Abstract. Dense subgraph discovery is a key issue in graph mining, due to its importance in several applications, such as correlation analysis, community discovery in the Web, gene co-expression and protein-protein interactions in bioinformatics. In this work, we study the discovery of the top-\(k\) dense subgraphs in a set of graphs. After the investigation of the problem in its static case, we extend the methodology to work with dynamic graph collections, where the graph collection changes over time. Our methodology is based on lower and upper bounds of the density, resulting in a reduction of the number of exact density computations. Our algorithms do not rely on user-defined threshold values and the only input required is the number of dense subgraphs in the result (\(k\)). In addition to the exact algorithms, an approximation algorithm is provided for top-\(k\) dense subgraph discovery, which trades result accuracy for speed. We show that a significant number of exact density computations is avoided, resulting in efficient monitoring of the top-\(k\) dense subgraphs.

1 Introduction

Many modern applications require the management of large volumes of graph data. Graphs are very important in scientific applications such as bioinformatics, chemoinformatics, link analysis in social networks, to name a few. Dense subgraph discovery is a fundamental graph mining task [1,5] with increasing importance. Density is a significant property of graphs, because it is highly related to how well a graph is connected and can be used as a measure of the graph coherence. Usually, the densest the graph the more likely that the connectivity among the graph nodes will be higher.

Among the various density definitions, we adopt the one that relates the graph density to the average degree [7]. More formally, for a graph \(G(V,E)\), where \(V\) is the set of nodes and \(E\) the set of edges, the density of \(G\), denoted as \(den(G)\), is given by the number of edges over the number of nodes, i.e., \(den(G) = |E|/|V|\). For the rest of the work, we focus on undirected and unweighted graphs. Generalizations to other graph classes are performed easily with appropriate modifications. Figure 1 depicts some examples of density computation.

Algorithms for dense subgraph discovery that have been proposed in the literature require one or more constraints to be defined by the user [2]. For example,
all subgraphs having a density value higher than a threshold are reported back to the user. The major limitation of such an approach is that the determination of threshold values is difficult: i) if the threshold is too low, then we risk a cumbersome answer set and ii) if the threshold is too high then an empty answer set may be returned. A meaningful threshold value may be difficult to define without at least a limited a priori knowledge regarding the nature of the data (e.g., distribution of densities). To overcome this limitation, in this work, we focus on dense subgraph discovery by taking a top-$k$ oriented approach. More specifically, the only input required by our methods is the number $k$ of the dense subgraphs. The basic benefit of this approach is that the cardinality of the answer set is always known and is equal to $k$, thus, no surprises are expected regarding the number of answers. In addition, no “wild guesses” regarding the density threshold need to be performed by the user.

The second novelty of our research is that we focus on dynamic graph collections that take the form of a stream of graphs. More specifically, computation is performed on a count-based sliding window of size $w$, defined over a stream of graphs $G_1, G_2, \ldots, G_w$. New graph objects may arrive whereas old ones expire. The value of $w$ depends on the application. The challenge is to monitor the $k$ densest subgraphs induced by the graphs that are currently active in the sliding window. This translates in performing the necessary actions when a new graph arrives or an old one expires. Evidently, computation must be as efficient as possible to avoid significant delays during updates. Previously proposed methods working in graph streams do not consider deletions.

There are many applications that benefit from the support of top-$k$ dense subgraph discovery. In web usage mining, the links followed by users form a directed graph. Many such graphs may be available in a streaming fashion, corresponding to different time instances or different geographical areas. Dense subgraphs are useful in this case because they enable community discovery. The use of the sliding window enables to center our focus at the most recent data available, rather than base the discovery process on the complete history. Mining these graphs on-the-fly is extremely important towards continuous knowledge discovery. As a second example, consider a large social network, where users communicate by means of message exchange. Each graph in this case, corresponds to the interactions among users for a specific time period, e.g., a day. By collecting these graphs for a period of time, we can monitor the evolution of interactions among