Computer Science and Information Systems 23(1):75–112 https://doi.org/10.2298/CSIS240430073G

DomiCom: Discovery of Top-k Dominant Communities in Networks with Node Attributes

Nikolaos Georgiadis[1], Eleftherios Tiakas[2], and Apostolos N. Papadopoulos[1]

[1] Aristotle University of Thessaloniki
Thessaloniki, Greece
ngeorgii@csd.auth.gr
papadopo@csd.auth.gr
[2] International Hellenic University
Thessaloniki, Greece
tiakas@ihu.gr

Abstract. The community structure is an inherent property of real-world networks. Broadly, a set of nodes S forms a community if its nodes exhibit a significantly higher level of interconnection with each other than with nodes outside S. The precise notion of a community, however, depends on the mathematical formulation adopted. The problem becomes more intriguing when graph nodes are enriched with attributes, as these attributes can influence how communities are defined. In this work, we incorporate node attributes to construct more meaningful communities that reflect both structural connections and attribute information. Specifically, we introduce the concept of dominance relationships between nodes: a node u is considered more important than a node v if the attributes of u *dominate* those of v. Experimental evaluation on real-world attributed networks demonstrate the efficiency and effectiveness of the proposed approach. The reported communities are meaningful and robust based on their significance and structural coherence.

Keywords: community detection, attributed graphs, dominance relationships.

1. Introduction

The study of real-world networks [1] enables the discovery of significant knowledge that can be used to solve important problems such as community detection, outlier discovery, link prediction, just to name a few. A network is represented as a graph $G(V, E)$ where V is the set of nodes or vertices and E is the set of edges or links. In this study, we focus on undirected annotated graphs, where each node contains additional information (attributes) in the form of a vector. Our target is to detect subgraphs that are robust with respect to node interconnections and also strong with respect to the domination score, which is being used as a measure of node (and subgraph) significance.

In this work, we focus on the task of community detection in annotated (attributed) graphs. More specifically, given an annotated graph, where each node $u \in V$ contains a set of d attributes represented as a vector $vec(u)$, our aim

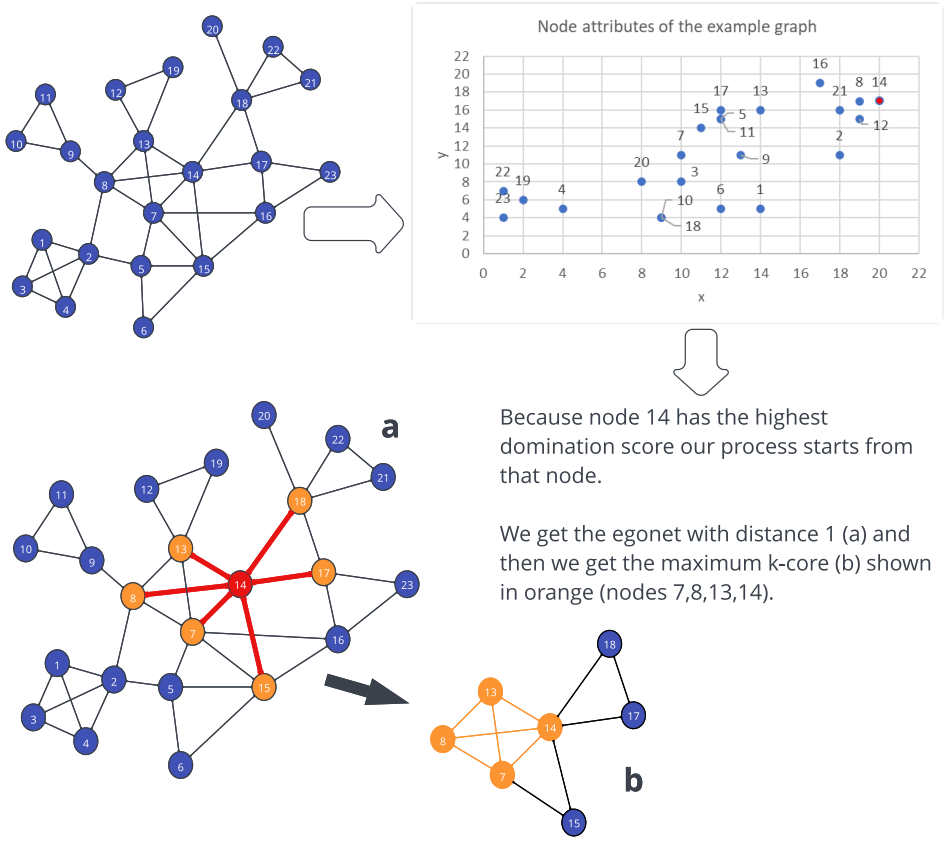


Fig. 1. Overview of the DomiCom process. An initial attributed graph (top left) is transformed into a scatter plot of node attributes (top right) to compute domination scores. The highest-scoring node (14) is chosen as the seed. (a) The 1-hop egonet around node 14 is induced. (b) The maximum k-core of this egonet yields the final dense community 7, 8, 13, 14

is to discover the c *most important* subgraphs (i.e., communities). The importance of a community is quantified using graph properties as well as properties related to node attributes. Without loss of generality, we assume that the larger attribute values are more significant than the smaller ones. Therefore, a node u with $vec(u) = [5, 7, 4]$ is considered more significant than another node v with $vec(v) = [2, 3, 2]$.

The domination score of a node u is computed by counting the number of nodes *dominated* by u and expresses the *relative power* of a node with respect to its attributes and can be interpreted differently depending on the application: in citation

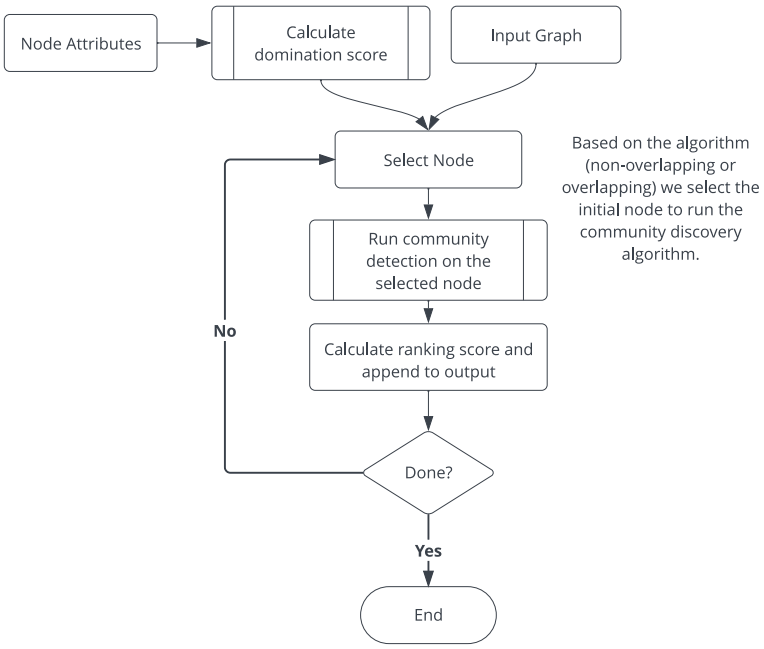


Fig. 2. Data flow of the DomiCom algorithm. The process starts by calculating domination scores from node attributes. A seed node is then selected from the input graph, and the community detection algorithm is run. A ranking score is calculated for the new community, and the process iterates until the desired number of communities is found

networks it reflects researcher *influence*, in social networks it captures *activity* or *visibility*, while in biological networks it may indicate *functional centrality*. Beyond ranking individual nodes, the distribution of domination scores strongly affects the resulting communities: skewed distributions yield star-like structures around a few dominant nodes, whereas balanced distributions produce denser, multi-centered groups.

Briefly, the formation of communities is based on two different concepts: a) the *domination score* of the nodes participating in the community, which is calculated using node attributes, and b) the *structural coherence* of the community, which is computed by means of the *core decomposition* process. The domination score of a node represents the power of the node to dominate other nodes based on the values of the attributes. For example, node u with $vec(u) = [5, 6]$ dominates another node v with $vec(v) = [1, 2]$. The core decomposition process identifies subgraphs that are highly connected and are characterized by structurally coherent. More details of these concepts will be given in the subsequent sections.

In the sequel, we provide a simple example to introduce the methodology of the proposed approach. Figure 1 illustrates a high-level overview of our approach using a small example graph, while Figure 2 shows the data flow of the proposed approach. The first step in our algorithm is to compute the domination score for

Table 1. The domination score of nodes in the example

node ID	**dom. score**	**node ID**	**dom. score**
14	21	7	8
8	20	3	7
21	18	1	5
16	17	6	4
12	16	20	4
13	16	18	1
17	13	4	1
2	12	10	1
5	11	19	1
11	11	22	1
9	10	23	0
15	9		

each node. Based on node attributes, the domination score for each node is given in Table 1.

In this example, our objective is to detect communities at distance one (i.e., 1-hop). However, our techniques also work for larger hop values, as well as in cases where random walks are used to generate node context [18]. Starting from the node with the highest domination score, we induce the subgraph of distance 1 as shown in Figures 3a and 3b. This results in a subgraph that contains the following set of nodes {7, 8, 13, 14, 15, 16, 17, 18}. Next, we compute the maximum core, which in our example is the subgraph shown in Figure 3b, consisting of nodes {7, 8, 13, 14}. Depending on the method we use to form communities, i.e., non-overlapping or overlapping, we continue until the top c communities are finally formed. Once the process is completed, the communities are ranked using a specialized ranking function and returned to the user.

In Figure 4, we illustrate two different results of our dominant community detection algorithm for the non-overlapping and overlapping case. The difference between the non-overlapping and the overlapping case is that in the non-overlapping case the nodes can be part of only one community, whereas in the overlapping case each node may participate in more than one community, offering greater flexibility.

Overlapping community/group search can have an advantage in several scenarios, as for example: (i) Modern researchers often straddle multiple fields, e.g., someone working at the intersection of machine learning and healthcare informatics. Assigning them to just "Computer Science" or just "Health Sciences" loses critical nuance. Therefore, overlapping has the advantage of capturing multiple research domains per researcher. (ii) In topic modeling of documents, it is unrealistic that a document belongs only to one topic. Here, overlapping allows multi-topic attribute search. (iii) In the case of social circles search, it is unrealistic that friends belong only in distinct groups. Therefore, overlapping can capture people in multiple social circles.

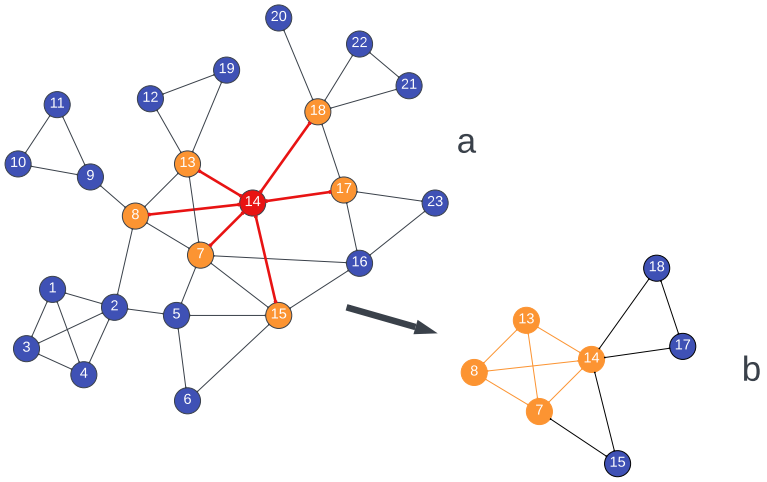
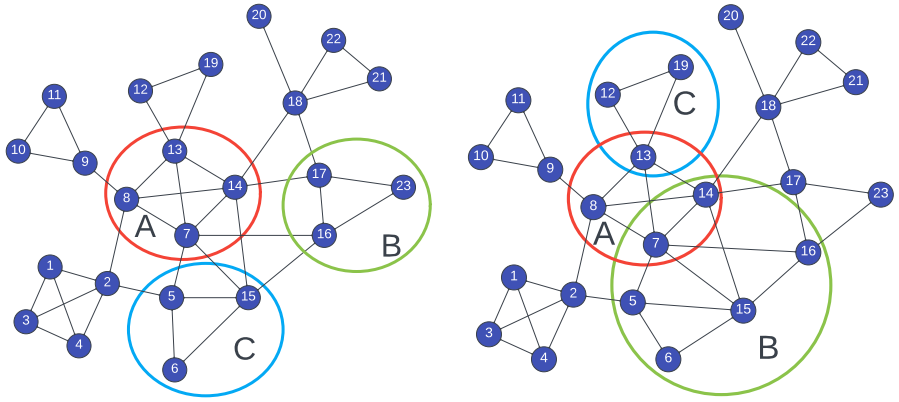


Fig. 3. The two-step community formation process. (a) First, the 1-hop egonet is generated from a seed node (node 14, in red) and its immediate neighbors (in orange). (b) The maximum k-core is then computed from this egonet, yielding the final, densely connected community of nodes 7, 8, 13, 14



(a) non-overlapping (b) overlapping

Fig. 4. Comparison of non-overlapping and overlapping community structures. (a) In the non-overlapping case, each node belongs to only one community. (b) In the overlapping case, nodes (such as 7, 13 and 14) can be members of multiple communities

According to the definitions and properties of each method (overlapping and non-overlapping search), we provide some practical guidelines:

– Use non-overlapping search when the domain implies exclusivity (e.g., membership in mutually exclusive categories), or overlap may exist but is minor and not relevant to your goals.

– Use overlapping search when nodes naturally belong to multiple groups, or the understanding of the overlap itself is valuable (bridging nodes, interdisciplinarity), or you need richer modeling (e.g., weighted memberships, soft boundaries), or you aim to discover structural overlap phenomena like community bridges, layered hierarchies, or fuzzy boundaries.

A practical advice is first to start simple and run a non-overlapping method to get rough intuition. This can be our baseline. Then, we may inspect "unnatural" nodes: for example, nodes with high modularity but ambiguous affiliation may be a hint to try overlapping models. Sometimes, a hybrid approach can be valuable, thus we may first cluster non overlapping and then allow overlapping sub communities within large clusters. In each case, the results must be interpreted carefully. Domain experts can guide or validate the results.

Contributions. The main contributions of our paper are briefly described as follows:

– Many of the existing works consider graphs with specific assumptions and domains on their attributes. Our methodology can be applied to attributed graphs without any assumption for node attributes.
– Current state-of-the-art methods define and use domination relationships between entire communities, and they are taking into account aggregations based on node attributes. Moreover, different algorithms are required for different dimensionalities. In our method, we use the natural domination relationship between nodes of the graph by taking into account only their original attributes before the extraction of communities. Therefore, the proposed algorithms can be used on data with any dimensionality.
– We propose efficient algorithms for both non-overlapping and overlapping community detection on attributed graphs. This flexibility allows for choosing the most suitable technique based on the application. We note that existing algorithms are mainly focused on the non-overlapping case.
– As an alternative solution, we also propose more efficient *approximation* algorithms for both non-overlapping and overlapping community detection. These algorithms enable to trade accuracy for speed, meaning that the final result may not be 100% accurate, but the computations require significantly less time.

We demonstrate the efficiency and effectiveness of the proposed approach by using real-life network data with node attributes.

Roadmap. The rest of the article is organized as follows. Section 2 presents related work in the area. The proposed methodology is described in detail in Section 3. Performance evaluation results based on real-life data sets are given in Section 4, whereas Section 5 concludes our work and briefly discusses interesting future research directions.

2. Related Work

In this section, we discuss briefly related work in the area, focusing mainly on research that focuses on community detection or community search in attributed graphs, highlighting also research efforts that use preference-based techniques.

2.1. Community detection in attributed graphs

In [6] a community search methodology in attributed graphs is proposed. The supported graphs contain textual attributes and the returned results are communities that satisfy both structural cohesiveness and keyword cohesiveness.

Another related research [13],[14] introduces a community model (called k-influential community) based on the concept of k-core that takes into account the node's influence. Improved online search algorithms are also proposed for the influential community search variation [4],[2].

Another aspect in the literature is to find k-truss communities, instead of k-core. In [7] a k-truss community model is proposed with several efficient community search algorithms. The proposed algorithms can be applied also in a dynamic graph setting with frequent insertions and deletions of graph vertices and edges. An extension of the influence-based works has been proposed in [22] where a novel scoring function and algorithm is designed, that tries to find the attributed k-truss community by maximizing the attribute and influence relevance scores.

The works of [13],[14], [4],[2], [22] consider only one numerical attribute, i.e. the node's influence, or they are based on scoring functions related to that specific attribute.

In [9] a greedy-based algorithmic framework is proposed, which detects a closest truss community with small diameter. The method iteratively removes the furthest nodes from the graph in order to support the small diameter, and achieves approximations to the optimal solution. In [8] an extension is proposed to iteratively remove nodes with the least popular attributes, and shrink the graph into an attributed k-truss community.

In [10] efficient algorithms for searching k-truss communities are also proposed. These algorithms can be applied in dynamic graphs with frequent insertions and deletions of vertices and edges, and can also support k-truss community search over massive graphs which cannot entirely fit in main memory.

The work of [21] propose an efficient algorithm for complex network community partitioning by combining fitness optimization with community similarity.

In [5] the authors present an algorithm for detecting overlapping communities base on node similarity.

All the above works are substantially different from our study as they are applied on graphs of specific domains on their attributes, or they do not consider preference-based values in their calculations.

2.2. Preference-based community detection

In [12] the skyline community search problem is presented and studied, where the skyline operator is applied over all communities that are derived from a multi-valued graph. The domination relationship between two communities is defined by

Table 2. Key differences between our method and most related methods

Feature	**Proposed Method (Ours)**	**Methods of [12], [11], [24]**
Domination Scope	Between individual **nodes** using original attribute values	Between **entire communities**, often using aggregated attributes
Dimensionality Handling	Single algorithm handles **all dimensionalities**	Requires **different algorithms** for different dimensionalities
Use of Skyline Operator	**Not used**; relies on domination score	**Used** to identify key communities
Community Cohesiveness (k-core)	Considered as a **second step** in the algorithm	Not always explicitly incorporated
Community Type	Supports both **overlapping and non-overlapping** communities	Often restricted to **non-overlapping** communities
Top-k Community Retrieval	**Supported** via user-defined k parameter	**Not supported**; number of communities is uncontrolled

comparing the corresponding vectors that are derived from the minimum values for all their nodes in each dimension i (f_i-values). The authors apply specific criteria for their proposed community detection approach: a resulted community H must be a k-core subgraph, there does not exist any induced subgraph that is also k-core and dominates H, and there does not exist a subgraph of H that is also k-core and it has the same f_i-values. The proposed algorithms handle cases with specific attribute dimensionalities ($d = 2, 3$), and a different algorithm is proposed for $d > 3$. Moreover, a dimensionality reduction-based algorithm is also proposed.

An extension is presented in [11], where for the cases of $d = 3$ and $d > 3$ more details are presented, and additional pruning rules are proposed, improving the overall performance.

In [24], more performance improvements are proposed to algorithms of [12] and [11], taking into account dimensionality reduction and pruning strategies. Additional criteria, like the maximum entropy model, are applied in order to remove insignificant attributes and to improve the efficiency of community search strategies. The basic algorithm MICS and the improved algorithm P-MICS are proposed, which can be applied in larger graphs.

In our study, which is very different compared to [12], [11], [24], we apply the natural domination relationship between the graph nodes, by taking into account their original attribute values. We do not define domination relationships between entire communities, or taking aggregations (like minimum) of node attributes. Thus, in our approach, there is no need to have different algorithms for different dimensionalities. Moreover, we do not apply the skyline operator, but instead we use the domination score in order to retrieve important nodes. We are also taking into account the cohesiveness (k-core communities), but as a second step to our algorithms. In addition, we study the community detection problem with both overlapping and non-overlapping communities. This flexibility was not possible using the previously proposed approaches. Finally, our technique supports top-k retrieval of the best communities, where k is a user-defined parameter. The skyline-based community detection algorithm does not support this parameterization, and

the number of returned communities cannot be controlled by the user. This means that, depending on the dataset, the number of communities may be too small or too large.

Table 2 presents a summary of key differences between our method and the most related prior work to ours ([12], [11], [24]). Note that, as these prior methods are very similar, the key differences hold for all. To the best of the authors' knowledge, this work is the first to study the graph community discovery problem by using the domination score of nodes based on multi-valued node attributes, supporting both non-overlapping and overlapping community detection.

3. DomiCom: The Proposed Approach

In this section, first we present some fundamental topics related to our proposed methodology and then we study in detail the algorithmic techniques supporting the proposed approach for preference-based community detection.

3.1. Background

The concept of dominance has been used in many research works related to multi-criteria optimization and preference-based query processing [17]. Dominance refers to a property that decides if an object p is *better* than another object q, where objects are represented as multi-dimensional points (vectors). We assume, without loss of generality, that in each dimension, large attribute values are preferable.

Definition 1 (dominance). *An object* $p = (p.x_1, p.x_2, \ldots, p.x_d) \in \mathbb{R}^d$ *dominates another object* $q = (q.x_1, q.x_2, \ldots, q.x_d) \in \mathbb{R}^d$*, i.e.,* $p \succ q$*, when:* $\forall i \in \{1, \ldots, d\} : p.x_i \geq q.x_i \wedge \exists i \in \{1, \ldots, d\} : p.x_i > q.x_i$*. This means that* p *is as good as* q *in all dimensions, and it is strictly better than* q *in at least one dimension.*

The previous definition of dominance holds also when there are dimensions in which lower attribute values are better (e.g., age, cost), if we transform their scales so that higher values represent better outcomes. The most common transformation to preserve the total ordering, is to subtract the values from their global max: $x' = x_{max} - x$. In cases that in a dimension there are ordinal attribute values, there is a clear natural ordering or magnitude (which value is considered better), and the dominance definition can also be applied. Therefore, we can encode the ordinal values with appropriate numbers (integers or reals) by preserving this natural ordering (e.g. low=1, medium=2, high=3). In cases that in a dimension there are q distinct categorical attribute values without order (there is not any kind of ordering or magnitude), we can replace this dimension with q binary dimensions, one for each categorical value. Therefore, if a node belongs to a specific category, we assign the value 1 in the corresponding binary dimension and 0 in the rest binary dimensions. Moreover, we can also encode nodes that may belong to multiple categories by assigning 1 in all corresponding binary dimensions (extremely useful for overlapping search).

Based on the concept of dominance, one can identify points that are *not dominated*. These points are considered as the *best points* we can get from the multi-dimensional dataset. In the database literature, these *best* points are known as the *skyline* of the dataset [3]. Based on the definition of the skyline, all points in the skyline are equivalent, since there is no *ranking* concept for skyline points. The concept of *domination score* has been introduced in [23] in order to rank points based on their dominance power. Formally we have:

Definition 2 (domination score). *The domination score of a point p, dom(p) is defined as: dom(p) = $|\{q \in D : p \succ q\}|$. A top-k dominating query returns the k objects with the highest domination scores.*

The domination score is an essential ingredient of our approach. We use the domination score to determine the order in which we discover communities but also to compute the score of each community. The multi-dimensional points correspond to the attribute vectors of the graph nodes. There are in general two different techniques to compute the domination score: *i*) exact and *ii*) approximate. To enable efficient computation of the domination score in both cases, a simple *grid-based* approach, firstly reported in [23], is being used. The address space is partitioned into *cells* and each point is assigned to a unique cell based on its coordinates. The number of cells is an important parameter that affects performance.

Exact computation of domination scores. For the exact domination score computation, we need to scan the multi-attribute dataset and assign each point to the grid cell it belongs based on its attribute values. The size of the grid is a parameter that defines in how many parts the specific dimension is divided. For example, if we want to process a two-dimensional dataset we may define a 10×10 grid, which means that the x and y axes are divided in ten equal sections based on the min and max of each attribute (see Figure 5).

This approach does not take into account the data distribution, meaning that if the data concentration is denser in some areas it may impact performance. However, to avoid such a case, we could construct the grid based on the data distribution and place the data points to the grid as evenly as possible. To keep the discussion simple, we use the uniform grid approach. Depending on the definition of the domination score, the process starts from the highest nonempty dominant grid cell and continues towards the lowest by first computing the *base domination score*, i.e. the score that all elements in this grid cell have. To compute each cell's *base domination score* we need to take the sum of the number of elements that are in the d-1 dimension of the current cell.

The next step involves the examination of the points that are either in the same cell and in the same dimension (in case of two-dimensional grid, the same row and column). In the exact domination score algorithm we need to compare each node in the under investigation set (nodes that are on the same cell, row and column, for the two-dimensional example) exhaustively with every other node to get the domination score of that node.

In the example of Figure 5, suppose that we are trying to compute the domination score of a point (n_x) that resides in cell A located at (9,8). We can decide

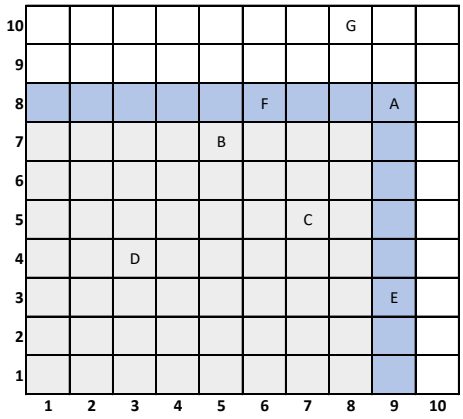


Fig. 5. Illustration of the grid-based approach for domination score computation. For a point located in cell A, it immediately dominates all points in the shaded region (cells B, C, D). Dominance relationships for points in partially dominated cells (E, F) must be calculated individually, while points in non-dominated cells (G) can be ignored

immediately that n_x dominates all points in cells B, C, D and that it does not dominate any node from cell G. Then, we need to calculate the number of points

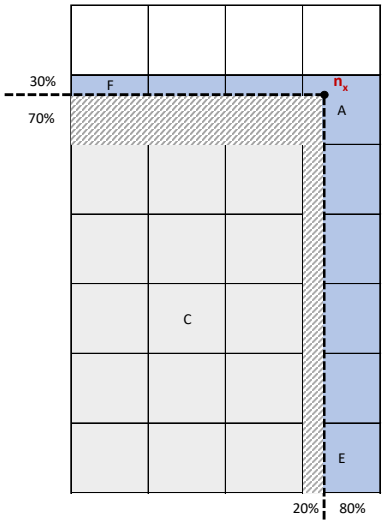


Fig. 6. Approximation of domination score using the piecewise uniformity assumption. For a point n_x in cell A, its dominance over points in partially dominated cells (e.g., F and E) is estimated based on its relative position. Assuming a uniform distribution within each cell, n_x approximately dominates a fraction of points in these cells

that point n_x dominates from the remaining cells that we could not get an immediate answer (cells A, E, and F). The number of points that we need to examine in this second pass depends on the dimensionality of the data set as well as the data distribution. The domination score of point n_x equals the total number of points contained in cells B, C, and D plus the number of points from cells A, E, and F dominated by n_x.

Approximate computation of domination scores. In this case, the domination score is approximated only for the part where we were unable to get an immediate result. To approximate the domination score, we assume that all points in the related cells are distributed uniformly (piecewise uniformity assumption). Thus, the coordinates of the point n_x determine the number of points dominated. In the example shown in Figure 6, the point n_x approximately dominates 70% of cell F regarding the y axis, 20% of cell E regarding the x axis, and therefore approximately 14% of the total number of points residing in cell A.

The use of the approximate domination score computation results in faster processing, and as we illustrate in the performance evaluation section, this approximation does not affect the accuracy of the final results significantly. The outline of the exact and approximate domination score computation is shown in Algorithm 1.

Illustrative Example

To compute the domination scores, we created a toy dataset, shown in Table 3, based on the grid example of Figure 5. Then we executed the Algorithm 1 step by step to compute the exact domination scores of each node. The computation proceeds as follows:

1. **Grid assignment:** Each attribute is normalized to a 10×10 grid using its minimum and maximum values. Each node is then mapped to a grid cell (n_1, n_2). For example, node $A = (87, 75)$ is placed in cell $(9, 8)$, while node $D = (22, 37)$ is placed in cell $(3, 4)$.
2. **Base score:** A node immediately dominates all nodes in strictly dominated cells (i.e., cells to the right, below, or both). For instance, node A dominates all points lying in cells that are to its lower-left region.
3. **Refinement:** For nodes that lie in the same row, column, or partially dominated cells, pairwise comparisons are carried out to confirm whether one node dominates the other.
4. **Final score:** The domination score of a node is the sum of its base score and refinements. For example:
   - Node $A = (87, 75)$ dominates B, C, D, E, and $F \Rightarrow$ score $= 5$.
   - Node $G = (78, 94)$ dominates B, C, D, and F, but not $A \Rightarrow$ score $= 4$.
   - Node $F = (55, 73)$ dominates B and $D \Rightarrow$ score $= 2$.
   - Node $C = (67, 44)$ dominates $D \Rightarrow$ score $= 1$.
   - Node $B = (43, 67)$ dominates $D \Rightarrow$ score $= 1$.
   - Node $D = (22, 37)$ cannot dominate anyone $\Rightarrow$ score $= 0$.
   - Node $E = (85, 24)$ cannot dominate anyone $\Rightarrow$ score $= 0$.

Algorithm 1: Domination Score Computation

```
// points is any array that holds the nodes attributes
Data: points
// These are the parameters of the algorithm
// approximate: boolean, if true we use approximate domination score
   calculation
// stats: min and max of each dimension
// gsize: grid size, defines in how many parts each dimension is
   divided
Input: approximate, stats, gsize
Result: domination score of each point
1  begin
       // distribute points into grid cells
2      foreach p in points do
           // translate point to grid cell key
3          key ← translate(p, stats)
           // append point to corresponding cell
4          append(grid[key]) ← p
       // get all non-empty grid keys and assign them to an array
5      grid_keys ← unique grid keys that each correspond to a cell with at least
         one point
       // sort grid keys in descending order
6      sort(grid_keys)
7      for i in grid_keys do
8          for j in grid_keys[i :] do
9              if j is less than i then
10                 base_score += number of points in grid[j]
11             else
12                 append(process_next) ← the points in grid[j]
13         for n in grid[i] do
14             if approximate then
15                 score ← approximate domination score of n over process_next
                     points
16             else
17                 foreach l in process_next do
18                     if n dominates l then
19                         score += 1
20             domination[n] = base_score + score
```

This staged process illustrates how raw attribute values are translated into grid coordinates, and then systematically transformed into domination scores.

Table 3. Toy dataset (nodes A–G) with attribute values, grid-cell assignments (10×10 grid), and exact domination scores

Node	**Attribute 1**	**Attribute 2**	**Grid Cell** (n_1, n_2)	**Domination Score**
A	87	75	(9,8)	5
B	43	67	(5,7)	1
C	67	44	(7,5)	1
D	22	37	(3,4)	0
E	85	24	(9,3)	0
F	55	73	(6,8)	2
G	78	94	(8,10)	4

3.2. Dominant Community Detection

In this section, we present details for the definition of the community score and the algorithms that perform non-overlapping and overlapping community detection. Table 4 includes the most frequently used symbols for the next discussion.

Table 4. Frequently used symbols

Symbol	Description
G	The undirected graph
δ	Maximum node degree in G
d	Number of dimensions of node vectors
DSA	Domination score array sorted in descending order
MAX_{dom}	Global maximum domination score of the current dataset
N	Total number of nodes in the community
n	Current node
k	Number of top-k nodes to be processed
h	Number of hops of the community to be formed
c	Number of communities reported

The domination score can help us rank the graph nodes regarding its attributes. The higher the domination score, the better attributes the node has compared to the other. Since we want to form communities that include nodes with high domination score the best possible option would be to have nodes that are closer to the maximum domination score of the entire graph. To achieve that, we compute a ranking score for each community based on Equation 1.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} |dom_i - MAX_{dom}|^2}{N}} \tag{1}$$

For a community with N nodes we compute the deviation (σ) from the maximum domination score of the entire graph (MAX_{dom}). With this expression we

are able to evaluate a community and see how close or far away the participating nodes are from the global maximum domination score. If σ is small then all the community nodes are closer to the maximum and so the community has a great value.

We are essentially trying to capture the relationship between the global maximum domination score and the domination score of each node in the community. Communities that have nodes with wide a range of domination score will rank lower that communities with narrower range and same maximum. For example in Figure 7, community B will have better ranking score than A, because the nodes of B are closer to the global maximum. For a community C with N nodes, if

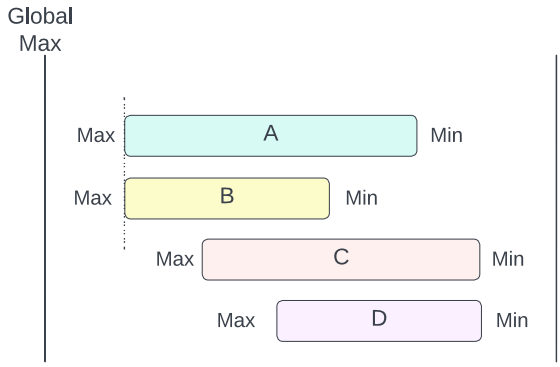


Fig. 7. An illustration of the community ranking score calculation σ. Each bar represents the range (Min to Max) of domination scores of nodes within four different communities (A, B, C, D). The ranking function rewards communities whose scores are tightly clustered and close to the global maximum domination score. For example, Community B would receive a better ranking than Community A because its nodes have a smaller deviation from the global maximum

$dom_i, i = 1, 2, ..., N$ are the corresponding domination scores of the nodes of C, the final score of C is defined as:

$$norm(C) = \frac{1}{MAX_{norm}\sqrt{\frac{1}{N}\sum_{i=1}^{N}|MAX_{dom} - dom_i|^2}} \tag{2}$$

where MAX_{dom} is the global maximum domination value from all nodes and $maxnorm$ is the global maximum $norm$ value (from all communities). The larger the value of $norm(C)$, the better the community. Note that, the value of MAX_{norm} is used only for the normalization of score values in $[0, 1]$.

3.3. Comparison with other ranking functions

We selected the deviation-based ranking function because it captures our goal of finding communities where nodes are not just strong on average, but are consistently dominant. The function evaluates a community by measuring how far its node domination scores deviate from the global maximum score. A small deviation signifies that all members of the community have scores tightly clustered near this peak value. This approach rewards groups that are uniformly powerful and ensures that a highly-ranked community is significant in the context of the whole network rather than just having a high internal average.

This method is more robust than simpler alternatives. A ranking based on the average score, for example, can be misleading, as a few nodes with exceptionally high scores could inflate the average and mask the presence of much weaker members. Our deviation-based approach provides a more balanced evaluation by considering the scores of all nodes while specifically rewarding the cohesive dominance that is central to our definition of a top-k dominant community.

To demonstrate the effectiveness of our deviation-based ranking function, we compared it with an average-based ranking function using the same dataset. We applied both ranking methods to the same set of communities and analyzed the spread of their scores. The results in Figure 8, show that the deviation-based ranking function produces a tighter spread of domination score distribution, indicating that it effectively identifies communities with consistently high domination scores. In contrast, the average-based ranking function resulted in a wider spread of domination scores which suggests that it may include communities with significant internal disparities. Figures 9 (a and b) illustrate the scatter plots of the average-based and deviation-based ranking functions, respectively. The deviation-based ranking function (Figure 9b) shows a more concentrated distribution of spread, highlighting its ability to identify communities with tighter domination scores.

There are two main approaches regarding the community detection algorithms: a) the non-overlapping and b) the overlapping approach. In the non-overlapping case, the nodes that participate in a community cannot be part of another, whereas in the overlapping case, nodes can participate in multiple communities. Our methodology supports both cases.

In our community detection algorithm, we start from the node with the highest domination score and stop when we have processed the top-k nodes. With each node we try to form a community by taking its neighbors with distance h (number of hops) and then applying the *maximum core* [15] operation. Next we compute the ranking score of the community and add it to the output list.

Recall that the maximum core of a graph G is the maximum induced subgraph S of G, where each node in S has a degree at least r, where r is the maximum possible value. By using this approach, not only we get nodes that are important based on the domination score, but also the community is cohesive.

Depending on the current case (non-overlapping or overlapping), we can point out some characteristics. In the non-overlapping case the high domination score nodes will have the opportunity to include nodes first to their community. On the other hand the overlapping method does not have this issue because the same node can be part of multiple communities. Another important difference between the

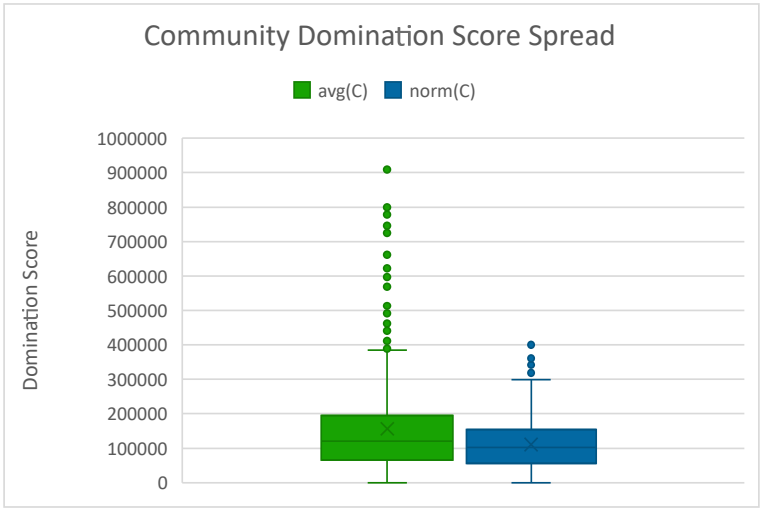
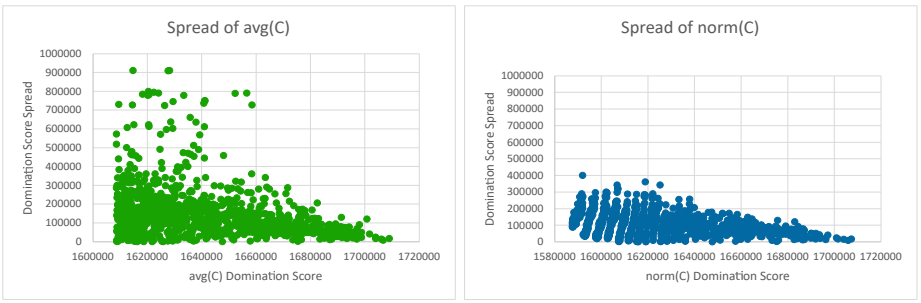


Fig. 8. Distribution of domination scores of the average-based (avg(C)) versus the deviation-based (norm(C)) ranking function. The deviation-based function yields tighter domination score distribution with fewer outliers, indicating its effectiveness in identifying communities with consistently high domination scores



(a) (b)

Fig. 9. Scatter plots visualizing the domination score spread of the two ranking functions (a) The average-based (avg(C)) function (b) The proposed deviation-based function (norm(C))

two cases regarding the computation is that in the non-overlapping case we have to compute all communities sequentially from top to bottom, because we exclude nodes that are part of a community, while the overlapping method does not have this restriction. This means that if we want to get the community that its initial node is in position 50, in the non-overlapping case we have to find all 49 previous communities, while in the overlapping case we can directly get the community we are interested in.

Before we start the process we have to define some parameters. a. the number of hops, b. how many nodes we will be processed and c. which method we will

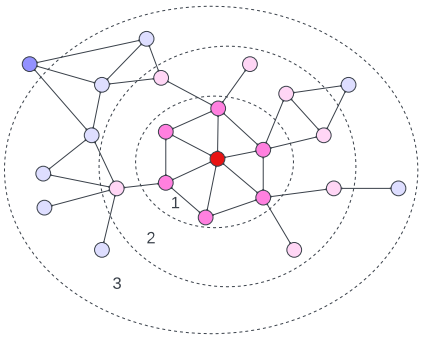


Fig. 10. Illustration of h-hop neighbors for community formation. Starting from a seed node (red), a candidate community includes all nodes within h hops (distance). Nodes at 1-hop, 2-hops, and 3-hops are shown, demonstrating how h controls the initial scope of the search

use. The number of hops is the distance from the initial node, and we will use this setting to form our basic community. Figure 10 depicts an example of a graph and the nodes that will be included using different number of hops. It is evident that as the number of hops increases, more nodes are included in the subgraph which will finally form the community.

3.4. Non-overlapping Communities

Having the exact domination scores of all nodes computed we start the community detection process from the node with the highest domination score. Based on this initial node we will try to discover a community by taking the neighbors of distance h, i.e. the egonet. From this subgraph we filter out nodes that have been visited previously (i.e. are part of another community discovered before this one). Because we want the discovered community to have dense structure, we proceed in another operation and get induced subgraph of the max core. At this point the structure of the community is complete and what we want next is to calculate the ranking score of the community and append it to the output list. All the nodes from the recently discovered community will be excluded from future communities, since this is the non-overlapping case. Once we process the top-k nodes from our sorted domination score list, we stop, sort the output by the ranking score and return the results. We expect that the communities at the top of this list will have nodes with high domination score and dense structure. The outline of the non-overlapping case is shown in Algorithm 2.

Complexity Analysis. Let $n = |V(G)|$ and $m = |E(G)|$. Also, let δ denote the maximum node degree in G, and c the desired number of top communities.

Algorithm 2: Non-overlapping algorithm

Input: G, DSA, max_dom, k, h
Output: *results*

```
// Initialize variable to hold the visited nodes
```
1 *visited* ← {}

```
// Assign the top-k nodes from the sorted domination score array
```
2 top_k ← DSA[0:k]

3 **foreach** n *in top_k* **do**

```
    // Skip if node has been visited
```
4 **if** $n \in$ *visited* **then**
5 continue

```
    // Get the induced subgraph starting from n and include all
       neighbors with distance h from the graph G
```
6 $e \leftarrow$ egonet(G, n, h)
7 filter out all nodes from $e \in$ *visited*
8 *comm* ← max core induced subgraph of e
9 add all nodes of *comm* to *visited*
10 *norm* ← calculate ranking score from *comm*
11 append *comm* and *norm* to *results*

12 sort *results* based on norm in descending order
13 return *results*

Now let us derive the complexity of Algorithm 2. The max domination scores have been calculated from Algorithm 1, thus lines 1,2 have a total complexity of $O(n+m) + O(nlogn) + O(n) = O(nlogn + m)$

The main for loop (lines 3-11) is executed till we found the desired top communities (c times). In lines 6-11 we derive a sub-graph from a starting node with h hops. Therefore, in the worst case the number of nodes in this graph are: $1 + \delta + \delta^2 + ... + \delta^h = \frac{\delta^{h+1}-1}{\delta-1}$ (if all derived nodes have the maximum degree). Thus, we have a complexity of $O(\delta^{h+1})$. The max core step has a complexity of the sum of nodes and edges in the derived sub-graph, which in the worst case that is a complete graph returns $O(\delta^{h+1}(\delta^{h+1}-1)/2) = O(\delta^{2h+2})$. The remaining calculations do not contribute more to the complexity. Therefore, the overall complexity of the algorithm is: $O(c\delta^{2h+2} + nlogn + m)$.

Theorem 1. *If community C_b is discovered after community C_a then it always holds that: $norm(C_a) \geq norm(C_b)$, according to the defined norm measure.*

Proof. Let $doma_i, i = 1, 2, ..., N_a$ the domination scores of the N_a nodes of the community C_a, and $domb_j, j = 1, 2, ..., N_b$ the domination scores of the N_b nodes of the community C_b.

In the case of non-overlapping communities and without loss of generality, we have the following domination scores order:

$doma_1 \geq doma_2 \geq ... \geq doma_{N_a} \geq domb_1 \geq domb_2 \geq ... \geq domb_{N_b}$
$\Leftrightarrow MAX_{dom} - doma_1 \leq MAX_{dom} - doma_2 \leq ... \leq MAX_{dom} - doma_{N_a} \leq$
$\leq MAX_{dom} - domb_1 \leq MAX_{dom} - domb_2 \leq ... \leq MAX_{dom} - domb_{N_b}$
where MAX_{dom} is the global maximum domination value.

As all domination scores are less than or equal to MAX_{dom}, all the above differences are non-negative numbers, thus we can take their squares without changing any order:
$|MAX_{dom} - doma_1|^2 \leq |MAX_{dom} - doma_2|^2 \leq ... \leq |MAX_{dom} - doma_{N_a}|^2 \leq$
$\leq |MAX_{dom} - domb_1|^2 \leq |MAX_{dom} - domb_2|^2 \leq ... \leq |MAX_{dom} - domb_{N_b}|^2$
Therefore, it definitely exists a real non-negative number L, such that:

$$\begin{aligned} |MAX_{dom} - doma_1|^2 \leq |MAX_{dom} - doma_2|^2 \leq ... \\ ... \leq |MAX_{dom} - doma_{N_a}|^2 \leq L \leq |MAX_{dom} - domb_1|^2 \leq \\ \leq |MAX_{dom} - domb_2|^2 \leq ... \leq |MAX_{dom} - domb_{N_b}|^2 \end{aligned} \tag{3}$$

From Inequality 3 due to transitivity property for the $doma_i, i = 1, 2, ..., N_a$ values we derive:
$|MAX_{dom} - doma_1|^2 \leq L$
$|MAX_{dom} - doma_2|^2 \leq L$
...
$|MAX_{dom} - doma_{N_a}|^2 \leq L$
and by using summation we get
$\sum_{i=1}^{N_a} |MAX_{dom} - doma_i|^2 \leq N_a L \Leftrightarrow \frac{1}{N_a} \sum_{i=1}^{N_a} |MAX_{dom} - doma_i|^2 \leq L.$
Therefore, it holds that:

$$\sqrt{\frac{1}{N_a} \sum_{i=1}^{N_a} |MAX_{dom} - doma_i|^2} \leq \sqrt{L} \tag{4}$$

With a similar way, from Inequality 3 due to transitivity property for the $domb_j, j = 1, 2, ..., N_b$ values we derive:
$L \leq |MAX_{dom} - domb_1|^2$
$L \leq |MAX_{dom} - domb_2|^2$
...
$L \leq |MAX_{dom} - domb_{N_b}|^2$
and by summing we derive:
$N_b L \leq \sum_{j=1}^{N_b} |MAX_{dom} - domb_j|^2 \Leftrightarrow L \leq \frac{1}{N_b} \sum_{j=1}^{N_b} |MAX_{dom} - domb_j|^2,$
Therefore, it holds that:

$$\sqrt{L} \leq \sqrt{\frac{1}{N_b} \sum_{j=1}^{N_b} |MAX_{dom} - domb_j|^2} \tag{5}$$

By combining Inequalities 4 and 5 we derive:
$\sqrt{\frac{1}{N_a} \sum_{i=1}^{N_a} |MAX_{dom} - doma_i|^2} \leq \sqrt{\frac{1}{N_b} \sum_{j=1}^{N_b} |MAX_{dom} - domb_j|^2}$
$\Leftrightarrow \frac{1}{\sqrt{\frac{1}{N_a} \sum_{i=1}^{N_a} |MAX_{dom} - doma_i|^2}} \geq \frac{1}{\sqrt{\frac{1}{N_b} \sum_{j=1}^{N_b} |MAX_{dom} - domb_j|^2}}$

$$\Leftrightarrow \frac{1}{MAX_{norm}\sqrt{\frac{1}{N_a}\sum_{i=1}^{N_a}|MAX_{dom}-doma_i|^2}} \geq \frac{1}{MAX_{norm}\sqrt{\frac{1}{N_b}\sum_{j=1}^{N_b}|MAX_{dom}-domb_j|^2}}$$

where MAX_{norm} is the global maximum $norm$ value.

Therefore, we get that: $norm(C_a) \geq norm(C_b)$, and the proof has been completed.

3.5. Overlapping Communities

The overlapping case is outlined in Algorithm 3, and it is very similar to the non-overlapping case except that in this case we do not exclude any node or keep track of visited nodes. Consequently, nodes are allowed to participate in multiple communities without any restriction.

A variation that could combine the non-overlapping and the overlapping case, is to apply a threshold on the number of communities that each node is allowed to participate.

Algorithm 3: Overlapping algorithm

Input: G, DSA, max_dom, k, h
Output: $results$

```
// Assign the top-k nodes from the sorted domination score array
```
top_k ← DSA[0:k]
foreach n *in top_k* **do**
```
    // Get the induced subgraph starting from n and include all
       neighbors with distance h from the graph G
```
    e ← egonet(G, n, h)
    $comm$ ← max core induced subgraph of e
    $norm$ ← calculate ranking score from $comm$
    append $comm$ and $norm$ to $results$
sort results based on norm in descending order
return $results$

4. Performance Evaluation

In this section, we provide performance evaluation results with respect to the computation of the domination score and the community detection algorithms. All experiments have been conducted on a machine with AMD Ryzen 7 PRO 5850U CPU@1.90 GHz with 32GB RAM and SSD drive.

The dataset used in the performance evaluation is the publicly available *Aminer* dataset [3] [20], from which we have used the co-authorship network which contains

[3] https://www.aminer.org/aminernetwork

1,712,433 authors and 4,258,615 collaboration relationships. Each author contains the following attributes: Index ID, Name, Affiliations, Publication Count, Citation Number, H-Index, P-Index with equal A-index of this author, the P-index with unequal A-index of this author, and Research Interests. We have used the following four (4) numerical attributes in our experiments: Publication Count (PC), Citation Number (CN), H-Index (HI) and P-Index (PI) [19].

The purpose of these experiments is to evaluate the performance for different setups for the exact and approximate algorithm, and also to test the error of the approximate domination computation. The algorithm takes as a parameter the grid size, which depends on the dimensionality of the data set. For example if we use a two-dimensional dataset we have to apply also the size of the grid defining in how many parts the two-dimensional data set will be divided to. If we apply a grid size of 10×10, then each dimension is divided into 10 equal parts, resulting in 100 cells. Since we do not normalize the values of each dimension, the resulting cells will not be squares. The grid sizes in our tests had the same size in each dimension, e.g., for the 3-dimensional data sets we tested the cases of (10,10,10), (25,25,25), (50,50,50), (100,100,100), (250,250,250) and (500,500,500).

In the experimental study, we evaluate our community detection algorithms by measuring the performance and community structure for various combinations of the algorithm's parameters, such as total number of communities, number of hops, overlapping or non-overlapping. More specifically, we evaluate the proposed algorithms for both non-overlapping and overlapping cases and for different parameter combinations (1, 2, 3 hops, 5k, 10k, 50k, 100k, 500k communities). The source code for all algorithms covered in this paper is available at `https://github.com/ngeorgiadis/community-detection`.

4.1. Comparison with Existing Techniques

Before diving into details related to the performance of the proposed approach, we present representative results with respect to the comparison between our DomiCom approach and the SkyCom approach, which is also based on dominance relationships. We point out that in general the two approaches are not directly comparable, since they generate communities under different criteria. However, it is interesting to see some results that mainly demonstrate that our proposal is more general and also more effective.

First, we present some results showing the performance of SkyCom for the Aminer data set. Table 5 presents the runtime, the number of communities returned, the average number of nodes per community, and the total number of nodes contained in each core subgraph. Moreover, Table 6 presents the same information where the node attributes follow an anticorrelated distribution, which is expected to pose significant challenges due to the computation of the skyline. Runtime results for the DomiCom approach are given in Table 7.

Based on the values reported in the previous tables, we observe the following:

– The SkyCom approach cannot control the number of communities returned, since it is based on skylines. This means that the number of communities strongly depends on the distribution of attribute values. For example, for the

Table 5. SkyCom results for the AMiner data set

Core	Time (s)	Num. of Results	Avg. Num. of Nodes	Num. of Nodes
1	69.77	5	2.00	1560640
2	46.23	4	3.50	1238105
3	140.91	16	5.81	878167
4	99.41	16	5.75	578194
5	46.44	9	6.11	379976
6	31.14	8	7.75	256078
7	36.60	11	9.00	180276
8	32.06	12	9.83	130556
9	19.71	11	10.91	97208
10	13.70	10	12.10	72943
11	9.20	8	13.63	56046
12	7.63	7	15.00	43093
13	7.22	8	14.38	33588
14	5.26	8	15.00	25917
15	3.35	7	16.00	20200
16	3.11	9	17.00	15848
17	1.61	8	18.00	12295
18	0.90	6	19.00	10399
19	0.58	5	20.00	9246
20	0.59	5	23.00	8233
21	0.84	8	22.00	7073
22	1.31	9	23.00	6313
23	0.53	9	24.00	5831
24	0.56	11	25.00	5118
25	0.53	8	26.00	4699
26	0.33	7	27.00	4290
27	0.50	5	28.00	3716
28	0.21	6	29.00	3408
29	0.17	8	30.00	2980
30	0.13	7	31.00	2719

Table 6. SkyCom results for the anticorrelated distribution

Core	Time(s)	Num. of Results	Avg. Num. of Nodes	Num. of Nodes
50	195.526	19512	51.000	899
40	291.567	23738	41.000	1436
30	721.305	24241	31.001	2719
20	2451.07	20056	21.127	8233
15	7563.35	15157	16.011	20200

AMiner data set we observe that the maximum number of communities returned is 12 and the minimum is 4 (Table 5). On the other hand, for the anticorrelated distribution, we get several thousand communities in the result, which cannot be easily managed (Table 6). In contrast, DomiCom accepts the

Table 7. Exact and approximate domination score computation and total running time (in seconds) for AMiner and Anti-correlated distribution

Aminer Dataset		
	1 hop (sec)	**2 hop (sec)**
Exact domination score	21.84	21.84
Evaluations of the top-50k communities	18.84	195.27
Total time	40.68	217.11
Approximate domination score	7.86	7.86
Evaluations of the top-50k communities	18.84	195.27
Total time	26.70	203.13
Anti-correlated Dataset		
	1 hop (sec)	**2 hop (sec)**
Exact domination score	313.25	313.25
Evaluations of the top-50k communities	18.84	195.27
Total time	332.09	508.52
Approximate domination score	105.12	105.12
Evaluations of the top-50k communities	18.84	195.27
Total time	123.96	300.39

number of returned communities as a parameter, and therefore the number of results may vary based on user preferences.

- In SkyCom, the number of nodes per community is highly restricted by the core value being used (first column). We observe that, in general, the number of nodes per community is around the core value being applied, resulting in communities with a very small number of nodes. This is evident in both the AMiner data set and in the data set with an anti-correlated distribution of node attribute values.
- The performance of SkyCom is highly dependent on the core value used. Higher core values are expected to require less computational cost, because the number of nodes in the selected subgraph is significantly reduced. This is shown in Table 5. On the other hand, in anticorrelated distributed data, the run-time explodes due to the difficulty in the skyline computation. In addition, the number of returned communities is totally uncontrolled and, also, the number of nodes per community is almost the same as the core number. In contrast, DomiCom our which depends primarily on the domination score computation, performs very well even if the attributes follow an anticorrelated distribution (Table 7).
- By default, SkyCom returns overlapping communities and there is no way to force the algorithm to return non-overlapping ones. In contrast, in DomiCom the user may select either the non-overlapping or the overlapping case, depending on the application.

We highlight again that the two algorithms use different definitions for the formation of communities and therefore they cannot be compared directly. We have pointed out some important findings that reveal some of the advantages and disadvantages of SkyCom and DomiCom and presented some representative results. In the following sections, we focus on our approach and present more detailed experimental evaluation results.

4.2. Domination Score Computation

In Figure 11 we observe the domination score histogram. We notice that almost half of the authors have domination score 0 (over 800k authors). We also notice, that the next non-zero domination score is placed in the 0.8M-0.9M bin.

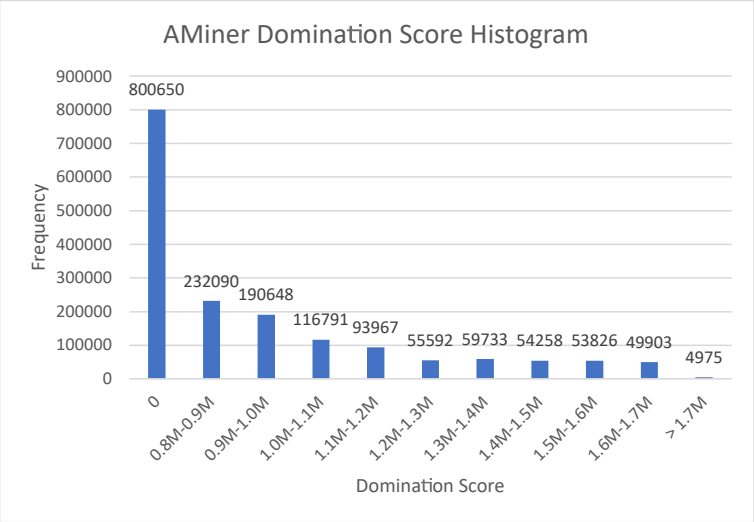


Fig. 11. Domination score histogram of the AMiner dataset

In the domination score performance evaluation, we show how much time is needed to compute the domination score of each author for the entire dataset. There are basically two parameters that we vary, one is the dimensionality and the other one is the grid size. In Figures 12 and 13 we present how the computation time progress as the grid size increases, for the exact and approximate cases respectively.

Figure 14 shows the cost comparison between the exact and the approximate domination score computation algorithms for the 4D case. The same performance has been observed for smaller dimensionalities. We observe that as the grid size increases, initially, the computation time decreases up to a point (grid size of 100 cells per dimension). After this point, the computation cost increases since the number of grid cells increases significantly.

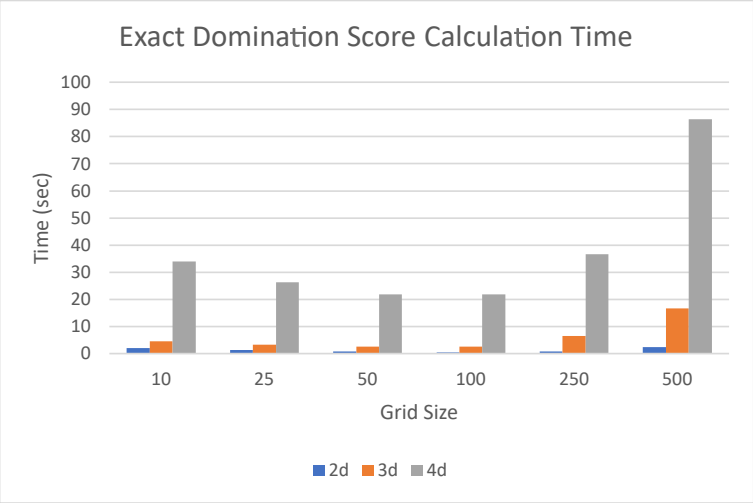


Fig. 12. Exact domination score computation time (sec) for 2, 3 and 4-d data

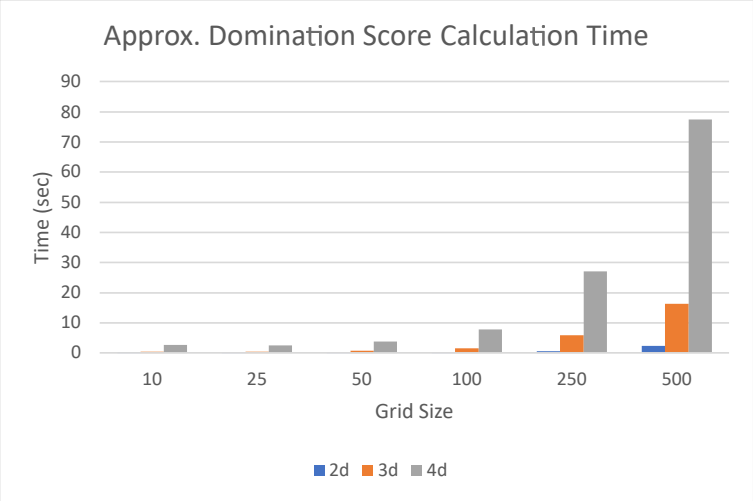


Fig. 13. Approximate domination score computation

When we approximate the domination score, we expect an error compared to the exact computation. This error highly depends on the grid size because as the grid size increases, the approximate score values tend to get closer to the exact ones. In this experiment, we compute the exact and approximate domination scores for grid sizes of 50, 100, 250, 500, and then sort by domination and compute the error on the top-10k nodes.

Figure 15 shows the Average and Root-mean-squared error, respectively. We observe that the error is very small in the 250, 500 grid size cases, whereas it is

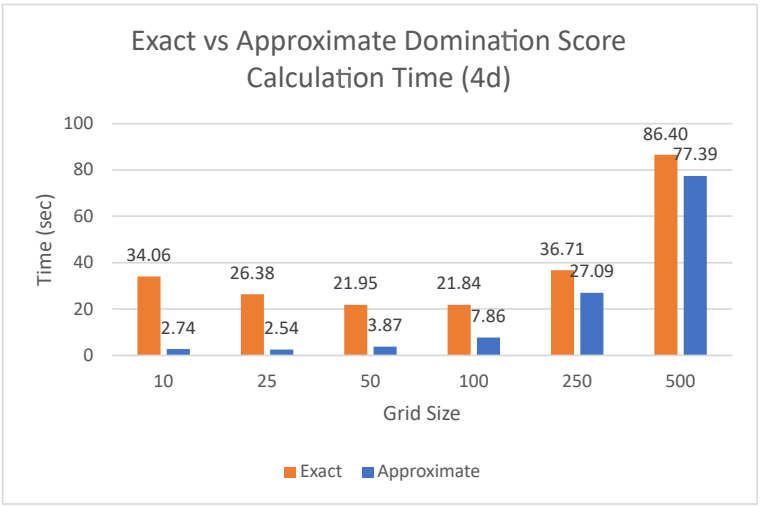
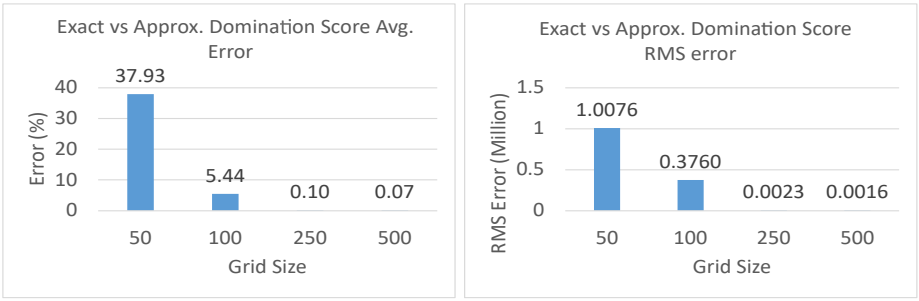


Fig. 14. Exact vs. approximate domination score calculation for 4-d data



(a) Average % error (b) RMSE

Fig. 15. Approximation error of domination scores (a) Average percentage error (b) Root mean squared error (RMSE)

quite large for a grid size of 50. Therefore, a grid size between 100 and 200 is a good compromise between runtime and error.

To summarize these observations, we provide some practical recommendations for selecting between the two variants. Exact computation is more appropriate when the dataset has low dimensionality or when the attribute distribution is highly skewed or anticorrelated, since in such cases approximation may introduce significant inaccuracies. On the other hand, approximate computation is recommended in higher dimensional settings or when runtime is a primary constraint. As shown in Figures 14 and 15, for grid sizes of 100 or larger per dimension the approximation achieves negligible error while offering up a significant speedup. Therefore, the approximate variant is particularly suitable for large-scale or exploratory analyses, while the exact variant should be preferred in accuracy-critical applications.

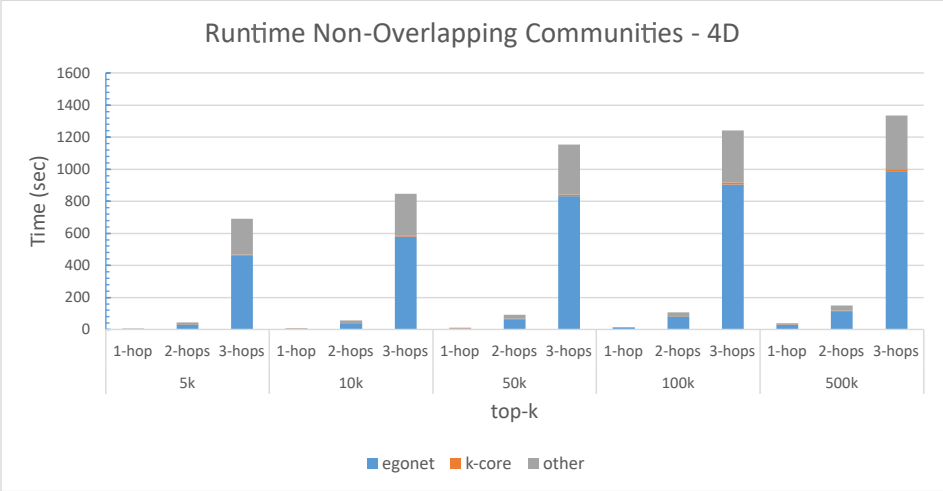


Fig. 16. Runtime breakdown of the non-overlapping community detection for 4D data. Time is split into egonet construction, k-core extraction, and other steps

4.3. Community Detection

In this section we evaluate the performance of our community detection algorithm for both non-overlapping and overlapping cases. We will examine how they perform varying the data dimensionality as well as the input parameters which are the number of the top-k nodes that are processed (5k, 10k, 50k, 100k, 500k) and the number of hops (1, 2, 3). We split the total computation time in three parts: the egonet time, core time and the rest of the process (denoted as other). In the following sections, for the domination score computation we have used the exact algorithm with a grid size of 100.

The non-overlapping case for the 4-d data is presented in Figures 16. We observe that most of the computation time is attributed to the egonet computation, i.e. to the creation of the subgraph starting from a certain node with distance h, while the core operation is the less time-consuming. There is a significant increase in runtime between the 2-hop and the 3-hop cases, while comparing the execution time between different data dimensionalities, we do not observe any significant difference. The rest of the computation marked as *other* is also significant because we need to maintain some additional information regarding the nodes that have been already processed.

The results for the overlapping are shown in Figure 17 for the 4-d case. We observe that the overlapping case is, in general, more time-consuming than the non-overlapping case. The difference between them is low when the top-k is small but, it becomes significantly large as the top-k number increases, especially when we are exploring 3-hop communities. We point out that overlapping communities provide the opportunity to include the same nodes in different communities. On the other hand, this process is more time-consuming because the number of potential communities is significantly larger.

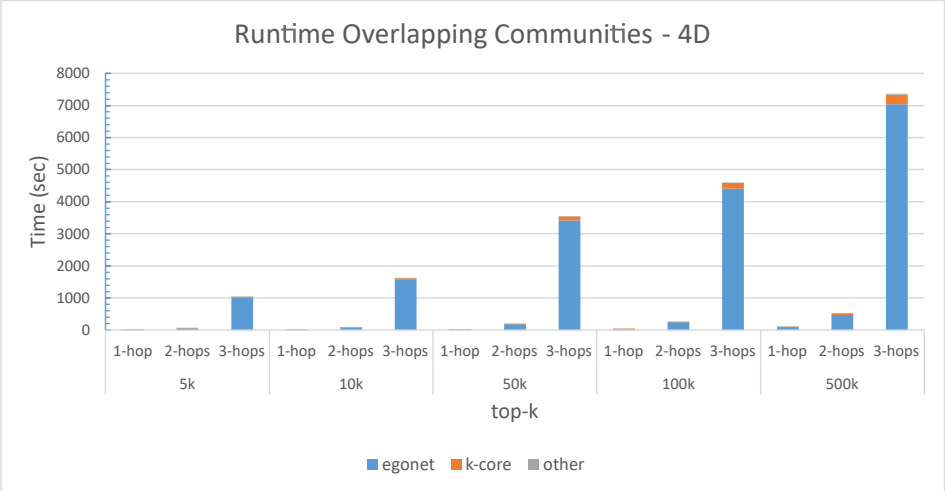


Fig. 17. Runtime breakdown of the overlapping community detection for 4D data. Time is split into egonet construction, k-core extraction, and other steps

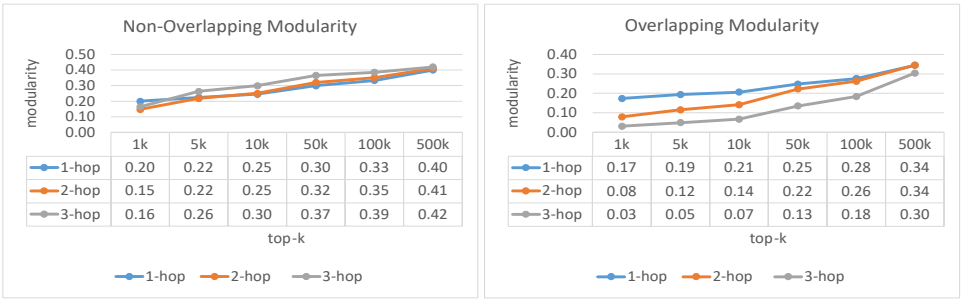
For example, in the overlapping case for the 4-dimensional data and top-50k nodes the algorithm needs 18.84 seconds for 1-hop, 195.27 seconds for 2-hops and 3546.57 seconds for 3-hops case. The non-overlapping execution time for the same parameters is 9.73 seconds for 1-hop, 91.77 seconds for 2-hops and 1155.05 seconds for 3-hops. As we observe, the overlapping case is 2 to 3 times more time-consuming than the the non-overlapping algorithm. The difference is increasing as the top-k increases and in the top-500k case the overlapping algorithm is 2.5 - 5.5 times slower.

The majority of the time here is also consumed by the egonet operation. In contrast, the *other* part in the overlapping case is very small since we don't need to monitor and maintain any list regarding which nodes have been visited.

4.4. Community Evaluation

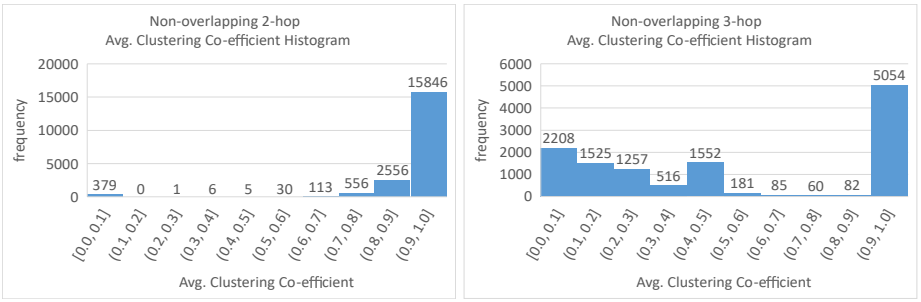
In this section, we evaluate the communities discovered by our algorithms. We also present some quality measures for both the non-overlapping and the overlapping case. In Figure 18, we observe how the average modularity of the returned communities progresses. For the non-overlapping case we observe almost the same values for 1-hop and 2-hop, while the 3-hop case has a slightly increased modularity. In the overlapping case we observe the opposite. The 1-hop case has better values than the 2-hop and 3-hop cases. In both non-overlapping and overlapping cases the highest modularity is observed when we compute the top-500k. The modularity value is above 0.3 which indicates a strong community structure, as studied in [16].

Next we explore the average clustering coefficient and the density of the discovered communities for different options for the non-overlapping and the overlapping case. In this experiment, the top-k parameter was set to 50k, whereas the detection process was executed for 1, 2 and 3 hops.



(a) non-overlapping (b) overlapping

Fig. 18. Average modularity of discovered communities varying parameters k and h for (a)non-overlapping case and (b) overlapping case



(a) 2-hop (b) 3-hop

Fig. 19. Average clustering coefficient for non-overlapping communities

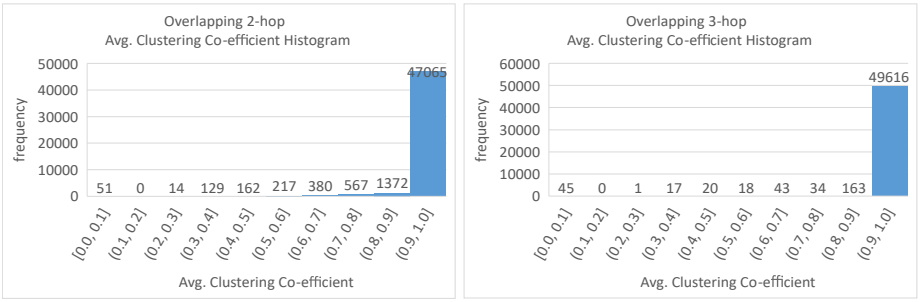
In Figure 19 we present the results of the average clustering coefficient for the non-overlapping case for the 2-hop and 3-hop cases. We observe that while most of the communities in all three cases are in the 0.9-1.0 bin, the 3-hop non-overlapping case also has a significant amount of communities fall in the lower bins (0.0 - 0.5).

In Figure 20 we present the results of the average clustering coefficient for the overlapping case for the 2-hop and 3-hop case. Here also the bin 0.9-1.0 contains the most communities and in contrast to the non-overlapping case, in the 3-hop case almost all 50k communities fall in the 0.9-1.0 bin.

In Figure 21, we present the community density histograms for the non-overlapping case for 2-hop and 3-hop. The pattern we observe here is analogous in some part to the average clustering coefficient. Most of the communities fall into the 0.9-1.0 bin. The overlapping case is similar to the non-overlapping one, as presented in Figure 22.

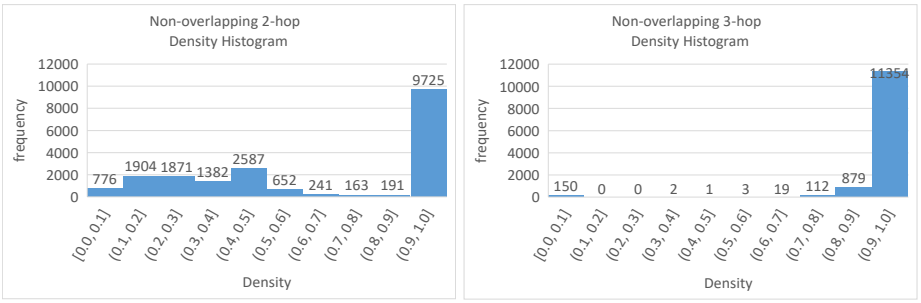
4.5. Precision of Communities with Approximate Domination Score

In this series of experiments, we evaluate the non-overlapping and the overlapping algorithms to get the top-10k communities applying domination scores that were



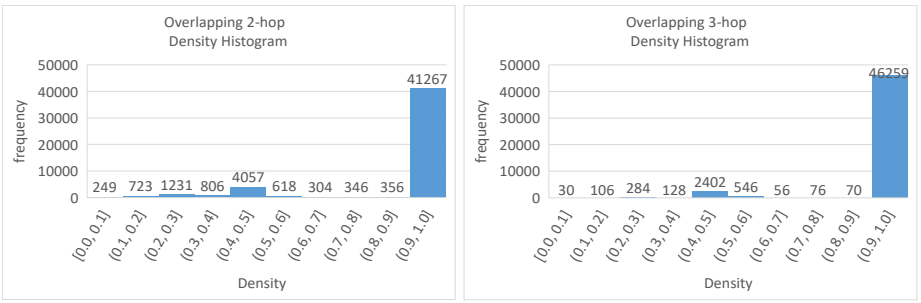
(a) 2-hop (b) 3-hop

Fig. 20. Average clustering coefficient for overlapping communities



(a) 2-hop (b) 3-hop

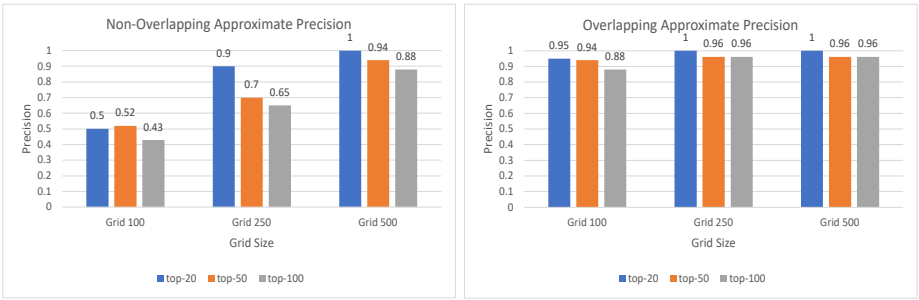
Fig. 21. Density histogram for non-overlapping communities



(a) 2-hop (b) 3-hop

Fig. 22. Density histogram for overlapping communities

computed using the approximate algorithm for various grid sizes. Then we measure the precision of the communities returned compared to the communities discovered when we apply the exact domination score computation.



(a) non-overlapping (b) overlapping

Fig. 23. Precision of approximate domination score

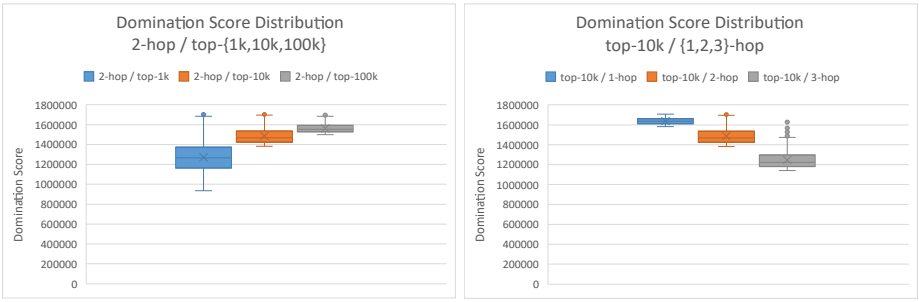
In Figure 23 we present the precision of the top 20, 50 and 100 communities for the non-overlapping and overlapping algorithms when we apply the approximate domination score for grid size 100, 250 and 500. The precision for the top-20 communities is higher for all grid sizes, with the highest precision achieved with the 100-grid. The precision for the top-50 and top-100 items is also higher for the smaller grid sizes, but the 250-grid and 500-grid still performs relatively well.

For the non-overlapping case, the precision is generally lower than the overlapping case. The precision for the top-20 communities is the highest for the 500-grid, followed by the 250-grid and then the 100-grid. For the top-50 and top-100 communities, the precision is lower for all grid sizes, with the highest precision achieved for the 500-grid.

4.6. Parameter Sensitivity

In this section we study and evaluate the impact of our method's key parameters. The number of top-k nodes to be processed (k) and the number of hops (h). To provide a robust comparison of their effects, for each parameter configuration (e.g., h=2, k=100k), we generate a set of candidate communities and then analyze the score distribution of the top-500 resulting communities, ranked by their final score based on our proposed metric ($norm(C)$). This approach allows us to evaluate which parameter settings are most effective at producing a high volume of elite, high-scoring communities, which is a key objective for practitioners seeking optimal results. The goal is to isolate the effect of each parameter on this elite subset, thereby offering guidance on tuning the method.

To achieve this, we conducted two sets of experiments. In the first, we fix the hop count (h=2) and vary k to measure how the initial search scope impacts the quality of the final top-500 communities. The hypothesis is that a broader initial search (larger k) will yield a superior elite subset. In the second, we fix k=10k and vary h to quantify how search granularity affects the quality of even the best communities found. This allows us to determine if broader, large-hop communities can compete in quality with more compact, small-hop ones when comparing their respective top-ranked results.



(a) fixed h=2, varying k (b) fixed k=10k, varying h

Fig. 24. Top-500 community score distribution. (a): fixed h=2, varying k. (b): fixed k=10k, varying h

Figure 24 presents the score distribution of the top-500 discovered communities, illustrating the impact of the search scope (k) and the community radius (h). The plot on the left (a) demonstrates that increasing the number of top-k seed nodes from 1k to 100k systematically improves the quality of the discovered communities, evidenced by a clear upward shift in the median and overall score distribution. This suggests that a broader search scope is more effective at identifying optimal communities. On the other hand, the plot on the right (b) reveals that increasing the hop count (h) from 1 to 3, causes a sharp decrease in the community scores, indicating that the highest-quality communities are compact and localized. Therefore, these findings strongly suggest that for maximizing the community score, the ideal strategy is to employ a large search scope (k) combined with a minimal community radius (h=1 or 2). Additionally, if we also consider the runtime as discussed in Section 4.3, we can see that the choice of h has a much more significant impact on performance than k. Higher selection of $h \geq 3$, is leading to exponentially longer runtimes. This further reinforces the recommendation to keep h low for practical applications.

To provide a clear and actionable summary of these findings, we present a detailed guide for parameter selection in Table 9 and Table 8. Table 9 outlines the role of the hop count (h), detailing how its value can be tuned to control the desired community granularity, from dense, high-quality local clusters (h=1) to broader, macro-level structures for exploratory analysis ($h \geq 3$), while considering the significant trade-offs in performance and average community score. Concurrently, Table 8 provides guidance on selecting the search scope (k), explaining how a focused search (small k) offers efficiency, whereas an exhaustive search (large k) maximizes the potential for discovering the optimal, highest-scoring communities. Together, these tables offer a practical framework for practitioners to configure our method based on their specific analytical goals and computational constraints.

Table 8. Guidance for Selecting the k Parameter (Search Scope)

Setting	Use-Case	Outcomes & Trade-offs
k = Small (Focused)	For fast, efficient community detection when the goal is to find good communities seeded only by the most elite nodes.	**Outcome:** A limited set of high-quality communities. **Trade-off:** May miss the single best community if it is not seeded by a top-ranked node.
k = Medium (Robust)	To achieve balance between search time and the quality of results, often representing the optimal setting.	**Outcome:** A comprehensive set of excellent communities. **Trade-off:** Our experiments show this range provides a great return on computational investment.
k = Large (Exhaustive)	Maximize the likelihood of finding the absolute best-scoring community, especially in graphs where many nodes have similar high scores.	**Outcome:** The highest potential for community quality. **Trade-off:** Increased runtime, best used when result optimality is important.

Table 9. Guidance for Selecting the h Parameter (Community Granularity)

Setting	Use-Case	Outcomes & Trade-offs
h = 1 (Local)	To find the highest-quality, dense core communities with maximum efficiency.	**Outcome:** Small, elite communities with high domination scores. **Trade-off:** Highly localized; will not discover broader structures.
h = 2 (Balanced)	To find well-structured communities that extend beyond immediate neighbors, offering a balance between size and quality.	**Outcome:** Medium-sized communities that are often a good compromise between local detail and broader context. **Trade-off:** Slower than h=1 but often manageable.
h ≥ 3 (Broad)	For exploratory analysis of large-scale, macro-level structures across the network.	**Outcome:** Very large communities. **Trade-off:** Significantly lower average quality and exponentially higher runtime.

5. Conclusions

In this work, we have studied the problem of dominant community detection in networks with node attributes. Our approach is composed of two main concepts: the power of nodes expressed by means of the domination score of the attributes and the community coherence which is computed by means of the maximum core. One of the novelties of our approach is the support of both non-overlapping and overlapping community detection. The whole process is parameterized by the number of hops. Performance evaluation results based on real-life attributed networks demonstrate the efficiency and effectiveness of the proposed approach. Efficiency is measured with respect to the overall runtime, whereas effectiveness is measured by means of the density and the clustering coefficient of the discovered communities. In addition to the exact computation of domination scores, approximate domination scores are also applied towards faster processing, without significant sacrifice of the accuracy of the results.

The proposed methodology can be applied to a variety of real-world analytical tasks. In academic citation and co-authorship networks, nodes correspond to authors annotated with attributes such as publication count, citation number or h-index. High domination score nodes highlight influential researchers, and the induced dominant communities reveal cohesive groups of high-impact authors or laboratories. The non-overlapping case can be applied to partition the network into disjoint fields, whereas the overlapping case is appropriate to capture cross-disciplinary researchers belonging to multiple groups. In this way, DomiCom can support tasks such as identifying elite research clusters, emerging teams, or bridging authors who connect different disciplines. In social networks, attributes like activity level, number of followers, and engagement metrics allow DomiCom to discover communities of influential users, supporting applications such as viral marketing, influence maximization, or misinformation monitoring. In biological interaction networks, nodes represent genes or proteins with attributes such as expression levels or disease relevance, and dominant communities correspond to functional modules that are both structurally coherent and jointly strong across biological attributes. These examples illustrate that DomiCom can be deployed in diverse settings, from compact elite clusters to broader macro-level structures, by tuning parameters such as hop count, overlapping vs. non-overlapping detection and exact or approximate score computation.

Some interesting future research directions are: 1) the use of random walks instead of hops, 2) the use of parallelism to speed-up computation, and 3) the application of node embedding techniques.

References

1. Barabási, A.L., Pósfai, M.: Network science. Cambridge University Press, Cambridge (2016), http://barabasi.com/networksciencebook/
2. Bi, F., Chang, L., Lin, X., Zhang, W.: An optimal and progressive approach to online search of top-k influential communities. Proc. VLDB Endow. 11(9), 1056–1068 (may 2018), https://doi.org/10.14778/3213880.3213881

3. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of the 17th IEEE International Conference on Data Engineering (ICDE). pp. 421–430 (2001), https://doi.org/10.1109/ICDE.2001.914855
4. Chen, S., Wei, R., Popova, D., Thomo, A.: Efficient computation of importance based communities in web-scale networks using a single machine. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. p. 1553–1562. CIKM '16, Association for Computing Machinery, New York, NY, USA (2016), https://doi.org/10.1145/2983323.2983836
5. Chen, Z., Jia, M., Yang, B., Li, X.: Detecting overlapping community in complex network based on node similarity. Comput. Sci. Inf. Syst. 12(2), 843–855 (2015)
6. Fang, Y., Cheng, R., Luo, S., Hu, J.: Effective community search for large attributed graphs. Proc. VLDB Endow. 9(12), 1233–1244 (aug 2016), https://doi.org/10.14778/2994509.2994538
7. Huang, X., Cheng, H., Qin, L., Tian, W., Yu, J.X.: Querying k-truss community in large and dynamic graphs. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. p. 1311–1322. SIGMOD '14, Association for Computing Machinery, New York, NY, USA (2014), https://doi.org/10.1145/2588555.2610495
8. Huang, X., Lakshmanan, L.V.S.: Attribute-driven community search. Proc. VLDB Endow. 10(9), 949–960 (may 2017), https://doi.org/10.14778/3099622.3099626
9. Huang, X., Lakshmanan, L.V.S., Yu, J.X., Cheng, H.: Approximate closest community search in networks. Proc. VLDB Endow. 9(4), 276–287 (dec 2015), https://doi.org/10.14778/2856318.2856323
10. Jiang, Y., Huang, X., Cheng, H.: I/o efficient k-truss community search in massive graphs. The VLDB Journal 30(5), 713–738 (apr 2021), https://doi.org/10.1007/s00778-020-00649-y
11. Li, R.H., Qin, L., Ye, F., Wang, G., Yu, J.X., Xiao, X., Xiao, N., Zheng, Z.: Finding skyline communities in multi-valued networks. The VLDB Journal 29(6), 1407–1432 (jun 2020), https://doi.org/10.1007/s00778-020-00618-5
12. Li, R.H., Qin, L., Ye, F., Yu, J.X., Xiao, X., Xiao, N., Zheng, Z.: Skyline community search in multi-valued networks. In: Proceedings of the 2018 International Conference on Management of Data. p. 457–472. SIGMOD '18, Association for Computing Machinery, New York, NY, USA (2018), https://doi.org/10.1145/3183713.3183736
13. Li, R.H., Qin, L., Yu, J.X., Mao, R.: Influential community search in large networks. Proc. VLDB Endow. 8(5), 509–520 (jan 2015), https://doi.org/10.14778/2735479.2735484
14. Li, R.H., Qin, L., Yu, J.X., Mao, R.: Finding influential communities in massive networks. The VLDB Journal 26(6), 751–776 (dec 2017), https://doi.org/10.1007/s00778-017-0467-4
15. Malliaros, F.D., Giatsidis, C., Papadopoulos, A.N., Vazirgiannis, M.: The core decomposition of networks: theory, algorithms and applications. The VLDB Journal 29(1), 61–92 (nov 2019)
16. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical review E 69(2), 026113 (2004)
17. Papadopoulos, A., Tiakas, E., Tzouramanis, T., Georgiadis, N., Manolopoulos, Y.: Skylines and Other Dominance-Based Queries. Synthesis Lectures on Data Management, Morgan-Claypool (2020)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (03 2014)

19. Stallings, J., Vance, E., Yang, J., Vannier, M.W., Liang, J., Pang, L., Dai, L., Ye, I., Wang, G.: Determining scientific impact using a collaboration index. Proceedings of the National Academy of Sciences 110(24), 9680–9685 (2013), https://www.pnas.org/doi/abs/10.1073/pnas.1220184110
20. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: Extraction and mining of academic social networks. In: KDD'08. pp. 990–998 (2008)
21. Wang, M., Yang, S., Wu, L.: Improved community mining method based on LFM and EAGLE. Comput. Sci. Inf. Syst. 13(2), 515–530 (2016)
22. Xie, X., Song, M., Liu, C., Zhang, J., Li, J.: Effective influential community search on attributed graph. Neurocomputing 444, 111–125 (2021), https://www.sciencedirect.com/science/article/pii/S0925231221001223
23. Yiu, M.L., Mamoulis, N.: Efficient processing of top-k dominating queries on multi-dimensional data. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB). pp. 483–494 (2007), http://www.vldb.org/conf/2007/papers/research/p483-yiu.pdf
24. Yu, D., Zhang, L., Luo, Q., Cheng, X., Yu, J., Cai, Z.: Fast skyline community search in multi-valued networks. Big Data Mining and Analytics 3(3), 171–180 (2020)

Nikolaos Georgiadis is a Ph.D. candidate at the School of Informatics of Aristotle University of Thessaloniki (AUTH). He received a B.Sc. in Software Engineering from the Technical University of Kavala and a M.Sc. in Information Systems from the School of Informatics of AUTH. His main research interests focus on efficient algorithmic techniques for Knowledge Discovery from Big Data, Data mining, Social Networks, Data Structures, Similarity Search.

Eleftherios Tiakas, received a B.Sc. in Mathematics (1994) from the Department of Mathematics, a B.Sc. in Computer Science - Informatics (2006), an M.Sc. in Computer Science - Informatics with focus on Information Systems (2006), and a Ph.D. in Informatics (2011) from the Department of Informatics, Faculty of Sciences, Aristotle University of Thessaloniki (AUTH). During the period 2011-2022 he served as Adjunct Lecturer, a postdoctoral researcher, and in the Laboratory Teaching Staff of the Department of Informatics, AUTH. Since 2022, he is an Assistant Professor in the Department of Accounting and Information Systems, International Hellenic University (IHU). His research interests include Databases, Data Structures, Data Mining, Information Retrieval, Similarity Search, Social Networks. He is the author/co-author of 16 articles published in international journals, 26 papers published in international conference proceedings, 1 book, 2 book chapters, and 2 technical reports. His publications have received over 850 citations. He participated in 16 research projects and programs funded by the European Union and Greece. He also serves as a PC-Member and reviewer for several international journals and conferences.

Apostolos N. Papadopoulos received his 5-year Diploma Degree from the Computer Engineering and Informatics Department of the University of Patras and his PhD Degree in Computer Science from the Department of Informatics of Aristotle University of Thessaloniki, Greece. Since May 2023, he is a Full Professor with the Department of Informatics of Aristotle University. His research interested include:

efficient query processing techniques in database systems, algorithms for mining graphs, parallel and distributed algorithms for big data, and data management and mining data streams. According to Google Scholar he has 4567 citations and his h-index is 35, whereas according to American Mathematical Society his Erdős number is 3. https://scholar.google.com/citations?user=b8ptH6oAAAAJ&hl=en

Received: April 30, 2024; Accepted: October 30, 2025.