SCISPACE
formerly Typeset

Open access • Book Chapter • DOI:10.1007/978-3-319-57351-9_25

Knowledge Discovery in Graphs Through Vertex Separation — Source link

Marc Sarfati, Marc Sarfati, Marc Queudot, Catherine Mancel ...+1 more authors

Institutions: École Polytechnique, Université du Québec à Montréal, École nationale de l'aviation civile

Published on: 16 May 2017 - Canadian Conference on Artificial Intelligence

Topics: Chordal graph, Indifference graph, Maximal independent set, Modular decomposition and Pathwidth

Related papers:

- Continual and Cost-Effective Partitioning of Dynamic Graphs for Optimizing Big Graph Processing Systems
- Relation between the set-complexity of a graph and its structure
- Study on Partitioning Real-World Directed Graphs of Skewed Degree Distribution
- CHAPTER 10 – Graph Algorithms
- An investigation of big graph partitioning methods for distribution of graphs in vertex-centric systems

Share this paper:

View more about this paper here: https://typeset.io/papers/knowledge-discovery-in-graphs-through-vertex-separation-qxiquighwm

Knowledge Discovery in Graphs Through Vertex Separation

Marc Sarfati, Marc Queudot, Catherine Mancel, Marie-Jean Meurs

▶ To cite this version:

Marc Sarfati, Marc Queudot, Catherine Mancel, Marie-Jean Meurs. Knowledge Discovery in Graphs Through Vertex Separation. AI 2017, 30th Canadian Conference on Artificial Intelligence, May 2017, Edmonton, Canada. pp 203-214, 10.1007/978-3-319-57351-9_25 . hal-01521890

HAL Id: hal-01521890
https://hal-enac.archives-ouvertes.fr/hal-01521890

Submitted on 18 May 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Knowledge Discovery in Graphs through Vertex Separation

Marc Sarfati[1,2], Marc Queudot[2], Catherine Mancel[3], and Marie-Jean Meurs[2]

[1] École Polytechnique, Palaiseau, France
[2] Université du Québec à Montréal, Montréal, QC, Canada
[3] ENAC (École Nationale de l'Aviation Civile), Université de Toulouse, Toulouse, France

Abstract. This paper presents our ongoing work on the Vertex Separator Problem (VSP), and its application to knowledge discovery in graphs representing real data. The classic VSP is modeled as an integer linear program. We propose several variants to adapt this model to graphs with various properties. To evaluate the relevance of our approach on real data, we created two graphs of different size from the IMDb database. The model was applied to the separation of these graphs. The results demonstrate how the model is able to semantically separate graphs into clusters.

Keywords: graph partitioning, knowledge discovery, Vertex Separator Problem

1 Introduction

Extracting knowledge from graphs is often performed by clustering algorithms. While several methods are based on random walks or spectral decomposition for example [12], this paper focuses on graph separation through the Vertex Separator Problem (VSP) and its applications. The VSP can be formally defined as follows. Given a connected undirected graph $G = (V, E)$, a *vertex separator* in G is a subset of vertices whose removal disconnects G. In other words, finding a vertex separator is finding a partition of V composed of 3 non-empty classes A, B and C such that there is no edge between A and B, as depicted in Figure 1. C is usually called the separator while A and B are the pillars. The VSP appears in a wide range of applications like very-large-scale integration design [15], matrix factorization [4], bioinformatics [6], or network security [13].

Fig. 1: Example of vertex separation

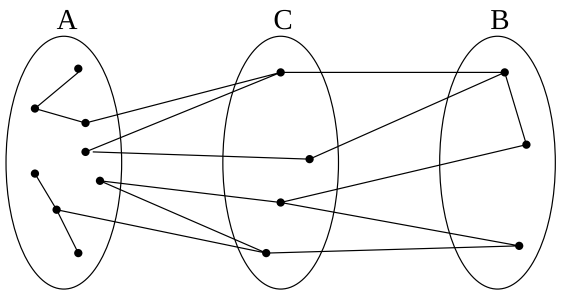
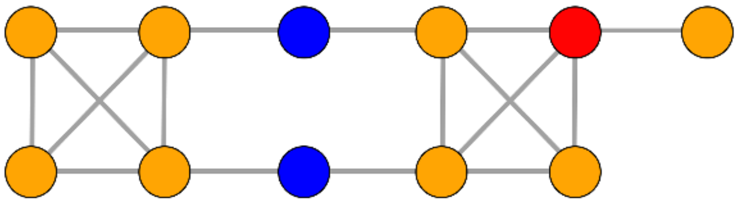


Fig. 2: Trade-off between separator-size and balanced components



The VSP consists in finding a minimum-sized separator in any connected graph. In any finite connected graph which is not complete, one can find a minimum-sized separator since the number of subsets of the vertex set is finite. The VSP is NP-hard [8,3], even if the graph is planar [7]. In [10], Kanevsky provided an algorithm to find all minimum-sized separators in a graph. However, the method is not efficient for computing balanced pillars, especially on large instances. Indeed, computing a minimum-sized separator often leads to finding pillars of highly unbalanced cardinalities. Since most of the applications to real-life problems look for reasonably balanced pillars, adding cardinality constraints often results in finding slightly larger separators. Figure 2 shows on a small graph this trade-off between finding a small-sized separator and finding a balanced result. A minimum-sized separator is only composed of the red vertex, which separates the graph in two subsets whose cardinalities are 9 and 1. Any graph which contains a terminal vertex (*i.e.* which degree is 1). In Figure 2, the separator composed of blue vertices may be a better choice because it separates the graph into balanced dense subsets.

Other approaches to solve the VSP have been recently proposed. Among them, Benlic and Hao present an approximate solution, which uses Breakout Local Search [2]. The algorithm requires an initial solution, and then looks for a local optimum using a mountain climbing method. In order to avoid getting stuck on a local optimum, once the algorithm reaches a local optimum, a perturbation is added to leap to another search zone. A continuous polyhedral solution is proposed by Hager and Hungerford in [9] where the optimization problem relaxes the separation constraint ($|(A \times B) \cap E| = 0$) to include it in the objective function. However, the optimization problem is quadratic thus not easily solvable by a mathematical programming solver. Following the seminal work from Balas and De Souza [1,14], the work we present in this paper is based on the polyhedral approach introduced by Didi Biha and Meurs. In [5], they presented a formulation of a Mixed-Integer Linear Programming (MILP) to solve the VSP. This solution is useful and elegant since it is exact and allows to easily add constraints on the sizes of the pillars.

Our initial model and the proposed variants are presented in the next Section. To evaluate the efficiency of our approach and its ability to provide semantic sets of vertices (pillars), we applied our approach to several graphs extracted from the Internet Movie Database (IMDb)[1]. Section 3 describes these experiments and results, while we conclude and discuss future work in Section 4.

[1] Information courtesy of IMDb (http://www.imdb.com). Used with permission.

2 Solving the VSP

2.1 Initial Model

Since this paper presents an extension of [5], it uses the same notations for the sake of clarity. Let $G = (V, E)$ be a finite undirected graph, the goal is to find a partition (A, B, C) as defined previously. Let **x** and **y** be two vectors indexed by the vertices in V, which are indicators of the presence of a vertex in A or B (*i.e.* $x_u = 1$ if $u \in A$, $x_u = 0$ otherwise). Computing the VSP is equivalent to finding a partition (A, B, C) such as there is no edge between A and B and if n denotes $|V|$, $|A| + |B| = n - |C| = \sum_{v \in V}(x_v + y_v)$ is maximized. The integer linear program is hence formulated as follows:

$$\max_{\mathbf{x},\mathbf{y} \in \{0,1\}^n} \sum_{v \in V} (x_v + y_v) \tag{VSP}$$

Subject to

$$\forall v \in V, x_v + y_v \leq 1 \tag{1}$$

$$\forall (uv) \in E, x_u + y_v \leq 1 \tag{2}$$

$$\forall (uv) \in E, x_v + y_u \leq 1 \tag{3}$$

Note that (VSP) is equivalent to

$$\max_{\mathbf{x},\mathbf{y} \in \{0,1\}^n} |A| + |B| \tag{VSP'}$$

Constraint (1) reflects the fact that a vertex can not simultaneously be in A and in B. (2) and (3) express that there is no edge between A and B. (VSP) subject to (1)-(3) computes a minimum-sized separator of a graph G. An extension of the Menger's theorem defined in [10] states that the size of a separator in a graph is necessarily greater than or equal to the connectivity of the graph. Adding this constraint reduces the size of the polyhedron admissible solutions, shrinking down the computation time needed to solve the model. Let k_G denote the connectivity of the graph, the linear program can be strengthened by adding a new constraint :

$$\sum_{v \in V} (x_v + y_v) \leq n - k_G \tag{4}$$

As seen previously, it can be useful to compute separators which lead to balanced pillars. Didi Biha and Meurs added two constraints to the linear program (VSP) in order to avoid large imbalances. Given $\beta(n)$ an arbitrary number, they forced the cardinalities of the pillars A and B to be smaller than or equal to $\beta(n)$. The literature usually recommends $\beta(n) = 2n/3$ [11].

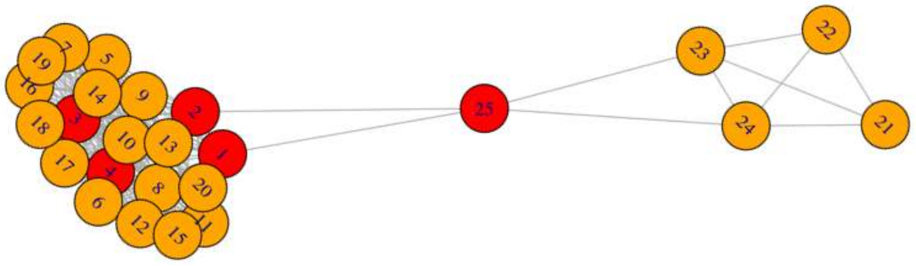
$$\sum_{v \in V} x_v \leq \beta(n) \tag{5}$$

$$\sum_{v \in V} y_v \leq \beta(n) \tag{6}$$

2.2 Relaxing the constraints on the partition size

(VSP) subject to (1)-(6) computes a low-sized separator of a graph while generally avoiding large imbalances in the pillars. However, $\beta(n)$ is arbitrarily chosen so it may lead to inconsistency. The graph in Figure 3 is clearly composed of 2 different dense blocks (cliques) linked by a single vertex (denoted 25). The bigger clique contains more than $2n/3$ vertices, hence the linear program can not choose $\{25\}$ as a separator. In order to satisfy constraints (5) and (6), the solver must add unnecessary vertices in the separator, which are part of a dense connected component.

Fig. 3: Inconsistency due to the choice of $\beta(n)$



In this paper we present a new approach to compute low-sized balanced separator in a graph. Instead of fixing an arbitrary upper bound to the sizes of the pillars, we relax constraints (5) and (6) but we penalize large gaps between $|A|$ and $|B|$ in the objective function. The objective function becomes:

$$\max_{\mathbf{x},\mathbf{y}\in\{0,1\}^n} |A| + |B| - \gamma(|B| - |A|)$$

where $\gamma \in [0,1]$ and

$$|A| \leq |B| \tag{7}$$

Note that because pillars A and B have no predefined meaning, one can set $|A| \leq |B|$ or $|B| \leq |A|$ with no loss of generality.

The objective function can be simplified as follows:
$|A| + |B| + \gamma(|A| - |B|) = (1+\gamma)|A| + (1-\gamma)|B| = (1+\gamma)[|A| + \frac{1-\gamma}{1+\gamma}|B|]$.
Since $(1+\gamma)$ is positive, we can remove this term from the objective function and still find the same solutions **x** and **y**. Moreover $\gamma \mapsto \frac{1-\gamma}{1+\gamma}$ induces a bijection between $[0,1]$ and $[0,1]$, thus the objective function of the relaxed problem is:

$$\max_{\mathbf{x},\mathbf{y}\in\{0,1\}^n} |A| + \lambda|B| \qquad \text{with } \lambda \in [0,1] \tag{VSPr}$$

The new linear program is formulated as (VSPr) subject to (1)-(4) and (7). We can notice that λ measures the trade-off between finding a small size separator and finding balanced pillars. If $\lambda = 1$, (VSPr) is equivalent to the first linear program (VSP) so the solver will output a minimum-sized separator. If $\lambda = 0$ then the program will maximize $|A|$ given (7), thus generating a well balanced partition.

2.3 Adding weights to the vertices

Unlike Kanevsky's algorithm, the linear programming solution allows to take into account vertex weights. Let **c** denote the vector containing the vertex weights. For any subset $S \subset V$, we define the weight of S, $c(S) = \sum_{v \in S} c_v$. The program which uses vertex weights is formulated as is :

$$\max_{\mathbf{x},\mathbf{y} \in \{0,1\}^n} c(A) + \lambda\, c(B) \tag{VSPw}$$

subject to (1)-(4) and

$$c(A) \leq c(B) \tag{8}$$

This version, though taking into account vertex weights, is as simple as the previous one. Note that $c(A)$ (respectively $c(B)$) can be calculated as $\mathbf{c}^\top \mathbf{x}$ (respectively $\mathbf{c}^\top \mathbf{y}$).

2.4 Taking distances into account

We have previously seen that the VSP can be used in order to split a graph into balanced pillars. In such a context, the data points (graph vertices) have relational links (graph edges) but these links can also be associated to intrinsic information which leads to defining metrics, as for example distances between vertices. In order to have more consistent results, it is interesting to take these distances into account. The main goal here is to find a well-balanced partition separated by a small size separator while minimizing intern distances in the resulting pillars.

Let $d : (u, v) \mapsto d(u, v)$ denote the metric between the vertices. Since we only care about the distances between vertices that are both in A or B, we only consider pairs (u, v) where ($x_u = 1 \wedge x_v = 1$) or ($y_u = 1 \wedge y_v = 1$). This means we only take into account the pairs (u, v) where $x_u x_v = 1$ or $y_u y_v = 1$, *i.e.* $x_u x_v + y_u y_v = 1$ because both terms can not be equal to 1.

The constraints of the optimization program remain (1)-(4), (8) but the objective function becomes:

$$\max_{\mathbf{x},\mathbf{y} \in \{0,1\}^n} (1 - \mu)(c(A) + \lambda\, c(B)) - \mu \sum_{\{u,v\} \in V^2} d(u, v)(x_u x_v + y_u y_v) \tag{VSPd}$$

where $\mu \in [0, 1]$. As for λ, μ measures the trade-off between finding a partition with a small size separator and whose weights are balanced, and minimizing the distances within the pillars A and B.

If $\mu = 1$ the solver will minimize the distance within the subsets, so it will surely put all the vertices in the separator C to reach its optimal value. If $\mu = 0$, the optimization program is exactly (VSPw). The higher μ is, the more the optimization program takes the distances into account.

The program (VSPd) is not linear anymore since it contains quadratic terms in the objective function. Common free and commercial solvers are also able to solve Mixed-Integer Non-Linear Programs (MINLP), however the computation time is usually much higher. To linearize (VSPd), we introduce a new family of variables (z_{uv}), for all $(u, v) \in V^2, v > u$. We want z_{uv} to be an indicator whether u and v are both in the same pillar. We can not define $z_{uv} = x_u x_v + y_u y_v$ because it would add quadratic constraints to the linear program. Hence in order to have the family (z_{uv}) be such indicators, we add the following constraints to the linear program.

$$\forall (u,v) \in V^2, v > u, z_{uv} \geq x_u + x_v - 1 \tag{9}$$

$$\forall (u,v) \in V^2, v > u, z_{uv} \geq y_u + y_v - 1 \tag{10}$$

$$\forall (u,v) \in V^2, v > u, z_{uv} \in \{0,1\} \tag{11}$$

The new objective function is :

$$\max_{\mathbf{x},\mathbf{y} \in \{0,1\}^n} (1-\mu)(c(A) + \lambda\, c(B)) - \mu \sum_{\{u,v\} \in V^2} d(u,v) z_{uv} \tag{VSPdist}$$

Let u, v be two different vertices such that $v > u$, if both u and v are in A, then $x_u = 1$ and $x_v = 1$ so necessarily $z_{uv} = 1$. Similarly if u and v are both in B, $z_{uv} = 1$. If u and v are not in the same subset then constraints (9)-(11) force $z_{uv} \geq 0$; since μ and $d(u,v)$ are both positive then the maximization of the objective function leads to $z_{uv} = 0$.

(VSPdist) subject to constraints (1)-(4), (8)-(11) represents a new extension of the Vertex Separator Problem which computes a small size separator and splits the graph into balanced subsets while minimizing the distances within these subsets. Moreover, selecting such a trade-off is easily doable by setting the values of the hyper-parameters λ and μ.

2.5 a, b-separation

Vertex separation can also be useful to separate two particular nodes in a graph. For instance in a transportation network, it may be desired to know which stations are necessarily part of any path from station a to station b. Thus, the goal in this context is to find a separation that puts a in a given pillar and b in the other.

Towards this goal, we present a variation of the VSP called a, b-separation, which can be combined with every version of the VSP discussed earlier. An a, b-separation problem model is a linear program composed of any objective function seen previously (VSP), (VSPr), (VSPw) or (VSPdist) subject to their associated constraints plus the two following ones:

$$x_a = 1 \tag{12}$$

$$y_b = 1 \tag{13}$$

Note that we fixed $a \in A$ and $b \in B$, but besides in the (VSP) version, A and B do not have symmetrical roles. Indeed (7) and (8) created an asymmetry in the problem. This is why setting $a \in B$ and $b \in A$ can lead to a better value than setting $a \in A$ and $b \in B$. In order to avoid having to try both cases, we defined a new set of constraints which sets a and b in A and B without forcing knowing the order.

$$x_a + y_a = 1 \tag{14}$$

$$x_a = y_b \tag{15}$$

$$y_a = x_b \tag{16}$$

(14) implies that a is either in A or B, a can not be in the separator. (15) and (16) imply that b is necessarily in the subset where a is not. Since either x_a or y_a is equal to 1, then b can not be in the separator as well.

3 Application to the Internet Movie Database (IMDb)

In order to evaluate how our approach performs,we applied it to graphs created from the IMDb database. We have access to the list of all the movies inventoried in the database and the list of the movies each actor / director contributed to. We created a graph where the vertices correspond to the movies, which are connected by an edge if they have at least one common contributor.

For our first experiment, we kept the 100 most popular movies. We constructed the graph and only kept the biggest connected component (the graph must be connected).

We considered equivalent weights for each vertices and a trade-off parameter $\lambda = 0.7$. The VSPr model was implemented in R and solved using the free solver lpSolve on a 2.5 GHz Intel Core i7 processor. This VSPr instance was solved in 4ms CPU.

The resulting graph is composed of 74 vertices. Figure 4 presents the graph with its separator. The pillars A and B are in yellow and green respectively while the separator C is colored in red. The cardinalities of A, B and C are respectively 32, 32, 10.

The separator contains famous movies – movies that are in average 1.5 times more voted than the other movies. Since the size of the separator is quite small, we present here the full list of the movies it contains : {The Godfather: Part II (1974); The Dark Knight (2008); One Flew Over the Cuckoo's Nest (1975); The Silence of the Lambs (1991); Raiders of the Lost Ark (1981); The Departed (2006); Django Unchained (2012); The Dark Knight Rises (2012); The Dark Knight Rises (2012); Requiem for a Dream (2000); Inglourious Bastards (2009)}

In order to have a deeper understanding of how the graph has been separated, we plotted the histograms of the countries and years of release of the movies in A and B. Figure 5 shows that the countries of production are fairly distributed between the two pillars. However, Figure 6 shows that the algorithm semantically splits the graph according to the year of production. Indeed, it separated older movies from more recent ones. Note that we had not given any of this information to the algorithm.

For the second experiment, we kept 1000 movies. The corresponding VSPr model (with same parameters than previously, also implemented in R) was solved with Gurobi in 260s on Calcul Québec's servers (8 cores).

The biggest connected component was a graph composed of 273 vertices. Figure 7 presents the graph with its separator. The pillars A and B are in yellow and green while the separator C is colored in red. The cardinalities of A, B and C are respectively 93, 152, 28.

Figures 8 and 9 present the histograms of the years and countries of release of the movies in A and B. Figure 8 shows that pillar A contains older movies while B contains more recent movies. However, one can notice that many movies released in 1987 belong to B. Further investigation is needed to understand this result, and link it to

Fig. 4: Graph with separator for the first experiment

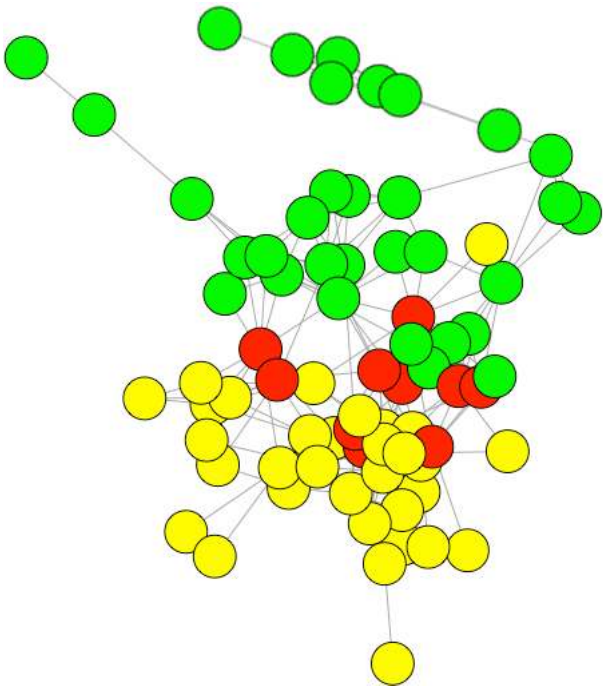
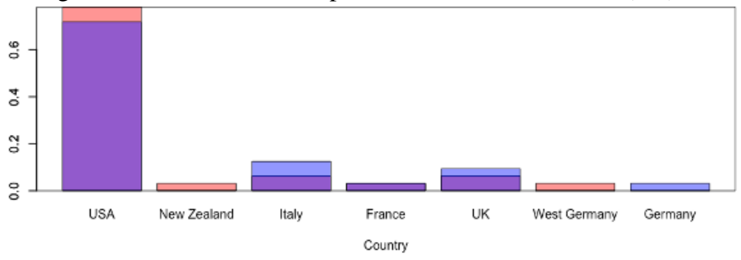


Fig. 5: Histograms of the countries of production for movies in A (red) and B (blue).



the graph structure. Figure9 shows that the algorithm separated the graph according to the countries of production. Besides the United States of America, for all the countries, the movies produced there are almost all in the same pillar.

4 Conclusion and Future Work

In this paper, we proposed several variants of the VSP (VSPr, VSPw, a-b separation) for which we propose integer linear models. We also proposed a new version of the VSP able to take into account distances between vertices (VSPd), and its linearization (VSPdist).

Preliminary experiments on the IMDb database provided promising results: the proposed approach has been able to semantically split graphs without considering the in-

Fig. 6: Histograms of the years of release for movies in A (red) and B (blue).

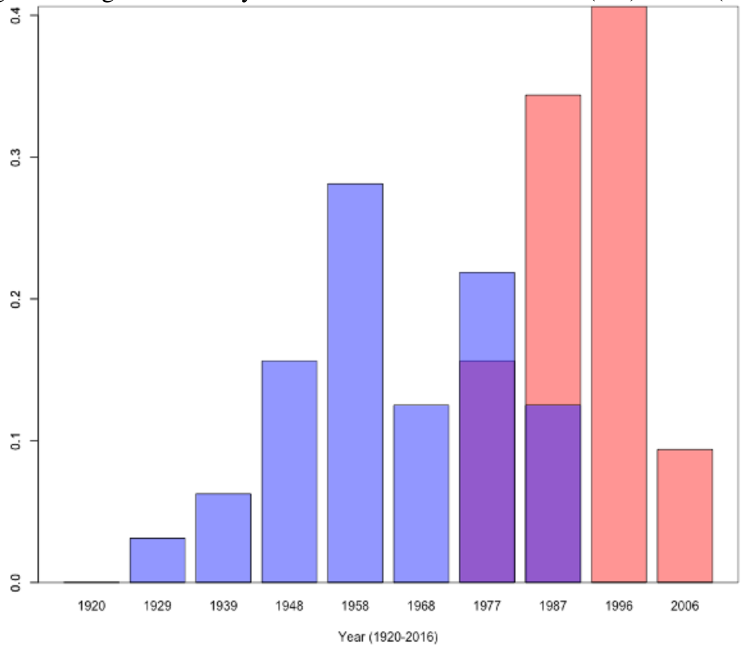
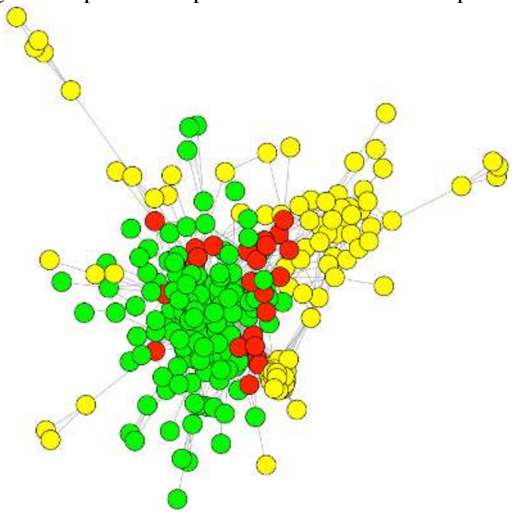


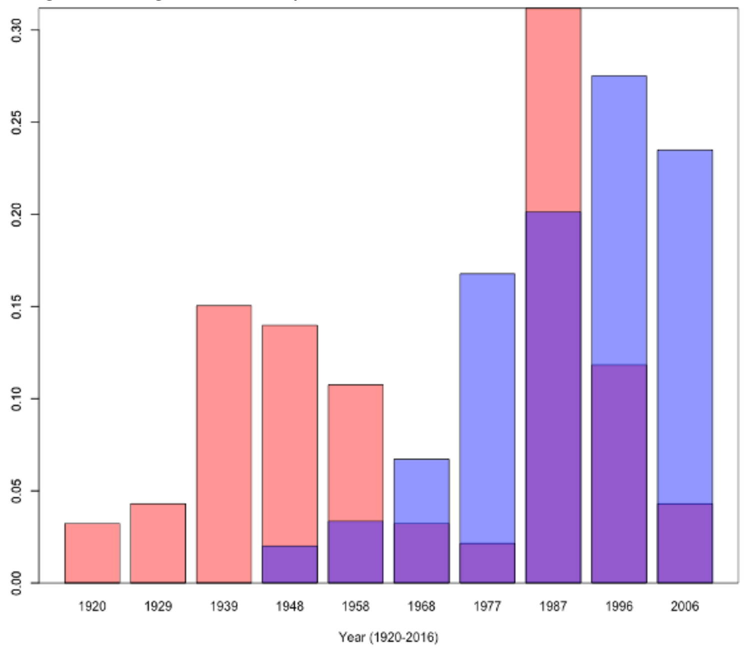
Fig. 7: Graph with separator for the second experiment



trinsic information associated to the vertices. This shows that the VSP could be used for unsupervised clustering purposes.

Our future work is focused on applying the VSP to air transportation. Ongoing computational experiments are conducted on several graphs that represent air transportation network. This field of application is of particular interest to evaluate the variants of our algorithm, especially regarding distance related constraints or a, b-separation.

Fig. 8: Histograms of the years of release and A (red) and B (blue).

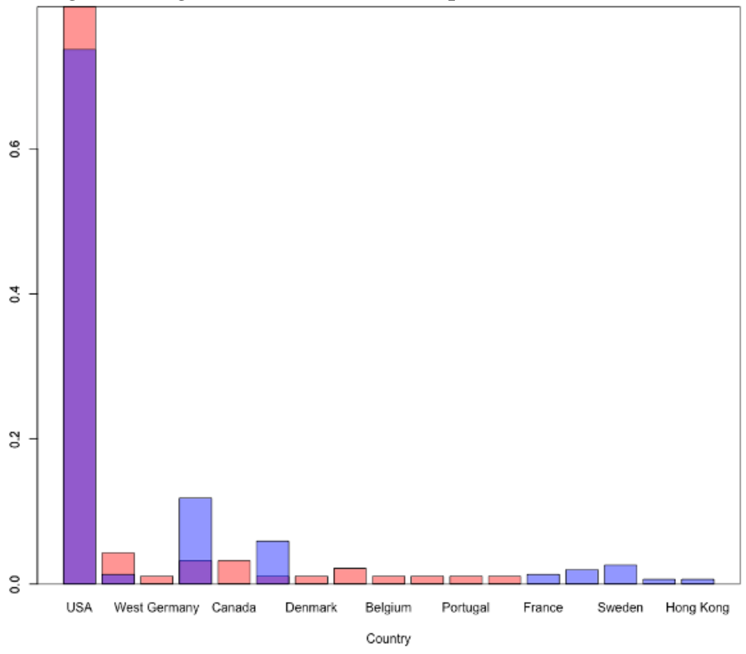


To ensure full reproducibility and comparisons between systems, our source code (in R) will be publicly released as an open source software in the following repository: `https://github.com/BigMiners/vsp`.

References

1. Balas, E., de Souza, C.: The vertex separator problem: a polyhedral investigation. Mathematical Programming 103(3), 583–608 (2005)
2. Benlic, U., Hao, J.K.: Breakout local search for the vertex separator problem. In: IJCAI (2013)
3. Bui, T.N., Jones, C.: Finding good approximate vertex and edge partitions is np-hard. Information Processing Letters 42(3), 153–159 (1992)
4. Davis, T.A.: Direct methods for sparse linear systems, vol. 2. Siam (2006)
5. Didi Biha, M., Meurs, M.J.: An exact algorithm for solving the vertex separator problem. Journal of Global Optimization 49(3), 425–434 (2011)
6. Fu, B., Oprisan, S.A., Xu, L.: Multi-directional width-bounded geometric separator and protein folding. International Journal of Computational Geometry & Applications 18(05), 389–413 (2008)
7. Fukuyama, J.: NP-completeness of the Planar Separator Problems. J. Graph Algorithms Appl. 10(2), 317–328 (2006)
8. Garey, M.R., Johnson, D.S.: Computers and intractability, vol. 29. wh freeman New York (2002)
9. Hager, W.W., Hungerford, J.T.: Continuous Quadratic Programming Formulations of Optimization Problems on Graphs. European Journal of Operational Research 240(2), 328–337 (2015)

Fig. 9: Histograms of the countries of production and A and B.



10. Kanevsky, A.: Finding all minimum size separating vertex sets in a graph. Coordinated Science Laboratory Report no. ACT-93 (UJLU-ENG 88-2233) (1988)
11. Lipton, R.J., Tarjan, R.E.: A Separator Theorem for Planar Graphs. SIAM Journal on Applied Mathematics 36(2), 177–189 (1979)
12. Schaeffer, S.E.: Graph clustering. Computer Science Review 1(1), 27–64 (2007)
13. Schuchard, M., Geddes, J., Thompson, C., Hopper, N.: Routing around decoys. In: Proceedings of the 2012 ACM conference on Computer and communications security. pp. 85–96. ACM (2012)
14. de Souza, C., Balas, E.: The vertex separator problem: algorithms and computations. Mathematical Programming 103(3), 609–631 (2005)
15. Ullman, J.D.: Computational aspects of VLSI. Computer Science Press (1984)