Theoretical Computer Science 600 (2015) 59–69

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Complexity of the cluster deletion problem on subclasses of chordal graphs ☆

Flavia Bonomo [a], Guillermo Durán [b,c], Mario Valencia-Pabon [d,*,1]

[a] CONICET and Dep. de Computación, FCEN, Universidad de Buenos Aires, Argentina
[b] CONICET and Dep. de Matemática and Instituto de Cálculo, FCEN, Universidad de Buenos Aires, Argentina
[c] Dep. de Ingeniería Industrial, FCFM, Universidad de Chile, Santiago, Chile
[d] Université Paris-13, Sorbonne Paris Cité LIPN, CNRS UMR7030, Villetaneuse, France

ARTICLE INFO

Article history:
Received 28 October 2014
Received in revised form 30 June 2015
Accepted 1 July 2015
Available online 7 July 2015
Communicated by V.Th. Paschos

Keywords:
Block graphs
Cliques
Edge-deletion
Cluster deletion
Interval graphs
Split graphs
Submodular functions
Chordal graphs
Cographs
NP-completeness

ABSTRACT

We consider the following vertex-partition problem on graphs, known as the CLUSTER DELETION (CD) problem: given a graph with real nonnegative edge weights, partition the vertices into clusters (in this case, cliques) to minimize the total weight of edges outside the clusters. The decision version of this optimization problem is known to be NP-complete even for unweighted graphs and has been studied extensively. We investigate the complexity of the decision CD problem for the family of chordal graphs, showing that it is NP-complete for weighted split graphs, weighted interval graphs and unweighted chordal graphs. We also prove that the problem is NP-complete for weighted cographs. Some polynomial-time solvable cases of the optimization problem are also identified, in particular CD for unweighted split graphs, unweighted proper interval graphs and weighted block graphs.

1. Introduction

Clustering is an important task in the data analysis process. It can be viewed as a data modeling technique that provides an attractive mechanism for automatically finding the hidden structure of large data sets. The input to the problem is typically a set of elements and pairwise similarity values between elements. The goal is to partition these elements into subsets called *clusters* such that two meta-criteria are satisfied: *homogeneity* – elements in a given cluster are highly similar to each other; and *separation* – elements from different clusters have low similarity to each other. In the graph theoretic approach to clustering, one builds from the raw data a *similarity graph* whose vertices correspond to elements and there is an edge between two vertices if and only if the similarity of their corresponding elements exceeds a predefined threshold [13,14]. Cluster graphs have been used in a variety of applications whenever clustering of objects is studied or when consistent data

☆ Partially supported by MathAmSud Project 13MATH-07 (Argentina–Brazil–Chile–France), UBACyT Grant 20020130100808BA, CONICET PIP 112-200901-00178 and 112-201201-00450CO and ANPCyT PICT 2012-1324 (Argentina), FONDECYT Grant 1140787 and Millennium Science Institute "Complex Engineering Systems" (Chile).

* Corresponding author.
E-mail addresses: fbonomo@dc.uba.ar (F. Bonomo), gduran@dm.uba.ar (G. Durán), valencia@lipn.univ-paris13.fr (M. Valencia-Pabon).

[1] Currently "en délégation" at the INRIA Nancy-Grand Est.

http://dx.doi.org/10.1016/j.tcs.2015.07.001
0304-3975/

is sought among noisy or error-prone data [1,5]. Ideally, the resulting graph would be a *cluster graph*, that is, a graph in which every connected component is a clique (i.e., a complete subgraph). In practice, it is only close to being such, since similarity data is experimental and therefore error-prone.

The *cluster deletion* problem consists in finding the minimum number of edges that must be removed from an input graph to make the resulting graph a cluster graph. In its decision version, the cluster deletion problem has a non-negative integer parameter W and asks if one can remove a set of at most W edges from the input graph such that the resulting graph is a cluster graph. There exist several results for the cluster deletion problem (see for example [3,17,22] and references therein). The cluster deletion problem is known to be NP-complete [22] for general graphs. Moreover, Shamir et al. [22] showed that it remains NP-hard when imposing that the input graph should be clustered into exactly $d \geq 3$ components. They also showed that when the input graph is clustered into exactly 2 components, the problem is polynomial-time solvable. Komusiewicz et al. [17] proved that cluster deletion is hard for C_4-free graphs with maximum degree 4 and gave an $O(n^{1.5}\log^2 n)$ time algorithm for solving cluster deletion on graphs with maximum degree 3, where n is the number of vertices of the graph.

Based on results obtained by Demaine et al. [7] for a variant of a clustering problem, Dessmark et al. [8] provided a polynomial $O(\log n)$-approximation algorithm for the edge-weighted version of the cluster deletion problem. In this version, the edges of the graph have an associated weight and the aim is to minimize the sum of the weights of the removed edges. Considering it as a decision problem, the aim is to determine, for some input parameter W, if there is a set of edges with a total weight of at most W such that removing it from the input graph will make the resulting graph a cluster graph. Note that if we allow the weight function to be negative on some edges, we can reduce any clustering problem to a clustering problem whose input graph is a weighted complete graph by assigning a negative weight with a large enough absolute value to the edges that are missing in the original graph. Thus, the problem with arbitrary weights is NP-complete for any graph class admitting arbitrarily large cliques. We will assume throughout that all of the weight functions are nonnegative.

Dessmark et al. [8] also showed that for the unweighted version of cluster deletion on general graphs, the greedy algorithm that finds iteratively maximum cliques gives a 2-approximation algorithm to the optimal cluster deletion. The complexity of such an algorithm reflects the complexity of iteratively finding maximum cliques, so it is a polynomial-time approximation algorithm for certain graph classes. Recently, Gao et al. [11] showed that the greedy algorithm that finds iteratively maximum cliques gives an optimal solution for the class of graphs known as *cographs*. This implies that the cluster deletion problem is polynomial-time solvable on unweighted cographs. With a different approach based on modular decomposition, it is proved in [4] that the unweighted cluster deletion problem is polynomial-time solvable on a subclass of P_4-sparse graphs that strictly includes P_4-reducible graphs (which are, in turn, a superclass of cographs). Gao et al. [11] also showed that the cluster deletion problem is NP-hard on (C_5, P_5)-free graphs, on $(2K_2, 3K_1)$-free graphs and on $(C_5, P_5, \text{bull}, 4\text{-pan}, \text{fork}, \text{co-gem}, \text{co-4-pan})$-free graphs. For weighted graphs, the cluster deletion problem can be solved in polynomial time on the class of triangle-free graphs given that it is equivalent to maximum weighted matching [9]. The cluster deletion and other clustering problems have been studied extensively in the context of fixed-parameter tractability (FPT) ([6,18] and references therein). Many of the recently-developed FPT algorithms rely on being able to solve cluster deletion in polynomial-time on restricted graph structures [3].

A heuristic for solving clustering problems consists in modifying a given input graph into another graph having some nice algorithmic properties and then solving the clustering problem for the modified graph. For example, to solve a genetic clustering problem, Kaba et al. [16] transform any input graph into a chordal graph via minimal triangulations of the former one. Once the input graph has been so transformed, they exploit the algorithmic properties of chordal graphs to obtain good solutions to their clustering problem. If solving a clustering problem for a specific graph family $\mathcal{F}$ is computationally hard, however, the heuristic which first transforms the input graph into a graph in $\mathcal{F}$ and then solves the problem on the resulting graph may not be a good approach. Therefore, it is important to know how to solve a clustering problem on specific graph families before using the above-described heuristic for general input clustering graphs.

Some known results are summarized in Table 1; those obtained in the present work are shown in bold face. We conclude this introduction with some definitions.

Let $G = (V, E)$ be a graph. For each vertex $v \in V$, we denote as $N(v) = \{u : vu \in E\}$ the set of neighbors of v in G. Two vertices v and w are called *true twins* if $N(v) \cup \{v\} = N(w) \cup \{w\}$. A graph G is said to be *weighted* if there is a nonnegative weight function $w : E \to \mathbb{R}^+$ associated with it. For the algorithms involving weighted graphs we will assume that the weights are rational (or belong to any ordered field in which we can perform the field operations and the order comparisons algorithmically). An *unweighted* graph is a graph in which each edge has a weight equal to 1. We say that a set F of edges of a given graph has a *uniform* weight if all the edges in F have the same weight.

Let H and G be graphs. If G contains no induced subgraph isomorphic to H then G is an *H-free graph*. Let P_k (resp. C_k) denote a path (resp. cycle) on k vertices. Let $K_{m,n} = (A \cup B, E)$ denote the complete bipartite graph, where A (resp. B) is an independent set of size m (resp. n) and E is the set of all the edges with an endpoint in A and an endpoint in B. We refer to [23] for standard definitions and results in graph theory. A graph is *chordal* if and only if it does not contain a cycle of length at least four as an induced subgraph. Given a vertex partition $S = C_1, \ldots, C_k$ of a graph G, we call the *weight* of S, denoted $w(S)$, the sum of the weights of all edges $e = uv$ such that $u \in C_i$, $v \in C_j$, with $i \neq j$. An edge is called *external* with respect to the partition S if its endpoints belong to distinct sets of S, and *internal* otherwise. The cluster deletion problem for an (un)weighted graph G can be redefined as the problem of finding a clique partition of G with minimum weight. We will assume throughout that all NP-completeness results concern the decision version of the cluster deletion problem.