Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 2519782, 12 pages
https://doi.org/10.1155/2017/2519782

Research Article

On the Accuracy and Parallelism of GPGPU-Powered Incremental Clustering Algorithms

Chunlei Chen,[1] Li He,[2] Huixiang Zhang,[3] Hao Zheng,[4] and Lei Wang[1]

[1]*School of Computer Engineering, Weifang University, Weifang, Shandong 261061, China*
[2]*School of Electromechanical Engineering, Guangdong University of Technology, Guangzhou 510006, China*
[3]*School of Automation, Northwestern Polytechnical University, Xi'an, China*
[4]*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA*

Correspondence should be addressed to Chunlei Chen; chunlei.chen@wfu.edu.cn

Received 29 January 2017; Revised 17 July 2017; Accepted 31 July 2017; Published 11 October 2017

Academic Editor: Amparo Alonso-Betanzos

Incremental clustering algorithms play a vital role in various applications such as massive data analysis and real-time data processing. Typical application scenarios of incremental clustering raise high demand on computing power of the hardware platform. Parallel computing is a common solution to meet this demand. Moreover, General Purpose Graphic Processing Unit (GPGPU) is a promising parallel computing device. Nevertheless, the incremental clustering algorithm is facing a dilemma between clustering accuracy and parallelism when they are powered by GPGPU. We formally analyzed the cause of this dilemma. First, we formalized concepts relevant to incremental clustering like evolving granularity. Second, we formally proved two theorems. The first theorem proves the relation between clustering accuracy and evolving granularity. Additionally, this theorem analyzes the upper and lower bounds of different-to-same mis-affiliation. Fewer occurrences of such mis-affiliation mean higher accuracy. The second theorem reveals the relation between parallelism and evolving granularity. Smaller work-depth means superior parallelism. Through the proofs, we conclude that accuracy of an incremental clustering algorithm is negatively related to evolving granularity while parallelism is positively related to the granularity. Thus the contradictory relations cause the dilemma. Finally, we validated the relations through a demo algorithm. Experiment results verified theoretical conclusions.

1. Introduction

1.1. Background. Due to the exciting advancements in digital sensors, advanced computing, communication, and massive storage, tremendous amounts of data are being produced constantly in the modern world. The continuously growing data definitely imply great business value. However, data are useless by themselves; analytical solutions are demanded to pull meaningful insight from the data, such that effective decisions can be achieved. Clustering is an indispensable and fundamental data analysis method. The traditional clustering algorithm is executed in a batch-mode; namely, all data points are necessarily loaded into memory of the host machine. In addition, every data point can be accessed unlimited times during the algorithm execution. Nevertheless, the batch-mode clustering algorithm can not adjust the clustering result in an evolving manner. For instance, it is necessary to incrementally cluster the evolving temporal data such that the underlying structure can be detected [1]. In stream data mining, a preprocessing task like data reduction needs the support of incremental clustering [2, 3]. In addition, incremental clustering can significantly contribute to massive data searching [4]. To sum up, the evolving capability of incremental clustering is indispensable under certain scenarios, such as memory-limited applications, time-limited applications, and redundancy detection. A practical application may possess arbitrary combination of the three aforementioned characteristics. The incremental clustering algorithm proceeds in an evolving manner, namely, processing the input data step by step. In each step, the algorithm receives a newly arrived subset of the input and obtains the new knowledge of this subset. Afterwards, historic clusters of the previous step are updated with the new knowledge. Subsequently, the updated clusters serve as input of the next step. With regard to

the first step, there is no updating operation. The new knowledge obtained in the first step serves as input of the second step. Application scenarios of incremental clustering generally raise high requirements on computing capacity of the hardware platform. General Purpose Graphic Processing Unit (GPGPU) is a promising parallel computing device. GPGPU has vast development prospects due to following superiorities: much rapider growing computing power than CPU, high efficiency-cost ratio, and usability.

1.2. Motivation and Related Works. Our previous work revealed that the existing incremental clustering algorithms are confronted with an accuracy-parallelism dilemma [5, 6]. In this predicament, the governing factor is the evolving granularity of the incremental clustering algorithm. For instance, the point-wise algorithms proceed in fine granularity. In each step, the algorithms only receive a single data point (serving as the new knowledge); this point will either be assigned to an existing historic cluster or induce an independent cluster in the historic clusters. Such algorithms generally achieve favorable clustering accuracy. However, they sacrifice parallelism due to strong data dependency. Namely, the next data point cannot be processed until the current one is completely processed. Modern GPGPUs can contain thousands of processing cores, while the number of existing historic clusters needs to increase progressively even if the number eventually reach the magnitude of thousand. GPGPU can fully leverage its computing power only if it runs abundant threads in a SIMD (Single Instruction Multiple Data) manner (commonly twice the number of processing cores or even more). In addition, the work-depth is inevitably no less than the amount of input data points under point-wise setting. Consequently, the computing power is ineluctably underutilized. Moreover, more kernel launches (GPGPU code execution) are required if work-depth is larger. Time overhead of kernel launch is generally high. Some representative point-wise incremental clustering algorithms were elaborated in [7–19]. Ganti et al. used block-evolving pattern to detect changes in stream data [20]. Song et al. adopted the block-wise pattern and proposed an incremental clustering algorithm of GMM (Gaussian Mixture Model) [21]. The algorithm of [21] proceeds in coarse granularity. Each step contains three substeps. First, obtain the new knowledge by running the standard EM (Expectation Maximum) algorithm on a newly received data block. Second, identify the statistically equivalent cluster pairs between the historic clusters and the new clusters. Finally, merge the equivalent cluster pairs separately. The standard EM algorithm of GMM is inherently GPGPU-friendly [22]. The algorithm of [21] maintains the inherent parallelism. However, its clustering accuracy exhibits degradation by order of magnitude, compared to its batch-mode counterpart (the standard EM algorithm of GMM) [5]. Moreover, we qualitatively analyzed the reason why block-wise pattern tends to induce accuracy degradation in our previous work [6]. Algorithms of [23, 24] are also block-wise. D-Stream is ostensibly point-wise [25]. Nevertheless, D-Stream is essentially block-wise due to the fact that mapping data points into grids can be parallelized in a SIMD manner. As far as we know, most existing works focus on clustering accuracy. However, existing algorithms, even the block-wise ones, do not explicitly consider algorithm parallelism on SIMD many-core processing devices like GPGPU. Some recent works formally analyzed issues of clustering or machine learning algorithms. Ackerman and Dasgupta pointed out a limitation that incremental clustering cannot detect certain types of cluster structure [26]. They formally analyzed the cause of this limitation and proposed conquering this limitation by allowing extra clusters. Our work is similar to that of [26] in the sense that we also formally analyzed the reason why incremental clustering is inefficient under certain conditions. In contrast, we elaborated our work under the context of GPGPU-acceleration. Ackerman and Moore also formally analyzed perturbation robustness of batch-mode clustering algorithm [27, 28]. Nevertheless, works of [27, 28] only concentrated on classical batch-mode clustering algorithms. Gepperth and Hammer qualitatively analyzed challenges that incremental learning is facing [29]. They pointed out a dilemma between stability and plasticity. However, we focus on the dilemma between accuracy and parallelism.

1.3. Main Contribution. In this paper, we extend our previous works of [6, 30] in the following ways. First, some vital concepts (such as incremental clustering and evolving granularity) are formally defined. Second, we formally proved how evolving granularity exerts influence on accuracy and parallelism of incremental clustering algorithms. In this way, the starting point of our previous works can be formally validated. Finally, we demonstrated the theoretical conclusions through a demo algorithm. The conclusions will be the footstone of our future work.

2. Formal Definition of Terminologies

2.1. Terminologies on Incremental Clustering

Definition 1 (incremental clustering). $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_T$ is a series of data points ($\mathbf{x}_i \in R^d$, $1 \le i \le T$). The data points are partitioned into $T1$ sets: $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \ldots, \mathbf{X}_{T1}$. This partition satisfied the following conditions:

(1) $\mathbf{X}_j \neq \emptyset$ ($j = 1, 2, 3, \ldots, T1$).

(2) If $\mathbf{x}_{i1} \in \mathbf{X}_{j1}$, $\mathbf{x}_{i2} \in \mathbf{X}_{j2}$ ($i1 \neq i2$, $j1 \neq j2$), then $j2 > j1 \Leftrightarrow i2 > i1$.

A data analysis task adopts discrete time system and time stamps are labeled as $1, 2, 3, \ldots$. *This task is incremental clustering if and only if the following applies:*

(1) When $t = 1$, the task receives $\mathbf{X}_1$. In time interval $[1, 2]$, the task partitions $\mathbf{X}_1$ into clusters and $\mathbf{C}_1$ is the set of these clusters. The entire input to the task is $\mathbf{X}_1$.

(2) When $t = j$ ($j = 2, 3, 4, \ldots$), the task receives $\mathbf{X}_t$. In time interval $[t, t+1]$, the task resolves the set of clusters $\mathbf{C}_t$ such that $\forall \mathbf{x}_j \in \bigcup_{1 \le k \le t} \mathbf{X}_k$ can find its affiliated cluster in $\mathbf{C}_t$. Entire inputs to the task are $\mathbf{X}_t$ and $\mathbf{C}_t$.

Definition 2 (the tth step and historic clustering result of the tth step). An algorithm for incremental clustering is an incremental clustering algorithm. The time interval $[t, t+1]$ $(t = 1, 2, 3, \ldots)$ is the tth step of an incremental clustering algorithm (or step t of an incremental clustering algorithm). $\mathbf{C}_t$ is the historic clustering result of the tth step.

Definition 3 (micro-cluster). Let *batchAlgorithm* represent a batch-mode clustering algorithm. $\overrightarrow{\xi} = (\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(p)}) \in (R^+)^p$ $1 \leq p \leq d$ is the parameter of *batchAlgorithm*. *batchAlgorithm* can partition $\mathbf{X}_t$ into subsets Sub_k $(k = 1, 2, 3, \ldots, K_{t,\text{new}})$. $\overrightarrow{\xi}_0 = (\xi_0^{(1)}, \xi_0^{(2)}, \ldots, \xi_0^{(p)}) \in (R^+)^p$ is a constant vector. Sub_k is a $\overrightarrow{\xi}_0$-micro-cluster produced by *batchAlgorithm* if and only if the following applies:

(1) $\mathbf{X}_t = \bigcup_{1 \leq k \leq K_{t,\text{new}}} \text{Sub}_k$.

(2) $\forall k1 \neq k2$, $\text{Sub}_{k1} \cap \text{Sub}_{k2} = \emptyset$.

(3) Sub_k forms a data cloud in d-dimensional space. The hypervolume of this data cloud is positively related to $\xi_0^{(u)}$ $(u = 1, 2, 3, \ldots, p)$. Sub_k contains and only contains one data point if $\overrightarrow{\xi}_0 = \overrightarrow{\mathbf{0}}$.

(4) $\xi_0^{(u)}$ $(u = 1, 2, 3, \ldots, p)$ are all preset constant values.

Definition 4 (batch-mode part and incremental part of step t). Some incremental clustering algorithms divide step t $(t = 1, 2, 3, \ldots)$ into two parts [21, 23–25]: In the first part, $\mathbf{X}_t$ is partitioned into *new clusters (or new micro-clusters)* pursuant to certain similarity metrics. $\mathbf{C}_{t,\text{new}}$ is the set of these clusters (or micro-clusters); in the second part, $\mathbf{C}_t$ is resolved based on $\mathbf{C}_{t,\text{new}}$ and $\mathbf{C}_{t-1}$ (if any). The number of clusters (or micro-clusters) in $\mathbf{C}_{t,\text{new}}$ is denoted as $K_{t,\text{new}}$. The first part can be accomplished by a batch-mode clustering algorithm; this part is *the batch-mode part of step t*. The second part is *the incremental part of step t*.

Definition 5 (benchmark algorithm, benchmark clustering result, and benchmark cluster). Denote $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \ldots, \mathbf{X}_{T1}$ (pursuant to Definition 1) as *StrParti*. Incremental clustering is applied to *StrParti*, and $\mathbf{C}_{T0}$ is the historic clustering result of the $T0$th step $(T0 \leq T1)$; let $\mathbf{X}_B = \bigcup_{1 \leq k \leq T0} \mathbf{X}_k$. If $\mathbf{X}_B$ could be entirely loaded into memory of the host machine and were processed by a certain batch-mode clustering algorithm, then the resulting clusters were in a set of clusters denoted as $\mathbf{C}_{T0,\text{benchmark}}$. The batch-mode algorithm is called *benchmark algorithm of the incremental clustering algorithm*. $\mathbf{C}_{T0,\text{benchmark}}$ is the benchmark clustering result up to step $T0$. An arbitrary cluster of $\mathbf{C}_{T0,\text{benchmark}}$ is a *benchmark cluster*.

Definition 6 (local benchmark cluster). $\mathbf{C}_{T,\text{Benchmark}} = \{\text{Clu}_k \mid k = 1, 2, 3, \ldots, K_{T,\text{Benchmark}}\}$ is the benchmark clustering result up to the Tth step; Clu_k $(k = 1, 2, 3, \ldots, K_{T,\text{Benchmark}})$ represent the benchmark clusters in $\mathbf{C}_{T,\text{benchmark}}$. $\mathbf{X}_t$ is the newly received data set of step t. All data points of $\mathbf{X}_t$ are labeled such that points with the same label are affiliated to the same benchmark cluster. Partition $\mathbf{X}_t$ into nonempty subsets, noted as SubClu_k $(k = 1, 2, 3, \ldots, K_t')$. These subsets satisfy the following conditions:

(1) $\mathbf{X}_t = \bigcup_{1 \leq k \leq K_t'} \text{SubClu}_k$ and $\forall k1 \neq k2$, $\text{SubClu}_{k1} \cap \text{SubClu}_{k2} = \emptyset$.

(2) If $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_t$ then $\mathbf{x}_i, \mathbf{x}_j \in \text{SubClu}_u \Leftrightarrow \mathbf{x}_i$ and $\mathbf{x}_i$ possess the same label.

(3) $\text{SubClu}_u, \text{SubClu}_v \subset \mathbf{X}_t$; if $\mathbf{x}_i \in \text{SubClu}_u$, $\mathbf{x}_j \in \text{SubClu}_v$ $(u \neq v)$, then $\mathbf{x}_i$ and $\mathbf{x}_j$ have different labels.

SubClu_k $(k = 1, 2, 3, \ldots, K_t')$ are called the *local benchmark clusters of step t* or *local benchmark clusters* for short. We abbreviate local benchmark cluster to LBC.

Definitions 1–4 actually provide terminologies to formally interpret the concept of incremental clustering as well as execution mechanism of incremental clustering. Definitions 5 and 6 furnish a benchmark to evaluate the accuracy of an incremental clustering algorithm.

2.2. Terminologies on Evolving Granularity, Clustering Accuracy, and Parallelism

Definition 7 (containing hypersurface of a data set). Let $\text{Clu}_w \subset R^d$ represent a set of data points. HS is a hypersurface in the d-dimensional space. HS is the *containing hypersurface* of Clu_w, if and only if HS is a close hypersurface and an arbitrary point of Clu_w is within the interior of HS.

Definition 8 (envelope hypersurface, envelop body, and envelop hypervolume of a data set). Let $\text{Clu}_w \subset R^d$ represent a set of data points. $S_{\text{HS}} = \{\text{HS}_u \mid u = 1, 2, 3, \ldots, U\}$ is the set of containing hypersurfaces of Clu_w. Let $\mathbf{V}_u$ represent the hypervolume encapsulated by HS_u. Let HS_v be a hypersurface. HS_v is the envelope hypersurface of Clu_w, if and only if $\text{HS}_v = \arg\min_{\text{HS}_u \in S_{\text{HS}}} \mathbf{V}_u$. Let EN_w represent the *envelope hypersurface* of Clu_w; the region encapsulated by EN_w is the envelope body of Clu_w; the hypervolume of this envelope body is the *envelope hypervolume* of Clu_w.

Definition 9 (core hypersphere, margin hypersphere, core hypervolume, and margin hypervolume of a data set). Let $\text{Clu}_w \subset R^d$ be a data set. EN_w is the envelope hypersurface of Clu_w; $\mathbf{center}_w$ represents the geometric center of the envelope body of Clu_w. $\text{dist}(\mathbf{x})$ represents the distance between $\mathbf{center}_w$ and an arbitrary point on EN_w. $\text{dist}_{w,\min} = \min_{\mathbf{x} \in \text{EN}_w} \text{dist}(\mathbf{x})$; $\text{dist}_{w,\max} = \max_{\mathbf{x} \in \text{EN}_w} \text{dist}(\mathbf{x})$.

A hypersphere is the *core hypersphere* of Clu_w if and only if this hypershpere is centered at $\mathbf{center}_w$ and its radius is $\text{dist}_{w,\min}$, noted as $\text{SPSur}_{\min}$. The hypervolume encapsulated by $\text{SPSur}_{\min}$ is the *core hypervolume* of Clu_w, noted as $\text{SP}_{\min}(\text{Clu}_w)$.

A hypersphere is the margin hypersphere of Clu_w if and only if it is centered at $\mathbf{center}_w$ and its radius is $\text{dist}_{w,\max}$, noted as $\text{SPSur}_{\max}$. The hypervolume encapsulated by $\text{SPSur}_{\max}$ is the margin hypervolume of Clu_w, noted as $\text{SP}_{\max}(\text{Clu}_w)$.

Definition 10 (core evolving granularity, margin evolving granularity, and average evolving granularity). In the tth step, the incremental clustering algorithm receives data set $\mathbf{X}_t$. $\mathbf{X}_t$ is partitioned into nonempty subsets pursuant to certain metrics: Sub_k $(k = 1, 2, 3, \ldots, K_{t,\text{new}})$ such that $\forall k1 \neq k2$,

$\text{Sub}_{k1} \cap \text{Sub}_{k2} = \emptyset$, and $\mathbf{X}_t = \bigcup_{1 \le k \le K_{t,\text{new}}} \text{Sub}_k$. Let $\text{SP}_{\min}(\text{Sub}_k)$ be the core hypervolume of Sub_k. Then *in the tth step, the core evolving granularity* of the algorithm is $\text{SubGra}_{\min,t} = \min_{1 \le k \le K_{t,\text{new}}} \text{SP}_{\min}(\text{Sub}_k)$.

Up to the Tth step, the core evolving granularity of the algorithm is $\text{Gra}_{\min,T} = \min_{1 \le t \le T} \text{SubGra}_{\min,t}$.

Let $\text{SP}_{\max}(\text{Sub}_k)$ be the margin hypervolume of Sub_k. Then *in the tth step, the margin evolving granularity* of the algorithm is $\text{SubGra}_{\max,t} = \max_{1 \le k \le K_{t,\text{new}}} \text{SP}_{\max}(\text{Sub}_k)$.

Up to the Tth step, the margin evolving granularity of the algorithm is $\text{Gra}_{\max,T} = \max_{1 \le t \le T} \text{SubGra}_{\max,t}$. Let EN_k be the envelope hypersurface of Sub_k, and $\text{SubSet}_k = \text{EN}_k \cap \text{Sub}_k$; $|\text{SubSet}_k|$ is the number of data points within SubSet_k; $\mathbf{center}_w$ represents the geometric center of the envelope body of Sub_k. $\text{dist}_{k,u}$ represents the distance between $\mathbf{center}_w$ and $\mathbf{x}_u$. Let $\text{ave\_dist}_k = (\sum_{\mathbf{x}_u \in \text{SubSet}_k} \text{dist}_{k,u}) / |\text{SubSet}_k|$.

$\text{SP}_{\text{ave}}(\text{Sub}_k)$ is the hypervolume of the hypersphere whose center is at $\mathbf{center}_w$ and radius is ave_dist_k. *In the tth step, the average evolving granularity* of the algorithm is $\text{SP}_{\text{ave}}(\text{Sub}_k)$.

Up to the Tth step, the average evolving granularity of the algorithm is $\text{Gra}_{\text{ave},T} = (\sum_{t=1}^{T} \text{Gra}_{\text{ave},t} K_{t,\text{new}}) / \sum_{t=1}^{T} K_{t,\text{new}}$.

Definition 11 (different-to-same mis-affiliation). Different-to-same mis-affiliation is the phenomenon that, in step t, data points from different benchmark clusters are affiliated to the same cluster of $\mathbf{C}_{t,\text{new}}$ or $\mathbf{C}_t$.

Definition 12 (same-to-different mis-affiliation). Same-to-different mis-affiliation is the phenomenon that, in step t, data points from the same benchmark cluster are affiliated to the different clusters of $\mathbf{C}_{t,\text{new}}$ or $\mathbf{C}_t$.

We adopt Rand Index [31] to measure clustering accuracy. Larger Rand Index means higher clustering accuracy.

There are numerous criterions of cluster separation measurement in the existing literatures. We select Rand Index due to the fact that this criterion directly reflects our intent: measuring the clustering accuracy by counting occurrences of data point mis-affiliations.

Definition 13 (serial shrink rate (SSR)). Let incAlgorithm represent an incremental clustering algorithm. In step t ($t = 1, 2, 3, \ldots$), the batch-mode part generates micro_t micro-clusters. Suppose incAlgorithm totally clustered N data points up to step T_0. Up to step T_0, the serial shrink rate of incAlgorithm is

$$\text{SSR} = \frac{\left(\sum_{t=1}^{T_0} \text{micro}_t\right)}{N}. \tag{1}$$

Lower SSR means that less computation inevitably runs in a non-GPGPU-friendly manner. Consequently, smaller SSR means improved parallelism. Work-depth [32] of the algorithm can shrink if SSR is smaller. Hence, more computation can be parallelized.

3. Theorems of Evolving Granularity

3.1. Further Explanation on the Motivation. GPGPU-accelerated incremental clustering algorithms are facing a dilemma between clustering accuracy and parallelism. We endeavor to explain the cause of this dilemma through formal proofs. The purpose of our explanation is that the formal proofs reveal a possible solution to seek balance between accuracy and parallelism. This basic idea of this solution is discussed as follows.

In the batch-mode part of the tth step, data points from different local benchmark clusters may be mis-affiliated to the same micro-cluster of $\mathbf{C}_{t,\text{new}}$. Theorem 14 points out that the upper and lower bounds of the mis-affiliation probability are negatively related to margin evolving granularity and core evolving granularity, respectively. The proof of this theorem demonstrates that larger evolving granularity results in more occurrences of different-to-same mis-affiliation.

The batch-mode part should evolve in fine granularity to produce as many homogeneous micro-clusters as possible. Only in this context, the operations of incremental part are sensible. Namely, the incremental part cannot eliminate different-to-same mis-affiliation induced by the batch-mode part. The incremental part absorbs advantages of point-wise algorithms by processing micro-clusters sequentially. This part should endeavor to avoid both same-to-different and different-to-same mis-affiliations on the micro-cluster-level.

Nevertheless, Theorem 15 proves that parallelism is positively related to evolving granularity. Thus, the contrary relations cause the dilemma.

However, we can adopt GPGPU-friendly batch-mode clustering algorithm in the batch-mode part. Moreover, the total number of micro-clusters is much smaller than that of data points up to a certain step. Consequently, the work-depth can be dramatically smaller than that of a point-wise incremental clustering algorithm.

3.2. Theorem of Different-to-Same Mis-Affiliation

Theorem 14. *Let $P_{t,mis}$ represent the probability of different-to-same mis-affiliations induced by the batch-mode part of the tth step. $Gra_{min,T}$ and $Gra_{max,T}$ are the core evolving granularity and margin evolving granularity up to the Tth step, respectively. The upper bound of $P_{t,mis}$ is negatively related to $Gra_{max,T}$ and the lower bound of $P_{t,mis}$ is negatively related to $Gra_{min,T}$.*

Suppose $\mathbf{X}_t$ contains K_t' local benchmark clusters (LBC). $LBCSet_t$ is a set containing these LBCs. Between any two adjacent LBCs there exists a boundary curve segment. The boundary curve segment between LBC $SubClu_{k1}$ and LBC $SubClu_{k2}$ ($k1, k2 = 1, 2, 3, \ldots, K_t'$; $k1 \neq k2$) is noted as $Cur^{(k1,k2)}$. Obviously, $Cur^{(k1,k2)}$ and $Cur^{(k2,k1)}$ represent the same curve segment. We define a probability function to represent $P_{t,mis}$:

$$P_{t,mis}(r) = \frac{\sum_{SubClu_{k1}, SubClu_{k2} \in LBCSet_t, k1<k2} curV^{(r)}\left(Cur^{(k1,k2)}\right)}{\sum_{SubClu_k \in LBCSet,} V\left(SubClu_k\right)}, \tag{2}$$

where

$$curV^{(r)}\left(Cur^{(k1,k2)}\right) = \begin{cases} \text{the hypervolume enclosedby } D_t^{(k1,k2)}, & \left(\text{if } SubClu_{k1} \text{ is adjacent to } SubClu_{k2}, D_t^{(k1,k2)} = \left\{x \in R^d \mid \text{the distance from } x \text{ to } Cur^{(k1,k2)} \text{ is smaller than } r\right\}\right), \\ 0, & \left(\text{if } SubClu_{k1} \text{ is not adjacent to } SubClu_{k2} \text{ or } k1 = k2\right), \end{cases} \quad (3)$$

and $V(SubClu_k)$ is the hypervolume enclosed by envelope surface of $SubClu_k$.

$P_{t,mis}(r)$ reaches the upper bound if r equals the radius corresponding to the margin evolving granularity (Definition 10) in step t; $P_{t,mis}(r)$ reaches the lower bound if r equals the radius corresponding to the core evolving granularity (Definition 10) in step t.

Proof. In order to more intuitively interpret this theorem, we discuss the upper and lower bounds in two-dimensional space. The following proof can be generalized to higher-dimensional space.

In two-dimensional space, the envelop hypersurface (Definition 8) degenerates to an envelope curve. The core hypersphere and margin hypersphere (Definition 9) degenerate to a core circle and a margin circle, respectively. The envelop body degenerates to the region enclosed by the envelope curve. The envelope hypervolume degenerates to the area enclosed by the envelope curve. Let incAlgorithm represent an incremental clustering algorithm. Each step of incAlgorithm includes batch-mode part and incremental part.

(1) Partition Data Points Pursuant to Local Benchmark Clusters. incAlgorithm receives $\mathbf{X}_t$ in the tth step. Partition $\mathbf{X}_t$ into local benchmark clusters (Definition 6, LBC for short): SubClu_u $(u = 1, 2, 3, \ldots, K_t')$. Let $\text{AR} = \bigcup_{u=1}^{K_t'} \text{SubClu}_u$. EN_u is the envelope curve of SubClu_u. $V(\text{SubClu}_u)$ represents the area enclosed by SubClu_u. Assume that SubClu_u $(u = 1, 2, 3, \ldots, K_t')$ are convex sets. (We can partition SubClu_u into a set of convex sets if it is not a convex set.)

(2) Partition the Boundary Curve between Two Local Benchmark Clusters into Convex Curve Segments. Let **Cur** be the set of boundary curves between any two adjacent LBCs. $\mathbf{Cur} = \{\text{Cur}_k \mid k = 1, 2, 3, \ldots, K\}$ where Cur_k is the boundary curve segment between two adjacent LBCs. Consider an arbitrary LBC SubClu_u. Suppose that there are totally U LBCs adjacent to SubClu_u. The boundary curves segments are $\text{Cur}_u^{(1)}, \text{Cur}_u^{(2)}, \ldots, \text{Cur}_u^{(U)}$. These boundary curve segments can be consecutively connected to form a closed curve such that only data points of SubClu_u are within the enclosed region. Further partition Cur_k into a set of curve segments such that $\text{Cur}_{k,g}$ $(g = 1, 2, 3, \ldots, G_k)$ are all convex curve segments. $\text{Cur}_{k,g}$ can be viewed as a set of data points.

(3) Construct Auxiliary Curves. Figure 1(a) illustrates an example of adjacent LBCs and a boundary curve. The black, gray, and white small circles represent three distinct LBCs (For clarity of the figure, we use small circles to represent data points). The black bold curve is the boundary curve between the black and white LBCs. We cut out a convex curve segment from this boundary curve, noted as $\text{Cur}_{k,g}$. Figure 1(b) magnifies $\text{Cur}_{k,g}$. Assume that the analysis formula of $\text{Cur}_{k,g}$ is $y = t(x)$ $(x \in D_{k,g} = [x_1, x_2])$.

Let Cir_E be a circle centered at point E. The radius of Cir_E is *threshold* $(threshold \in R^+)$. Place Cir_E to the right of $\text{Cur}_{k,g}$. Roll Cir_E along $\text{Cur}_{k,g}$ such that Cir_E is always tangent to Cir_E. Let Cir_I be a circle centered at point I. The radius of Cir_I is also *threshold*. Place Cir_I to the left of $\text{Cur}_{k,g}$. Roll Cir_I along $\text{Cur}_{k,g}$ such that all points of Cir_I are always to the left of $\text{Cur}_{k,g}$, except the points of tangency between Cir_I and $\text{Cur}_{k,g}$. The trajectories of points E, I form two curves, $y = t_1(x)$ and $y = t_2(x)$, respectively. Adjust the starting and ending points of $t1(x)$ and $t2(x)$ such that the definition domains of both curves are $D_{k,g} = [x_1, x_2]$.

(4) Characteristics of New Clusters. In step t, $\mathbf{X}_t$ is partitioned into new clusters (or new micro-clusters) (Definition 3) pursuant to certain methods. Let GR be a set containing data points of an arbitrary new cluster. Without loss of generality, let envelope curve of GR be a circle centered at $\mathbf{center}_{\text{GR}}$, noted as ENGR. The radius of this circle is noted as *radius*. We can view $\mathbf{center}_{\text{GR}}$ as a random variable. This random variable represents the possible coordinates of GR's center. Let $D(\text{SubClu}_u)$ represent a set of all vectors enclosed by envelope curve of SubClu_u in d-dimensional $(d = 2)$ space (including vectors on the envelope curve). Let $D(\text{AR}) = \bigcup_{u=1}^{K_t'} D(\text{SubClu}_u)$. The statistical characteristic of $\mathbf{X}_t$ is unknown before $\mathbf{X}_t$ is processed. Consequently, it is reasonable to assume that $\text{center}_{\text{GR}}$ obeys the uniform distribution on $D(\text{AR})$. Namely, we assume that every point within $D(\text{AR})$ is possible to be GR's center and the probabilities of every point are equal.

(5) Criterion of Different-to-Same Mis-Affiliation. Assume that we can neglect distance between the boundary curve and the right side of the black LBC's envelope curve in Figure 1. Similarly, assume that we can neglect distance between the boundary curve and the left side of the white LBC's envelope curve. The criterion of different-to-same mis-affiliation is as follows.

If $\text{ENGR} \cap \text{Cur}_{k,g} \neq \emptyset$, then GR contains data points of at least two distinct local benchmark clusters.

Smaller distance between $\mathbf{center}_{\text{GR}}$ and $\text{Cur}_{k,g}$ means higher probability of different-to-same mis-affiliation induced by the batch-mode part; the larger the radius is, the higher the probability is.

In Figure 1(b), two lines $(x = x_1,\ x = x_2)$ and two curves $(y = t_1(x),\ y = t_2(x))$ form an open domain, noted as

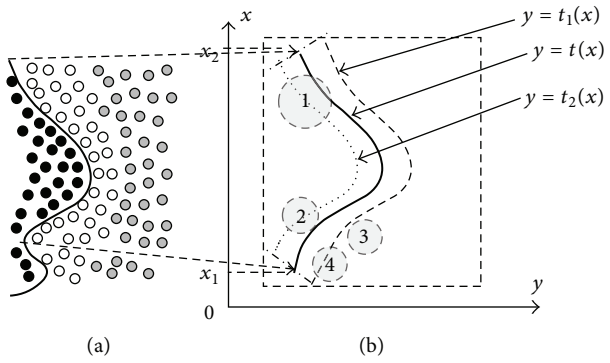


Figure 1: Relation between evolving granularity and different-to-same mis-affiliation induced by the batch-mode part.

$D_{\text{threshold}} \in R^2$. Let Set represent the set of threshold values that make the following causal relationship hold:

center$_{\text{GR}} \in D_{\text{threshold}}$ and $threshold \leq radius \Rightarrow \text{ENGR} \cap \text{Cur}_{k,g} \neq \emptyset$.

Part (3) of this proof explained the meaning of *threshold* ($threshold \in R^+$). GR's *threshold with regard to* $\text{Cur}_{k,g}$ is $threshlod_{\max} = \max_{threshold \in \text{Set}} threshold$. $D_{\text{critical}} = D_{\text{threshold}}|_{\text{threshold}=\text{threshlod}_{\max}}$. Different-to-same mis-affiliation can still occur as long as *radius* is sufficiently large even if $\text{center}_{\text{GR}} \notin D_{\text{critical}}$.

(6) Three Typical Situations of Different-to-Same Mis-Affiliation Induced by Batch-Mode Part. The value of $threshold_{\max}$ is dramatically influenced by the following factors: first: shape of LBC's envelop curve and second: the relative positions of $\text{Cur}_{k,g}$ and GR. Shape of LBC's envelope curve is generally irregular. We simplify the proof without loss of generality. As illustrated by Figure 2, assume that three sectors are consecutively connected to form the envelope curve. In addition, three sectors' centers overlapped on point O. Radiuses of sectors 1, 2, and 3 are $r1$, $r2$, and $r3$, respectively. $r1$ equals to the radius of the margin envelope circle. $r3$ equals to the radius of the core envelope circle. $r2$ is between $r1$ and $r3$. Generally, distances between point O and points on the envelope curve are between $r1$ and $r3$. Sector 2 can represent these ordinary points.

Let GR rotate around O. Figures 2(a), 2(b), and 2(c) illustrate three characteristic positions of GR during the rotation. In Figure 2(a), threshold is r_1. D_{critical} covers the largest area. In Figure 2(c), threshold is r_3. D_{critical} covers the smallest area.

(7) Probability of Different-to-Same Mis-Affiliation Induced by Batch-Mode Part: Lower Bound. As aforemetioned, a boundary curve between two LBCs can be partitioned into curve segments. Data points on both sides of a curve segment are affiliated to the same cluster if different-to-same mis-affiliation occurs (this mis-affiliation occurs in the batch-mode part of a certain step). Let $\text{Cur}_{k,g}$ represent a curve segment from a boundary curve. Let $P_{k,g}$ be the probability that data points on both sides of $\text{Cur}_{k,g}$ are affiliated to the same cluster. Considering all boundary curves in $\mathbf{X}_t$, $P_{t,\text{mis}}$ represents the total probability of different-to-same mis-affiliation in the batch-mode part of step t.

Let $r_{\min,T}$ be the radius of the hypersphere corresponding to core evolving granularity up to step T. Assume that auxiliary curves $y = t_1(x)$ and $y = t_2(x)$ are constructed under $threshold = r_{\min,T}$. On the basis of the previous parts of proof, we can draw the following inequalities:

$$P_{k,g} \geq \frac{\left\{\int_{D_{k,g}} \left|t_1(x) - t_2(x)\right| dx\right\}}{V(\text{AR})},$$
$$P_{t,\text{mis}} \geq \sum_{k=1}^{K} \sum_{g=1}^{G_k} P_{k,g}, \tag{4}$$
$$\left(threshold_{\max} = r_{\min,T}\right).$$

(8) Probability of Different-to-Same Mis-Affiliation Induced by Batch-Mode Part: Upper Bound. Let $r_{\max,T}$ be the radius of the hypersphere corresponding to margin evolving granularity. Assume that auxiliary curves $y = t_1(x)$ and $y = t_2(x)$ are constructed under $threshold = r_{\max,T}$. On the basis of the previous parts of proof, we can draw the following inequalities:

$$P_{k,g} \leq \frac{\left\{\int_{D_{k,g}} \left|t_1(x) - t_2(x)\right| dx\right\}}{V(\text{AR})},$$
$$P_{t,\text{mis}} \leq \sum_{k=1}^{K} \sum_{g=1}^{G_k} P_{k,g}, \tag{5}$$
$$\left(threshold_{\max} = r_{\max,T}\right).$$

□

3.3. Discussions on Theorem 14. (1) Theorem 14 focuses on different-to-same mis-affiliations other than same-to-different mis-affiliations. Hence we assume that we can neglect the distance between $\text{Cur}_{k,g}$ and envelope curve of the corresponding LBC (part (5) of the proof).

The probability of different-to-same mis-affiliation will be lower if this distance is not negligible. Consequently, inequalities (5) still hold. In this case, inequalities (4) give the lower bound under the worst situation.

(2) For simplification, we assume the envelope curve of GR is a circle (part (4) of the proof). Theorem 14 still holds even if the envelope curve is of arbitrary shape. The core evolving granularity is determined by $\text{dist}_{w,\min}$ (Definition 9). Larger $\text{dist}_{w,\min}$ always means higher $P_{t,\text{mis}}$ regardless of the envelope curves shape, or vice versa.

(3) Based on Theorem 14, we can declare that $P_{t,\text{mis}}$ is positively related to the hypervolume of GR's envelope body. In the batch-mode part of step t, we can partition $\mathbf{X}_t$ into more clusters (or micro-clusters) such that data points of each cluster (or micro-cluster) scatter within a smaller range in the d-dimensional space.

(4) The batch-mode part of ordinary block-wise algorithm clusters $\mathbf{X}_t$ without a global view of $\bigcup_{t'=1}^{t} \mathbf{X}_{t'}$.

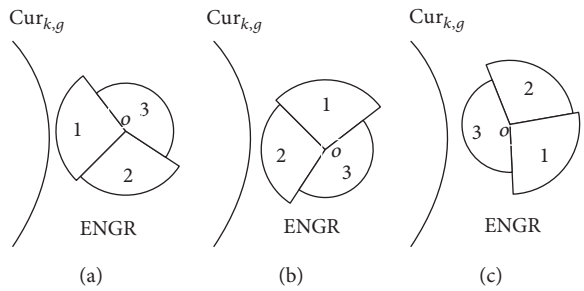


Figure 2: Typical examples of different-to-same mis-affiliations induced by batch-mode part.

Consequently, $\mathbf{C}_{t,\text{new}}$ is facing a high probability of different-to-same mis-affiliations. This means that numerous new clusters (or micro-clusters) in $\mathbf{C}_{t,\text{new}}$ are containing heterogeneous data points. No matter what methods the algorithm uses to identify and merge homogeneous clusters (or micro-clusters) between $\mathbf{C}_{t-1}$ and $\mathbf{C}_{t,\text{new}}$ (incorporates $\mathbf{C}_{t,\text{new}}$ into $\mathbf{C}_{t-1}$ to obtain $\mathbf{C}_t$), clusters in $\mathbf{C}_t$ will dramatically differentiate from the benchmark clusters up to step t.

3.4. Theorem of Parallelism. Let incAlgorithm represent an incremental clustering algorithm. Step t of incAlgorithm includes two parts: batch-mode part and incremental part. The batch-mode part uses a GPGPU-friendly batch-mode clustering algorithm to partition $\mathbf{X}_t$ into micro-clusters. The incremental part merges these micro-clusters into C_{t-1} sequentially and obtains C_t if $t \geq 2$. The incremental part clusters the micro-clusters sequentially to obtain C_1 if $t = 1$.

Theorem 15. *Let $\text{Gra}_{ave,T}$ represent average evolving granularity of incAlgorithm up to step T. Serial Shrink Rate of incAlgorithm is negatively related to $\text{Gra}_{ave,T}$.*

Proof. Suppose incAlgorithm totally clustered N data points up to step T. Let $\text{mClu}_i \in \bigcup_{t=1}^{T} \mathbf{C}_{t,\text{new}}$ ($i = 1, 2, 3, \ldots, \text{micro}N_T$) be a micro-cluster, where $\text{micro}N_T = \bigcup_{t=1}^{T} \text{micro}_t$ and micor_t is the number of micro-clusters obtained by batch-mode part of step t. Larger $\text{Gra}_{\text{ave},T}$ means that each cluster in $\bigcup_{t=1}^{T} \mathbf{C}_{t,\text{new}}$ tends to contain more data points. Thus $\bigcup_{t=1}^{T} \text{micro}_t$ decreases. Pursuant to the above analysis and Definition 13, SSR declines when $\text{Gra}_{\text{ave},T}$ increases. □

Pursuant to Theorem 15 and Definition 13, parallelism of incAlgorithm is positively related to $\text{Gra}_{\text{ave},T}$ under measurement of work-depth.

3.5. Cause of the Accuracy-Parallelism Dilemma. incAlgorithm degrades to a point-wise incremental clustering algorithm if every micro-cluster produced by the batch-mode part of step t ($t = 1, 2, 3, \ldots$) contains and only contains one data point. In this case, $\text{Gra}_{\text{min},T}$, $\text{Gra}_{\text{max},T}$, and $\text{Gra}_{\text{ave},T}$ all reach the lowest bound. No different-to-same mis-affiliation occurs in batch-mode part and the clustering accuracy tends to rise. However, SSR = 1 and parallelism of incAlgorithm reach the lowest bound under measurement of work-depth.

Parallelism of incAlgorithm rises with the growth of $\text{Gra}_{\text{ave},T}$. Nevertheless, different-to-same mis-affiliation inevitably occurs. Whatever method the incremental part of step t ($t = 1, 2, 3, \ldots$) adopts to execute micro-cluster-level clustering, incAlgorithm cannot eliminate such mis-affiliations. Moreover, such mis-affiliations will exert negative influence on the operation of identifying homogeneous micro-clusters. Consequently, clustering accuracy drops. However, SSR decreases and parallelism rises.

4. Experiments

In this section, we validate Theorems 14 and 15 through a demo algorithm. Details of this demo algorithm can be found in our previous work [6]. Let demoAlgorithm represent the demo algorithm. The batch-mode part of demoAlgorithm uses mean-shift algorithm to generate micro-clusters. The incremental part of demoAlgorithm extends Self-organized Incremental Neural Network (SOINN) algorithm [7] to execute micro-cluster-level clustering. We validate the variation trends of accuracy and parallelism with respect to evolving granularity. Issues such as improving cluster accuracy of the demo algorithm and adaptively seeking balance between accuracy and parallelism are left as future work.

The benchmark algorithm of demoAlgorithm is batch-mode mean-shift algorithm. Accuracy metrics include the final cluster number, Peak Signal to Noise Ratio (PSNR), and Rand Index. In addition to Serial Shrink Ratio (SSR), we define parallel-friendly rate (PFR) to measure parallelism of demoAlgorithm.

In order to intuitively show the clustering results, images are used as input datasets. The clustering output is the segmented image. Evolving granularity is measured by bandwidth parameters of mean-shift algorithm. Bandwidth parameters are noted in the format of (feature bandwidth, spatial bandwidth).

4.1. Additional Performance Metrics

(1) Accuracy Metrics. Up to step $T0$, the *final cluster number* of demoAlgorithm is the quantity of clusters in $\mathbf{C}_{T0}$. Homogenous data points tend to fall into different clusters in the final result if the incremental clustering algorithm produces excessively more clusters than the benchmark algorithm, or vice versa. Consequently, demoAlgorithm tends to achieve

Table 1: Comparison of final cluster number.

	Demo algorithm			Benchmark algorithm		
	(2, 1)	(4, 2)	(6, 4)	(2, 1)	(4, 2)	(6, 4)
Boat	2372	1444	589	2517	1536	632
Cars	2348	1151	261	2481	1133	270
f16	1388	749	406	1469	827	356
Hill	2466	1552	525	2540	1623	540
Peppers	1715	845	385	1838	943	413
Sailboat	2208	1762	1051	2326	1817	1106
Stream	2558	2312	1524	2656	2437	1574
Tank	2374	1137	172	2425	1146	178
Truck	1631	986	339	1700	996	338
Trucks	2607	2256	681	2766	2360	693

higher accuracy if demoAlgorithm can resolve a closer final cluster number to that of the benchmark algorithm.

PSNR values of both incremental clustering result and benchmark result are calculated with respect to the original image.

(2) Parallelism Metrics. Assume that demoAlgorithm runs on a single-core processor. In step t ($t = 1, 2, 3, \ldots$), the batch-mode part consumes time $T_{1,t}$ and generates micro$_t$ micro-clusters; the incremental part consumes time $T_{2,t}$. Suppose demoAlgorithm totally clustered N data points up to step T_0. Up to step T_0, the *parallel-friendly rate* (PFR) of demoAlgorithm is PFR $= \sum_{t=1}^{T0} T_{1,t} / (\sum_{t=1}^{T0} (T_{1,t} + T_{2,t}))$. Larger PFR means superior parallelism.

4.2. Hardware and Software Environment. All experiments are executed on a platform equipped with an Intel Core2TM E7500 CPU, and a NVIDIA GTX 660 GPU. The main memory size of CPU is 4 GB. Linux of 2.6.18-194.el5 kernel is used with gcc 4.1.2 and CUDA5.0.

CUDA grid size is always set to 45. CUDA block size is set to 64. Double precision data type is used in floating point operations on both CPU and GPGPU. Batch-mode mean-shift is used as benchmark algorithm. Uniform kernel is used with mean-shift algorithm [33] for both batch-mode part of demoAlgorithm and benchmark algorithm.

4.3. Input Data Sets. We use grayscale images as input. One pixel corresponds to a input data point. The dimension of a data point (vector) is $d = 3$. Components of the 3-dimensional vector are X-coordinate of the pixel, Y-coordinate of the pixel, and grayscale value of the pixel, respectively. We selected ten natural scenery images (data set 1) and twenty aerial geographical images (data set 2) from USC SIPI data set [34]. Each image is evenly divided into 128×128 data blocks. These blocks are successively input to demoAlgorithm. Every image of data set 1 is incrementally clustered in CPU-only mode and GPGPU-powered mode, separately. Incrementally clustering an image of data set 2 in CPU-only mode is excessively time-consuming due to the large data volume. Consequently, the task of incrementally clustering these images is only executed in GPGPU-powered mode.

Table 2: Dataset 2: max and min Rand Index under ascending granularities.

	Granularity (measured by bandwidth)		
	(2, 1)	(4, 2)	(6, 4)
Max	0.9997 (usc2.2.05)	0.9983 (usc2.2.17)	0.9850 (usc2.2.17)
Min	0.8369 (usc2.2.02)	0.5025 (usc2.2.02)	0.1501 (usc2.2.07)

4.4. Experiment Results and Discussion. Bandwidth parameters (8, 6) and (10, 8) are excessively large for mean-shift algorithm (both incremental part of demoAlgorithm and benchmark algorithm). Under this parameter setting, accuracy of benchmark clustering results is excessively inferior. Consequently, it is not meaningful to compare accuracy between the incremental clustering algorithm and the benchmark. Thus, we omitted the experiment results under granularities (8, 6) and (10, 8) in Tables 1 and 2 and Figures 4, 5, and 6.

(1) Data Set 1: Natural Scenery Images. Table 1 shows the final cluster numbers resolved by the demo algorithm and the benchmark algorithm. Suppose we single out the cluster number values from column 2 (or 3, 4). Afterwards, we compute the average of this column of values. Let aveN_{inc} represent this average. Suppose we single out cluster number values out of column 5 (or 6, 7). Afterwards, we compute the average of this column of values. Let aveN_{bench} represent this average. For example, aveN_{inc} is $(2372 + 2348 + 1388 + 2466 + 1715 + 2208 + 2558 + 2374 + 1631 + 2607)/10 = 2166.7$ in terms of column 2. aveN_{bench} is $(2517 + 2481 + 1469 + 2540 + 1838 + 2326 + 2656 + 2425 + 1700 + 2766)/10 = 2271.8$ with regard to column 5.

Under the three ascending granularities, values of aveN_{inc}/aveN_{bench} are 0.9537, 0.9578, and 0.9726, respectively. The final cluster number of demo algorithm is in acceptable agreement with that of the benchmark algorithm.

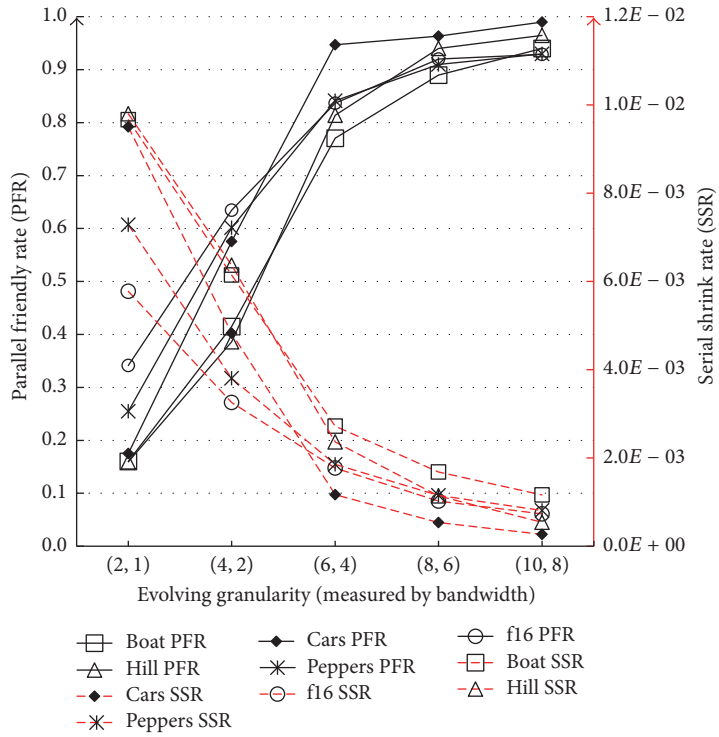


Figure 3: Data set 1: variation trends of PFR and SSR.

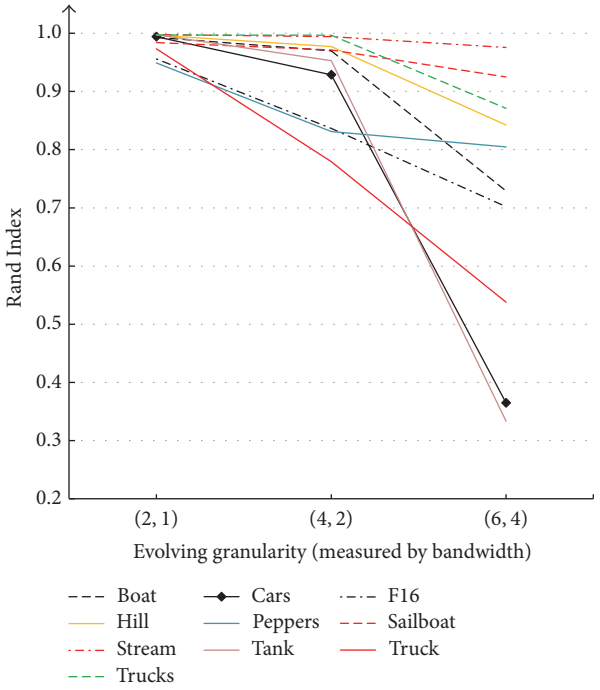


Figure 4: Data set 1: variation trends of Rand Index with respect to evolving granularity.

Figure 3 shows PFR and SSR of the first five inputs from data set 1. The figure reflects positive relation between evolving granularity and parallelism. We omitted the other images due to the fact that they show analogue trends. Figure 4 illustrates the negative relation between accuracy and evolving granularity.

We select granularity (4, 2) as a balance point between accuracy and parallelism based on experience. *truck* has the *lowest* Rand Index value on this balance point.

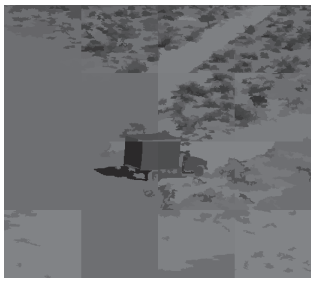
Figure 5 shows the original image and experiment results on *truck*. We can draw a similar conclusion from Figure 5 as that of Figures 3 and 4: speedup rises and accuracy degrades when evolving granularity increases. Moreover, Figure 5 shows the final cluster number values and PSNR values of both incremental clustering result and benchmark result, as well as the speedup. In addition to validating Theorems 14 and 15, our demo algorithm can produce acceptable-accuracy result on certain inputs such as *truck*.

(2) Data Set 2: Aerial Geographical Images. We continue to use symbols explained in part one of this subsection. Values of $(\text{ave}N_{\text{inc}}/\text{ave}N_{\text{bench}})$ under three ascending granularities are 0.9954, 1.1063, and 1.1345, respectively. Overall, the final cluster number of our demo algorithm is close to that of the benchmark algorithm.

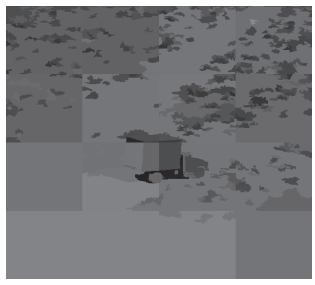
In order to avoid unnecessary details, we only list the maximum and minimum Rand Index values (and corresponding image names) under each evolving granularity in Table 2. Similarly, we only list the maximum and minimum SSR values in Table 3. These experiment results can still validate Theorems 14 and 15. We still choose granularity (4, 2) as a balance point between accuracy and parallelism based on experience. *usc*.2.02 has the *lowest* Rand Index value on granularity (4, 2).

Original input image

Granularity: (2, 1)
Rand Index = 0.9733
Final cluster number = 1631
Final cluster number (benchmark) = 1700
PSNR = 27.87 db
PSNR (benchmark) = 29.96 db
PFR = 0.3151 SSR = 0.0066
Speedup = 3.12



Granularity: (4, 2)
Rand Index = 0.7795
Final cluster number = 986
Final cluster number (benchmark) = 996
PSNR = 25.02 db
PSNR (benchmark) = 27.75 db
PFR = 0.6355 SSR = 0.0041
Speedup = 3.95



Granularity: (6, 4)
Rand Index = 0.5378
Final cluster number = 339
Final cluster number (benchmark) = 338
PSNR = 22.87db
PSNR (benchmark) = 24.14 db
PFR = 0.9017 SSR = 0.0015
Speedup = 4.93

Figure 5: *truck*: original image and incremental clustering results under three ascending granularity values.

Original input image

Granularity: (2, 1)
Rand Index = 0.8369
Final cluster number = 3799
Final cluster number (benchmark) = 3813
PSNR = 33.21 db
PSNR (benchmark) = 33.91 db
PFR = 0.3571 SSR = $3.79E-03$

Granularity: (4, 2)
Rand Index = 0.5025
Final cluster number = 2213
Final cluster number (benchmark) = 1926
PSNR = 31.58 db
PSNR (benchmark) = 30.96 db
PFR = 0.4804 SSR = $2.19E-03$

Granularity: (6, 4)
Rand Index = 0.4024
Final cluster number = 732
Final cluster number (benchmark) = 621
PSNR = 29.46 db
PSNR (benchmark) = 26.8 db
PFR = 0.7464 SSR = $7.93E-04$

Figure 6: *usc22.02*: original image and incremental clustering results under three ascending granularity values.

Table 3: Dataset 2: max and min SSR values under ascending granularities.

	Granularity (measured by bandwidth)				
	(2, 1)	(4, 2)	(6, 4)	(8, 6)	(10, 8)
Min	$3.7E-03$ (usc2.2.02)	$2.1E-03$ (usc2.2.02)	$6.3E-04$ (usc2.2.03)	$6.1E-04$ (usc2.2.03)	$1.2E-04$ (usc2.2.03)
Max	$1.0E-02$ (usc2.2.17)	$9.5E-03$ (usc2.2.17)	$6.2E-03$ (usc2.2.17)	$3.3E-03$ (usc2.2.01)	$2.1E-03$ (usc2.2.08)

Figure 6 shows the final cluster number values and PSNR values of both incremental clustering result and benchmark result. In addition to validating Theorems 14 and 15, our demo algorithm can produce acceptable-accuracy result on certain inputs such as *usc*.2.02.

5. Conclusion

In this paper we theoretically analyzed the cause of accuracy-parallelism dilemma with respect to the GPGPU-powered incremental clustering algorithm. Theoretical conclusions were validated by a demo algorithm.

Our future work will focus on identifying the suitable granularity for a given incremental clustering task and decreasing mis-affiliations through variable data-block-size.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors' sincere thanks go to Dr. Bo Hong (AT & T Corporation; School of Electrical and Computer Engineering, Georgia Institute of Technology, USA). This work is supported by the following foundations: Science and Technology Development Program of Weifang (2015GX008, 2014GX028), Doctoral Program of Weifang University (2016BS03), Fundamental Research Funds for the Central Universities (Project no. 3102016JKBJJGZ07), Colleges and Universities of Shandong Province Science and Technology Plan Projects (J13LN82), Natural Science Foundation of China (no. 61672433), and the Ph.D. Programs Foundation of Ministry of Education of China (no. 20126102110036).

References

[1] P. Wang, P. Zhang, C. Zhou, Z. Li, and H. Yang, "Hierarchical evolving Dirichlet processes for modeling nonlinear evolutionary traces in temporal data," *Data Mining and Knowledge Discovery*, vol. 31, no. 1, pp. 32–64, 2017.

[2] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[3] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, Springer, 2015.

[4] A. Ordoñez, H. Ordoñez, J. C. Corrales, C. Cobos, L. K. Wives, and L. H. Thom, "Grouping of business processes models based on an incremental clustering algorithm using fuzzy similarity and multimodal search," *Expert Systems with Applications*, vol. 67, pp. 163–177, 2017.

[5] C. Chen, D. Mu, H. Zhang, and B. Hong, "A GPU-accelerated approximate algorithm for incremental learning of Gaussian mixture model," in *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops (IPDPSW '12)*, pp. 1937–1943, Shanghai, China, May 2012.

[6] C. Chen, D. Mu, H. Zhang, and W. Hu, "Towards a moderate-granularity incremental clustering algorithm for GPU," in *Proceedings of the 2013 5th International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC '13)*, pp. 194–201, Beijing, China, October 2013.

[7] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Networks*, vol. 19, no. 1, pp. 90–106, 2006.

[8] S. Furao, T. Ogura, and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," *Neural Networks*, vol. 20, no. 8, pp. 893–903, 2007.

[9] J. Zheng, F. Shen, H. Fan, and J. Zhao, "An online incremental learning support vector machine for large-scale data," *Neural Computing and Applications*, vol. 22, no. 5, pp. 1023–1035, 2013.

[10] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking clusters in evolving data streams over sliding windows," *Knowledge and Information Systems*, vol. 15, no. 2, pp. 181–214, 2008.

[11] X. Zhang, C. Furtlehner, and M. Sebag, "Data streaming with affinity propagation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5212, no. 2, pp. 628–643, 2008.

[12] X. Zhang, C. Furtlehner, J. Perez, C. Germain-Renaud, and M. Sebag, "Toward autonomic grids: analyzing the job flow with affinity streaming," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '09)*, pp. 987–995, July 2009.

[13] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag, "Data stream clustering with affinity propagation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1644–1656, 2014.

[14] S. Lühr and M. Lazarescu, "Incremental clustering of dynamic data streams using connectivity based representative points," *Data and Knowledge Engineering*, vol. 68, no. 1, pp. 1–27, 2009.

[15] D. Yang, E. A. Rundensteiner, and M. O. Ward, "Neighbor-based pattern detection for windows over streaming data," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '09)*, pp. 529–540, March 2009.

[16] E. Lughofer, "A dynamic split-and-merge approach for evolving cluster models," *Evolving Systems*, vol. 3, no. 3, pp. 135–151, 2012.

[17] H. Yang and S. Fong, "Incrementally optimized decision tree for noisy big data," in *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (BigMine '12)*, pp. 36–44, Beijing, China, August 2012.

[18] X.-T. Yuan, B.-G. Hu, and R. He, "Agglomerative mean-shift clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 2, pp. 209–219, 2012.

[19] L.-Y. Wei and W.-C. Peng, "An incremental algorithm for clustering spatial data streams: exploring temporal locality," *Knowledge and Information Systems*, vol. 37, pp. 453–483, 2013.

[20] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Mining data streams under block evolution," *ACM SIGKDD Explorations Newsletter*, vol. 3, no. 2, pp. 1–10, 2002.

[21] M. Song and H. Wang, "Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering," in *Proceedings of the Defense and Security (SPIE '05)*, vol. 5803, p. 174, Orlando, Florida, USA, 2005.

[22] N. S. L. P. Kumar, S. Satoor, and I. Buck, "Fast parallel expectation maximization for gaussian mixture models on GPUs using

CUDA," in *Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications (HPCC '09)*, pp. 103–109, June 2009.

[23] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases (VLDB '03)*, 2003.

[24] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: Theory and practice," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 515–528, 2003.

[25] L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 3, article 12, 2009.

[26] M. Ackerman and S. Dasgupta, "Incremental clustering: the case for extra clusters," *Advances in Neural Information Processing Systems*, vol. 1, pp. 307–315, 2014.

[27] M. Ackerman and J. Moore, "When is Clustering Perturbation Robust?" *Computing Research Repository*, 2016.

[28] M. Ackerman and J. Moore, "Clustering faulty data: a formal analysis of perturbation robustness," 2016, https://arxiv.org/abs/1601.05900.

[29] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *Proceedings of the 24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN '16)*, pp. 357–368, April 2016.

[30] C. Chen, D. Mu, H. Zhang, and W. Hu, "Nonparametric incremental clustering: a moderate-grained algorithm," *Journal of Computational Information Systems*, vol. 10, no. 3, pp. 1183–1193, 2014.

[31] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[32] Y. Shiloach and U. Vishkin, "An O (n2 logn) parallel max-flow algorithm," *Journal of Algorithms*, vol. 2, pp. 1128–1146, 1982.

[33] M. Huang, L. Men, and C. Lai, "Accelerating mean shift segmentation algorithm on hybrid CPU/GPU platforms," in *In Proceedings of the International Workshop on Modern Accelerator Technologies for GIScience (MAT4GIScience '12)*, 2012.

[34] "USC-SIPI[DB/OL]," 2013, http://sipi.usc.edu/database/.