

Efficient Mining of Combined Subspace and Subgraph Clusters in Graphs with Feature Vectors

Stephan Günnemann, Brigitte Boden, Ines Färber, and Thomas Seidl

RWTH Aachen University, Germany
{guennemann,boden,faerber,seidl}@cs.rwth-aachen.de

Abstract. Large graphs are ubiquitous in today’s applications. Besides the mere graph structure, data sources usually provide information about single objects by feature vectors. To realize the full potential for knowledge extraction, recent approaches consider both information types simultaneously. Thus, for the task of clustering, combined clustering models determine object groups within one network that are densely connected and show similar characteristics. However, due to the inherent complexity of such a combination, the existing methods are not efficiently executable and are hardly applicable to large graphs.

In this work, we develop a method for an efficient clustering of combined data sources, while at the same time finding high-quality results. We prove the complexity of our model and identify the critical parts inhibiting an efficient execution. Based on this analysis, we develop the algorithm EDCAR that approximates the optimal clustering solution using the established GRASP (Greedy Randomized Adaptive Search) principle. In thorough experiments we show that EDCAR outperforms all competing approaches in terms of runtime and simultaneously achieves high clustering qualities. For repeatability and further research we publish all datasets, executables and parameter settings on our website¹.

1 Introduction

In recent years, real world networks have become bigger and also more numerous. Their growing availability motivated researchers and practitioners to analyze and use them for several purposes. One aim is the cluster analysis of graph data, which can be done in various ways [1] including the task of mining densely connected subgraphs hidden in one large graph. This task is useful for, e.g., social network analysis. Besides partitioning approaches [10, 9] some methods assume that the given graph naturally divides into (possibly overlapping) subgraphs of certain patterns, e.g. cliques or γ -quasi-cliques [20, 8].

Restricting the considerations to the nodes’ relations only, however, does not realize the full potential for knowledge extraction. Usually for all objects a variety of additional information is available in form of attribute data (cf. Fig. 1).

¹ <http://dme.rwth-aachen.de/EDCAR>

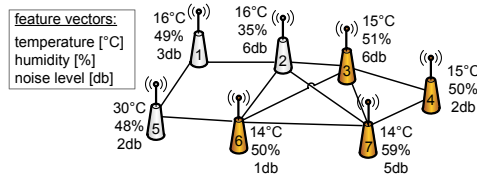


Fig. 1. Exemplary graph with feature vectors and a combined subspace and dense subgraph cluster (located in subspace {temperature, humidity})

This information allows for finding homogeneous node sets. In order to gain more informative patterns it is preferable to consider relationships together with shared characteristics. As shown, e.g., in [6, 18], clustering methods using both information sources can outperform methods using just a single one. Recently introduced techniques aim at combining traditional clustering (using attributes) and dense subgraph mining (using relationships). They group objects based on a high connectivity as well as on a high similarity concerning their attribute values. This responds to the requirements of many applications: To reduce energy consumption in sensor networks, the long distance reports of connected sensors with similar measurements can be accumulated and transferred by just one representative. In systems biology, functional modules can be determined, which are groups of highly interacting genes with similar expression levels. In social networks, closely related friends with similar interests are useful for target marketing.

While the domain's data usually represents a multitude of different recorded characteristics, not all of them need to be relevant for each cluster. In, e.g., social networks it is very unlikely that people are similar within all of their characteristics. In Fig. 1 the sensors 3, 4, 6, 7 are highly connected and they show similarity in two of their three measurements. In such scenarios, applying full-space clustering leads to questionable clustering results since irrelevant dimensions strongly obfuscate the clusters. Subspace clustering methods solve this problem by finding clusters in their locally relevant subspace projections of the attribute data [7]. Consequentially, recent approaches [12, 2–4] combine the paradigms of *dense subgraph mining* and *subspace clustering*.

These methods enable us to detect more meaningful clusters in the data, like, e.g., the sensor group 3, 4, 6, 7 in Fig. 1. However, combining the paradigms of dense subgraph mining and subspace clustering poses several efficiency challenges. First, analyzing subspace projections is inherently hard since the number of subspaces grows exponentially in the number of attributes. Second, as shown in [2], to obtain high quality clusterings, an unbiased synthesis of both paradigms has to be conducted. Thus, the clustering process has to realize a complex optimization to fairly trade off the cluster properties 'size', 'density', and 'dimensionality'. Last, often an overlap between clusters is reasonable since objects can belong to multiple clusters when regarding different attribute subsets. Musicians of an orchestra, e.g., may share similar musical interests but probably will practice sports with different persons. However, if the clustering

model allows clusters to overlap, it is indispensable to avoid redundancy induced by highly overlapping clusters. As known from usual subspace clustering, redundancy elimination is highly complex [11, 13].

As we have seen so far, for a proper combination of subspace clustering and dense subgraph mining, a model has to handle numerous aspects. Although mostly not accommodating all requirements, previous approaches already have high runtime and space consumptions. Thus, an execution on large datasets (if possible at all) is not efficient. In our work we deal with all the aforementioned aspects, but lay special focus on the *efficiency challenges*.

We start by taking the idea of GAMER [2] to the next level. While GAMER restricts the underlying clustering model to just greedily select good clusters for the result, which does not necessarily result in the most interesting clustering, we aim for a globally optimizing clustering model. Since even the previous models are rarely efficiently computable, it is not surprising that such a model, aiming at a global optimization, has a high complexity. We therefore analyze our model's complexity to identify the most critical parts, which inhibit an efficient execution. We substitute these critical parts through highly efficient heuristics that, however, influence the clustering quality only marginally. Thorough experiments demonstrate that our algorithm not only is far superior to all other approaches in terms of runtime but also shows better quality in nearly all experiments. Our main contributions are: (a) We develop a novel clustering model for a result having globally maximal quality, allowing clusters to overlap in general, and avoiding redundancy (b) We propose the efficient algorithm EDCAR exploiting the GRASP principle and approximating the optimal result.

2 Related Work

Recently, clustering methods have been introduced analyzing *graph data in combination with attribute data*. [6] transforms the network into a distance and combines it with the original feature distance. Afterwards any distance-based clustering method can be applied. The clusters are difficult to interpret since they do not have to obey a certain graph structure. In [19] the attribute information is transformed into a graph and densely connected subgraphs are mined by combining this novel graph with the original one. The work of [18] uses a combined objective function extending the modularity idea. All three approaches [6, 19, 18] perform full-space clustering on the attributes. [21] enriches the graph by further nodes corresponding to (categorical) attribute values and connects them to nodes showing this value. The clustered objects are only pairwise similar and no specific relevant dimensions can be defined. Furthermore, the previous methods determine disjoint clusters.

Only a few approaches deal with *subspace clustering and dense subgraph mining*. CoPaM's [12] combination of both paradigms, however, is not sound since it solely maximizes the number of nodes; the density of subgraphs and the subspace dimensionality are incidental. Furthermore, CoPaM does not eliminate redundancy, which fast leads to an overwhelming result size. The GAMER approach [2]

simultaneously considers the density, the size, and the dimensionality of clusters by trading off these characteristics. Furthermore, GAMER uses a redundancy model to confine the result to a manageable size. A disadvantage, however, is the simple determination of the final clustering: GAMER does not globally examine the result but simply successively adds (in a greedy manner) clusters to the result. Thereby, the resulting clustering does not necessarily correspond to the most interesting one. In [3, 4] a cluster definition has been introduced for finding arbitrarily shaped subspace clusters in graphs with feature vectors. The work uses the same redundancy model as proposed in [2].

The major drawback of all methods is their high runtime and large space requirement, which prevents an application on larger datasets.

3 Maximum Quality Clustering

EDCAR (**E**fficient **D**etermination of **C**lusters regarding **A**tttributes and **R**elationships) realizes a novel clustering model. The model is based on the cluster definition introduced and already verified for its effectiveness in GAMER [2]. The input of our model is a vertex-labeled graph $G = (V, E, l)$ with vertices V , edges $E \subseteq V \times V$ and a labeling function $l : V \rightarrow \mathbb{R}^d$ where $Dim = \{1, \dots, d\}$ is the set of dimensions. We assume an undirected graph without self-loops. We use $l(O) = \{l(o) \mid o \in O\}$ to denote the set of vectors that is associated to the set of vertices $O \subseteq V$.

3.1 Clustering Model

Our method combines objectives from subspace clustering and dense subgraph mining. Thus, the desired clusters are sets of objects $O \subseteq V$ that are meaningful subspace clusters in the attribute space and also form dense subgraphs within the input graph. For identifying subspace clusters, we adapt the cell-based model of DOC [15]. According to this definition the values of all objects in a subspace cluster vary at most by a threshold w in the relevant dimensions. For identifying dense subgraphs, we use the definition of quasi-cliques [8]. The density of a quasi-clique is determined by $\gamma(O) = \frac{\min_{v \in O} \deg^O(v)}{|O|-1}$, where $\deg^O(v) = |\{o \in O \mid (v, o) \in E\}|$ is the vertex degree restricted to the set O .

Definition 1. (Twofold cluster [2]) *A twofold cluster $C = (O, S)$ is a set of vertices $O \subseteq V$ and a set of dimensions $S \subseteq Dim$ with the following properties*

- $(l(O), S)$ is a subspace cluster with dimensionality $|S| \geq s_{min}$
- O is a quasi-clique with density $\gamma(O) \geq \gamma_{min}$
- the induced subgraph of O is connected and $|O| \geq n_{min}$

The resulting clusters are meaningful in the attribute space as well as in the graph. For example in Fig. 2 (choosing $w = 0.5$, $n_{min} = 3$, $\gamma_{min} = 0.4$ and $s_{min} = 2$) the vertex set $C_1 = \{v_1, v_2, v_4, v_5, v_6, v_7\}$ is a valid twofold cluster with

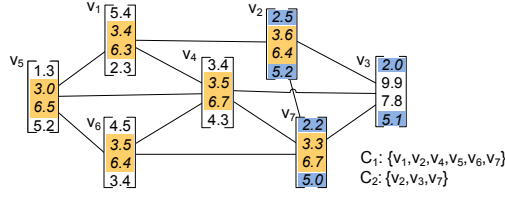


Fig. 2. Exemplary twofold clusters

the relevant dimensions 2 and 3 (marked in orange). The set $C_2 = \{v_2, v_3, v_7\}$ is a twofold cluster with the relevant dimensions 1 and 4 (marked in blue).

Based on the above definition, the number of node sets fulfilling this definition is potentially very large and probably many clusters will overlap (e.g. C_1 and C_2 in Fig. 2). Furthermore, some subsets of the clusters are twofold clusters as well, e.g. $\{v_1, v_4, v_5, v_6\}$. Though it makes sense to allow overlapping clusters in general since one node can belong to several meaningful groups, clusters that are too similar to each other often contain nearly the same information.

Since these redundant clusters are not beneficial but obstructing, they should be excluded from the result. To identify a redundant cluster C w.r.t. another cluster C' , several properties have to apply. First, the structural information of the corresponding clusters has to be similar, i.e. they have to share a large portion of their vertices and their dimensions. Second, the cluster C should be less interesting than the cluster C' ; otherwise one would prefer C . Formally, the redundancy of C w.r.t. C' is based on the following relation:

Definition 2. (Redundancy relation) *Given the redundancy parameters $r_{obj} \in [0, 1]$ and $r_{dim} \in [0, 1]$, the binary redundancy relation \prec_{red} is defined by:*

For all twofold clusters $C = (O, S), C' = (O', S')$:

$$C \prec_{red} C' \Leftrightarrow Q(C) < Q(C') \wedge \frac{|O \cap O'|}{|O|} \geq r_{obj} \wedge \frac{|S \cap S'|}{|S|} \geq r_{dim}$$

Using the parameters r_{obj} and r_{dim} the user can determine to which extent two clusters may overlap without being defined as redundant. For example, with $r_{obj} = r_{dim} = 0.5$ the cluster C_2 would not be redundant w.r.t. C_1 . Although many of C_2 's nodes are covered by C_1 , the relevant dimensions of the two clusters do not overlap. With the values $r_{obj} = 0.5$ and $r_{dim} = 0$, C_2 would be redundant w.r.t. C_1 .

Cluster selection based on global optimization. Our goal in selecting the final clustering is a solution, that (a) does not contain clusters that are redundant to each other, i.e. it has to be redundancy-free, and (b) is most interesting. While the GAMER method greedily selects clusters according to their quality, we perform a more sophisticated selection. Instead of deciding locally which cluster to select next for the result, we perform a global optimization to get the most interesting clustering. We, thus, do not prefer the selection of *single* interesting clusters, since that carries the risk of selecting only uninteresting clusters afterwards, but we select the *overall* most interesting clustering. Correspondingly, we require of our *Result* that the sum of its clusters' qualities is maximal compared

| | | | | | | | | | | |
|------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|------------|
| EDCAR | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | sum = 39.7 |
| GAMER | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | sum = 29.6 |
| cluster | C ₁ | C ₂ | C ₃ | C ₄ | C ₅ | C ₆ | C ₇ | C ₈ | C ₉ | |
| quality | 9.0 | 8.2 | 7.1 | 6.9 | 5.2 | 4.3 | 4.2 | 3.8 | 3.4 | |
| redundancy | | | | | | | | | | |

Fig. 3. Global optimization in EDCAR vs. greedy selection in GAMER

to all other possible clusterings. Formally, the maximum quality clustering is defined as follows:

Definition 3. (Maximum quality clustering) *Given the set of all twofold clusters $Clusters$, the maximum quality clustering $Result \subseteq Clusters$ fulfills*

- (redundancy-freeness) $\neg \exists C_i, C_j \in Result : C_i \prec_{red} C_j$
- (maximum quality sum) $\neg \exists Res' \subseteq Clusters : Res' \text{ fulfills the redundancy-free property and } \sum_{C_i \in Res'} Q(C_i) > \sum_{C_i \in Result} Q(C_i)$

Fig. 3 shows an example for the final clusterings of GAMER and EDCAR: Nine clusters, their quality values, and the redundancy relation are illustrated. The quality sums of the overall clusterings are depicted on the right. GAMER selects the cluster C_2 since it is not redundant w.r.t. any other cluster. A greedy selection according to the quality values is performed. In EDCAR, cluster C_2 is not selected for the final clustering. While C_2 has a high quality itself, its admittance would prohibit the clusters C_3 , C_4 , and C_6 . However, by including these clusters and excluding C_2 , our final clustering has a higher quality (39.7 vs. 29.6). As the example illustrates, EDCAR optimizes the interestingness of the overall clustering, which can yield better results but is computationally more challenging.

3.2 Complexity Analysis

The complexity of our clustering model is given by the following two theorems (proofs on the web). First, the overall complexity of our model, i.e. of generating the twofold clusters *and* selecting the maximum quality clustering, is #P-hard.

Theorem 1. *Given a vertex-labeled graph $G = (V, E, l)$, determining the maximum quality clustering according to Def. 3 is #P-hard w.r.t. $|V|$.*

Second, even if the set of twofold clusters $Clusters$ is given, selecting the maximum quality clustering $Result \subseteq Clusters$ (cf. Def. 3) is NP-complete w.r.t. the input size.

Theorem 2. *Given a set of twofold clusters $Clusters$, selecting the maximum quality clustering according to Def. 3 is NP-complete w.r.t. $|Clusters|$.*

Conclusions. From Theorem 1 we can infer that the input size $|Clusters|$ can be exponential in $|V|$. Overall we can identify *two parts* that, especially

Algorithm 1. generateCandidates()

```

1:  $Cands \leftarrow \{\}$ 
2: for (  $1 \dots maxClus$  )
3:    $\alpha \leftarrow \text{rand}([0 \dots 1])$  // trade off greedy & randomized
4:    $O \leftarrow \{\}$  // root of set enumeration tree
5:   while ( true ) // generate path  $P$ 
6:     determine  $cand_O$  and prune (cf. [2])
7:     if (  $cand_O = \emptyset$  ) break;
8:     determine  $RCL_O$  as in Equation 1
9:      $v \leftarrow \text{rand}(RCL_O)$ 
10:     $O \leftarrow O \cup \{v\}$  // extend path  $P$  by  $v$ 
11:    select cluster  $C^+$  along path  $P$  with highest quality
12:     $Cands \leftarrow Cands \cup \{C^+\}$ 
13: return  $Cands$ 

```

path selection is illustrated in Fig. 5 by the solid lines. Even though this method reduces the number of clusters considerably, this set is still unnecessarily large and will be reduced in a second step. Since along each selected path the object sets successively grow by one vertex (cf. Fig. 4), the clusters along this path are most likely redundant to each other. Using all these clusters as the input for the selection step is needless since most clusters will be discarded anyway. Based on the definition of our redundancy relation we know that clusters with high quality are preferred. Thus, instead of using all clusters, we select along each path just the cluster with the highest quality (cf. dots in Fig. 5). Overall, we realize by our two strategies that only few candidates are used as the input for the cluster selection step. In the remaining of this section, we present more details on our path selection technique.

GRASP for efficient path selection. To systematically (and efficiently) achieve a selection of interesting paths, we adapt the GRASP principle (Greedy Randomized Adaptive Search Procedure) [14, 16]. Naively, one could randomly determine a path. This, however, does not assure to generate high quality clusters. Alternatively, one could decide at node O which successor $v \in cand_O$ (potentially) leads to a good cluster and one descends in the subtree with the highest potential. We use a function $g(v|O)$ to estimate the potential of each node $v \in cand_O$ w.r.t. the current set O . The definition of $g(v|O)$ will be derived in the next subsection. This approach corresponds to a greedy construction of the path. The huge advantage of this greedy method is that the graph structure and the cluster definition can be incorporated into the estimation function g to rate the potential of the path. Thus, it corresponds to an *informed search* and high quality clusters can be expected. Disadvantageously is the risk of reaching only local maxima and generating always very similar paths. These problems do not hold for the randomized approach, which is able to generate a *diversity of paths* in an *uninformed* fashion.

To exploit the advantages of both methods (informed and randomized search), we use the GRASP principle, which acts as a metaheuristic to combine them. Several studies show that this principle often leads to optimal or nearly optimal solutions [14]. According to the GRASP principle, we first construct a restricted candidate list (RCL) corresponding to a *set* of potentially meaningful vertices

for expanding the path. Afterwards, we randomly select one vertex $v \in RCL_O$ to descend into a subtree. Formally,

$$RCL_O = \{v \in cand_O \mid g(v|O) \geq g_m + \alpha \cdot (g_M - g_m)\} \quad (1)$$

with $g_m = \min_{v \in cand_O} \{g(v|O)\}$ and $g_M = \max_{v \in cand_O} \{g(v|O)\}$.

By choosing $\alpha=1$ we can simulate the greedy approach whereas $\alpha=0$ corresponds to a completely randomized selection. The determination of a single path is shown in Algorithm 1, line 5 to 10: In line 6 we determine the candidate set $cand_O$ for the current vertex set O , which is then reduced using the pruning techniques from [2]. Next we determine the RCL as described above and add a randomly selected node to O . This procedure is repeated until $cand_O = \emptyset$, i.e. no more nodes can be added to the path.

As depicted in Fig. 5 we want to descend in several paths. This is done by line 2 and we use a randomly determined α to trade off the two GRASP principles for each novel path, leading to more stable results [16]. Overall, we efficiently generate different paths containing high quality clusters based on the estimated potential.

Potential of Paths. At last, we have to determine the potential of a subtree. Since our goal is to maximize the sum of qualities, we want to use the quality as our estimation function $g(v|O)$. If efficiency was not required, one could use $g'(v|O) = \gamma(O \cup \{v\})^a \cdot |O \cup \{v\}|^b \cdot |S(O \cup \{v\})|^c$ where $S(X)$ denotes the subspace of the corresponding vertex set. However, efficiency is crucial in our case. While the size and the subspace can be efficiently determined, the exact density $\gamma(O \cup \{v\})$ is computationally expensive. Keep in mind that $g(v|O)$ has to be evaluated for each $v \in cand_O$. Therefore, we approximate the density $\gamma(O \cup \{v\})$ of the potential cluster $O \cup \{v\}$ by the lower bound $\bar{\gamma}(O, v) := \frac{\min_{o \in O} \{deg^O(o), deg^O(v)\}}{|O|} \leq \gamma(O \cup \{v\})$. The density approximation $\bar{\gamma}(O, v)$ can be efficiently computed for different vertices v because it is mostly independent of v . We only have to compute the term $deg^O(v)$. Our overall estimation function is $g(v|O) = \bar{\gamma}(O, v)^a \cdot (|O| + 1)^b \cdot |S(O \cup \{v\})|^c$.

4.2 GRASP for Efficient Clustering Selection

So far, we reduced the number of candidates used as the input for the cluster selection step. Now we approximate the maximum quality clustering (Def. 3)) itself since its determination is NP-hard. We again use the GRASP principle. Therefore, we relax Def. 3 by only demanding the resulting clustering Res to be redundancy-free and maximal: $(\neg \exists C, C' \in Res : C \prec_{red} C') \wedge (\forall C \in Cands \setminus Res : \exists C' \in Res : C \prec_{red} C' \vee C' \prec_{red} C)$. Starting with an empty result Res , we now successively add further clusters $C \in Cands$ based on the GRASP principle. Since our goal is to find clusterings with a high quality sum, we instantiate the estimation function $h(C|Res)$, which assesses the potential of adding C to Res , by the quality of clusters: $h(C|Res) = Q(C)$. This value is already given at this time. Thus, no additional computation has to be done. The

Algorithm 2. selectClustering(...)

```

1: input: set of clusters  $Cands$ 
2:  $Res \leftarrow \{\}$  // (preliminary) result
3:  $\alpha \leftarrow \text{rand}([0 \dots 1])$ 
4: while(  $Cands \neq \emptyset$  )
5:    $g_m = \min_{C \in Cands} h(C|Res)$ ,  $g_M = \max \dots$ 
6:    $RCL = \{s \in Cands \mid h(C|Res) \geq g_m + \alpha(g_M - g_m)\}$ 
7:    $C \leftarrow \text{rand}(RCL)$ 
8:    $Res \leftarrow Res \cup \{C\}$ 
9:    $Cands \leftarrow \{C \in Cands \mid \neg \exists C' \in Res : C \prec_{red} C' \vee C' \prec_{red} C\}$ 
10: // improve clustering by local search
11: for(  $NewRes \in Neighborhood(Res)$  )
12:   if(  $Q(NewRes) > Q(Res)$  ) // higher quality sum
13:      $Res \leftarrow NewRes$ 
14:   goto line 10 // efficient first-improving strategy
15: return  $Res$ 

```

Algorithm 3.EDCAR algorithm

```

1:  $Res \leftarrow \{\}$  // preliminary result
2: do
3:    $Cands \leftarrow \text{generateCandidates}()$ 
4:    $Tmp \leftarrow Res \cup Cands$ 
5:    $Res \leftarrow \text{selectClustering}(Tmp)$ 
6: while(  $Cands \neq \emptyset$  )
7: return  $Res$ 

```

pseudo code for selecting the subset is given in Algorithm 2. Since our model requires redundancy-freeness, we are able to remove in line 9 all clusters that induce such a redundancy. These clusters can no longer be added to Res .

To further increase the overall quality, we conduct in line 10-14 a local search on the set of valid clusterings. The idea is to replace a cluster $C \in Res$ by a set of not yet selected clusters New_C to get the potentially better clustering $Res \setminus \{C\} \cup New_C$. The set New_C is built by collecting clusters from $Cands \setminus Res$, in decreasing order w.r.t. their quality values, as long as the redundancy-freeness property of the overall result $Res \setminus \{C\} \cup New_C$ is not violated. Thus, an efficient greedy approach can be used to generate New_C . Formally, for each cluster $X \in Cands \setminus Res$ not selected for New_C it holds: $X \notin New_C \Leftrightarrow \exists C' \in New_C : (X \prec_{red} C') \vee \exists C' \in Res \setminus \{C\} : (X \prec_{red} C' \vee C' \prec_{red} X)$. The overall neighborhood of a clustering Res is the whole set of such generated alternatives $Neighborhood(Res) = \{Res \setminus \{C\} \cup New_C \mid \forall C \in Res\}$. If no better clustering in the neighborhood exists, we have reached a local maximum and the cluster selection in this iteration is finished.

4.3 Overall Processing Scheme

The two phases of our method, generating a small number of candidates and selecting the resulting clustering based on these candidates, lead to an overall efficient execution. Since we select just the one cluster with the highest quality along each path, however, lower quality clusters do not get the chance to be selected for the result. Nevertheless, also clusters with lower qualities can contribute to the overall result, as C_9 in Fig. 3. To give these low-quality but valuable clusters the chance to be considered as result candidates, we repeat both phases recurrently (cf. Algorithm 3).

In the first iteration no information is given ($Res = \emptyset$) and we select the clusters with highest quality along each path. In subsequent iterations Res is used to avoid considering redundant candidates. We thus block redundant parts of a path and select the most interesting cluster among the remaining non-redundant ones as additional candidate. Overall, we generate in each iteration only candidates

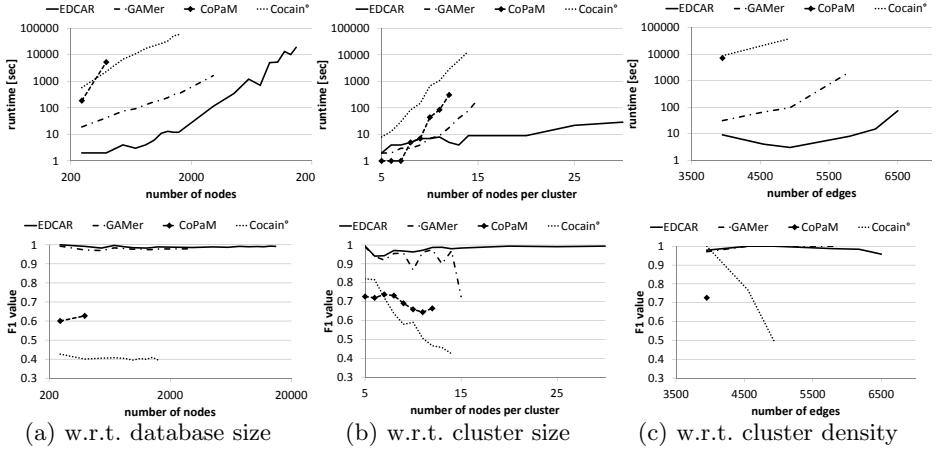


Fig. 6. Scalability and Quality w.r.t. different data or ground truth characteristics

fulfilling $\forall C \in Cands : \neg \exists C' \in Res : C \prec_{red} C'$. It is very likely that some of these clusters can be added to the final clustering. Thus, we perform the clustering selection phase on the enriched set $Cands \cup Res$ to get the novel preliminary result $Res^* \subseteq Cands \cup Res$. Overall, our processing interweaves the generation and the selection of clusters by cyclically invoking both phases. The method automatically terminates if no further non-redundant candidates can be found.

5 Experiments

We compare EDCAR with GAMER [2] and CoPaM [12]; both consider subspaces and dense subgraphs. As a further competitor we use the extension Cocain^o of Cocain [20] as described in [2]. Efficiency is measured by the approaches' runtime. All experiments were conducted on Opteron 2.3GHz CPUs using Java6 64 bit. Methods that did not finish within two days were aborted. Clustering quality is calculated via the F1 value [5]. We use several public real world datasets and synthetic data, by default with 80 clusters, each with 15 nodes, a density of 0.6 and 5-10 relevant dimensions out of 20 dimensions. 6% of the clusters' nodes overlap. *We provide all datasets with descriptions, executables, and parameter settings on our website.*

Database size. First, we vary the number of vertices in the graph by increasing the number of clusters and keeping the number of objects per cluster fixed. As depicted in Fig. 6(a) (top), EDCAR is several orders of magnitude faster than all competing approaches (note the logarithmic scale on both axes). EDCAR is the only method applicable on large data sets. Especially CoPaM is no longer executable for these settings since its limited redundancy model leads to an impracticably large amount of clusters. GAMER scales worse than EDCAR, too. It has to analyze the complete set of all twofold clusters, leading to a high runtime.

Although EDCAR uses an even more complex clustering model, the runtime is lower since we systematically generate and select only the most interesting clusters.

Besides EDCAR’s high efficiency, we observe in Fig. 6(a) (bottom) its high effectiveness. Despite the used approximations, the quality of EDCAR is similar or even higher than that of GAMER. The remaining approaches CoPaM and Cocain° achieve only low qualities, as also shown in [2]. This experiment has shown that EDCAR is also applicable on large datasets. Though, for the following experiments we chose medium-sized datasets to enable a comparison with the other algorithms.

Cluster size. In this experiment, we keep the number of clusters fixed but increase the number of vertices per cluster. This setting is more challenging since larger clusters correspond to longer paths in the set enumeration tree. In Fig. 6(b) (top) we observe only a slow increase in runtime for EDCAR. Since the number of hidden clusters is fix, the number of required iterations in EDCAR is almost constant (about 15). All competing approaches show heavily increasing runtimes and are not applicable at an early stage. Fig. 6(b) (bottom) shows nearly perfect quality for EDCAR. The qualities of the other methods decrease to different extents. The advantage of our novel clustering model becomes apparent.

Graph density. In Fig. 6(c) we increase the graph’s density by adding edges between the clustered nodes. Again, EDCAR is orders of magnitudes faster confirming the usefulness of our solution.

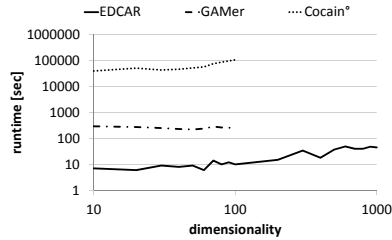


Fig. 7. Data dimensionality

Data dimensionality. In Fig. 7 we increase the data’s dimensionality. Though the runtimes of the competing methods do not increase significantly, they are not applicable for larger datasets due to the extreme memory usage. CoPaM is not applicable at all. The other methods have to manage a tremendous amount of clusters whereas our algorithm generates incrementally a small set of clusters. Overall, all experiments indicate that EDCAR achieves far better runtimes than all competitors. EDCAR is the only method applicable to large datasets. At the same time the results of our approximation achieve high effectiveness.

Real world data. We use *gene data*² and their interactions (3548 nodes; 8334 edges; 115d), an extract of the *Arxiv database*³ (27769; 352284; 300d), *patent information*⁴ (492007; 528333; 5d), and a co-author graph extracted out of the *DBLP database*⁵ (133097; 631384; 2695d). Since for real world data no hidden clusters are given, we analyze in Fig. 8 different properties of the clustering results (runtime, avg. number of vertices, density, dimensionality of the detected clusters). For all datasets EDCAR is orders of magnitude faster than the other methods. GAMER is only applicable on three of the datasets. CoPaM can only be executed on the gene data and achieves extremely high runtimes. Cocain^o finished on none of the datasets within 2 days. The clusters identified by EDCAR and GAMER have nearly similar properties. Thus, our approximations do not impair the clustering quality.

| | Gene | | | Arxiv | | Patent | | DBLP |
|----------------------|-------|-------|-------|-------|-------|--------|-------|-------|
| | EDCAR | GAMER | CoPaM | EDCAR | GAMER | EDCAR | GAMER | EDCAR |
| runt. [s] | 49 | 76 | 33060 | 179 | 945 | 282 | 574 | 25752 |
| \emptyset vertices | 9 | 8.8 | 9.67 | 7.7 | 8.2 | 12 | 11.7 | 9.3 |
| \emptyset density | 0.74 | 0.72 | 0.24 | 0.69 | 0.6 | 0.6 | 0.6 | 0.83 |
| \emptyset dim. | 15.8 | 15.47 | 12.21 | 5 | 5 | 3.0 | 3.0 | 3.02 |

Relevant dimensions

- IEEE ICME
- ACM Multimedia
- TREC

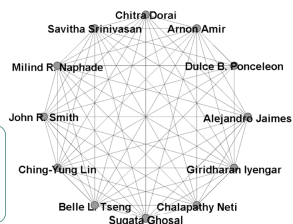


Fig. 8. Clustering properties for real world data **Fig. 9.** Exemplary cluster for DBLP

An exemplary cluster from EDCAR’s clustering result on the DBLP co-author graph is shown in Fig. 9. Here, each node represents an author, each edge corresponds to a co-authorship, and the 2695 attributes of a node indicate the conferences which an author has attended. Fig. 9 illustrates a cluster consisting of 12 authors who jointly published papers at the conferences IEEE ICME, ACM Multimedia, and TREC (i.e., the cluster is located in a 3d subspace). Please note that the cluster does not form a clique (its quasi-clique density is 0.64), thus not all authors collaborated together. EDCAR is the only method that can handle this data set; all competing methods fail due to their high runtime and space complexity.

6 Conclusion

We introduced the method EDCAR for efficiently detecting clusters showing high density in graphs as well as feature similarity in subspace projections. Our model combines subspace clustering with dense subgraph mining and performs

² <http://thebiogrid.org>, <http://genomebiology.com/2005/6/3/R22>

³ <http://www.cs.cornell.edu/projects/kddcup/datasets.html>

⁴ <http://www.nber.org/patents/>

⁵ <http://dblp.uni-trier.de>

an overall optimization of the result to get the most interesting, redundancy-free clustering. Based on the proven complexity of our model, we developed the algorithm EDCAR to efficiently calculate an approximate solution. By interweaving the process of cluster generation and cluster selection, which both make use of the GRASP principle, EDCAR determines high quality clusters and ensure low runtimes. Thorough experiments demonstrate that EDCAR has high effectiveness and at the same time constantly outperforms all competing approaches in terms of efficiency.

Acknowledgments. This work has been supported by the UMIC Research Centre, RWTH Aachen University, Germany.

References

1. Aggarwal, C., Wang, H.: *Managing and Mining Graph Data*. Springer (2010)
2. Günnemann, S., Färber, I., Boden, B., Seidl, T.: Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In: *ICDM*, pp. 845–850 (2010)
3. Günnemann, S., Boden, B., Seidl, T.: DB-CSC: A density-based approach for subspace clustering in graphs with feature vectors. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) *ECML PKDD 2011, Part I. LNCS (LNAI)*, vol. 6911, pp. 565–580. Springer, Heidelberg (2011)
4. Günnemann, S., Boden, B., Seidl, T.: Finding density-based subspace clusters in graphs with feature vectors. *Data Min. Knowl. Discov.* 25(2), 243–269 (2012)
5. Günnemann, S., Färber, I., Müller, E., Assent, I., Seidl, T.: External evaluation measures for subspace clustering. In: *CIKM*, pp. 1363–1372 (2011)
6. Hanisch, D., Zien, A., Zimmer, R., Lengauer, T.: Co-clustering of biological networks and gene expression data. *Bioinformatics* 18, 145–154 (2002)
7. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD* 3(1), 1–58 (2009)
8. Liu, G., Wong, L.: Effective pruning techniques for mining quasi-cliques. In: Daelemans, W., Goethals, B., Morik, K. (eds.) *ECML PKDD 2008, Part II. LNCS (LNAI)*, vol. 5212, pp. 33–49. Springer, Heidelberg (2008)
9. Long, B., Wu, X., Zhang, Z.M., Yu, P.S.: Unsupervised learning on k-partite graphs. In: *KDD*, pp. 317–326 (2006)
10. Long, B., Zhang, Z.M., Yu, P.S.: A probabilistic framework for relational clustering. In: *KDD*, pp. 470–479 (2007)
11. Moise, G., Sander, J.: Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In: *KDD*, pp. 533–541 (2008)
12. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining cohesive patterns from graphs with feature vectors. In: *SDM*, pp. 593–604 (2009)
13. Müller, E., Assent, I., Günnemann, S., Krieger, R., Seidl, T.: Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In: *ICDM*, pp. 377–386 (2009)
14. Pitsoulis, L., Resende, M.: Greedy randomized adaptive search procedures. In: *Handbook of Applied Optimization*, pp. 168–183. Oxford University Press, New York (2002)

15. Procopiuc, C., Jones, M., Agarwal, P., Murali, T.: A Monte Carlo algorithm for fast projective clustering. In: SIGMOD, pp. 418–427 (2002)
16. Resende, M.G.C., Ribeiro, C.C.: Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications. Int. Series in Op. Research & Management Science, pp. 283–320 (2010)
17. Rymon, R.: Search through systematic set enumeration. In: KR, pp. 539–550 (1992)
18. Shiga, M., Takigawa, I., Mamitsuka, H.: A spectral clustering approach to optimally combining numerical vectors with a modular network. In: KDD, pp. 647–656 (2007)
19. Ulitsky, I., Shamir, R.: Identification of functional modules using network topology and high-throughput data. BMC Systems Biology 1(1) (2007)
20. Zeng, Z., Wang, J., Zhou, L., Karypis, G.: Coherent closed quasi-clique discovery from large dense graph databases. In: KDD, pp. 797–802 (2006)
21. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. PVLDB 2(1), 718–729 (2009)