

Parallel Ant-Inspired CF-Tree Clustering for Scalable Collaborative Learning Group Formation

Malak Chniter

`maleek.chniter@ieee.org`

Faculty of Economics and Management of Sfax

Habib Kammoun

Faculty of Sciences of Sfax

Ilhem Kallel

ReGIM-Lab: Research Groups in Intelligent Machines

Slim Kanoun

École nationale d'Ingénieurs de Sfax

Research Article

Keywords: Ant-inspired clustering, CF-tree, collaborative learning, computational complexity, learning analytics, parallel computing, student group formation

Posted Date: October 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11427821/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Parallel Ant-Inspired CF-Tree Clustering for Scalable Collaborative Learning Group Formation

Malak Chniter^{1,2,5*}, Habib M. Kammoun^{2,3,5†}, Ilhem Kallel^{2,4,5*†},
Slim Kanoun^{4,5}

^{1*}FSEGS: Faculté des Sciences Economiques et de Gestion de Sfax.

²ReGIM-Lab: Research Groups in Intelligent Machines.

³FSS: Faculté des Sciences de Sfax.

⁴ENIS: Ecole Nationale d'Ingénieurs de Sfax.

⁵University of Sfax, Tunisia.

*Corresponding authors. E-mails: maleek.chniter@ieee.org;
ilhem.kallel@isims.usf.tn;

Contributing authors: habib.kammoun@fss.usf.tn; slim.kanoun@enis.tn;

[†]These authors contributed equally to this work.

Abstract

Forming learner groups from large and heterogeneous cohorts is an important challenge for technology-supported collaborative learning. Learning-analytics approaches can help instructors identify learner profiles from digital traces; however, group-recommendation methods must also scale to large volumes of learner data. This paper proposes a parallel ant-inspired clustering framework for scalable student group formation. The framework combines clustering-feature (CF) tree data summarization with a coordinator-executor architecture that distributes ant-inspired clustering tasks across multiple processors, thereby reducing the computational complexity associated with processing large learner datasets. It also parallelizes neighborhood-similarity calculations to reduce the computational cost of local comparisons. The method is evaluated on two synthetic datasets and on an Open University Learning Analytics Dataset-derived dataset comprising 29,741 learners represented by engagement, performance, and timing features. The evaluation examines intra- and inter-neighborhood similarity, processing time, speed-up, and parallel efficiency. Results show that CF-tree-based representation reduces processing time by approximately threefold compared with the conventional ant-inspired clustering procedure. Parallel execution further reduces runtime, while the resulting clusters retain cohesive neighborhood structures. The proposed framework provides a scalable computational basis for

learning-analytics-informed group recommendations. Its educational value should be validated further through studies that assess group composition, collaborative processes, and learning outcomes in authentic learning settings.

Keywords: Ant-inspired clustering, CF-tree, collaborative learning, computational complexity, learning analytics, parallel computing, student group formation

1 Introduction

Collaborative learning enables learners to exchange perspectives, articulate their reasoning, and develop knowledge through interaction with peers. In technology-supported learning environments, however, forming groups is not a trivial administrative task. Instructors must reconcile the pedagogical objective of an activity with heterogeneous learner characteristics, including prior performance, engagement, participation patterns, and practical constraints such as group size and class scale. Although groups can be created randomly, through learner self-selection, or by instructors, these approaches may be difficult to manage in large cohorts and may not consistently account for information relevant to a particular collaborative task [1, 2, 3]. Learning analytics can assist this process by transforming traces collected in virtual learning environments into learner profiles and candidate group recommendations.

proposed dynamic group formation based on feedback from prior collaboration [3], while Revelo Sánchez *et al.* investigated homogeneous group formation using personality traits and genetic algorithms [4]. More recently, Liang *et al.* studied group optimization in an active-reading context using learner-generated marker data, illustrating the growing interest in data-informed group composition and its relationship to collaborative learning processes [5].

Scalability remains a central challenge when learner data is large, multidimensional, and continuously generated by online learning platforms. Ant-inspired clustering algorithms offer a self-organizing mechanism for discovering group structures and have been investigated for building collaborative learning teams [2, 1]. However, conventional ant-inspired clustering can incur substantial computational cost because the processing load depends on the number of learner records, artificial ants, iterations, and neighborhood-similarity calculations. This limitation is important when a learning platform must process large cohorts or periodically update group recommendations. Prior work has analyzed this complexity in the context of student grouping and identified the need for more scalable ant-based clustering approaches [6].

This paper addresses this limitation through a parallel ant-inspired clustering framework for scalable student group formation. The framework uses a clustering-feature (CF) tree to summarize learner data as representative centroids before clustering. A coordinator-executor architecture partitions these centroids and distributes ant-inspired clustering tasks across multiple processors. Parallel neighborhood-similarity calculations further reduce the wall-clock cost of local comparisons. The framework is evaluated on two synthetic datasets and on an Open University Learning Analytics Dataset (OULAD)-derived dataset [7] containing 29,741 learner records represented

by engagement, performance, and timing features. Rather than replacing pedagogical judgment, the method is designed to provide a scalable learning-analytics component that generates candidate learner groups for instructor review.

The contributions of this paper are fourfold:

- We propose a parallel ant-inspired clustering framework that combines CF-tree data summarization, distributed task execution, and parallel neighborhood-similarity computation for large-scale learner data.
- We provide a computational-complexity analysis of the framework and examine how data representation and parallel execution affect its processing cost.
- We evaluate the method using two synthetic datasets and OULAD-derived learner data, reporting intra- and inter-neighborhood similarity, processing time, speed-up, and parallel-efficiency measures.
- We discuss how the framework can support learning-analytics-informed candidate-group recommendations for collaborative learning and future peer-tutoring activities, while identifying the pedagogical validation required in authentic learning settings.

To guide the study, we address the following research questions:

- **RQ1:** How can CF-tree data summarization and parallel execution reduce the computational cost of ant-inspired clustering for large-scale learner data?
- **RQ2:** To what extent does the proposed framework preserve intra-neighborhood cohesion and inter-neighborhood separation while improving processing time, speed-up, and parallel efficiency?
- **RQ3:** How can scalable learner-profile clustering provide candidate group recommendations for collaborative learning and future peer-tutoring scenarios?

The remainder of this paper is organized as follows. Section 2 reviews technology-supported student group formation and scalable ant-inspired clustering and defines the educational problem and learner-data representation. Section 3 presents the proposed parallel ant-inspired CF-tree framework and its computational-complexity analysis. Section 4 describes the experimental design, and Section 5 reports the findings. Section 6 discusses implications, practical deployment, and limitations. Finally, Section 7 concludes the paper.

2 Related Work

2.1 Technology-Supported Student Group Formation

The composition of learner groups is a central design decision in collaborative learning. Group formation can be performed randomly, through learner self-selection, or purposefully by instructors according to criteria such as prior knowledge, academic performance, learning behavior, interests, language, and other contextual characteristics. Each strategy entails trade-offs. Self-selection can strengthen learner agency and ownership of collaborative work; however, it may also lead to unbalanced groups, uneven participation, or the concentration of learners with similar working habits. Instructor-led group formation can incorporate pedagogical knowledge and contextual

information, but it becomes difficult to manage consistently when cohorts are large, learner data evolve throughout a course, or multiple activities require different grouping strategies [1, 2].

Technology-supported group formation aims to make these decisions more systematic by combining learner data with explicit pedagogical and organizational constraints. Depending on the learning activity, a system may recommend relatively homogeneous groups for targeted instruction, heterogeneous groups to stimulate the exchange of perspectives and distributed expertise, or complementary groups to support peer tutoring. The appropriateness of a grouping strategy therefore depends on the learning objective, the group size, the available learner information, and the instructor’s pedagogical intentions. Algorithmic group formation should consequently be considered as decision support for instructors rather than as an autonomous replacement for teacher judgment. The literature on group formation is multidisciplinary, bringing together computer-supported collaborative learning, learning analytics, learner modelling, and optimization. Cruz and Isotani provided a systematic mapping of group-formation algorithms in collaborative-learning contexts, identifying a growing body of work but also limited common ground for comparing algorithms and evaluating their educational benefits [8]. Odo, Masthoff, and Beacham subsequently reviewed group formation for collaborative learning and highlighted the diversity of learner characteristics, group-composition criteria, and computational techniques used in the field [9]. These reviews emphasize that the pedagogical purpose of a grouping strategy must be explicit and that algorithmic performance alone is insufficient to establish its educational value. Dynamic group formation is particularly relevant in online and blended learning settings, where learner characteristics, activity traces, and collaboration patterns may evolve over time. Srba and Bieliková proposed a dynamic group-formation approach for collaborative learning support that uses feedback from prior collaboration when forming subsequent groups [3]. Their study demonstrates the importance of connecting group formation to collaborative processes rather than treating it only as a static optimization problem. Revelo Sánchez *et al.* investigated homogeneous group formation using personality traits and genetic algorithms [4]. These works motivate group-formation methods that are computationally feasible, pedagogically configurable, and usable in teacher-facing learning technologies. Recent research has increasingly evaluated data-driven group formation through the lens of collaborative processes and learning outcomes. Liang *et al.* investigated algorithmic group composition in an active-reading setting using learner-generated marker data and a mixed genetic-algorithm strategy [5]. Their work illustrates that grouping methods may need to balance diversity of learner perspectives with common engagement in learning content. In collaborative programming, Ding *et al.* examined how prior knowledge and multiple dimensions of engagement, including behavioral, cognitive, emotional, and social engagement, relate to collaborative learning processes and group-level learning modes [10]. Their findings reinforce the importance of representing learner engagement as a multidimensional construct when designing technology-supported collaborative learning activities. Algorithmic support is also expanding from static group formation to adaptive peer-learning recommendations. Naik *et al.* proposed a hybrid method combining collaborative filtering with interaction-based clustering

to generate dynamic peer-learning recommendations in e-learning environments [11]. This direction is relevant to the present work because scalable learner-group recommendations may ultimately support both collaborative activities and peer-tutoring scenarios. Nevertheless, the current study focuses on efficiently generating candidate groups from learning behavior and engagement data. The assessment of peer-support quality, collaboration quality, and learning outcomes requires further investigation in authentic learning settings. Accordingly, the present work treats clustering as a learning-analytics component for producing candidate groups from learner engagement and performance traces. It does not assume that a mathematically cohesive cluster is automatically an effective learning group. The final composition of groups should remain configurable according to the learning activity and subject to teacher review, learner needs, and institutional constraints.

2.2 Scalable Ant-Inspired Clustering for Learning Analytics

Ant-inspired algorithms draw on collective behaviors of ant colonies, including foraging, pheromone-mediated communication, and self-organized sorting. Ant colony optimization has been applied to diverse problems and domains, including the traveling salesman problem [12], scheduling [13], network routing [14], traffic management [15], resource allocation [16], bioinformatics [17], and machine learning [18].

In traffic management, Kammoun et al. [15] proposed Adapt-Traf, which combines ant-inspired route selection with hierarchical fuzzy evaluation within a hierarchical multiagent architecture, illustrating the use of hybrid ant-inspired mechanisms for adaptive decision-making beyond clustering. Ant-inspired clustering, by contrast, draws on collective sorting behaviors, with early models developed for physical-object clustering and subsequently extended to numerical-data clustering [19, 20].

In these methods, artificial ants move within a grid, evaluate local neighborhoods, and probabilistically pick up or drop data objects. Through repeated local interactions, objects with similar characteristics tend to aggregate into emergent clusters. This self-organizing principle is attractive for learner-profile clustering because it can accommodate multidimensional learner representations without requiring the number of clusters to be specified in advance. Earlier work has explored ant-colony-inspired methods for constructing collaborative learning teams and for analyzing parameter sensitivity in student-grouping contexts [2, 21]. However, conventional ant-inspired clustering may become computationally expensive as the number of learner records, artificial ants, grid cells, iterations, and neighborhood-similarity calculations increases. Chniter *et al.* analyzed this issue in the context of student grouping and highlighted the need to reduce the computational cost of ant-colony clustering when learner data become large [6]. Three main strategies have been investigated to improve the scalability of ant-inspired clustering. First, *data representativeness* reduces redundant computations by representing large or high-dimensional datasets through samples, prototypes, centroids, or preliminary clusters. Examples include streamlined ant-based clustering, kernel-based preprocessing, sampling, and hybrid approaches combining ant-based search with other clustering mechanisms [22, 23, 24, 25, 26, 27, 28]. These approaches can reduce iteration costs and support the processing of large or noisy datasets. However, overly coarse representations may obscure fine-grained structures

and affect clustering quality. Second, *parallelization* distributes the computational work across several processing units. Qasem *et al.* used multi-core and GPU architectures to accelerate ant brood clustering while preserving clustering quality [29]. Parallel computation can substantially reduce wall-clock execution time, although its benefits may be limited by synchronization, communication, and load-balancing overheads. Third, *distributed computing* partitions data and clustering tasks across multiple workers or machines. MapReduce-based ant-clustering approaches and multi-ant-colony ensembles have been proposed to handle larger datasets [30, 31]. Distributed execution can increase the available computational capacity, but it introduces additional coordination and communication costs. The framework proposed in this paper combines data representativeness and parallelization. A clustering-feature (CF) tree summarizes learner data as representative centroids before the ant-inspired clustering process. A coordinator-executor architecture then partitions these centroids and distributes clustering tasks across multiple processors. Parallel neighborhood-similarity calculations further reduce the wall-clock cost of local comparisons. This combined design differs from methods relying only on data reduction or only on parallel execution: it examines the joint effect of representative learner-data structures and parallel processing on runtime and emergent clustering quality. Table 1 summarizes the principal scalability strategies identified in the literature and their relation to the proposed framework.

2.3 Collaborative Learning Group Formation Requirements

Forming learner groups is a pedagogical design task rather than only a clustering problem. Depending on the learning activity, instructors may seek relatively homogeneous groups for differentiated support, heterogeneous groups for collaborative problem solving, or complementary profiles for peer tutoring. In large online or blended courses, these decisions are difficult to make manually because learner information is heterogeneous, evolves during the course, and must be considered together with operational constraints such as group size and the number of available learners [9, 5, 10]. The proposed framework addresses the computational component of this task. It identifies learner profiles from learning behavior and engagement traces and provides candidate groups that can support subsequent instructor-led allocation. The clustering output is not assumed to be the final grouping policy. Instead, it can be interpreted according to the pedagogical purpose of an activity, as summarized in Table 2. This teacher-in-the-loop positioning allows an instructor to apply contextual knowledge, learner preferences, accessibility requirements, and activity-specific constraints that are not fully represented in platform data.

2.4 Learner Representation from Virtual Learning Environment Data

The real-world evaluation uses the Open University Learning Analytics Dataset (OULAD), which contains linked information about learner interactions with a virtual learning environment (VLE), assessment activity, registration, and learner context [7]. The learner profiles used in this study are constructed from the `studentVle`, `studentAssessment`, `studentRegistration`, and `studentInfo` tables. These records

Table 1: Scalability Strategies for Ant-Inspired Clustering and Their Relation to the Proposed Framework

Strategy	Potential benefits	Potential limitations	Role in the proposed framework
Data representativeness	Reduces redundant comparisons and iteration costs; summarizes high-dimensional or large datasets; can support outlier handling	May remove fine-grained structures or reduce clustering fidelity if the representation is too coarse	CF-tree leaf entries represent learner data through centroids before ant-inspired clustering
Parallelization	Reduces wall-clock runtime by distributing tasks across cores or processors; can accelerate local similarity computations	Synchronization, coordination, communication, and load-balancing overhead can limit performance gains	Coordinator-executor task distribution and parallel neighborhood-similarity calculations
Distributed computing	Increases available computational capacity and can support processing across multiple machines and large data volumes	Network communication, aggregation overhead, and operational complexity	The proposed architecture supports local multi-core execution and can be extended to cluster-mode deployment

Table 2: Illustrative Uses of Learner-Profile Clusters for Collaborative Learning

Objective	Desired composition	Use of clustering output
Targeted support	Relatively similar engagement or performance profiles	Use a cluster as a candidate group or divide a large cluster into smaller groups
Collaborative problem solving	Diverse or balanced learner profiles	Use cluster profiles to compose teams across clusters
Peer tutoring	Complementary learner profiles	Use membership and profile characteristics as inputs for tutor-tutee pairing or small-group allocation

are aggregated at the learner level to produce a dataset of 29,741 records represented by nine engagement, performance, and timing features. The feature set includes interaction indicators, such as total clicks, average daily clicks, and early engagement; assessment indicators, such as average score and number of submissions; and timing-related indicators, such as submission delay and registration-related status. These features provide an operational representation of learner activity patterns. They do

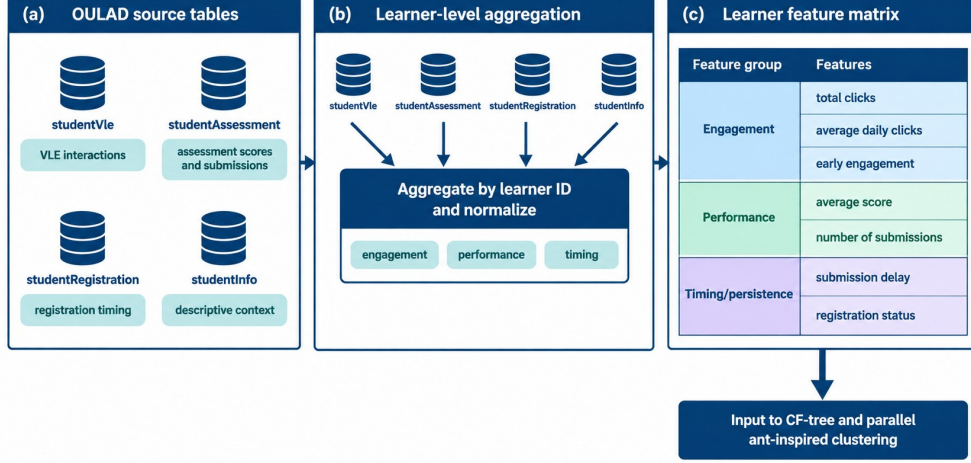


Fig. 1: Construction of learner profiles from OULAD data. The source tables provide virtual-learning-environment interaction, assessment, registration, and descriptive information. Records are aggregated by learner identifier and normalized to construct engagement, performance, and timing-related features used as input to the CF-tree and parallel ant-inspired clustering framework.

Table 3: Learner Features Used for Clustering

Category	Derived features
Engagement	Total clicks, average daily clicks, early engagement
Performance	Average assessment score, number of submissions
Timing and persistence	Submission delay, registration timing, registration-related status

not capture all characteristics relevant to group formation, including communication skills, motivation, availability, learner preferences, or interpersonal compatibility. Therefore, the framework is intended to inform instructor decisions rather than automate the final allocation of learners. As illustrated in Figure 1, information from the OULAD source tables is aggregated by learner identifier and normalized to construct a learner-feature matrix. This matrix captures engagement, performance, and timing patterns and provides the input to the CF-tree data summarization and the parallel ant-inspired clustering framework described in Section 3. Table 3 summarizes the learner-feature categories used in the clustering experiments. The exact preprocessing procedure, including missing-value handling, normalization, and exclusion criteria, is specified in Section 4.

3 Parallel Ant-Inspired CF-Tree Framework

This section presents the proposed framework for scalable clustering of learner profiles. The framework combines clustering-feature (CF) tree data summarization with parallel ant-inspired clustering. Its purpose is to reduce the amount of data processed by the ant-inspired procedure and to distribute the remaining clustering workload across multiple processing units. The output consists of emergent learner-profile clusters that can subsequently support instructor-facing group recommendations, as described in Section 6.

3.1 Framework Overview

The framework follows a coordinator-executor architecture, illustrated in Fig. 2. The coordinator reads the learner-feature matrix, constructs a CF-tree, extracts representative centroids from its leaf entries, partitions the centroids, and coordinates the execution of ant-inspired clustering tasks. Each executor processes one partition of centroids and returns its local clustering result to the coordinator, which collects and aggregates the outputs.

The architecture supports multi-core execution in local mode and can be extended to cluster-mode deployment when resources are available. In the current implementation, Apache Spark is used to distribute partitions to executors. A thread pool can additionally be used within an executor to compute neighborhood similarities concurrently when the neighborhood size makes this operation a bottleneck. This two-level design separates inter-partition parallelism from local similarity computations.

3.2 CF-Tree-Based Learner Data Representation

A CF-tree is used to summarize the learner-feature matrix before the ant-inspired clustering stage. Each clustering feature is represented by

$$CF = (n, LS, SS), \quad (1)$$

where n denotes the number of observations represented by the feature, LS is the linear sum of the observations, and SS is their squared sum. The CF-tree organizes these summary statistics hierarchically and represents each leaf entry by a centroid. For leaf entry i , the centroid is calculated as

$$C_i = \frac{LS_i}{n_i}. \quad (2)$$

Let n denote the number of learner records and C the number of centroids extracted from the CF-tree, where typically $C \ll n$. Instead of operating directly on all learner records, the ant-inspired clustering stage operates on the C representative centroids. This reduces the number of objects involved in grid initialization, neighborhood comparison, and pickup-drop operations. The level of data compression is controlled by the CF-tree configuration, including the branching factor and the diameter threshold.

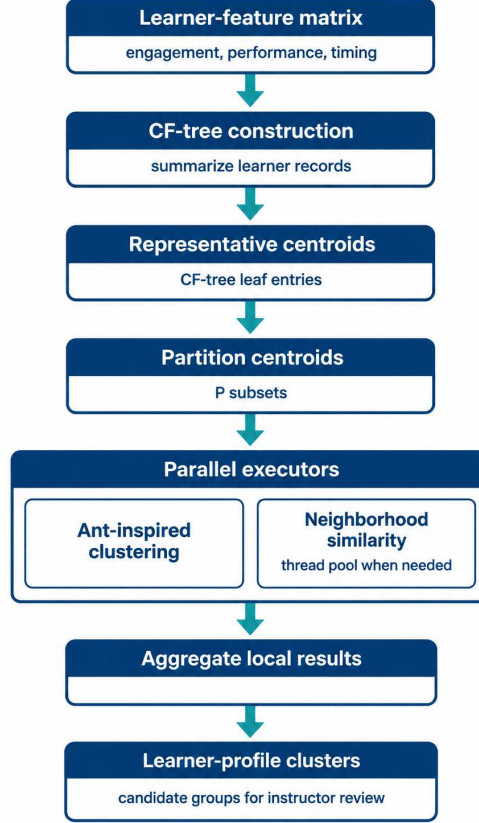


Fig. 2: Architecture of the proposed parallel ant-inspired CF-tree framework. The coordinator summarizes the learner-feature matrix using a CF-tree, extracts representative centroids, partitions them across executors, and aggregates local clustering results. Each executor runs ant-inspired clustering, with parallel neighborhood-similarity calculations when appropriate.

These settings must preserve sufficient information about learner-profile structures while reducing redundant computational work.

3.3 Parallel Ant-Inspired Clustering

The coordinator partitions the centroids into P subsets and distributes them among executors. Each executor initializes a two-dimensional grid, scatters its assigned centroids, initializes artificial ants, and performs the local ant-inspired clustering procedure. Following the adaptive parameterization used in the original implementation, the number of ants and grid size are defined as functions of the number of data objects processed by an executor:

$$N_{\text{ants}} = \frac{n}{10}, \quad (3)$$

$$N_{\text{cells}} = 10n, \quad \text{GridSize} = \sqrt{10n}. \quad (4)$$

For scalability, n in Eqs. (3)–(4) denotes the number of representative centroids assigned to the executor after CF-tree summarization and partitioning. This avoids sizing the grid and ant population from the complete original learner dataset.

At each iteration, an ant evaluates the neighborhood of the carried object or the object located at its current grid position. The pickup and drop probabilities are computed as

$$P_{\text{pick}} = \frac{k_{\text{pick}}}{k_{\text{pick}} + F_1(C_i)}, \quad (5)$$

$$P_{\text{drop}} = \frac{F_1(C_i)}{k_{\text{drop}} + F_1(C_i)}, \quad (6)$$

where k_{pick} and k_{drop} are positive constants controlling pickup and drop behavior, respectively, and $F_1(C_i)$ is the local similarity of centroid C_i . The similarity function is

$$F_1(C_i) = \frac{1}{\sigma} \sum_{C_j \in \text{Neigh}(C_i)} [1 - d_{\text{norm}}(C_i, C_j)], \quad (7)$$

where σ is the neighborhood size, $\text{Neigh}(C_i)$ denotes the set of neighboring centroids, and $d_{\text{norm}}(C_i, C_j)$ is the normalized Euclidean distance between centroids C_i and C_j . A high local similarity lowers the pickup probability and increases the drop probability, thereby promoting the aggregation of similar learner-profile representatives.

When a neighborhood contains many objects, the executor submits local similarity computations to a thread pool. The resulting similarities are collected and averaged before evaluating Eqs. (5) and (6). Algorithm 1 summarizes the procedure.

3.4 Computational Complexity Analysis

The computational cost of the framework can be decomposed into CF-tree construction, centroid extraction, partitioning, local ant-inspired clustering, and result aggregation. Building the CF-tree requires $O(nB)$ operations, where B denotes the branching factor and $B \ll n$ in the intended application. Extracting C centroids from the leaf entries requires $O(C)$ operations, while partitioning them into P subsets requires $O(C)$ operations.

For an executor processing approximately C/P centroids, the local ant-inspired procedure has an approximate work complexity of

$$O\left(I_{\text{max}} \cdot N_{\text{ants}} \cdot \frac{C}{P} \cdot q\right), \quad (8)$$

where q denotes the average cost of evaluating a local neighborhood. If all neighborhoods contain at most σ objects and similarity tasks are processed with T threads, the wall-clock cost of a neighborhood calculation is approximately $O(\sigma/T)$, excluding task-management overhead. The total parallel wall-clock time also depends on data partitioning, synchronization, executor startup, and result aggregation. Consequently, parallel execution primarily reduces observed runtime; it does not eliminate the underlying computational work or communication overhead.

Algorithm 1 Parallel Ant-Inspired CF-Tree Clustering

Require: Learner-feature matrix X ; CF-tree parameters; number of partitions P ; maximum iterations $I_{\max}$; neighborhood size σ

Ensure: Aggregated learner-profile clusters

```
1: Construct a CF-tree from  $X$  and extract leaf-entry centroids  $C$ 
2: Partition  $C$  into  $P$  subsets and assign one subset to each executor
3: for all executors in parallel do
4:   Initialize a grid and scatter the assigned centroids
5:   Initialize  $N_{\text{ants}}$  ants using Eq. (3)
6:   for  $I = 1$  to  $I_{\max}$  do
7:     for all ants do
8:       Identify the current centroid and its neighborhood
9:       Compute  $F_1$  using sequential or thread-pool similarity tasks
10:      if the ant carries a centroid then
11:        Drop it with probability  $P_{\text{drop}}$  in Eq. (6)
12:      else
13:        Pick up the current centroid with probability  $P_{\text{pick}}$ 
14:        in Eq. (5)
15:      end if
16:      Move the ant to a random neighboring grid location
17:    end for
18:  end for
19:  Return the local clustering result
20: end for
21: Collect and aggregate local results from all executors
22: return Aggregated learner-profile clusters
```

Compared with conventional ant-inspired clustering over all n learner records, the proposed framework reduces the effective problem size to C representative centroids and distributes centroid subsets across P executors. Its expected runtime benefit is therefore driven by both the data-representation ratio n/C and the effective parallelism achieved by the computing environment. Section 4 describes the datasets, configuration, and evaluation measures used to empirically assess these effects, while Section 5 reports the resulting clustering quality and scalability findings.

4 Experimental Design

4.1 Datasets

The proposed framework is evaluated using two synthetic datasets and one real-world learner dataset. The synthetic datasets are used to isolate the effect of CF-tree data representation and parallel execution under controlled dimensionality and cluster settings. The real-world dataset is derived from the Open University Learning Analytics Dataset (OULAD) and evaluates the feasibility of processing large-scale learner

Table 4: Datasets Used in the Experiments

Dataset	Records	Features	Reference clusters
Synthetic 1	9,000	50	20
Synthetic 2	10,000	55	22
OULAD-derived learner data	29,741	9	–

data for group recommendation [7]. Table 4 summarizes the datasets. The first synthetic dataset contains 9,000 instances with 50 features and 20 reference clusters. The second contains 10,000 instances with 55 features and 22 reference clusters. The OULAD-derived dataset contains 29,741 learner records represented by nine engagement, performance, and timing features, as described in Section 2.4. The synthetic datasets are used only for computational benchmarking; their cluster labels are not used to train the proposed unsupervised method.

4.2 Computing Environment and Baselines

Experiments were performed on a high-performance virtual server at the University of Sfax, Tunisia, equipped with an Intel[®] Xeon Gold 5218 processor running at 2.30 GHz, 24 CPU cores, and Linux. The framework was implemented using Apache Spark to distribute centroid partitions across executors. Within an executor, a thread pool was used for neighborhood-similarity calculations when the neighborhood size justified additional local parallelism. The principal baseline is the conventional sequential ant-inspired clustering algorithm operating without CF-tree summarization or distributed execution. The proposed framework is assessed in two stages: first, the effect of CF-tree data representation is examined by comparing the conventional method with the CF-tree-based sequential variant; second, the impact of parallel execution is assessed by varying the number of processing cores. This design makes it possible to distinguish the contribution of data summarization from that of parallel processing. The number of artificial ants and the grid size are set adaptively according to the number of objects processed by each executor, as specified in Eqs. (3)–(4). The clustering parameters include the neighborhood size σ , pickup parameter k_{pick} , drop parameter k_{drop} , CF-tree diameter threshold, and branching factor. To support reproducibility, the final version of the paper will report the exact values of these parameters, the number of independent runs, the random-seed policy, Spark version, Python version, memory configuration, and the numbers of cores evaluated.

4.3 Evaluation Metrics

The evaluation considers clustering-structure and computational-performance measures. The clustering-structure analysis uses intra-neighborhood similarity and inter-neighborhood similarity [32]. Intra-neighborhood similarity indicates the cohesion of

local learner-profile structures, defined as

$$S_{intra} = \frac{1}{N} \sum_{C_j \in Neigh(C_i)} F_1(C_i) \quad (9)$$

whereas inter-neighborhood similarity characterizes the distinction between neighboring structures, defined as

$$F_2(Neigh(p_j), Neigh(p_i)) = \frac{1}{|Neigh(p_j)| \cdot |Neigh(p_i)|} \sum_{C_j \in Neigh(p_j)} \sum_{C_i \in Neigh(p_i)} (1 - d_{normalised}(C_j, C_i)) \quad (10)$$

$$S_{inter} = \frac{1}{\frac{N_p(N_p-1)}{2}} \sum_{p_j \neq p_i} F_2(Neigh(p_j), Neigh(p_i)) \quad (11)$$

with $Neigh(p_j)$ and $Neigh(p_i)$ are representing the neighborhoods of two positions p_j and p_i , C_j and C_i are two centroids at two different positions p_j and p_i . Then, S_{inter} is the average of the inter-neighbourhood similarity equation with N_p representing the total number of positions. These internal measures are appropriate for evaluating emergent clusters when external ground-truth group labels are not available for the OULAD learner data. Computational performance is evaluated through processing time, speed-up, and parallel efficiency. For p processing cores, speed-up is defined as

$$S(p) = \frac{T(1)}{T(p)}, \quad (12)$$

where $T(1)$ is the execution time using one core and $T(p)$ is the execution time using p cores. Parallel efficiency is defined as

$$E(p) = \frac{S(p)}{p}. \quad (13)$$

The experiments therefore evaluate whether the CF-tree and parallel architecture reduce wall-clock runtime while preserving cohesive and distinguishable emergent clusters. Because the OULAD evaluation does not include observed collaboration processes or learning-outcome data after group assignment, the reported measures demonstrate computational feasibility and learner-profile clustering quality rather than the causal educational effectiveness of the recommended groups.

5 Results

This section reports the effects of CF-tree data representation and parallel execution on clustering structure and computational performance. The analysis is organized according to the research questions. First, the conventional ant-inspired clustering procedure is compared with its CF-tree-based sequential variant to isolate the effect

Table 5: Summary of Experimental Comparisons and Reported Measures

Experiment	Compared configurations	Reported measures	Purpose
CF-tree representation	Conventional ant-inspired clustering versus CF-tree-based sequential clustering	Processing time, intra-neighborhood similarity, inter-neighborhood similarity	Isolate the effect of representative centroids before parallel execution
Synthetic parallel scalability	CF-tree-based clustering under varying numbers of CPU cores	Processing time, speed-up, parallel efficiency, intra-neighborhood similarity, inter-neighborhood similarity	Evaluate the runtime benefit and quality stability of parallel execution
OULAD-derived learner data	CF-tree-based clustering under varying numbers of CPU cores	Processing time, speed-up, parallel efficiency, intra-neighborhood similarity, inter-neighborhood similarity	Assess computational feasibility on learner engagement, performance, and timing data

of data summarization. Second, the scalability of the parallel framework is examined on the synthetic datasets. Third, the framework is evaluated on the OULAD-derived learner dataset. Table 5 summarizes the measures reported in Figs. 3–5.

5.1 Effect of CF-Tree Data Representation

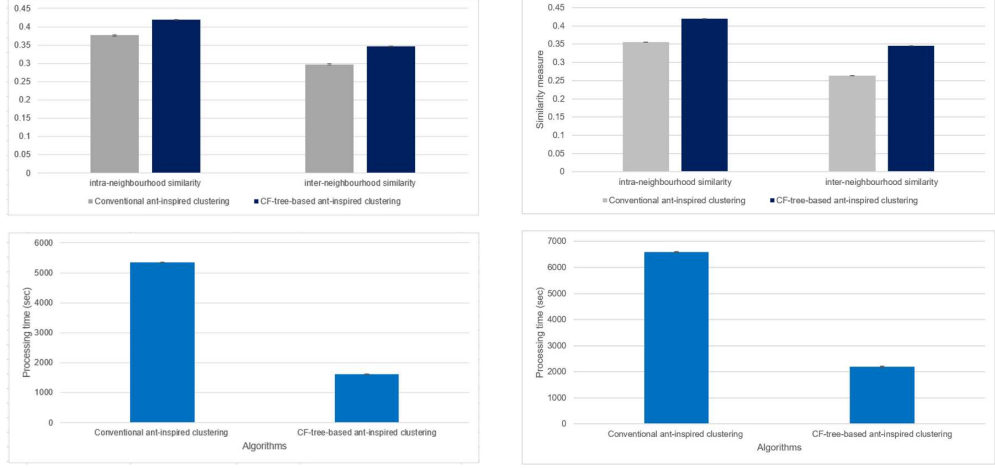
Figure 3 compares the conventional ant-inspired clustering procedure with the CF-tree-based sequential variant on the two synthetic datasets. As summarized in Table 5, this comparison isolates the effect of representing the original objects through CF-tree leaf-entry centroids before applying the ant-inspired procedure. In both datasets, the CF-tree-based variant substantially reduces processing time relative to conventional ant-inspired clustering. The observed reduction is approximately threefold, as shown in Fig. 3. This result is consistent with the intended effect of CF-tree summarization: the ant-inspired procedure initializes its grid, evaluates local neighborhoods, and performs pickup-drop operations on representative centroids rather than on every original observation.

The runtime improvement does not coincide with a visible deterioration in the reported neighborhood-similarity measures. Figure 3 shows that the CF-tree-based variant preserves intra-neighborhood similarity values comparable to those of the conventional procedure, while maintaining lower inter-neighborhood similarity. Thus, the representative centroids retain sufficient information for the algorithm to form cohesive and distinguishable local structures under the selected internal measures.

These results answer RQ1: CF-tree data summarization reduces the effective number of objects processed by ant-inspired clustering and therefore reduces its processing time, while preserving the internal neighborhood structure assessed in this study.

5.2 Parallel Scalability on Synthetic Datasets

Figure 4 presents the results of the parallel CF-tree-based framework on the two synthetic datasets under varying numbers of CPU cores. In accordance with the comparison summarized in Table 5, the figure reports intra- and inter-neighborhood



(a) Dataset: $n = 9,000$, $m = 50$, and $c = 20$; (b) Dataset: $n = 10,000$, $m = 55$, and $c = 22$

Fig. 3: Comparison of conventional ant-inspired clustering and the CF-tree-based sequential variant on the synthetic datasets: (a) $n = 9,000$, $m = 50$, and $c = 20$; and (b) $n = 10,000$, $m = 55$, and $c = 22$. CF-tree representation reduces processing time while preserving cohesive and distinguishable neighborhood structures.

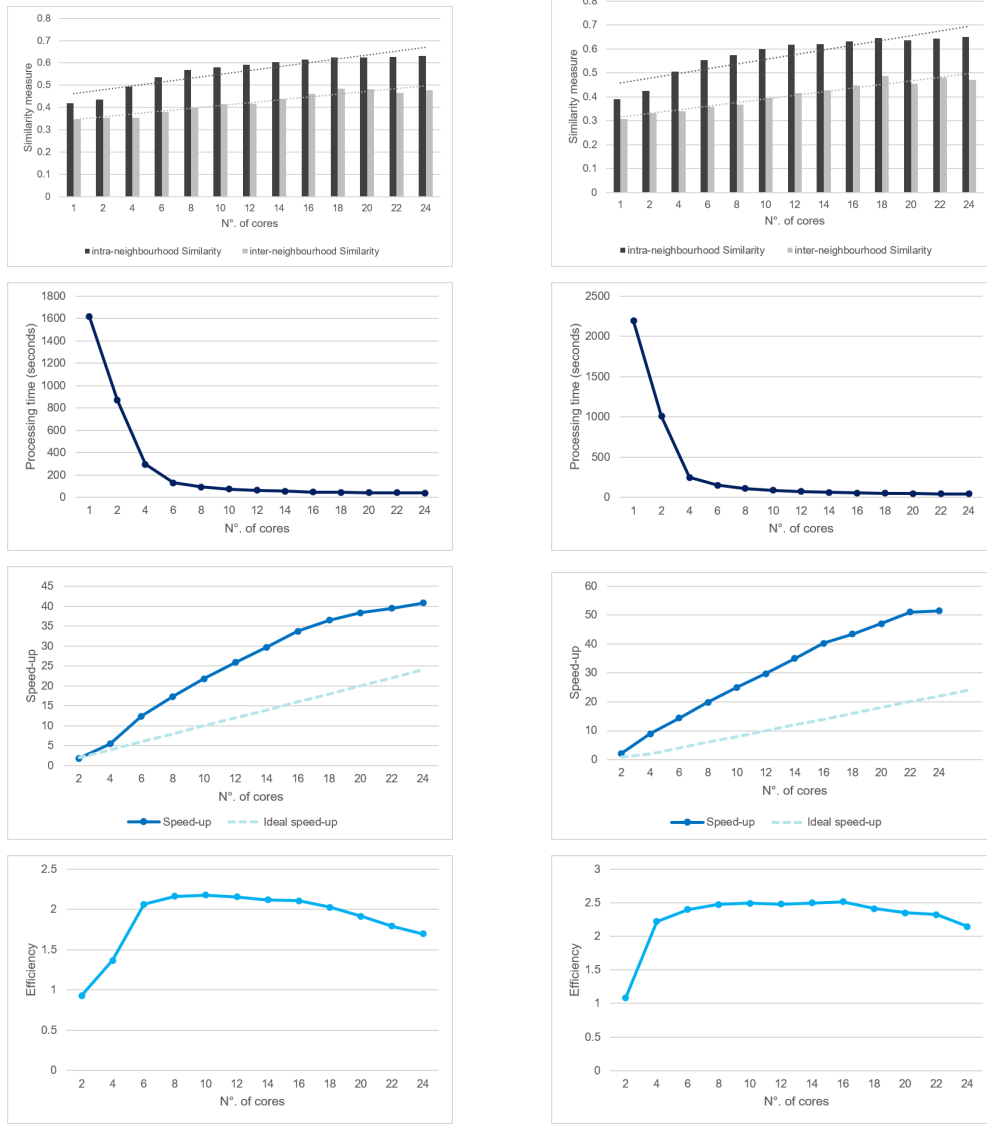
similarity, processing time, speed-up, and parallel efficiency for each evaluated configuration.

For both datasets, processing time decreases when additional CPU cores are used. The corresponding speed-up curves in Fig. 4 confirm that distributing CF-tree centroids among executors reduces the wall-clock cost of the ant-inspired clustering stage. The results therefore show that the benefit obtained from data summarization can be extended through parallel execution.

The relationship between the number of cores and performance is not perfectly linear. As shown by the parallel-efficiency curves in Fig. 4, efficiency decreases as more cores are allocated. This behavior reflects the proportion of the execution that remains sequential or incurs coordination overhead, including centroid partitioning, task scheduling, executor coordination, local synchronization, and result aggregation. Therefore, the largest available core configuration is not necessarily the most resource-efficient configuration.

The internal similarity measures remain broadly stable across the parallel configurations. Intra-neighborhood similarity remains higher than inter-neighborhood similarity in Fig. 4, indicating that the parallel framework continues to form cohesive local neighborhoods while maintaining separation between different structures. The computational gains are consequently not explained by a clear degradation of the reported internal clustering structure.

Together, these results address RQ2. The combined CF-tree and parallel architecture reduces runtime and produces measurable speed-up while preserving the relationship between intra- and inter-neighborhood similarity. At the same time, the



(a) Dataset: $n = 9,000$, $m = 50$, and $c = 20$; (b) Dataset: $n = 10,000$, $m = 55$, and $c = 22$

Fig. 4: Scalability of the proposed CF-tree-based parallel ant-inspired clustering framework on synthetic datasets under varying numbers of CPU cores. The panels report intra- and inter-neighborhood similarity, processing time, speed-up, and parallel efficiency for Synthetic 1 and Synthetic 2.

efficiency curves demonstrate the practical limitation imposed by coordination and other nonparallel costs.

5.3 Results on OULAD-Derived Learner Data

Figure 5 reports the performance of the proposed framework on the OULAD-derived learner dataset, which contains 29,741 learner records represented by engagement, performance, and timing features. As specified in Table 5, this experiment evaluates whether the framework retains its computational benefits when applied to learner data rather than controlled synthetic data.

The processing-time results in Fig. 5 show that the framework can process the learner dataset using CF-tree representation and multi-core parallel execution. Increasing the number of CPU cores reduces processing time and increases speed-up relative to the single-core configuration. This result demonstrates that the architecture can provide a practical computational basis for large-scale learner-profile clustering.

The parallel-efficiency results also show diminishing returns as more cores are added. CF-tree construction, centroid extraction, partition management, executor startup, local task management, and aggregation remain sources of sequential or coordination-related overhead. The results therefore suggest that a moderate allocation of parallel resources may offer a more favorable balance between runtime reduction and resource use than the largest available core configuration.

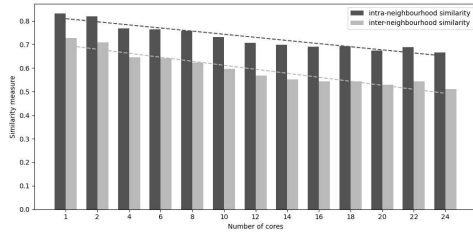
Across the evaluated configurations, intra-neighborhood similarity remains higher than inter-neighborhood similarity, as shown in Fig. 5. This pattern indicates that parallel execution does not eliminate the local cohesion of the learner-profile structures produced by the ant-inspired procedure. It is evidence of internal clustering coherence under the selected metrics, not evidence that the clusters automatically constitute pedagogically optimal learner groups.

These findings address RQ3 at the computational level. The proposed framework can generate learner-profile clusters from large-scale learning-platform data within a reduced processing time, enabling its use as a candidate-generation component for instructor-reviewed group formation. The experiment does not assess group allocation, collaboration quality, peer-tutoring interactions, learner satisfaction, or learning outcomes; those educational outcomes require teacher-in-the-loop and classroom-based validation.

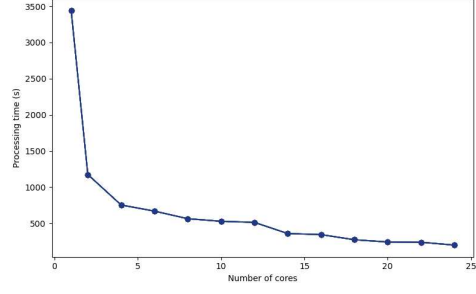
5.4 Summary of Findings

The results across Figs. 3, 4, and 5 provide three principal findings. First, CF-tree summarization reduces the volume of data handled by the ant-inspired procedure and produces an approximately threefold reduction in processing time on the synthetic datasets. Second, parallel execution further reduces wall-clock time and produces speed-up, although the corresponding efficiency is constrained by coordination and other nonparallel overheads. Third, the intra- and inter-neighborhood similarity measures remain consistent with cohesive and distinguishable emergent structures after CF-tree representation and across the evaluated parallel configurations.

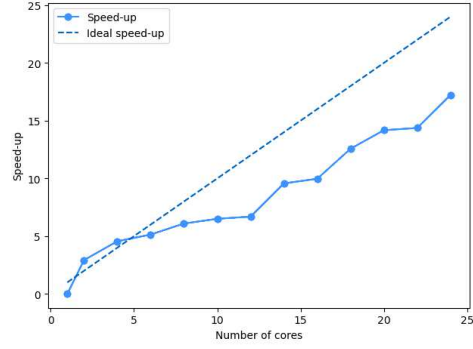
The proposed framework therefore improves the computational feasibility of ant-inspired learner-profile clustering for large cohorts. The next section discusses the implications for instructor-facing group recommendations, practical deployment in



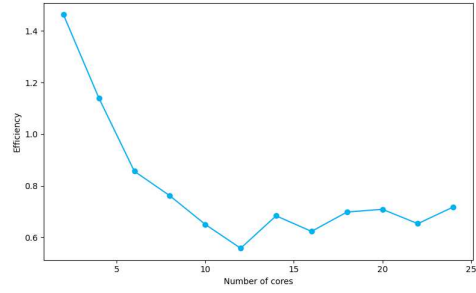
(a) intra- and inter-neighborhood similarity



(b) processing time



(c) speed-up



(d) parallel efficiency

Fig. 5: Results of the proposed CF-tree-based parallel ant-inspired clustering framework on the OULAD-derived learner dataset ($n = 29,741$, $m = 9$) under varying numbers of CPU cores: (a) intra- and inter-neighborhood similarity, (b) processing time, (c) speed-up, and (d) parallel efficiency.

learning platforms, and the pedagogical validation required before using the resulting clusters as final learner groups.

6 Discussion

The results show that the proposed framework improves the computational feasibility of ant-inspired learner-profile clustering by combining CF-tree data summarization with parallel execution. As summarized in Table 5, CF-tree representation reduces the number of objects processed during the ant-inspired stage, whereas the coordinator-executor architecture reduces the wall-clock time required to process the resulting representative centroids. The results in Figs. 3–5 therefore support the use of the framework as a scalable learning-analytics component for generating candidate learner profiles in large online and blended learning settings. The results do not, however, establish that clusters obtained from learner traces automatically constitute effective collaborative learning groups. This distinction is central to the interpretation of the present findings.

6.1 Implications for Learning-Analytics-Supported Group Formation

The computational gains observed in Fig. 3 are relevant when learner data are large or when recommendations must be updated during a course. Representing learner records by CF-tree centroids reduces the workload associated with grid initialization, local similarity calculations, and the pickup-drop behavior of artificial ants. The approximately threefold reduction in processing time on the synthetic datasets suggests that this data-representation strategy can make ant-inspired clustering more practical for large learner cohorts without producing a clear loss of the internal neighborhood structure assessed in the experiments.

The experiment on the OULAD-derived dataset further demonstrates that the approach can process learner profiles built from engagement, performance, and timing features. As shown in Fig. 5, the framework processes 29,741 learner records while maintaining a higher intra-neighborhood similarity than inter-neighborhood similarity across the evaluated processor configurations. In this context, clusters should be interpreted as learner-profile structures that summarize patterns in available platform traces. They can assist instructors in understanding variation within a cohort and in generating candidate pools for subsequent group formation.

The appropriate use of a learner-profile cluster depends on the pedagogical purpose of the activity. Table 2 illustrates how the same clustering output may support different decisions. For targeted support, an instructor may use a relatively homogeneous cluster as a candidate group or divide a large cluster into smaller groups. For collaborative problem solving, cluster profiles may help an instructor compose balanced teams across clusters. For peer tutoring, membership and profile characteristics may be used as one input among others for pairing learners with complementary strengths and needs. The framework does not prescribe one universally optimal grouping strategy; rather, it produces computationally tractable information that can support context-dependent pedagogical decisions.

The teacher-in-the-loop role is therefore necessary rather than incidental. An instructor can interpret the learner-profile structures in light of information that is often unavailable in platform logs, including the learning objective, task difficulty,

learner preferences, availability, communication needs, language, accessibility requirements, social relationships, and prior collaboration experience. Teacher review also allows the grouping process to account for institutional constraints, such as the required group size and the number of groups, which are not optimization objectives of the current clustering procedure.

6.2 Scalability and Deployment Considerations

The synthetic and learner-data experiments demonstrate that parallel execution reduces processing time, but they also show that performance gains are constrained by overhead. The speed-up and parallel-efficiency curves in Figs. 4 and 5 indicate that additional cores do not produce proportional reductions in runtime for all configurations. This behavior is expected because CF-tree construction, centroid extraction, partition management, executor startup, task scheduling, local synchronization, and aggregation are not completely parallelized.

For practical deployment, the results imply that allocating the maximum number of available cores is not always the best operational choice. A moderate configuration can provide a useful compromise between response time and resource consumption, particularly when the framework is deployed as part of a learning platform shared by multiple courses or services. The appropriate configuration should be selected empirically for the expected number of learners, the dimensionality of the learner features, the CF-tree compression level, and the frequency with which recommendations are regenerated.

Figure 5 also supports a workflow in which learner profiles are periodically refreshed as new learning traces become available. For example, a platform may aggregate interaction and assessment data after a learning unit, update the learner-feature matrix, and regenerate candidate profile clusters before a new collaborative activity. Such periodic analysis can support adaptive recommendations, but it should not lead to automatic or excessively frequent regrouping. Repeatedly changing groups may disrupt social cohesion, learner continuity, and the development of shared group practices. The timing of regrouping should therefore be determined by the instructional design of the course and reviewed by the instructor.

The current architecture is evaluated in a multi-core execution setting. Its coordinator-executor design may also support future deployment across distributed resources; however, such a deployment should be evaluated separately. Network communication, fault tolerance, data transfer, and result-merging costs may become more significant when executors operate on different machines. The present evidence therefore supports parallel multi-core scalability, while distributed and cloud-based performance remain directions for future work.

6.3 Limitations and Future Evaluation

The first limitation concerns the interpretation of clustering quality. The experiments use intra-neighborhood similarity and inter-neighborhood similarity to assess the internal structure produced by the algorithm. As reported in Figs. 3–5, these measures

indicate that local neighborhood cohesion is retained while the framework reduces runtime. They do not, however, show whether learners in a cluster collaborate effectively, experience equitable participation, provide helpful peer support, or achieve stronger learning outcomes. The results should thus be interpreted as evidence of computational feasibility and internal learner-profile clustering structure, not as causal evidence of educational effectiveness.

Second, the OULAD-derived feature set is necessarily partial. The learner-feature matrix includes engagement, performance, and timing-related indicators, as summarized in Table 3. It does not include all factors that may affect collaborative group quality, such as communication skills, motivation, collaboration preferences, language, scheduling availability, task-specific expertise, interpersonal relationships, or accessibility needs. In addition, the substantive interpretation of a cluster depends on preprocessing choices, feature engineering, normalization, CF-tree parameters, ant-clustering parameters, and the selected computational environment.

Third, algorithmic group recommendations require careful governance. Learning-platform data can be incomplete, unevenly distributed, or influenced by unequal access to learning resources. A system that relies on such data without transparency or oversight may reproduce or amplify existing inequalities. Any practical implementation should therefore apply data-minimization principles, document the features and grouping criteria used, provide instructors with understandable explanations of the recommendations, enable appropriate learner and teacher oversight, and evaluate whether recommendation quality differs across relevant learner populations.

Future work should evaluate the framework through teacher-in-the-loop and classroom-based studies. These studies should compare algorithmically supported recommendations with random, self-selected, and instructor-created groups. Beyond computational measures, the evaluation should assess group-size compliance, profile diversity or complementarity, group stability, collaboration quality, participation equity, learner satisfaction, peer-support interactions, and learning outcomes. Such studies should also examine how instructors interpret candidate profiles and which interface features help them adjust recommendations in accordance with their pedagogical goals.

6.4 Discussion Summary

The proposed framework addresses an important computational barrier to using ant-inspired clustering with large learner datasets. The results in Figs. 3–5 show that CF-tree representation reduces the effective clustering workload and that parallel execution further reduces wall-clock time, although parallel efficiency is limited by coordination overhead. These results support the use of the framework as a scalable candidate-generation component for learning-analytics-informed group formation.

Its educational value depends on how the resulting learner-profile clusters are interpreted and used. The framework should support, not replace, instructor judgment. The next section concludes the paper by summarizing the computational contribution, its learning-technology relevance, and the pedagogical validation that remains necessary.

7 Conclusion

This paper presented a parallel ant-inspired CF-tree framework for scalable learner profile clustering and student group formation. The framework combines CF-tree data summarization with a coordinator-executor architecture and parallel neighborhood-similarity computation. By representing learner data through CF-tree centroids and distributing ant-inspired clustering tasks across processing resources, the framework reduces the effective volume of data processed by the clustering procedure and improves practical execution time.

Experiments on two synthetic datasets and an OULAD-derived dataset containing 29,741 learner records demonstrated the computational feasibility of the approach. CF-tree representation reduced processing time by approximately threefold compared with the conventional ant-inspired clustering method. The parallel implementation further reduced wall-clock processing time while preserving cohesive learner-profile neighborhood structures. These findings show that the framework can provide a scalable computational component for learning-analytics-informed group recommendations in large online and blended learning environments.

The study does not claim that clustering-based recommendations alone produce effective collaborative learning groups. Future work will validate the framework through teacher-in-the-loop and classroom-based studies, comparing its recommendations with random, self-selected, and instructor-designed groups. Future investigations should also incorporate pedagogical grouping constraints, evaluate collaboration quality and learning outcomes, assess fairness and transparency, and explore distributed, cloud-based, and GPU-enabled implementations.

Acknowledgements. Perplexity AI assisted with language editing and manuscript presentation. All scientific contributions are the authors' own; they reviewed the final text and take full responsibility for its content, accuracy, and originality.

Declarations

Funding. This work was supported by the Tunisian Ministry of Higher Education and Scientific Research under grant LR11ES48.

Competing Interests. The authors declare that they have no competing interests.

Data Availability. The Open University Learning Analytics Dataset (OULAD) used in this study is publicly available at <https://research.stem.open.ac.uk/ouanalyse/dataset/>. The synthetic datasets and the preprocessing procedures used to construct the OULAD-derived learner-feature dataset are available from the corresponding author upon reasonable request.

Code Availability. The source code used to implement the proposed framework and conduct the experiments is available from the corresponding author upon reasonable request.

Author Contribution. M. Chniter developed the methodology, implemented the framework, conducted the experiments, analyzed the results and drafted the initial manuscript. I. Kallel revised and restructured the manuscript and proposed the

application of the framework to collaborative learning group formation. H. M. Kamoun provided technical support for the university's high-performance virtual server, helped resolve technical issues, and reviewed the manuscript. S. Kanoun supervised the research and reviewed the manuscript. All authors approved the final manuscript.

Ethics approval and consent to participate. This study used synthetic data and existing publicly available, de-identified OULAD records. No participants were recruited, and no new personal data were collected.

Consent for publication. Not applicable. The manuscript does not contain identifiable personal information.

References

- [1] Kallel, I., & Chniter, M. (2019, October). Building Collaborative e-learning teams in a smart education environment. In 2019 IEEE International Smart Cities Conference (ISC2) (pp. 324-329). IEEE.
- [2] Chniter, M., Abid, A., & Kallel, I. (2018, April). Towards a bio-inspired ACO approach for building collaborative learning teams. In 2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET) (pp. 1-8). IEEE.
- [3] Srba, I., & Bielikova, M. (2014). Dynamic group formation as an approach to collaborative learning support. *IEEE transactions on learning technologies*, 8(2), 173-186.
- [4] Sánchez, O. R., Ordonez, C. A. C., Duque, M. A. R., & Pinto, I. I. B. S. (2021). Homogeneous group formation in collaborative learning scenarios: An approach based on personality traits and genetic algorithms. *IEEE Transactions on Learning Technologies*, 14(4), 486-499.
- [5] Liang, C., Toyokawa, Y., & Ogata, H. (2025). Optimizing group formation with a mixed genetic algorithm: an empirical study in active reading using marker data: C. Liang et al. *International Journal of Computer-Supported Collaborative Learning*, 20(4), 519-548.
- [6] Chniter, M., Abid, A., Kallel, I., & Kanoun, S. (2022, October). Computational complexity analysis of ant colony clustering algorithms: Application to student's grouping problem. In 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 736-741). IEEE.
- [7] Kuzilek, J., Hlosta, M., & Zdrahal, Z. (2017). Open university learning analytics dataset. *Scientific data*, 4(1), 170171.
- [8] Cruz, W. M., & Isotani, S. (2014, September). Group formation algorithms in collaborative learning contexts: A systematic mapping of the literature. In CYTED-RITOS International Workshop on Groupware (pp. 199-214). Cham: Springer International Publishing.
- [9] Odo, C., Masthoff, J., & Beacham, N. (2019). Group formation for collaborative learning. In *International conference on artificial intelligence in education* (pp. 206-212). Springer, Cham.

- [10] Ding, J., Fan, Z., Liu, H., Slotta, J., Ji, F., & Hu, K. (2026). Understanding collaborative programming dynamics: The role of prior knowledge, engagement and ICAP learning modes. *International Journal of Computer-Supported Collaborative Learning*, 21(1), 7-45.
- [11] AJ, D., & Naik, D. (2026, January). Dynamic peer learning recommendations in e-learning using hybrid collaborative filtering and interaction-based clustering. In *Frontiers in Education* (Vol. 10, p. 1660954). Frontiers.
- [12] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, 26(1), 29-41.
- [13] Neto, R. T., & Godinho Filho, M. (2013). Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering applications of artificial intelligence*, 26(1), 150-161.
- [14] Sim, K. M., & Sun, W. H. (2003). Ant colony optimization for routing and load-balancing: survey and new directions. *IEEE transactions on systems, man, and cybernetics-Part A: systems and humans*, 33(5), 560-572.
- [15] Kammoun, H. M., Kallel, I., Casillas, J., Abraham, A., & Alimi, A. M. (2014). Adapt-Traf: An adaptive multiagent road traffic management system based on hybrid ant-hierarchical fuzzy model. *Transportation Research Part C: Emerging Technologies*, 42, 147-167.
- [16] Chaharsooghi, S. K., & Kermani, A. H. M. (2008). An effective ant colony optimization algorithm (ACO) for multi-objective resource allocation problem (MORAP). *Applied mathematics and computation*, 200(1), 167-177.
- [17] Moss, J., & Johnson, C. G. (2003, April). An ant colony algorithm for multiple sequence alignment in bioinformatics. In *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Roanne, France, 2003* (pp. 182-186). Vienna: Springer Vienna.
- [18] Hou, C., Yuan, Z., Zhang, Y., Lei, Z., & Gao, S. (2026). Temporal difference reinforcement learning-based ant colony optimization with extremized probability construction for feature selection. *Results in Engineering*, 109721.
- [19] Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chrétien, L. (1991, February). The dynamics of collective sorting robot-like ants and ant-like robots. In *From animals to animats: proceedings of the first international conference on simulation of adaptive behavior* (pp. 356-365).
- [20] Lumer, E. D., & Faieta, B. (1994). Diversity and adaptation in populations of clustering ants.
- [21] Abid, A., Kallel, I., Sanchez-Medina, J. J., & Ayed, M. B. (2023). Parameters sensitivity analysis of ant colony based clustering: Application for student grouping in collaborative learning environment. *IEEE Access*, 12, 24751-24761.
- [22] Tan, S. C., Ting, K. M., & Teng, S. W. (2011). Simplifying and improving ant-based clustering. *Procedia Computer Science*, 4, 46-55.
- [23] Zhang, L., & Cao, Q. (2011). A novel ant-based clustering algorithm using the kernel method. *Information Sciences*, 181(20), 4658-4672.

- [24] Menéndez, H. D., Otero, F. E., & Camacho, D. (2016). Medoid-based clustering using ant colony optimization. *Swarm Intelligence*, 10(2), 123-145.
- [25] Fahy, C., Yang, S., & Gongora, M. (2018). Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams. *IEEE transactions on cybernetics*, 49(6), 2215-2228.
- [26] Singh, N., Singh, D. P., & Pant, B. (2019). ACOCA: ant colony optimization based clustering algorithm for big data preprocessing. *International Journal of Mathematical, Engineering and Management Sciences*, 4(5), 1239-1250.
- [27] Malik, S., Patro, S. G. K., Mahanty, C., Lasisi, A., & Al-sareji, O. J. (2025). MutaSwarmClus: enhancing data clustering efficiency with mutation-enhanced swarm algorithm. *Cluster Computing*, 28(3), 188.
- [28] Moufok, S., Belkadi, K., & Lebbah, F. Z. (2025). An artificial ant-based approach using polynomial algorithms to tackle the text aspect of clustering web pages. *Journal of Applied Data Sciences*, 6(2), 1050-1075.
- [29] Qasem, M., Liu, Y. Y., Wang, Z., Thulasiraman, P., & Thulasiram, R. K. (2017, August). Enhancing ant brood clustering with adaptive radius of perception and non-parametric estimation on multi-core architectures. In *International Conference on Intelligent Networking and Collaborative Systems* (pp. 301-312). Cham: Springer International Publishing.
- [30] Gemici Gunes, O., & Sima Uyar, A. (2010). Parallelization of an ant-based clustering approach. *Kybernetes*, 39(4), 656-677.
- [31] Yang, Y., Teng, F., Li, T., Wang, H., Wang, H., & Zhang, Q. (2015). Parallel semi-supervised multi-ant colonies clustering ensemble based on mapreduce methodology. *IEEE Transactions on Cloud Computing*, 6(3), 857-867.
- [32] Chniter, M., Kallel, I., Garcia, A. B., & Medina, J. S. (2024, November). Creative writing and social action: A data analysis based on learning analytics about how social, cultural and linguistic barriers can be overcome through educational innovation and international cooperation. In *2024 21st international conference on information technology based higher education and training (ITHET)* (pp. 1-10). IEEE.