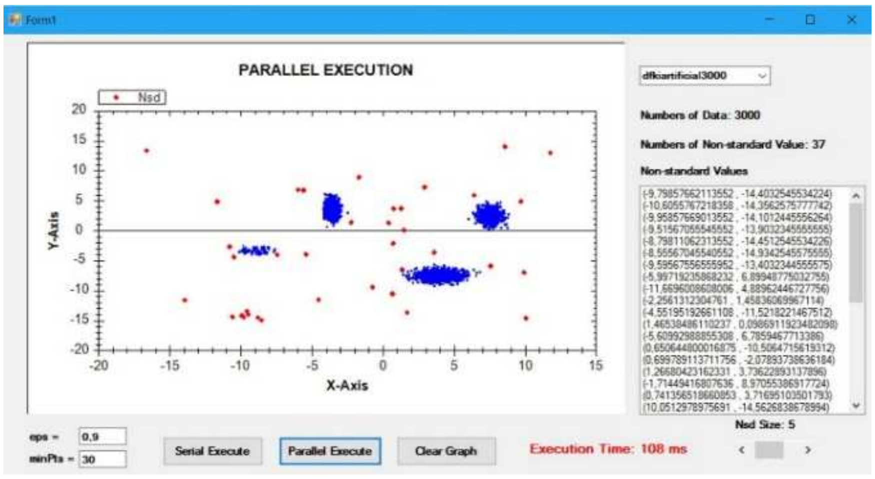
Figure 7. dfkiartificial300 Data set analysis

Parallel code analysis images for the same data sets are shown in Fig. 8 and Fig. 9. Table 4 shows the results of serial and parallel application. As it can be seen from the figures, as the length of the data set increases the parallel execution time is shortened and computation performance is improved.



```
real_2$
Numbers of Data: 1439
Numbers of Non-standard Value: 15
Non-standard Values
(1361 , 51251)
(1362 , 17953)
(1363 , 170029)
(1427 , 36705)
(1428 , 15091)
(1430 , 20628)
(1431 , 35399)
(1432 , 84250)
(1433 , 117800)
(1434 , 153159)
(1435 , 108454)
(1436 , 37714)
(1437 , 27883)
(1438 , 31215)
(1439 , 46988)
Nsd Size: 5
eps = 1000
minPts = 50
Execution Time: 39 ms
```

Figure 8. real_2 Data set analysis



```
dfkiartificial3000
Numbers of Data: 3000
Numbers of Non-standard Value: 37
Non-standard Values
(-9,79857662113552 , -14,4032545534224)
(-10,6055767218358 , -14,3562575777742)
(-9,95857669013552 , -14,1012445556264)
(-9,51567055545552 , -13,9032345555555)
(-8,79811062313552 , -14,4512545534226)
(-8,55567045540552 , -14,9342545575555)
(-9,59567556555952 , -13,4032344555575)
(-5,99719235868232 , 6,89948775032755)
(-11,6696008608006 , 4,88962446727756)
(-2,2561312304761 , 1,45836069967114)
(-4,55195192661108 , -11,5218221467512)
(1,46538486110237 , 0,0986911923482098)
(-5,60992988855308 , 6,7859467713386)
(0,650644800016875 , -10,5064715619312)
(0,699789113711756 , -2,07893738636184)
(1,26680423162331 , 3,73622893137896)
(-1,71449416807636 , 8,97055386917724)
(0,741356518660853 , 3,71695103501793)
(10,0512978975691 , -14,5626838678994)
Nsd Size: 5
eps = 0,9
minPts = 30
Execution Time: 108 ms
```

Figure 9. dfkiartificial300 Data set analysis

Table 4. Running time of serial and parallel executions

Table Name	Data Set Length	Eps ve minPts Values	Run Time Serial(mS)	Run Time Parallel (mS)
real_2	1439	Eps: 1000 minPts: 50	54	39
dfkiartificial300	3000	Eps: 0,9 minPts: 30	232	108

D. Non-Standard Value Detection Performance of the Developed Application

The non-standard values specified in the data sets are compared with the non-standard values found by the developed application. The success rates of the application in finding non-standard values are given according to the eps and minPts values detected in the previous sections. Although there are 16 non-standard values in the real_2 data set, the application detected 15 of them. In the dfkiartificial300 data set, the success rate was found to be 94% (Table 5), even though the number of nonstandard values found by the application was equal to the number of nonstandard values in the data set.

The reason is that the application detected the values that are not non-standard in the data set as non-standard values and that could not detect certain non-standard values. In the kNN-dist chart obtained by using R Project software, eps and minPts values were obtained by visually selecting (approximate value) over the graph. When selecting these values, developing a more accurate method of calculation with sensitive decimals can improve the accuracy rate.

Table 5. Non-standard value detection success rates

Table Name	Eps and minPts Values	Number of Nonstandard Values in Data Set	Number Detected of Nonstandard Values	Success Rate (%)
real_2	Eps:1000 minPts:50	16	15	93,75
dfkiartificial300	Eps:0,9 minPts:30	37	37	94

E. *Testing of the Application for Different Configurations and the Results*

This study has been tested on two separate systems to measure the computation time of different performance computers. Tests are performed in two different computers with features: Intel(R) Core (TM) i7 / 3630QM CPU 2.40GHz (4 cores - 8 threads) 16GB memory and Intel(R) Core (TM) i5 / 2410M CPU 2.30GHz (2 cores - 4 threads) 4GB memory. Their execution run time were measured. The obtained results are shown in Table 6. As a result of the experiment, the processor speed, the number of cores and the high memory size, have shortened the total processing time, thus affecting the result positively (Fig. 10).

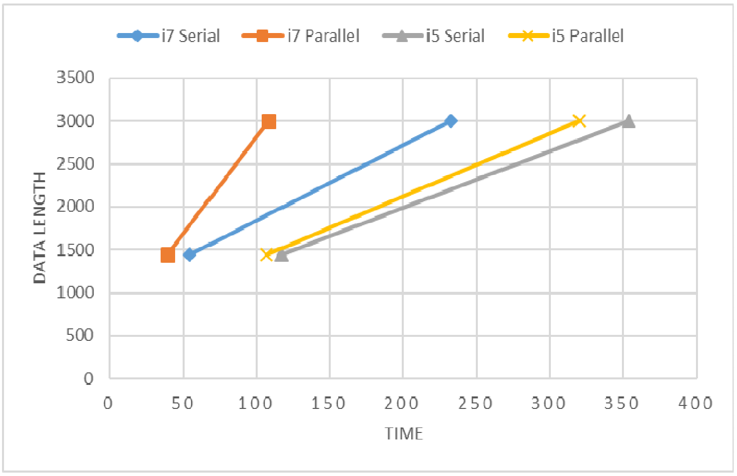


Figure 10. Data Length - Time (mS) Graph

Table 6. Application run time in different computer configurations

Data Set	Computer Configuration			
	Intel Core i7 / 3630QM CPU 2.40GHz (4 core - 8 thread) / Memory: 16GB		Intel Core i5 / 2410M CPU 2.30GHz (2 core - 4 thread) / Memory: 4GB	
	Run Time (mS)			
	Serial	Parallel	Serial	Parallel
real_2	54	39	76	70
dfkiartificial300	232	108	291	191

III. CONCLUSION

Digital data sizes are increasing gradually nowadays. Generating approximately 2.5 quintillion bytes of digital data per day shortens data processing and computation times, making the need for increased performance inevitable. Non-standard data detection of these data sets and their quick analysis have vital importance. In this study, DBSCAN algorithm is used as a density-based clustering algorithm which is frequently used in data mining processes for the detection of non-standard data. The DBSCAN algorithm clusters the points in the data sets and marks the points outside the cluster as non-standard data. The application was developed in Visual Studio environment with C# in the Microsoft .Net platform. 2 different data sets were studied in the developed application and the data sets were analyzed. The points in the data sets were shown on the graphs non-standard values were visually monitored. Although the DBSCAN algorithm is successful in clustering and detecting non-standard values, the need for eps and minPts parameters was seen as a disadvantage of the algorithm. The kNN-dist graph is plotted with the R Project software to determine the optimum eps and minPts values and the appropriate values are determined from the graph. The application made suitable for programming through the parallel classes in the TPL provided with. Net Framework 4.0.

As a result of the application, serial and parallel programming performances were compared on relatively large data sets. As the number of parallel operation data increased, it performed better and the application reached the result in a shorter time. Parallel programming in small data sets did not make a contribution to performance. The application was tested separately in Intel i7 and i5 processors, performance graphs were shown.

IV. DISCUSSION

This study shows that the DBSCAN algorithm is sensitive to non-standard values and can be used to detect these values. The fact that the analysis performed by the application detects the non-standard values at high accuracy rates is important in terms of finding the anomalies in datasets and is open for improvement.

In the future, developing an application that automatically selects the appropriate eps and minPts parameters for the DBSCAN algorithm will be useful for detecting non-standard data in the data set being analyzed. It will be useful to develop and test applications using different clustering algorithms for the detection of nonstandard values. In terms of performance and speed, applications can be developed and tested with other programming languages besides C# (C, C++, Java, Python etc.). Another approach is to use GPU programming as an alternative.

ACKNOWLEDGMENTS

This study was granted by Süleyman Demirel University Scientific Research Unit (SDU BAP) with the project number of 4199-YL1-14. We thank for its valuable supports.

REFERENCES

[1] IBM. (2016). What is Big Data? https://www 01.ibm.com/software/data/bigdata/what-is-big-data.html, (25.03.2020).

[2] Güçlü, M. (2012). Detection of Outlier Value with Artificial Immune System Based Algorithm. M.Sc. Thesis, Yıldız Technical University, Institute of Science and Technology, Istanbul, Turkey

[3] Duan, L., Xu, L., Liu, Y., Lee, J. (2009). Cluster-Based Outlier Detection. Annals of Operations Research, 168(1), 151-168.

[4] Ercan, U., Akar, H., Koçer, A. (2013). Basic Algorithms Used in Parallel Programming. Academic Informatics Conference, 23-25 January 2013, Antalya, Turkey, 861-866.

[5] Durmuş, B. (2013). Virtual Parallel Machine. M.Sc. Thesis, Dumlupınar University, Institute of Science and Technology, Kütahya, Turkey.

[6] Yang, J., He, Q. (2018). Scheduling Parallel Computations by Work Stealing: A survey. International Journal of Parallel Programming, 46(2), 173-197.

[7] İnce, K. (2013). Application of Genetic Algorithms with Parallel Programming in Multicore Architectures. M.Sc. Thesis, İnönü University, Institute of Science and Technology, Malatya, Turkey.

[8] Akçay, M., Erdem, H.A. (2013). Parallel Computing with Intel Parallel Studio. XVIII. Internet Conference in Turkey, 9-11 December 2013, İstanbul University, 79-83.

[9] Anthes, G. (2014). Researchers Simplify Parallel Programming. Communications of the ACM, 57(11), 13-15.

[10] Kalva, H., Colic, A., Garcia, A., Furht, B. (2011). Parallel Programming for Multimedia Applications. Multimedia Tools and Applications, 51(2), 801-818.

[11] Güneş, A. (2011). Recognition of Handwriting Numbers by Parallel Programming. M.Sc. Thesis, Süleyman Demirel University, Institute of Science and Technology, Isparta, Turkey.

[12] Drew, J. (2013). Parallel Programming in C # and Other Alternatives, https://www.codeproject.com/articles/701175/parallel-programming-in-csharp-and-other-alternati, (15.01.2019).

[13] Lazar, A. (2018). Task Parallel Library for Easy Multi-Threading in .NET Core [Tutorial]. https://hub.packtpub.com/task-parallel-library-multi-threading-net-core/, (25.03.2018).

[14] Ovla, H.D., Taşdelen, B. (2012). Outlier Value Management. Mersin University Journal of Health Sciences, 5 (3), 1-8.

[15] Aktürk, Z., Acemoğlu, H. (2010). Research and Practice Statistics for Health Care Workers. Anadolu Matbaası, İstanbul, 325.

[16] Vural, A. (2007). Effects of Outliers on Regression Models and Robust Estimators. Master Thesis, Marmara University, Institute of Social Sciences, İstanbul, Turkey.

[17] Moreira, A., Maribel, Y.S., Carneiro, S. (2005). Density-Based Clustering Algorithms-DBSCAN and SNN. University of Minho, Portugal. https://pdfs.semanticscholar.org/6227/2d87e82ffdec283c6da9d16f5065d7c44835.pdf?_ga=2.241964354.1371730934.1589407236-1802936592.1589407236, (15.05.2018).

[18] Cassisi, C., Ferro, A., Giugno, R., Pigola, G., Pulvirenti, A. (2013). Enhancing Density-Based Clustering: Parameter Reduction and Outlier Detection. Information Systems, 38(3), 317-330.

[19] Khan, I., Capozzoli, A., Corgnati, S.P., Cerquitelli, T. (2013). Fault Detection Analysis of Building Energy Consumption using Data Mining Techniques. Energy Procedia, 42, 557-566.

[20] Birant, D., Kut, A. (2006). Spatio-Temporal Outlier Detection in Large Databases. Journal of Computing and Information Technology, 14(4), 291-297.

[21] Bilgin, T.T., Çamurcu, Y. (2005). Applied Comparison of DBSCAN, OPTICS and k-Means Clustering Algorithms. Politeknik Journal, 8 (2), 139-145.

[22] Yahoo. (2016). Computing Systems Data. S5 - A Labeled Anomaly Detection Dataset, version 1.0 (16M). http://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70, (03.12.2016).

[23] German Research Center for Artificial Intelligence. (2016). http://madm.dfki.de/_media/downloads/dfki-artificial-3000-unsupervised-ad.zip, (15.05.2016).

[24] Freeman, A. (2010). Pro .NET 4 Parallel Programming in C#. Apress, USA, 311.

[25] Elbatta, M.T.H., Ashour, W.M. (2013). A Dynamic Method for Discovering Density Varied Clusters. International Journal of Signal Processing, Image Processing and Pattern Recognition, 6 (3), 123-134.

[26] Microsoft Docs. (2018). Map dependencies with code maps. https://docs.microsoft.com/tr-tr/visualstudio/modeling/map-dependencies-across-your-solutions?view=vs-2019, (30.03.2019).