Multiplicity in the Partitioning of Signed Graphs

Nejat Arinik

▶ **To cite this version:**

Nejat Arinik. Multiplicity in the Partitioning of Signed Graphs. Other [cs.OH]. Université d'Avignon, 2021. English. NNT : 2021AVIG0285 . tel-03384624

HAL Id: tel-03384624
https://theses.hal.science/tel-03384624

Submitted on 12 Jan 2023

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.

AVIGNON UNIVERSITÉ

THESIS

presented at the Avignon Université to obtain
the degree of Doctor

Graduate School Nº 536
Agrosciences & Sciences
Sciences, Technologies, Santé
Speciality : *Computer Science*

By

Nejat ARINIK

Multiplicity in the Partitioning of Signed Graphs

Research Unity : EA 4128 LIA – Laboratoire Informatique d'Avignon

Jury members :

Reviewers :	M^{me} Nadia Brauner	Professor, G-SCOP - Université Grenoble Alpes
	M. Mathieu Latapy	Professor, LIP6 - CNRS and Sorbonne Université
Examiners :	M. Zacharie Ales	Associate professor, UMA - CEDRIC - ENSTA Paris
	M^{me} Christine Largeron	Professor, LabHC - Université Jean Monnet
Advisor :	M. Rachid Elazouzi	Professor, LIA - Avignon Université
Co-Advisor :	M. Vincent Labatut	Associate professor, LIA - Avignon Université
Co-Supervisor :	M^{me} Rosa Figueiredo	Associate professor, LIA - Avignon Université

LABORATOIRE INFORMATIQUE D'AVIGNON

ACKNOWLEDGEMENT

First and foremost, I would like to express my special appreciation and gratitude to my supervisors Associate Professor Vincent Labatut, Associate Professor Rosa Figueiredo and Professor Rachid Elazouzi. I would like to thank them for encouraging my research. After all our meetings, I have felt that a new perspective has opened in my mind not only about my research but also about science and universe. Since we met, I have always felt lucky to have such experienced and talented advisors. Your characters with endless patience and discipline have become a good role model for my further academic life. I deeply believe that the end of this thesis only marks a fresh start for future and fruitful collaborations with them.

I warmly thank Nadia Brauner, Professor at the University of Grenoble Alpes, and Mathieu Latapy, Professor at Sorbonne University, for accepting to review my PhD thesis. I thank them for all the time and efforts they have engaged to read the manuscript and write the reviews. I deeply thank as well Professor Christine Largeron and Associate Professor Zacharie Ales to be part of my PhD committee.

I would like to thank all my colleagues from the LIA for the numerous and various discussions and shared moments.

I am particularly grateful to Assistant Professor Özgün Pınarer, for his support, guidance and valuable comments when I was asking him questions about doing a PhD. Thanks to his advice, I contacted Vincent Labatut, who was searching for a PhD student on signed graph partitioning with Rosa Figueiredo.

Many thanks to Assistant Professor Günce Keziban Orman and Associate Professor Özlem Durmaz İncel, which had highly and positively impacted my early research career during my Bachelor thesis in Galatasaray University.

The support of my family motivated me during my studies. I dedicate my thesis to my parents, for their love and for fostering all my academic endeavors.

Finally, I thank my partner Ana Valenzuela. Her love, encouragement and tolerance made possible everything. Thank you.

Titre : Multiplicité dans le partitionnement de graphes signés

Mot clés : Graphe Signé, Correlation Clustering, Équilibre Structural, Graphe Multiplexe Signé, Espace des Solutions Optimales, Mesures d'Évaluation entre les Partitions.

Résumé : Selon la théorie de l'équilibre structural, un graphe signé est *structurellement équilibré* s'il peut être partitionné en sous-groupes mutuellement hostiles (i.e. reliés seulement par des liens négatifs) tout en exhibant une solidarité interne (i.e. contenant uniquement des liens positifs). Mais un réseau réel (i.e. un graphe représentant un système du monde réel) est rarement *parfaitement* équilibré : on trouvera quelques liens positifs entre les groupes et/ou quelques liens négatifs à l'intérieur de certains groupes. L'un des défis du domaine est de quantifier le niveau de déséquilibre d'un tel réseau et d'identifier les liens qui causent ce déséquilibre. Le problème *Correlation Clustering* (CC) se définit précisément par l'obtention d'une partition possédant un déséquilibre minimal.

Le partitionnement de graphes signés constitue une tâche importante du point de vue applicatif, étant donné que trouver une partition équilibrée aide à comprendre le système modélisé par le graphe signé. Cependant, l'approche standard dans la littérature se contente de chercher une seule partition, comme si elle caractérisait suffisamment le système étudié. Or, on peut avoir besoin de plusieurs partitions pour construire une image plus juste du système étudié. Même si cette notion de la multiplicité est extrêmement important du point de vue des utilisateurs finaux, elle a été très peu abordée dans la littérature.

Dans cette thèse, on veut relaxer l'hypothèse de partition unique pour en chercher plusieurs, et ce dans deux situations séparées. La première concerne les graphes multiplexes signés. Un tel graphe est composé de plusieurs graphes séparés, appelés *couches*, et chacun contient les mêmes noeuds, mais possiblement des liens différents. Toutes les approches traditionnelles proposées pour les graphes multiplexes produisent une seule partition à la fin. Pour pallier ce problème, nous proposons une nouvelle approche qui intègre un processus de clustering avant de fusionner les couches individuelles, ce qui permet de regrouper les couches structurellement similaires. En l'appliquant sur les données du Parlement Européen, cela nous a permis non seulement d'extraire plusieurs comportements de vote caractéristiques, ce qui améliore l'interprétation, mais aussi d'expliquer le contexte associé aux comportements, grâce aux documents législatifs concernés.

La deuxième situation est spécifique au problème CC. Quand on résout une instance de ce problème, plusieurs partitions optimales peuvent coexister. La question qui se pose est de savoir ce qu'on perd, si on considère une seule partition optimale, alors qu'il en existe plusieurs. Idéalement, il faut les énumérer toutes avant de faire une analyse concluante. Pour ce faire, on propose une nouvelle méthode d'énumération et un framework basé sur l'analyse de clustering afin de d'abord complètement énumérer l'espace des partitions optimales, puis étudier empiriquement un tel espace. Nos résultats ont révélé une typologie de l'espace de partitions optimales : 1) une seule partition optimale ; 2) quelques partitions constituant une seule classe ; 3) beaucoup de partitions optimales constituant une seule classe de forme allongée ; 4) plusieurs partitions optimales constituant plusieurs classes de partitions.

Dans les deux situations décrites ci-dessus, mesurer la similarité entre deux partitions est un point essentiel. Pourtant, il n'existe pas une définition de la meilleure mesure pour cela, étant donné que toute définition de la similarité est subjective et dépend notamment de l'application considérée. En conséquence, de nombreuses mesures ont été décrites dans la littérature. Cependant, l'abondance de mesures de similarité est une limitation plutôt qu'un avantage, car la sélection de la mesure la plus appropriée pour une application donnée devient un défi pour les utilisateurs finaux. Pour pallier ce problème, nous proposons un nouveau framework. Celui-ci prend en compte plusieurs paramètres liés au partitionnement (e.g. nombre de noeuds, nombre de modules), et évalue statistiquement les mesures dans plusieurs scénarios de partitionnement à travers ces paramètres, via l'analyse de régression. Nos résultats montrent l'importance d'une évaluation générique et paramétrique, car le comportement d'une mesure dans un scénario spécifique peut être complètement différent en fonction des valeurs des paramètres.

Title: Multiplicity in the Partitioning of Signed Graphs

Keywords: Signed Network, Correlation Clustering, Structural Balance, Signed Multiplex Network, Space of Optimal Solutions, External evaluation measures.

Abstract: According to the structural balance theory, a signed graph is considered structurally balanced when it can be partitioned into a number of modules such that positive edges are located inside the modules and negatives ones are in-between them. In practice, real-world networks are rarely perfectly balanced. When it is not the case, one wants to measure the *magnitude* of the imbalance and to identify the set of edges related to the network imbalance. The Correlation Clustering (CC) problem is precisely defined as finding the partition with minimal imbalance.

Signed graph partitioning is an important task, which has many applications, as finding a balanced partition helps understanding the system modeled by the graph. However, the standard approach used in the literature is to find a *single* partition and focus the rest of the analysis on it, as if it was sufficient to fully characterize the studied system. Yet, it may not reflect the meso-structure of the network, and one may need to seek for other partitions to build a better picture. Although this need to look for *multiplicity* is extremely important from the end user's perspective, only a very few works took it into consideration in their analysis, up to now.

In this thesis, we want to relax this traditional single-partition assumption to allow searching for *multiple* partitions in two separate situations. The first one arises in the context of signed multiplex networks. All traditional approaches proposed to partition multiplex networks in general are based on the single-partition assumption. To overcome this limitation, we propose a new partitioning method that integrates a *meta-clustering* process before merging the partitions of individual layers, which allows identifying structurally similar layers. By applying it to a European Parliament dataset, we could obtain multiple partitions, each corresponding to a different characteristic voting pattern of the same considered legislators. The emergence of such patterns was completely hidden when considering only traditional approaches. For instance, we could not only confirm that the French S&D and ALDE MEPs alternatively side with the left- and right-wing groups, but also identified which topics are concerned by these swings.

The second situation is specific to the CC problem. When solving an instance of such problem, several or even many *optimal* partitions may coexist. If multiple optimal partitions coexist, one can then wonder how different/diverse they are. Put differently, we want to know what we loose when considering only one partition, while there might be multiple ones. In order to answer these questions, one should ideally enumerate completely the space of optimal partitions, and perform its analysis. To this end, we propose a new efficient solution space enumeration method and a cluster analysis-based framework in order to first enumerate the space of optimal partitions and then empirically study such space. Based on our empirical study, our main finding is the identification of 4 different situations: 1) unique solution; 2) single class of similar solutions; 3) several classes of similar solutions; 4) multiple solutions without a clear clustering structure.

Lastly, each of these previous situations requires to compute the similarity between partitions. In the context of graph partitioning, this task can be done through a so-called external evaluation measure. However, there exist many such measures, each having different characteristics. This makes it challenging to select the most appropriate for a given situation for the end user. To this end, we propose a new empirical evaluation framework in order to produce results that end users can easily interpret. For a collection of candidate measures, it first consists in describing their behavior by computing them for a generated dataset of parametric partitions, obtained by applying a set of predefined parametric partition transformations. Second, our framework characterizes the measures in terms of how they are affected by these parameters and transformations. Our results show that our framework allows identifying the desirable properties possessed by each measure. For some of them, our results confirm empirical and theoretical findings already published in the literature. For others, the systematic nature of our approach even uncovers properties not mentioned before in the literature.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Context

In a *signed graph*, each edge is associated to a sign, which can be either positive ($+$) or negative ($-$). This type of graph was originally introduced in Psychology, as a means to describe relationships between people belonging to distinct social groups [110]. More generally, they can be used to model any system containing two types of antithetical relationships (like/dislike, for/against, similar/different...). A signed graph is considered *structurally balanced* if it can be partitioned into two [39] or more [56] modules, such that positive edges are located inside the modules, and negatives ones are in-between them. For instance, in the case of a social network whose edges represent like/dislike relationships, this amounts to having mutually hostile social groups with internal friendship.

However, it is very rare for a real-world network to have a perfectly balanced structure: the question is then to quantify how imbalanced it is. Various measures have been defined for this purpose, the simplest consisting in counting the numbers of frustrated edges, i.e. negative ones located inside the modules, and positive ones located between them [39]. Such measures are expressed relatively to a graph partition, so processing the graph balance amounts to identifying the partition corresponding to the lowest imbalance measure. In other words, calculating the graph balance can be formulated as an optimization problem. This problem can be compared to *Community Detection*, which consists in partitioning *unsigned* networks in order to detect groups of vertices more densely connected relatively to the rest of the network [93]. The main difference is of course the presence of signs attached to edges, which represent additional information one has to take into account [68, 69, 65]. Doing so is a non-trivial task, which cannot be conducted by simply performing minor adaptations of community detection methods [46, 164].

Signed graph partitioning is also an important problem in terms of applications: it is used in a number of situations to get a better understanding of the studied real-world system, be it social [63], biological [55], diplomatic [72], business-related [120], sports-related [125], judicial [170], bibliographic [113], political [51], conversational [109], geographic [45], financial [156], etc. It is also used to solve some problems of interest related to the considered systems, e.g. portfolio optimization [156], opinion group detection in online discussions [109], decomposition of biological systems [55], document classification [27], surface detection in 3D images [133], or detection of em-

bedded matrix structures [91].

1.2 Signed graph partitioning related to structural balance

One appropriate way of studying the structural balance of a signed network is by solving the *Correlation Clustering* problem (Section 1.2.2) and its relaxed version (Section 1.2.3). Before presenting them, let us first introduce the notations necessary for defining both problems, as well as the common notations used throughout this thesis (Section 1.2.1).

1.2.1 Notations

Let $G = (V, E, w)$ be an *undirected weighted graph*, where V and E are the sets of vertices and edges, respectively, and $w : E \to \mathbb{R}^+$ is a function associating a *positive weight* to each edge. We note $n = |V|$ the number of vertices, and $w(e)$ the weight of edge $e \in E$. If the graph is unweighted, then the definition is the same except all weights are 1. Now, consider a function $s : E \to \{+, -\}$ that assigns a *sign* to each edge in E. An undirected weighted graph G together with a function s is called an *undirected weighted signed graph* (*signed graph* for short), denoted by $G = (V, E, w, s)$. An edge $e \in E$ is called negative if $s(e) = -$ and positive if $s(e) = +$. We note E^- and E^+ the sets of negative and positive edges in a signed graph, respectively. Let also define the positive graph G^+ of a given signed graph $G = (V, E, w, s)$ as the subgraph (V, E^+, w) (i.e. same vertices, but only the positive edges). Throughout this thesis, we use the terms *graph* and *network* interchangeably.

Let $P = \{M_1, ..., M_\ell\}$ $(1 \leq \ell \leq n)$ be an ℓ-partition of V, i.e. a division of V into ℓ non-overlapping and non-empty subsets M_i $(1 \leq i \leq \ell)$ called *modules*. Given a partition P, an edge (u, v) is called *internal* if it is located inside a module, i.e., u and v belong to the same module. On the contrary, an edge (u, v) is called *external* if it is located between any two modules, i.e., u and v belong to two different modules. Given two modules $M_i, M_j \in P$ and $\sigma \in \{+, -\}$, let us define $E^\sigma(M_i, M_j) = \{(u, v) \in E^\sigma \mid u \in M_i \text{ and } v \in M_j\}$ the edges of sign σ connecting two modules M_i and M_j. We also define $E(M_i, M_j) = E^-(M_i, M_j) \cup E^+(M_i, M_j)$ as the set of edges connecting both modules, without regards for their sign. Similarly, given two modules $M_i, M_j \in P$ and $\sigma \in \{+, -\}$, let us define $\Omega^\sigma(M_i, M_j) = \sum_{(u,v) \in E^\sigma(M_i, M_j)} w(u, v)$ the weighted sum of edges of sign σ connecting these modules. We also define $\Omega(M_i, M_j) = \Omega^+(M_i, M_j) - \Omega^-(M_i, M_j)$ as the signed sum of the edge weights connecting both modules.

1.2.2 Correlation Clustering (CC)

In its original version [27], and consistently with the definition of structural balance given earlier in Section 1.1, the CC problem consists in finding a partition of the set of vertices V which maximizes

both the number of positive edges located *inside* the modules, and that of negative edges located *between* them.

The *Imbalance* $I(P)$ of a partition P is defined as the total weight of positive edges located between modules, and negative edges located inside them, i.e.

$$I(P) = \sum_{1 \leq i \leq \ell} \Omega^-(M_i, M_i) + \sum_{1 \leq i \neq j \leq \ell} \Omega^+(M_i, M_j). \tag{1.1}$$

Problem 1.1 (CC problem)**.** *For a signed graph $G = (V, E, w, s)$, the Correlation Clustering problem consists in finding a partition P of V such that the imbalance $I(P)$ is minimized.*

The partition P is called a *solution* of the CC problem for the given graph G. We note that the CC problem can be independently solved for each subgraph induced by a component of G^+ [8].

To the best of our knowledge, this NP-hard minimization problem [27] appears under this name for the first time in Bansal's paper [27], but it has been addressed before in the literature, e.g. [63]. Moreover, we note that the CC problem is equivalent to the *Multicut* problem [57], where their objective functions differ by a constant (see [141] for the calculation details). Finally, the CC problem is also equivalent to the *Cluster Editing* problem [35] when the input graph is complete and unweighted.

1.2.3 Relaxed Correlation Clustering (RCC)

In [64], the definition of a structurally balanced signed graph was extended in order to include relevant processes (polarization, mediation, differential popularity and subgroup internal hostility) that are counted in Equation 1.1 as violations of the structural balance. According to this new definition, a signed graph is considered *relaxed ℓ'-balanced* if it can be ℓ-partitioned, with $\ell \leq \ell'$, in such a way that: 1) all the edges within a given module have the same sign (not necessarily $+$) ; and 2) all the edges between two given modules have the same sign (not necessarily $-$).

Using this new definition, the structural balance was generalized to a version named *Relaxed Structural Balance* [64], resulting in a new definition for the imbalance of a graph partition. For an ℓ-partition $P = \{M_1, ..., M_\ell\}$, the *Relaxed Imbalance* $RI(P)$ is defined as

$$RI(P) = \sum_{1 \leq i \leq \ell} \min\{\Omega^+(M_i, M_i), \Omega^-(M_i, M_i)\} \tag{1.2}$$

$$+ \sum_{1 \leq i \neq j \leq \ell} \min\{\Omega^+(M_i, M_j), \Omega^-(M_i, M_j)\}. \tag{1.3}$$

This generalized imbalance defines a new criterion to evaluate the partition of a signed graph, and gives rise to the following graph clustering problem.

Problem 1.2 (RCC problem)**.** *Let $G = (V, E, w, s)$ be a signed graph, and ℓ' an integer value satisfying $1 \leq \ell' \leq n$. The Relaxed Correlation Clustering problem consists in finding an ℓ-partition*

P of V, with $\ell \leq \ell'$, such that the relaxed imbalance $RI(P)$ is minimized.

It is worth noticing that, for a given graph, the RCC solution is necessarily equally or more balanced than the CC one. In the worst case, the CC solution holds, and in the best case, the graph can be partitioned more efficiently by taking advantage of the fact some violations of the original structural balance are accepted in its relaxation.

Both CC and RCC problems have been proved to be NP-hard [27, 90]. Exact approaches (e.g. [57, 90, 185]) can be used to solve both problems to optimality. Numerical experiments have shown that the additional parameter ℓ, in the RCC definition, makes the graph clustering problem more difficult to solve numerically [90]. Both problems can be efficiently solved by heuristic methods (e.g. [148, 185]. We review and compare later exact and heuristic methods proposed for the CC problem in Chapter 2.

1.3 Challenges

Signed graph partitioning is an important task, which has many applications. In this regard, obtaining a partition for this task helps understand the studied system. However, the standard approach in the literature is to find a *single* partition and focus the rest of the analysis on it, as if finding it was sufficient to understand the considered system at hand. Yet, a single partition of a given set of vertices may not reflect the meso-structure of the network. It is possible that one needs to seek for multiple partitions to get a better understanding. Although this need to look for *multiplicity* is extremely important from the end user's perspective, only a very few works took it into consideration in their analysis, up to now. This notion of multiplicity is the main challenge that we want to tackle in this thesis. We want to relax this traditional single-partition assumption to allow searching for *multiple* partitions in two separate situations.

The first one is related to the optimal solutions of a given graph partitioning problem. Such multiplicity raises several questions in this context. First, if *several* optimal partitions (i.e. optimal solutions) coexist, one can wonder which network properties lead to this situation, and how many of these partitions are equally relevant to the application problem at hand. Put differently, we want to know what we lose when we consider only one optimal partition, while there might be multiple ones. Second, what methodology would be appropriate to obtain these partitions? Third, how different do the obtained partitions need to be? Application-wise, very similar partitions could be given the same interpretation [1], whereas substantially different ones might correspond to dramatically different ways of seeing the studied system. Fourth and finally, in such different partitions, what distinguishes them from each other? Identifying these characteristic differences could provide some valuable information to understand the studied system. More generally, the answers to all these questions could drive the task of searching multiple partitions.

1. Although a difference of one vertex could be important in a given application, if the vertex is central for instance.

The second situation concerns multiplex signed networks. A multiplex network consists of a collection of single ones, called *layers*, each containing the same vertices but possibly different edges. Each layer provides a different, possibly conflicting, view of the same system under different conditions. In the literature, all approaches proposed to partition multiplex networks in general (i.e. not necessarily signed) are based on the single-partition assumption [132]. Yet, there could be several relevant partitions, each one fitting only certain layers, and ignoring them would result in an unsatisfying resolution. The notion of multiplicity in this context differs from the previous one in that the number of partitions can be an input parameter of a resolution method, rather than the consequence of it. Finding the most appropriate number could be challenging for the end-user, and depends on the application at hand.

This paradigm of multiplicity naturally gives rise to a secondary challenge, which is related to the similarity between obtained partitions. In the presence of a large number of partitions, one needs to calculate the similarity between them. In the context of graph partitioning, this task can be done through a so-called external evaluation measure, i.e. a measure originally designed to compare two partitions. However, many such measures have been proposed in the literature, each having different characteristics. This makes it challenging to select the most appropriate for a given situation for the end user. In general, this issue is overlooked in the literature. Researchers tend to follow tradition and use the standard measures of their field, although they often became standard only because previous researchers started consistently using them. Ideally, one should identify desirable properties that a measure should satisfy relative to the application context. Directly using a standard measure amounts to completely discard the application context.

1.4 Contributions

This thesis contributes to the research field in several different ways, by tackling the challenges described in Section 1.3 and related to multiplicity, in two separate situations. The first one arises in the context of signed multiplex networks and the second one is specific to the CC problem. In both situations, we also address the issue of selecting the most appropriate external measure. The most prominent three contributions are:

Multiplicity for the CC problem: It has been pointed out, but not studied, in the literature that an instance of the CC problem can possess several *optimal* partitions [56, 63, 62, 57, 38]. In order to answer the questions asked in Section 1.3 and also to select some of these partitions (e.g. the representative ones), one should ideally enumerate completely the space of optimal partitions of the CC problem, and then proceeding with confidently the subsequent analysis. To this end, we make the following two contributions.

1. We design a new efficient method for the enumeration of all optimal partitions of an instance.

Such method needs to employ exact approaches to guarantee the completeness of the optimal partition space. This is not a trivial task, because due to the NP-hard nature of the problem, even methods finding a *single* optimal partition do not scale well. In order to accelerate this process, our proposed enumeration method includes a local search mechanism and several pruning strategies.

2. We propose a cluster analysis-based framework, which allows us to study how the nature of multiple partitions is affected by the network characteristics. Put differently, we empirically study the CC problem itself.

Multiplicity for signed multiplex networks: To overcome the single-partition assumption in multiplex signed networks, we propose a new partitioning method able of outputting as many partitions as required. Each layer is partitioned separately, then the resulting partitions are gathered to form clusters of structurally similar solutions. A consensual merging process tailored for signed networks is then performed for each cluster, in order to obtain a set of representative partitions, each associated to certain layers of the multiplex graph.

Selection of evaluation measure: We also address the issue of selecting the most appropriate measure for a given situation by proposing a new empirical evaluation framework. For a collection of candidate measures, it first consists in describing their behavior by computing them for a generated dataset of partitions, obtained by applying a set of predefined parametric partition transformations. Second, our framework characterizes the measures in terms of how they are affected by these parameters and transformations. This allows both describing and comparing the measures.

Beside these prominent contributions, there are two secondary contributions of this thesis:

Comparison of resolution methods proposed for the CC problem: Solving the CC problem is a compulsory task that we will consistently address throughout this thesis. For this reason, we need an efficient exact method and high quality heuristic(s). To this aim we review and evaluate both exact and heuristic methods that have been proposed for the CC problem.

Validation of the proposed methods: We evaluate each proposed framework and method either on a synthetic or real-world dataset, and show its usefulness by extensively interpreting the obtained results. Regarding the synthetic signed networks, we propose two random network generation methods:

1. The first one is based on a model inspired by the Erdős-Rényi random graph model and allows generating signed graphs with controlled balance.

2. The second method guarantees that the optimal solution is known by construction, thanks to the definition of stability range [176]. This allows to skip the application of an exact partitioning method, which can be very time-consuming when graph order n increases.

On the side, this thesis contributes also for :

Data availability: In order to promote transparency and reproducibility in research we made publicly available all datasets and results used throughout this thesis: 10.6084/m9.figshare.14551113. Our source codes are also publicly available, and indicated in each chapter.

1.5 Personal Bibliography

All the contributions listed in Section 1.4 allowed us to make the following publications and submissions:

— **International Journals**

— N. Arınık et al., « Multiple partitioning of multiplex signed networks: Application to European parliament votes », *in*: *Social Networks* 60 (2020), pp. 83–102, DOI: https://doi.org/10.1016/j.socnet.2019.02.001 (**Chapter 4**)

— N. Arınık et al., « Multiplicity and Diversity: Analyzing the Optimal Solution Space of the Correlation Clustering Problem on Complete Signed Graphs », *in*: *Journal of Complex Networks* (2020), DOI: 10.1093/comnet/cnaa025 (**Chapters 6**)

— N. Arınık et al., « Characterizing and comparing external measures for the assessment of cluster analysis and community detection », *in*: *IEEE Access* (2021), pp. 1–22, DOI: 10.1109/access.2021.3054621 (**Chapter 3**)

— **International Conferences / Workshops**

— N. Arınık et al., « Signed Graph Analysis for the Interpretation of Voting Behavior », *in*: *International Conference on Knowledge Technologies and Data-driven Business - International Workshop on Social Network Analysis and Digital Humanities*, 2017 (**Chapter 4**)

— N. Arınık et al., « Study of the European Parliament votes through the multiple partitioning of signed multiplex networks », *in*: *29th European Conference On Operational Research (EURO)*, 2018 (**Chapter 4**)

— **National Conferences / Workshops *with* proceedings**

— N. Arınık et al., « Analysis of Roll-Calls in the European Parliament by Multiple Partitioning of Multiplex Signed Networks », *in*: *9ème Conférence Modèles & Analyse des Réseaux : Approches Mathématiques & Informatiques (MARAMI)*, 2018 (**Chapter 4**)

  - N. Arınık et al., « Multiple Optimal Solutions but Single Search: A Study of the Correlation Clustering Problem », *in*: *20ème congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Société Française de Recherche Opérationnelle et d'Aide à la Décision*, 2019 (**Chapter 6**)
  - N. Arınık et al., « Multiple Partitioning of Multiplex Signed Networks », *in*: *21ème Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Société Française de Recherche Opérationnelle et d'Aide à la Décision*, 2020 (**Chapter 4**)
  - N. Arınık et al., « Characterizing measures for the assessment of cluster analysis and community detection », *in*: *11ème Conférence Modèles & Analyse de Réseaux : approches mathématiques et informatiques (MARAMI)*, 2020 (**Chapter 3**)

- **National Conferences / Workshops *without* proceedings**
  - N. Arınık et al., « Exploiting Antagonistic Relations in Signed Graphs under the Structural Balance Hypothesis », *in*: *Programme Gaspard Monge Pour l'Optimisation (PGMO) Days*, 2018 (**Chapter 4**)
- **Submitted Articles for International Journals**
  - N. Arınık et al., « Efficient Enumeration of Correlation Clustering Optimal Solution Space (submitted) », *in*: *Journal of Global Optimization* (2021) (**Chapter 5**)

1.6 Organization

The thesis is organized into six additional chapters, which we summarize as follows.

- Chapter 2: We review and evaluate both exact and heuristic methods that have been proposed for the CC problem.
- Chapter 3: We introduce our own framework designed to study and compare evaluation measures and their properties. We put it into practice on a selection of widespread measures and discuss its results.
- Chapter 4: We describe the approach we propose for the analysis of multiplex signed networks. We put the proposed method into practice through an application to a European Parliament dataset.
- Chapter 5: We present our method for enumerating the space of optimal solutions of the CC problem. We put the proposed method into practice on a collection of complete and incomplete unweighted synthetic signed networks and discuss its results.

— Chapter 6: We explain the approach we propose for the characterization of the optimal solutions of the CC problem. We describe and discuss our results based on the same synthetic graphs generated in Chapter 5.

— Chapter 7: We summarize and criticize our work, propose some leads to solve the existing limitations, and identify our major perspectives.

Chapter 2

Correlation clustering resolution methods

2.1 Introduction

In the literature, a large number of works deal with the CC problem to get a better understanding of the studied system at hand, or to solve a specific problem of interest. When solving an instance of the CC problem, most of these works rely on *heuristic* approaches, e.g. [67, 66, 148, 52, 26], especially when dealing with large networks, since the problem is NP-hard [27], as described in Section 1.2. But a non-negligible number of studies are also concerned with *optimality*, e.g. [10, 38, 90, 19, 12, 130]. As we see in the next chapters, solving the CC problem is a compulsory task that we consistently address throughout this thesis. For this reason, we need an efficient exact method and high quality heuristic(s). Nevertheless, to the best of our knowledge, the literature lacks

a comprehensive review of the resolution methods that have been proposed for the CC problem. Therefore, our objective in this chapter is to review and compare both exact and heuristic methods, and identify the efficient ones through computational experiments. In this review and comparison, we exclude from our discussion the approximation methods, as their primary interest is to get provable approximation bounds. Interested readers can refer to [44, 220] and references therein.

The literature provides three main methods to solve an instance of the CC problem exactly: 1) *Branch-and-Bound (B&B)*, 2) *Weighted partial MaxSAT-based* and 3) *Parameterized Enumeration*. B&B can be performed using two different algorithmic approaches: Integer Linear Programming (ILP) vs. *ad hoc* B&B programming. Their main difference is that the former can tackle any problem translated in the language of mathematical programming through any optimization solver, whereas in the latter the construction of the B&B tree relies on problem-specific idiosyncrasies. An *ad hoc* B&B programming method for a clustering problem systematically constructs partial solutions by assigning the vertices to one of the existing modules. This forms a search tree, whose branches correspond to assignments of vertices to modules. As shown in Figueiredo and Moura [90] for the CC problem, the *ad hoc* B&B method proposed in [38] is outperformed by an ILP approach, since it cannot deal well with finding an optimal solution of graphs containing more than 20 vertices.

The weighted partial MaxSAT-based method transforms an instance of the CC problem into a Maximum Satisfiability formulation and solves it through an appropriate MaxSAT solver. The corresponding MaxSAT formulation consists of two sets of clauses, soft and hard ones, of Boolean variables and a function that associates a non-negative cost with each of the soft clauses. Any truth assignment that satisfies the hard clauses gives a valid solution. Berg and Järvisalo [32] propose three different MaxSAT formulations for the CC problem and obtain their best results based on a compact bit-wise formulation by using a hybrid MaxSAT solver *MaxHS*[1]. However, as shown by Miyauchi et al. [168], their method is outperformed by an ILP-based solution method.

Parameterized Enumeration is slightly different from the previous ones in that it is an *ad hoc* principle which requires to transform a part of the considered problem as a new input parameter. This in turn guarantees to bound the overall running time in function of the problem inputs, assuming that the new parameter is small in practice for an efficient resolution. In the literature, Böcker et al. [35] propose an FPT (Fixed-Parameter Tractable) algorithm for the *Cluster Editing* problem, which is equivalent to the CC problem when the input graph is complete and unweighted. Concretely, this FPT algorithm makes the number of frustrated edges in structural balance a parameter to solve the problem. However, a drawback of this approach is that the amount of imbalance $I(P)$, the input parameter, can be very large. Indeed, Böcker et al. [35] show in their experiments that an ILP-based method outperforms the proposed FPT algorithm. Due to the efficiency of mathematical programming-based approaches (e.g. dual tightening), in this chapter we therefore focus only on ILP-based exact methods proposed for the CC problem.

1. http://www.maxhs.org

Contributions. This chapter makes the following contributions:

1. **Random network generator.** We propose two open source unweighted signed graph generators. The first one is based on a model inspired by the Erdős-Rényi random graph model which generates signed graphs with controlled balance. The second one accomplishes this task by perturbing existing signed graphs without affecting their initial optimal partitions thanks to the definition of stability range [176].
2. **Evaluation.** We evaluate a large number of exact and heuristic approaches proposed for the CC problem on synthetic unweighted complete and incomplete signed networks of moderate size. We also include several large real-world networks in the evaluation of the heuristic methods.

The rest of this chapter is organized as follows. In Section 2.2, we review ILP-based exact approaches proposed for the CC problem. Next, in Section 2.3, we pass to heuristic methods and describe the most relevant ones for this work. We put both exact and heuristic methods into practice on a selection of complete and incomplete signed networks in Section 2.4 and discuss their results in Section 2.5. Finally, we review our main findings in Section 2.6, and identify some perspectives for our work.

2.2 ILP-based exact methods

In this section, we describe two classical exact approaches proposed for the CC problem. These are Branch-and-Cut (B&C) and Benders Decomposition (BD) methods. In the literature, they are solved through a Cutting Plane (CP) approach. In the following, we give the preliminary context for these approaches, followed by their descriptions.

The CC problem can be independently solved for each subgraph induced by a component of G^+ [8]. Therefore, throughout this chapter, we assume that there is a single connected component in G^+ of a given graph G.

Let us introduce the necessary definitions and notations used in the rest of this chapter. We call a cycle of G *chordless* if no two vertices of the cycle are connected by an edge that does not itself belong to the cycle. Moreover, we call a cycle of G *conflicted* if it contains precisely one negative edge. In the following, $\mathscr{C}(G)$ denotes the set of all chordless cycles of G. Also, $\mathscr{C}^-(G)$ denotes the set of all conflicted cycles of G, i.e., $\mathscr{C}^-(G) = \{\text{cycle } C \text{ of } G \ : \ |C \cap E^-| = 1\}$.

2.2.1 ILP formulations

The CC problem can be modeled by using any graph clustering formulation. In the following, we present the most widespread ILP formulations, where the decision variables can be defined on edges or vertex pairs.

2.2.1.1 Decision variables on edges

For any G, the CC problem can be modeled, as in [47], by means of an ILP proposed for the *graph partitioning problem*. For each edge $(u, v) \in E : u < v$, a binary variable is defined to indicate whether it is located inside a module, i.e.,

$$x_{uv} = \begin{cases} 1, & \text{if edge } (u, v) \text{ is inside a module,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Then, the ILP formulation of the CC problem for signed networks is written as follows:

$$\min \quad \sum_{(u,v)\in E^+} w_{uv}(1 - x_{uv}) + \sum_{(u,v)\in E^-} w_{uv}x_{uv} \tag{2.2}$$

$$\text{s.t.} \quad \sum_{(u,v)\in C\setminus(i,j)} x_{uv} - x_{ij} \leq |C| - 2, \forall C \in \mathscr{C}(G), \forall (i, j) \in C \tag{2.3}$$

$$x_{uv} \in \{0, 1\}, \quad \forall x_{uv} \in E. \tag{2.4}$$

The objective function (2.2) allows looking for a feasible solution minimizing the imbalance defined in Equation (1.1). A feasible solution (i.e. decision variables corresponding to a partition) must satisfy the cycle inequalities [47] defined in Equation (2.3). They ensure that if a cycle C contains an edge located between modules (i.e., $x_{uv} = 0$), then another edge of C must be also located between modules. Finally, constraints (2.4) are the domain constraints for the variables in the formulation.

It is important to stress that there are an exponential number of cycle inequalities, and this number substantially increases with increasing values of $|E|$. In [57, 142] (and in [231] for its relaxation version), the authors show that Equation (2.3) can be replaced by Equation (2.5). This indicates that a substantial amount of cycles in $\mathscr{C}(G)$ are actually redundant, and not needed to be included in the model.

$$\sum_{(u,v)\in C\setminus(i,j)} x_{uv} - x_{ij} \leq |C| - 2, \forall C \in \mathscr{C}^-(G), (i, j) = C \cap E^-. \tag{2.5}$$

We denote $F_e(G)$ as the formulation defined in Equations (2.2) to (2.4), and $F_e^\star(G)$ as the formulation defined in Equations (2.2), (2.4) and (2.5). Moreover, we define $P_e(G)$ as the corresponding polytope of these formulations. In practice, these formulations are particularly appropriate for sparse signed networks [10, 130].

2.2.1.2 Decision variables on vertex pairs

The CC problem can be also modeled, as in [57, 90], by means of an ILP proposed to solve the *clique partitioning* problem [103, 47]. This formulation defines a decision variable for each pair

of vertices, even when the original graph G is not complete. In such case, we can reduce the cycle inequalities to a polynomial number of triangle inequalities.

Let us introduce the second ILP formulation for the CC problem. For each pair of vertices $u, v \in V : u < v$, a binary variable is defined to describe pairs of vertices that are in the same module, i.e.,

$$x_{uv} = \begin{cases} 1, & \text{if } u \text{ and } v \text{ are in a same module,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

Then, the ILP formulation of the CC problem for undirected signed graphs is written as follows:

$$\min \quad \sum_{(u,v)\in E^+} w_{uv}(1 - x_{uv}) + \sum_{(u,v)\in E^-} w_{uv}x_{uv} \tag{2.7}$$

$$\text{s.t.} \quad x_{uv} + x_{vr} - x_{ur} \leq 1, \quad \forall u < v < r \in V \tag{2.8}$$

$$x_{uv} - x_{vr} + x_{ur} \leq 1, \quad \forall u < v < r \in V \tag{2.9}$$

$$-x_{uv} + x_{vr} + x_{ur} \leq 1, \quad \forall u < v < r \in V \tag{2.10}$$

$$x_{uv} \in \{0, 1\}, \quad \forall u, v \in V. \tag{2.11}$$

The objective function (2.7) is the same as in the previous model, which minimizes the imbalance defined in Equation (1.1). A feasible solution (i.e. decision variables corresponding to a partition) must satisfy the triangle inequalities defined in Equations (2.8)-(2.10). For instance, the triangle inequality in Equation (2.8) says that if vertices u and v are in the same module, as well as vertices v and r, then vertices u and r must also be in this module. Finally, constraints (2.11) are the domain constraints for the variables in the formulation.

In this formulation, the number of triangle inequalities is cubic, which is still too large for practical efficiency. Miyauchi et al. [168] reveal that some of these triangle constraints are redundant. Therefore, it is sufficient to use only those in Equations (2.12)-(2.14). The triangle constraints other than Equations (2.12)-(2.14) are then called *redundant*.

$$x_{uv} + x_{vr} - x_{ur} \leq 1, \quad \forall u < v < r \in V, w_{uv} > 0 \vee w_{vr} > 0 \tag{2.12}$$

$$x_{uv} - x_{vr} + x_{ur} \leq 1, \quad \forall u < v < r \in V, w_{uv} > 0 \vee w_{ur} > 0 \tag{2.13}$$

$$-x_{uv} + x_{vr} + x_{ur} \leq 1, \quad \forall u < v < r \in V, w_{vr} > 0 \vee w_{ur} > 0 \tag{2.14}$$

We denote $F_v(G)$ as the formulation defined in Equations (2.7)-(2.10) and (2.11), and $F_v^\star(G)$ as the one defined in Equations (2.7), (2.11), (2.12)-(2.14). Moreover, we define $P_v(G)$ as the corresponding polytope of these formulations. It is important to stress that $F_v^\star(G)$ may fail to obtain a valid optimal solution on incomplete signed networks in the absence of the redundant triangle constraints. This is because some variables associated to the edges with zero weight can have inconsistent val-

ues so as to minimize the objective function, since they do not contribute to the calculation of the imbalance. To correct this, Miyauchi et al. [168] propose the following simple post-processing. Let $\boldsymbol{x}^*$ be an optimal solution to $F_v^\star(G)$. Also, let $E_1 = \{(u,v) \in E \mid x^*_{uv} = 1, w_{uv} > 0\}$ the decision variables located in the same modules in G^+. We obtain a set of weakly connected components $\{V_1, V_2, ..., V_k\}$ of (V, E_1) by a depth-first search. The final optimal solution corresponds to a 0-1 vector reflecting the result of the weakly connected components.

2.2.2 Facet-defining inequalities

Both ILP models are sufficient to find an optimal solution through an optimization solver, but it can be very time-consuming. One way to deal with this issue is to strengthen the formulations through facet-defining inequalities. We are interested in such inequalities because they are the *strongest* ones to improve a formulation, as we can see in the following definitions.

A set $P \subseteq \mathbb{R}^n$ is called a polyhedron if P is the intersection of finitely many half-spaces, that is:

$$P = \{\boldsymbol{x} \in \mathbb{R}^n | A\boldsymbol{x} \leq b\}, \tag{2.15}$$

for some $m \times n$ matrix A and some $b \in \mathbb{R}^m$. If P is bounded, then P can also be written as the convex hull of many points in $\mathbb{R}^n$, and it is a polytope. The dimension of P, denoted by $dim(P)$, is one less than the maximum number of affinely independent points in P. If, for some $a \in \mathbb{R}^n$, $a \neq 0$, $a_0 \in \mathbb{R}$, P is contained in the half-space $\{\boldsymbol{x} \in \mathbb{R}^n | a^T\boldsymbol{x} \leq a_0\}$, then the inequality $a^T\boldsymbol{x} \leq a_0$ is said to be *valid* for P. The set $F = P \cap \{\boldsymbol{x} \in \mathbb{R}^n | a^T\boldsymbol{x} = a_0\}$ is called a *face* of P. Notice that F itself is a polyhedron (or a polytope). We have $dim(F) \leq dim(P) - 1$, if F is a face of P. If $dim(F) = dim(P) - 1$, then F is called a facet, and the inequality $a^T\boldsymbol{x} \leq a_0$ is said to be *facet-defining*. See, e.g., Nemhauser and Wolsey [174] for further details.

2.2.2.1 Facet-defining inequalities for $P_e(G)$

The first class of known facet-defining inequalities is cycle inequalities (2.3) defined for chordless cycles [47].

Odd-Wheel inequalities are another class of known facet-defining inequalities for the $P_e(G)$ polytope. A wheel is a cycle, in which all vertices are connected to a further vertex. Let a q-wheel be a connected subgraph $G_q = (V_q \cup \{j\}, E_q)$, where V_q is the set of q vertices constituting the cycle and $E_q = C \cup \widetilde{C}$ is composed of cycle edges $C = \{(v_i, v_{i+1}) | i = 1, .., q\}$ and non-cycle edges $\widetilde{C} = \{(j, v_i) | i = 1, .., q\}$. All indices are modulo q with $q(mod\ q) = 0$. For $q = 3$, an example of 3-wheel is shown in Figure 2.1a.

For every q-wheel, a valid partitioning x satisfies the inequality

$$\sum_{(u,v)\in C} x_{uv} - \sum_{(u,v)\in \widetilde{C}} x_{uv} \leq \lfloor \frac{q}{2} \rfloor. \tag{2.16}$$

Deza et al. [58] prove for $P_e(G)$ that the odd-wheel inequalities (2.16) are facet-defining for every odd $q \geq 3$.

For completeness, we note that other inequalities are known to further tighten the LP relaxation of $P_e(G)$, which are also facet-defining. These are the *clique-web* [58] and *bicycle* [47] inequalities. Nevertheless, in practice, taken together the inequalities (2.3) and (2.16) describe a tight polynomial-time solvable relaxation to $P_e(G)$ [176, 126].

2.2.2.2 Facet-defining inequalities for $P_v(G)$

We note that $P_e(G)$ is a projection of $P_v(G)$, therefore, any facet for $P_e(G)$ is also valid for $P_v(G)$. In the following, we describe those defined only for $P_v(G)$.

Grötschel and Wakabayashi [104] show that the triangle inequalities (2.12)-(2.14) define facets of the $P_v(G)$ polytope, as well as the lower bound constraints $x_{uv} \geq 0$ $(u, v \in V)$ in the LP-relaxation. Furthermore, Grötschel and Wakabayashi [104] also prove that the following two classes of inequalities define facets of the P_v polytope:

— Let $S, T \subseteq V$ be two nonempty disjoint subsets of V. Then, the inequality

$$\sum_{u\in S}\sum_{v\in T} x_{uv} - \sum_{\substack{(u,v)\in S \\ u\neq v}} x_{uv} - \sum_{\substack{(u,v)\in T \\ u\neq v}} x_{uv} \leq min\{|S|, |T|\} \tag{2.17}$$

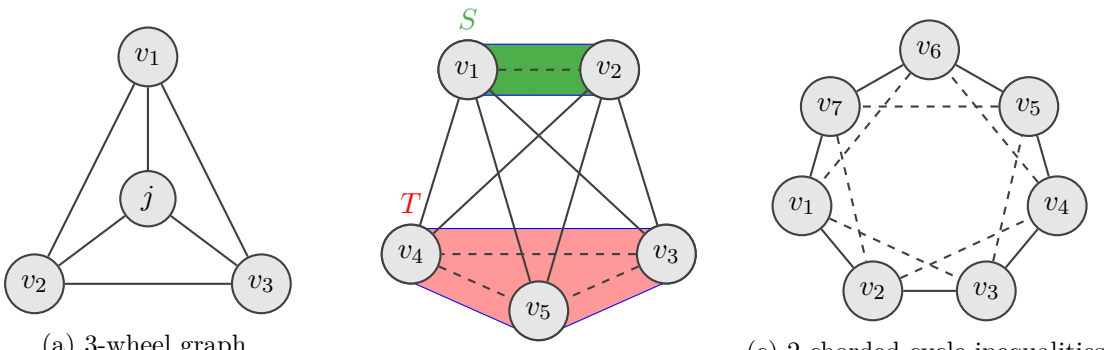
is valid. It defines a facet if and only if $|S| \neq |T|$. It is called a 2-partition inequality (see Figure 2.1b).

— Let $C \subseteq E$ be a cycle of length at least 5, and $\overline{C} = \{v_i v_{i+2} | i = 1, .., |C| - 2\} \cup \{v_1 v_{|C|-1}, v_2 v_{|C|}\}$ be a 2-chorded cycle of C. Then, the inequality

$$\sum_{(u,v)\in C} x_{uv} - \sum_{(u,v)\in \overline{C}} x_{uv} \leq \lfloor \frac{|C|}{2} \rfloor \tag{2.18}$$

is valid. It defines a facet if and only if $|C|$ is odd, and it is called a 2-chorded cycle inequality (see Figure 2.1c).

For completeness, we note that other inequalities can further tighten the LP relaxation of $P_v(G)$. These are the 2-*chorded even wheel* [104] inequality and those defined in [178]. Nevertheless, in practice, together the inequalities (2.17) and (2.18) describe a tight polynomial-time solvable relaxation to $P_v(G)$.



(a) 3-wheel graph. (b) 2-partition inequalities. (c) 2-chorded cycle inequalities.

Figure 2.1 – Illustrations of some facet-defining inequalities for $P_v(G)$ and $P_e(G)$.

2.2.3 Resolution methods

2.2.3.1 Cutting Plane (CP) approach

Both B&C and BD approaches rely on a Cutting Plane (CP) algorithm [228]. When the number of constraints is too large to deal with explicitly, it provides an efficient method to solve relaxations of ILPs. The CP approach consists of four steps.

Step 1: Define a set of linear valid inequalities $Form$ which approximates the polytope $P_v(G)$ or $P_e(G)$.

Step 2: Solve the current formulation $Form$.

Step 3: Solve a subproblem, which is used to improve the approximation $Form$ by including additional inequalities. In case of B&C, the subproblem is a separation problem, which strives to find valid inequalities that are violated by the current primal solution. In case of BD, the subproblem is an optimization problem in the dual domain, where a feasible solution to it can yield new Benders inequalities.

Step 4: Use the inequalities found in step 3 and update the current approximation $Form$.

We repeat the steps 2-4 until one of the three termination criteria are met: 1) the solution obtained in step 2 is integral, 2) this solution is still fractional but there is no violated valid inequalities found, and 3) this solution is still fractional but a predefined time limit is reached.

2.2.3.2 Branch-and-Cut

In combinatorial optimization, Branch-and-Cut [174] is a well-known efficient method, which improves the traditional B&B procedure by adding violated valid inequalities during the optimization process. This can be done in many ways. Nevertheless, empirical works in the literature [126, 4] show that there exist two successful applications of Branch-and-Cut for $F_v(G)/F_v^\star(G)$ and $F_e(G)$: 1)

adding violated valid inequalities only at the root of the B&B tree through the CP method and then proceeding to the branching phase, 2) adding violated valid inequalities only for integer solutions during the branching phase.

The first strategy is based on the idea that the CP method improves the current LP relaxation faster than the branching process at the beginning, and switches to the branching phase when the improvement gathered by the CP method starts to become negligibly small. Ales et al. [4] show for a related problem that the first strategy performs best for $F_v(G)$ (this is also valid for $F_v^\star(G)$). The second strategy takes advantage of integer solutions to produce more efficient optimization process. The authors in [126] obtain the best results based on the second strategy, when they deal with $F_e(G)$ on sparse signed graphs. Moreover, they point out that using an efficient calculation of an initial lower bound at start (see [126] for more details) seems to be beneficial for accelerating the optimization process.

In our computational experiments, we stick to the two successful applications of Branch-and-Cut described above. In the following, we denote the B&C procedures based on $F_e(G)$ and $F_v^\star(G)$ by B&C($F_e(G)$) and B&C($F_v^\star(G)$), respectively.

2.2.3.3 Benders decomposition (BD)

The main objective of a BD method [174] is to turn a hard-to-solve formulation with a subproblem structure into a simpler one by temporarily fixing a subset of variables and introducing an exponential number of constraints. Adding all the constraints would be too costly. Therefore, the CP method is applied for an efficient solving process, where new cuts, called *Benders cuts*, are found by solving subproblems of the original problem.

Originally, the CC problem does not fit the definition of the BD method, in the sense that it does not possess a subproblem structure. Nevertheless, the decomposition scheme proposed by Keuper et al. [130] makes this possible for the formulation $F_e^\star(G)$. In their decomposition scheme, the authors take advantage of the definition of the conflicted cycles and decompose the edges into subproblems based on a minimal vertex cover S for E^-. They re-write the CC problem as

$$\min \quad \sum_{(u,v)\in E^-} w_{uv}x_{uv} + \sum_{(u,v)\in E^+} w_{uv}(1-x_{uv}) + \sum_{s\in S} Q(w,s,\boldsymbol{x}) \tag{2.19}$$

$$\text{s.t.} \quad x_{uv} \in \{0,1\}, \quad \forall u,v \in V. \tag{2.20}$$

The term $Q(w,s,\boldsymbol{x})$ provides the cost to *correct* $\boldsymbol{x}$ in order to satisfy all conflicted cycles in-

equalites. It is defined as follows:

$$\text{Min} \quad \sum_{(u,v)\in E_s^-} w_{uv}x_{uv}^s + \sum_{(u,v)\in E^+} w_{uv}(1-x_{uv}^s) \tag{2.21}$$

$$\text{s.t.} \quad \sum_{(s,t)\in C\setminus(u,v)} (x_{st}+x_{st}^s) - x_{uv} - x_{uv}^s \leq 2|C|-3, \forall C \in \mathscr{C}_s^-(G), (u,v)\in C\cap E^- \tag{2.22}$$

$$x_{uv}^s \in \{0,1\}, \quad \forall u,v \in s. \tag{2.23}$$

The set $\mathscr{C}_s^-(G) \subseteq \mathscr{C}^-(G)$ represent a subset of cycles $\mathscr{C}^-(G)$ associated with the negative edges E_s^- in subproblem s. This means that in each subproblem a certain number of a slightly modified version of cycle inequalities are enforced. With this new form of cycle inequalities in (2.22), $Q(w,s,\boldsymbol{x})$ provides the cost to alter $\boldsymbol{x}$ by increasing (resp. decreasing) x_{uv} for (u,v) in E^+ (res. E^-). The authors reformulate $Q(w,s,\boldsymbol{x})$ as a cut-problem and dualize it in order to obtain the Benders cut.

The master problem is as follows:

$$\min \quad \sum_{(u,v)\in E^-} w_{uv}x_{uv} + \sum_{(u,v)\in E^+} w_{uv}(1-x_{uv}) \tag{2.24}$$

$$\text{s.t.} \quad \sum_{(u,v)\in E} x_{uv}\lambda_{uv}^z \leq 0, \forall z \in \mathcal{Z} \tag{2.25}$$

$$x_{uv} \in \{0,1\}, \quad \forall (u,v) \in E. \tag{2.26}$$

where $\mathcal{Z}$ is the set of dual feasible solutions found for all subproblems $s \in S$, and λ_{uv}^z the dual variables associated.

Using the CP method, the BD algorithm repeatedly solves the master problem, which includes only a subset of constraints, to obtain a trial value for the x variables, i.e. $\bar{x}$. It then solves Benders subproblem with $\bar{x}$. If any Benders cut are violated by the current solution, they are inserted into the current formulation and the process repeats. During the CP, the authors also includes another class of cuts, called *Magnanti-Wong Benders Rows* [158], to accelerate the optimization process.

In the following, we denote the corresponding BD procedure based on the formulation $F_e^\star(G)$ by BD($F_e^\star(G)$).

2.3 Heuristic methods

Despite advances in the development of exact methods, the partitioning of large signed networks can be tackled only by heuristic approaches. Moreover, these problem-specific approaches are often appropriate as a primal heuristic in the design of an exact method to provide an upper bound.

In the following, we describe a selection of heuristic methods proposed in the literature for the CC problem, which we divide into three categories: LP-based rounding (Section 2.3.1), and Trajectory-

based meta-heuristic (Section 2.3.2) and population-based meta-heuristic (Section 2.3.3) methods. All these heuristics use somehow some neighborhood search operators for discovering new partitions. For this reason, given a partition P, we define an r-neighborhood structure, denoted by $N_r(P)$, as the family of all partitions obtained by moving r vertices in P from one module into another.

2.3.1 LP-based rounding methods

A traditional rounding procedure proposed in the literature first obtains a primal LP relaxation for the $P_v(G)$ polytope, which outputs a fractional solution. Then, this procedure rounds the values of the variables in the fractional solution to either 0 or 1 based on a rounding scheme (e.g., putting vertices u and v in the same module, if x_{uv} is close to 1). For the CC problem, the final solution must satisfy the triangle inequalities, so that this results in a valid partitioning solution. In the literature, several approximation methods (e.g., [42]) rely on this procedure. However, because the number of triangle inequalities largely increases with n, they do not scale well. For this reason, we consider in this section two recent rounding methods based on a dual LP relaxation, which scales much better. We are also interested in these methods because their performances with respect to the existing heuristics are not known.

These methods possess a different rounding scheme from a traditional one. They somehow estimate new signed edge weights from the solution of a dual LP relaxation, then apply a local-search method onto the same input graph, whose edges are re-weighted by the previous step. The idea behind this weight adjustment is to guide primal heuristics toward better feasible solutions.

2.3.1.1 Message Passing (MP+GAEC+KLj)

In [207], the authors decompose the original problem into easily solvable combinatorial subproblems defined on graph substructures (edges, triangles and odd-wheels) in order to obtain a LP relaxation.

The proposed algorithm is performed in two steps. In the first step, the relaxation problem is solved in a fashion similar to the cutting plane approach: it alternates for a fixed number of iterations between the message passing and separation procedures. The message passing procedure [208] solves a dual of the relaxation of the proposed decomposition and tightens the previous relaxation by adding new cycle and odd-wheel inequalities (described in Section 2.2.2.1), which are violated by the current solution. Finally, in the second step, based on the solution obtained for the relaxation, a feasible integer solution is found through GAEC and KLj heuristics (see Section 2.3.2.2), which are two local search algorithms widely used in the community of computer vision.

2.3.1.2 Iterative Cycle Packing (ICP+GAEC+KLj)

Lange et al. [142] design a rounding procedure different than a decomposition approach as in MP+GAEC+KLj. They propose an iterative heuristic, called *ICP*, to solve the maximum set packing problem with respect to conflicted cycles, which is obtained by taking the dual of the $F_e^\star(G)$ formulation (see [142] for the definition of this problem). The resulting solution indicates which conflicted cycles are selected, and knowing this allows to compute a lower bound for the $F_e^\star(G)$ formulation. In addition to that, ICP outputs another set of edge weights, called *residual weights*, related to the selection of the conflicted cycles. They can be interpreted as an indication of how likely pair of vertices are placed in the same module in an optimal solution. The authors propose to adjust the original edge weights w in the $F_e^\star(G)$ formulation by taking into account these residual weights $c \in \mathbb{R}^{|E|}$. This is obtained through the convex combination $\lambda|c_e| + (1-\lambda)w_e$ of w and c with a suitable choice of $\lambda \in (0,1)$ for each existing edge e (see Lange [141] for the calculation of the residual weights c). Finally, as in MP+GAEC+KLj, the authors propose to use the GAEC and KLj heuristics (see Section 2.3.2.2) onto the given input signed graph whose edge weights are adjusted by ICP.

2.3.2 Trajectory-based (meta-)heuristic methods

In this section, we rely on trajectory-based (meta-)heuristics. The term *trajectory* refers to the search process characterized by a trajectory, i.e. the evolution of a dynamical system considered through discrete time. These methods start from an initial partition, and always deal with a single partition at a time throughout the trajectory. In the following, we describe the trajectory methods designed for the CC problem.

2.3.2.1 Basic Local Search: Iterative Improvement

Iterative Improvement designates a local search process, where the initial partition is iteratively improved by exploring the neighborhood of the visited partitions. This neighborhood exploration can be a very time-consuming process. For this reason, only a small number of vertex changes (relative to the current partition) is usually investigated. Two different improvement criteria terminate such investigation: *first* vs. *best* improvement. In the former, the local search updates the current solution whenever it finds an improved solution, whereas the latter exhaustively explores the r-neighborhood $N_r(P)$ and replace the current solution by one of the partitions with the best objective value. Some examples of Iterative Improvement are the Kernighan-Lin (KL) [129] and BOEM [67] methods.

Iterative Improvement constitutes the most basic form of local search, because it does not involve any technique to escape from local minima during the search. This makes its performance limited. Therefore, it is usually embedded in (meta-)heuristics. For this reason, we do not consider it as a separate heuristic in this chapter.

2.3.2.2 Hierarchical Iterative Improvement

Hierarchical Iterative Improvement is a natural extension of the iterative improvement heuristic, such that the neighborhood exploration of the current solution is not only based on vertices, but also on modules. Such module-based exploration can be *agglomerative* or *divisive*. In the former, vertices initially constitute their own modules and these modules are iteratively merged. In the latter, there is initially a single module containing all vertices and it is recursively split.

In the literature, there are several heuristics of this kind [147, 181, 26]. In the following, we consider one agglomerative and one hybrid iterative improvement methods: GAEC+KLj and CGC, respectively. The latter combines both agglomerative and divisive strategies. In [147], both methods give better results compared to standard ones.

Greedy Additive Edge Contraction and Kernighan-Lin Algorithm with Joins (GAEC+KLj) The algorithm proposed by Levinkov et al. [147] is composed of two independent methods: GAEC and KLj. First, GAEC starts by forming an initial partition: vertices are initially single-module vertices. In every iteration, a pair of modules sharing at least one positive edge are merged, if this operation yields the maximal gain in terms of objective value. Then, KLj takes the result of GAEC as an input and starts to modify it. KLj is a slight extension of the Kernighan-Lin algorithm [129]. In addition to the vertex-exchange move used in [129], KLj includes two additional types of moves: merging two modules and moving some of vertices from an existing module into an empty one. These three moves are used in each iteration in order to update greedily the underlying partition so as to approach to being pairwise optimal.

GAEC+CGC+QPBO The idea behind Cut, Glue & Cut (CGC) [29] is to improve an existing partition locally, by focusing on only a subset of vertices at a time. The proposed method consists of two phases: the *cut* and *glue&cut* phases. In the cut phase, the underlying graph is recursively split by solving max-cut (also known as the weighted 2-coloring problem) problems. Since solving the max-cut problem is NP-hard, the authors sovle it approximately by using the QPBO-I method [194]. In the glue&cut phase, any two neighboring modules are first merged and then a new partition is sought between them by solving again a max-cut problem. These two phases are repeated and the process stops when the value of the objective function cannot be improved anymore. The method keeps track of the previously solved subproblems, therefore it avoids resolving the same subproblems. According to [147], this methods gives better results when it takes the result of GAEC as an input.

2.3.2.3 Simulated Annealing (SA)

In physics, the simulated annealing (SA) technique simulates the evolution of a physical system towards its thermodynamic equilibrium at a given temperature. Some pioneer works [131, 40] establish the connection between this physical system and the search for global minima for a discrete

optimization problem. These works rely on the results from statistical mechanics (Metropolis algorithm) for the simulation of the underlying system, which employs an explicit strategy to escape from local minima. SA can be considered as one of the first algorithms that includes such local minima escape strategy.

In the literature, SA has been successfully used to solve the CC problem [173, 216, 211]. In our computational experiments, we use the algorithm proposed by Néda et al. [173]. It starts by generating a random initial solution P and initializing the so-called temperature parameter T. In each iteration, many simulation steps are performed. In each simulation step, a vertex is randomly chosen and is reassigned to a randomly chosen module, which results in a new solution P'. If the value of imbalance $I(P')$ is better than $I(P)$, then P' is accepted to replace P with a probability of 1. Otherwise, P' is accepted to replace P, with a probability, defined as a function of T and the difference $I(P') - I(P)$. This probability is computed following the Boltzmann distribution $exp(-\frac{I(P')-I(P)}{T})$. Introducing parameter T in this formula allows to control the probability of accepting a worse solution. At the beginning, this probability is high, which allows the exploration of the search space. Then, it gradually decreases, converging to a simple iterative improvement algorithm.

2.3.2.4 Tabu Search (TS)

Tabu Search is another heuristic, which also employs a strategy to escape from local minima, but through a so-called *tabu list*. This list keeps track of a limited number of previously visited solutions and forbids any move towards them. Thus, using the tabu list restrains the neighborhood of the current solution in each iteration. On the one hand, this restriction scheme gives the ability of preventing from cycling (from local optima through non-improving moves). On the other hand, the search process tend to be too local, i.e. exploring only a small portion of the search space most of the time, which can be considered an intensification process.

To take advantage of the intensification nature of the tabu search Brusco and Doreian [37] propose an algorithm, in which the tabu search is preceded by a multi-start iterative first improvement heuristic. Since their proposed algorithm seeks a partition with a fixed number of modules, we adapt it for any number of modules in our computational experiments. The algorithm starts by generating an initial good-quality solution P through many runs of the iterative first improvement heuristic based on a single vertex move and by initializing the tabu list L. In each iteration, the move of a single vertex v that provides the largest improvement (or smallest worsening) of the objective function is obtained and this results in a new partition P'. If P' is the best solution obtained so far in terms of imbalance, then P is replaced by P' and the tabu list L is emptied. Otherwise, vertex v is placed into L, and it is kept in for a fixed number τ of iterations and cannot be moved to any module. Moreover, the remaining number of iterations for the other vertices in L is updated by reducing this number by one. The algorithm is run until a prespecified time limit is reached.

2.3.2.5 Variable Neighborhood Search (VNS)

Unlike the iterative improvement where a single neighborhood is used, Variable Neighborhood Search (VNS) [169] alternates between different neighborhoods within the local search phase. It is able to perform moves to distant neighborhoods, which enables to tackle the trajectory difficulties in the search space. This technique is based on the idea that local optima can be different regarding different neighborhood structures. In its original version [169], VNS considers a set of neighborhoods $N_1(P)$ to $N_r(P)$ and dynamically varies the value of r in $N_r(P)$ in order to improve the current solution through increasingly distant neighborhoods. Namely, if an enhanced solution is found with the current value of r, then r resets to its initial value and the best solution is updated. Otherwise, the next neighborhood is traversed by increasing r.

A variant of VNS is introduced by Brusco and Doreian [37] for the CC problem. Like TS, since their proposed algorithm seeks a partition with a fixed number of modules, in our computational experiments we adapt it for any number of modules. In this variant, the size of the neighborhood is controlled by four parameters: y_{pert}, y_{min}, y_{max} and y_{step}. Parameter y_{pert} is the probability that any given vertex will be moved from its current module to a different module. For instance, if $y_{pert} = 0.10$, then we expect 10% of the vertices to be moved to other modules. Parameters y_{min} and y_{max} are the minimum and maximum value that y_{pert} can take, respectively, and y_{step} represents the step size for moving from y_{min} to y_{max}. In [37], VNS is preceded by a multi-start iterative first improvement heuristic. The algorithm starts by generating an initial good-quality solution P through many runs of the iterative first improvement heuristic based on a single vertex move and by initializing the parameters of VNS. In each iteration, the algorithm first perturbs the best solution found so far with respect to the current value of y_{pert}, then a local minimum solution P' is found by performing the iterative first improvement heuristic. If P' is the best solution obtained so far in terms of imbalance, y_{pert} resets to y_{min}. Otherwise, y_{pert} is incremented by y_{step} for examining larger neighborhoods. The overall process is repeated until a prespecified time limit is reached.

2.3.2.6 Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP [87, 88] is a meta-heuristic that combines randomized constructive heuristic and local search. It is composed of two phases: solution construction and solution improvement. The first phase, the construction of an initial solution P is done step-by-step by starting with an empty partition and adding one new vertex at a time through a randomized greedy function. The second phase is a local search process, which strives to improve P through a basic local search algorithm such as iterative improvement, or a more involved technique such as VNS.

In our computational experiments, we consider a multi-start GRASP heuristic proposed by Levorato et al. [148], which is a slightly modified version of the algorithm proposed by the same authors in [66]. The algorithm is composed of several iterations. The multi-start nature enables GRASP to

start from a different solution in each iteration. In each iteration, the classical GRASP procedure, as described above, is performed. In the first phase, the construction of an initial solution P is shaped based on how we choose the next vertex to add into the partition. This is done by picking it uniformly at random from a candidate list, where the vertices are ranked by their fitness function. The random choice is guided by the parameter α, which allows to pick only one of the first (hence, best) α vertices from the candidate list. In the extreme case, the best vertex would be added when $\alpha = 1$, whereas the choice would be completely random when $\alpha = n$. Therefore, α is a critical parameter which affects how the regions of the search space are explored. In the second phase, the original VNS heuristic, as in [169], is applied to improve P. In the end, the best solution is determined over all iterations.

2.3.2.7 Iterated Local Search (ILS)

A related alternative of the GRASP heuristic for escaping from a local minimum is Iterated Local Search [152]. Instead of a multiple solution construction phase, ILS relies on perturbations applied onto the current solution, followed by a local search. ILS starts by finding a first local optimum $\widetilde{S}$ from an initial solution S through a local search procedure. Then, in each iteration, it tries to improve $\widetilde{S}$ in three steps: 1) $\widetilde{S}$ is perturbed to obtain S' as a diversification process, 2) a local search is performed on S' to obtain another local optimum $\widetilde{S}'$, 3) $\widetilde{S}$ is replaced by $\widetilde{S}'$ if the *acceptance criterion* is satisfied.

In our computational experiments, we consider the multi-start ILS heuristic proposed by Levorato et al. [148]. Since the authors embed into their ILS framework a classical single GRASP iteration and a VNS procedure together, it can be seen as an improved version of the GRASP and VNS heuristics. Their algorithm is composed of several iterations. Like GRASP, the multi-start nature enables ILS to start from a different solution in each iteration. The algorithm finds a first local optimum $\widetilde{S}$ in each iteration by applying the classical single GRASP iteration, which is described in the previous section. Then, another local optimum $\widetilde{S}'$ is found by applying a VNS algorithm similar to one used in [37] and described in Section 2.3.2.5. The main differences are that the perturbation intensity is controlled by the number of random vertex movements, rather than a proportion value, and VNS is terminated when y_{pert} attains y_{max}, instead of a time limit. In the end, if $\widetilde{S}'$ is better than $\widetilde{S}$ in terms of imbalance, $\widetilde{S}$ is replaced by $\widetilde{S}'$.

2.3.3 Population-based (meta-)heuristic methods

Population-based approaches deal in every iteration of the algorithm with a set (i.e. a population) of solutions rather than with a single one. Therefore, these methods differ from those presented in Section 2.3.2 on the way the search space is explored. Although several population-based methods exist in the literature for other clustering problems, such as Genetic/Memetic algorithms [100, 206, 155], Ant [203] and Bee [127] Colony Optimization, to the best of our knowledge, only Memetic

algorithms are proposed for the CC problem.

2.3.3.1 Memetic algorithm: MLMSB

A memetic algorithm is an extension of the traditional Genetic algorithm (GA), in that it is strengthen with a local search process, which is supposed to decrease the possibility of premature convergence. In the context of GA, the individuals (also sometimes called chromosomes) correspond to the solutions for the problem at hand. Since those individuals evolve simultaneously, they constitute a *population*. In the beginning, an initial population is created randomly or high-quality solutions from another heuristics can be provided. It evolves by being exposed to several genetic operations throughout several iterations until some termination criterion is met. Those genetic operations are designed to improve the current population by escaping from a local optima. The pipeline is constituted mainly of three such operations: 1) the so-called *tournament selection*, i.e. selection of best solutions in terms of solely the objective function, or in combination with another criterion; 2) *crossover* operator executed for each pair of solutions selected in the tournament selection, i.e. switching some genes from one parent to another; 3) *mutation* operator which randomly perturbs current solutions to escape from a local optima.

In our computational experiments, we consider the memetic algorithm (MA) proposed by Ma et al. [155]. This method outperforms the others in the literature of MA through its *multi-level learning based local search*. This search consists of three processes, which are applied as a pipeline. The first one, *vertex learning*, aims at performing the iterative best improvement based on a single vertex move. The second one, *module learning*, consists in applying the agglomerative iterative best improvement. Finally, the last one, *partition learning*, perturbs the solution generated by module learning, by creating an intersection partition with respect to the best solution in the population, and then applies again the first and second steps. This multi-level local search is performed onto the current population, once genetic operations are applied.

2.4 Dataset

This section is dedicated to the description of the datasets used in the experiments presented in this chapter. As mentioned in the introduction, in this chapter we focus on *unweighted* signed networks. Application-wise, these networks fit certain modeling situations and methodological choices. Indeed, in some works binary values better represent the studied relations (e.g. alliance/conflict between countries in international relationships [65]), or the authors prefer to use such values for practical reasons (e.g. limited or unreliable information [70]). We conduct our experiments in this chapter on three datasets of random signed networks.

Dataset 2.1: These networks are generated through our random signed network generator,

which is publicly available online [2]. For *complete* unweighted signed networks, this model relies on only three parameters: n (number of vertices), ℓ_0 (initial number of modules) and q_m (proportion of misplaced edges, i.e. edges meant to be frustrated by construction). Moreover, we make the assumption that the proportion of misplaced edges is the same inside and between the modules. For complete unweighted signed networks, we generate 3 replications for parameter values $\ell_0 = 3$, $n = 40$ and $q_m = \{0.3, 0.4, 0.5\}$. In these networks, the value of q_{neg} with the considered parameters is approximately 0.7. When it comes to incomplete unweighted signed networks, we introduce two more parameters, which are the density d of the graph and the proportion q_{neg} of the negative edges. The last parameter allows to control the ratio of positive to negative edges. In these incomplete networks we generate 5 replications for parameter values $d = \{0.25, 0.50\}$, $n = \{40, 50\}$, $q_m = \{0.3, 0.5\}$ and $q_{neg} = \{0.3, 0.5, 0.7\}$. In total, we produce 9 and 120 instances for complete and incomplete networks, respectively, which makes a total of 129 instances. We use this dataset in Section 2.5.1.

Dataset 2.2: This dataset uses the same random signed network generator as in *Dataset 2.1*, but with different parameter values. We use this dataset in the evaluation of heuristics (Section 2.5.2) and we concentrate on only the maximal value of graph order n that exact methods can handle within reasonable time limit. For complete unweighted signed networks, we generate 20 replications for parameter values $\ell_0 = 3$, $n = 50$ and $q_m = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. In these networks, the value of q_{neg} with the considered parameters is approximately 0.7. For incomplete unweighted signed networks with $d = \{0.25, 0.50\}$ and $q_{neg} = \{0.3, 0.5, 0.7\}$, we generate 20 replications with the same parameter values as in the complete networks. In total, we produce 120 and 720 instances for complete and incomplete networks, respectively, which makes a total of 840 instances.

Dataset 2.3: For each network in this dataset, the optimal solution is known by construction. This allows to consider relatively large graph orders n without being limited by the long running time of an exact partitioning method. For given n, d and ℓ_0, we first create a perfectly structurally balanced (i.e. internally positive and externally negative) signed network with a built-in module structure through the same random signed network generator used in *Dataset 2.1*. Clearly, the underlying module structure constitutes the optimal partition. Then, in order to take into account different positive to negative ratio values for internal and external edges, we generate several signed networks by increasingly perturbing the initial signed network without affecting its underlying optimal partition, thanks to its definition of stability range [176]. We perform this generation with parameter values $n = 150$, $d \in \{0.25, 0.50, 1.00\}$ and $\ell_0 = \{6, 8, 10\}$ through our random signed network generator, which is publicly available online [3]. In each generated network, the associated optimal solution corresponds to the planted partition defined for the corresponding network without perturbation. In total, we produce 13 and 49 instances for complete and incomplete networks, respectively, which makes

2. https://github.com/CompNet/SignedBenchmark
3. https://github.com/arinik9/SignedStabilityBenchmark

a total of 62 instances. We use this dataset in Section 2.5.2.

Dataset 2.4: In addition to artificial graphs, we also consider 6 sparse unweighted large real-world networks of different sizes from various sources: Slashdot [145], Wikipedia Election [145] and 4 Biological datasets [115]. The Slashdot social network contains friend/foe edges between the users of the Slashdot website in February 2009. The Wikipedia Election social network is based on all adminship elections before January 2008, and the edges between Wikipedia users indicate if one approves/disapproves the other. The biological networks include two gene regulatory networks: Yeast and E. coli, related to two organisms: a eukaryote (the yeast *Saccharomyces cerevisiae*), and a bacterium (*Escherichia coli*), respectively, and two other networks related to the molecular interaction map of a white blood cell (macrophage) and the epidermal growth factor receptor (EGFR) pathway (see [115] for more details). We use an undirected version of these social and biological signed networks, which are made publicly available in [146] and [115], respectively [4]. The considered networks range in order from 329 to 82,144 vertices and from 779 to 498,532 edges. We use this dataset in the evaluation of heuristics (Section 2.5.2).

2.5 Experiments

We now assess the performances of the methods. We first investigate the efficiency of the exact methods (Section 2.5.1), then pass to the evaluation of the heuristics (Section 2.5.2). Our source codes are publicly available [5].

2.5.1 Experiments for exact methods

In this section, we evaluate the performances of the exact methods B&C($F_v^\star(G)$), B&C($F_e(G)$) and BD($F_e^\star(G)$) presented in Section 2.2. The implementation of B&C($F_v^\star(G)$) is based on the code obtained by the authors of [4] and is in Java. We implemented B&C($F_e(G)$), as described in Section 2.2.3.2, in Java. For BD($F_e^\star(G)$), we kindly obtained the implementation from the authors of [130]. While it is implemented in MATLAB, all computation-heavy parts are delegated to C++ functions via MEX wrappers, such that its comparison with both B&C methods is fair. All methods use CPLEX for solving ILPs and LPs. The experiments ran with a time limit of 1 hour up to 8 threads on the same machine, an Intel i7-2100 CPU, operating at 3.10 GHz and equipped with 16 GB of RAM. We present a selection of the most relevant results in Tables 2.1 and 2.2.

We first describe our results generically here, before interpreting them. In these tables the columns are divided into three parts. The first part describes the network characteristics, whereas the second and third ones correspond to the two methods to be compared. In each of these last

4. Most of them can also be retrieved from http://www.doi.org/10.1093/comnet/cny015

5. https://github.com/arinik9/HeuristicsCC, https://github.com/arinik9/ExCC, https://github.com/arinik9/BenchmarkCC.

two parts there are six columns, each corresponds to a different pair of values of q_m and q_{neg}. For ease of comparison, some corresponding columns of both methods are shaded. For each parameter set, five random networks are generated and they constitute the rows of the tables. Rows are also constituted of several parts, and each corresponds to a different pair of values of n and d. For each network, we report the execution time in seconds, or the best percentage optimality gap if the optimization process could not be terminated before the time limit.

We start by comparing B&C($F_e(G)$) and BD($F_e^\star(G)$), which both use the formulation based on edge variables. The results are shown in Table 2.1. First, we observe that both methods are severely affected when n increases. Indeed, they both handle well the instances with $n = 40$, but often require much more time for those with $n = 50$. This is because increasing n can exponentially increase the number of cycle constraints that need to be generated to guarantee optimality. Second, increasing q_m affects the running times of both methods, especially for $n = 50$ and $q_{neg} = 0.5$. This can be explained as introducing more and more misplaced edges tend to increase the number of conflicted cycles. Nonetheless, the instances with small and large values of q_{neg}, i.e. $q_{neg} = \{0.3, 0.7\}$, are usually easier to solve for both methods. Finally, B&C($F_e(G)$) outperforms BD($F_e^\star(G)$) in all instances. This is probably because the considered sparse networks with $d = 0.25$ is probably not as sparse as the authors of BD($F_e^\star(G)$) assume in their work, for which this method is originally tested. To conclude this part, B&C seems to be a more efficient choice compared to BD on the formulation $F_e(G)$ and we stick to B&C($F_e(G)$) in the following when we use $F_e(G)$.

		B&C($F_e(G)$)						BD($F_e^\star(G)$)					
		$q_m = 0.3$			$q_m = 0.5$			$q_m = 0.3$			$q_m = 0.5$		
	Graph	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7
$n = 40$ $d = 0.25$	G_1	1	1	1	1	2	1	55	41	9	86	12%	9
	G_2	1	1	1	1	1	1	105	48	9	61	41	9
	G_3	1	1	1	1	1	1	119	44	6	400	60	15
$n = 50$ $d = 0.25$	G_1	1	155	1	4	300	1	45	12%	35	595	16%	1372
	G_2	1	2	1	1	300	1	328	3%	2%	479	14%	295
	G_3	1	3	1	2	300	1	81	7%	4%	470	18%	49

Table 2.1 – Evaluation of the two exact methods BD($F_e^\star(G)$) and B&C($F_e(G)$) based on the random networks from *Dataset 2.1* for $d = 0.25$. Both methods use the formulation based on edge variables. For each parameter set, five random networks are generated and they constitute the rows of the tables. For each network, we report the execution time in seconds, or the best percentage optimality gap (%) if the optimization process could not be terminated before the time limit. When reporting the execution times, we fix to 1 second those being below this value.

We now compare two Branch&Cut methods B&C($F_e(G)$) and B&C($F_v^\star(G)$), which are based on different ILP formulations. As one can expect, modelling complete networks through $F_v^\star(G)$ is

more appropriate than $F_e(G)$, whereas the latter is supposed to better handle sparse networks. We confirm this with our experiments based on $d = \{0.25, 1.0\}$. Nonetheless, the networks with $d = 0.5$ are more challenging for both methods. For this reason, our discussion in the following concentrates on only these cases. The results are shown in Table 2.2. First, we see that increasing n affects less B&C($F_v^\star(G)$) than B&C($F_e(G)$). This is because the number of triangle constraints in $F_v^\star(G)$ polynomially increases with n. This can explain why B&C($F_e(G)$) is efficient for $n = 40$, whereas B&C($F_v^\star(G)$) better handles the instances with $n = 50$. Second, small and larges values of q_{neg} seem to advantage one formulation to another, independently of the values of n and q_m. Indeed, B&C($F_v^\star(G)$) better performs for $q_{neg} = 0.3$, whereas it is outperformed by B&C($F_e(G)$) for $q_{neg} = 0.7$. Finally, the harder instances, which are "unsolved" by both methods within the time limit of 1 hour, are better dealt with by B&C($F_v^\star(G)$). It seems to be more beneficial to reduce the optimality gap through the CP approach, rather than the branching process, in these instances. This aspect is better exploited by B&C($F_v^\star(G)$). To conclude this part, there is not a single method which always gives the best results, and both Branch&Cut methods have desirable performances depending on the network characteristics. Sparse networks are better handled by B&C($F_e(G)$), whereas B&C($F_v^\star(G)$) better performs on complete networks. Moreover, for a medium graph density, B&C($F_v^\star(G)$)) is less sensitive to increase in n than B&C($F_e(G)$)).

		$B\&C(F_e(G))$						$B\&C(F_v^\star(G))$					
		$q_m = 0.3$			$q_m = 0.5$			$q_m = 0.3$			$q_m = 0.5$		
	Graph	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7
$n = 40$ $d = 0.5$	G_1	38	**12**	**2**	229	1,564	**1**	**1**	254	180	**2**	**768**	253
	G_2	1	**1**	**2**	199	**256**	**2**	1	12	181	**9**	541	205
	G_3	1	**31**	**1**	36	1,136	**1**	1	293	216	**1**	**884**	35
	G_4	**1**	**2**	**1**	8	**46**	**10**	2	16	185	**1**	335	191
	G_5	1	**331**	**1**	16	**91**	**8**	1	343	152	**1**	606	301
$n = 50$ $d = 0.5$	G_1	**2**	3,177	2.6%	**1**	8.7%	**1,104**	3	**1,072**	**2,286**	5	**4.7%**	1,451
	G_2	64	5.5%	**304**	**1**	8.8%	**2,260**	**3**	**1.4%**	1,352	4	**3.8%**	3,551
	G_3	121	2.4%	1.9%	7	10.8%	567	**4**	**1,163**	**1,059**	**3**	**6.5%**	**249**
	G_4	**1**	**4**	**75**	17	7.5%	**1,077**	2	93	649	**3**	**3.1%**	1,175
	G_5	121	2.6%	**30**	121	10.3%	**170**	**2**	**2,121**	542	**3**	**6.9%**	605

Table 2.2 – Evaluation of the two exact methods B&C($F_e(G)$) and B&C($F_v^\star(G)$) based on the random networks from *Dataset 2.1* for $d = 0.5$. Each Branch&Cut method uses a different formulation. For each parameter set, five random networks are generated and they constitute the rows of the tables. For each network, we report the execution time in seconds, or the best percentage optimality gap (%) if the optimization process could not be terminated before the time limit. When reporting the execution times, we fix to 1 second those being below this value.

2.5.2 Experiments for heuristic methods

In this section, we evaluate the performances of the 10 heuristic methods described in Section 2.3 based on *Datasets 2.2, 2.3 and 2.4*. Most heuristics are either publicly available or made available upon demand. We had to implement SA, TS and VNS by following the instructions in the associated papers. For MLMSB, GRASP and ILS, we kindly obtained the implementations by the authors of [155] and [148], respectively. For the remaining heuristics, we used the publicly available codes of the corresponding authors [6]. Although some methods can be run in parallel, not all methods have this option. For a fair comparison, all methods were executed without multi-threading, on the same machine. Certain of these methods are stochastic (SA, TS, VNS, GRASP, ILS, MLMSB), whereas others are purely deterministic (GAEC+KLj, ICP+GAEC+KLj, MP+GAEC+KLj and CGC+QPBO). We run the former 20 times to account for variability, and the latter only once. Finally, all methods were configured with the recommended values for their input parameters. We refer the reader to the corresponding articles for more details.

	$q_m = 0.1$		$q_m = 0.3$			$q_m = 0.5$		
Heuristics	q_{neg} 0.5	q_{neg} 0.7	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7	q_{neg} 0.3	q_{neg} 0.5	q_{neg} 0.7
GAEC+KLj	**1**	**1**	0.65	0.4	0	**1**	0	0
MP+GAEC+KL	**1**	**1**	0.65	0.4	0	**1**	0	0
ICP+GAEC+KL	0.9	0.4	0.15	0	0	0.05	0	0
GAEC+CGC+QBPO	0.75	0.75	0.15	0	0	**1**	0	0
SA	**1** (**1**)	**1** (**1**)	0.59 (0.75)	0.97 (**1**)	0.84 (**1**)	0.62 (0.8)	0.86 (**1**)	0.76 (**1**)
TS	**1** (**1**)	**1** (**1**)	**1** (**1**)	**1** (**1**)	0.9 (**1**)	**1** (**1**)	**1** (**1**)	**0.96** (**1**)
VNS	**1** (**1**)	**1** (**1**)	0.99 (**1**)	**1** (**1**)	**0.97** (**1**)	**1** (**1**)	0.96 (**1**)	0.84 (**1**)
GRASP	**1** (**1**)	**1** (**1**)	**1** (**1**)	0.9 (**1**)	0.26 (0.65)	**1** (**1**)	0.63 (**1**)	0.12 (0.55)
ILS	**1** (**1**)	**1** (**1**)	0.93 (**1**)	0.8 (**1**)	0.42 (0.95)	**1** (**1**)	0.4 (**1**)	0.19 (0.85)
MLMSB	**1** (**1**)	**1** (**1**)	0.86 (**1**)	0.99 (**1**)	0.7 (1)	**1** (**1**)	0.65 (**1**)	0.44 (0.78)

Table 2.3 – Average proportion of optimal solution discovery, by the considered 10 heuristics, based on the random networks from *Dataset 2.2* for $d = 0.5$, $q_m = \{0.1, 0.3, 0.5\}$ and $q_{neg} = \{0.3, 0.5, 0.7\}$. For each network, the first statistic (without parenthesis) shows how persistent the optimal solution discovery is over 20 repetitions. It equals 1 when an optimal solution is reached for each repetition. The second one (within parenthesis) is a binary value and equals 1 when a heuristic finds an optimal solution at least once out of its 20 repetitions, 0 otherwise. These statistics are averaged over 20 random networks from the same parameter set.

We start by comparing the heuristics based on the *Dataset 2.2*. We want to see how often these methods can find an optimal solution. For this purpose, we already computed the optimal $I(P)$ value

6. https://github.com/abailoni/GASP for GAEC+CGC+QPBO, and https://github.com/LPMP/LPMP for GAEC+KLj, ICP+GAEC+KLj and ICP+GAEC+KLj

for each instance of this dataset through the exact methods. The results are shown in Table 2.3. The table structure is similar to that of Table 2.2, with the exception that the rows correspond to heuristics. Each table cell depicts two average statistics over 20 random networks generated from the same parameter set. Both compute the average proportion of optimal solution discovery by the considered heuristics, but they differ from the way the optimal solution discovery is computed. For each generated network, the first one computes an average optimality value over 20 repetitions, whereas the second one (shown in parenthesis) is a binary value and equals to 1 when a heuristic finds an optimal solution at least once out of its 20 repetitions, 0 otherwise. Clearly, the first statistic shows how persistent the optimal solution discovery is over 20 repetitions. It gets 1, when an optimal solution is reached for each repetition.

We can summarize the results of the *Dataset 2.2* in three points. First, graph density d slightly affects the performances (not shown in the table), in that the frequency of finding an optimal solution over 20 repetitions slightly worsens when d decreases. But, the value of d does not change the overall trends observed, though. Second, when q_m and q_{neg} increase jointly, it is more challenging for heuristics to find an optimal solution. Finally, we identify three different types of heuristic performances: 1) those usually finding an optimal solution only on slightly imbalanced networks (GAEC+CGC+QPBO, GAEC+KLj, ICP+GAEC+KLj, MP+GAEC+KLj), 2) those usually finding an optimal solution, but not persistently over 20 repetitions (ILS, GRASP, MLMSB), and 3) those performing well overall (TS, VNS, SA). Of course, as we show later with *Dataset 2.4*, a method which is a type 3 on small networks does not necessarily get the best performance on larger networks, as increasing the graph order introduces other challenges in the solution improvement process. Nevertheless, if a heuristic hardly finds an optimal solution, even in these networks, this would put into question its underlying perturbation mechanism to escape from local optima. Indeed, this indicates that its performance severely depends on the quality of the first solution(s) generated. When this quality is not good at start, it is difficult to reach optimality (or high-quality solution in case of large networks) at the end.

We now compare the considered heuristics based on the *Dataset 2.3*. Remember that the considered networks are generated by incrementally perturbing structurally balanced signed networks thanks to the definition of stability range [176]. For this reason, this dataset allows us to answer when a heuristic starts to fail in finding an optimal solution. This aspect was missing in *Dataset 2.2*. Only the results for $n = 150$, $l_0 = 6$ and $d = 0.5$ are shown in Figure 2.2, as the other values give similar results. In this figure, each subfigure corresponds to a different heuristic. The x-axis represents the proportion of the perturbed edges with respect to the initial perfectly balanced signed network and this simply amounts to the *detected* graph imbalance $I(P)$. The y-axis is log-scaled and represents the gap to the considered q_m value (equivalently, gap to the optimal $I(P)$ value) in percentage. Since we ran heuristics in 20 repetitions, we show the results in a shaded region to illustrate the minimal and maximal gap values obtained.

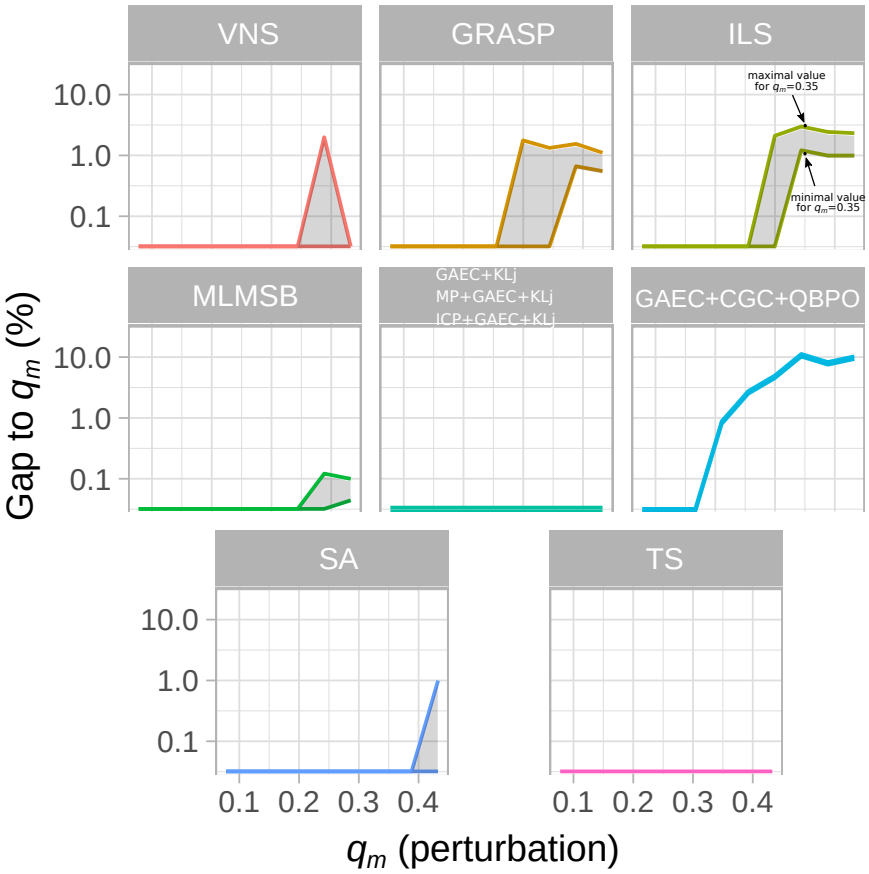


Figure 2.2 – Evaluation of the considered 10 heuristics based on the random networks from *Dataset 2.3* for $n = 150$, $l_0 = 6$ and $d = 0.5$. Each subfigure corresponds to a different heuristic. The x-axis represents the proportion of perturbed edges with respect to the initial perfectly balanced signed network and this simply amounts to the *detected* graph imbalance $I(P)$. The y-axis is log-scaled and represents the gap to the considered q_m value (equivalently, gap to the optimal $I(P)$ value) in percentage. Since we ran heuristics in 20 repetitions, we show the results in a shaded region to illustrate the minimal and maximal gap values obtained.

By looking at the results of Figure 2.2, we observe that TS, GAEC+KLj and the rounding methods ICP+GAEC+KLj and MP+GAEC+KLj always find an optimal solution, and the others fail at least once over the 20 repetitions. The performance of TS is as expected, since it belongs to type 3 in the typology identified for *Dataset 2.2*. However, the results of GAEC+KLj and its rounding methods are surprising, because they belong to type 1. More than that, the performance of ILS is also surprising, as it usually finds at least one optimal solution out of 20 repetitions in *Dataset 2.2*. However, it never finds any in this dataset, starting from $q_m = 0.3$. To summarize, all these observations highlight that both *Datasets 2.2 and 2.3* are complementary, since each can reveal different heuristic performances. In the literature, the heuristics are usually compared based on large real-world net-

works. However, considering *Datasets 2.2 and 2.3* seems to be also beneficial when evaluating the performances of the heuristics.

	Slashdot $n = 82,144$ $\|E\| = 498,532$	**Wikipedia Election** $n = 8,298$ $\|E\| = 99,917$	**E. coli** $n = 1,461$ $\|E\| = 3,215$	**Yeast** $n = 690$ $\|E\| = 1,080$	**Macrophage** $n = 678$ $\|E\| = 1,425$	**EGFR** $n = 329$ $\|E\| = 779$
GAEC+KLj	117,599	44,370	287	**36**	323	197
MP+GAEC+KLj	117,599	44,351	287	**36**	323	197
ICP+GAEC+KLj	117,599	46,651	281	**36**	326	209
SA	376,748	68,031	297	70	356	194
TS	**68,912**	47,239	306	37	341	203
VNS	69,212	61,100	280	**36**	324	**192**
GRASP	69,255	**34,460**	315	37	337	203
ILS	69,157	35,194	360	37	327	195
MLMSB	68,927	45,002	**266**	41	**316**	195

Table 2.4 – The best imbalance $I(P)$ detected by the considered 10 heuristics within the limit of 2 hours (4 hours for Slashdot due to its very large size) based on the six real-world signed networks from *Dataset 2.4*.

Finally, we evaluate the considered heuristics based on *Dataset 2.4*, which includes six real-world signed networks. We exclude GAEC+CGC+QPBO from the comparison, since it does not scale well. For each network, we ran the remaining 10 heuristics twice with a time limit of 2 hours. For Slashdot, the time limit was set to 4 hours due to its very large size. The results are shown in Table 2.4. Overall, we see that our previous observations do not hold for these networks, since there is not a single method, or a heuristic category, which always gives the best results. Even those belonging to type 1 in the typology identified in *Dataset 2.2* can be beneficial, depending on the network at hand. The most surprising result is the performance of SA in general. Its performance is worse than the others in all networks but EGFR and E. coli. In particular, it appears that it does not deal well with very large networks with a time limit of 2 hours, such as Slashdot. Another noticeable result is related to both rounding methods ICP+GAEC+KLj and MP+GAEC+KLj. In the experiments with *Datasets 2.2* and *2.3* they do not seem to improve GAEC+KLj. Nonetheless, we see from Table 2.4 that both methods can occasionally improve GAEC+KLj, as for the Wikipedia Election and E. coli networks.

To conclude this part, we evaluated the considered heuristics on three different datasets of different sizes. We empirically showed that all these datasets are useful in evaluating the performances of the heuristics. We can conclude that TS and VNS seem to be the best method as a primal heuristic in the design of an exact method to provide an upper bound. However, when it comes to larger networks, it is better to take all or most of these heuristics into consideration.

2.6 Conclusion

In this chapter, we first reviewed the state-of-the-art on exact methods and a large number of heuristics proposed for the CC problem, and then assessed their performances on a collection of signed networks. In our experiments related to exact methods, we showed that the choice of the formulation and its resolution method depends on the characteristics of the network at hand. When a network is sparse (resp. dense), B&C($F_e(G)$) (resp. B&C($F_v^\star(G)$)) better performs. Moreover, for a medium graph density, B&C($F_v^\star(G)$)) is less sensitive to increase in n than the other methods, hence preferable in this case. In our experiments related to heuristic methods, we showed that TS and VNS seem to be the best methods to be used as a primal heuristic (i.e. to provide upper bounds) in the design of an exact method. However, when it comes to larger networks, it is better to take all these heuristics into consideration.

Chapter 3

Characterizing measures for partition comparison

3.1 Introduction

The problem of comparing two partitions of the same set occurs in a number of situations. The most widespread is probably the assessment of clustering (or cluster analysis) and graph partitioning results. In the rest of this chapter, we will approach this problem from the perspective of graph partitioning, for the sake of consistency with this thesis. In such context, one wants to compare the estimated partition of a network with some ground-truth also taking the form of a partition. Alternatively, one has computed several such estimations, and wants to compare them to each other.

This comparison is traditionally performed through some measure able to quantify the similarity between two such partitions. These are called *external* measures, as they allow comparing the output of the partitioning method to an independent solution (generally the ground truth). In the rest of this chapter, we will simply call them *measures*, as there is no possible confusion in our context. Examples of such measures include Adjusted Rand Index (ARI) [114], Normalized Mutual Information [205] and so on. There are many ways to formalize what one means by "similar", resulting in the proposition of a very large number of measures over the years [225, 163]. In turn, this situation inevitably leads to the publication of a number of surveys aiming at reviewing and comparing all these measures [223].

In the literature, authors proposing new external measures follow a relatively standard workflow. First, they list some mathematical properties which they deem desirable in such measures, e.g. not being sensitive to the number ℓ of modules in a partition [184, 189, 95]. They then show that existing measures do not possess these properties. Finally, they solve this issue by proposing a new measure having these properties, or modifying an existing one to this end.

There are mainly two ways to check whether a measure has a given property. The most robust approach is to proceed analytically, through a mathematical proof (e.g. [162]). However, this task requires certain skills, and can be difficult or even impossible depending on the considered measure and property. Moreover, the proof is generally not transposable to other measures and properties, which makes it a one-shot effort. This is why the second approach, which is empirical, is much more frequent in the literature (e.g. [186, 189]). It consists in applying some predefined transformations to certain partitions, both designed in a way that is related to the property of interest, and to study how the measure reacts to these perturbations by using it to compare those partitions. For instance, to assess the sensitivity to ℓ, one could increase the number of modules in the transformed partition, and check how this affects the measure values.

Each application case is likely to bring its own constraints and requirements, so there is no such thing as a "best" measure that would fit *all* situations. One trait considered as positive in one case could very well be perceived as a drawback in another. However, due to the profusion of available measures, selecting the most appropriate one for a given situation is a challenge for the end user. As mentioned before, some survey articles try to compare them, but they focus only a small number of measures [223] and/or properties [3]. More importantly, the comparisons they perform are specific to these measures and properties [223], preventing the end user from including additional measures or properties in the comparison. In practice, the problem of selecting an appropriate measure to compare partitions is generally overlooked, and researchers tend to follow tradition and use the measures frequently appearing in the literature of their field.

In this chapter, we propose a new framework to solve this issue. It is based on the empirical approach mentioned above, and consequently relies on a set of predefined partitions and parametric partition transformations. We study the effect of each parameter on the measure through multiple

linear regression, in order to produce results that the end user can interpret. Our framework is not tied to any specific measure, property, or transformation, so it can be applied to any situation. We illustrate its relevance by applying it to a selection of popular measures. In addition to these contributions, we review the literature for desirable properties and the partition transformations used to test their presence, and propose a typology of the latter.

Contributions. The following content is based on our work published in *IEEE Access* [23]. This chapter makes the following contributions:

1. **Method.** We propose a new framework to study the effect of partition parameters on the measures through multiple linear regression.
2. **Evaluation.** We evaluate our framework on a synthetic dataset and a selection of popular external measures, and show its usefulness by extensively interpreting the obtained results.

The rest of this chapter is organized as follows. First, in Section 3.2, we review the literature on external measures, focusing on desired properties, partition transformations, and property assessment methods. Next, in Section 3.3, we introduce our own framework designed to study and compare measures and their properties. We put it into practice on a selection of widespread measures in Section 3.4 and discuss its results in Section 3.5. Finally, we review our main findings in Section 3.6, and identify some perspectives for our work.

3.2 Literature Survey

In this section, we perform a review of the literature, focusing on three aspects directly related to our work. We first discuss the desirable properties used to characterize measures (Section 3.2.1). We then survey the partition transformations proposed to empirically show the absence or presence of these properties (Section 3.2.2). Finally, we give an overview of the evaluation methods used for assessing and comparing the measures based on these transformations (Section 3.2.3).

3.2.1 Desirable Properties

As mentioned in the introduction, measures can be characterized in terms of a number of distinct desirable properties. There are many of them, sometimes with minor differences, and the same property is likely to appear under different names and forms in the literature: this makes it difficult to list them exhaustively and compare them. Here, we focus on the most frequently used, and propose a typology to ease their comparison. These properties are listed in Table 3.1, with a short description, as well as examples of popular measures known to possess them. When the bibliographic sources explicitly name the property, we use the same name in the table. Otherwise, we propose

a name based on its description. In the following, we distinguish three main categories, depending on whether the property is related to the *measure interpretation*, to the way it handles *random partitions*, and to its *sensitivity* to certain characteristics of the partitions.

3.2.1.1 Interpretation-Related Properties

The first category of properties is related to the interpretability of a measure, i.e. how easily its values can be understood by a human operator. This concerns the interpretation of a single value, i.e. what its magnitude means, but also the comparison of several values, and the interpretation of their difference.

Understandability [61, 191, 163, 177] means that the measure has a straightforward interpretation. For instance, the Rand index [187] (RI) is the proportion of vertex pairs for which both partitions agree. Other measures have less direct interpretations, for example the Standardized Mutual Information [191] (SMI) is a normalized version of the mutual information corresponding to the number of standard deviations the mutual information is away from the mean value, for a specific null distribution. At the other end of the spectrum, composite measures such as the F-measure [25] do not have a straightforward interpretation, as they combine other measures. This property is generally obtained by construction.

The *Fixed Range* property [225, 229, 223] means that the measure is designed so that its values are restricted to a predefined interval, which is often $[0, 1]$. This property eases the comparison of scores obtained on different partitions.

It is also the case of the *Value Validity* property [137]. Let m_1, m_2, m_3 and m_4 represent four numerical values obtained with some external measure for several pairs of partitions. In addition to order (size) comparisons such as $m_1 > m_2$, when a measure possesses the Value Validity property, the *difference* between several pairs of partitions, such as $m_1 - m_2 > m_3 - m_4$ or $m_1 - m_2 > k(m_3 - m_4)$ (for constant ℓ), can also be interpreted.

Convex Additivity [161, 162] concerns the case where one partition is a refinement of another partition (i.e. there is a hierarchical relationship between them). With a measure possessing this property, the difference in overall score can be expressed as a weighted sum of the score differences between individual modules.

3.2.1.2 Handling of Independent Partitions

The second category of properties focuses on how two independent partitions should be treated by the measure.

The *Constant Baseline* [225, 223, 191, 186, 2] property deals with statistical independence, i.e. the case where one compares two partitions sampled independently at random. This property specifies that in this situation, the measure should return a constant value. In practice, this constant

value is very often zero, in particular when the maximal value is 1 (cf. also the *Fixed Range* property), see for instance the Adjusted Rand Index [114] (ARI).

The traditional approach to bring this property to an existing measure is to apply a so-called *correction for chance*. It consists in subtracting to the measure the score estimated for two independent partitions, and possibly in normalizing the resulting expression, in order to get a fixed range. This is how Hubert & Arabie derived their Adjusted Rand Index [114] (ARI) from the original Rand Index [187], but the method had been used before in other contexts [102, 48]. Note that there is a number of ways to define the null model used to estimate the measure score under the assumption of independent partitions [98], with no consensus emerging regarding which of these models is the most appropriate.

Certain authors consider two independent partitions as the worst possible case [184], meaning that the resulting score should correspond to the measure minimal value. On the contrary, others make a distinction between independence and worst case [233, 97], a property that is called *Baseline-Minimum Distinction*. They generally place the constant baseline midway between the respective scores of the worst and base cases. This is for instance the case of the ARI, which ranges from -1 to $+1$, 0 being the constant baseline. In practice though, cases with scores lower than the constant baseline are rare, and have not been studied much in the literature [233].

3.2.1.3 Sensitivity to Partition Characteristics

The last category of properties concerns the sensitivity of the measures to certain characteristics of the compared partitions. The main such characteristics are the number of modules, the number of vertices, the size of the modules, and various descriptors allowing to express how similar the partitions are. These characteristics are often considered separately, and sometimes several at once.

In this category, the most frequent property is probably ℓ*-invariance*. Certain measures such as the Normalized Mutual Information tend to favor partitions depending on the number of modules they contain when compared with a reference partition [223], a bias that a number of authors want to avoid [225, 222, 232, 192, 9, 175]. For example, suppose that one compares a ground truth partition to two estimated partitions differing only in their number of modules. A biased measure will reach a noticeably higher value for one of these partitions due to this single difference.

By comparison, the *Discriminativeness* property relies on the difference in number of modules between the compared partitions [186, 9, 112]. It states that the measure score should decrease when this difference increases. Put differently, the score should be larger when both partitions contain similar numbers of modules than when they differ on this point.

The n*-invariance* property is analogous to the ℓ-invariance, except it is defined relative to the number of vertices in the partition [162, 225, 229], instead of the number of modules. It allows comparing measure scores computed on graphs of different sizes, as n-invariant measures are not

affected by such changes.

Authors do not agree on whether a measure should be sensitive or not to module size. This disagreement concerns partitions constituted of modules which are imbalanced in terms of size, i.e. containing large and small modules. Certain authors want the measure to focus mainly on the larger modules, as they consider smaller ones as negligible [229, 111, 166]. Others adopt the *Insensitivity to Module Size* property and assume that all modules are equally important regardless of their size, and that the measure should not be sensitive to module size imbalance [184, 189, 95].

Finally, some properties focus on how the measure should quantify the differences between pairs of partitions. Suppose we compare one primary partition to two different secondary partitions, resulting in two scores. The *Monotonicity* property states that the score of the most similar pair of partitions should be consistently higher or smaller (depending on whether the measure expresses similarity or dissimilarity) [189, 230, 97]. In addition, the *Proportionality* property states that the difference between these scores should be proportional to how close the secondary partitions are [151]. On the contrary, certain authors expect the measure score to rapidly change in presence of even the smallest differences, which corresponds to a non-linear behavior [184]. More generally, some authors want the measure to be *sensitive to small differences* [94, 162], whereas some others, on the contrary, want the measure to ignore what are considered as marginal differences [95]. It is important to stress that these are very generic properties, as the notion of proximity between two partitions can be understood in a number of ways.

3.2.1.4 Discussion

As explained in the introduction, and as summarized in Table 3.1, certain of the properties described in this section are obtained by construction, or verified through an analytical proof, whereas others are shown empirically, by applying specific transformations to a set of partitions. This is generally the case when the mathematical proof is impractical or too difficult to make.

In this article, we adopt an empirical approach, therefore we focus only on the latter type of properties. This includes the properties of our second (Comparison with Random Partitions) and third (Sensitivity to Partition Characteristics) categories. The framework that we propose does not necessarily handles the properties exactly as they are described here: we sometimes had to reformulate them to ease experiments and make the framework more generic. It relies on a set of variables similar to those used in the literature to define these properties (number of modules, number of vertices, module size distribution, etc.). Our framework is able to handle properties on which authors disagree, such as the sensitivity to module size distribution or to small differences.

Table 3.1 – Overview of the main desirable properties appearing in the literature, with examples of measures possessing them, and transformations used for their assessment.

Category	Desirable Property	Example measures	Related Transformations
Interpretation-Related	Fixed Range [225, 229, 223]	NMI [205], NVI [223]	— None (proof)
	Convex Additivity [162]	RI [187], Mirkin [167], VI [162], χ^2 distance [114]	— Splitting into unequal parts [162] (proof)
	Understandability [61, 191, 162, 177]	RI [163], JI [163], SMI [191], Split-Join [61]	— None (proof)
	Value Validity [137]	MI_c [137]	— None (proof)
Handling of Independent Partitions	Constant Baseline [225, 223, 191, 186, 2]	ARI [114], AMI [222], rNMI [232], RMI [175], FNMI [9], cNMI [139]	— Fragmenting every module [184] — Random shuffling [222, 191, 189, 192, 9, 98, 97]
	Baseline-Minimum Distinction [233, 97]	ARI [114], SMI [191]	— Random shuffling [97]
Sensitivity to Partition Characteristics	ℓ-invariance [225, 222, 232, 192, 9, 175]	ARI [114], VI [162]	— Random shuffling [222, 232, 192, 9, 97] — Splitting into singleton modules [175] — Swap with single module [189]
	n-invariance [162, 225, 229]	VI [162], FMI [94], NMI [205], ARI [114]	— None (proof)
	Discriminativeness [186, 9, 112]	ARI [114], GNMI [9], FNMI [9]	— Merging whole modules [186, 9] & Splitting into unequal parts [186, 9]
	Sensitivity to Small Differences [184, 94, 162, 95]	VI [162],FMI [94]	— Swap with all modules [184]
	Insensitivity to Module Size [184, 189, 95]	PSI [189]	— Swap with single module & remove [189] — Fragmenting a single module [189] — Random shuffling [111, 226]
	Monotonicity [189, 230, 97]	PSI [189], Element-centric [97]	— Merging a whole module with a part of other module [189] — Merging parts of different modules [60, 193] — Merging whole modules & splitting into equal parts [230] — Swap with single module [189] — Swap with all modules [189] — Random shuffling [97]
	Proportionality [162, 151] vs. Non-linearity [184]	Kappa index [151]	— Random shuffling [151]

3.2.2 Partition Transformations

Like for the desired properties, the literature exhibits a large number of different partition transformations, which are not always named, and when they are, not always similarly. This makes it difficult to identify and compare them. Here, we focus on the most frequent ones and use their most consensual names. Table 3.1 indicates the transformations used in the literature to assess the presence of each listed property. One can distinguish two types of partition transformations: random vs. deterministic.

3.2.2.1 Random Transformations

Random transformations consist in randomly distributing all the vertices of the reference partition over a number of modules to form the new partition. These transformations mainly differ in the probability distributions they rely upon. Such processes can be seen more as shuffling than transformations, as the original partition has no effect on the result. In essence, the goal is to obtain a partition as independent as possible from the original one. They are mainly used to check the existence of the Handling of Independent Partitions category of properties [222, 191, 189, 192, 9, 98, 97]. But several works leverage random transformations to look for other desirable properties, too. Certain authors force the shuffled partition to have various numbers of modules and imbalanced module sizes, in order to check the ℓ-invariance [222, 232, 192, 9, 97] and Insensitivity to Module Size [189, 226] properties, respectively. Others shuffle the original partition with an increasing level of randomness in order to test for the Monotonicity [97] and Proportionality [151] properties.

3.2.2.2 Deterministic Transformations

Deterministic transformations are used more frequently in the literature, probably because they offer a better control of the changes applied to the original partition. We distinguish five categories of such transformations. We call the first one *Remove*, and it consists in deleting some vertices from a module without erasing it completely. Although it is used to check the Insensitivity to Module Size property in the literature [189], it has the drawback of affecting simultaneously two aspects of the partition: module size, and number of vertices n. For this reason, it is not frequently used.

The second transformation category is *Split*, which consists in dividing a module into multiple smaller parts. Two variants mainly appear in the literature: splitting into *equal* [162, 163] vs. *unequal* parts [162, 163, 186, 9]. There is also a specific case of the first variant, consisting in splitting a module into only singleton modules [187, 188, 175]. This transformation category is used in the literature to test several distinct properties. Hierarchical splits (i.e. refinements of a partition) constitute an important part of the small experiments proposed by Meilă [162, 163], and allow to check the Convex Additivity property. Reichart and Rappoport [188] compare a reference partition to two estimated partitions differing mainly in their number of modules: slightly perturbed reference vs.

singleton modules. They expect that singleton modules are less similar to the reference, and a measure should not favor singleton modules in such a case (cf. ℓ-invariance property). Rabbany et al. [186] apply repeated split operations onto the ground truth of several real-world networks and then compare them to check the Discriminativeness property.

Transformation *Merge* is the reciprocal of Split, as it gathers nodes belonging to different modules into the same module. It also appears under three forms: merging a whole module with a *whole* other module [162, 186, 9, 230] vs. a *part* of another module [189], and merging parts of different modules [193]. Note that the last two transformations are not *pure*, in the sense that a Split is performed before the Merge. Regarding the desirable properties, since Merge is the reciprocal of Split, all the properties tested through Split can be also be tested by using Merge. On top of that, some authors leverage Merge to test for *Monotonicity*, in two different ways: Rezaei and Fränti [189] enlarge incrementally a specific module by moving vertices from the other modules, whereas Rosenberg [193] merge same-sized parts of each module to create new modules, which they consider as *noise*.

The next two transformations can be viewed as combinations of Split and Merge, and they are also frequently used in the literature. *Swap* consists in interchanging a number of vertices between pairs of (generally equal-sized) modules. In practice, this operation is usually repeated for each module, using one of two different forms: each module swaps vertices with *only one* different module [162, 189] vs. *all* other modules [162, 184, 189]. In the literature, the first form is mainly used with a range of the number of modules to check the ℓ-invariance property. In the experiments of Rezaei and Fränti [189], the authors keep the module sizes fixed, independently from the number of modules. However, this increases the number of total vertices, which arguably introduces a side effect in their experiments. The second form induces more perturbation of the original partition compared to first one, and the experiments in the literature mainly focus on the desirable properties related to this aspect, which are Monotonicity [189] and Sensitivity to Small Differences [184].

Finally, the idea behind the *Fragment* transformation is that vertices belonging to the same module in the original partition are placed in different modules in the transformed partition, as much as possible. Two variants mainly appear in the literature: fragmenting a *single* module vs. *all* of them. The former [189] is only used to change marginally the underlying partition structure, whereas the aim of the latter [187, 184, 111] is to obtain two maximally different partitions. In the literature, these variants are used to check the Insensitivity to Module Size [189, 111] and Constant Baseline [184] properties, respectively.

3.2.2.3 Discussion

Besides these categories, the literature also contains transformations which can be expressed as combinations of some of these categories [230, 186, 189]. It is important to stress that transformations are typically defined *ad hoc*, specifically to test for a particular property of interest, and on

some predefined partitions. For this reason, each author adopts a different angle, and it is hard to find two articles with the exact same methodology, targeting the exact same desired properties. In turn, this makes it difficult to compare transformations and measures from one paper to the other. To solve this issue, there is clearly a need for a unified view.

Another important limitation of the existing work is the lack of control over the original partition and its transformation. Some authors use a single parameter, for example the number of modules in the transformed partition [60, 191]. However, there are other aspects likely to affect the outcome, such as the number and size of the modules in the original partition, or the intensity of the transformation, and they are not taken into account simultaneously in the literature. This results in a relatively incomplete assessment of the measure properties.

In Section 3.3.1.2, we try to solve both these issues, by proposing a unified set of transformations designed to cover most of the literature, and by defining a set of parameters to get the appropriate level of control.

3.2.3 Assessment Methods

After having described the properties that authors want to find in partition comparison measures and the related partition transformations, we now turn to the methods used in the literature to check the presence or absence of these desired properties based on these transformations. We distinguish two families of approaches: visual inspection vs. statistical methods, more specifically correlation and regression.

3.2.3.1 Visual Inspection

Visual inspection is perhaps the most intuitive way to characterize the behavior of a measure. Typically, one plots the value of the measure as a function of some parameter used to control the partition transformation, e.g. the number of modules produced. Authors usually expect a monotonic trend, e.g. proportional increase or decrease in [97]. Some are more specific and look for a specific pattern, e.g. the so-called *knee shape* used in [186] for a parameter controlling the number of modules in the transformed partition. It requires the function to reach its maximum when the numbers of modules in the original and transformed partitions match, and to decrease when there are too few or too many modules in the transformed partition.

There are mainly two limitations to visual inspection. First, it is not an objective method, so limit cases can be difficult to judge. Second, it can handle only a very limited number of distinct parameters at once, especially if one wants to compare several measures and consider several properties, or assess how parameters interact. Statistical methods allow to solve the first issue, by providing an objective score. There are mainly two types of statistical tools used in the literature to assess measure properties: correlation and regression.

3.2.3.2 Correlation

A *correlation coefficient* quantifies the dependence between two random variables. In our context, and like with visual comparison, these variables are on the one hand the score computed with the measure of interest, and on the other hand a parameter controlling the partition definition or transformation. Many authors [112, 233] use the popular Pearson's product-moment correlation coefficient, which measures the linear dependence between the variables. Others use a rank-order correlation coefficient such as Kendall's (e.g. [229]) or Spearman's (e.g. [186]), which relies on the rank of the values rather than on the values themselves. Compared to Pearson's, such coefficients are able to detect a non-linear dependence, and can thus lead to different conclusions [186].

Besides objectivity, another advantage of correlation coefficients over visual inspection is that they summarize the dependence through a single value, which allows representing a number of pairwise relationships in a single table. However, this approach too becomes cumbersome when one wants to consider simultaneously a certain number of parameters and/or measures [60]. Moreover, multiple pairwise correlation values are not able to capture the potential interactions between the parameters (i.e. changing one parameter value may affect the partition or transformation feature controlled by another parameter).

3.2.3.3 Regression

Regression analysis does not suffer from this limitation, though. In its simplest form, it consists in describing the functional relation between a dependent variable and an independent variable [5]. In our context, those are the considered measure and a parameter of interest, respectively. However, multiple regression allows considering several independent variables at once, i.e. several parameters in our case. Another advantage over correlation is that the regression model can be used not only for interpretation, but also for prediction purposes [153].

To the best of our knowledge, the work of Saxena & Navaneetham [201] is the only one that uses multiple regression analysis to assess the similarity of external evaluation measures. The authors study the effects of three input parameters (module size, number of dimensions and number of modules) on a single measure (the ARI). On top of the regression, they also assess the significance of these effects, and compare the relative importance of the parameters through their associated regression coefficients.

As we will see in Section 3.3.2.2, our method goes in the same direction as Saxena and Navaneetham [201], but with a more complex model, for the following reasons. First, the set of transformations that we propose in Section 3.3.1.2 requires to handle more parameters, and therefore to include more independent variables in the model. Second, not only do we study the direct effect of each parameter on the measure, but also their interactions. Third, we consider several distinct measures, and we want to assess and compare the relative importance of the effects that the parameters

have on them, which requires a specific processing.

3.3 Proposed Framework

In this section, we describe the framework that we propose to analyze the behavior of a set of considered measures. It is independent from these measures, so we describe it in a generic way, for any selection of measures.

Our framework is constituted of two parts. The first one consists in characterizing the considered measures through the partition transformation-based principle mentioned in the Introduction (Section 3.3.1). The second part is to perform an appropriate regression analysis in order to interpret these characteristics and compare the measures (Section 3.3.2).

3.3.1 Characterization of the Measures

Our objective is to quantify how similar two partitions are through several external measures, under different scenarios, and then to assess how the resulting values are affected when one of the partitions undergoes systematic and controlled changes. Unlike the common approach taken in the literature, we generate the necessary data in a fully parametric way in order to get a greater control. For the same reason, our approach is deterministic.

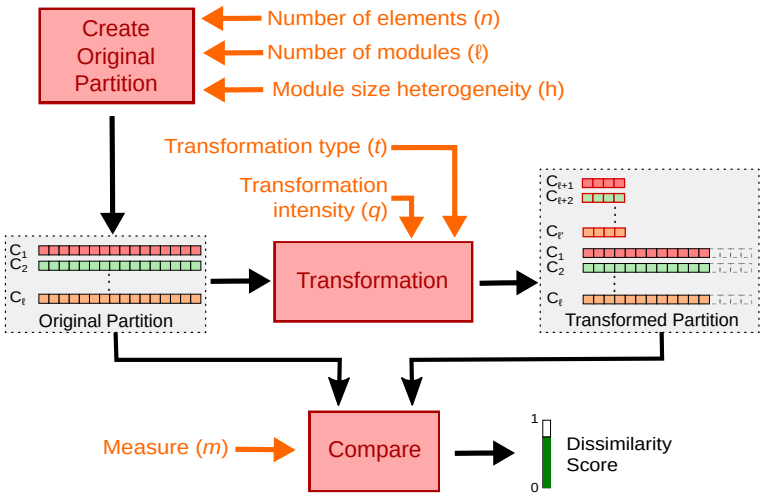


Figure 3.1 – General Framework, with parameters represented in orange. For illustration purposes, the ℓ *New Modules* transformation is used to produce the output partition with ℓ new modules.

Our three-step method is summarized in Figure 3.1, and detailed in the rest of this section. The first step is to create a base partition, called *original partition*, and controlled by three parameters (Section 3.3.1.1). The second step consists in applying to this partition a transformation controlled by

two other parameters (Section 3.3.1.2). This leads to a second partition, which we call *transformed partition*. Finally, the third step is to compute the selected external measures in order to assess how similar the original and transformed partitions are (Section 3.3.1.3). The whole process is repeated with an adequate number of different parameter values, in order to cover the parameter space.

3.3.1.1 Creating the Reference Partition

We control the generation of the reference partition through three parameters: the number of vertices n, the number of modules ℓ and the heterogeneity of the module sizes h. The first two parameters allow to control the most basic aspects of the partition. These are frequently targeted in the literature, albeit not always through explicit parameters.

The last one is much more uncommon, and lets us control how much module sizes vary in the same partition, and therefore to get more realistic module sizes. Similar concepts appear in the literature, for example when dealing with balanced vs. imbalanced module sizes, but not under the form of such a convenient parameter, to the best of our knowledge. It ranges from 0 to 1. When $h = 0$, all modules have the same size (i.e. so-called *balanced* module sizes), whereas they get imbalanced when $h > 0$, and the differences between their sizes increase when h gets closer to 1. More formally, the smaller module has a size of $s_1 = \alpha$ and the i^{th} smallest module has a size of $s_i = s_{i-1} + \beta$, whereas α and β depend on ℓ, n and h. In particular, β is proportional to h. See Appendix A.2 for details. This choice is a form of compromise allowing to obtain very heterogeneous module sizes even for a small n and/or a large ℓ.

3.3.1.2 Applying the Parametric Transformations

After having generated the original partition at the previous step, we now want to change it in order to get the transformed partition. Based on our review of the existing work (Section 3.2.2), we propose a set of five parametric transformations aiming at fulfilling several constraints. We want to cover most of the transformations used in the literature, in order to deal with as many desired properties as possible, while keeping our transformations as simple (and thus interpretable) as possible and avoiding overlap between them. We discard *Remove*, as it changes n, which in our context is a parameter of the first step of our process. As mentioned before, all our transformations are deterministic in order to offer better control.

We note t the *nature* of the transformation, and use it later as a categorical variable during the regression analysis. We define a parameter q to specify the *intensity* of the transformation, i.e. the proportion of vertices it involves. It ranges from 0, meaning no transformation at all, to 1, in which case the transformation involves all vertices. We want to give the same importance to all modules when applying the transformation, which means that it affects all of them. However modules may have different sizes, depending on the heterogeneity of module sizes h. To deal with this situation,

we make the number of vertices concerned by the transformation in each module proportional to the module size.

The five transformations that we propose are illustrated in Figure 3.2, on two example reference partitions (Subfigure 3.2a). Both contain $n = 72$ vertices, represented as numbered squares in the figure, and distributed over $k = 3$ modules, represented by colors. However, the top partition is balanced ($h = 0$) whereas the bottom one is moderately imbalanced ($h = 0.5$). Each other subfigure shows the partitions resulting from a specific transformation with intensity $q = 1/6$. Note that all these transformations allow to test by construction whether or not a measure is sensitive to some framework parameters. On top of that, they can be used to test certain desirable properties from Section 3.2.1, as explained in the rest of this section and summarized in Table 3.2.

3.3.1.2.1 ℓ New Modules, $t_{\ell nm}$

This transformation takes a predefined proportion of each module from the original partition, and creates a new module with these vertices, resulting in ℓ additional modules (Subfigure 3.2b). The effect of this proportion on the transformed partition is mirrored in 0.5. For instance, transforming 40% and 60% of the vertices give the same transformed partition. For this reason, we scale q so that it corresponds to twice this proportion, which allows us to keep the same $[0; 1]$ range as for the other transformations.

It is worth noting that the transformed partition is a subpartition of the original one, in the sense that each one of its modules is included in one original module. Parameters ℓ and h therefore affect the way the created submodules relate to the original modules. This transformation consequently allows testing for the Convex Additivity property, which states that a measure should not be affected when comparing refinements of the same partition. Concretely, we conclude that a measure has this property if it is not affected by ℓ and h when applying this transformation.

3.3.1.2.2 Singleton Modules, t_{sm}

All the vertices affected by this transformation become singletons, i.e. single-vertex modules (Subfigure 3.2c). This can be viewed as an extreme form of partition refinement, in the sense that each such singleton module is fully part of one of the original modules. Therefore, like ℓ New Modules, but to a lesser extent, this transformation allows testing the Convex Additivity property through parameters ℓ and h. Moreover, it allows checking the Sensitivity to Small Differences by considering the effect of parameter q. To be consistent with the nature of this property, it is necessary to focus on relatively small values of q (i.e. a limited transformation magnitude), though.

Parameter q can also be used to assess the Discriminativeness property, as increasing q largely increases the number of modules in the transformed partition. Therefore, a measure which is affected by an increasing q is likely to discriminate more between transformed partitions whose num-

ber of modules is closer to ℓ (and hence to possess this property [186]). This is particularly true when the measure scores cover the whole $[0; 1]$ range. Parameter ℓ can also be used, indirectly, to check the ℓ-invariance property. Indeed, the number of modules created by this transformation does not depend on ℓ, and is generally much larger than ℓ. So increasing ℓ changes noticeably the number of modules in the original partition, but not in the transformed one. Consequently, a measure which is marginally or never affected by changes in ℓ when undergoing this transformation can be considered as ℓ-invariant.

3.3.1.2.3 1 New Module, t_{onm}

Like the previous transformation, this one takes a proportion of each original module, but it gathers these vertices to create a *single* module instead of ℓ distinct ones (cf. Subfigure 3.2d). If we switch the original and transformed partitions, this transformation can alternatively be seen as the removal of a same-sized module, i.e. distributing proportionally the vertices of a single module over the others. This is similar to the transformations used in [189] to test for the Insensitivity to Module Size property. In our case, if increasing ℓ results in a substantial change in the measure score (all other things remaining equal), then this indicates that the measure is likely to treat the modules equally, i.e. that it holds the property [189].

3.3.1.2.4 Neighbor Module Swaps, t_{nms}

This transformation moves a proportion of each module into its neighbor module. Each module swaps vertices with exactly one different module (cf. Subfigure 3.2e). Like for ℓ New Modules, the effect of this proportion on the transformed partition is mirrored in 0.5 for certain values of h. We therefore rescale it in the same way as before, in order to obtain a parameter q ranging from 0 to 1. This transformation allows to test for the Insensitivity to Module Size property through parameter h. By design, the number of modules in the original and transformed partitions are the same. Hence, this transformation does not interfere with ℓ and h. If increasing h has a substantial effect on the measure score, then this indicates that the measure is not likely to treat the modules equally, i.e. it does not hold the property.

3.3.1.2.5 Orthogonal Modules, t_{om}

This transformation uses a proportion of each module to create new modules, in such a way that all of their vertices come from different original modules (cf. Subfigure 3.2f). The resulting modules are orthogonal to the original ones, in the sense that each original module is represented equally in the new modules.

Applying this transformation with different values of ℓ has an effect on the number of modules

in the transformed partitions, such that the difference in number of modules between the original and compared partitions substantially decreases, when ℓ increases. This is similar to the transformations used in [97]. The main difference is that the authors shuffle completely the transformed partitions, whereas this randomization process is tuned with the parameter q in our case. Therefore, like in [97], this transformation can be used, to a lesser extent, to test for the ℓ-invariance property. If a measure is marginally or never affected by changes in ℓ when undergoing this transformation can be considered as ℓ-invariant.

Moreover, this transformation can also be used to check the Proportionality property with parameter q, as in [151]. If the scores of a distance measure increase linearly with increasing values of q, then we say that the measure validates this property. Finally, like in *Singleton Modules*, the Sensitivity to Small Differences property can be also checked through small values of q, i.e. a limited transformation magnitude [184].

Property	**Transformation & Parameter**	**Description**
ℓ-*invariance*	t_{sm} & ℓ [175]	The measure is marginally affected by changes in ℓ when undergoing this transformation.
	t_{om} & ℓ [97]	The measure is marginally affected by changes in ℓ when undergoing this transformation.
Discriminativeness	t_{sm} & q [186]	Increasing q results in a substantial change in the measure score for this transformation.
Insensitivity to Module Size	t_{onm} & ℓ [189]	Increasing ℓ results in a substantial change in the measure score this transformation.
	t_{ncs} & h [189]	The measure is marginally or never affected by this transformation, for increasing h.
Convex Additivity	t_{sm} & ℓ, h [162]	The measure is not affected by ℓ or h for this transformation.
	$t_{\ell nm}$ & ℓ, h [162]	The measure is not affected by ℓ or h for this transformation.
Proportionality	t_{om} & q [151]	The measure score increases proportionally with q.
Sensitivity to Small Differences	t_{om} & q [184]	Even small values of q have a substantial effect on the measure score.
	t_{sm} & q	Even small values of q have a substantial effect on the measure score.

Table 3.2 – The six desirable properties selected from Section 3.2.1, together with the framework parameters and transformations that allow testing them. The bibliographic references indicate matching situations from the literature, when available.

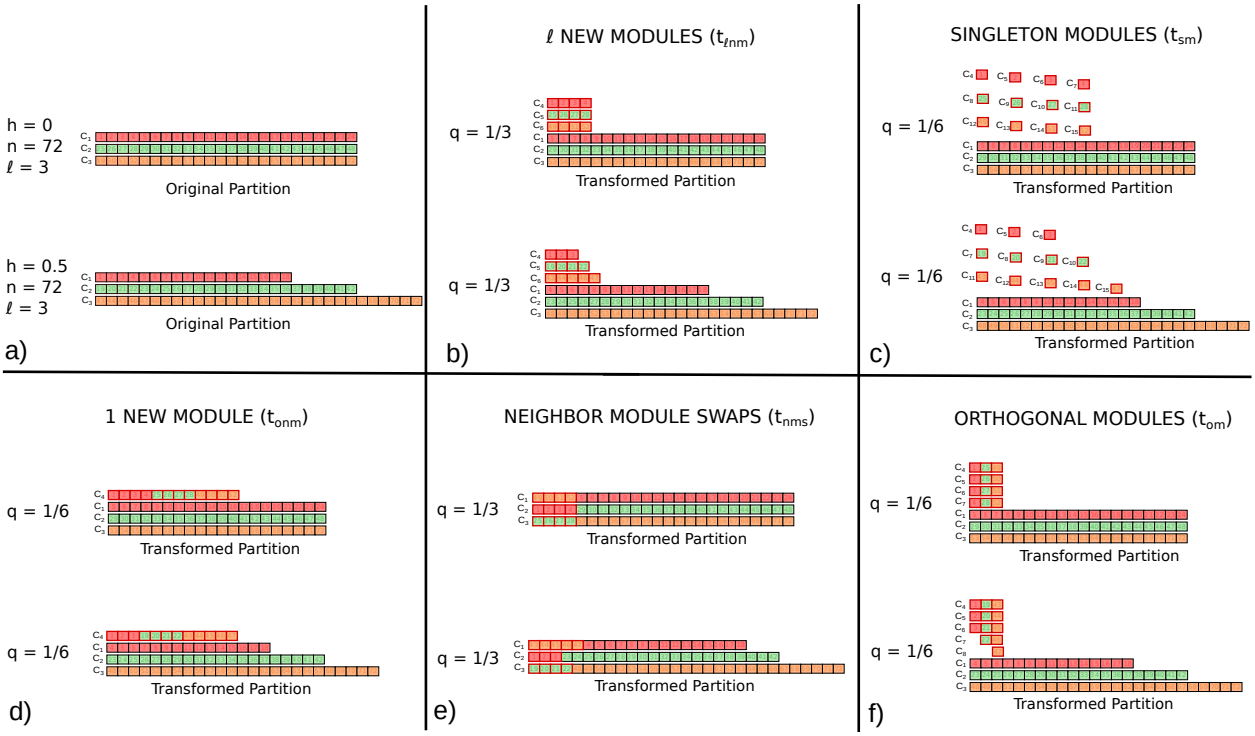


Figure 3.2 – Parametric partition transformations used in our framework. Subfigure a) shows two reference partitions containing both $n = 72$ vertices and $k = 3$ modules, but differing on the heterogeneity of the module sizes: balanced ($h = 0$) vs. moderately imbalanced ($h = 0.5$). The 5 transformations, illustrated in Subfigures b)–f), are applied to these two original partitions to produce corresponding transformed partitions. Transformation intensity is $q = 1/6$, or equivalently $1/3$ for both transformations concerned with rescaling.

3.3.1.3 Computing and Normalizing the Measures

The third step is very straightforward and simply consists in computing the measures for each pair of partitions generated, in order to compare the reference partition with each transformed partition. Note that during the regression analysis, the measure of interest is considered as a categorical variable noted m.

In order for these values to be comparable, one has to make sure they respect two constraints, though. First, some measures of the literature quantify the similarity between two partitions, whereas others assess their *dis*similarity. For comparison purposes, all measures compared within our framework should express the same concept, be it similarity or dissimilarity. Without loss of generality, we assume in the rest of our framework that all considered measures are dissimilarity measures (possibly after having undergone an appropriate transformation).

Second, all measures are not necessarily defined on the same range, which means that some of them must be normalized in order to allow comparison. Many measures are defined on $[0; 1]$, so

this seems like a consensual choice.

3.3.2 Regression Analysis

The second part of our framework consists in analyzing all the dissimilarity values obtained during the first part. In the following, we first introduce our proposed regression model (Section 3.3.2.1). We then turn to relative importance analysis (Section 3.3.2.2), which aims at determining how much the framework parameters affect the measures depending on the applied transformations.

3.3.2.1 Model Design

In our context, the dependent variable is a dissimilarity score in $[0, 1]$, which we note y, whereas the independent variables correspond to the five parameters of the framework (n, ℓ, h, q, t) and the nature of the measure used to compute the score (m). Four of them are therefore quantitative (n, ℓ, h and q), and two are categorical (t and m).

We study the relation between these variables through a multiple linear regression model. Note that in this type of model, the linearity constraint concerns the regression coefficients, and not the independent variables. This means that independent variables can appear as polynomial terms in the model, and that the model can contain interaction terms corresponding to products of independent variables. There exist more complex types of regression models (e.g. polynomial regression), which could better fit our data. We chose to use a linear regression nevertheless, because it is much more interpretable [49], a property which is particularly important in our case.

The presence of *categorical* independent variables makes it necessary to adopt a specific approach, by comparison to a straightforward model including only numeric dependent variables, and there are several methods to do so [49]. Among them, we decide to use so-called *dummy variables*, as they allow to avoid splitting the model in several parts, which in turns makes it easier to compare the estimated regression coefficients [108].

Our multiple linear regression model is as follows

$$
\begin{aligned}
y = \sum_i \sum_j \Big(& \beta_{0ij} t_i m_j \\
& + \beta_{1ij} n t_i m_j + \beta_{2ij} k t_i m_j + \beta_{3ij} q t_i m_j + \beta_{4ij} h t_i m_j \\
& + \beta_{5ij} n k t_i m_j + \beta_{6ij} n h t_i m_j + \beta_{7ij} n q t_i m_j \\
& + \beta_{8ij} k h t_i m_j + \beta_{9ij} k q t_i m_j + \beta_{10ij} h q t_i m_j \Big) \\
& + \epsilon,
\end{aligned}
\tag{3.1}
$$

where the $\beta_{.ij}$ are the regression coefficients, t_i and m_j are the dummy variables, and ϵ is the common error, which is assumed independent and normally distributed with mean 0 and standard

deviation σ. Each dummy variable is binary, and represents one specific value of a categorical variable: transformations for t_i ($1 \leq i \leq T$) and measures for m_j ($1 \leq j \leq M$), where T (resp. M) is the number of transformations (resp. measures). The model focuses on various types of interactions between the independent variables. The second line contains terms describing interactions between the categorical variables and each *single* numeric variable. The third line deals, in addition, with interactions between *pairs* of quantitative variables. These terms are likely to introduce some amount of collinearity with the corresponding terms from the previous line. In order to solve this issue, we center all the quantitative independent variables [135]. In order to keep the model interpretable, we do not include any higher order term.

3.3.2.2 Relative Importance Analysis of Independent Variables

As this stage, we have a multiple linear regression model able to represent the relations between our framework parameters and the scores of the measures. Next, we want to assess the relative importance (also called relative strength [107] or effect size [218]) of the terms constituting our model.

In our context, *relative importance* refers to the contribution of an independent variable, by itself and in combination with other independent variables, to the prediction or the explanation of the dependent variable [121]. Such notion can be formalized in a number of ways, therefore several methods have been proposed [121], originating from different research fields. Nevertheless, they are designed with a common goal in mind, which is to handle both problems frequently occurring in multiple regression analysis and making this task challenging: 1) multi-collinearity between independent continuous variables; and 2) non-linearity of regression models. Since our independent variables are perfectly uncorrelated by design, and since we consider a purely linear model, all of these methods are relatively equivalent in our case. Therefore, we select the most straightforward approach, consisting in using squared standardized regression coefficients (SRC), or *squared* β *weights* [121, 172], to assess the relative importance.

When the regression terms are by design perfectly uncorrelated, zero-order correlations and β weights are equivalent [121]. Thus, squared β weights sum to the explained variance of the dependent variable [121], generally noted R^2. This implies that squared β weights can be used as a means of decomposing R^2 according to the terms of the model [172]. That is, a squared β weight close to zero makes a regression term less important, from which we can deduce that it does not play a key role in explaining the observed variance for the dependent variable y.

Having a similar beta weight is not sufficient to conclude that two terms have the same importance: the significance of their difference must be statistically tested [108]. In the presence of such significance we can confirm the superiority of the same variable in one transformation type (similarly, for one measure) over the others. The importance analysis framework includes this test for all pairs of β weights.

3.4 Experimental Setup

In order to illustrate how to use our framework and interpret its results, we now apply it to a selection of popular external measures. In this section, we define our experimental setup. We first describe briefly these measures (Section 3.4.1), before turning to the dataset and the regression assumptions (Section 3.4.2). The results are presented afterwards, in Section 3.5.

3.4.1 Selected Measures

In the literature, external measures are divided into three main categories based on the basic principle they rely upon [225, 163]: 1) Pair-counting, 2) Set-matching (or set overlaps) and 3) Information-theory. Among them, the pair-counting measures are the most studied ones. In line with this, for our experimental setup we select 6 widely used measures covering all three categories, with a prevalence of pair-counting measures. The formal description is given in the Appendix (Section A.1): in this section, we focus on the principle underlying these measures, as well as their similarities and differences.

A pair of vertices can be handled in only two different ways in a given partition: either they belong to the same module or to two different modules. *Pair-counting* measures are based on the idea of comparing how two partitions of the same graph handle each pair of vertices. For a given pair, there is *positive agreement* between the partitions if its vertices belong to the same module in both partitions; *negative agreement* if they belong to different modules in both partitions; and *disagreement* otherwise. The *Rand Index* (RI) [187] is the proportion of agreement relative to the total number of vertex pairs. Hubert and Arabie's *Adjusted Rand Index* (ARI) [114] is based on the RI, but additionally includes a *correction for chance*. The *Jaccard Index* (JI) was originally defined to compare sets [118], but it is also used as an external measure [30]. It completely ignores negative disagreements, as it corresponds to the proportion of positive agreements relative to the number of disagreements and positive agreements. The *Fowlkes-Mallows Index* (FMI) [94] also ignores negative agreements, as it is based on a score corresponding to the proportion of positive agreements relative to the number of pairs belonging to the same module *in one partition*. This score is computed separately for each one of the two compared partitions, and the Fowlkes-Mallows Index is the geometric mean of the resulting values.

To represent the category of *set-matching* measures, we select the F-measure (F). Note that this name is sometimes used in the literature as a synonym of *harmonic mean*, and therefore covers several distinct measures (e.g. [186, 98]). We use the definition of Artiles et al. [25], according to which the F-measure is the harmonic mean of two quantities called *Purity* and *Inverse Purity*. In order to compute the Purity of a module from the first considered partition, one needs first to identify the module from the second partition with which it has the largest intersection. The Purity then corresponds to the proportion of the first module which belongs to this intersection. The Purity

of the first partition is the total purity of its modules. The Inverse Purity is simply the Purity of the second partition relative to the first. Finally, the F-measure is the harmonic mean of the Purity and Inverse Purity.

Information-theoretical measures are generally based on the notion of *Mutual Information* [50]. The principle behind these measures is to consider each partition as a categorical random variable, whose possible values are the modules. The mutual dependence between these variables can then be interpreted as the similarity between the partitions. There are a number of variants of the notion of mutual information, in particular several normalizations have been proposed (see for instance [223]). In this work, we focus on the sum normalization as defined in [205], which is very widespread, and results in the so-called *Normalized Mutual Information* (NMI).

As mentioned in Section 3.3.1.3, our framework expects that all measures express the same concept, either dissimilarity or similarity, and that they are all defined on the same fixed range. Regarding the latter point, all the selected measures are originally ranging from 0 to 1 except ARI, which can output negative values in theory. However, in practice it is very rare to get negative values for ARI. In the context of our experiments, it is always positive, so we decided not to perform any additional change. Regarding the former point, we adjust our selected measures through a simple subtraction, so that they all quantify the *dissimilarity* between partitions. We note the resulting measures as follows: D_{RI} (Rand Index), D_{ARI} (Adjusted Rand Index), D_{FMI} (Fowlkes-Mallows Index), D_{JI} (Jaccard Index), D_F (F-measure) and D_{NMI} (Normalized Mutual Information).

3.4.2 Dataset and regression assumptions

We generate our data through the process presented in Section 3.3.1, using the following parameter values. For the number of vertices n, we choose values arithmetically compatible with the desired numbers of modules, ranging from $3,240$ to $12,960$ with increments of $1,080$. The number of modules ℓ ranges from 2 to 11. The heterogeneity of the modules size h ranges from 0 to 0.9 with increments of 0.1. Regarding the transformations, their intensity q ranges from 0.1 to 1, also by increments of 0.1, and the nature t of the transformation itself is one among t_{sm} (*Singleton Modules*), t_{onm} (*1 New Module*), $t_{\ell nm}$ (ℓ *New Modules*), t_{ncs} (*Neighbor Module Swaps*), t_{om} (*Orthogonal Modules*), as defined in Section 3.3.1.2. In the end, the different combinations of our parameter values produce a total of $50,000$ pairs of partitions.

There are several standard assumptions to check before performing a linear regression [107, 135, 49]: 1) sufficient sample size, 2) linear relationships, 3) no or little multicollinearity, 4) multivariate normality, and 5) homoscedasticity. We review them here for our dataset and framework. First, our sample size of $50,000$ observations is large enough for getting reliable estimates of the regression. Second, after a visual inspection we observe that the relation between the dependent variable and the independent variables appear to be linear, except for ℓ and q in which case it looks rather curvilinear. We stick to the linear model for the sake of readability and understandability, though.

Third, by design of our dataset, the observations are independent and there is no collinearity between the independent variables. Fourth, the large size of our dataset makes the possible presence of outliers unlikely to affect our results [86]. For the same reason, the central limit theorem guarantees that the residuals will be approximately normally distributed. Fifth, a visual inspection reveals that the variance of y increases with parameters q (intensity of the transformation) and ℓ (number of modules), which means the data are not completely homoscedastic. The standard way of solving this issue is to introduce non-linear terms in the model [107, 135, 49], but again we want to keep it simple, and moreover the observed level of heteroscedasticity does not prevent us from interpreting the regression coefficients [107].

3.5 Results and Discussion

We now assess, compare and discuss the performance of the considered measures when applied to the generated dataset. We first show the relevance of our method through visual inspection (Section 3.5.1), then present our results in further detail (Section 3.5.2). Our source code is publicly available [1].

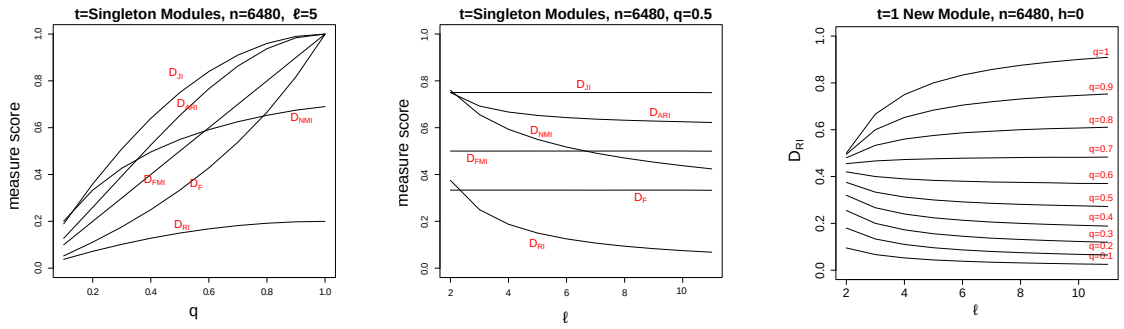
3.5.1 Visual inspection

To show the relevance of our method we highlight two aspects of our analysis through the visual inspection of Figure 3.3: 1) slope coefficients and 2) interaction between parameters. As we will see in Section 3.5.2, those aspects allow our method to identify similarities and differences between the considered measures, and therefore to discriminate between them.

Plot 3.3a shows how the measure scores evolve as functions of q, for the *Singleton Modules* transformation, while the other parameters are fixed to arbitrary values. One can observe that all the scores increase with q, albeit in different ways. Overall, D_{RI} has the smallest slope coefficient, followed by D_{NMI}, and they are therefore the least sensitive to this transformation. We observe that D_{JI}, D_{ARI}, D_{FMI} and D_F get similar scores for extreme q values, but are relatively different when q gets closer to 0.5. Plot 3.3b is built upon the same principle, except it focuses on ℓ instead of q. As before, all measures differ in terms of absolute score values, but this time one can detect similar certain trends. In particular, D_{JI}, D_{FMI} and D_F remain unchanged, whereas D_{RI}, D_{ARI} and D_{NMI} decrease with ℓ. These two plots show that our framework is able to produce situations for which the measures behave differently. Moreover, they also show that the slope coefficients, which constitute the basis of our analysis, are able of capturing these differences.

As mentioned in Section 3.2.3.1, the common way to assess the performances of the measures is trough visual inspection, which requires fixing many parameters, as we did just now, as such plots

1. https://github.com/CompNet/ExtMeasEval.



(a) (b) (c)

Figure 3.3 – (a) Score of each measure as a function of q, for the *Singleton Modules* transformation. (b) Score of each measure as a function of ℓ, for the *Singleton Modules* transformation. (c) D_{RI} score as a function of ℓ, for the *1 New Module* transformation, and for several values of q.

are able to handle only a limited number of parameters at once. Plot 3.3c illustrates the limitation of this approach by showing the evolution of the D_{RI} score for the *1 New Module* transformation, as a function of both ℓ and q. When considering only ℓ, the D_{RI} score is always monotonic. However the nature and slope of the trend depend on q: increasing for $q \geq 0.7$ vs. decreasing for $q < 0.7$. This means that there is an interaction between both parameters. This type of joint effect between parameters is hard to detect when using only plots, as it requires considering all possible combinations of parameters. However, it is captured by the interaction terms present in our regression model, as we will see in Section 3.5.2.

3.5.2 Relative importance analysis

We first discuss the effect of the framework parameters on each measure (Section 3.5.2.1), and compare them. Along with our discussion, we identify the desirable properties possessed by each measure, as well. We then show how this analysis can be leveraged to derive a typology of the measures (Section 3.5.2.2).

3.5.2.1 Effect of the Parameters

We show all the results from our relative importance analysis in Figure 3.4, using stacked barplots. We describe these plots globally here, for matters of convenience, before interpreting them in the rest of this section. The figure contains 6 barplots (i.e. subfigures), each one corresponding to a specific dissimilarity measure. Each barplot is constituted of 5 stacked bars, each one corresponding to a different transformation. The segments constituting these stacked bars represent the regression terms from (3.1). Their colors and order match the legend, and their height corresponds to the associated regression coefficient β in (3.1). More precisely, the segment heights are

proportional to the square root of the squared β coefficients.

The larger the segment height, the more important the regression term for the measure and transformation represented by the considered stacked bar. The values they represent are unitless, and we perform no upper bound normalization in order to ease comparisons between transformations and measures. Differences between segment heights are not always statistically significant, though. The exhaustive list of significant differences at p-value ≤ 0.05 is given in Appendix (Figures A.1 and A.2) for the sake of completeness. However, we find it difficult for the reader to cross-check them systematically with Figure 3.4. It is more intuitive to use the following rule of thumb: if one can visually detect a difference between two bars of Figure 3.4, then it is statistically significant.

Finally, there is a last bit of information in Figure 3.4, under the form of triangles placed over certain segments and representing monotonic behaviors. Upward (resp. downward) triangles indicate that the measure score consistently increases (resp. decreases) when the concerned parameter increases, independently from the other parameters. This information can be seen as complementary to the relative importance analysis. Suppose that a given parameter is similarly important for several measures, i.e. it affects them to roughly the same extend. The triangles allow distinguishing the measures qualitatively, based on the nature of this effect.

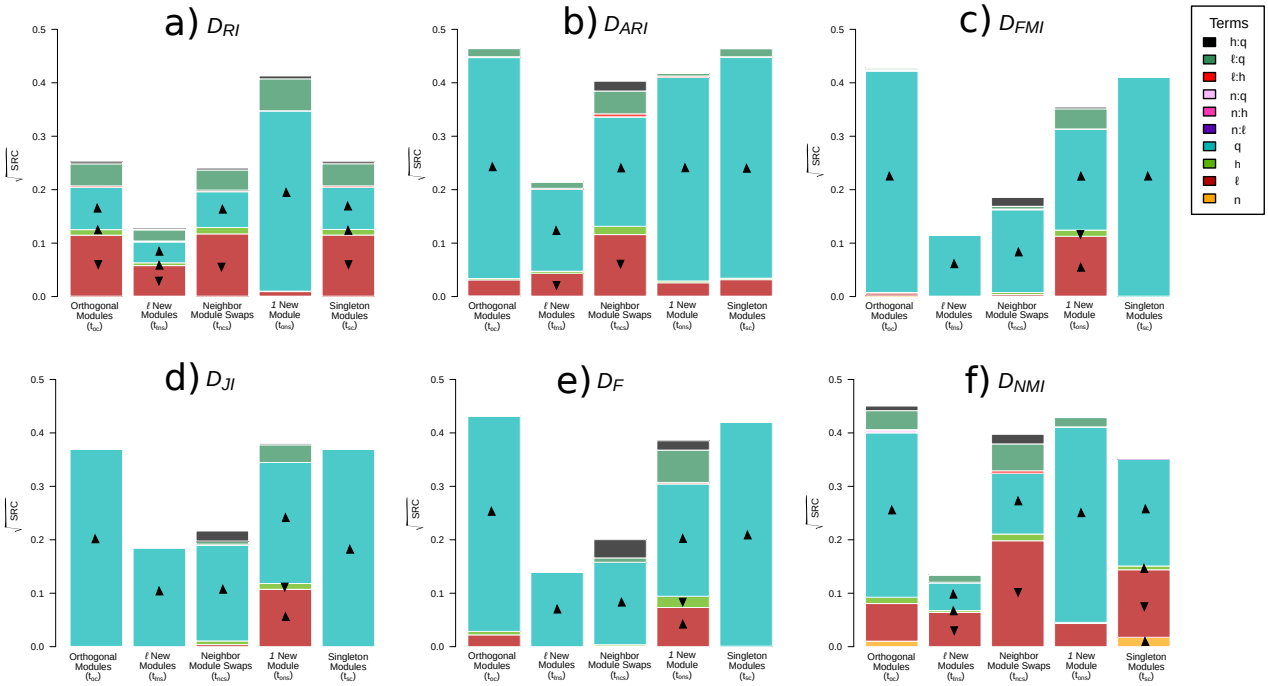


Figure 3.4 – Results of the relative importance analysis, for measures a) D_{RI}, b) D_{ARI}, c) D_{FMI}, d) D_{JI}, e) D_F and f) D_{NMI}. The order of the terms in each bar is shown in the legend. The relative importance scores represented on the y-axis are square-roots. Upper (resp. lower) triangles indicate an increasing (resp. decreasing) trend of measure scores, when the corresponding parameter increases, independently of the values of the other parameters.

Overall, we can observe that all measures are strongly affected by q, and to a lesser extent by ℓ and h. On the contrary, n has close to no effect on the measures. This effect of q on all measures also appears under a different form in Figure 3.3a. As shown by the triangles in Figure 3.4, the measure score increases with q in all cases. This general behavior is intuitively sound, as q controls the intensity of the transformation. There are differences, as illustrated in Figure 3.3a, in the way the measures are affected by q and the other parameters, though, and we can also see some punctual effects due to interactions between parameters. In the following, we consider each measure and discuss the results displayed in Figure 3.4.

3.5.2.1.1 RI

We can distinguish roughly two categories of transformations regarding D_{RI}, depending on how the measure is affected by the parameters. The first category contains only *1 New Module*, for which we observe a sensitivity almost twice as important as for the other transformations, which is unique among the considered measures. Also, this transformation exhibits a very strong effect of q, and to a lesser extent of the interaction between q and ℓ, as also illustrated in Figure 3.3c from a different perspective. The second category contains the rest of the transformations, for which parameter importance is more balanced between q, ℓ, and their interaction.

As mentioned in Section 3.4.1, D_{RI} considers positive and negative agreements equally. When applying a transformation of the second category, an increase in q causes the number of positive agreements to decreases, whereas the negative agreements are largely preserved. This prevents D_{RI} from using its whole nominal range $[0, 1]$, as also pointed out by Meilă [163] and Vinh et al. [223]. This in turns explains the observed smaller effect of q. As explained in Section 3.3.1.2.2, we can infer from a small q effect for *Singleton Modules* that D_{RI} does not possess the Discriminativeness property, a conclusion that confirms the results of Rabbany et al. [186].

On the contrary, there is a relatively substantial effect of ℓ for the transformations of the second category, which is due to them largely preserving negative agreements, as already noticed for q. As explained in Section 3.3.1.2, the large effect of ℓ for the *Singleton Modules* and *Orthogonal Modules* transformations indicates that the measure is not ℓ-invariant. Similarly, the large effect of ℓ for the *Singleton Modules* and ℓ *New Modules* transformations indicates that it does not have the Convex Additivity property. These findings are in line with the results of Rabbany et al. [186] and Amelio & Pizzuti [9] regarding ℓ-invariance, and Meilă [163] regarding Convex Additivity.

The relatively small effect of ℓ for *1 New Module* shows that D_{RI} is sensitive to variations in the module sizes (cf. no Insensitivity to Module Size), as already pointed out by Rezaei and Fränti [189] for pair-counting measures. The absence of any significant effect of h in the results of *Neighbor Module Swaps* corroborates this finding. Regarding the remaining parameters, *1 New Module* is also the only transformation which seems not to be affected by h. Finally, n does not seem to affect D_{RI} at all.

3.5.2.1.2 ARI

Overall, q has a much stronger effect on D_{ARI} when compared to D_{RI}, which results in a total sensitivity approximately twice as large for all transformations except *1 New Module* (which is already large in D_{RI}). Based on the effects observed for *Singleton Modules*, D_{ARI} seems to validate the Discriminativeness property much more than D_{RI}.

The effect of ℓ is much lower than in D_{RI}, for all transformations except *Neighbor Module Swaps*. According to certain results obtained by Meilă [163] for a similar transformation, the correction term in D_{ARI} can be sensitive to variations in the module number and sizes, which may explain our observation. Regarding *1 New Module*, the effect of ℓ in D_{RI} is already small, and the correction present in D_{ARI} only slightly increases it. Therefore, D_{ARI} is sensitive to the variations of the module sizes, too (cf. Section 3.3.1.2.3).

The effect of ℓ observed for *Singleton Modules* and *Orthogonal Modules* is much smaller than in D_{RI}, which indicates that the measure is ℓ-invariant (Sections 3.3.1.2.2 and 3.3.1.2.5). This is consistent with the fact that D_{ARI} was designed specifically to make D_{RI} ℓ-invariant, a property already verified empirically by Rabbany et al. [186]. However, this effect is still noticeable, which shows that the measure is not completely ℓ-independent. Similarly, the effect of ℓ for ℓ *New Modules* is smaller than in D_{RI} but still considerable. Based on these two observations, we can conclude that D_{ARI} does not possess the Convex Additivity property (Section 3.3.1.2.1).

The introduction of chance correction has a side-effect on h, as it has a much smaller effect on D_{ARI} compared to D_{RI}, for all transformations. This is consistent with a similar observation pointed out by Romano et al. [192]. Interaction-wise, the effect of ℓ:q is much weaker than in D_{RI}, probably due to the lower overall effect of ℓ, except for *1 New Module*. Finally, n does not seem to affect D_{ARI} at all.

3.5.2.1.3 FMI & JI

We jointly discuss both other pair-counting measures, D_{FMI} and D_{JI}, because their results are very similar and differ only on the magnitude of the effect of q. The main difference with the other measures is that q is the only perceptible effect for three transformations: *Orthogonal Modules*, ℓ *New modules* and *Singleton modules*. Consequently, both measures differ from the two previous ones regarding certain desirable properties. First, like D_{ARI} but unlike D_{RI}, both measures possess the Discriminativeness property. Second, unlike D_{RI} and D_{ARI}, they seem to validate the Convex Additivity property. It is worth stressing that, in theory, D_{FMI} and D_{JI} are not supposed to possess this last property [162], *strictly* speaking. However, our results show that in practice they behave as if they do, at least *to some extent*, and under some conditions (here: when the number of vertices n is large enough).

The effect of ℓ is negligible for all transformations but *1 New Module*, i.e. the second category of

transformations previously identified for D_{RI}. These transformations affect only marginally negative agreement, which explains why the effect of ℓ is so small here, compared to D_{RI}. This effect is small for *Singleton Modules* and *Orthogonal Modules*, so we can conclude that both measures appear to validate the *ℓ-invariance property* (Sections 3.3.1.2.2 and 3.3.1.2.5). The strong effect of ℓ for *1 New Module* indicates that these measures possess the Insensitivity to Module Size property (Section 3.3.1.2.3).

Regarding the other effects, one can observe that unlike D_{RI} and D_{ARI}, h has a small effect only for *1 New Module*. Furthermore, not only do ℓ and q have a strong effect for this transformation, but their interaction does too. Finally, overall, n has no significant effect on both measures.

3.5.2.1.4 F-measure Unlike the previous measures, which rely on pair-counting, D_F is based on set-matching. Nevertheless, the observed effects are very similar to those of D_{FMI} and D_{JI}. We observe essentially two differences. The first is that ℓ and h have a relatively noticeable effect for *Orthogonal Modules*. The second is that the effect of interaction h:q is stronger for *Neighbor Module Swaps* and *1 New Module*. D_F still validates the same properties as D_{FMI} and D_{JI} do, despite these small differences.

3.5.2.1.5 NMI

The results obtained for the information-theoretical measure D_{NMI} are very similar to those of D_{RI}, qualitatively speaking, and to those of D_{ARI}, in terms of magnitude of the effect observed for each transformation. Like D_{RI}, D_{NMI} behaves in the same way for all the four desirable properties, and this is consistent with the observations from the literature. For instance, Meilă [162] proves that the rescaling performed by some measures for normalization purposes, such as NMI, have the effect of breaking the Convex Additivity property. Moreover, Newman et al. [175], like others [223, 186, 9], show that NMI tend to favor partitions with more modules when compared with a reference partition (cf. *no* ℓ-invariance), and that this behavior can be smoothed by correcting *NMI* for chance.

A clear difference between D_{NMI} and all the other measures is that n has a very visible effect for *Orthogonal Modules* and *Singleton Modules*. This seems to be an artefact of the normalization for these transformations, which would match the observation made by Amelio & Pizzuti [9], rather than a violation of the n-invariance property. Indeed, the information-theoretic measures are n-invariant by construction [162].

3.5.2.1.6 General Observations

For the sake of clarity, we roughly summarize in Table 3.3 the discussion that takes place throughout the current section regarding the presence or absence of desirable properties within the considered measures. We observe that three measures validate all 4 properties (D_F, D_{JI}, D_{FMI}), whereas two

measures have none of them (D_{RI}, D_{NMI}). The last one, D_{ARI}, holds an intermediary position, as it possesses the ℓ-invariance and Discriminativeness properties like D_F, D_{JI} and D_{FMI}, whereas it shares the same behavior with D_{RI} and D_{NMI} regarding Insensitivity to Module Size and Convex Additivity.

Let us now conclude this section by highlighting the main observations we could draw from the relative importance analysis. First, it is important to stress that the results produced by our framework are consistent with those published in the literature, including both theoretical and empirical works. This is summarized in Table 3.3. Second, the systematic nature of our approach helps uncovering properties not already described in the literature. For instance, Rezaei & Fränti [189] state that set matching measures are more suitable regarding the Insensitivity to Module Size property. Nevertheless, we find out that the pair-counting measures D_{JI} and D_{FMI} also possess this property. Third, our framework allows us to state that some measures possess certain properties at least partially, or under certain conditions. Indeed, our framework does not predict the presence of a property in a Boolean way, but rather on some continuous spectrum, through regression. Put differently, instead of predicting whether a measure has a property or not, we can estimate *how much* it possesses this property, and assess how this can change depending on the parameter values. For instance, as mentioned above, we can say that D_{ARI} validates the Discriminativeness property much more than D_{RI}, based on the effect of q for *Singleton Modules*.

	ℓ-invariance (t_{sm} and t_{om} with ℓ)	**Discriminativeness** (t_{sm} with q)	**Insensitivity to Module Size** (t_{onm} with ℓ, t_{ncs} with h)	**Convex Additivity** (t_{sm} and $t_{\ell nm}$ with ℓ and h)
D_{RI}	✗ [186, 9, 163]	✗ [186]	✗ [204, 189]	✗ [163]
D_{ARI}	✓ [186]	✓ [186]	✗ [204, 189]	✗ [163]
D_{FMI}	✓ [97]	✓	✓	✓
D_{JI}	✓ [97]	✓ [186]	✓	✓
D_F	✓	✓	✓ [204]	✓
D_{NMI}	✗ [223, 186, 9, 97, 175]	✗ [186, 9]	✗ [204, 189]	✗ [163]

Table 3.3 – Relations between four desirable properties and the considered measures, based on our results presented in Figure 3.4. The method used to check whether a measure has a property is summarized between parenthesis in the first line, and additional details can be found in Section 3.3.1.2. The bibliographic references show matching observations found in the literature, when available.

3.5.2.2 Typology of Measures

We now show how a typology of the measures can be built based on the results shown in Figure 3.4, through a cluster analysis. First, we compute a distance matrix comparing all pairs of stacked bars constituting the plots from this figure. For this purpose, we represent each stacked bar by a vector of proportions, each value corresponding to a term of the regression model (i.e. a segment of the stacked bar). We use the Hellinger distance [143], which was designed to compare pairs of discrete probability distributions. Second, we perform the cluster analysis by applying the k-

medoids method [128] to our distance matrix. This method requires us to specify the desired number of modules, though. To find the most appropriate number, we apply the standard approach consisting in performing the clustering using all possible values, and then selecting the most appropriate one. For this purpose, we use the Silhouette measure, a well-known internal criterion [196], but we also take into account a more subjective constraint of parsimony (i.e. we want a small number of modules).

The analysis results in 5 clusters of stacked bars, for a Silhouette of 0.55. Table 3.4 shows the distribution of the bars from Figure 3.4 over these clusters, each one being represented as a specific color. The blue cluster corresponds to bars in which there is a relatively balanced main effect of ℓ and q, and a minor effect of h and ℓ:q. In the brown cluster, the situation is quite similar but q supersedes ℓ. In the red cluster, q even more prevalent, and both minor effects are even smaller. The orange cluster contains bars in which all effects are negligible compared to q. Finally, bars from the green cluster are dominated by q and exhibit a minor effect of h:q.

	Orthogonal Modules	**ℓ New Modules**	**Neighbor Module Swaps**	**1 New Module**	**Singleton Modules**
D_{RI}					
D_{ARI}					
D_{FMI}					
D_{JI}					
D_F					
D_{NMI}					

Table 3.4 – Comparison of the measures based on the characterization provided by our framework and shown in Figure 3.4. We use the Hellinger distance and k-medoids to identify groups of similar behaviors, each one being represented by a color in the table.

Table 3.4 shows that each transformation produces a different vertical pattern, which indicates that the transformations we selected in our framework are not redundant in the way they allow characterizing the measures. The measures can be compared using the horizontal patterns present in the table. Roughly speaking, there is a first group constituted of D_{FMI}, D_{JI}, D_F; a second containing D_{RI} and D_{NMI}; and D_{ARI} is apart. We see that this characterization is consistent with the results in Table 3.3. The fact that these groups of measures, which are automatically obtained, match the ones identified manually based on our knowledge of the desired properties, indicates that this clustering-based method could be useful when the user is not able to (or does not want to) express their desired properties *a priori*. Indeed, for a given collection of available measures, this method allows identifying clusters of measures possessing a similar behavior: these clusters can then be characterized *a posteriori*, and the user can select a measure from the cluster considered as the most appropriate to the considered application.

To sum up, not only does our analysis allows distinguishing the effects of the framework pa-

rameters over transformation types and measures, but it also makes it possible to categorize the measures based on their empirical behavior. Our results confirm the findings of Pfitzner et al. [184], which indicate that the categorization of the measures based on their sole definitions (cf. Section 3.4.1) does not necessarily hold when it comes to comparing them through experiments.

3.6 Conclusion

In this chapter, we have presented a new evaluation framework to address the problem of selecting an appropriate measure to compare partitions. We want not only to compare measures, but also to produce results that the end user can easily interpret. For this purpose, based on our review of the literature, we designed a set of predefined partitions and parametric partition transformations in order to generate a benchmark dataset. Our two-step framework first computes the considered measures for these partitions, then conducts a regression and relative importance analysis to determine how the measures are affected by the transformations. We illustrated its relevance by applying it to a selection of standard measures. We showed that our framework allows identifying the desirable properties possessed by each measure. For some of them, our results confirm empirical and theoretical findings already published in the literature. For others, the systematic nature of our approach even uncovers properties not mentioned before in the literature. Finally, we propose a typology of the considered measures based on their characteristics. Overall, our results confirm the findings of Pfitzner et al. [184], which indicate that categorizing measures based on their mathematical definitions does not necessarily match experimental comparison.

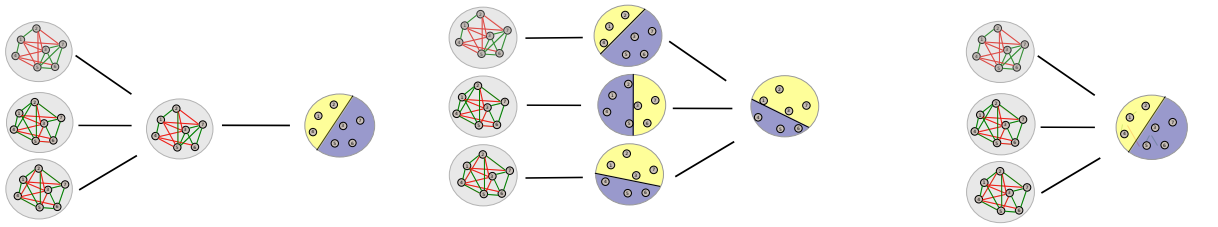
We believe that this work opens new directions for future research. First, our method can be applied systematically to other external measures, for the sake of completeness. It is particularly important to include the recently proposed measures for an up-to-date comparison, which would prevent from following the tradition of using only well-established measures without regard for their relevance. Second, similar to the previous point, some new parametric transformations can be proposed to closely investigate the performance of the measures on a specific subject. For instance, there is an important number of measures aiming at correcting Mutual Information for chance in the literature. Including some specific transformations could enable to concentrate more on the aspect related to the number of clusters. Finally, by proposing relevant parameters and transformations, our general method could be adapted to handle objects similar to partitions, such as covers, to compare overlapping clusters (e.g. [112, 97]), or edge-aware community similarity measures, to compare community structures while taking graph topology into account (e.g. [186, 138]).

Chapter 4

Multiple partitioning of multiplex signed networks

4.1 Introduction

In certain specific situations, one needs to partition the set of vertices not according to a single network, but to several ones, each containing the same vertices but possibly different edges. Each network provides a different, possibly conflicting, view of the same system under different conditions. For instance, in [41] the authors consider a set of functional connectivity brain networks, each one corresponding to one specific subject. All of them are assumed to share a general modular structure,



(a) Layer aggregation approach.

(b) Ensemble clustering approach.

(c) Multiplex approach.

Figure 4.1 – Traditional partitioning approaches for multiplex networks.

but also exhibit inter-individual variations. Such a collection of networks can be viewed as a multiplex network, in which one layer corresponds to one uniplex network.

The literature contains 3 main approaches to partition multiplex networks in general [132]. The first one, called *layer aggregation*, merges the layers to obtain a single *aggregated* network (also called flattened [210] or projected [28] network) and then apply a traditional partitioning method to it, as illustrated in Figure 4.1a. For instance, Macon et al. [157] aggregates vote results of the separate annual sessions in the United Nations General Assembly to characterize the countries' voting behaviors. Barigozzi et al. [28] combine the layers by trade category to analyze geopolitical and socio-economic patterns across trading blocks of countries. Miller [165] fuses different relationship types between terrorists operating in Indonesia and finds groups of terrorists for network disruption purposes.

The second approach, called *partition integration*, applies a traditional partitioning method separately to each layer and merges the resulting partitions to obtain a consensual one, as illustrated in Figure 4.1b. Santra et al. [200] combine the partitions of three-layered Road Accidents multiplex network by intersecting them, if they are self-preserving. Whether a partition is self-preserving or not is determined by calculating the internal clustering coefficient of the vertices. Tagarelli et al. [210] uses a cluster ensemble technique [205] for merging, when they extensively study this approach based on random multiplex networks and datasets from various domains. Berlingerio et al. [33] adopt the methodology of frequent itemset mining when they study DBLP multiplex networks, which can be seen as another form of merge process.

In both first approaches, each module of the resulting partition contains all instances of the same vertex over all layers: it can be considered as a consensual partition, fitting all layers at once. Consequently, it is more likely that these approaches result in information loss [171, 212, 132] or distortion of properties [132, 183], when the layers display different community (modular) structures. The third and last approach, called *multiplex approach* [59] or multi-graph clustering [180], is proposed to tackle this drawback: a module does not necessarily span all layers. It uses a method specifically designed for multilayer networks, which directly partitions the set of all vertices over all layers, with the aim of combining different characteristics of the layers (e.g. sparse vs. dense) in a more ap-

propriate way, as illustrated in Figure 4.1c. Didier et al. [59] use an extension of the modularity for multiplex networks, called multiplex-modularity, to identify groups of protein cells in a multiplex biological network. Tang et al. [213] propose a factorization method based on linked matrices (similar to tensor decomposition) which allows to extract only informative structural information from layers. In addition to a tensor decomposition-based approach, Papalexakis et al. [180] formulates the problem as a data compression task and aim to find the partition that minimizes the total description cost of a multiplex network.

All 3 approaches are based on the assumption that one is looking for a *single* partition. In this chapter, we relax this single-partition assumption to allow searching for *several* partitions. To this end, we propose a new partitioning process for multiplex signed networks. It can be viewed as a partition integration approach with a main difference. It integrates a *meta-clustering* process before merging the partitions of layers, and this consequently allows to cluster structurally similar layers. A consensual merging process, tailored for signed networks, is then performed for each cluster with no sensible information loss.

Our method differs from the existing ones in that it strives to cluster layers based on their partition similarity. Nevertheless, different similarity criteria can be provided to the task of *layer clustering* [124]. The existing methods in literature mainly address this task for reducing the large number of layers in a multiplex network, and not specifically for finding multiple partitions. Hence, their goal is rather to construct composite networks without a sensible information loss by quantifying either layer similarity [224] or layer redundancy [124] through network-centric approaches, which can be at either micro- (vertex-centric [116] or edge-centric [209, 224]) or macro-scale (quantum information theory [124]). In our method, we want to compare the layers partition-wise based on well-studied external evaluation measures, without passing through this network comparison step.

Contributions. The following content is based on our two works published in *iKNOW* [21] and *Social Networks* [19]. This chapter makes the following contributions:

1. **Method.** We propose a new method which aims to find multiple informative partitions describing an underlying multiplex signed network.
2. **Evaluation.** We evaluate our method on European Parliament vote dataset from political analysis, and show its usefulness by extensively interpreting the obtained results.

The rest of this chapter is organized as follows. First, in Section 4.2, we introduce the fundamental concepts and the problem definition. We then turn to the methods in Section 4.3, and describe the approach we propose for the analysis of multiplex signed networks. In Section 4.4, we put the proposed method into practice through an analysis of voting data, and discuss its results in Section 4.5. Finally, we review our main findings in Section 4.6, comment the limitations of our work and describe how they can be overcome, and how our work can be extended.

4.2 Problem definition

As explained in Section 4.1, the methods tackling traditional approaches output a single partition for multiplex networks. In this work, we relax this constraint by allowing to produce multiple partitions. To this end, we define this task as a clustering problem over a set of partitions, as in a meta-clustering problem [205, 190]. The traditional partition integration approaches can be viewed as fitting a particular case of our more general problem, in which only one partition is output. In the following, we first introduce the fundamental concepts that we use throughout the chapter, followed by the definition of the problem itself. We start by defining a multiplex signed network.

Definition 4.1 (Multiplex signed network)**.** *A* ***multiplex signed network*** $\mathcal{G} = (G_1, G_2, .., G_p)$ *is a network constituted of* p *layers, where each layer is itself a signed network* $G_i = (V, E_i, s_i)$ *with* $i \in \{1, \ldots, p\}$*.*

Each layer in $\mathcal{G}$ contains the same set of vertices V, nevertheless, the edges can be different from one layer to the other. This type of graph should not be confused with the more general multilayer networks, which allow inter-layer edges and different vertices in each layer. It is worth noting that it is more valuable to use an automated method such as ours if the number of layers is not small. Therefore, we assume there are many layers in $\mathcal{G}$.

Since we will handle various types of partitions, we need to distinguish them in our terminology.

Definition 4.2 (Pattern)**.** *Given a multiplex signed network* $\mathcal{G}$*, a* ***pattern*** $P_i = \{M_1, M_2, ..., M_{\ell_i}\}$ *is a partition of the vertex set* V *of a single layer* G_i*, where* ℓ_i *denotes the number of modules in* P_i*.*

Concretely, if there are p layers in $\mathcal{G}$, then there will be p patterns. It is important to stress that any partitioning criterion or method related to signed graph partitioning, such as the one reviewed in Chapter 2, can be used for finding patterns.

To gather similar patterns together during cluster analysis, we need to define how dissimilar patterns are.

Definition 4.3 (Dissimilarity between patterns)**.** *The* ***dissimilarity*** D *between a pair of patterns* P_i *and* P_j *is score* $D_m(i, j)$ *computed by any external evaluation measure* m*.*

Score $D_m(i, j)$ indicates how dissimilar two patterns are by considering only the corresponding partitions and not the graph structures.

Since the cluster analysis is applied over a set of patterns, we need to define the result of this analysis as another type of partition:

Definition 4.4 (Cluster and clustering)**.** *The term* ***clustering****, denoted by* C_k*, refers to a partition of the set of all* p *patterns into* k *subsets, such that the dissimilarity between patterns being in the same cluster is minimized and that of patterns is maximized. The subsets constituting a clustering* C_k *are called clusters, and the* i^{th} *cluster, with* $i \in \{1, ..., k\}$*, is denoted by* C_k^i*.*

We assume that there are inherent similarities between the patterns of the same cluster, so that it is meaningful to characterize them. This requires us to define a specific type of pattern:

Definition 4.5 (Characteristic pattern of a cluster)**.** *The* ***characteristic pattern*** *of cluster C_k^i, with $i \in \{1, .., k\}$, denoted by $\widehat{P}_k^i$, is the most consensual partition that represents the patterns of this cluster, with respect to a* ***consensus function****.*

The consensus function can be one of several procedures, as illustrated in [219]. We describe one later in Section 4.3.4.

Now, we introduce formally the problem we tackle in this chapter.

Problem 4.1 (Multiple partitioning of a multiplex signed network)**.** *Let $\mathcal{G}$ be a multiplex signed network with p layers, and P_i with $i \in \{1, .., k\}$ the patterns detected in these layers. The* ***Multiple partitioning of a multiplex signed network*** *problem consists in finding a clustering of these patterns such that each cluster C_k^i with $i \in \{1, .., k\}$ produces a characteristic pattern C_k^i with respect to a consensus function.*

Problem 4.1 is generic enough so that it could be also defined for unsigned graphs. The problem can be interpreted in various ways depending on the application. In the next section, we will briefly discuss the steps of our proposed method by making some methodological choices for a general use.

4.3 Our method

In this section, we describe our four-step method to solve Problem 4.1, summarized in Figure 4.2.The first step is to separately partition each of the p layers of the multiplex signed network $\mathcal{G}$ through a standard signed graph partitioning method, in order to get as many patterns (Section 4.3.1). The second step consists in computing the dissimilarity between the patterns (Section 4.3.2), in order to perform the third step, which is a cluster analysis (Section 4.3.3). This leads to a set of k clusters, each one gathering similar patterns. The fourth step is to process the characteristic pattern of each cluster, which is supposed to consensually represent all the patterns constituting the cluster (Section 4.3.4).

4.3.1 Processing the Patterns

This step can be done by any signed graph partitioning method. As stated in Section 1, there are a number of partitioning problems and resolution methods for signed graphs. Among them, application-wise the CC problem has received the most consideration in the literature (see Section 2.1 and references therein). It is used in a number of situations to get a better understanding of

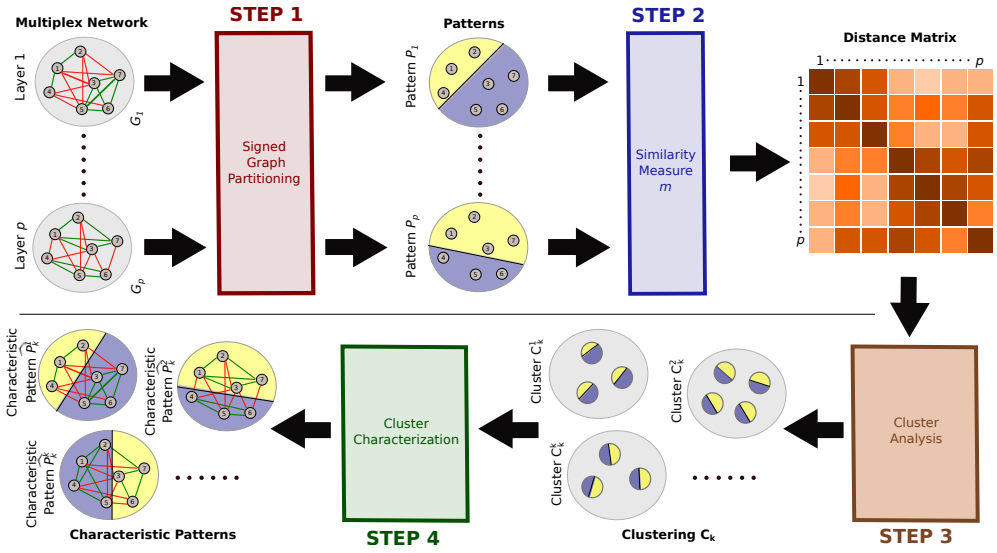


Figure 4.2 – General workflow of the proposed analysis method.

the studied real-world system. Therefore, we propose to solve the CC problem in our method when detecting the pattern P_i associated with the i^{th} signed network G_i of $\mathcal{G}$.

In order to identify the patterns, we use an exact method for medium-sized networks (e.g. for $n \leq 50$), because only exact methods guarantee to output the best solution in terms of imbalance. Particularly, as shown in Section 2.5.1, it is more appropriate to use B&C($F_v^\star(G)$) for graph density $d \geq 0.5$ and when graph order n increases, and switch to B&C($F_e(G)$) below this threshold. In case of large signed networks, as shown in Section 2.5.2, it is preferable to take into consideration a collection of heuristics methods, such as those presented in Section 2.3.

4.3.2 Computing the Dissimilarity Values

At this stage, we have identified the pattern P_i associated with each G_i. We now want to gather similar patterns together in a classic cluster analysis approach.

This clustering approach requires to process the dissimilarity matrix by comparing each pair of patterns. A number of measures have been defined to compare such patterns, each one possessing a specific behavior. These measures are sensitive to various characteristics of the compared patterns, such as the sizes or numbers of the clusters, as shown in Chapter 4. Moreover, the choice of an appropriate measure depends on a number of factors, including the broad situation, but also the nature of the application at hand and other contextual aspects such as the behavior expected by the user. We select the measure by expressing our expectations regarding the application at hand, and following the methodology proposed in Section 3.3. The selected measure m can be either a metric on the space of patterns or a similarity-based external evaluation measure, as defined in

Chapter 3.4.1. Based on m, we then build a dissimilarity matrix summarizing these comparisons.

4.3.3 Performing the Clustering

Next, we apply the k-medoids clustering [128] method to the previously extracted dissimilarity matrix to obtain a clustering C_k. It is similar to the well-known k-means algorithm in the sense that it tries to partition the dataset into k clusters, while minimizing the dissimilarity between the members of each cluster and some center of the cluster. The difference is that in k-means, this center is an average value, whereas in k-medoids it is one of the actual data points from the dataset. It is generally used in place of k-means when one cannot perform the required average operation, which is true in our case (we cannot straightforwardly process an average pattern).

This method requires us to specify k, but we do not know it in advance. In this situation, the standard approach is to use all possible values of k, from 2 to p, and assess the quality of the $p-1$ resulting clusterings through some internal criterion. The most widespread such measure is the *Silhouette* S, which characterizes the clustering in terms of internal cohesion and external separation of the clusters [196]. It takes a value between -1 and $+1$, where the latter represent the best possible clustering.

In theory, the value of k associated with the highest Silhouette $S(C_k)$ is the best candidate. However, in practice, one possibly has to consider other factors to make a choice. For example, marginal improvements of the Silhouette are sometimes caused by the creation of singleton clusters, which generally do not bring much relevant information in terms of interpreting the clustering. It is therefore necessary to study qualitatively how the clusters evolve with k to make an informed choice. This choice can be also guided by some constraints arising from the application context itself.

4.3.4 Computing the Characteristic Patterns

We now have k clusters, each C_k^i with $i \in \{1, .., k\}$ containing a certain number of patterns. The patterns constituting a cluster may differ slightly, but overall they are supposed to be very similar. The next step is to compute a characteristic pattern $\widehat{P_k^i}$, with $i \in \{1, .., k\}$, representing the whole cluster, such that these small differences are smoothed.

For this purpose, we use as a consensus function a similarity network-based approach, inspired by the work of Lancichinetti and Fortunato [140]. Based on a collection of p partitions of the same set, they derive a consensual partition by first extracting a weighted similarity network, and then performing community detection in this network. The resulting communities correspond to consensual clusters, and the community structure is the consensual partition. Their network is built as follow: each vertex represents an element of the partitioned set, and the edge weight is the proportion of partitions in which both connected vertices belong to the same module. We experimented with this approach, and found we obtained better results by using the following *signed* version. The weights

are now the difference between the proportion of patterns putting both vertices in the same module, and the proportion of patterns putting them in different ones. Finally, instead of a community detection algorithm, we solve the CC problem by applying the same signed graph partitioning method used in Section 4.3.1 so as to identify a partition corresponding to the characteristic pattern of the cluster.

4.4 Experiments

We apply our method to a dataset representing the voting activity during the 7$^{\text{th}}$ term (2009-2014) of the European Parliament (EP) in order to study the voting activity of the Members of the EP (MEPs). Our goal is not only to detect groups of MEPs which would be cohesive in terms of votes, but also to identify the different typical *voting behavior patterns* of the EP, i.e. the characteristic ways in which the MEP set is partitioned by these votes, as well as the legislative propositions to which they apply.

We want to illustrate how our method differs from those of the traditional approaches (see Section 4.1). Among them, we opt for a standard *layer aggregation* approach as a reference, which is prevalent in the literature, particularly in the context of vote analysis (e.g., [157, 227, 89, 148]). The standard way to deal with a long period of voting activity, such as the considered period 2009-2014, is to split it into several distinct periods, each one leading to a separate (aggregated) network. These networks are signed and weighted: a *positive* (resp. *negative*) numerical value associated with an edge represents the level of similarity (resp. dissimilarity) between the vertices (MEPs) it connects. Such a weight is processed thanks to a *similarity function*, whose role is to estimate the level of agreement or disagreement between two MEPs of interest, based on the votes they cast during a selection of roll-calls. This is an important methodological choice, because the resulting network is obtained by *temporal integration* of the raw data.

In this section, we first present the dataset and highlight its most relevant characteristics (Section 4.4.1). We then summarize how the signed network constituting the layers of our multiplex signed network are extracted from the raw voting data (Section 4.4.2). Finally, we explain how we select an appropriate evaluation measure for the application of EP, when such measure is needed in our method for calculating the dissimilarities between patterns (Section 4.4.3).

4.4.1 IYP Dataset

The raw data are publicly available on the official website of the EP [1], but they are technically difficult to collect. Some citizen oversight websites did the work of retrieving them, and publishing them online. We use such a dataset, provided by the website *It's Your Parliament* [2] (IYP).

1. http://www.europarl.europa.eu/
2. http://www.itsyourparliament.eu/

These data describe the activity of the MEPs during the 7th term of the EP (2009–14). They are constituted of the votes cast by all MEPs for all roll-calls taking place during a plenary at the EP. Note that the default voting procedure at the EP is the show of hands, during which individual votes are not recorded, and that roll-calls happen only under certain circumstances. During the 7th term, these noticeably include the final vote of any legislative proposition, or a written demand by a group of 40 MEPs or a parliamentary group [82]. These data are therefore incomplete by definition, but recent studies have shown that they are nevertheless representative of the overall voting behavior [123]. Several works have studied them under the form of networks [217, 164, 199, 21].

The vote of MEPs can take 3 distinct values: FOR (they support the proposition), AGAINST (they oppose the proposition) and ABSTENTION (they do not want to take a stand). It is also possible for MEPs not to vote at all. Officially, the EP distinguishes various types of absences or reasons for not voting (see [164]). But in this dataset, all are simply represented by the value ABSENT. Each text is itself associated with one among 21 specific policy domains (see [164] for the complete list).

A number of personal details are available for each MEP: *Name* (the full name of the MEP), *Country* (one of the 28 member states in which the MEP was elected), *Party* (the national political party to which the MEP belongs, in his own country), and *Group* (the parliamentary group to which the MEP belongs in the EP).

The EP groups are important when interpreting a voting pattern of the MEP set, because they correspond to the political position that MEPs are supposed to hold, at least theoretically. During the 7th term, there were seven groups, described in Table 4.1. Note that *NI* is a technical group containing MEPs not belonging to any of the other groups. For the 7th term, NI members were mainly far-right MEPs. One would expect that patterns automatically estimated based on the voting data would fit this division, but as we will see in Section 4.5, this is not necessarily the case.

Table 4.1 – Groups of the EP during the 7th term, in decreasing order of size.

Group name	**Acronym**	**Description**
European People's Party	EPP	Right/center-right conservatives
Progressive Alliance of Socialists and Democrats	S&D	Center-left
Alliance of Liberals and Democrats for Europe	ALDE	Right/center-right neoliberals
Greens–European Free Alliance	G-EFA	Left environmentalists, progressives and regionalists
European Conservatives and Reformists	ECR	Right euroskeptics and anti-federalists
European United Left–Nordic Green Left	GUE-NGL	Left/far-left, socialists and communists
Non-Inscrits	NI	Far-right

Instead of working on the whole dataset, we focus on some subsets in order to perform a more thorough and qualitative interpretation of the results. In particular, we restrict the analyses to the French MEPs, to propositions related to the *AGRI* (agriculture) policy domain, and considered sep-

arately the years constituting the term. This domain was selected due to its potentially polarizing nature: the *Common Agricultural Policy* (CAP) has historically been of great importance for Europe (38% of the total EU budget [81]), especially for France [73] and Italy [74], due to the importance of agriculture in their economies, because a part of the population wants to leave the industrial model of production. We provide more context in Appendix B.1 by defining the most important UE- and CAP-related concepts. In addition, the 2012-13 legislative year marks the start of a major reform of the CAP, as explained in further detail in Appendix B.2. These reforms cover various aspects of agriculture (e.g. production quotas, environmental aspects, market regulation), which allows analyzing the results from multiple perspectives.

At some point in our processing, we will need to characterize subgroups of legislative propositions in a topical way, in order to assess their thematic homogeneity (or lack thereof). Since all the propositions we examine are related to agriculture, we have to consider subdomains. For this purpose, we use the typology proposed by EUR-Lex, the EU website for the publication of official documents such as treaties and legislation. The detailed hierarchy of domains is described in Appendix B.3.

In order to interpret the groups of similarly voting MEPs identified by our method, we will also need some real-world reference regarding the positioning of the EP groups on agricultural questions. To this aim, we have manually reviewed the official material published by the EP groups on these topics for the 2012-13 period: manifestos, positioning papers, and revised written transcriptions of speeches made at the EP. Table 4.2 summarizes the information that we collected. Each row corresponds to one of the main agricultural topics, and each column to an EP group. Empty cells reflect either the fact that the concerned group did not take any position relatively to the considered topic, or that no source could be found to describe this position. It appears that agreement between groups covers all the spectrum, ranging from complete disagreement (e.g. GUE-NGL and G-EFA) to almost perfect agreement (e.g. GUE-NGL and G-EFA). The question is now to check whether these theoretical positions translate into actual votes.

Table 4.2 – Position of the EP groups on AGRI-related topics for the considered period of 2012-13.

Subject	GUE-NGL	G-EFA	S&D	ALDE	EPP	EFD	ECR	NI
1) Reduction of direct payments	FOR reduction (ceiling at 100 k€) [84]	FOR reduction (starting from 50 k€) [215]	FOR reduction (starting from 150 k€) [83]	-	AGAINST reduction [83]	AGAINST reduction [83]	AGAINST reduction [83]	-
2) Maintaining milk quotas	FOR quotas (with flexibility) [83]	FOR quotas (food security purposes) [83]	FOR quotas [83]	AGAINST quotas (competitiveness purposes); FOR transition period [6]	AGAINST quotas (competitiveness purposes) [83]	-	AGAINST quotas (low food price purposes) [83]	FOR quotas (fair price purposes) [144, 99]
3) Export subsidies	-	AGAINST subsidies [215, 214]	AGAINST subsidies [83]	AGAINST subsidies (with transition period) [6]	AGAINST subsidies; FOR exceptional subsidies [83]	-	-	-
4) Competitiveness	AGAINST current system (too competitive) [83]	AGAINST current system (too competitive) [83]	-	FOR competitiveness [6]	FOR competitiveness (better functioning of supply chain purposes) [83, 106]	-	FOR competitiveness (low food price purposes) [83]	AGAINST current system (fair price purposes) [144, 99]
5) Aid for rural development	-	AGAINST current scheme (not enough); FOR co-financing [83]	FOR transfer from Pillar I to Pillar II; AGAINST co-financing [105]	FOR transfer from Pillar I to Pillar II, co-financing [83]	FOR transfer from Pillar I to Pillar II, producers as market actors [83]	-	AGAINST risk management measures (not enough) [83]	AGAINST the current scheme (not enough) [144]
6) Food quality vs. quantity	Favors QUALITY [84]	Favors QUALITY [214, 83]	-	Favors QUANTITY [6]	Both QUANTITY and QUALITY [106]	Favors QUALITY [221]	Both QUANTITY and QUALITY [83]	Favors QUALITY [99]
7) Enhancing environmental measures	FOR organic farming; AGAINST GMO	FOR crop rotation and diversification, promoting biodiversity, organic farming, permanent grasslands, AGAINST GMO, intensive agriculture [214, 215]	FOR reducing use of chemicals, promoting biodiversity, energy savings; AGAINST intensive farming [105]	FOR greener CAP, energy savings, tackling climate change through innovative solutions [6]	-	AGAINST GMO [221]	Not the priority [83]	-

4.4.2 Network Extraction

The voting behavior of each MEP is represented by a series of vote values, each one corresponding to a specific roll-call of the considered period. A natural approach consists in modeling the data as a dynamic network, in which each time slice corresponds to a roll-call. However, this is not appropriate for the EP, due to its very specific scheduling, which differs greatly from a typical national parliament's. At the EP, there are only 4 days of plenary session by month, in average, during which the MEPs consider a large number of propositions. The chronology of the propositions therefore has little effect on the votes, as we verified empirically. This is why a multiplex representation, in which each layer models one roll-call, is more appropriate here.

The similarity function that we use is very basic, since it is applied to each roll-call considered separately (by opposition to the whole series). For a pair of MEPs u and v and a roll-call i, we have $(u, v) \in E_i$ if and only if both MEPs voted, i.e. neither were absent. Moreover, we set $s_i(u, v) = +$ if both MEPs voted similarly (FOR-FOR, AGAINST-AGAINST, or ABSTAIN-ABSTAIN) and $-$ otherwise. Unlike other approaches such as in Section 4.5.1, we do not need to filter the resulting edges, because their weights are not the result of some averaging, and all of them are assumed to be informative. It is difficult to decide how to treat abstention, since it can be considered as the expression of some intermediate position. In this extraction phase, we consider ABSTAIN exactly like the other forms of vote (FOR and AGAINST).

4.4.3 Measure Selection for Calculating Dissimilarities Between Patterns

We now use our results from Section 3.5.2, and in particular from Figure 3.4, to solve the measure selection problem presented in Section 4.3.2. The application of European Parliament brings specific constraints on any pattern P_i of $\mathcal{G}$, which can contain at most three modules, i.e. $1 \leq \ell_i \leq 3$: 1) a single module in case of unanimity (all MEPs vote either *For*, *Against* or *Abstain* the legislative text); 2) two modules when there is either an antagonistic situation (i.e. some MEPs support the concerned document and the rest oppose it), or a unanimous module with an additional module of abstentionists; 3) two antagonistic modules with an additional module of abstentionists. It is important to stress that some parameters and transformations presented in Chapter 4 are not relevant here, due to the application context. In the following, our discussion is based on the concepts and the notations introduced in Chapter 4, for the sake of consistency.

First, the single module of unanimity is not applicable for some transformations presented in Section 3.3.1.2. Therefore, we apply all the transformations if there is more than one module in the original partition. For *k New Modules* and *Singleton Modules*, this means that we get at least four modules in the transformed partition. This is incompatible with the fact that all the compared partitions of this application contain at most three modules, so we exclude both transformations. Second, in this context, the *Orthogonal Modules* transformation can be applied only when there

are two modules in the original partition, and only one vertex in each module is affected by the transformation. In this case, this transformation results in the same transformed partition as with *1 New Module*, therefore we also exclude *Orthogonal Modules*.

This leaves us with two transformations. The first is *1 New Module*, which we apply only when the original partition has two modules, in which case the transformation produces an additional module in the transformed partition. The second is *Neighbor Module Swaps*, which we apply only when the original partition has either two or three modules, but not a single one. For both these transformations, there is no constraint on parameters h and q. However, as explained above, our analysis must focus only on certain values of ℓ, due to $1 \leq \ell_i \leq 3$ in P_i. Finally, in the context of EP all the considered G_i contains the same number of vertices, which means that n is fixed in the framework and can therefore be ignored in our discussion.

Each application has its own desirable properties that an appropriate measure should satisfy regarding the application needs. In this application, we want the selected measure to be sensitive enough so that it detects relevant module changes, but not enough so that it is affected by simple individual exchanges of MEPs. Next, we express this desired behavior with respect to the remaining parameters and transformations. The measure must be sensitive to the *1 New Module* transformation as, in this context, detecting an extra module or missing one is an important error, since there are only a few possible module of MEPs. When ℓ increases, so does the diversity of the module created by this transformation, in the sense that its vertices come from more distinct original modules. In this context, this is an important difference with the original partition, so we want the score of the measure to increase with ℓ. By comparison, it is desirable that the dissimilarity score decreases when h increases, as this means most vertices of the extra module come from the same original module, an error which is less serious. For the same reason, the effect of ℓ should be stronger than that of h.Transformation *Neighbor Module Swaps* consisting in mixing the original modules to get the transformed partition without changing the number of modules makes the modules more different, when q increases. In this application context, it is important that this type of difference between partitions is taken into account, so the measure must be sensitive to it. Changes in ℓ and h do not affect the mixing much, so the measure score is expected to be largely independent from these parameters.

Let us now study which measures studied in Section 3.4.1 fit the constraints described above. Regarding transformation *1 New Module*, it appears that only D_{FMI}, D_{JI} and D_F behave appropriately. When considering transformation *Neighbor Module Swaps*, we can see that, even if it is a small one, ℓ has an effect on D_{FMI} and D_{JI}. In conclusion, based on these observations, we select D_F in this context.

4.5 Results

As explained previously, we want to illustrate how our method differs from those of the traditional approaches. We do so by comparing our method with a classic layer aggregation approach, which relies on the temporal integration of the raw vote data. For this purpose, as explained in Section 4.4.1, we focus on the votes of French MEPs related to agricultural questions during the 2012-13 legislative year. Our source code is publicly available [3].

We represent all networks and voting patterns using a circular layout (Figures 4.3) generated through Circos [4]. We describe them generically here, for matters of convenience. They shall be read from the center to the periphery. The negative and positive edges are drawn at the center, in red and green, respectively. Next, the inner colored ring represents the vertices (MEPs), and these colors correspond to the modules constituting the detected pattern. If a MEP was often absent, he is ignored (as explained in Section 4.3.4) and appears in white. The names of the MEPs are indicated separately in B.4 (Figure B.2). Finally, the outer ring shows the European political groups to which the MEPs belong. They are ordered according to the political spectrum, from left to right: GUE-NGL (red), G-EFA (green), S&D (pink), ALDE (orange), EPP (light blue), ECR (dark blue), EFD (purple) and NI (brown) (see Section 4.4.1 for their full names and descriptions).

The rest of this section is organized as follows. We start by presenting the results of a classic layer aggregation approach (Section 4.5.1), then turn to our own clustering results (Section 4.5.2), and finally discuss the obtained characteristic patterns and compare them with those of the traditional approach (Section 4.5.3).

4.5.1 Baseline

As a classic layer aggregation method, a.k.a traditional approach, we have extracted a layer aggregated network by integrating the votes of French MEPs related to agricultural questions over the whole considered legislative year (2012-13). Moreover, the weakest edges are filtered to sparsify the network, which eases both their processing and the interpretation of the results (see [21] for the filtering details). Figure 4.3a and Figure 4.3d show the best pattern obtained by solving the CC and RCC problems, respectively. When solving the RCC problem, we fix ℓ to the optimal number of modules for CC.

The pattern for the CC problem, which is the optimal solution to CC, contains 3 modules. There are two large modules of similar size: the left one is largely dominated by the environmentalists (G-EFL) and also contains the radical left (GUE-NGL) and NI ; and the right module contains the center-left (S&D), right and center-right (EPP and ALDE) groups. Both modules have a majority of positive internal and negative external edges. The third module (at the bottom) is a single vertex

3. https://github.com/CompNet/MultiNetVotes.
4. http://circos.ca

corresponding to *Philippe de Villiers*, the only French member of the right-wing euroskeptic EFD group. Unlike the three members of the other euroskeptic group (NI), he is connected to the rest of the graph only by negative edges. This pattern displays a clear left/right divide, with the exception of the three NI members, who are put together with the left/environmentalists. This divide can be explained when considering the texts voted this year, among which a noticeable proportion (further details are later given in Figure 4.4) concerns environmental issues and animal rights (questions of great importance for these groups). The fact one ALDE member was put with the Greens supports this, since it corresponds to *Corrine Lepage*, former Minister of the Environment in a right-wing French government. One could expect the S&D to vote similarly to the rest of the left on these topics. However, even more texts voted during this year concern the CAP, on which the center-left is more likely to side with the right. This also explains the position of NI, which is more likely to opportunistically support resolutions in favor of small family-owned farms (and therefore vote like the radical left).

The other pattern, illustrated in Figure 4.3d, is the solution of the RCC problem with $\ell = 3$. This pattern differs from the previous one in that it identifies 3 modules of comparable sizes (no more singleton module). Both large clusters from the previous pattern loose a number of members, which are gathered to form a new, intermediary group. It contains some of the radical left (GUE-NGL), the center-left (S&D), center-right (ALDE) as well as the NI group. The two other modules are the environmentalists (G-EFL with Lepage) and the rest of the right (EPP with de Villiers), respectively. This pattern is interesting, because it manages to identify a module of moderate MEPs, which sometimes vote like the environmentalists, and sometimes like the right. This type of structure could be identified due to the relaxed nature of RCC, which allows here to have positive edges *between* modules.

To conclude this part, the position of S&D and ALDE is interesting, because they belong to the right-wing module according to CC, whereas they hold an intermediate position with RCC. Nevertheless, a classic layer aggregation approach is not able to give a solid explanation for this, and to discover in which context this happens. But we assume that these groups are sometimes voting like the left-wing module, and sometimes like the right-wing one. We take this discussion as a reference when commenting the results of our method, highlighting both differences and similarities between these approaches.

4.5.2 Clustering

Figure 4.4 displays the results obtained after the third step of our method (k-medoids clustering, Section 4.3.3), when applied to the same raw data. The bottom right plot shows the Silhouette score $S(C_k)$ as a function of the k value used when performing the clustering. The highest Silhouette scores are obtained for the smallest k values. Note, in particular, the very large gap between $k = 1, ..., 8$ and $k \geq 9$. This confirms our assumption from Section 4.3.3 regarding the expected number

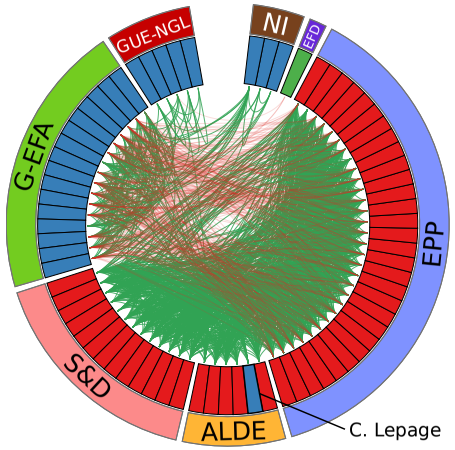
of clusters, and justifies that we discard the clusterings C_k obtained for $k \geq 8$ in our analysis.

The left plot of the same figure is an alluvial diagram [5] representing the changes underwent by the clustering depending on k. Each vertical bar corresponds to the clustering obtained for a given value of k, ranging from 2 to 8. Its vertical rectangles represent the constituting clusters. The value indicated below the bar is the corresponding Silhouette score. Each horizontal line represents a document, and its color depends on the document subdomain (cf. B.3). If a document has several subdomains, it is duplicated as many times (but here, this concerns only 3 documents).

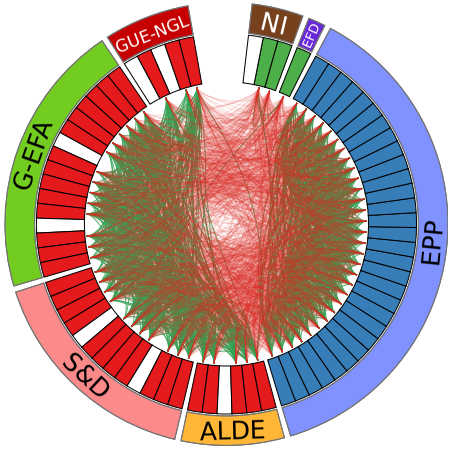
Let us first discuss the topical heterogeneity of the clusters. One could *a priori* assume that the opinions of MEPs do not change much when considering documents related to the same subdomain, and therefore expect the clusters to be somewhat homogeneous regarding this aspect, or at least to see all documents related to a given subdomain gathered in the same cluster. However, the alluvial diagram shows that this is not the case at all: each cluster contains several subdomains, and certain subdomains appear in several clusters. We can think of several reasons for this. First, the subdomains do not completely encompass the characteristics of the documents, and there are other factors to take into account. We would need to detect the sentiment (in the Natural Language Processing sense, i.e. positive or negative opinion) conveyed by the document's textual content to explore further this issue. Second, the dataset contains all available amendments: an amendment is generally a small modification of an existing document and can therefore lead to an easy consensus, which can introduce a bias toward this type of pattern. Third, the distribution of documents over the different subdomains is very unbalanced, and some rare subdomains are actually completely included in a single cluster, whereas other ones are widespread. For instance, themes mostly related to the economic aspects of agriculture (such as CAP mechanisms, social and structural measures, and multiple market organizations – see Table B.1 for details) appear in every cluster. Of course, the clusters mechanically become purer when k takes much larger values, as their sizes decrease.

Another important property of our clusterings is that, even though we do not use a hierarchical clustering method, they exhibit a quasi-hierarchical organization. More precisely, all clusters obtained for a given k are kept for $k+1$, except one which is split in two to get an additional cluster. For instance, when considering $k = 2$ and $k = 3$: C_2^1 is kept as C_3^1, whereas C_2^2 is split into C_3^2 and C_3^3. Interestingly, among the two clusters resulting from such a split, one always exhibits a characteristic pattern identical or very similar to that of its ancestor (e.g. C_3^3), whereas the other's is different (e.g. C_3^2). For $k > 5$ though, the characteristic patterns obtained for the new clusters are not different enough to present any interest, in terms of interpretation. In the end, if the highest Silhouette value is obtained for $k = 3$, the second best is $k = 5$, and it additionally leads to a larger number of sufficiently different characteristic patterns. As mentioned in Section 4.3.3, when identifying the best k, it is important to consider qualitative aspects in addition to the Silhouette. Under these terms, we identify $k = 5$ as our best trade-off, and discuss it in the rest of this subsection.

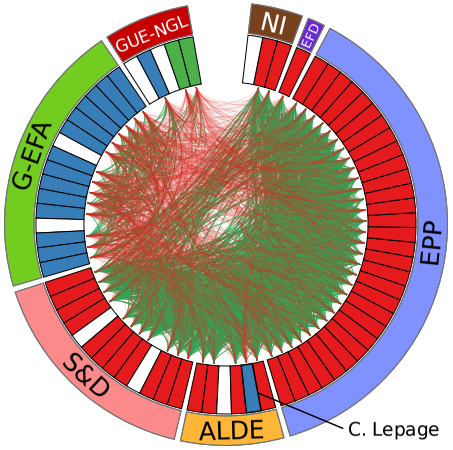
5. https://cran.r-project.org/web/packages/alluvial/



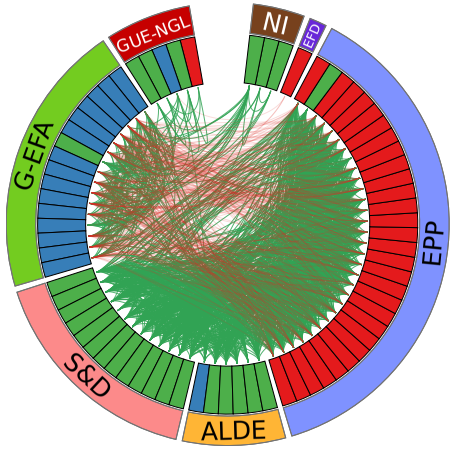
(a) CC (a classical layer aggregation approach used in Section 4.5.1).



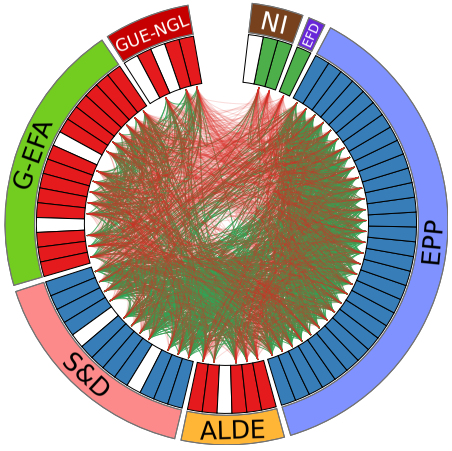
(b) $\widehat{P_5^2}$.



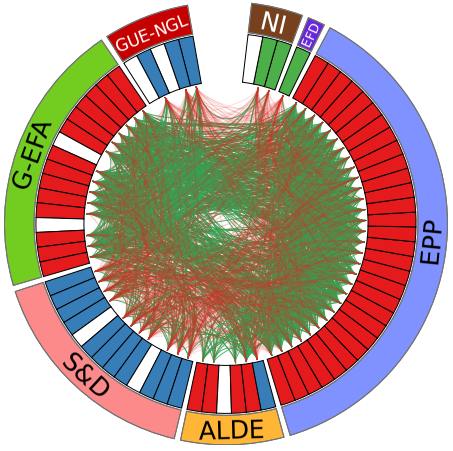
(c) $\widehat{P_5^3}$.



(d) RCC (a classical layer aggregation approach used in Section 4.5.1).



(e) $\widehat{P_5^4}$.



(f) $\widehat{P_5^5}$.

Figure 4.3 – Characteristic patterns of the French MEPs on AGRI questions in 2012-13. Red and green lines at the center represent negative and positive edges, respectively (red edges are drawn on top of green ones in order to improve readability). Around the edges, each MEP is represented by a colored tile, whose color corresponds to the MEP's faction in the displayed pattern. The green factions in plots (b), (c), (e) and (f) correspond to abstentionists. The MEPs' names are indicated separately in Figure B.2. The outer ring represents the political groups at the EP. The left plots show the patterns obtained by the classical layer aggregation approach used in Section 4.5.1 on the same integrated network when solving (a) CC and (d) RCC. The right plots show the second to fifth clusters obtained with our proposed method for $k = 5$: (b) $\widehat{P_5^2}$ (%15 of roll-calls), (c) $\widehat{P_5^3}$ (%32 of roll-calls), (e) $\widehat{P_5^4}$ (%8 of roll-calls), and (f) $\widehat{P_5^5}$ (%3 of roll-calls). The first cluster, $\widehat{P_5^1}$, which corresponds to a unanimity situation, is represented separately in Figure B.1.

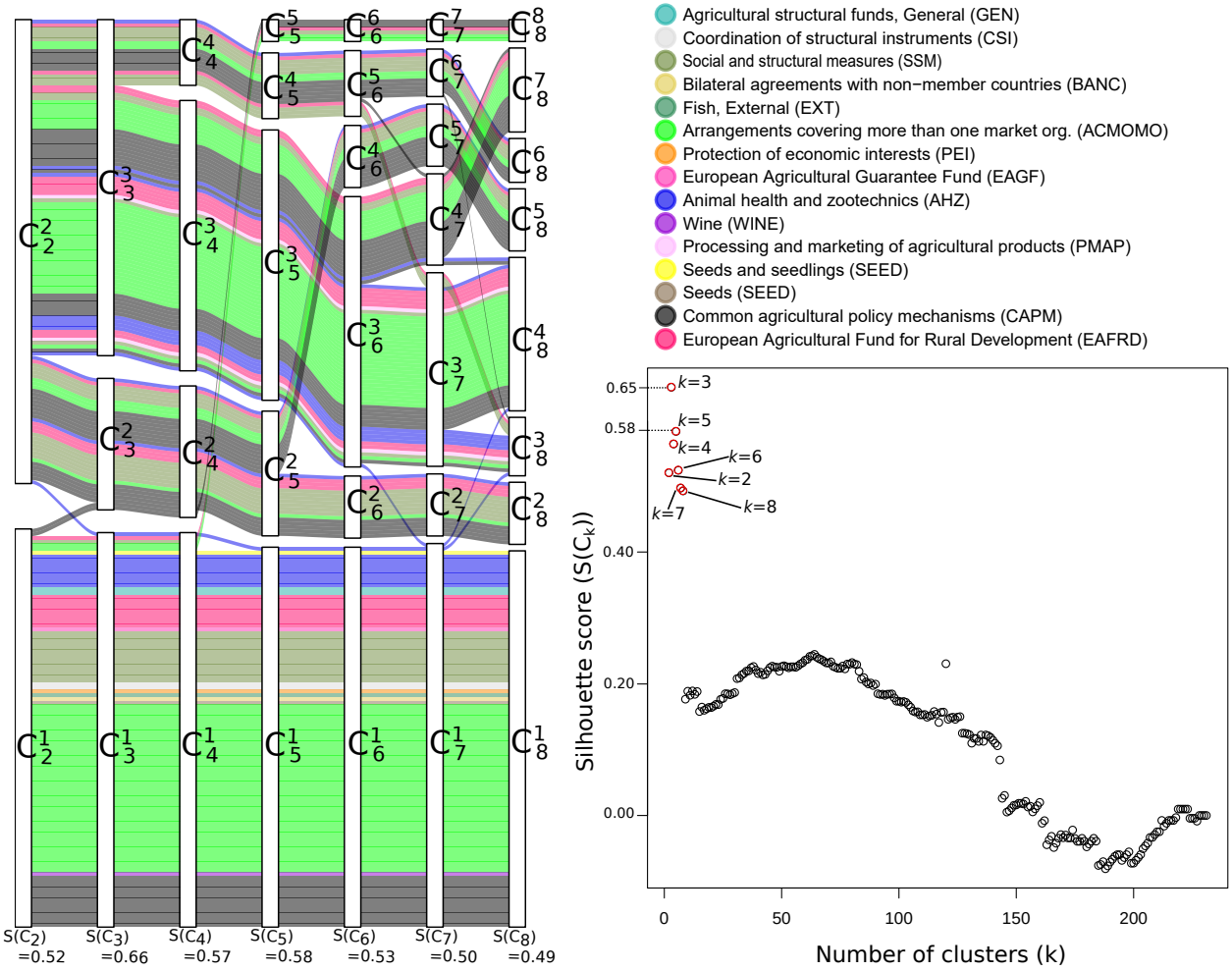


Figure 4.4 – Alluvial diagram (left) with its legend (top right), and Silhouette scores (bottom right) for the French MEPs.

4.5.3 Characteristic Patterns

In the following, we discuss each cluster obtained for $k = 5$ and its corresponding characteristic pattern.

4.5.3.1 Unanimity

For space considerations, the characteristic pattern $\widehat{P_5^1}$ associated with cluster C_5^1 is represented separately in Figure B.1, as it corresponds to a unanimity situation. Indeed, it contains a single module, and only one negative edge, between P. Le Hyaric (GUE-NGL) and M. Le Pen (NI). The emergence of such a high level of agreement was completely hidden when considering only the layer aggregated network, and therefore could not be detected in Section 4.5.1. C_5^1 is the largest cluster

with $100/232$ roll-calls (43%), so we can assume that $\widehat{P_5^1}$ represents the regular voting behavior in the considered context. All the other clusters correspond to characteristic patterns containing varying antagonistic modules. This is consistent with the fact that our clusters are supposed to correspond, by construction, to distinct voting patterns.

Although the unanimity case could be considered of less importance in terms of characteristic pattern, it is worth illustrating its occurrence in this context. One such roll-call is related to the improvement of applications for the protection of a designation of origin or a geographical indication (e.g. wine). Specifically, this text is about determining more explicitly the eligibility requirements to make such an application, and giving the concerned member state the responsibility to verify it.

4.5.3.2 Conservatives vs. the Rest

The characteristic pattern $\widehat{P_5^2}$ associated with C_5^2, shown in Figure 4.3b, finds the right-wing conservative group (EPP) opposing the rest of the MEPs, while both Euroskeptic groups (EFD and NI) abstain. This cluster contains $34/232$ (15%) roll-calls. An examination of the content of the corresponding legislative documents, as well as of certain positioning documents produced by the EP groups, such as election manifestos and public letters, reveals that this voting behavior corresponds to EPP trying to block radical changes related to the CAP. These changes, as well as their blocking by the right-wing conservatives, are confirmed in a positioning paper published by S&D about the 2013 CAP reform [105].

Among them, one of the most important was the capping of *direct payments* to farmers. Direct payments constitute a form of basic income conditioned on the implementation of certain EU rules. They represent a consequent budgetary item: 72% of the EU farm budget [75]. As such, they are difficult to reform, but the EP was willing to do so at this time, according to the chair of the AGRI Committee [85]. Most roll-calls related to this topic consequently correspond to amendments to the text proposed by the commission. For instance, the first amendment proposed by S&D, which matches the characteristic pattern, aims at capping the direct payment at 200 k€. Its goal here is to decrease the support granted to large agriculture structures without affecting small- and middle-sized businesses. This change is first rejected by EPP, but a compromise is later found: it consists in raising the cap to 300 k€. The corresponding roll-call belongs to cluster C_5^3 (discussed next), which therefore exhibits a much different characteristic pattern.

4.5.3.3 Environmentalists vs. the Rest

Cluster C_5^3 contains $74/232$ roll-calls (32%). Its characteristic pattern $\widehat{P_5^3}$, shown in Figure 4.3c, finds the environmentalist group (G-EFA) opposing a large module constituted of the rest of the MEPs. The far-left group (GUE-NGL) is apart, as one MEP agrees with the environmentalists whereas the rest of his group abstains. This is very similar to the pattern obtained by the tradi-

tional approach when solving CC on the layer aggregated network, except for the NI group and a few MEPs. In particular, Corinne Lepage, which was described in Section 4.5.1 as an environmentalist member of ALDE, is placed in the G-EFA module by our current method. The relevance of this module is confirmed by her activity both at the EP, where she is very active on issues such as *food safety*, and outside, through her own initiative reports and consciousness-raising conferences [7]. The roll-calls composing this cluster are mainly associated with amendments related to environmental aspects of agriculture, and most are proposed by G-EFA, sometimes in collaboration with C. Lepage.

Obviously, the singular position of G-EFA in this characteristic pattern is caused by its systematic opposition to the other groups on the texts associated with this cluster. However, one can distinguish two different situations. On the one hand, G-EFA tables amendments to enhance and complete the social and/or environmental regulations proposed in the amended text, and then vote in their favor. For instance, a legislative text was presented to include *crop diversification* (in opposition to monoculture) among the rules that farmers must enforce to obtain direct payments. G-EFA proposed to add *crop rotation* (growing different crops on the same land each year) to these requirements, as it considers it to be as crucial (in addition to crop diversification), because it helps increasing productivity as well as reducing the use of chemical fertilizer [214].

On the other hand, G-EFA opposes amendments considered as not environmentally and/or socially progressive enough, like in the case of milk quotas. An amendment was proposed to trigger milk quotas only in case of severe market imbalance. These quotas were introduced in the EU in 1984, in order to prevent milk overproduction [80]. In 2008, the EP had already decided to increase gradually the milk quotas and abolish the quota regime in 2015. According to G-EFA, the quotas should not be abolished at all, as they favor high-quality production rather than overproducing and selling EU surplus to other countries [83]. Therefore, the group opposed the amendment.

By comparison, all the other groups are in favor of activating milk quotas only in cases of severe market disturbance. Nonetheless, they position themselves differently on the quota system itself. For S&D, it allows balancing between supply and demand, and contributes to market efficiency. The group assumes that its complete removal would pose serious problems. On the contrary, ALDE and EPP aim at increasing competitiveness in order to meet the global demand, and want to reduce quotas as much as possible [83]. Interestingly, this characteristic pattern is the only one, with unanimity ($\widehat{P_5^1}$), for which the Euroskeptics (NI, EFD) do *not* abstain. It is difficult to assess why exactly it is the case, but this is likely to denote the importance they give to the concerned issues (yet, they have an official position on other topics, as shown in Table 4.2, and nevertheless abstained for some of them). Like G-EFA, they too express their will to support product quality and producer price guarantee. Moreover, it is worth noticing that this period corresponds to a significant broadening of their electoral base in rural areas [195]. Therefore, one reason for their behavior could be to support these small farmers.

4.5.3.4 S&D/EPP vs. the Rest

Cluster C_5^4 represents $18/232$ (8%) roll-calls. Its characteristic pattern $\widehat{P_5^4}$, shown in Figure 4.3e, contains a module formed by the far-left, environmentalist and liberal groups (GUE-NGL, G-EFA, ALDE), vs. another module containing the socialists and conservatives (S&D, EPP), while both Euroskeptical groups form an abstentionist module. This constitutes a new type of characteristic pattern, different from all the others met until now, including in the baseline. In particular, it is worth noticing that S&D and ALDE do not belong to the same module. Thus, if these groups alternatively side with left- and right-wing groups, as already assumed by the classical layer aggregation approach presented in Section 4.5.1 (and as illustrated before), our current method shows that they do not always do so simultaneously.

The example of the gradual elimination of *export refunds* [76], and its impact on developing countries, is particularly illustrative of the positions adopted by the different groups. Export refunds are subsidies granted for certain products (e.g. cereals, rice, sugar, beef and veal, milk and dairy products) that are exported outside the EU, to enable EU exporters to better compete on world markets. According to ALDE, which argues for agriculture to be liberalized as much as possible, keeping export subsidies would cause unfair competition and might deteriorate rural development [6]. As a result, ALDE was in favor of phasing-out export refunds. G-EFA was also against export subsidies, but for different reasons. The group normally supports any market regulation guaranteeing fair producer prices and encouraging product quality. However, it considers that this specific piece of regulation goes in the opposite direction of its ideology [215] , and therefore opposes it. S&D also criticizes continuing export subsidies, but at the same time, the group is not in favor of eliminating them, because this might weaken the EU's hand in worldwide trade negotiations [83]. EPP is not against eliminating the export subsidies in general, but they want to keep them in case of crisis.

4.5.3.5 Unholy Alliance

For Cluster C_5^5, the characteristic pattern $\widehat{P_5^5}$, as illustrated in Figure 4.3f, finds a module gathering environmentalists and right-wing liberals and conservatives (G-EFA, ALDE, EPP), opposing a module composed of the far-left and socialist groups (GUE-NGL and S&D), while the Euroskeptics abstain once again. These modules are surprising from a political standpoint, as they exhibit a somewhat unholy alliance between environmentalists and conservatives, whose views generally clash for AGRI matters. But the characteristic pattern is also surprising when considering the results of Section 4.5.1, as this alliance was not detected at all. The cluster contains only 6 roll-calls (2%), which shows that this situation does not happen often.

Examining the concerned documents and debates reveals that these groups vote similarly (in this specific context), but for different reasons, as shown later by the subsequent amendments they tabled. Let us take for example the case of *green payments* [78], i.e. direct payments specifically tar-

geting agricultural practices beneficial for the climate and the environment. On the one hand, G-EFA considers that the constraints related to biodiversity are not strong enough, as already mentioned for crops when discussing $\widehat{P_5^3}$. On the other hand, EPP thinks that there are too many requirements to earn an organic farming certification, and that a subset of these constraints would suffice [6]. Later, each group proposed a few amendments trying to pull the original text in its own ideological direction. Each group supported its own amendments and voted against the other's, fitting the previously discussed *Environmentalists vs. the Rest* and *Conservatives vs. the Rest.* patterns, and thus highlighting their disagreement. This specific example is important, as it shows that voting similarly is not equivalent to having the same opinion.

4.5.3.6 Comparison with the Baseline

Our results confirm in a more objective way the assumption of those of the traditional approach presented in Section 4.5.1, based on the RCC pattern from Figure 4.3, and according to which S&D and ALDE sometimes vote like the left-wing groups (as in $\widehat{P_5^2}$) and sometimes like the right-wing ones ($\widehat{P_5^3}$). Our method additionally identifies the documents for which the EP adopts these two patterns: it turns out most of them are amendments to the same legislative propositions, in both clusters C_5^2 and C_5^3. But our method also shows that these two groups vote differently on a number of occasions ($\widehat{P_5^4}$ and $\widehat{P_5^5}$), a fact overlooked when using the traditional approach.

In addition, our results uncover the fact that the Euroskeptics systematically abstain on most documents, and only vote for a specific subset corresponding to the *Green vs. the Rest* pattern. This specific behavior put them apart from the rest of the groups, and maybe this is why they had been categorized by the traditional approach as an intermediate group, like S&D and ALDE, in Section 4.5.1. However, our results show that this is an artifact of the Euroskeptics' abstentionist behavior, and that they hold a completely different position than S&D and ALDE.

Finally, our method allows identifying the *Unholy Alliance* pattern, which had completely been overlooked by the layer aggregation approach in Section 4.5.1. It corresponds to a very surprising coalition, politically speaking, which emerged when voting for a very specific set of legislative documents, also identified by our method. By leveraging amendments related to the concerned documents, our method even allows identifying specific points of agreement and disagreement in these texts.

4.6 Conclusion

In this chapter, we have presented a new method to partition multiplex signed networks. Our four-stepped method first partitions each layer separately to obtain as many patterns, computes the

6. Incidentally, it is worth noting that changing the definition of organic farming would make it easier to receive green payments.

similarity between these patterns, then groups them using cluster analysis, and characterizes each such cluster through a representative pattern. These steps can be implemented in various ways, depending on the application. By applying it to a subset of the 7^{th} term European Parliament dataset presented in [21], we could uncover findings which had been overlooked by traditional approaches. For instance, we could not only confirm that the French S&D and ALDE MEPs alternatively side with the left- and right-wing groups, but also identify which topics are concerned by these swings. We could also uncover more surprising results, such as the so-called unholly alliance between right conservatives and environmentalists on a specific subset of propositions. In addition, we could show that they were voting similarly, but for completely opposed rationales.

We believe that our work opens new directions for future research. First, our method is generic enough so that it could be also adapted to unsigned graphs or any kind of partitionable data, with slight modifications. Second, our method currently treats patterns equally during the characterization step. It is possible to weight appropriately the patterns by leveraging the topological information of their corresponding layers. This can be viewed as a hybrid approach with combines both network- and partition-centric approaches. Finally, another interesting future direction is to generalize our method for multilayer signed networks, which subsumes inter-layer edges. This can offer to take into account the relations/dependencies between layers, when such information exists.

Chapter 5

Enumeration of the space of optimal solutions for the Correlation Clustering problem

5.1 Introduction

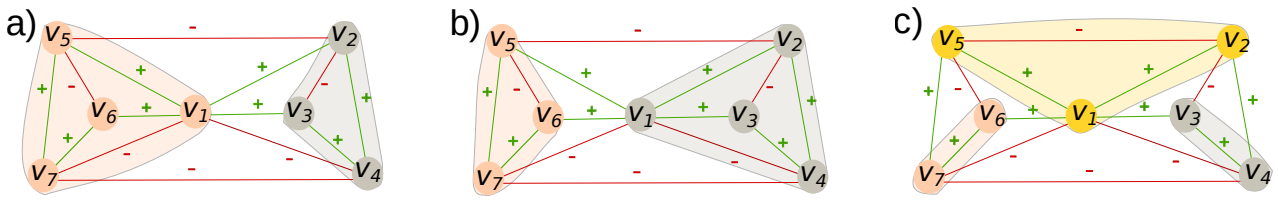
We have shown in Chapter 4, through an application of vote analysis, that a single partition of a given set of vertices may not reflect the meso-structure of a network. It is possible that one needs to seek for multiple partitions to get a better understanding. This need to look for multiplicity could be seen as related to such networks having multiple layers. However, we will see that it also holds even when considering a single layer, i.e. a uniplex network. When solving an instance of the CC problem, the standard approach in the literature, be it heuristic or exact, is to find a *single* solution and focus the rest of the analysis on it, as if it was the *only* solution.

Yet, it is possible that several, and even many, other optimal solutions exist for the considered instance [56, 63, 62, 57, 38]. In this chapter, we relax this single-partition assumption to allow searching for *all optimal* partitions. To the best of our knowledge, the issue of multiple optimal solutions for the CC problem is first pointed out by Davis [56] for perfectly balanced *incomplete* signed graphs, as he gives an example of how such graphs may have several optimal partitions. In particular, he states that a signed graph should have a unique optimal partition, otherwise, it amounts to a lack of cluster structure. The issue is then also confirmed by Doreian and Mrvar [63] (also in [62] with more networks) for imbalanced *incomplete* signed graphs, and the authors integrate this knowledge into their heuristic method to collect all discovered best partitions across a large number of restarts, evidently with the risk of obtaining local optima. Later, Brusco and Steinley [38] overcome this local optima issue by adapting their *ad hoc* B&B programming exact method to enumerate multiple optimal partitions.

Figure 5.1 illustrates the issue of multiplicity on a complete unweighted signed graph (see caption). Solving the CC problem for this graph of only 7 vertices yields no less than 22 distinct *optimal* solutions. We show only a few of them to highlight how different these can be. For instance, on the one hand P^a and P^b are very similar, partition-wise, as they are both bisections differing only in the module assignment of v_1. On the other hand, P^c is quite different from them: it contains an extra module obtained by separating an element from each module of the previous solutions, in addition to v_1.

Nevertheless, guaranteeing the completeness of the optimal solution space enumeration requires employing exact approaches. It is well known that, due to their NP-hard nature, exact approaches solving most clustering problems (including the CC problem) do not scale well even when looking for a *single* optimal solution [103]. This constrains the number of vertices that we can handle. We could instead look for *quasi*-optimal solutions, which can be found faster using a heuristic method. However, we want to study the CC problem *itself*, and not some of its existing resolution methods. Using a heuristic-based method would introduce a bias in the way the solution space is explored, and thus in our study of the problem.

In this chapter, we propose a new efficient method for the enumeration of the CC optimal so-



(a) P^a (b) P^b (c) P^c

Figure 5.1 – Three (out of 22) different optimal CC solutions obtained for the same network: a) $P^a = \{\{v_1,v_5,v_6,v_7\}, \{v_2,v_3,v_4\}\}$; b) $P^b = \{\{v_5,v_6,v_7\}, \{v_1,v_2,v_3,v_4\}\}$; and c) $P^c = \{\{v_1,v_2,v_5\}, \{v_6,v_7\}, \{v_3,v_4\}\}$. Red and green lines represent negative and positive edges, respectively. The graph is complete, but for clarity, some negative edges between modules are intentionally omitted.

lution space. Motivated by the high similarities that optimal solutions can share (e.g. Figure 5.1), it integrates into an exact approach several local search mechanisms combining neighborhoods from small to large sizes. The efficacy of our method relies on the use of neighborhoods of different sizes in a recurrent manner which allows to explore different regions of the optimal solution space. In order to accelerate the proposed enumeration method, we developed pruning strategies based on optimal conditions satisfied by partial solutions.

In this work, we focus only on *unweighted* signed graphs, although our proposed method can handle edge weights. Application-wise, unweighted signed networks are much less used in the literature, but they nevertheless fit certain modeling situations and methodological choices. Accordingly, it can be that binary values better represent the studied relations (e.g. alliance/conflict between countries in international relationships [65]), or that the authors prefer to use such values for practical reasons (e.g. limited or unreliable information [70]).

Contributions. The following content is based on our work submitted to *Journal of Global Optimization* [15]. This chapter makes the following contributions:

1. **Enumeration method.** We propose an efficient enumeration method to generate all optimal solutions of a given unweighted signed network.
2. **Evaluation.** We present extensive computational experiments established for the proposed method, relying on sparse and dense signed unweighted networks.

The rest of this chapter is organized as follows. In Section 5.2, we review the works identifying the issue of multiple optimal solutions and those related to the exact enumeration of all optimal solutions of the CC problem. Next, we introduce our enumeration method in Section 5.3 and detail our neighborhood search and pruning strategies in Section 5.4 and 5.5, respectively. We put the proposed method into practice on a selection of complete and incomplete signed networks and discuss our results in Section 5.6. Finally, we review our main findings in Section 5.7, and identify some perspectives for our work.

5.2 Related Work

There are some methods proposed in the literature to solve CC exactly [57, 90, 130], however very few methods were proposed for enumerating the CC optimal solution space. In fact, more generally very few methods were proposed for enumerating the optimal space of any combinatorial problem (e.g., [24] for maximal covering problem). In Section 5.2.1, we first summarize the works identifying the issue of multiple optimal solutions, we then review the existing and related optimal space enumeration methods for the CC problem in Section 5.2.2.

5.2.1 Existence of Multiple Optimal Solutions

As mentioned in Section 5.1, multiple optimal solutions for the CC problem are already identified in the literature [56, 63, 62, 38]. Although considerable efforts are made in both of these works to deal with the multiplicity of solutions, their authors do not try to study the optimal solution space of the CC problem. This might be due to the fact that the number of optimal partitions they encountered was small, around 20, for most of the networks they considered [62]. Doreian et al. [62] suggest to use the multiplicity as an additional criteria to select the most appropriate number of modules, in cases where the optimal imbalance value is reached for several values of this number of modules. Brusco and Steinley apply this principle in [38]. However, as we show in Section 5.6.3.1, in practice the number of optimal solutions can be much larger than 20.

The problem of multiplicity is of general interest, and was studied in the context of other optimization problems than CC. There are just a few of these works, thus for the sake of completeness we briefly cover them here. Paris [182] proposes to take advantage of multiple optimal solutions to perform a more thorough validation of linear programming economic models, in order to provide more flexibility at decision-making. Liu et al. [150] tackle the multiplicity for the *Optimal Load Distribution* problem to manage multiple generator units in hydropower plants. Ruiter et al. [197] show in the context of Adjustable Robust Optimization that even when all optimal solutions have the same worst-case cost, their mean costs can drastically differ, which allows discriminating between optimal solutions. Arthur et al. [24] also recognize the need to identify all optimal solutions for the *Maximal Covering* problem in the context of geosciences. In addition, they observe a connection between the size of the problem (number of units) and the number of optimal solutions.

All these works show that 1) there can be multiple optimal solutions in practical contexts; and 2) identifying all or several of these multiple solutions is informative, and therefore worthwhile, as they can be leveraged to improve the results application-wise. Among other things, the work we present in this chapter extends the findings of Davis [56] and Doreian et al. [62] by showing that the issue of multiplicity also occurs for *complete* imbalanced signed graphs. Moreover, we study how certain parameters of the problem affect the multiplicity of solutions.

5.2.2 Enumerating All Optimal Solutions

The literature provides three main methods to enumerate all optimal solutions of a combinatorial problem: *Branch-and-Bound (B&B)*, *Fixed-Parameter Enumeration* and *Weighted partial MaxSAT-based*. The literature for B&B is more prevalent for the CC problem and most works concentrate on it.

5.2.2.1 Branch-and-bound (B&B)

Like for any combinatorial problem, the enumeration of all optimal solutions of the CC problem through a B&B method can be performed by using two different algorithmic approaches: Integer Linear Programming (ILP) vs. *ad hoc* B&B programming. Their main difference is that the former can tackle any problem translated on the language of mathematical programming through any industrial optimization solver, whereas in the latter the construction of the B&B tree relies on problem-specific idiosyncrasies. Based on how the B&B tree is used, the ILP approach can be implemented in two different ways. The first one relies on a simple sequential process: a slightly modified version of the original problem is solved as many times as solutions in the optimal solution space (like in [24] for maximal covering problem). In each iteration, already-found optimal solutions are sequentially added as constraints into the mathematical model to exclude them. However, the drawback of this approach is that a B&B tree needs to be built from scratch for each new optimal solution found.

Danna et al. [54] reduce the computational effort with a more efficient two-step method than the simple sequential approach, called *OneTree*. As its name suggests, this methods constructs a single B&B tree. In simple terms, the authors first build and explore the search tree in order to find efficiently the first optimal solution, then enumerate all the other optimal solutions based on the same tree. This method is currently incorporated in the industrial optimization solver CPLEX [117] and we detail in Section 5.3.2 how we use it for the CC problem.

An *ad hoc* B&B programming method constructs the B&B tree different from the previous approach. Namely, these methods for a clustering problem systematically construct partial solutions by assigning the vertices to one of the existing modules. This forms a search tree, where the branches of the tree correspond to assignments of vertices to modules while the nodes correspond to partial assignments. Similar to Danna et al. [54], Brusco and Steinley [38] propose a two-step method for generating all CC optimal solutions. In their method, they first identify the optimal objective function value by finding an optimal solution, then use the optimal value as input for another branch-and-bound tree from scratch. As shown in Figueiredo and Moura [90] this method cannot deal well for finding a single optimal solution of graphs with more than 20 vertices.

5.2.2.2 Fixed-Parameter enumeration

This is also an *ad hoc* method which requires to transform a part of the considered problem as a new input parameter. This in turn guarantees to bound the overall running time in function of an input parameter. Assuming that this input parameter is small, the parameterized enumeration is efficient in practice. Damaschke [53] proposes an FPT (Fixed-Parameter Tractable) algorithm to enumerate all optimal solutions in a given graph for the *Cluster Editing* problem, which is equivalent to the CC problem when the input graph is complete and unweighted. Concretely, this FPT algorithm makes the number of frustrated edges in structural balance parameterized to solve the problem. However, a drawback of this approach is that the amount of imbalance, the input parameter, can be very large. Furthermore, Damaschke does not provide the computational results in his theoretical work.

5.2.2.3 Weighted partial MaxSAT-based enumeration

A weighted partial MaxSAT allows to define soft and hard clauses of Boolean variables and a function that associates a non-negative cost with each of the soft clauses. Any truth assignment that satisfies the hard clauses gives a valid solution. In the literature, Saikko et al. [198] propose a weighted partial MaxSAT solver, called *LMHS*, to enumerate all optimal solutions. Their enumeration strategy relies on a sequential approach, as in [24] for a ILP, such that an already-found optimal solution is added as a clause. The authors point out based on their experiments that integrating a preprocessing step into the solver can degrade solver performance compared to no preprocessing. Regarding the CC problem, as mentioned in Section 2.1, three different weighted partial MaxSAT formulations are proposed in the literature [32]. Nevertheless, the performance of *LMHS* for the CC problem is not known yet. Particularly, like the preprocessing step, its performance can be affected by the choice of a MaxSAT formulation.

The computational results presented in the works mentioned in this section places the ILP B&B as the more efficient method for exactly solving the CC problem. This is explained by a tighten initial linear relaxation of the ILP formulation and the quality of the cuts used. In this work, we apply a ILP B&B method for the complete enumeration of the CC optimal space and propose a compromise strategy between the sequential approach and the *OneTree* method from Danna et al. [54].

5.3 Enumeration of the optimal solution space for the CC problem

Any of the ILP formulations described in Section 2.2 is in theory sufficient to enumerate all optimal solutions. Nevertheless, this task is more difficult than finding a single optimal solution and requires dealing with different challenges, such as finding an alternative optimal solution and ensuring the completeness of the solution space. Moreover, a solution space enumeration method can be affected by the choice of the formulation and its exact method used to solve it. This is why, for

a fair comparison, we opt for using a single formulation and its best resolution method, which are beneficial for both solution space enumeration methods described in Section 5.3.2. Accordingly, we use $F_v(G)$, as defined in Chapter 2 and solve it by applying the following strategy. We strengthen its polytope $P_v(G)$ through a *cutting plane* approach [174] with 2-partition (Equation 2.17) and 2-chorded cycle (Equation 2.18) valid inequalities. We add the tight valid inequalities (only) during the root relaxation phase, before proceeding to the construction of the B&B search tree. Note that using $F_v(G)$ amounts to keep the redundant inequalities in the formulation (Section 2.2.1). This is a methodological choice, which allows to keep away from numerous invalid optimal solutions on incomplete signed networks, a fact already mentioned in Section 2.2.1. Otherwise, this can be very time-consuming, especially for the sequential approach. Nonetheless, removing the redundant inequalities, i.e. considering $F_v^\star(G)$ instead of $F_v(G)$, can be beneficial for the considered enumeration methods on complete signed networks, as we briefly see later in Section 5.6.4.

In the following, we first present an ILP-based method for finding an alternative optimal solution (Section 5.3.1), then pass to the methods enumerating all optimal solutions (Section 5.3.2). Throughout this work, we denote $F_v{+}(G)$ as the strengthened ILP obtained after executing $B\&C(F_v(G))$, i.e. the formulation obtained by adding to $F_v(G)$ all the cuts generated by $B\&C(F_v(G))$. Finally, let $B\&B(F_v{+}(G))$ be the Branch-and-Bound procedure based on $F_v{+}(G)$. In the rest of this chapter, by abuse of notation, the term *solution* designates a graph partition, as well as a feasible solution of the formulation $F_v(G)$.

5.3.1 Finding an alternative optimal solution

In the methods introduced in Section 5.3.2, we need to find an optimal solution different from a set S of previous ones found. For this matter, we need to define an extended model of $F_v{+}(G)$, that we denote $F_v{+}jump(G, S)$.

Consider a partition P. Let $x^p \in \{0,1\}^{\frac{n(n-1)}{2}}$ be the representative vector of P defined as: $x^p_{uv} = 1$ if u and v belong to the same module in P; and $x^p_{uv} = 0$ otherwise. $F_v{+}jump(G, S)$ simply includes the set of constraints defined in Equation (5.1) on top of $F_v{+}(G)$:

$$\sum_{u,v \in V : u<v} |x^p_{uv} - x_{uv}| > 0, \forall P \in S. \tag{5.1}$$

We refer readers to [92] for the implementation details of these constraints.

We see that this extended formulation allows us to find an alternative optimal solution other than the ones in S. Furthermore, given an optimal solution $P \in S$, to ensure that this alternative solution has the same objective value as $I(P)$, we add the objective function defined in (2.7) as a constraint,

as shown in Equation (5.2).

$$\sum_{u,v \in V:(u,v) \in E^-} x_{uv} + \sum_{u,v \in V:(u,v) \in E^+} (1 - x_{uv}) \leq I(P) \tag{5.2}$$

Equation (5.1) along with Equation (5.2) ensure that each feasible solution to $F_v{+}jump(S)$ is an optimal solution to the original problem. Finally, we denote the corresponding B&B procedure based on formulation $F_v{+}jump(G, S)$ by $B\&B(F_v{+}jump(G, S))$.

5.3.2 Enumerating all optimal solutions

As we have mentioned before, the CC problem can have multiple optimal solutions. In this section, we are interested in methods enumerating all optimal solutions of the CC problem for a given signed graph G. In the following, we first present $OneTreeCC$ [54] which is the best general enumeration solution space method available in the literature, incorporated in CPLEX [117], applied to formulation $F_v{+}(G)$. Then, we introduce our *ad hoc* method $EnumCC$ in the aim of obtaining a faster method able to handle relatively larger graphs.

We start with $OneTreeCC(G)$, which takes in input a signed graph G. The sketch of this two-step method is shown in Algorithm 1. In the first step (line 2), $OneTreeCC(G)$ finds an initial optimal solution based on $F_v{+}(G)$. We note $T_{B\&B}$ the B&B search tree constructed during this step. Besides storing the search tree, the fathomed nodes in this first step are stored for further examination during the second step. Moreover, Danna et al. [54] completely turn off dual tightening in the B&B procedure in order to guarantee exhaustive enumeration. According to the authors, the latter can cause a negative impact on performance. In the second step (line 3), $OneTreeCC(G)$ enumerates the set S of all other optimal solutions by expanding the search tree $T_{B\&B}$ until obtaining the complete enumeration of all optimal solutions.

Algorithm 1: $OneTreeCC(G)$

Result: The set of all optimal solutions S

1 $S = \emptyset$

`/* 1st step */`

2 Solve $B\&B(F_v{+}(G))$ to obtain an optimal solution P, and let $T_{B\&B}$ be the constructed B&B search tree

`/* 2nd step */`

3 Expand fathomed nodes in $T_{B\&B}$ until enumerating the set S of all other optimal solutions.

To compete with $OneTreeCC(G)$, we propose a new method for the complete enumeration of the CC optimal space. This method, $EnumCC(G, r_{max})$, with an input signed graph G and a maximum distance parameter r_{max}, shown in Algorithm 2, can be viewed as an improved version of the sequential approach proposed by [24] for the Maximal Covering problem (described in Sec-

tion 5.2). In the first step (line 2), $EnumCC(G, r_{max})$ starts obtaining an initial optimal solution with $B\&B(F_v{+}(G))$. In the second step, instead of directly "*jumping*" onto undiscovered optimal solutions one by one through $F_v{+}jump(G, S)$ (like in the sequential approach), we slightly change this process. $EnumCC(G, r_{max})$ first discovers the recurrent neighborhood $RN_{\leq r_{max}}(P)$ of the current optimal solution P (line 4), with the hope of discovering new optimal solutions. The recurrent neighborhood $RN_{\leq r_{max}}(P)$ of an optimal solution P, represents the set of optimal solutions, reached directly or indirectly from P depending on the maximum distance parameter r_{max}. The complete description of the RNS is given in Section 5.4. Whether a new solution is found or not through $RNS(G, P, r_{max})$, the jumping process into a new solution P is performed (line 6). This process is repeated, until all optimal solutions are discovered.

Clearly, $EnumCC(G, r_{max})$ can only improve the sequential approach used in [24], provided that RNS discovers at least one new solution and that its execution time is faster than that of $B\&B(F_v{+}jump(G, S))$. Furthermore, the choice of r_{max} is delicate: it does not contribute much to the method when it is small, whereas the method becomes slower than the sequential approach when r_{max} is large.

Algorithm 2: $EnumCC(G, r_{max})$

Result: The set of all optimal solutions S

1 $S = \emptyset$

/* 1st step */

2 Solve $B\&B(F_v{+}(G))$ to obtain an optimal solution P

/* 2nd step */

3 **while** $P \neq null$ **do**

/* step 2.1 */

4 Apply $RNS(G, P, r_{max})$ to generate the recurrent neighborhood $RN_{\leq r_{max}}(P)$ of P

5 $S = S \bigcup RN_{\leq r_{max}}(P)$

/* step 2.2 */

6 Jump onto an undiscovered optimal solution P, with $P \notin S$, through $B\&B(F_v{+}jump(G, S))$ // if not possible, then $P = null$

7 **end**

The jumping phase in step 2.2 of $EnumCC(G, r_{max})$ can be time-consuming for two reasons. First, even finding an alternative solution can be costly. Second, the number of jumps, denoted by the notation $n_{jump}(.)$ (here, $n_{jump}(EnumCC(G, r_{max}))$), can be very large, despite the use of the neighborhood $RN_{\leq r_{max}}(\pi)$. Nonetheless, we expect to save time and compensate the time spent in the jumping phase thanks to $RN_{\leq r_{max}}(\pi)$.

In the rest of this work, we leave out the common parameter G from the notations of the mentioned methods and ILP models, for the sake of convenience: $OneTreeCC()$, $EnumCC(r_{max})$ and $F_v{+}jump(S)$.

In the next section, we detail the recurrent neighborhood search RNS used in Algorithm 2.

5.4 Recurrent Neighborhood Search (RNS)

Recurrent Neighborhood Search (RNS) is the common and undoubtedly the most important part of our method *EnumCC*. As we have mentioned in the last section, if we want that this method can compete with $OneTreeCC()$, *RNS* needs to be efficient and fast.

Before defining the neighborhood of a solution, i.e. a partition, we need to determine a distance function for partitions. In the following, we first present minimum edit distance as our distance function (Section 5.4.1). Then, we introduce the algorithmic details of the *Complete Neighborhood Search (CoNS)* used in our Recurrent Neighborhood Search (Section 5.4.2). Finally, we explain the *Recurrent Neighborhood Search (RNS)* (Section 5.4.3).

Let us first introduce our notations related to graph partitioning. Remember that, $G = (V, E, w, s)$ is a signed graph, $n = |V|$ and each partition concerns the vertex set V. Since we consider only unweighted signed networks in this chapter, we define the entries, a_{uv}, of the signed adjacency matrix, **A**, as in Equation 5.3.

$$a_{uv} = \begin{cases} 1, & \text{if } (u,v) \in E^+, \\ -1, & \text{if } (u,v) \in E^-, \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

Let $P = \{M_1, ..., M_\ell\}$ $(1 \leq \ell \leq n)$ be an ℓ-partition of V, i.e. a division of V into ℓ non-overlapping and non-empty subsets M_i $(1 \leq i \leq \ell)$ called *modules*. The partition P is called a *solution* of the CC problem for the given graph G. A partition P with ℓ modules can be associated with one or more *membership vectors*. Such vector of size n, denoted by π, defines a function $\{1, 2, ..., n\} \rightarrow \{1, 2, ..., \ell\}$. For a given vertex u, $\pi(u)$ is the corresponding module *label* in partition P, while $M_{\pi(u)}$ denotes the module to which vertex u belongs in P. Finally, given a partition P, we define an r-neighborhood structure, denoted by $N_r(P)$, as the family of partitions obtained by moving r vertices in P from their source modules into the others. We also define $N_r(\pi)$ as the family of membership vectors obtained by moving r vertices in π.

5.4.1 Edit Distance

In Natural Language Processing (more generally in Computer Science) the edit distance [122] is defined as the minimum number of edit operations required to transform one string into another (or more precisely, their total cost). In the context of graph partitioning, when removing an existing vertex or inserting a new one is not allowed, edit operations define *neighborhood search operators* in local search heuristics developed for graph problems [34, 154, 1]. In this context, an edit operation is defined on two membership vectors, each storing the module labels of the vertices. Such an operation consists in moving one or more vertices, called *moving vertices*, from their current modules (called *source* modules) to other ones (called *target* modules). This transforms a *source*

membership vector into a *target membership* one. The number of moving vertices constitutes the cost of the edit operation. In this work, we are interested in edit operations whose cost is minimal.

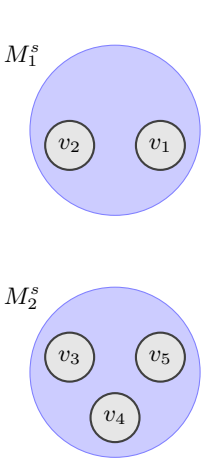
Definition 5.1 (Min-edit operation)**.** *Consider an edit operation transforming a source membership vector into a target membership one. If the set of moving vertices is the minimum set required for this transformation, then it is a* ***min-edit operation****. Otherwise, we call it* ***non-min-edit operation****.*

Notice that the cost of an edit operation is not always minimal. This is because two membership vectors, i.e. the module labels of two partitions, can be very different, but essentially suggest very similar module assignments for the vertices. The distinction between *min-edit* and *non-min-edit* operations is illustrated in Figure 5.2.

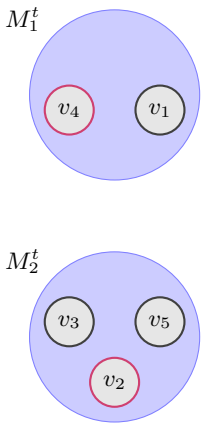
We also note that the *edit distance* between two membership vectors is the cost of the min-edit operation allowing to turn one vector into the other. The calculation of such distance between two membership vectors can be done in polynomial time by solving an assignment problem (see Appendix C.1 for more details).

Let us now introduce our notations related to edit distance. Let $\boldsymbol{\pi}^s$ and $\boldsymbol{\pi}^t$ represent the source and target membership vectors for an edit operation. They are associated with the source and target partitions $P^s = \{M_1^s, M_2^s, ..., M_{\ell^s}^s\}$ and $P^t = \{M_1^t, M_2^t, ..., M_{\ell^t}^t\}$ with ℓ^s and ℓ^t modules, respectively. Since the edit distance is symmetric, without loss of generality, let $\ell^s \leq \ell^t$. Moreover, we designate $\pi^s \rightarrow \pi^t$ an edit operation applied onto P^s to obtain P^t, whose set of moving vertices is defined as $\overset{s\rightarrow t}{V} = \{u \mid \pi^s(u) \neq \pi^t(u)\}$. This means that the remaining vertices, called *non-moving* vertices, do not change modules, hence their module labels are the same in π^s and π^t. The cost of this edit operation is denoted by $cost(\pi^s \rightarrow \pi^t)$, and computed by taking the cardinality of $\overset{s\rightarrow t}{V}$. In the rest of the text, we use r for short whenever we are interested only in the value of $cost(\pi^s \rightarrow \pi^t)$. Also, a *$r$-edit operation* (resp. *min-r-edit operation*) is any edit operation (resp. min-edit operation) whose cost equals r. Consider an edit operation $\pi^s \rightarrow \pi^t$ whose moving vertices are in set $\overset{s\rightarrow t}{V}$. The distinct labels of the source and target modules of a subset $V' \subset V$ of vertices are denoted, respectively, by $\pi^s(V') = \bigcup_{u \in V'} \pi^s(u)$ and $\pi^t(V') = \bigcup_{u \in V'} \pi^t(u)$. Finally, by abuse of notation, let us define $M_L^t = \bigcup_{l \in L} M_l^t$ for any $L \subseteq \{1, .., \ell^t\}$ and $M_L^s = \bigcup_{l \in L} M_l^s$ for any $L \subseteq \{1, .., \ell^s\}$.

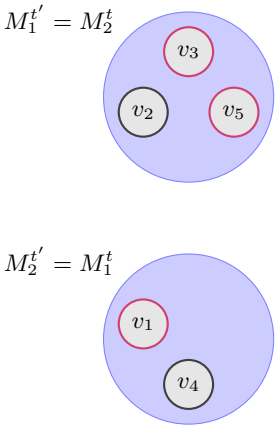
To illustrate the aforementioned concepts, let us consider the example of Figure 5.2. As shown in Figure 5.2.a, $\pi^s = (1, 1, 2, 2, 2)$, where $M_1^s = \{v_1, v_2\}$ and $M_2^s = \{v_3, v_4, v_5\}$. Let us consider an edit operation $\pi^s \rightarrow \pi^t$ with the moving vertices $\overset{s\rightarrow t}{V} = \{v_2, v_4\}$ (Figure 5.2.b). The edit operation consists in moving v_2 into M_2^s and v_4 into M_1^s, which produces π^t. Hence, we have $M_1^t = (M_1^s \setminus \{v_2\}) \cup \{v_4\}$ and $M_2^t = (M_2^s \setminus \{v_4\}) \cup \{v_2\}$. The labels of the source and target distinct modules of $\overset{s\rightarrow t}{V}$ are $\pi^s(\overset{s\rightarrow t}{V}) = \{1, 2\}$ and $\pi^t(\overset{s\rightarrow t}{V}) = \{1, 2\}$, respectively. Also, we have $M^s_{\pi^s(\overset{s\rightarrow t}{V})} = M_1^s \cup M_2^s$ and $M^t_{\pi^t(\overset{s\rightarrow t}{V})} = M_1^t \cup M_2^t$ for this example. The sets $M^s_{\pi^s(\overset{s\rightarrow t}{V})}$ and $M^t_{\pi^t(\overset{s\rightarrow t}{V})}$ could be different, if some of the moving vertices move into a module M_i^s other than $M^s_{\pi^s(\overset{s\rightarrow t}{V})}$.



(a) Source membership vector $\pi^s = (1, 1, 2, 2, 2)$, where $M_1^s = \{v_1, v_2\}$ and $M_2^s = \{v_3, v_4, v_5\}$.



(b) Target membership vector $\pi^t = (1, 2, 2, 1, 2)$, where $M_1^t = \{v_1, v_4\}$ and $M_2^t = \{v_2, v_3, v_5\}$. The moving vertices are shown with a purple border. $\pi^s \rightarrow \pi^t$ is a min-2-edit operation, where $\overset{s\rightarrow t}{V} = \{v_2, v_4\}$.



(c) Target membership vector $\pi^{t'} = \{2, 1, 1, 2, 1\}$, where $M_1^{t'} = \{v_2, v_3, v_5\}$ and $M_2^{t'} = \{v_1, v_4\}$. The moving vertices are shown with a purple border. Since $M_1^{t'} = M_2^t$ and $M_2^{t'} = M_1^t$, $\pi^s \rightarrow \pi^{t'}$ is not a min-3-edit operation, where $\overset{s\rightarrow t'}{V} = \{v_1, v_3, v_5\}$.

Figure 5.2 – Illustrative example for two edit operations from the same source membership vector π^s: $\overset{s\rightarrow t}{V}$ (Fig. 5.2.b) and $\overset{s\rightarrow t'}{V}$ (Fig. 5.2.c). The underlying partition in P^t and $P^{t'}$ is the same, but the corresponding membership vectors π^t and $\pi^{t'}$ are different. Consequently, the former is a *min-edit* operation, whereas the latter is not.

5.4.2 Complete Neighborhood Search (*CoNS*)

$CoNS(G, \pi^s, r)$, which takes in input a signed graph G, a membership vector π^s associated with an optimal partition P^s of ℓ modules and an edit distance r, is a neighborhood search method which generates the set $N_r(\pi^s)$ of membership vectors associated with *all* optimal partitions, obtained by applying min-r-edit operations to a given membership vector π^s. To ensure the completeness of set $N_r(\pi^s)$, $CoNS(G, \pi^s, r)$ analyses all combinations of moving vertices and their target modules. Two pruning strategies are used in order to eliminate some combinations that do not conduct to an optimal partition when an edit operation is applied. Both of them are described in this section while the properties in which they are based in are detailed in Section 5.5. In simple terms, they are based on the relations between a set of moving vertices and their target modules. Next, we present $CoNS(G, \pi^s, r)$ as a pipeline procedure consisting of four parts, as shown in Algorithm 3.

In the first part, $CoNS(G, \pi^s, r)$ starts by selecting a candidate set of r moving vertices $\overset{s\rightarrow t}{V}$ among all vertices V (line 2). The set of all possible r moving vertices is obtained by generating all combinations $\binom{V}{r}$. We know that probably not every $\overset{s\rightarrow t}{V}$ leads to a min-r-edit operation and an optimal partition P^t. To detect such cases, we apply the first pruning procedure (line 3), called *internalPruning* and depicted in Algorithm 4. This procedure checks, without the knowledge of target

Algorithm 3: $CoNS(G, \pi^s, r)$.

Input : signed graph $G(V, E, w, s)$, source membership vector π^s with ℓ module labels, edit distance r
Output: $N_r(\pi^s)$

$T = \{1, 2, .., \ell\}$ // module labels in π^s
`/* 1st part */`
for *each* $\overset{s\to t}{V} \in \binom{V}{r}$ **do**
  if *internalPruning(G,* π^s, $\overset{s\to t}{V}$ *)* **then**
    go to line 2
  end
  `/* 2nd part */`
  $T_0 = \pi(\overset{s\to t}{V}) \bigcup \{Unknown\}$ // initial target module labels
  $T_{rem} = T \setminus T_0$ // remaining module labels
  for *each* $T_0' \in \mathcal{P}(\binom{T_0}{r})$, *s.t.* $\pi^s(u) \notin T_0', \forall u \in \overset{s\to t}{V}$ **do** // $\mathcal{P}(.)$: `permutations`
    Let π^t be the partition after assigning T_0' to $\overset{s\to t}{V}$
    if *externalPruning(G,* π^s, π^t, $\overset{s\to t}{V}$ *)* **then**
      go to line 8
    end
    `/* 3rd part */`
    Let B be the the number of *Unknown* labels in π^t
    if $B \neq 0$ **then**
      Let $\overset{s\to t}{V}_{rem}$ be the moving vertices, whose $\pi^t(\overset{s\to t}{V}_{rem}) = Unknown$
      for $b \in \{1, .., B\}$ **do**
        Let $\mathcal{I}$ be the set of all vectors of size B with elements belonging to $\{1, .., b\}$, s.t. each element for each vector appears once
        for *each* $I \in \mathcal{I}$ **do**
          Update π^t with the assignment of I to $\pi^t(\overset{s\to t}{V}_{rem})$
          if *externalPruning(G,* π^s, π^t, $\overset{s\to t}{V}$ *)* **then**
            go to line 18
          end
          `/* 4th part */`
          $T^+_{rem} = T_{rem} \cup \{\ell + 1, .., \ell + b\}$ // expand for empty modules
          for *each* $T_r' \in \mathcal{P}(\binom{T^+_r}{b})$ **do**
            Replace the assignment of I to $\overset{s\to t}{V}_{rem}$ by T_{rem} in π^t
            if $I(\pi^s) = I(\pi^t)$ *and* $\neg$*externalPruning(G,* π^s, π^t, $\overset{s\to t}{V}$ *)* **then**
              $N_r(\pi^s) = N_r(\pi^s) \bigcup \pi^t$
            end
          end
        end
      end
    else
      if $I(\pi^s) = I(\pi^t)$ **then**
        $N_r(\pi^s) = N_r(\pi^s) \bigcup \pi^t$
      end
    end
  end
end

modules, if $\overset{s\to t}{V}$ satisfies some connectivity conditions related to the atomicity property (defined in Section 5.5.2) and the MVMO property up to three moving vertices (defined in Section 5.5.3). As we see in Section 5.5, whenever one of these two conditions is not satisfied, the candidate set $\overset{s\to t}{V}$ can be pruned.

Algorithm 4: *internalPruning*(G, π^s, $\overset{s\to t}{V}$)

Input : signed graph $G(V, E, w, s)$, source membership vector π^s, moving vertices $\overset{s\to t}{V}$
Output: Boolean variable
/* 1st part */
1 **if** $\neg$*intAtomic*(G, π^s, $\overset{s\to t}{V}$) **then** // Properties 5.5 and 5.6.a
2 | **return** true
3 **end**
/* 2nd part */
4 **if** $2 \leq |\overset{s\to t}{V}| \leq 3$ *and* $\neg$*intMVMO*(G, π^s, $\overset{s\to t}{V}$) **then** // Lemma 5.10.1 and Lemma 5.11
5 | **return** true;
6 **end**
7 **return** false;

At this point, the set of moving vertices $\overset{s\to t}{V}$ is fixed. Next, the algorithm defines step by step the target membership vector π^t by setting the target module of each vertex in $\overset{s\to t}{V}$. We handle the process of generating the target membership vector π^t in three steps, which constitutes our second, third and fourth parts in Algorithm 3, in order to take advantage of our pruning strategy. The pruning strategy used in these parts is slightly different from the one in the first part: it is based on external connections from $\overset{s\to t}{V}$ to $\overset{s\to t}{V}$. The procedure relying on that is called *externalPruning* and depicted in Algorithm 5. This procedure verifies, based on external relations among $\overset{s\to t}{V}$, if $\overset{s\to t}{V}$ satisfies min-edit (Section 5.5.1), atomicity (Section 5.5.2) and MVMO (Section 5.5.3) properties. As we see in Section 5.5, whenever one of these three conditions is not satisfied, the possibility of target modules for candidate set $\overset{s\to t}{V}$ is pruned. These pruning strategies can be applied even when some target modules are not already decided. Therefore, it is invoked, whenever the target modules of $\overset{s\to t}{V}$ are updated (lines 10, 20 and 26).

In the second part, a moving vertex is allowed to move into a module, whose label is in T_0. The set T_0, defined at line 6, is the set of all unique source modules of $\overset{s\to t}{V}$ and a label *Unknown*. We also define T_{rem} as the remaining modules. Notice that, from this point a target membership vector π^t can contain multiple vertices with target modules labeled as *Unknown*, which means their target modules are not already assigned. The use of the *Unknown* module allows us to handle in the third part the other modules in T_{rem} as target possibilities. At line 8, we consider all possible permutations in $\mathcal{P}(\binom{T_0}{r})$ such that, for each vertex in $\overset{s\to t}{V}$, the target and source modules are different.

At this point, the *externalPruning* procedure can be applied for vertices whose target modules

are already assigned. If the pruning conditions are not satisfied, the algorithm proceeds to the third part. Line 14 checks if no target module in π^t is unknown ($B = 0$). If so, i.e. a new optimal partition has been found, the algorithm proceeds to the final test (line 32). However, if it is not the case, the algorithm enters in the fourth part: all the combinations in T_{rem} (more precisely, T_r^+) are considered as a possible assignment of unknown target modules. As we can expect, this task can be computationally expensive. Then, instead of directly determining the target modules of each vertex in $\overset{s\rightarrow t}{V}_{rem}$, we first generate all the *coupling* scenarios of moving vertices going into the same *Unknown* target module. To handle this, let b be the number of distinct *Unknown* target modules (line 16), for each value in the range of $\{1, .., B\}$. Then, we construct all possible *coupling* scenarios $\mathcal{I}$ for the value b (line 17). A pair of remaining moving vertices is coupled, if they move into the same *Unknown* target module. For instance, $\{Unknown1, Unknown1, Unknown2\}$ indicates that the last moving vertex moves into a different *Unknown* target module than the first two. Once the target membership vector π^t is enriched with a *coupling* scenario $\mathcal{I}$, the external pruning is reapplied in line 20. If pruning conditions are not satisfied, the algorithm finally determines the target modules of $\overset{s\rightarrow t}{V}_{rem}$, by respecting the actual *coupling* scenario $\mathcal{I}$ (line 25). In the end, we add the current $\pi^s \rightarrow \pi^t$ into $N_r(\pi^s)$, if the optimality condition is met (line 26).

Algorithm 5: *externalPruning*(G, π^s, π^t, $\overset{s\rightarrow t}{V}$)

Input : signed graph $G(V, E, w, s)$, source membership vector π^s, target membership vector π^t, moving vertices $\overset{s\rightarrow t}{V}$

Output: Boolean variable

/* 1st part */

1 **if** $\neg minEdit(\pi^s, \pi^t, \overset{s\rightarrow t}{V})$ *or* $\neg extAtomic(G, \pi^s, \pi^t, \overset{s\rightarrow t}{V})$ **then** // Properties 5.2, 5.4, 5.6.b and 5.7

2 | **return** true

3 **end**

/* 2nd part */

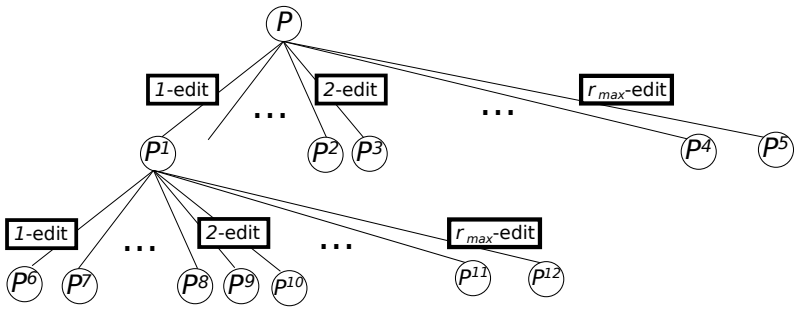
4 **if** $2 \leq |\overset{s\rightarrow t}{V}|$ *and* $\neg extMVMO(G, \pi^s, \pi^t, \overset{s\rightarrow t}{V})$ **then** // Lemmas 5.10.2, 5.11 and 5.12

5 | **return** true

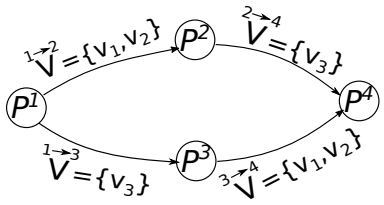
6 **end**

7 **return** false;

Algorithm 3 without problem-specific pruning strategies (internal and external verification of properties from Section 5.5.3) becomes a brute force method. Although $CoNS(G, \pi^s, r)$ can be very costly (even for small values of r), the properties announced in Section 5.5 allow us to develop a method whose cost is relatively lower than the brute force method. Indeed, given a membership vector of size n with ℓ module labels, the number of nonrepetitive permutations of all the combinations of r moving vertices is $n^r \times r!$ and the number of non-repetitive permutations of all the combinations of target modules is equal to $\ell^r \times r!$. In the end, the brute force method is $\Theta(n^r \times r! \times \ell^r \times r!)$,



(a) RNS Tree



(b) Example of duplication in RNS. Both P^2 and P^3 generate P^4.

Figure 5.3 – Illustrations for Recurrent Neighborhood Search (RNS). In both figures, circles represent membership vectors associated with optimal partitions.

where we assume that $\ell > r$. Computational results in Section 5.6.2 show that the application of the pruning strategies reduces the computational cost of this procedure.

5.4.3 Recurrent Neighborhood Search (*RNS*)

Recurrent Neighborhood Search (RNS) consists in constructing a search tree by applying *CoNS* from Section 5.4.2 in a recursive manner. We denote this search tree as *RNS tree*. This recursive procedure is depicted in Figure 5.3a. The root node of *RNS tree* corresponds to the initial partition P. All other nodes of the RNS tree constitute the set of optimal solutions reached directly or indirectly from P. A solution associated with a node in a level h of a *RNS tree* is obtained after $h-1$ applications of the *CoNS* procedure. The number of levels in *RNS* is unknown but necessarily finite as we see next.

$RNS(G, P, r_{max})$, which takes in input a signed graph G, an optimal partition P and a maximum edit distance r_{max}, is detailed in Algorithm 6. It starts with the initial partition P (line 1) to enumerate a set of its neighbor optimal partitions up to edit distance r_{max} (line 4), denoted by $RN_{\leq r_{max}}(P)$. Each partition P^t associated with $\pi^t \in N_r(\pi^s)$ is considered as a potential initial partition for seeking new neighbor partitions (line 8). Despite the pruning techniques which avoids some repetitions, it is possible that P^t has been already discovered and belongs to $RN_{\leq r_{max}}(P)$ (see such an example in Figure 5.3b). Line 9 checks this case, and discards P^t, if required. This process is repeated in a recursive manner, until there is no new partition obtained.

In the rest of this work, we leave out the common parameter G and P^s (or P) from the mentioned methods for the sake of convenience: $CoNS(r)$ and $RNS(r_{max})$.

Algorithm 6: $RNS(G, P, r_{max})$

Input : signed graph $G(V, E, w, s)$, partition P, maximum edit distance r_{max}
Output: $RN_{\leq r_{max}}(P)$

```
1  U = {P} // a set of unprocessed partitions
2  RN_{≤r_max}(P) = ∅ // a set of discovered partitions
3  while U ≠ ∅ do
4      for each r ∈ {1, .., r_max} do
5          Let P^s be an element of U, where π^s is an associated membership vector of P^s
6          RN_{≤r_max}(P) = RN_{≤r_max}(P) ⋃ P^s
7          N_r(π^s) = CoNS(G, π^s, r) // Complete Neighborhood Search
8          for π^t ∈ N_r(π^s) do // P^t is the associated partition of π^t
               /* memory-based duplication removal                    */
9              if P^t ∉ RN_{≤r_max}(P) then
10                 U = U ⋃ P^t
11             end
12         end
13     end
14 end
```

5.5 Pruning Strategies

In this section, we present the pruning strategies incorporated in $CoNS(G, \pi^s, r)$ presented in Section 5.4.

Before proceeding, some new definitions must be introduced. Consider an edit operation $\pi^s \rightarrow \pi^t$ with moving vertices in set $\overset{s\rightarrow t}{V}$, where membership vectors π^s (resp. π^t) is associated with source partition P^s (resp. target partition P^t). Let $\widetilde{G}[\overset{s\rightarrow t}{V}] = (\overset{s\rightarrow t}{V}, \widetilde{E}, w)$, where $\widetilde{E} = \{(u, v) \mid (u, v) \in E$ and $u, v \in \overset{s\rightarrow t}{V}$ and $(\pi^s(u) = \pi^s(v) \vee \pi^t(u) = \pi^s(v) \vee \pi^s(u) = \pi^t(v) \vee \pi^t(u) = \pi^t(v))\}$ and $w(e) = 1$ for each $e \in \widetilde{E}$, be the *interaction subgraph* defined for the edit operation $\pi^s \rightarrow \pi^t$. The extraction of a subgraph $\widetilde{G}[\overset{s\rightarrow t}{V}]$ from its original graph G is illustrated in Figure 5.5.

Moreover, let us define $\vec{V}^s_{\pi^s(u)} = \overset{s\rightarrow t}{V} \cap M^s_{\pi^s(u)}$ (resp. $\vec{V}^t_{\pi^s(u)} = \overset{s\rightarrow t}{V} \cap M^t_{\pi^s(u)}$) as the set of moving vertices being in the source module of vertex u in P^s (resp. in P^t). Also, let $\vec{V}^s_{\pi^t(u)} = \overset{s\rightarrow t}{V} \cap M^s_{\pi^t(u)}$ (resp. $\vec{V}^t_{\pi^t(u)} = \overset{s\rightarrow t}{V} \cap M^t_{\pi^t(u)}$) be the set of moving vertices being in the target module of u in P^s (resp. in P^t). Finally, we define $\overset{s\rightarrow t}{V}_u = (\vec{V}^s_{\pi^s(u)} \cup \vec{V}^t_{\pi^s(u)} \cup \vec{V}^t_{\pi^t(u)} \cup \vec{V}^s_{\pi^t(u)}) \setminus \{u\}$ as the set of moving vertices having edges with vertex u in $\widetilde{G}[\overset{s\rightarrow t}{V}]$. To illustrate the aforementioned notations related to $\overset{s\rightarrow t}{V}$, we consider again the same example in Figure 5.2. For $v_1 \in \overset{s\rightarrow t}{V}$, $\pi^s(v_1) = \{1\}$ and $\pi^t(v_1) = \{3\}$. Then, we have $\vec{V}^s_{\pi^s(v_1)} = \{v_1, v_2\}$ as the set of vertices in M^s_1 and $\vec{V}^t_{\pi^t(v_1)} = \{v_1\}$ since v_1 is the only moving vertex belonging to M^t_3. Also, we have $\vec{V}^s_{\pi^t(v_1)} = \vec{V}^s_{\{3\}} = \emptyset$ since there is no moving vertex in M^s_3 and $\vec{V}^t_{\pi^s(v_1)} = \{v_3\}$ since $v_3 \in M^t_1$. Finally, $\overset{s\rightarrow t}{V}_{v_1} = \{v_2, v_3\}$.

(a) Source partition P^s.

(b) Target partition P^t.

Figure 5.4 – Illustrative example for Property 5.2.a, where vertices in $\overrightarrow{V_a}$ (resp. $\overrightarrow{V_b}$) move from their source module M_a^s (resp. M_b^s) (Fig. 5.4.a) into target module M_b^s (resp. M_c^s, which is purposely not shown) (Fig. 5.4.b). $\overline{V}_a$ and $\overline{V}_b$ represent the non-moving vertices.

5.5.1 Non-Minimum Edit Operation Pruning

When generating the set $N_r(\pi^s)$ in RNS, only considering min-edit operations is sufficient. Next, we define a property allowing to detect, in some cases, that an edit operation is not minimum. The idea behind it is to define two sufficient conditions on the set of moving vertices implying r not to be minimum.

In the following, let $\pi^s \to \pi^t$ be an edit operation with moving vertices in $\overrightarrow{V}^{s\to t}$.

Property 5.2 (non-min-edit operation)**.** *Let $\overrightarrow{V_a} \subseteq \overrightarrow{V}^{s\to t}$ be a subset of $\overrightarrow{V}^{s\to t}$. Assume that the vertices in $\overrightarrow{V_a}$ constitute the majority part of module M_a^s (i.e. $|\overrightarrow{V_a}| > |M_a^s \setminus \overrightarrow{V_a}|$) and they are in the same target module M_b^t in P^t (i.e. $|\pi^t(\overrightarrow{V_a})| = 1$). Moreover, suppose there exists a subset of moving vertices $\overrightarrow{V_b} \subseteq \overrightarrow{V}^{s\to t}$, such that $\emptyset \subseteq \overrightarrow{V_b} \subseteq M_b^s$. Let us define $\overline{V}_a$ (resp. $\overline{V}_b$) as $M_a^s \setminus \overrightarrow{V}^{s\to t}$ (resp. $M_b^s \setminus \overrightarrow{V}^{s\to t}$) in P^s. Each condition in the following is a sufficient condition for $\pi^s \to \pi^t$ to be a non-min-edit operation:*

(a) If $\overrightarrow{V_b}$ does not move into M_a^s and $|\overrightarrow{V_a}| > |\overline{V}_a| + |\overline{V}_b|$, then $\pi^s \to \pi^t$ is not a min-edit operation.

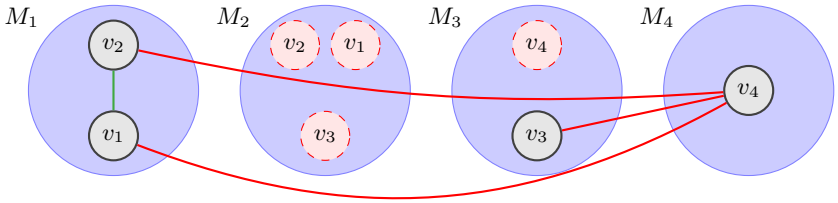
(b) If $\overrightarrow{V_b}$ moves into M_a^s and $|\overrightarrow{V_a}| + |\overrightarrow{V_b}| > |\overline{V}_a| + |\overline{V}_b|$, then $\pi^s \to \pi^t$ is not a min-edit operation.

Proof. Let us first handle the case of *(a)*, which is illustrated in Figure 5.4. When condition (a) is satisfied, keeping $\overrightarrow{V_a}$ in M_a^s and rather moving $\overline{V}_a$ (resp. $\overline{V}_b$) into M_b^s (resp. M_a^s) results in a r'-edit operation with $r' < r$. In the case that condition *(b)* is satisfied, keeping $\overrightarrow{V_b}$ in M_b^s while moving $\overline{V}_a$ and $\overline{V}_b$ results in a r'-edit operation with $r' < r$. □

5.5.2 Decomposable Edit Operation

The next set of properties is related to the decomposability of a r-edit operation $\pi^s \to \pi^t$. In these properties, we assume $I(P^s) = I(P^t)$ which is in line with our objective of enumerating all optimal solutions of a given graph G.

Definition 5.3 (Decomposable edit operation)**.** *We call a r-edit operation $\pi^s \to \pi^t$ decomposable if there is an intermediate membership vector $\pi^{t'}$, associated with a partition $P^{t'}$, between π^s and π^t such that:*



(a) An illustration of a r-edit operation $\pi^s \to \pi^t$ with $\overset{s\to t}{V} = \{v_1, v_2, v_3, v_4\}$. The membership vector π^s associated with partition P^s is defined by (1,1,3,4) while π^t by (2,2,2,3).

(b) Corresponding interaction graph $\widetilde{G}[\overset{s\to t}{V}]$.

Figure 5.5 – Illustrative example of interaction subgraph $\widetilde{G}[\overset{s\to t}{V}]$ definition.

1. $I(P^s) = I(P^t) = I(P^{t'})$;
2. $\overset{s\to t'}{V} \subset \overset{s\to t}{V}$; *and*
3. $\pi^{t'}(u) = \pi^t(u)$ *for each* $u \in \overset{s\to t'}{V}$.

Property 5.4 (atomic edit operation)**.** *If a r-edit operation $\pi^s \to \pi^t$ is atomic, then $I(P^s) < I(P^{t'})$ for all r'-edit operation $P^s \to P^{t'}$ with $r' < r$, satisfying (2) and (3) in Definition 5.3.*

Proof. We assume there is such a r'-edit operation with $I(P^s) = I(P^{t'})$. Then, we can construct a r''-edit operation $\pi^{t'} \to \pi^t$ satisfying (1) in Definition 5.3 and $\pi^{t'}(u) = \pi^t(u)$ for each $u \in \overset{s\to t}{V} \setminus \overset{s\to t'}{V}$. □

Atomicity imposes three types of connectivity conditions. The first one is stated in the following property.

Property 5.5 (Edge connectivity)**.** *In an atomic edit operation $\pi^s \to \pi^t$, $G[\overset{s\to t}{V}]$ is a single connected component.*

Proof. Let $\overset{s\to t'}{V}$ and $\overset{t'\to t}{V}$ be any two proper subsets of $\overset{s\to t}{V}$, such that $\overset{s\to t'}{V} \cap \overset{t'\to t}{V} = \emptyset$ and $E(\overset{s\to t'}{V}, \overset{t'\to t}{V}) = \emptyset$. Then, we can construct with $\overset{s\to t'}{V}$ a r'-edit operation satisfying Definition 5.3. □

Nevertheless, being connected in $G[\overset{s\to t}{V}]$ is not a sufficient condition. The next property states that the signs of edges between moving vertices play a key role in the atomicity. In the next results, we assume that $G[\overset{s\to t}{V}]$ is connected.

Property 5.6 (Fake edge connectivity)**.** *Each condition in the following is a sufficient condition for $\pi^s \to \pi^t$ to be a decomposable edit operation:*

(a) Let $S \subseteq \overset{s\to t}{V}$ be a subset such that $|\pi^s(S)| = 1$ and $E(S, \vec{V}^s_{\pi^s(S)}) \neq \emptyset$. If $\Omega(S, \vec{V}^s_{\pi^s(S)}) = 0$ and the graph $G' = (\overset{s\to t}{V}, E', w)$, where $E' = E \setminus E(S, \vec{V}^s_{\pi^s(S)})$ and $w(e) = 1$ for each $e \in E'$, is not connected, then $\pi^s \to \pi^t$ is decomposable.

(b) *Let* $S \subseteq \overset{s\rightarrow t}{V}$ *be a subset such that* $|\pi^t(S)| = 1$ *and* $E(S, \vec{V}^t_{\pi^t(S)}) \neq \emptyset$. *If* $\Omega(S, \vec{V}^t_{\pi^t(S)}) = 0$ *and* $G' = (\overset{s\rightarrow t}{V}, E', w)$, *where* $E' = E \setminus E(S, \vec{V}^t_{\pi^t(S)})$ *and* $w(e) = 1$ *for each* $e \in E'$, *is not connected, then* $\pi^s \rightarrow \pi^t$ *is decomposable.*

Proof. Straightforwardly, we can imagine S as a contracted vertex v in both cases. Since $\Omega(S, \vec{V}^s_{\pi^s(S)}) = 0$ ($\Omega(S, \vec{V}^t_{\pi^t(S)}) = 0$), the edges between v and $\vec{V}^s_{\pi^s(v)}$ (resp. $\vec{V}^t_{\pi^t(v)}$) do not play a role (i.e., as if they do not exist). Therefore, the proof is the same as in Property 5.5 with subsets S and $\overset{s\rightarrow t}{V} \setminus S$. □

The last connectivity condition allows to verify if a moving vertex depends on the simultaneous movement of a subset of other vertices in $\overset{s\rightarrow t}{V}$. It can be checked through the interaction subgraph $\widetilde{G}[\overset{s\rightarrow t}{V}]$, illustrated in Figure 5.5.

Property 5.7 (Interaction connectivity)**.** *Consider an atomic edit operation* $\pi^s \rightarrow \pi^t$. *The interaction subgraph* $\widetilde{G}[\overset{s\rightarrow t}{V}]$ *is a single connected component.*

Proof. Assume that $\widetilde{G}[\overset{s\rightarrow t}{V}]$ is not connected (e.g. Figure 5.5). This implies that there is a subset $\overset{s\rightarrow t'}{V} \subseteq \overset{s\rightarrow t}{V}$, such that $\widetilde{E}(\overset{s\rightarrow t'}{V}, \overset{s\rightarrow t}{V} \setminus \overset{s\rightarrow t'}{V}) = \emptyset$. Let $\pi^s \rightarrow \pi^{t'}$ be an edit operation with moving vertex set $\overset{s\rightarrow t'}{V}$, which satisfies (2)-(3) in Definition 5.3. When moving a vertex $u \in \overset{s\rightarrow t'}{V}$ from $M^s_{\pi^s(u)}$ to $M^s_{\pi^{t'}(u)}$ the contribution of vertex u to the imbalance is

$$\frac{\Omega(\{u\}, \vec{V}^s_{\pi^s(u)}) - \Omega(\{u\}, \vec{V}^s_{\pi^{t'}(u)}) + \Omega(\{u\}, \vec{V}^{t'}_{\pi^{t'}(u)}) - \Omega(\{u\}, \vec{V}^{t'}_{\pi^s(u)})}{2}. \tag{5.4}$$

From the definition of $\widetilde{E}$ and from $\widetilde{E}(\overset{s\rightarrow t'}{V}, \overset{s\rightarrow t}{V} \setminus \overset{s\rightarrow t'}{V}) = \emptyset$, there is no vertex $v \in \overset{s\rightarrow t}{V} \setminus \overset{s\rightarrow t'}{V}$ which contributes to the contribution of vertex u. Therefore, (1) in Definition 5.3 is also satisfied for $\pi^s \rightarrow \pi^{t'}$. □

For the sake of simplicity in Algorithm 4, we define the function *intAtomic*(G, π^s, $\overset{s\rightarrow t}{V}$) which is used to indicate whether $\pi^s \rightarrow \pi^t$ satisfies Properties 5.5 and 5.6.a. Notice that this is applied when the target modules of $\overset{s\rightarrow t}{V}$ are not known. Likewise, in Algorithm 5, we define the function *extAtomic*(G, π^s, π^t, $\overset{s\rightarrow t}{V}$), which indicates whether $\pi^s \rightarrow \pi^t$ satisfies Properties 5.2, 5.4, 5.6.b and 5.7. Notice that this is applied when the target modules of $\overset{s\rightarrow t}{V}$ are partially or completely known, i.e., in lines 10, 20 and 26 of Algorithm 3).

5.5.3 Multiple Vertex Moves between Optima (MVMO) Property

In the previous properties, P^s and P^t, associated with π^s and π^t, are not necessarily optimal. In this section, we introduce our last family of properties which assumes that both partitions are optimal in a edit operation.

Property 5.8 (MVMO on weighted signed networks)**.** *Consider an atomic edit operation $\pi^s \to \pi^t$ with $r > 1$, where P^s (and P^t) is optimal. The following condition must be satisfied, for each $u \in \overset{s\to t}{V}$:*

$$\gamma_u^{left} > \gamma_u^{right}, \tag{5.5}$$

where

$$\gamma_u^{left} = \Omega(\{u\}, \vec{V}^s_{\pi^s(u)}) - \Omega(\{u\}, \vec{V}^s_{\pi^t(u)}), \tag{5.6}$$
$$\gamma_u^{right} = -\Omega(\{u\}, \vec{V}^t_{\pi^t(u)}) + \Omega(\{u\}, \vec{V}^t_{\pi^s(u)}). \tag{5.7}$$
$$\tag{5.8}$$

Proof. First, we recall a simple condition satisfied by any optimal partition [43]. Given an optimal partition P, the placement of any vertex u is optimal, i.e., $\Omega(\{u\}, M^s_{\pi^s(u)}) \geq \Omega(\{u\}, M^s_{\pi^t(u)})$. Now, let $\pi^s \to \pi^t$ be an atomic min-r-edit operation. If $r > 1$, this condition becomes $\Omega(\{u\}, M^s_{\pi^s(u)}) > \Omega(\{u\}, M^s_{\pi^t(u)})$, for each $u \in \overset{s\to t}{V}$. Clearly, the atomicity assumption with $r > 1$ is contradicted whenever this condition is satisfied with equality. Next, in order to take into account the existence of other moving vertices in the source and target modules of vertex u, the last equation can be rewritten as $\gamma_u^{left} > \overset{s\to t}{\Delta}_u$, where $\overset{s\to t}{\Delta}_u = \Omega(\{u\}, M^s_{\pi^t(u)} \setminus \vec{V}^s_{\pi^t(u)}) - \Omega(\{u\}, M^s_{\pi^s(u)} \setminus \vec{V}^s_{\pi^s(u)})$. Similarly, considering the atomic min-r-edit operation $P^t \to P^s$, we have also $-\overset{t\to s}{\Delta}_u > \gamma_u^{right}$, where $\overset{t\to s}{\Delta}_u = \Omega(\{u\}, M^t_{\pi^s(u)} \setminus \vec{V}^t_{\pi^s(u)}) - \Omega(\{u\}, M^t_{\pi^t(u)} \setminus \vec{V}^t_{\pi^t(u)})$. Since both $\overset{s\to t}{\Delta}_u$ and $\overset{t\to s}{\Delta}_u$ are defined on relations of vertex u with non-moving vertices in P^s and P^t, we have $\overset{s\to t}{\Delta}_u = -\overset{t\to s}{\Delta}_u$. Finally, by rewriting the previous equations we obtain $\gamma_u^{left} > \overset{s\to t}{\Delta}_u = -\overset{t\to s}{\Delta}_u > \gamma_u^{right}$. □

Corollary 5.9 (MVMO on unweighted signed networks)**.** *The inequality $\gamma_u^{left} - \gamma_u^{right} \geq 2$ must be satisfied for each vertex $u \in \overset{s\to t}{V}$, on top of Property 5.8.*

Proof. The proof is straightforward from the proof of Property 5.8. □

Figure 5.6 can be used to illustrate Property 5.8 and Corollary 5.9. For instance, for vertex v_1, the equation in Property 5.8 becomes

$$a_{12} - a_{13} > a_{15} - a_{12} - a_{14}. \tag{5.9}$$

Similarly, for vertex v_3, the equation in Property 5.8 becomes

$$-a_{34} - a_{35} > a_{13} + a_{23} + a_{34}. \tag{5.10}$$

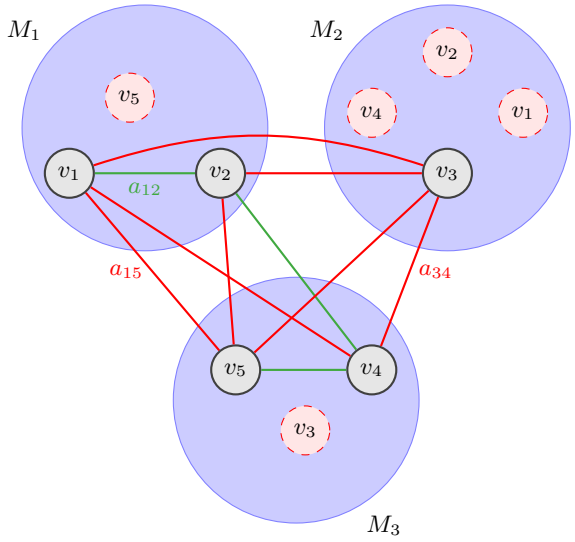


Figure 5.6 – Illustrative example, where five vertices v_1, v_2, v_3, v_4 and v_5 moves from their source modules into target ones. Dashed circles indicate the moving vertices, when they are in their target modules. Note that non-moving vertices are not drawn.

5.5.4 Tractable cases of the MVMO Property

Notice that the MVMO property requires π^s and π^t to be known, which makes this property not very useful for computational purposes (in Algorithm 3). In this section, we analyze the MVMO property when only partial information of π^t is known (lines 10 and 20 in Algorithm 3).

We start by analysing some special relational patterns for 2-edit and 3-edit operations. Lemma 5.10 focuses on 2-edit operations. Recall that $\widetilde{E} = \{(u,v) \mid (u,v) \in E$ and $u,v \in \overset{s\to t}{V}$ and $(\pi^s(u) = \pi^s(v) \vee \pi^t(u) = \pi^s(v) \vee \pi^s(u) = \pi^t(v) \vee \pi^t(u) = \pi^t(v))\}$.

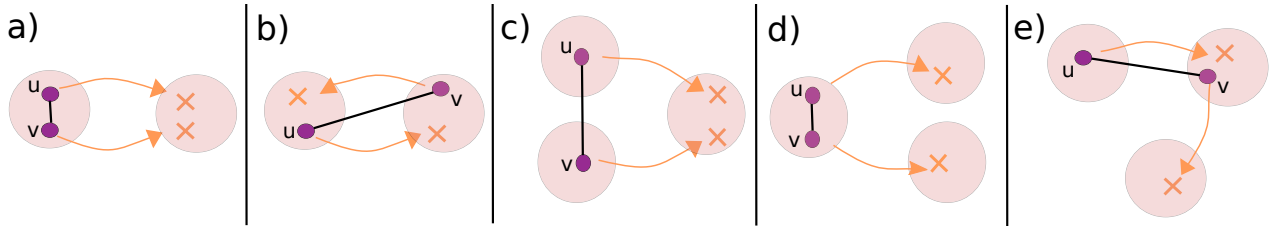


Figure 5.7 – All 2-edit operation scenarios, where $(u,v) \in \widetilde{E}$. Note that only scenarios (a) and (b) are atomic 2-edit operations, with $a_{uv} > 0$ and $a_{uv} < 0$, respectively.

Lemma 5.10 (MVMO 2-edit)**.** *Consider an atomic 2-edit operation $\pi^s \to \pi^t$, where $\overset{s\to t}{V} = \{u,v\}$. Since it is an atomic edit operation, we have $(u,v) \in \widetilde{E}$. Then*

1. *If $(\pi^s(u) = \pi^s(v)) \vee (\pi^t(u) = \pi^t(v)) = true$, then $a_{uv} > 0$.*
2. *If $(\pi^s(u) = \pi^t(v)) \vee (\pi^t(u) = \pi^s(v)) = true$, then $a_{uv} < 0$.*

Proof. From Property 5.5, we have $(u, v) \in E$. In this proof, we use an exhausting strategy: moving vertices u and v, with $(u, v) \in \widetilde{E}$, can be in one of the five scenarios presented in Figure 5.7. Since Corollary 5.9 must be satisfied for each vertex $u \in \overset{s\to t}{V}$, only scenarios (a) and (b) are atomic. Therefore, $a_{uv} > 0$ for scenario (a) whereas $a_{uv} < 0$ for scenario (b):

a) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = -a_{uv})$ and $(\gamma_v^{left} = a_{uv}) > (\gamma_v^{right} = -a_{uv})$. We see that a_{uv} must be positive. Since Corollary 5.9 is satisfied with $a_{uv} > 0$, it is atomic.

b) We have $(\gamma_u^{left} = -a_{uv}) > (\gamma_u^{right} = a_{uv})$ and $(\gamma_v^{left} = -a_{uv}) > (\gamma_v^{right} = a_{uv})$. We see that a_{uv} must be negative. Since Corollary 5.9 is satisfied with $a_{uv} < 0$, it is atomic.

c) We have $(\gamma_u^{left} = 0) > (\gamma_u^{right} = -a_{uv})$ and $(\gamma_v^{left} = 0) > (\gamma_v^{right} = -a_{uv})$. We see that a_{uv} must be positive and this satisfies Property 5.8. However, Corollary 5.9 is not satisfied, which makes this edit operation decomposable.

d) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = 0)$ and $(\gamma_v^{left} = a_{uv}) > (\gamma_v^{right} = 0)$. We see that a_{uv} must be positive and this satisfies Property 5.8. However, Corollary 5.9 is not satisfied, which makes this edit operation decomposable.

e) We have $(\gamma_u^{left} = -a_{uv}) > (\gamma_u^{right} = 0)$ and $(\gamma_v^{left} = 0) > (\gamma_v^{right} = -a_{uv})$. We see that a_{uv} must be negative and this satisfies Property 5.8. However, Corollary 5.9 is not satisfied, which makes this edit operation decomposable.

□

Next, we focus on 3-edit operations. We verify some cases in which Lemma 5.10 is also valid.

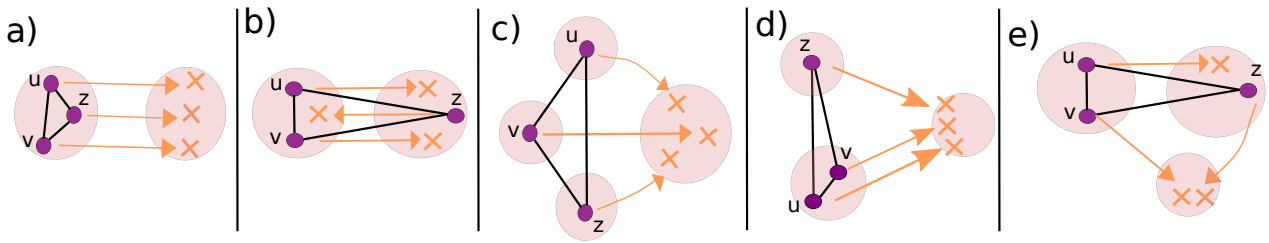


Figure 5.8 – Some of all possible 3-edit operation scenarios. The full list can be found in Appendix (Figure C.1). In this figure, all edit operations are atomic.

Lemma 5.11 (MVMO 3-edit)**.** *Consider an atomic r-edit operation $\pi^s \to \pi^t$ with $r = 3$, where $\overset{s\to t}{V} = \{u, v, z\}$. Since it is an atomic edit operation, we have $(u, v), (u, z), (v, z) \in \widetilde{E}$. Lemma 5.10 holds true for each pair (u, v) of vertices, with two exceptions:*

1. *all vertices in $\overset{s\to t}{V}$ are in the same source module and are moved into the same target module (Figure 5.8.a),*
2. *only two of $\overset{s\to t}{V}$ are in the same source module and are moved into the source module of the third moving vertex. Reciprocally, the third moving vertex moves into the source module of the others' source module (Figure 5.8.b).*

Proof. Without these two exceptions, $G[\overset{s\to t}{V}]$ forms a triangle. Nevertheless, in these two exceptions, $G[\overset{s\to t}{V}]$ can form a path, and, as we see next, this path ensures that moving vertices being in the same source module are positively connected. Similar to Lemma 5.10, the proof is based on all possible 17 scenarios of three moving vertices with $(u,v),(u,z),(v,z) \in \widetilde{E}$. Some of those scenarios are shown in Figure 5.8, and the full list is found in Appendix (Figure C.1). The proof is straightforward when one adapts Corollary 5.9 to those scenarios. We verify below only those in Figure 5.8. The full list of verification is found in Appendix (Section C.2).

a) We have $(\gamma_u^{left} = a_{uv} + a_{uz}) > (\gamma_u^{right} = -a_{uv} - a_{uz})$, $(\gamma_v^{left} = a_{uv} + a_{vz}) > (\gamma_v^{right} = -a_{uv} - a_{vz})$ and $(\gamma_z^{left} = a_{uz} + a_{vz}) > (\gamma_z^{right} = -a_{uz} - a_{vz})$. We see that a_{uv}, a_{uz} and a_{vz} cannot be negative. Since Corollary 5.9 is satisfied with a positive path formed in $G[\overset{s\to t}{V}]$ for each vertex $u \in \overset{s\to t}{V}$, it is atomic.

b) We have $(\gamma_u^{left} = a_{uv} - a_{uz}) > (\gamma_u^{right} = -a_{uv} + a_{uz})$, $(\gamma_v^{left} = a_{uv} - a_{vz}) > (\gamma_v^{right} = -a_{uv} + a_{vz})$ and $(\gamma_z^{left} = -a_{uz} - a_{vz}) > (\gamma_z^{right} = a_{uz} + a_{vz})$. We see that a_{uv} (resp. a_{uz} and a_{vz}) cannot be negative (resp. positive). Since Corollary 5.9 is satisfied with a path formed in $G[\overset{s\to t}{V}]$, it is atomic.

c) We have $(\gamma_u^{left} = 0) > (\gamma_u^{right} = -a_{uv} - a_{uz})$, $(\gamma_v^{left} = 0) > (\gamma_v^{right} = -a_{uv} - a_{vz})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = -a_{uz} - a_{vz})$. We see that a_{uv}, a_{uz} and a_{vz} must be positive. Since Corollary 5.9 is satisfied with a positive triangle formed in $G[\overset{s\to t}{V}]$, it is atomic.

d) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = -a_{uv} - a_{uz})$, $(\gamma_v^{left} = a_{uv}) > (\gamma_v^{right} = -a_{uv} - a_{vz})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = -a_{uz} - a_{vz})$. We see that a_{uv}, a_{uz} and a_{vz} must be positive. Since Corollary 5.9 is satisfied with a positive triangle formed in $G[\overset{s\to t}{V}]$, it is atomic.

e) We have $(\gamma_u^{left} = a_{uv} - a_{uz}) > (\gamma_u^{right} = 0)$, $(\gamma_v^{left} = a_{uv}) > (\gamma_v^{right} = -a_{vz})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = a_{uz} - a_{vz})$. We see that a_{uv} and a_{vz} (resp. a_{uz}) must be positive (resp. negative). Since Corollary 5.9 is satisfied with a triangle formed in $G[\overset{s\to t}{V}]$, it is atomic.

□

Unlike Lemma 5.11, we cannot completely generalize Lemma 5.10 for more than three moving vertices. Nevertheless, there are some circumstances, where Lemma 5.10 is still valid for a subset of $\overset{s\to t}{V}$, and this is formalized in Lemma 5.12.

Lemma 5.12 (MVMO r-edit)**.** *Consider an atomic r-edit operation $\pi^s \to \pi^t$ with $r \geq 4$ and a vertex $u \in \overset{s\to t}{V}$. If $2 \leq |u \cup \overset{s\to t}{V}_u| \leq 3$, Lemma 5.10 holds true for each pair (u,v) with $v \in \overset{s\to t}{V}_u$.*

Proof. The condition of $2 \leq |u \cup \overset{s\to t}{V}_u| \leq 3$ ensures that subset $u \cup \overset{s\to t}{V}_u$ represents one of the scenarios in 2- or 3-edit operation. We note that two exceptional scenarios mentioned in Lemma 5.11 are not of interest here, because such scenarios must necessarily satisfy the definition of atomic edit operation, hence $|u \cup \overset{s\to t}{V}_u| > 3$. □

For the sake of simplicity, in Algorithm 4, we define the function *intMVMO*(G, π^s, $\overset{s\rightarrow t}{V}$) which indicates whether $\pi^s \rightarrow \pi^t$ satisfies Lemmas 5.10.1 and 5.11, when the target modules of $\overset{s\rightarrow t}{V}$ are not known. Likewise, in Algorithm 5 we define the function *extMVMO*(G, π^s, π^t, $\overset{s\rightarrow t}{V}$), which indicates whether $\pi^s \rightarrow \pi^t$ satisfies Lemmas 5.10.2, 5.11 and 5.12, when the target modules of $\overset{s\rightarrow t}{V}$ are partially or completely known (10, 20 and 26 of Algorithm 3).

5.6 Experiments

We now assess the performances of the enumeration methods aiming to generate the space of optimal solutions for the CC problem. We first describe the datasets used in our experiments (Section 5.6.1). Then, we investigate the efficiency of the pruning conditions used in Algorithms 4 and 5 based on the MVMO property (Section 5.6.2). Next, we proceed to the evaluation of our method $EnumCC(r_{max})$, which includes all the pruning strategies used in the recurrent neighborhood search. We compare our method with $OneTreeCC()$, the best enumeration method available in the literature (Section 5.6.3). Finally, we perform an in-depth analysis for the instances, where the complete solution space is not achieved within the time limit of 12 hours (Section 5.6.4). Our source codes are publicly available [1].

5.6.1 Dataset

This section is dedicated to the description of the datasets used in our experiments. As mentioned in the introduction, in this chapter we focus on *unweighted* signed networks. Our experiments are conducted on two datasets of random signed networks generated through two network generation strategies presented in Section 2.4.

Dataset 5.1: We generate this dataset through the same random signed network generator used in *Dataset 1.1* and *Dataset 2.2* (Section 2.4), but with different parameter values. For complete unweighted signed networks, we generate 20 replications for parameter values $\ell_0 = 3$, $n \in \{32, 36, 40, 45, 50\}$ and $q_m = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. In these networks, the value of q_{neg} with the considered parameters is approximately 0.7. For incomplete unweighted signed networks with $d = \{0.25, 0.50\}$, we generate 20 replications for parameter values $\ell_0 = 3$, $n \in \{32, 36, 40\}$, $q_m = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ and $q_{neg} = \{0.3, 0.5, 0.7\}$. In total, we produce 600 and 1080 instances for complete and incomplete networks, respectively, which makes a total of 1680 instances. We use this dataset in Sections 5.6.3 and 5.6.4.

Dataset 5.2: We generate this dataset through the same random signed network generator used in *Dataset 2.3* (Section 2.4), but with different parameter values. We generate signed networks with

1. https://github.com/arinik9/EnumCC, https://github.com/arinik9/ExCC, https://github.com/arinik9/BenchmarkCC.

parameter values $n \in \{30, 40, 50, 60, 70, 90\}$, $d \in \{0.25, 1.00\}$ and $\ell_0 = \{2, 4, 6\}$. In total, we produce 214 and 184 instances for complete and incomplete networks, respectively, which makes a total of 398 instances. In each generated network, the associated optimal solution corresponds to the planted partition defined for the corresponding network without perturbation. We use this dataset only in Section 5.6.2.

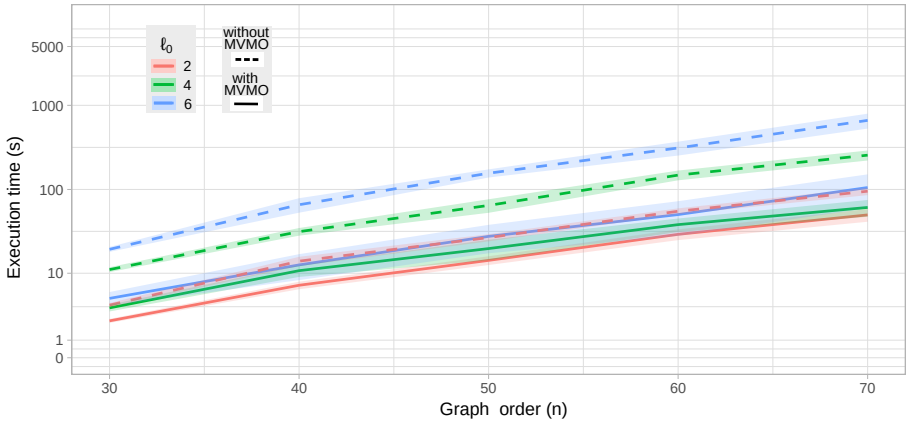
5.6.2 Evaluation of the MVMO-based pruning strategies

The MVMO property takes an important place in EnumCC due to its ability to prune unfeasible edit operations in several parts of $CoNS(r)$. To be able to consider relatively large graph orders n without being limited by long running time of an exact partitioning method we evaluate its performance based on *Dataset 5.2*.

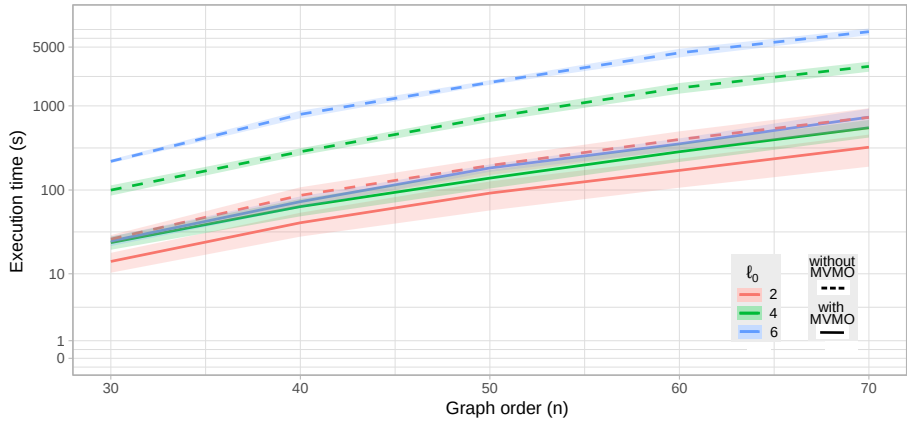
We apply $CoNS(r)$ *with* and *without* the MVMO property onto all generated networks with $r \in \{3, 4\}$. We mean by *with* MVMO property, the method as described in Section 5.3 and by *without*, the method obtained by removing this property from Algorithms 4 and 5. Due to space considerations, only results related to 4-edit are illustrated in Figure 5.9. In this figure, the results related to $d = 0.25$ and $d = 1.00$ are shown in separated subfigures. In each subfigure, the x-axis represents graph order n, and execution times (in seconds) are shown in the log-scaled y-axis. A plot line is solid (resp. dashed) when it corresponds to $CoNS(r)$ with (resp. without) the use of MVMO property. Each plot line in these subfigures corresponds to a specific value of ℓ_0 and is represented with a specific color. Each plot line includes a shaded colored region to depict a range of execution times based on the corresponding initial signed network and its perturbed versions.

By looking at the results, the first thing to notice is how graph density affects the execution times with increasing values of n. Indeed, the execution times with and without the MVMO property are much more larger with $d = 1.00$ than those with $d = 0.25$. Second, the execution times without the MVMO property are severely affected by increasing values of n, whereas including the MVMO property allows to better handle this drawback. Indeed, we observe based on $d = 0.25$ and 4-edit distance that average execution times without the MVMO property are approximately ℓ_0 times larger than those with the MVMO property. Finally, including the MVMO property also allows to better handle increasing values of ℓ_0. Indeed, when $n = 70$ and $d = 0.25$, the difference of average execution times between $\ell_0 = 2$ and $\ell_0 = 6$ without (resp. with) the MVMO property is 3.8s (resp. 0.6s) based on 3-edit and 595s (resp. 63s) based on 4-edit.

To conclude this part, the MVMO property makes a substantial improvement on $CoNS(r)$. This improvement is apparent even with small values of n. Note that, even a small improvement ($1s$ or $2s$) can make a clear difference in terms of execution time for a solution space with 100 or more solutions, since $CoNS(r)$ is repeated several times in our methods. Nevertheless, our results suggest that $CoNS(4)$ should not be used for computationally purposes, even with the MVMO property. For this reason, in the following we stick to $r_{max} = 3$ for $EnumCC(r_{max})$ and see if the improvements



(a) Instances with $d = 0.25$.



(b) Instances with $d = 1.00$.

Figure 5.9 – Benchmark results for *CoNS*$(r = 4)$ with vs. without MVMO-based pruning. Note that the y-axes are log-scaled.

gathered by the MVMO property and other pruning strategies can enable $EnumCC(3)$ to compete with $OneTreeCC()$.

5.6.3 Evaluation of EnumCC

We now compare the results of $EnumCC(3)$ against $OneTreeCC()$ based on *Dataset 5.1*. We run both methods with a time limit of 12 hours and a limit of $50,000$ on the number of solutions found. We first evaluate the results for several values of density d with a fixed value of n (Section 5.6.3.2), then pass to the evaluation of several values of n with a fixed value of d (Section 5.6.3.3).

Regarding the synthetic graphs, we present a selection of the most relevant results in Figures 5.10a to 5.11c, for $d \in \{0.25, 1.00\}$. We first describe these plots generically here, before interpreting them. In these figures there are 3 subfigures. Each subfigure corresponds to a spe-

cific value of q_m (hence, a specific parameter set), and displays the difference of execution times between $EnumCC(3)$ and $OneTreeCC()$ (i.e., $EnumCC(3)$ minus $OneTreeCC()$), represented on the log-scaled y-axis of the plots. When such difference takes a negative value, this means our proposed method runs faster than $OneTreeCC()$. The set of 20 graphs generated for each parameter set are indexed, in the x-axis. Finally, to guide our discussion, we show for each graph the maximal number of solutions found by the method(s) within a time limit and the number of jumps related to $EnumCC(3)$, i.e. $n_{jump}(EnumCC(3))$, where the latter is shown in parentheses.

5.6.3.1 Discussion for the number of solutions

In this section, we briefly discuss the number of optimal solutions obtained for different values of density in *Dataset 5.1*. The results are presented in Table 5.1.

We can summarize the results in three points. First, there is a unique optimal solution in 28% of the graph instances generated for $d = \{0.5, 1.0\}$ and 14% for $d = 0.25$. Indeed, we observe that the average and maximal number of solutions for $d = 0.25$ are much larger than those for $d = \{0.5, 1.0\}$. Second, for all values of density, when q_m increases, i.e. when we introduce more misplaced edges, multiple optimal solutions tend to be more and more frequent. Finally, for $d = \{0.25, 0.5\}$, increasing q_{neg} essentially results in the appearance of very large size of the solution space. In the random network generation, increasing q_{neg} with a low graph density is more likely to produce instances having a large number of vertices with only few positive edges. Then, these vertices are often placed at peripheral in the solutions, i.e. they can easily change their module from one solution to another. To conclude, an overwhelming number of instances has multiple solutions and they need to be enumerated efficiently.

Table 5.1 – Average and maximal number of optimal solutions. The latter is shown in parenthesis. These results are obtained by aggregating the instances by the values of n. N/A indicates that there is no available entry.

		$q_m = 0.1$	$q_m = 0.2$	$q_m = 0.3$	$q_m = 0.4$	$q_m = 0.5$	$q_m = 0.6$
$d = 0.25$	$q_{neg} = 0.3$	1.4 (10)	8.7 (243)	12.8 (181)	52.2 (2,036)	21.9 (615)	37.7 (635)
	$q_{neg} = 0.5$	24.8 (693)	133.4 (1,714)	249.7 (2,868)	546.4 (17,775)	318.4 (4,610)	292.7 (10,338)
	$q_{neg} = 0.7$	8,839.3 (50,000)	11,866 (50,000)	17,432 (50,000)	18,970 (50,000)	21,965 (50,000)	23,041 (50,000)
$d = 0.50$	$q_{neg} = 0.3$	N/A	1.2 (3)	1.9 (10)	2.2 (11)	2.6 (17)	3.1 (20)
	$q_{neg} = 0.5$	1.1 (4)	3.7 (28)	12.1 (107)	19.1 (145)	15.2 (69)	19.5 (232)
	$q_{neg} = 0.7$	10.2 (60)	99.7 (2,455)	325.2 (7,713)	417.7 (14,061)	1434.7 (50,000)	822.2 (36,871)
$d = 1.00$	$q_{neg} \approx 0.7$	1.1 (2)	8 (78)	20 (220)	36.3 (1,029)	24.5 (242)	29.5 (367)

5.6.3.2 Evaluation of EnumCC by density

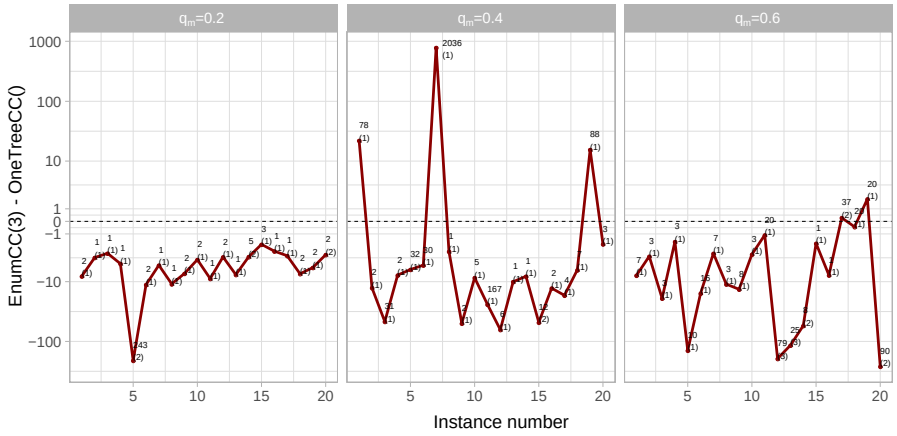
In this section, the results are based on $n = 36$ and $d \in \{0.25, 0.5, 1.0\}$. They are more or less representative for other values of n. We first consider the results with $d = 0.25$, shown in Figure 5.10. The first thing that we notice is how the results are affected by q_{neg}, independently of q_m. Indeed, we first see for $q_{neg} = 0.3$ that $EnumCC(3)$ runs faster than $OneTreeCC()$ in the overwhelming majority of instances. Then, for $q_{neg} = 0.5$, $OneTreeCC()$ increases the number of instances, where it runs faster, but the dominance of $EnumCC(3)$ is still preserved with a lesser extent. Finally, for $q_{neg} = 0.7$, $OneTreeCC()$ dominates $EnumCC(3)$ on almost all instances.

We observe that increasing q_{neg} essentially has three consequences, which advantage $OneTreeCC()$ over $EnumCC(3)$. The first one is a large value of $n_{jump}(EnumCC(3))$. Since the B&B tree needs to be built from scratch, each additional jump has an extra cost for $EnumCC(3)$ in terms of execution time. We observe that the values of $n_{jump}(EnumCC(3))$ are relatively larger for mainly $q_{neg} = 0.5$ and $q_m = \{0.4, 0.5, 0.6\}$. This shows that increasing q_m is likely to increase the dissimilarity between solutions, an aspect investigated in Chapter 6.

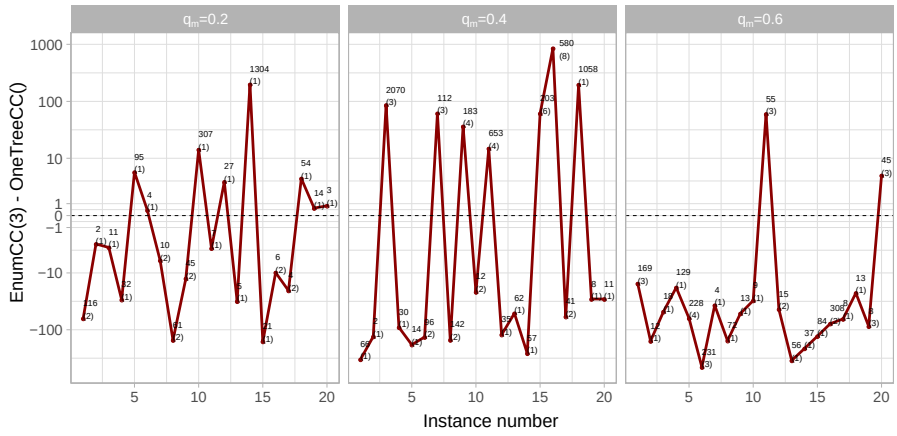
The second consequence is a very large size of the solution space. The results show that $OneTreeCC()$ does much better job in such case. Indeed, when $q_{neg} = 0.7$, $OneTreeCC()$ can solve instances associated with a very large number of solutions (e.g. $50,000$) within 5 minutes, whereas the same process often takes several hours for $EnumCC(3)$. Nevertheless, such an extreme case happens only with some specific graph topology, as in $q_{neg} = 0.7$. As mentioned in Section 5.6.3.1, these instances have a large number of vertices with only few positive edges. Then, these vertices can easily change their module from one solution to another. $OneTreeCC()$ seems to handle well these vertices in the enumeration process, thanks to the mathematical modeling behind it.

Finally, the third consequence, complementary to the previous one, is that instances having vertices with only few positive edges are often easier to solve, so that an initial optimal solution is often found at root relaxation, before passing to B&B. This usually results in a B&B tree with less branches for enumerating other optimal solutions in $OneTreeCC()$. It seems that this substantially advantages $OneTreeCC()$ over $EnumCC(3)$, when there are many optimal solutions to enumerate.

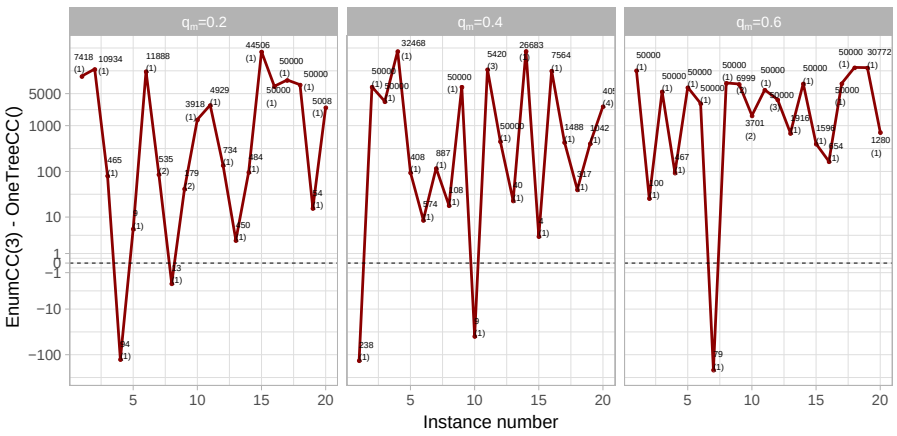
For space considerations, we do not present here the results for $d = \{0.5, 1.0\}$. Note that in these cases, we observe 3 such extreme instances with a very large number of solutions (as happens in $d = 0.25$ and $q_{neg} = 0.7$). This is probably because even though a vertex has only few positive edges, it has much more negative edges with large graph density, which does not allow to place it easily to other modules. This fact seems to advantage $EnumCC(3)$ over $OneTreeCC()$. Indeed, the dominance of $EnumCC(3)$ persists for $q_{neg} \in \{0.3, 0.5\}$ and even better than those for $d = 0.25$ (see Table 5.2). This fact is also true for $q_{neg} = 0.7$. $OneTreeCC()$ outperforms $EnumCC(3)$ only for $q_m = 0.1$, whereas this was true for all values of q_m for $d = 0.25$. These results confirm our previous observation: $OneTreeCC()$ outperforms $EnumCC(3)$ on instances with specific graph topology,



(a) Instances with $d = 0.25$ and $q_{neg} = 0.3$.



(b) Instances with $d = 0.25$ and $q_{neg} = 0.5$.



(c) Instances with $d = 0.25$ and $q_{neg} = 0.7$.

Figure 5.10 – Results based on $n = 36$, $d = 0.25$ and $q_{neg} = \{0.3, 0.5, 0.7\}$. Note that the y-axes are log-scaled. We show for each graph the maximal number of solutions found by the method(s) within a time limit and the number of jumps related to $EnumCC(3)$, i.e. $n_{jump}(EnumCC(3))$, where the latter is shown in parentheses.

where low density and large proportion of negative edges produce a very large number of solutions. In the other cases, $EnumCC(3)$ performs much better. Next, we see if increasing values of graph order n confirms these observations.

5.6.3.3 Evaluation of EnumCC by graph order

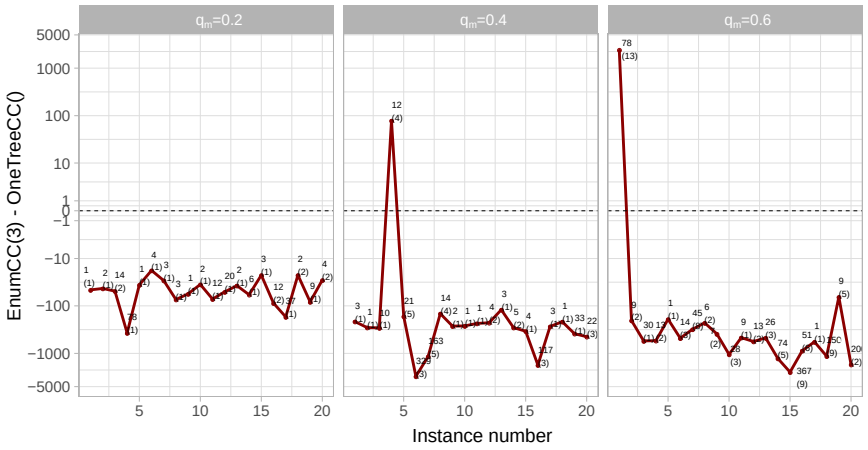
In this section, we analyze the effect of increasing values of graph order with $n \in \{40, 45, 50\}$. To do so, we focus only on instances with $d = 1.0$ in order to reduce the total number of instances, hence processing time of our analysis. We note that q_{neg} approximately equals 0.7 in these instances. We show the corresponding results in Figure 5.11. Overall, we see that our observations made for $n = 36$ and $d \in \{0.5, 1.0\}$ in Section 5.6.3.2 are still valid, when increasing n: $EnumCC(3)$ performs much more better. Furthermore, unlike the results with $d = 0.25$, $EnumCC(3)$ handles much better the instances with a large value of $n_{jump}(EnumCC(3))$ (when we exclude some exceptional instances with more than 10 jumps), and runs faster than $OneTreeCC()$ in most of the instances. This point is even more valid, with increasing values of n.

Another interesting point is the hardness of instances, when increasing values of n. We observe that solving instances takes much more time in this case, which often results in exceeding the time limit of 12 hours. A total of 21 instances were not solved within the time limit by both methods. These instances are identified as the ones at $y = 0$ (see Figure 5.11c). $EnumCC(3)$ handles these time-consuming instances relatively better than $OneTreeCC()$. Indeed, $EnumCC(3)$ could solve 7 instances that $OneTreeCC()$ could not within the limit of 12 hours. Moreover, among the 21 instances not solved by any method, $EnumCC(3)$ finds more solutions than $OneTreeCC()$ in 20 instances. These "*unsolved*" instances is analysed in a detail way in Section 5.6.4.

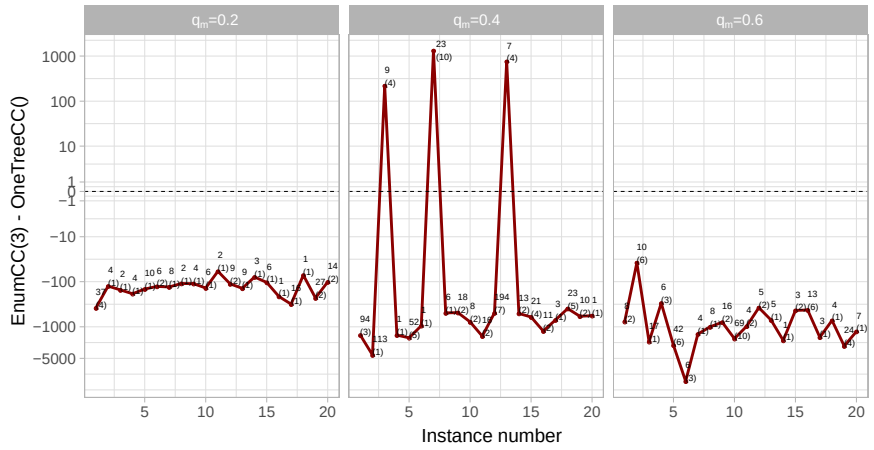
To summarize the evaluation of $EnumCC(3)$, there is not a single method which always gives the best results. On the one hand, $OneTreeCC()$ handles very well the instances with a very large solution space. This extreme case is associated with a specific graph topology, based on low graph density and large q_{neg}. On the other hand, $EnumCC(3)$ better performs in the rest of the instances, which constitutes the overwhelming majority of them. Indeed, when we summarize our results in Table 5.2 by showing the proportion of cases where $EnumCC(3)$ runs faster, this point is clearer.

5.6.4 Investigation on harder instances

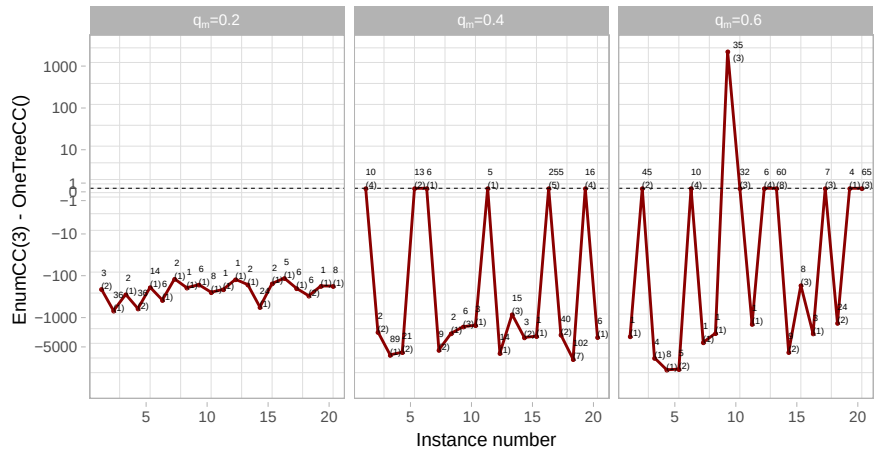
In this last section, we want to perform an in-depth analysis for the instances where the complete solution space is not achieved within the time limit of 12 hours. There are 21 such instances and each of them is with $n = 50$. The number of solutions obtained by $OneTreeCC()$ and $EnumCC(3)$ on these instances are summarized in the first two rows of Table 5.3. We further consider the $F_v^\star(G)$ formulation, i.e. removing the redundant triangle inequalities from $F_v(G)$, to see if it brings any improvement. These versions of both methods are denoted by $OneTreeCC+()$ and $EnumCC+(3)$



(a) Instances with $n = 40$ and $d = 1.00$.



(b) Instances with $n = 45$ and $d = 1.00$.



(c) Instances with $n = 50$ and $d = 1.00$.

Figure 5.11 – Results with $n = 40$, $n = 45$ and $n = 50$ based on $d = 1.00$. Note that the y-axes are log-scaled. We show for each graph the maximal number of solutions found by $EnumCC(3)$ within a time limit and the number of jump ofs $EnumCC(3)$, i.e. $n_{jump}(EnumCC(3))$, where the latter is shown in parentheses.

Table 5.2 – Summary regarding the proportion of the cases where EnumCC(3) is faster than OneTreeCC. These results are obtained by aggregating the instances by the values of n (excluding the instances, when in the margin of 5 seconds). N/A indicates that there is no available entry.

		$q_m = 0.1$	$q_m = 0.2$	$q_m = 0.3$	$q_m = 0.4$	$q_m = 0.5$	$q_m = 0.6$
$d = 0.25$	$q_{neg} = 0.3$	1.00	0.92	1.00	0.87	0.96	0.92
	$q_{neg} = 0.5$	0.00	0.52	0.77	0.61	0.61	0.86
	$q_{neg} = 0.7$	0.00	0.08	0.08	0.05	0.08	0.03
$d = 0.50$	$q_{neg} = 0.3$	N/A	1.00	1.00	1.00	1.00	1.00
	$q_{neg} = 0.5$	N/A	1.00	1.00	0.92	0.84	0.75
	$q_{neg} = 0.7$	0.11	0.80	0.95	0.90	0.89	0.87
$d = 1.00$	≈ 0.7	1.00	0.98	0.98	0.94	0.95	0.98

and their results are summarized in the last two rows of Table 5.3. Each column of the table represents an instance, identified by its id and its associated value of q_m. First, we want to compare the number of solutions obtained by $OneTreeCC()$ and $EnumCC(3)$, since both of them could not finish their optimization process. Second, we want to see if using the $F_v^\star(G)$ formulation, rather than $F_v(G)$, would be beneficial for both methods.

We first analyze the number of optimal solutions obtained by $OneTreeCC()$ and $EnumCC(3)$. When we focus on the instances where there are at least 2 solutions (i.e. 20 out of 21), we see that the number of solutions found by $OneTreeCC()$ is always smaller than those found by $EnumCC(3)$. Furthermore, by further investigation, we found out in 15 instances that $EnumCC(3)$ already enumerate all the solutions and it passes the remaining time by trying to ensure that the solution space is complete.

Table 5.3 – Number of optimal solutions found by the considered methods within the time limit of 12 hours on hard instances with $n = 50$. The method OneTreeCC+() (resp. EnumCC+(3)) corresponds to OneTreeCC() (resp. EnumCC(3)) without redundant triangle inequalities. Note that $\star$ is used for last two methods to indicate when a method could finish the resolution within the time limit of 12h.

instance no / methods	$q_m = 0.4$						$q_m = 0.5$							$q_m = 0.6$							
	1	5	6	11	16	19	2	11	12	13	15	18	20	2	6	10	12	13	17	19	20
OneTreeCC()	4	4	1	1	13	2	7	21	3	1	10	2	2	3	1	3	4	5	3	1	3
EnumCC(3)	10	13	6	5	255	16	217	44	8	1	115	46	13	45	10	32	6	60	7	4	65
OneTreeCC+()	7	2	2	1	33	5	4	14	3	1	10	5	1	1	3	9	4	17	$7^\star$	1	2
EnumCC+(3)	$10^\star$	$13^\star$	$6^\star$	$5^\star$	$255^\star$	$16^\star$	231	$45^\star$	$8^\star$	$1^\star$	$115^\star$	46	13	45	5	$32^\star$	$6^\star$	60	$7^\star$	$4^\star$	78

Finally, we now analyze the number of solutions obtained by $OneTreeCC+()$ and $EnumCC+(3)$ based on the same instances. On the one hand, we observe that using $F_v^\star(G)$ is very beneficial for $EnumCC+(3)$, since it could finish its solving process in 14 instances. The only drawback is that it generates less number of solutions for the 6^{th} instance of $q_m = 0.6$. On the other hand,

surprisingly, using $F_v^\star(G)$ is not usually beneficial for $OneTreeCC+()$, since $OneTreeCC+()$ finds more (resp. less) solutions than $OneTreeCC()$ in 9 (resp. 6) instances. Moreover, it could finish its solving process for only 1 instance.

To conclude this part, $EnumCC(3)$ outperforms $OneTreeCC()$ in terms of the number of solutions, when a time limit is applied. Furthermore, a further investigation on the use of $F_v^\star(G)$ for both methods reveals that removing redundant triangle inequalities is much more beneficial for $EnumCC(3)$ than $OneTreeCC()$. Nevertheless, an in-depth analysis on incomplete and complete signed networks is required before drawing any conclusion on this subject, which is out of scope for this current work.

5.7 Conclusion

For most clustering problems, due to their NP-hard nature, exact approaches do not scale well even when looking for a *single* optimal solution. In this chapter, we proposed an efficient enumeration method to retrieve the complete optimal solution space of the CC problem for a given signed graph. It combines an exhaustive enumeration strategy with neighborhoods from small to large sizes, designed for our problem. In our experiments, we first showed that a signed network can have a very large number of optimal solutions for the CC problem, e.g., $50,000$ and more. Further investigation indicates that this extreme case is associated with a specific graph topology, based on low graph density and large proportion of negative edges. Otherwise, multiple solutions can still exist, mostly for the graphs with considerably more imbalance. Furthermore, we also showed that our method $EnumCC(3)$ performs better than $OneTreeCC()$ in the overwhelming majority of the instances. Nevertheless, there is not a single method which always gives the best results. $OneTreeCC()$ also handles very well the instances with a very large solution space. We conclude that it is more appropriate to use $OneTreeCC()$ in this specific graph topology, whereas $EnumCC(3)$ is more suitable for all the remaining cases.

We believe that this work opens new directions for future research. First, the most straightforward perspectives are to consider weighted signed graphs; and to take advantage of the optimal solutions found by heuristic methods in order to enumerate through an exact method only the undiscovered ones. For instance, when we combine the optimal solutions found by all the heuristics presented in Section 2.3 for complete networks with $n = 50$, we observe that they are able to generate more than half of the solution spaces in 61% of the cases. Second, one can wonder how different are the solutions of a given solution space. Application-wise, very similar solutions could be given the same interpretation, although a difference of one vertex could be important in a given application, if the vertex is central for instance. Nevertheless, structurally different solutions are much of interest [202], since they might correspond to dramatically different ways of seeing the studied system. We address this point in chapter 6. Third, one can study how robust the solution spaces are, when we slightly

introduce some perturbations into the corresponding networks. This would also enable to identify critical vertices when the underlying space of optimal solutions is changed. This could be done through either repeating the process from scratch for each perturbed signed graph, or based on the definition of stability range [176].

Chapter 6

Investigation of the space of optimal solutions for Correlation Clustering problem

6.1 Introduction

In chapter 5, we showed that it is possible to obtain many optimal solutions when solving the CC problem. Such multiplicity raises several questions. First, one can wonder how many of these solutions are equally relevant to the application problem at hand. Put differently, it is worth enumerating

all optimal solutions when solving CC for a given application? Perhaps it would be necessary to design a more appropriate version of the CC problem, in order to distinguish them, possibly based on some additional criteria related to the application context. Second, how different are these solutions? Application-wise, very similar solutions could be given the same interpretation, whereas substantially different ones might correspond to dramatically different ways of seeing the studied system. Third, when dissimilar solutions coexist for the same problem, is it possible to detect classes of similar solutions? Indeed, if such classes exist, one could need only to find one representative solution in each class, which would ease the exploration of the solution space. Fourth and finally, in case of the existence of multiple such classes, what distinguishes them from each other? Identifying these characteristic differences could provide some valuable information to understand the studied system. More generally, the answers to all these questions can drive the choice of the method used to solve CC.

In this chapter, our goal is to answer these questions through the characterization of the space of optimal solutions associated with the same collection of unweighted signed graphs generated in Chapter 5 (*Dataset 5.1* in Section 5.6.1). To this aim, we propose a cluster analysis-based framework, which allows us to study how nature of multiple solutions is affected by the network characteristics.

Contributions. The following content is based on our work published in *Journal of Complex Networks* [20]. This chapter makes the following contributions:

1. **Method.** we propose a cluster analysis-based framework to study the solution space, which is generic enough to be applied to other combinatorial problems.
2. **Evaluation.** We present extensive computational experiments established for the proposed method, relying on sparse and dense signed unweighted networks.

The rest of the chapter is organized as follows. In Section 6.2, we review the literature related to the enumeration and analysis of solution spaces containing multiple optimal solutions. In Section 6.3, we then justify our approach through a few simple examples. We turn to the methods in Section 6.4, and explain the approach we propose for the analysis of the space of optimal solutions for CC. In Section 6.5, we describe and discuss our results, in order to answer the questions asked above. Finally, in Section 6.6 we summarize our findings, comment the limitations of our work and describe how they could be overcome, and how our work can be extended.

6.2 Related Work

As we explain in Section 5.2, there is only a very limited number of methods proposed in the literature to solve CC *exactly*. Some of them, as well as subsequent works, identify the issue of

multiple optimal solutions, but only scratch the surface, as already summarized in Section 5.2.1. Indeed, once getting the complete set of optimal solutions for the considered instance, studying the space of optimal solutions requires to deal with additional methodological points, in particular: determining how similar or different they are. However, we could not find any work dealing with this for CC in the literature. For this reason, we widen the scope of our review on these aspects, and consider works conducted on other problems than CC. In Section 6.2.1, we first focus on the comparison of solutions; and finally in Section 6.2.2 we review works concerned with the diversity of these solutions.

6.2.1 Comparison Between Solutions

The works identifying the existence and relevance of multiple optimal solutions usually do not try to compare them, or only in terms of cost [63, 24]. From this perspective, the approach of Good et al. [101] is interesting, even if it deals with *sub-optimal* solutions, as it aims to compare the *nature* of these solutions. They study the *Modularity Maximization* problem, which consists in detecting a community structure in an unsigned graph, i.e. to partition it in order to get cohesive and well separated modules. They show that this problem admits an exponential number of distinct quasi-optimal solutions, and that moreover, these can be structurally very different, an issue they call *degeneracy*.

The connection with our own work is double. First, they deal with the partitioning of graphs, albeit unsigned ones. Second, we perform a similar comparison between graph partitions, with the difference that we focus only on *optimal* solutions. Such comparison is very important, as one can consider a given solution as a view or interpretation of the studied system. Therefore, in addition to identifying the multiplicity of optimal solutions, it is necessary to study *how much* they differ.

6.2.2 Diversity of Solutions

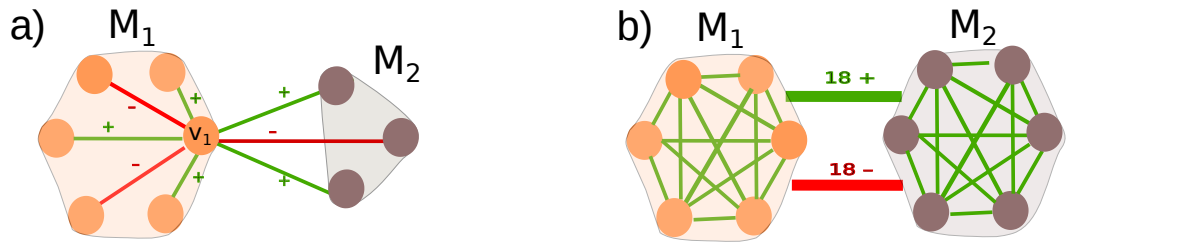
Enumerating all optimal solutions is costly, so one alternative is to discover only *certain* of them, often with some additional criterion of diversity (similar in principle to multi-objective optimization approaches). Appa [11] proposes an LP-based algorithm which, starting from an already-found optimal solution, finds an alternative optimal solution which is as different as possible. One can apply the method a number of times to sample the space of optimal solutions, and then select the most diverse ones. Danna et al. [54] do the same but for binary linear models. In the context of *Data Clustering*, some methods such as [119] have been proposed to detect multiple partitions, but these are not necessarily optimal.

In any case, the limitation of these methods is usually the estimation of the correct number of solutions: if it is underestimated, the solution space may not be sufficiently covered, whereas if it is overestimated, the computational cost stays high. Here, the connection with our work is the idea that

diversity is important when dealing with multiple optimal solutions. We explore this aspect through the notion of *classes of similar solutions*, which we define later in Section 6.4.

6.3 Illustrative Cases

In this section, we show with examples how structurally very similar or very different solutions can be formed. Figure 6.1a gives an example of the kind of situation that can lead to two very similar optimal solutions in complete signed graphs. Other examples exist in the literature for *incomplete* signed graphs, e.g. [56, 62]. Note that the graph in Figure 6.1a is fully connected, but only the edges attached to v_1 are represented, for matters of readability. The displayed bisection (i.e. modules M_1 and M_2) corresponds to an optimal solution. Consequently, the module assignment of v_1 is optimal as well. This implies that the signed sum of its *external* edges towards any module (currently, $+1$ for M_2) cannot be greater than that of its *internal* edges (currently, $+1$ for M_1). This case of equality between the internal and external edges means that moving v_1 to module M_2 instead of M_1 does not change the imbalance. Consequently, this change produces another optimal solution.



(a) Two structurally similar optimal solutions: 1) $v_1 \in M_1$, 2) $v_1 \in M_2$.

(b) Two structurally different optimal solutions: 1) a single module, 2) modules M_1 and M_2.

Figure 6.1 – Illustrative examples regarding optimal multiplicity for the CC problem.

This example shows how two similar optimal solutions can be obtained through a simple vertex change (see also Figures 5.1a and 5.1b). Of course, the same principle can be extended to larger changes involving more vertices (e.g. Figure 5.1c). Our point here is that it is relatively straightforward to explain the existence of multiple similar optimal solutions. However, it is equally easy to give examples of very different optimal solutions, as well. For instance, Figure 6.1b shows the case of a network constituted of two positive cliques, both connected by the same number of positive and negative edges. Again, the network is complete, but only the relevant edges are displayed. Solving the CC problem for this network yields a bisection whose modules M_1 and M_2 correspond to the positive cliques. But putting all the vertices in the same module is also an optimal solution: in both cases, the imbalance is 18. This example shows that structurally different optimal solutions can coexist for the CC problem.

In conclusion, we have shown that it is possible to obtain structurally very similar as well as very

different optimal solutions when solving the CC problem. However, we do not know whether these situations coexist in the same solution space, how frequent they are, or how this depends on the graph topology. These observations motivate us to adopt a more systematic approach for further investigations in the rest of this chapter.

6.4 Methods

In this section, we describe the method that we propose to analyze the space of optimal solutions for the CC problem. First, we need to clarify our terminology, as we handle various types of partitions. As mentioned before, the *optimal solution* (or *solution* for short) obtained by solving CC for a given graph is a partition of the vertex set that minimizes the imbalance measure. A subset of vertices in this partition is called a *module*. We reserve the term *clustering* to refer to a partition of the set of all solutions. A subset of solutions in such clustering is simply called a *solution class* (or a *class*, for short).

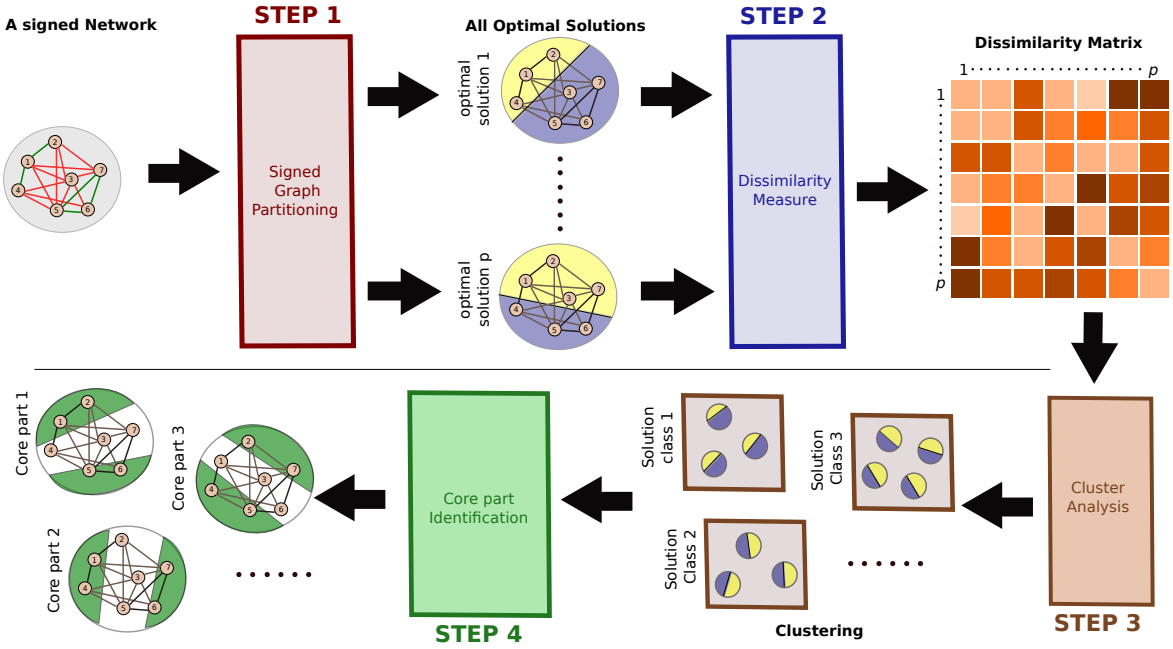


Figure 6.2 – Workflow proposed to study the solution space.

Our goal here is to determine whether it is worth enumerating all optimal solutions when solving CC for a given application. Put differently, we want to know what we lose when we consider only one solution, while there might be multiple ones. To this aim, we propose a 4-step pipeline approach which is represented in Figure 6.2. Each step allows answering a question that naturally arises in our analysis of the space of optimal solutions, and it is implemented through a well-known existing

tool deemed appropriate for this purpose. Our methodological contribution is found in the combination of these tools to build our pipeline. The input of the pipeline is a complete unweighted signed network. The first step is to enumerate all optimal solutions for this network (Section 6.4.1), allowing to determine whether *several optimal solutions coexist*. If so, the second step consists in computing the dissimilarity between them (Section 6.4.2), in order to assess *how different the obtained solutions are*. The third step consists in performing a cluster analysis of the solutions (Section 6.4.3), to check for *the existence of classes of similar solutions*. If there are several of them, the fourth and final step is to identify their *core parts* (Section 6.4.4), in order to *characterize them*. These cores correspond to the subset of vertices that stays constant, partition-wise, over all solutions constituting a class. Note that our workflow is relatively generic, in the sense that one could apply it to another optimization problem, provided steps 2 and 4 are adjusted to fit the nature of the solutions. In the rest of this section, we review the different steps of our framework in detail.

6.4.1 Enumerating All Optimal Solutions

The enumeration of all distinct optimal solutions can be very time- and memory-consuming, so we need an efficient method. We handle this step with two ILP-based methods, $OneTreeCC(G)$ and $EnumCC(G,3)$, presented in Chapter 5. According to their performances assessed in Chapter 5, it is preferable to use $OneTreeCC(G)$ on low density and highly negative signed networks, and $EnumCC(G,3)$ otherwise.

6.4.2 Computing the Dissimilarity Values

At this stage, we have identified all optimal solutions associated to the input graph. Let us denote as p the number of solutions found. We now want to gather similar solutions together. For this purpose, we perform a classic cluster analysis, which in turns requires the computation of a dissimilarity matrix. This matrix is obtained by comparing each pair of solutions. The literature contains a number of similarity or dissimilarity measures to perform such a task, each one possessing specific behavior and characteristics (Chapter 3).

We now use our results from Section 3.5.2 to solve the measure selection problem. The application of optimal solution space for the CC problem does not bring any specific constraint on the solutions. Nevertheless, it is less likely for an optimal solution space to have solutions with a large number of single-vertex modules, since positive edges between these modules would cause more imbalance. This can happen when the *Orthogonal Modules* and *Singleton Modules* transformations (Section 3.3.1.2) are applied with medium and large values of q. Therefore, these cases are not applicable in our context. Furthermore, the number of vertices across all the considered instances does not vary much. This means that n can be fixed in the framework and can therefore be ignored in our discussion.

Each application has its own desirable properties that an appropriate measure should satisfy regarding the application needs. In this application, the number and size of modules affected by a transformation are two important criteria to consider. This is because neighbor optimal solutions of a given optimal solution are usually reached by moving a few vertices. Additionally, a move of these vertices usually affects a few modules. When one of these two usual situations does not occur, we want the selected measure to be sensitive enough. This means that it is desirable for the selected measure to have the *Discriminativeness* property and not to satisfy the *Insensitivity to Module Size* property (Section 3.2.1.3).

Let us now study which measures studied in Section 3.4.1 fit the constraints described above. It appears from Table 3.3 that overall only D_{ARI} behave appropriately. In conclusion, we select D_{ARI} as a dissimilarity measure in this context.

6.4.3 Performing the Clustering

Next, we apply the k-medoids clustering method [128] to our dissimilarity matrix. As already explained in Section 4.3.3, this method partitions the dataset into k clusters, while minimizing the dissimilarity between the members of each cluster and some center of the cluster. It requires us to specify k, which we do not know in advance. In this situation, the standard approach is to use all possible values of k, from 2 to p (the number of optimal solutions), and assess the quality of the $p-1$ resulting clusterings through some internal criterion. As in Section 4.3.3, we use *Silhouette* for this purpose.

In theory, the k value associated with the highest Silhouette is the best candidate. However, in practice, one possibly has to set a threshold value large enough to ensure a reasonable cluster structure. Obtaining a Silhouette score above this threshold indicates that each cluster contains very similar solutions, and is at the same time clearly separated from the others. Otherwise, a Silhouette score below this threshold means that there is no cluster structure (i.e. a single cluster containing all solutions), or at least that the clustering is inconclusive. Kaufman & Rousseeuw recommend to use a threshold value of 0.51 (resp. 0.71) to get a reasonable (resp. strong) cluster structure [128]. Deciding the value of such a threshold can be considered either as an issue, as it can be a delicate operation, or an advantage, as it allows controlling the strength of the cluster structure. An alternative to determine the existence of a proper cluster structure is to use significance testing. However, using this type of test requires a certain number of observations, so this approach is not always applicable in practice (as in our case).

6.4.4 Identifying the Core Parts

At the end of the previous step, we obtain a collection of k clusters, each corresponding to a class of solutions that are, by construction relatively similar. We now want to assess how different

these classes can be. For this purpose, we leverage the concept that we call *core part*. The core part of a class is the maximal subset of vertices whose relative module assignment stays constant over all the solutions constituting the class. When two vertices belong to the core part, we call them *core vertices*, and they are either always in the same module, or always in different modules, for all the solutions of the class. Consequently, vertices that are always *isolated* (i.e. that constitute their own module) are core vertices, as their module assignment always differ from the rest of the core part. It is also possible to obtain an *overall core part* by proceeding similarly with all the solutions in the space (i.e. not focusing on a single class).

To identify the core part of a class, we rely on the idea of *consensus matrix* (a.k.a. co-association matrix) originating from Consensus Clustering [140]. The consensus matrix C of a class is an $n \times n$ matrix, whose entry C_{ij} indicates the number of solutions in which vertices v_i and v_j are assigned to the same module, divided by the total number of solutions constituting the solution class. Entries equal to one indicate vertices that are always assigned together to the same modules over all solutions.

6.5 Results

In this part (Sections 6.5.1 to 6.5.3), we investigate the space of optimal solutions of *Dataset 5.1* generated in Section 5.6. We present the results in the order that our workflow follows (see Section 6.4), since it is a pipeline.

Regarding the synthetic graphs, we present a selection of the most relevant results in Figures 6.5 and 6.6. Accordingly, we discuss only the results for $d = 0.5$, since the results for the other density values are similar. We point out the differences between d values, when applicable. Our source code is publicly available [1].

6.5.1 General remarks

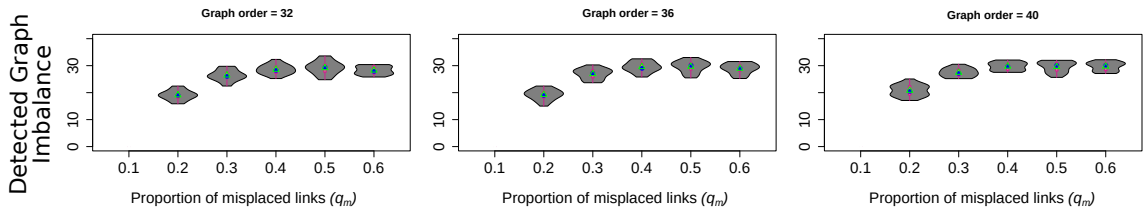
We first describe our plots generically here, for matters of convenience, before interpreting them. In these figures there are as many subfigures as the number of considered values of the parameter q_n. Each subfigure is a block of 3 (resp. 5) plots for $d \in \{0.25, 0.5\}$ (resp. $d = 1$), and focuses on a specific variable of interest, represented on the y-axis of the plots. The x-axis can either represent the *detected* graph imbalance $I(P)$ (e.g. Figures 6.5 and 6.6) or parameter q_m (e.g. Figures 6.3 and 6.4). Each plot in a subfigure corresponds to a different graph order n (number of vertices). The plots in Figure 6.5 represent the data as histograms, whereas the others contain violin plots, each one representing the results from 20 replications for the same parameter set. In each violin plot, the

1. https://github.com/arinik9/BenchmarkCC

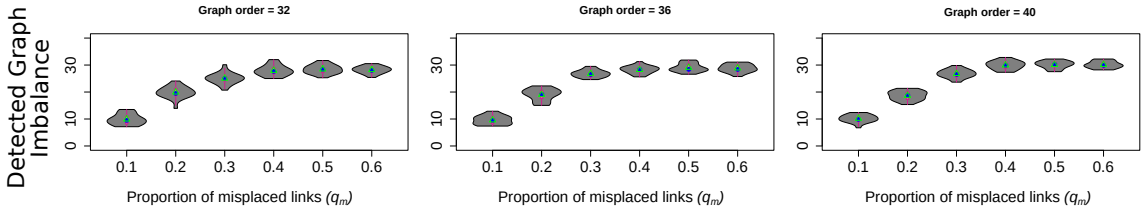
interquartile range is shown as a purple thick line, the mean as a green triangle and the median as a blue dot. In case of a unique value, only the mean and median appear.

To guide our discussion in the following, we plot the number of solutions presented in Table 5.1 with the same illustration scheme described above and they can be found in Appendix (Figures D.1, D.2 and D.3).

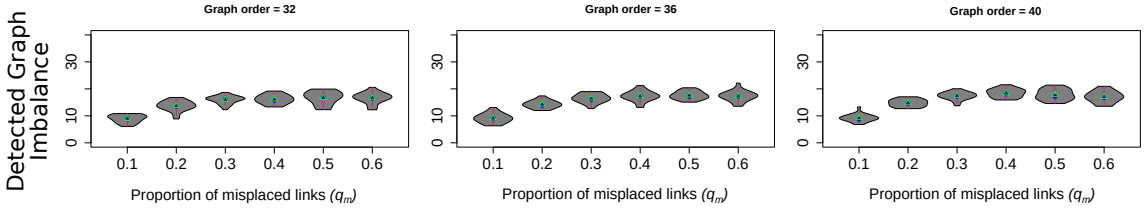
In the subsequent figures, we deem more suitable to present the results as a function of the graph imbalance and this choice is justified in Figure 6.3. For all d values, we observe that when q_m increases, $I(P)$ also increases for small q_m values, but then reaches a plateau. Yet, one would expect the imbalance to directly depend on the number of misplaced edges introduced in the graph. However, it turns out that when q_m exceeds some threshold, the number of misplaced edges (relative the initial partition) becomes so large that it provides some form of flexibility to graph partitioning. Consequently, even if these misplaced edges are randomly distributed, it becomes possible to partition the graph into a larger number of smaller modules allowing to reach a lower imbalance than expected (though still high), particularly when q_{neg} increases (see Figure 6.4).



(a) Imbalance percentage for $d = 0.5$ and $q_n = 0.3$.

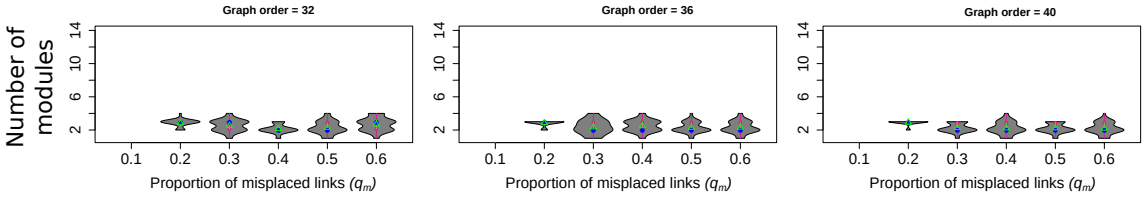


(b) Imbalance percentage for $d = 0.5$ and $q_n = 0.5$.

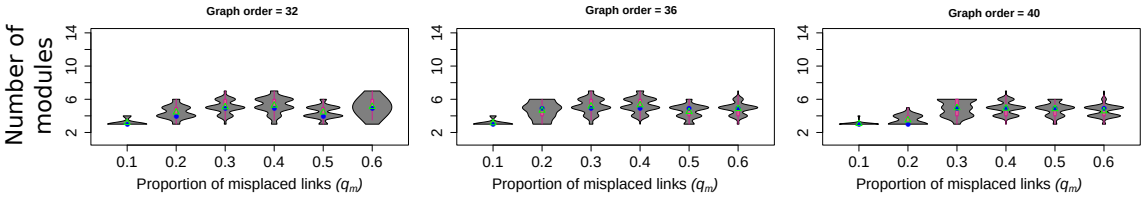


(c) Imbalance percentage for $d = 0.5$ and $q_n = 0.7$.

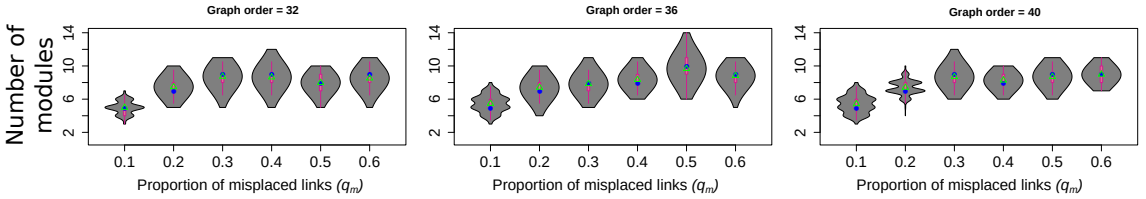
Figure 6.3 – Detected graph imbalance $I(P)$ as a function of q_m for $d = 0.5$, (a) for $q_{neg} = 0.3$, (b) for $q_{neg} = 0.5$ and (c) for $q_{neg} = 0.7$. Notice that an x-axis value may be empty if the parameter set is not defined or no data is available.



(a) Number of detected modules for $d = 0.5$ and $q_n = 0.3$.



(b) Number of detected modules for $d = 0.5$ and $q_n = 0.5$.



(c) Number of detected modules for $d = 0.5$ and $q_n = 0.7$.

Figure 6.4 – Number of detected modules as a function of q_m for $d = 0.5$, (a) for $q_{neg} = 0.3$, (b) for $q_{neg} = 0.5$ and (c) for $q_{neg} = 0.7$. Notice that an x-axis value may be empty if the parameter set is not defined or no data is available.

6.5.2 Diversity of the Solutions

Our first question is how different the obtained solutions are, in case of multiplicity. We answer it by analyzing the numbers of classes of solutions produced by our framework. Remember that, by construction, a class is a cluster of highly similar solutions. Figure 6.5 displays the proportions of cases for which there is a single solution class, as a function of the detected imbalance $I(P)$. Note that we do not include the instances for which there is only a *unique* optimal solution. This results in the absence of certain histogram bars in the plot.

It appears that different trends are associated with $q_{neg} = \{0.3, 0.5\}$ and $q_{neg} = 0.7$, respectively, for incomplete graphs. For $q_{neg} = \{0.3, 0.5\}$, our method usually detects a single class for slightly imbalanced graphs, and that the number of classes increases with the imbalance. There is an exception though, as the proportion of single class cases increases again for $I(P) = [0.30, 0.35[$ in graphs with $n = \{32, 36\}$. It seems that these cases could not reduce much their imbalance when

a large number of misplaced edges (relative the initial partition) is introduced, although this large number provides some form of flexibility to graph partitioning. In turn, this mostly results in a few similar solutions, hence a single class. Note that all these observations are also valid for complete graphs.

For $q_{neg} = 0.7$ on incomplete graphs, we usually observe the opposite trend. Our method frequently detects a relatively small proportion of a single class case for slightly imbalanced graphs, and that this proportion increases with the imbalance. This is surprising, as there can be much more optimal solutions when graph imbalance increases, but this could be explained by the concept of elongated class developed next. Overall, single class instances represent 53% of the cases for $d = 0.25$, 67% for $d = 0.5$ and 71% for $d = 1$.

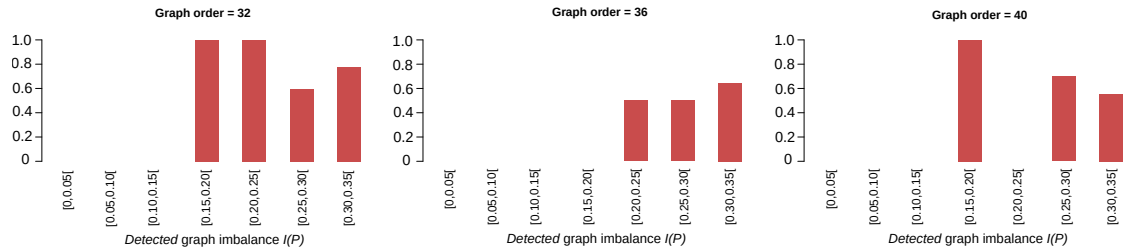
To summarize our findings up to now, complete graphs and incomplete ones with small imbalance and $q_{neg} = \{0.3, 0.5\}$ tend to lead to a unique solution, and even when there are several, these tend to constitute a single class (100% and 83% of the cases with a detected imbalance $I(P) \in [0.05, 0.15[$ for $d = \{0.5, 1.0\}$ and $d = 0.25$, respectively). Nevertheless, incomplete graphs with small imbalance and $q_{neg} = 0.7$ tend to lead to multiple solutions in most cases and they can exhibit multiple class structure in their solution spaces. All of these observations reveal the necessity of exploring further the solution space. Because, there is no absolute guarantee to get a single class structure for networks with a low imbalance and this depends on density and proportion of negative edges.

6.5.3 Analysis of the Core Parts

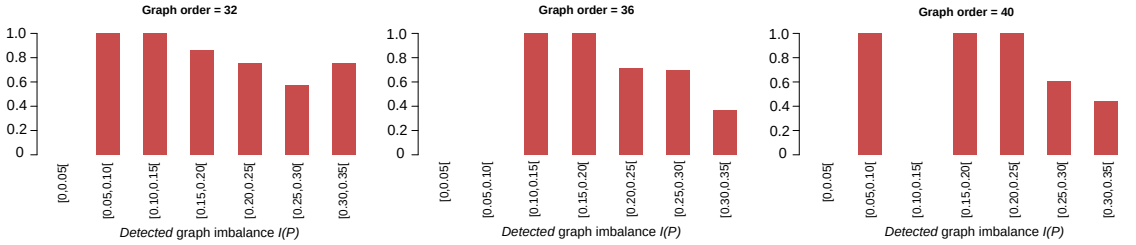
We now turn to the characterization and comparison of the classes, through the analysis of their core parts. As a reminder, the core part corresponds to the maximal subset of vertices that always belong to the same modules over all solutions constituting the class. We express the size of a core part in terms of proportion of the graph order n (number of vertices). Our assumption is that, for a class to be considered as cohesive, its core part (processed over the solutions of that class) should be large enough. Otherwise, it should be small.

Figure 6.6 shows the distribution of class core part sizes as a function of k, the number of solution classes (bottom x-axis). In addition, the values are grouped using the detected imbalance $I(P)$ (top x-axis of each plot). Like before, these plots do not show cases with only a unique solution.

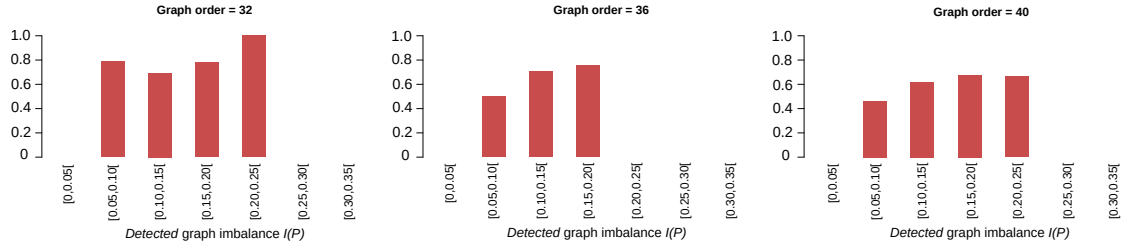
First, we observe that the core part size is always very large for $q_{neg} = 0.3$, independently of the number of classes k. This is mainly because there is only a few solutions obtained for the considered networks, where the maximal number is 17. Second, the range of core part sizes becomes much wider with increasing values of q_{neg}. This point is also true for $d = 1$, where $q_{neg} \approx 0.7$. Especially, in the single-class case, the core part size can be extremely small, close to zero. This indicates that, in certain cases, the cluster analysis is not conclusive: the Silhouette score is too low (below the threshold) to conclude there are several classes, but the single class is not cohesive, and contains



(a) Proportion of single-class cases as a function of the detected imbalance for $d = 0.5$ and $q_{neg} = 0.3$.



(b) Proportion of single-class cases as a function of the detected imbalance for $d = 0.5$ and $q_{neg} = 0.5$.

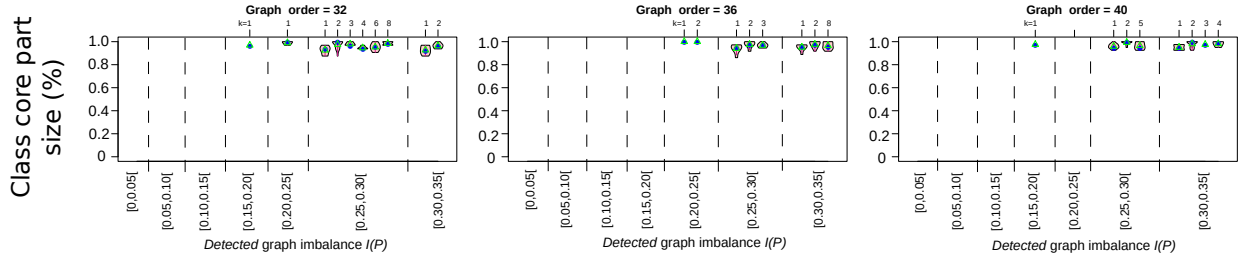


(c) Proportion of single-class cases as a function of the detected imbalance for $d = 0.5$ and $q_{neg} = 0.7$.

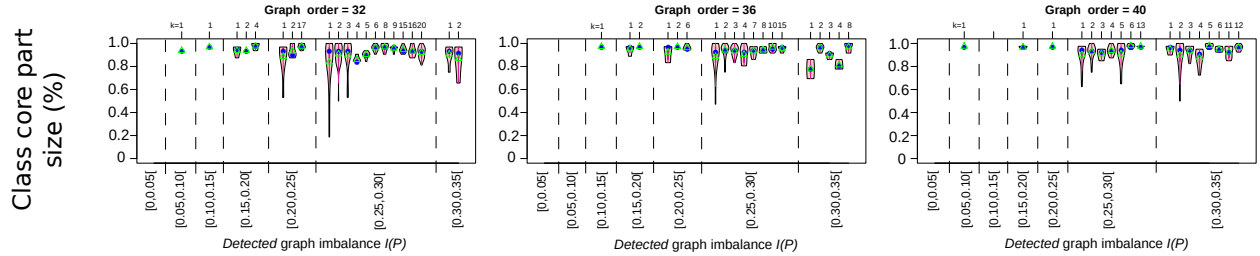
Figure 6.5 – Proportion of single-class cases as a function of the detected imbalance for $d = 0.5$, (a) for $q_{neg} = 0.3$, (b) for $q_{neg} = 0.5$ and (c) for $q_{neg} = 0.7$. Notice that an x-axis value may be empty if the parameter set is not defined or no data is available.

some sensibly different solutions. This can be explained by the concept of *elongated class*. If such a class is indeed a dense group of *locally* similar solutions, its most extreme members are nevertheless quite different. For a specific real-world application, one would need to manually consider this situation. Finally, the core part size for $q_{neg} = \{0.5, 0.7\}$ seems to increase with the number of classes k, at least until it reaches a plateau. This means that the classes are more and more cohesive internally. Moreover, the dispersion also decreases when k increases.

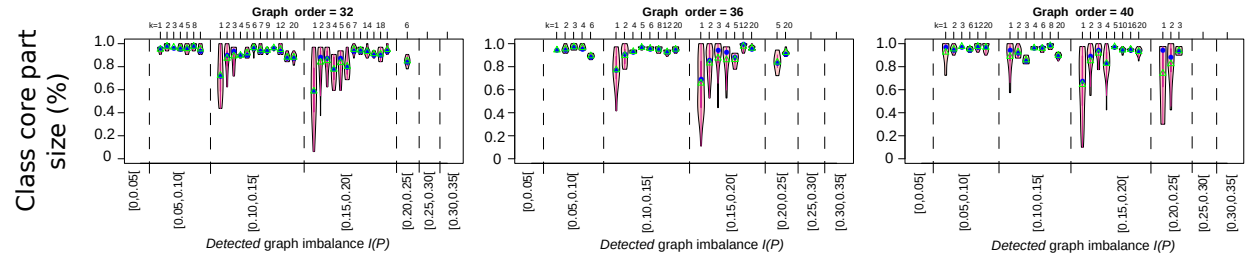
Let us now conclude this section related to synthetic networks. We empirically identified four different types of solution spaces. In the first, which tends to happen in only slightly imbalanced graphs, there is only one optimal solution. The second type corresponds to the case where there are multiple solutions distributed over several distinct and cohesive classes. This tends to happen for larger imbalance values. In the third type, we have a single class containing multiple solutions that are very similar, resulting in a large core. A small core means that this class is not cohesive,



(a) Proportion of the graph covered by the class core parts for $d = 0.5$ and $q_{neg} = 0.3$.



(b) Proportion of the graph covered by the class core parts for $d = 0.5$ and $q_{neg} = 0.5$.



(c) Proportion of the graph covered by the class core parts for $d = 0.5$ and $q_{neg} = 0.7$.

Figure 6.6 – Proportion of the graph covered by the class core parts for $d = 0.5$, as a function of the detected imbalance $I(P)$ and number of classes k, (a) for $q_{neg} = 0.3$, (b) for $q_{neg} = 0.5$ and (c) for $q_{neg} = 0.7$. Notice that an x-axis value may be empty if the parameter set is not defined or no data is available.

and corresponds to the fourth type. This typology shows that the answers to our initial questions are multiple and depend on the considered graph. Our work highlights the necessity to develop a method allowing to handle these different cases.

6.5.4 Real-World Example

To show the relevance of the questions at the origin of our work, as well as the usefulness of our method, we further analyze a small real-world graph representing the relations between the main actors of the ongoing Syrian conflict. We choose this dataset because of its size, which eases the interpretation of the obtained CC solutions, but also because it depicts a very interesting situation, as the affiliations of the involved parties are multiple: they are positioned relative to terrorist group ISIS, the Syrian government, and various other geopolitical interests.

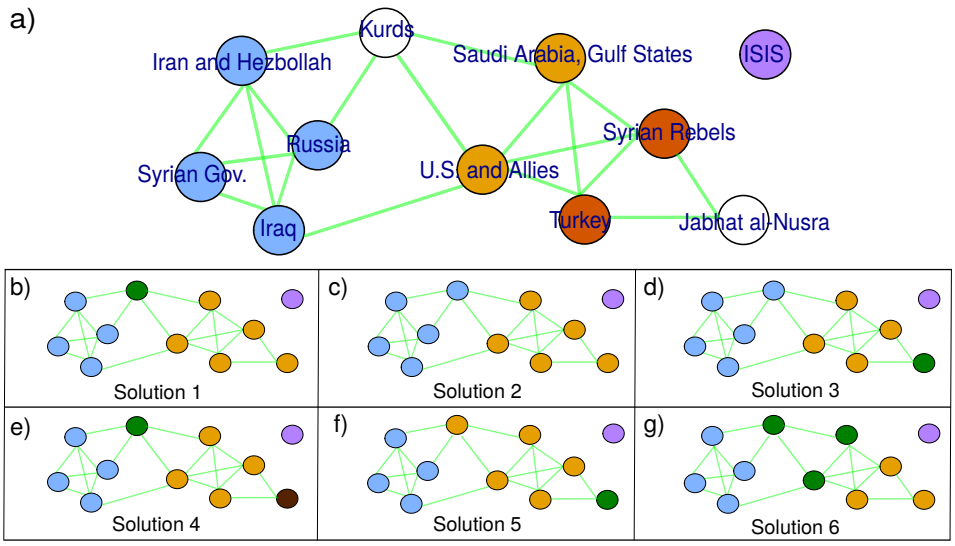


Figure 6.7 – *a)* Signed graph representing the Syria conflict in 2015. *b–g)* show optimal solutions. The graph is complete, but for the sake of clarity, only positive edges are shown (the missing ones thus represent negative edges). Colored vertices constitute the core part in *a)* (non-core vertices are white), and the module assignments for the other graphs.

Our source is the 2015 press article *A Guide to Who Is Fighting Whom in Syria* published by Keating & Kirk in the online news magazine Slate [2], which aims at depicting the Syrian situation as it was in 2015. This is a journalistic work, not an academic work, and it contains certain simplifications, for instance several distinct entities are collapsed together to ease understanding. However, we deem it sufficient in our case, as we are not Political Science or International Relations scholars ourselves, and do not intend at making a thorough political analysis of the situation, but simply use the article content for illustration purposes.

The article is constituted of a chart and its textual description. The chart is an update of the so-called *Middle East Friendship Chart*, which lists the actors of the conflict as well as the nature of their interactions: *enmity*, *friendship*, or *complicated*. The latter do not correspond to neutral relationships, but rather to undetermined ones, corresponding to a mix of hostile and friendly interactions. The associated text discusses this chart and explains the undetermined relationships. To build a signed graph based on this article, we interpret the chart as the adjacency matrix of a signed graph, representing the operating forces by vertices and their enmity and friendship relationships with negative and positive edges, respectively. Moreover, to keep the network complete and for the sake of illustration, we resolve undetermined relationships into hostile or friendly ones, by leveraging the analysis carried in the text. This results in the network presented in Figure 6.7.

We apply our framework to the Syria graph, like we did with the synthetic networks. Solving

2. www.slate.com/blogs/the_slatest/2015/10/06/syrian_conflict_relationships_explained.html

the CC problem yields 6 optimal solutions, each containing 7 frustrated edges (i.e. 12% of the edges). We describe and discuss each of them for the sake of completeness. In all the solutions, the actors are positioned relative to terrorist group ISIS, which is always detected as a single-vertex module. On top of that, the first to fifth solutions are bipolar: they contrast a pro- (Syrian government, Russia, Iran-Hezbollah, and Iraq) and an anti-Syrian government modules, whereas the module assignments of the rest of the actors (Kurds, Jabhat al-Nusra, and ISIS) are not consistent. The sixth (and last) solution is tripolar: the anti-Syrian government module is split in two: a pro-Kurds module (Kurds, U.S. and Allies, and Saudi Arabia-Gulf States) and an anti-Kurds one (Jabhat al-Nusra, Syrian rebels, and Turkey). Overall, the solution space shows that even for solutions differing only in the assignment of one vertex, the interpretation may change substantially (e.g. whether Kurds forms an alliance with the Syrian government or not). This confirms the necessity to explore the solution space.

When performing the cluster analysis over the space of optimal solutions, we obtain a maximal Silhouette score of 0.31 for $k = 2$, which corresponds to the two types of solutions identified manually above (1st–5th vs. 6th). Although this value is below Kaufman & Rousseeuw's 0.51 threshold (cf. Section 6.4.3), this clustering makes a lot of sense here, and shows that this threshold is not necessarily always relevant. The overall core part for these classes is represented by the vertex colors in Subfigures 6.7a: each color corresponds to a maximal group of vertices always assigned to the same module over all solutions. This highlights how core parts can be used to interpret the differences/similarities between the solution classes. Indeed, the figure reflects common knowledge regarding the geopolitical situation: the tight relationship between the USA and Saudi Arabia, Turkey supporting the Free Syrian Army to create a *buffer zone* in northern Syria from Kurds, and the disagreement between the USA and Turkey regarding Kurds. Interestingly, ISIS is a core vertex, as it is never placed with other vertices.

6.6 Conclusion

In this chapter, we empirically studied through our cluster analysis-based framework the space of optimal solutions for the CC problem based on the complete set of optimal solutions identified for *Dataset 5.1* in Section 5.6. Our main finding is the identification of 4 different situations: 1) unique solution; 2) single class of similar solutions; 3) several classes of similar solutions; 4) multiple solutions without a clear clustering structure. We also showed that both density and proportion of negative edges can affect the solution class structure. Indeed, a higher imbalance tend to lead to several classes for both complete graphs and incomplete graphs with $q_{neg} = \{0.3, 0.5\}$, whereas this multiple class structure is mostly obtained for slightly imbalanced incomplete networks with $q_{neg} = 0.7$. Finally, we illustrated the usefulness of our framework on a small real-world network.

Our work can be extended in several ways. First, the most straightforward perspectives are to

Table 6.1 – Average proportion of classes present in the solutions found by the heuristic methods presented in Section 2.3 for the instances possessing multiple solution classes on complete networks with $n = 50$, which makes in total 29 instances. Only stochastic heuristics which frequently find optimal solutions are considered. We run them 20 times. We observe that VNS always discovers all the solution classes for the considered instances.

Heuristics \ $I(P)$	$[0.15, 0.20[$	$[0.20, 0.25[$	$[0.25, 0.30[$
SA [173]	**1**	0.98	0.98
TS [37]	**1**	0.98	0.98
VNS [37]	**1**	**1**	**1**
GRASP [66]	**1**	0.82	0.39
ILS [148]	**1**	0.90	0.69
MLMSB [155]	**1**	0.92	0.86

apply our framework to weighted signed graphs; and to consider quasi-optimal solutions provided by heuristic methods (such as those presented in Section 2.3) for large graphs, following the example of Good et al. [101] with unsigned networks. Second, certain steps of our pipeline could be improved. The detection of single-class cases is not satisfying, as it can lead to undetermined situations. Maybe certain solution spaces do not have a crisp clustering structure, in which case a fuzzy clustering method could be more appropriate. Third, we plan to do a thorough investigation in order to determine whether core vertices possess certain specific topological properties compared to other vertices. Fourth, our results could be used to improve the search for optimal solutions. From a practical perspective, it is not possible to exhaustively enumerate all optimal solutions in large graphs. However, we could leverage the concept of class of similar solutions to design algorithms able to exploit a known optimal solution and find optimal solutions belonging to other classes. Such an exact approach would produce a set of diverse optimal solutions offering a better summary of the whole solution space than the traditional single optimal solution discussed in this paper. This approach can be also done through existing heuristics by slightly modifying them. For instance, when we analyze the optimal solutions found by the heuristics presented in Section 2.3, we observe that some of them give promising results (see Table 6.1). Fifth, we could work directly on the CC problem itself to reduce the number of optimal solutions. This can be done by optimizing a different imbalance measure (e.g. cycle- [39] or walk-based measures [71]), capable to discriminate between partitions otherwise considered optimal by the classic imbalance used in this chapter. It is also possible to add extra constraints in the problem formulation, e.g. by requiring modules to be internally connected in a stronger way (similarly to what is done in [31] for the clique partitioning problem).

Chapter 7

Conclusions and Perspectives

7.1 Conclusions

In this thesis, we tackled the notion of multiplicity in the partitioning of signed networks and could confirm its importance. In the literature, this concept is largely overlooked, because the standard approach is to find a single partition, as if finding it was sufficient to understand the considered system at hand. However, since a single partition is only one particular way of seeing the studied system, it is possible that one needs to seek for multiple partitions to get a better understanding, for interpretation purposes.

We relaxed this traditional single-partition assumption in two specific situations. The first one was in the context of signed multiplex networks. We proposed a new partitioning method which clusters partition-wise structurally similar layers of such a network and characterizes each obtained cluster through a consensual merging process, tailored for signed networks. By applying it to a European Parliament dataset, we could obtain multiple partitions, each corresponding to a different characteristic voting pattern of the same considered legislators. The emergence of such patterns was completely hidden when considering only traditional approaches. For instance, we could not only confirm that the French S&D and ALDE MEPs alternatively side with the left- and right-wing groups, but also identify which topics are concerned by these swings.

This need to look for multiplicity also holds even when considering a single (uniplex) network. Indeed, it has been pointed out, but not studied, in the literature that an instance of the CC problem can possess several *optimal* solutions. This type of multiplicity differs from the previous one in that the number of partitions is not tied to any user-defined parameter and depends solely on the topological features of a given network. This motivated us to study the space of optimal solutions of the CC problem as a second situation. Ideally, one should enumerate completely such space, and then proceeding confidently with the subsequent analysis. However, since this task needs to employ exact approaches to guarantee the completeness of the solution space, it can be very time-consuming. For this reason, we first designed a new efficient enumeration method, which includes a local search mechanism and several pruning strategies. Once the space of optimal solutions is completely enumerated, we then proceeded with its characterization through a cluster analysis-based framework. This allowed us to study how the nature of multiple solutions is affected by the network characteristics. Based on our empirical study on synthetic networks we first showed that it is possible, unlike

what has been observed in the literature, that a very large number of optimal partitions (e.g. 50,000) populates the solution space, and that this multiplicity mostly depends on graph density, proportion of negative edges and graph imbalance. Second, we could identify a typology of 4 different situations: 1) unique solution; 2) single class of similar solutions; 3) several classes of similar solutions; 4) multiple solutions without a clear clustering structure. Finally, we illustrated the usefulness of our characterization framework on a small real-world network.

In both situations described above, computing the similarity between partitions is a required task. In the context of graph partitioning, this task can be conducted through a so-called external evaluation measure, i.e. a measure originally designed to compare two partitions. However, there exist many such measures, each having different characteristics. This makes it challenging to select the most appropriate for a given situation for the end user. The widespread tendency among end users is to use popular measures, without considering much their relevance relative to the application at hand. This motivated us to solve this issue, and help the end user selecting an appropriate measure for their application. To this aim, we proposed a new empirical evaluation framework. For a collection of candidate measures, it first consists in describing their behavior by computing them for a generated dataset of partitions, obtained by applying a set of predefined parametric partition transformations. Second, our framework characterizes the measures in terms of how they are affected by these parameters and transformations. This allows both describing and comparing the measures. We illustrated its relevance by applying it to a selection of standard measures. Furthermore, when addressing each of the situations described above, our framework helped us select the most relevant measure.

7.2 Perspectives

The different investigations conducted in this work suggest diverse questions that, to the best of our knowledge, remain open. In the end of each chapter, we already discussed the most technical aspects related to these questions. In the following, we review the larger methodological points deserving some longer term efforts.

When evaluating our solution space enumeration method, we generated a collection of synthetic signed networks based on a model inspired by the Erdős-Rényi random graph model. Nevertheless, as showed by Orman et al. [179] for the Community Detection problem on unsigned graphs, including more realistic topological properties in such generative model can substantially affect the performances of certain resolution methods. It is likely that this holds when partitioning signed graphs, too. For this reason, an interesting research line would be to design a more realistic random model, reflecting the same topological properties as signed real-world networks. In the literature, there are some straightforward signed extensions of the community structure-generating LFR model (e.g. [69]) originally designed for unsigned networks, but it is not clear how realistic they are.

Designing a proper realistic model would require to review the existing random models, but also to analyze the characteristics of a large number of real-world signed networks, in order to identify their topological property, and embed them into the random model. Such a tool would allow to perform a more reliable evaluation of our own methods, but more generally it would be very beneficial to the research community, e.g. for assessing the performance of heuristics methods and estimate their robustness, even when looking for a single partition.

When evaluating our enumeration method of the solution space, we found out that the number of optimal partitions can be very large, even for small graphs. From an applications perspective, this poses serious problems in terms of interpretation. Either all obtained optimal partitions are equally relevant, in which case the application at hand has indeed many partitions of interest, or the problem is not well-defined for this application. In this case, it is necessary to perform adjustments to the problem itself in order to match the application needs. This can be done either by considering a related problem, such as RCC and others (e.g. [36, 32, 149]); or by coming up with a new problem definition with extra constraints allowing to reduce the solution space and focus only on relevant solutions. For instance, we have been recently experimenting with an extension of the CC problem that includes a two positive edge connectivity requirement at the module level, a work that led to an oral communication [18]. In any case, whatever the choice of the extended problem, we need to reapply our cluster-based characterization method to see if this would reach the same typology of solution space that we proposed during this thesis. More generally, it would be interesting to apply more systematically this characterization method onto a collection of extended versions of the CC problem. Hence, this would allow us to classify problems themselves depending on the shape of their spaces of optimal solutions.

Even when considering extensions of the CC problem for interpretation purposes, there can still be several structurally different optimal partitions in the solution space, depending on the application at hand. In this case, one still needs to explore this solution space, and as we saw in Chapter 5, this can be very costly computationally. To overcome this limitation, another interesting research line would be to infer some information related to the solution space (e.g. the number of solution classes) directly from the network characteristics, before applying an exact method. Recently, we started to experiment with machine learning techniques, especially based on graph embeddings, in order to predict the number of optimal partitions and that of solution classes directly from the extracted network characteristics. This is still an ongoing research and we believe that performing this task would be very beneficial for estimating in an instant way the answers of the research questions that we asked regarding the notion of multiplicity. In turn, these characteristics could then be used to drive the search for optimal solutions.

Appendices

Appendix A

Evaluation Measures

This appendix presents additional details and results, which were not included in Chapter 3. We first give the formal definition of the considered evaluation measures (Section A.1), then explain the additional experimental details mentioned in Section 3.3.1.1 about the heterogeneity of module sizes (Section A.2). Finally, we include two additional figures depicting the significance results regarding the comparison of the segment heights (Section A.3).

A.1 Definitions of evaluation measures

A common point of the evaluation measures is that they can be computed using the so-called *confusion matrix* (also called *association matrix* or *contingency table*) based on the two partitions.

We note n the numbers of vertices of a graph G. Also, let $P = \{M_1, ..., M_\ell\}$ $(1 \leq \ell \leq n)$ be a non-overlapping ℓ-partition of G. Let have another partition P' formed by ℓ' modules, where ℓ' may be different from ℓ. Then, the *confusion matrix* is a $\ell \times \ell'$ integer matrix, whose ii'th cell is the number of vertices in the intersection of modules M_i and $M_{i'}$, as shown in Table A.1.

Table A.1 – The confusion matrix for two partitions $P = \{M_1, ..., M_\ell\}$ and $P' = \{M'_1, ..., M'_{\ell'}\}$ of n vertices, where $n_{ij} = |M_i \cap M'_j|$ are the number of vertices in both modules $M_i \in P$ and $M'_i \in P'$.

		Partition P'			
	Module	M'_1	...	$M'_{\ell'}$	Marginal sum
Partition P	M_1	n_{11}	...	$n_{1\ell'}$	$n_{1\cdot}$
	·	·		·	·
	·	·		·	·
	M_ℓ	$n_{\ell 1}$	...	$n_{\ell\ell'}$	$n_{\ell\cdot}$
	Marginal sum	$n_{\cdot 1}$	...	$n_{\cdot\ell'}$	$n_{\cdot\cdot} = n$

A.1.1 Rand Index, *RI*

The formulation of all pair-counting measures can be expressed in terms of four types of element pairs. The *positive agreement* N_{11} corresponds to the number of element pairs which are in the

same module in *both* partitions P and P'. The *negative agreement* N_{00} is the number of element pairs which are in *different* modules in *both* P and P'. The partitions *disagree* on the remaining element pairs, as N_{10} (resp. N_{01}) corresponds to the number of element pairs which are in the same module in P (resp. P'), but not in P' (resp. P). The formula of each term is shown in Table A.2.

Table A.2 – Formulae for the number of (unordered) element pairs of the four types

Type	Formula
N_{11}	$\sum_{i=1}^{\ell}\sum_{j=1}^{\ell'}\binom{n_{ij}}{2} = \frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell'} n_{ij}(n_{ij}-1)$
N_{00}	$\binom{n}{2} - (N_{11}+N_{10}+N_{01})$
N_{10}	$\frac{1}{2}\Big(\sum_{i=1}^{\ell} n_{i\cdot}^2 - \sum_{i=1}^{\ell}\sum_{j=1}^{\ell'} n_{ij}^2\Big)$
N_{01}	$\frac{1}{2}\Big(\sum_{j=1}^{\ell'} n_{\cdot j}^2 - \sum_{i=1}^{\ell}\sum_{j=1}^{\ell'} n_{ij}^2\Big)$
$N_{\cdot\cdot} = N_{11} + N_{10} + N_{01} + N_{00}$	$\binom{n}{2} = n(n-1)/2$

The *Rand Index* (RI) [187] is the proportion of total agreement, i.e. when counting both positive and negative agreement:

$$RI(P,P') = \frac{N_{11}+N_{00}}{N_{\cdot\cdot}}. \tag{A.1}$$

Its values lie between 0 and 1, where 0 occurs for the absence of any positive and negative agreements, whereas 1 corresponds to the case where the partitions are perfectly identical.

A.1.2 Adjusted Rand Index, *ARI*

The *Adjusted Rand Index* (ARI) [114] is a well-known extension of the Rand Index, with additional correction for chance. It aims at dealing with the statistical independence of two partitions (see Section 3.2.1.2). Its formula is

$$ARI(P,P') = \frac{RI(P,P') - \mathbb{E}[RI(P,P')]}{1 - \mathbb{E}[RI(P,P')]}. \tag{A.2}$$

where $\mathbb{E}[RI(P,P')]$ corresponds to the estimated score of $RI(P,P')$ for independent partitions under hypergeometric assumption (so-called permutation model). This term is defined as

$$\mathbb{E}\Big(\sum_{ij}\binom{n_{ij}}{2}\Big) = \sum_{i}\binom{n_{i\cdot}}{2}\sum_{j}\binom{n_{\cdot j}}{2}\Big/\binom{n}{2}. \tag{A.3}$$

The ARI takes a value of 1 for identical partitions, whereas 0 indicates a case of statistical independence. Moreover, ARI can take a negative value for very dissimilar partitions [163], when the observed RI is smaller than expected.

A.1.3 Jaccard Index, *JI*

The Jaccard Index (JI) was originally defined to compare sets [118], but it is also used as an external measure [30]. As reported in [163], the negative agreement N_{00} can be often almost as large as the maximum number of element pairs $\binom{n}{2}$. The Jaccard Index is an improved version of RI on this aspect, as it does not take N_{00} into account. It is defined as

$$JI(P, P') = \frac{N_{11}}{N_{11} + N_{01} + N_{10}}. \tag{A.4}$$

The Jaccard Index ranges from 0 (absence of any positive agreement) to 1 (identical partitions). Note that one minus the Jaccard Index is a metric on the finite sets [159].

A.1.4 Fowlkes-Mallows Index, *FMI*

The Fowlkes-Mallows Index [94] is the final pair-counting measure that we consider in this work. It was originally introduced to ease the comparison of hierarchical dendrograms. Like the Jaccard Index, it ignores negative agreements. It can be described as the geometric mean of two asymmetric forms of positive agreement: the proportion of positive agreements relative to the number of pairs belonging to the same module in P vs. those in P'. Its formal description is

$$FM(P, P') = \frac{N_{11}}{\sqrt{(N_{11} + N_{10})(N_{11} + N_{01})}}. \tag{A.5}$$

A.1.5 F-measure, *F*

In the category of set-matching measures, we select the F-measure (F). Note that this name is sometimes used in the literature as a synonym of *harmonic mean*, and therefore covers several distinct measures (e.g. [186, 98]). We use the definition of Artiles et al. [25], according to which the F-measure is the harmonic mean of two quantities called *Purity* and *Inverse Purity*.

The formal definition of *Purity* is as follows:

$$Purity(P, P') = \sum_i \frac{n_{i\cdot}}{n} \max_j \frac{n_{ij}}{n_{i\cdot}}. \tag{A.6}$$

The Inverse Purity is simply the Purity of the second partition relative the first, i.e. $Purity(P', P)$.

Finally, the F-measure is the harmonic mean of the Purity and Inverse Purity

$$F(P, P') = 2\frac{Purity(P, P') \times Purity(P', P)}{Purity(P, P') + Purity(P', P)}. \tag{A.7}$$

A.1.6 Normalized Mutual Information, *NMI*

The last measure that we consider is the Normalized Mutual Information (NMI), which belongs to the category of information-theoretical measures. It is based on the notions of *entropy* and *Mutual Information* [50]. The principle behind these notions is to consider each partition as a categorical random variable, whose possible values are the modules.

In the context of clustering, entropy in the sense of Shannon is defined as

$$H(P) = -\sum_{i=1}^{\ell} \frac{n_{i\cdot}}{n} \log \frac{n_{i\cdot}}{n}. \tag{A.8}$$

Each vertex in G has an equal probability of being picked, so its probability of being in module M_i is n_i/n. Thus, we have a discrete random variable taking ℓ values, which is associated to the partition P. If the partition P has only 1 module containing all the points, then $H(P)$ will be zero, since there is no uncertainty in the clustering structure. If the partition P consists of as many modules as n, it will reach its maximum value. Note that $H(P)$ does not depend on n, but on the relative proportions of the modules.

The Mutual Information can be described as the mutual dependence between these variables, and it can then be interpreted as the similarity between the partitions. It is formally described as

$$MI(P, P') = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell'} \frac{n_{ij}}{n} \log \frac{\frac{n_{ij}}{n}}{\frac{n_{i\cdot}}{n}\frac{n_{\cdot j}}{n}}. \tag{A.9}$$

There are a number of variants of the notion of mutual information, in particular several normalizations have been proposed (see for instance [223]). In this work, we focus on the sum normalization as defined in [136, 205], which is very widespread. The resulting NMI is

$$NMI(P, P') = \frac{2MI}{H(P) + H(P')}. \tag{A.10}$$

A.2 Experimental details about the heterogeneity of module sizes

There are many ways to make modules imbalanced. In this work, we opt for a sequence based on an arithmetic progression. Consider the sizes of the modules in a partition as a sequence of values S_ℓ whose sum is equal to the number of nodes n, as in (A.11). In this equation, α corresponds to

the first value and β corresponds to the constant increment value

$$\begin{aligned} n &= \alpha + (\alpha + \beta) + (\alpha + 2\beta) + ... + (\alpha + (\ell - 1)\beta) \\ &= \alpha\ell + \frac{\beta\ell(\ell - 1)}{2}. \end{aligned} \tag{A.11}$$

Note that the sequence contains as many terms as the number of modules.

In such a sequence, each term is a constant increment value β larger than the previous term (e.g. $\beta = 2$ for the sequence 3, 5, 7, ..). This β is computed based on the parameter h (heterogeneity of module sizes). When $h = 1$, it reaches its maximal value that we note β_{max}. In the case of $h < 1$, β is proportional to β_{max} to the extent of h (i.e. $\beta = h \times \beta_{max}$). The value of β_{max} can be determined in different ways. In order not to introduce an additional parameter for this, our approach is to assign the first term α and the constant increment β_{max} to the same value, i.e. $\alpha = \beta_{max}$. Then, we compute β as follows

$$\begin{aligned} n &= \frac{\beta_{max}\ell(\ell + 1)}{2} \\ \beta &= \lfloor h\beta_{max} \rfloor. \end{aligned} \tag{A.12}$$

Note that $\lfloor . \rfloor$ denotes the floor function (returning the greatest integer less than or equal to the input value). Finally, we obtain the value of α as follows

$$\alpha = \frac{n - \frac{\beta\ell(\ell-1)}{2}}{\ell}. \tag{A.13}$$

A.3 Significance results regarding the comparison of the segment heights

This section presents two additional figures, which were not included in Section 3.5.2. Figures A.1 and A.2 visually report the significance results regarding the comparison of the segment heights performed in Section 3.5.2.

Figure A.1 – Significance of the results regarding the comparison of the segment heights performed in Section 3.5.2 over all pairs of transformations, considered for each measure and parameter set. For instance, the top four matrices correspond to Figure 3.4.a. Green (resp. red) cells represent significant (resp. non-significant) differences between the considered transformations, with a significance level of $\alpha = 0.05$. Figure available at 10.6084/m9.figshare.13109813 under CC-BY license.

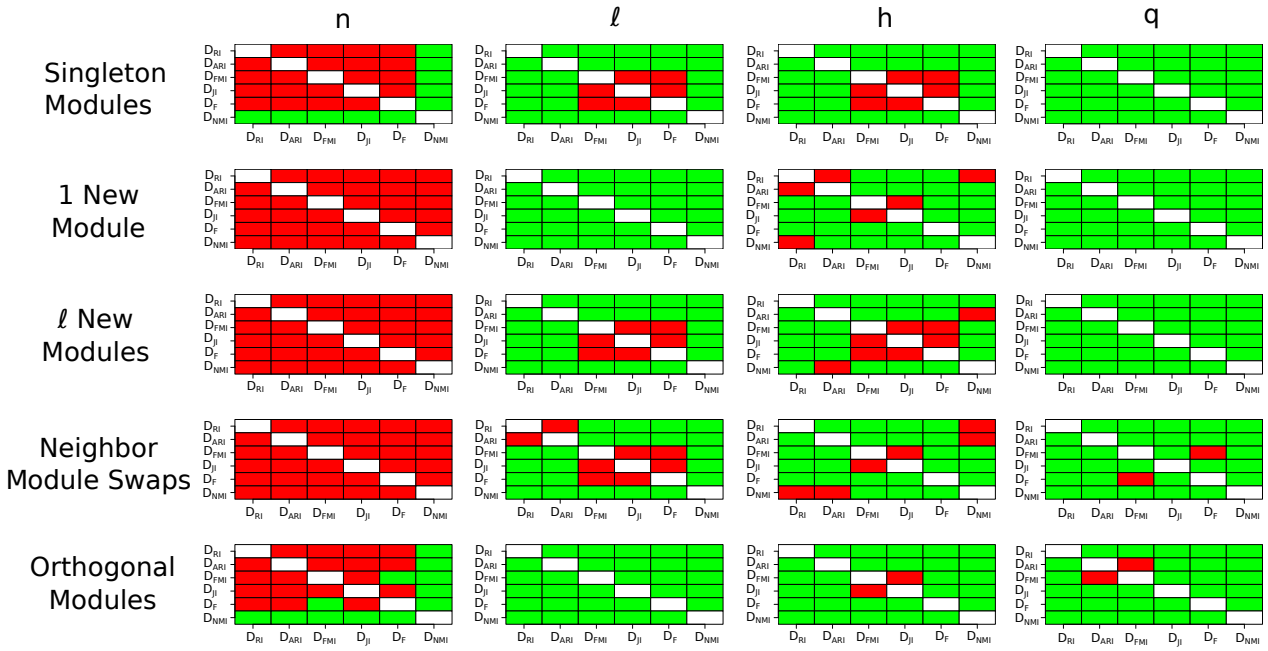


Figure A.2 – Significance of the results regarding the comparison of the segment heights performed in Section 3.5.2 over all pairs of measures, considered for each transformation and parameter set. For instance, the top four matrices correspond to the last stacked bar in each barplot of Figure 3.4 (*Singleton Modules*). Green (resp. red) cells represent significant (resp. non-significant) differences between the considered measures, with a significance level of $\alpha = 0.05$. Figure available at 10.6084/m9.figshare.13109813 under CC-BY license.

Appendix B

Common Agricultural Policy and Additional Agriculture-Related Results

This appendix presents additional details and results, which were not included in Chapter 4. We first give some context regarding the European Parliament and the Common Agricultural Policy (Section B.1) and the 2013 CAP Reforms (Section B.2), then present the hierarchy of topics related to Agriculture Policy Domain (Section B.3). Finally, we present additional plots, which were not included in Figure 4.3 (Section B.4).

B.1 EP- and CAP-Related Concepts

This section aims at providing the reader with some context regarding the European Parliament and the Common Agricultural Policy, through the definition of the main related concepts.

Common Agricultural Policy The CAP was established in 1957 to help EU farmers face agricultural challenges, which can be social (e.g. low labor force), economic (e.g. increasing inequalities, subsidies) or environmental (e.g. water pollution, global warming) [81].

The CAP is funded and shared by all the members state of the European Union. It currently takes action with income support (i.e. income stability through direct payments), market measures (i.e. dealing with difficult market situations such as severe market imbalance) and rural development measures (specific challenges facing rural areas).

Pillars I and II The CAP has two main principles called pillars in EU jargon [77]. The *Pillar I* is support to farmers' incomes. It is established through direct payments and market measures, and is entirely financed from the European Agricultural Guarantee Fund (EAGF). *Pillar II* is support to the development of rural areas. It is established through Rural Development programmes, and is co-financed by the European Agricultural Fund for Rural Development (EAFRD).

Direct payments Direct payments were introduced in 1992 to support the incomes of farmers [77]. Before that, the CAP was supporting the prices instead (an indirect form of payment).

Green payment A specific form of direct payment, granted for agricultural practices beneficial

for the climate and the environment. Member states must allocate 30% of their direct payment allocation to this greening payment [78]. After the 2013 CAP reform, the three greening obligations are: 1) Crop diversification, 2) Maintenance of permanent grassland, 3) Ecological focus area.

Co-financing scheme Member states have a certain discretion regarding the final design of Rural Development measures. The support granted under each measure is shared between the EU and the concerned member state. This arrangement is known as co-financing [77].

Milk quota system A quota system designed to control the production of dairy products in the EU. Its aim was to avoid overproduction. It was first established in 1983, after the occurrence of large volume of milk (and other dairy products) surplus, which had to be bought by European Commission [80].

Export subsidy Export subsidies (or *export refunds*) are special payments provided by the government to encourage export of goods, and to discourage sale of goods on the domestic market [77].

Cross-compliance scheme The cross-compliance scheme is a set of requirements (e.g. food safety, animal health, plant health) that farmers should satisfy in order to receive direct payments [77].

Permanent grassland It corresponds to a land used permanently (usually more than five consecutive years) [77].

Natura 2000 Natura 2000 is a network of nature protection areas in the European Union. The aim is to protect the most seriously threatened habitats and species across Europe [77].

Water Framework Directive (WFD) This legislation regarding water protection aim for the future [77].

B.2 Key Elements of the 2013 CAP Reforms

The 2013 CAP reform, which covers the 2014-2020 period, differed from the previous reforms, because it was the first time that the EP had a say in their adoption, by co-legislating with the Council of the EU. The reform process was launched by the EU commission in April 2010, through public debates and conferences. The final adoption of the legal texts took place on December 2013. Many of the regulations took effect in 2015, so that member states would have enough time to prepare.

The 2013 CAP reforms can be grouped in the following 4 different sets of regulations [81, 79, 160]:

- Rules for direct payments to farmers:
  - *Greening* of farm payments, through the introduction of environmental practices, such as crop diversification, and maintaining ecologically rich landscape features and a minimum area of permanent grassland;
  - More equality in the distribution of funds received by farmers across the EU, and a reduction in payments above a certain amount for the biggest farms;

- Better targeting of income support to farmers (active farmers, young farmers, small farmers);

- A common organization of the markets in agricultural products:
  - Strengthening producer power in the food chain: Contracts (e.g. contract duration, quantity and quality requirements for agricultural products), price negotiations in dairy sector (e.g. maintaining effective competition), withdrawals (e.g. implementing private supply management to fix prices under difficult situations);

- Support for rural development:
  - Encouraging knowledge transfer and innovation in forestry and rural areas;
  - Enhancing farm viability and competitiveness;
  - Promoting food chain organization, including processing and marketing of agricultural products, risk management in agriculture;
  - Promoting resource efficiency and supporting environment-friendly practices (e.g. low carbon use);
  - Promoting social inclusion, poverty reduction.

- Financing, management and monitoring of the common agricultural policy:
  - European Price Monitoring Tool: It aims at collecting necessary information about the stages of food supply chain in the EU and the Member States (e.g. exports and imports, farm gate prices, consumer prices);
  - Food security and assessment of impact on developing countries.

B.3 Hierarchy of AGRI-related topics

In order to characterize subgroups of legislative propositions in a topical way, we use EUR-Lex [1], the website of the EU for the publication of official documents such as treaties and legislation. For indexing matters, this website provides a hierarchical nomenclature of topics.

The agriculture domain (AGRI) is represented over 4 hierarchical levels. The fourth one is too specific: for the considered period, each document basically concerns a different subdomain, which would prevent us from detecting any relevant pattern. For this reason, we work with the third level, which is represented in Table B.1. The subdomains relevant for the considered time period are represented in bold. Based on the titles and summaries of the legislative propositions, we manually annotate each of them with the most appropriate subdomains. Note that one document can be associated with several subdomains. Moreover, in addition to their agriculture-related subdomain(s),

1. http://eur-lex.europa.eu

Domains and subdomains for the legislative propositions related to agriculture in 2012-13

- Agriculture (AGRI)
  - Basic provisions (BP)
    - (22.9%) Common agricultural policy mechanisms (CAPM)
  - Agricultural structures (AS)
    - (15.9%) Social and structural measures (SSM)
    - (0.8%) Processing and marketing of agricultural products (PMAP)
  - Agricultural structural funds (ASF)
    - (0.8%) General (GEN)
    - (0.4%) European Agricultural Guarantee Fund (EAGF)
    - (10.8%) European Agricultural Fund for Rural Development (EAFRD)
  - Approximation of laws and health measures (ALHM)
    - (8.6%) Animal health and zootechnics (AHZ)
    - (0.4%) Seeds and seedlings (SS)
  - Products subject to market organization (PSMO)
    - (0.4%) Seeds (SEED)
    - (0.4%) Wine (WINE)
    - (37.9%) Arrangements covering multiple market organizations (ACMOMO)
- Regional policy and coordination of structural instruments (REGP)
  - (0.8%) Coordination of structural instruments (CSI)
- Environment, consumers and health protection (ENVI)
  - Consumers(CONS)
    - (0.4%) Protection of economic interests (PEI)
- External relations (EXTR)
  - (0.4%) Bilateral agreements with non-member countries (BANC)
- Fisheries (FISH)
  - (0.4%) External (EXT)

Table B.1 – EUR-Lex subdomains used to categorize the legislative propositions themes

some documents also belong to other *domains*, such as environment (ENVI) or fisheries (FISH). We proceed similarly to identify their subdomains, and these are also shown in the table. Finally, Table B.1 also displays the proportions of propositions described by each subdomain, for the considered selection.

B.4 Additional Plots for Figure 4.3

Figure B.1 represents cluster $\widehat{P_5^1}$, which was not included in Figure 4.3. Figure B.2 contains the names of all French MEPs. Their position in the figure match the one they have in Figure 4.3.

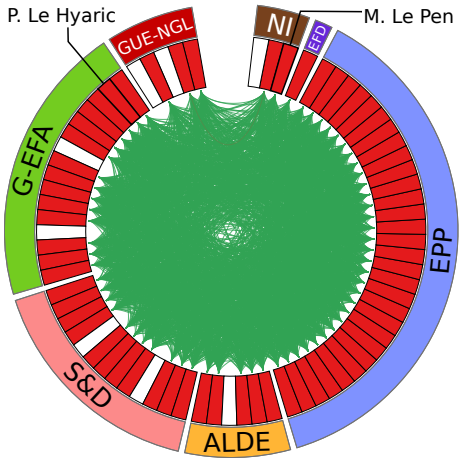


Figure B.1 – Characteristic pattern $\widehat{P_5^1}$ for French MEPs, in complement to Figure 4.3.

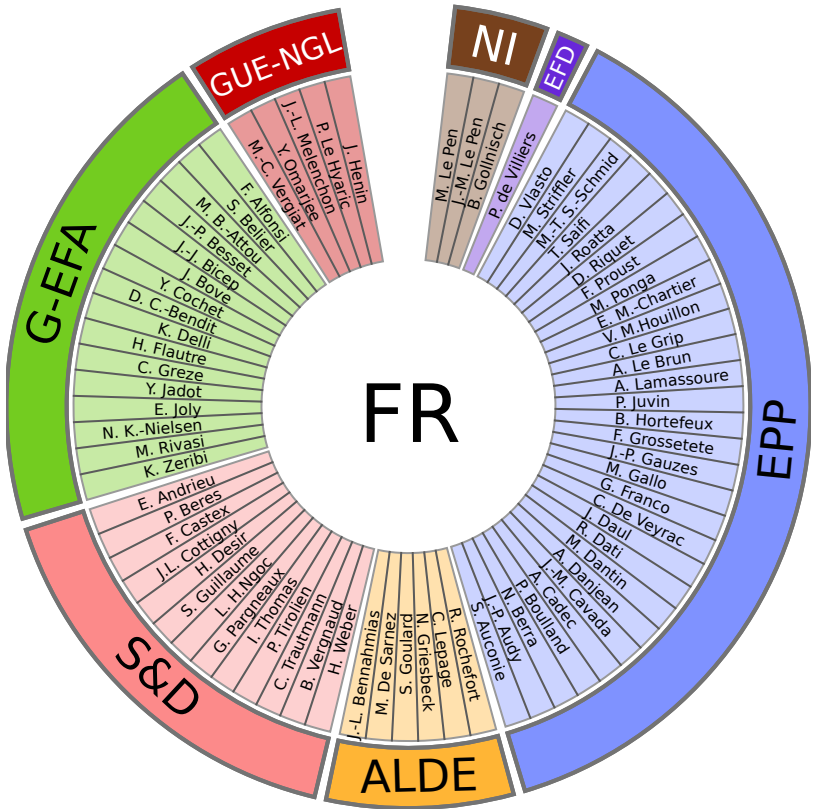


Figure B.2 – Name of the French MEPs studied in Section 4.5. The layout is the same as in Figure 4.3.

Appendix C

Edit Distance for Partitions, and Related Proofs

This appendix presents additional details and results regarding Chapter 5. We first detail how to calculate the edit distance between two membership vectors (Section C.1), then pass to the complete details regarding the proof of Lemma 5.11 (Section C.2). Our source code regarding the calculation of the edit distance between two membership vectors is publicly available [1].

C.1 Edit distance between two membership vectors

Before calculating the edit distance between two membership vectors, we need to determine one of two membership vectors as a *reference vector* in order to adapt the module assignments of the other membership vector based on the reference one. Hence, the edit distance is calculated between the reference vector and this newly changed one, that we call *relative vector*.

The task of adapting the relative vector w.r.t the reference one can be transformed into assignment problem, also known as maximum weighted bipartite matching problem, as already done in the literature [151]. Let π^s and π^t be two membership vectors of length n associated with the partitions P^s and P^t with ℓ^s and ℓ^t modules respectively. Also, since edit distance is symmetric, without loss of generality, let $\ell^s \leq \ell^t$. Moreover, let CM be the $\ell^s \times \ell^t$ confusion matrix of π^s and π^t. The term CM_{ij}, with $1 \leq i \leq \ell^s$ and $1 \leq j \leq \ell^t$, represents the number of vertices in the intersection of modules M_i^s and M_j^t, i.e. $|M_i^s \cap M_j^t|$. Then, we look for a bijection $f : \{1, 2, ..., \ell^s\} \rightarrow \{1, 2, ..., \ell^t\}$ such that the objective is to maximize the number of vertices that is common between modules of those membership vectors, i.e.

$$Max \sum_{i=1}^{\ell^s} CM_{if(i)}. \tag{C.1}$$

Since this problem can be modelled as assignment or maximum weighted bipartite matching problem, it can be solved in various ways. One of them is through well-known the Hungarian algorithm with the complexity $O(n^3)$ [134]. Nevertheless, the best polynomial time algorithm is currently

1. https://github.com/arinik9/ClusteringEditDistance.

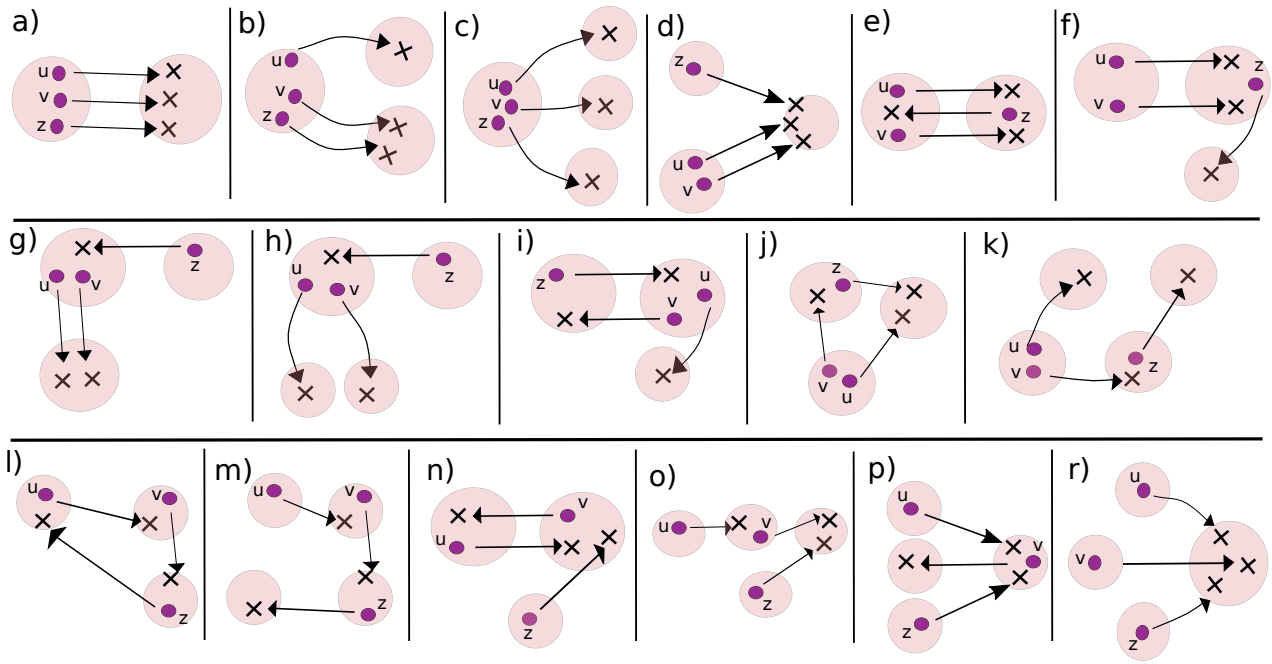


Figure C.1 – All atomic 3-edit operations

based on network simplex algorithm, and it runs in $O(|V||E|+|V|^2 log(|V|))$ time using the Fibonacci heap data structure [96]. One final remark is about the case of $\ell^s < \ell^t$, in which there will be $|\ell^t - \ell^s|$ unassigned module labels in π^t. In which case, one can arbitrarily renumber those labels, starting from $\ell^s + 1$.

Finally, the edit distance between two membership vectors is calculated by simply counting the number of cases where the module labels of the vertices in reference and relative vectors are different.

C.2 Proof of Lemma 5.11

This section gives the complete details regarding the proof of Lemma 5.11 based on the full list of 17 scenarios of three moving vertices with $uv, uz, vz \in \widetilde{E}$. Given source and target membership vectors π^s and π^t, $\widetilde{E}$ is defined as $\{(u,v) \mid (u,v) \in E$ and $u, v \in {}^{s\rightarrow t}V$ and $(\pi^s(u) = \pi^s(v) \vee \pi^t(u) = \pi^s(v) \vee \pi^s(u) = \pi^t(v) \vee \pi^t(u) = \pi^t(v))\}$. These 17 scenarios are depicted in Figure C.1. The proof is straightforward when one adapts Corollary 5.9 to those scenarios. We detail below all of them.

a) We have $(\gamma_u^{left} = a_{uv} + a_{uz}) > (\gamma_u^{right} = -a_{uv} - a_{uz})$, $(\gamma_v^{left} = a_{uv} + a_{vz}) > (\gamma_v^{right} = -a_{uv} - a_{vz})$ and $(\gamma_z^{left} = a_{uz} + a_{vz}) > (\gamma_z^{right} = -a_{uz} - a_{vz})$. We see that a_{uv}, a_{uz} and a_{vz} cannot be negative.

b) We have $(\gamma_u^{left} = a_{uv} + a_{uz}) > (\gamma_u^{right} = 0)$, $(\gamma_v^{left} = a_{uv} + a_{vz}) > (\gamma_v^{right} = -a_{vz})$ and $(\gamma_z^{left} = a_{uz} + a_{vz}) > (\gamma_z^{right} = -a_{vz})$. We see that a_{uv}, a_{uz} and a_{vz} cannot be negative.

c) We have $(\gamma_u^{left} = a_{uv} + a_{uz}) > (\gamma_u^{right} = 0)$, $(\gamma_v^{left} = a_{uv} + a_{vz}) > (\gamma_v^{right} = 0)$ and $(\gamma_z^{left} = a_{uz} + a_{vz}) > (\gamma_z^{right} = 0)$. We see that a_{uv}, a_{uz} and a_{vz} cannot be negative.

d) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = -a_{uv} - a_{uz})$, $(\gamma_v^{left} = a_{uv}) > (\gamma_v^{right} = -a_{uv} - a_{vz})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = -a_{uz} - a_{vz})$. We see that a_{uv}, a_{uz} and a_{vz} must be positive.

e) We have $(\gamma_u^{left} = a_{uv} - a_{uz}) > (\gamma_u^{right} = -a_{uv} + a_{uz})$, $(\gamma_v^{left} = a_{uv} - a_{vz}) > (\gamma_v^{right} = -a_{uv} + a_{vz})$ and $(\gamma_z^{left} = -a_{uz} - a_{vz}) > (\gamma_z^{right} = a_{uz} + a_{vz})$. We see that a_{uv} (resp. a_{uz} and a_{vz}) cannot be negative (resp. positive).

f) We have $(\gamma_u^{left} = a_{uv} - a_{uz}) > (\gamma_u^{right} = -a_{uv})$, $(\gamma_v^{left} = a_{uv} - a_{vz}) > (\gamma_v^{right} = -a_{uv})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = a_{uz} + a_{vz})$. We see that a_{uv}, a_{uz} and a_{vz} cannot be negative.

g) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = a_{uz} - a_{uv})$, $(\gamma_v^{left} = a_{uv}) > (\gamma_v^{right} = a_{vz} - a_{uv})$ and $(\gamma_z^{left} = -a_{uz} - a_{vz}) > (\gamma_z^{right} = 0)$. We see that a_{uv}, a_{uz} and a_{vz} cannot be negative.

h) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = -a_{uv})$, $(\gamma_v^{left} = a_{uv}) > (\gamma_v^{right} = -a_{uv})$ and $(\gamma_z^{left} = -a_{uz} - a_{vz}) > (\gamma_z^{right} = 0)$. We see that a_{uv}, a_{uz} and a_{vz} cannot be negative.

i) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = a_{uz})$, $(\gamma_v^{left} = a_{uv} - a_{vz}) > (\gamma_v^{right} = a_{vz})$ and $(\gamma_z^{left} = -a_{uz} - a_{vz}) > (\gamma_z^{right} = +a_{vz})$. We see that a_{uv} (resp. a_{uz} and a_{vz}) cannot be negative (resp. positive).

j) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = -a_{uz})$, $(\gamma_v^{left} = a_{uv}) > (\gamma_v^{right} = -a_{vz})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = -a_{uz} + a_{vz})$. We see that a_{uv} and a_{vz} (resp. a_{uz}) must be positive (resp. negative).

k) We have $(\gamma_u^{left} = a_{uv}) > (\gamma_u^{right} = 0)$, $(\gamma_v^{left} = a_{uv} - a_{vz}) > (\gamma_v^{right} = 0)$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = a_{vz})$. We see that a_{uv} and a_{vz} (resp. a_{uz}) must be positive (resp. negative).

l) We have $(\gamma_u^{left} = -a_{uv}) > (\gamma_u^{right} = a_{uz})$, $(\gamma_v^{left} = -a_{vz}) > (\gamma_v^{right} = a_{uv})$ and $(\gamma_z^{left} = -a_{uz}) > (\gamma_z^{right} = a_{vz})$. We see that a_{uv} and a_{vz} (resp. a_{uz}) must be positive (resp. negative).

m) We have $(\gamma_u^{left} = -a_{uv}) > (\gamma_u^{right} = 0)$, $(\gamma_v^{left} = -a_{vz}) > (\gamma_v^{right} = a_{uv})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = a_{vz})$. We see that a_{uv} and a_{vz} (resp. a_{uz}) must be positive (resp. negative).

n) We have $(\gamma_u^{left} = -a_{uv}) > (\gamma_u^{right} = a_{uv} - a_{uz})$, $(\gamma_v^{left} = -a_{uv}) > (\gamma_v^{right} = a_{uv} + a_{vz})$ and $(\gamma_z^{left} = -a_{vz}) > (\gamma_z^{right} = -a_{uz})$. We see that a_{uv} (resp. a_{uz} and a_{vz}) cannot be negative (resp. positive).

o) We have $(\gamma_u^{left} = -a_{uv}) > (\gamma_u^{right} = 0)$, $(\gamma_v^{left} = 0) > (\gamma_v^{right} = a_{uv} - a_{vz})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = -a_{vz})$. We see that a_{uv} (resp. a_{uz} and a_{vz}) cannot be negative (resp. positive).

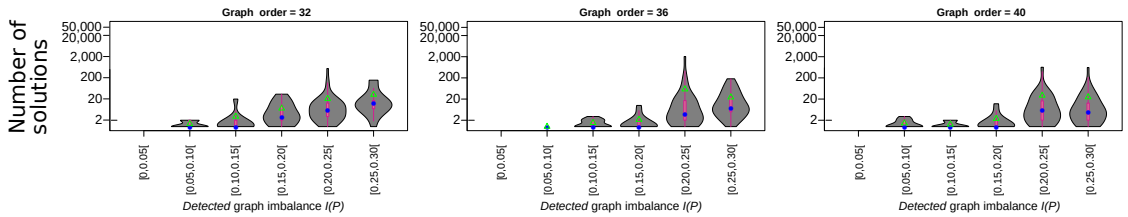
p) We have $(\gamma_u^{left} = -a_{uv}) > (\gamma_u^{right} = -a_{uz})$, $(\gamma_v^{left} = 0) > (\gamma_v^{right} = a_{uv} + a_{vz})$ and $(\gamma_z^{left} = -a_{vz}) > (\gamma_z^{right} = -a_{uz})$. We see that a_{uv} (resp. a_{uz} and a_{vz}) cannot be negative (resp. positive).

r) We have $(\gamma_u^{left} = 0) > (\gamma_u^{right} = -a_{uv} - a_{uz})$, $(\gamma_v^{left} = 0) > (\gamma_v^{right} = -a_{uv} - a_{vz})$ and $(\gamma_z^{left} = 0) > (\gamma_z^{right} = -a_{uz} - a_{vz})$. We see that a_{uv}, a_{uz} and a_{vz} must be positive.

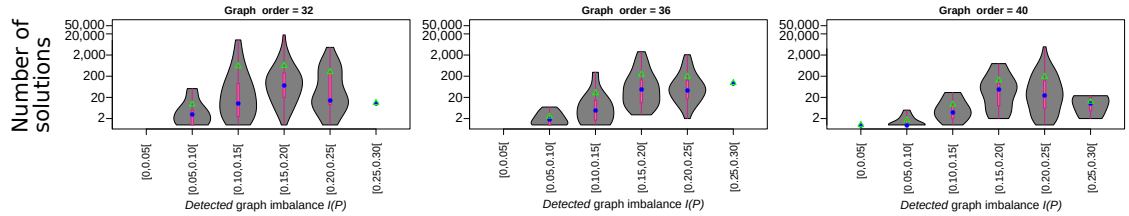
Appendix D

Number of Solutions of the CC Problem

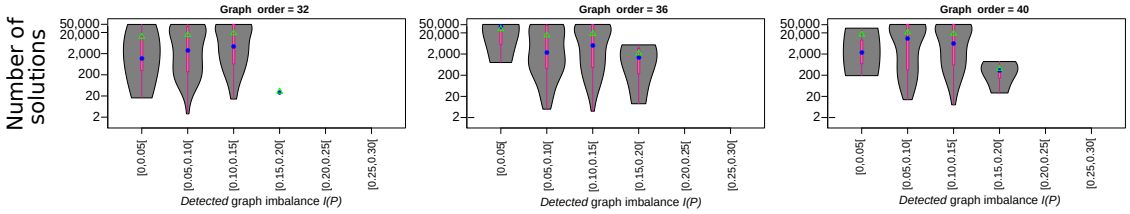
This appendix presents additional three figures regarding the number of solutions as a function of graph imbalance $I(P)$ based on the instances of *Dataset 5.1*. Figures D.1, D.2 and D.3 are complementary to Table 5.1.



(a) Number of solutions for $d = 0.25$ and $q_n = 0.3$.

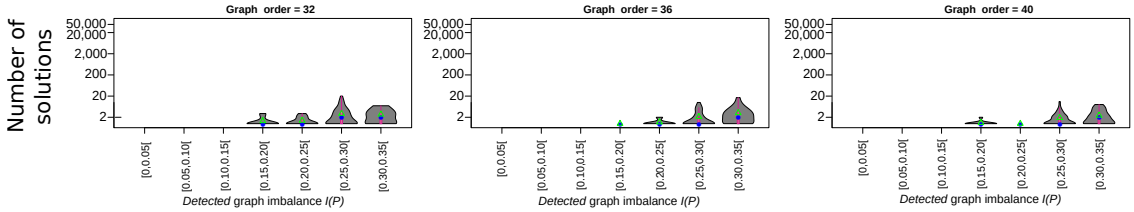


(b) Number of solutions for $d = 0.25$ and $q_n = 0.5$.

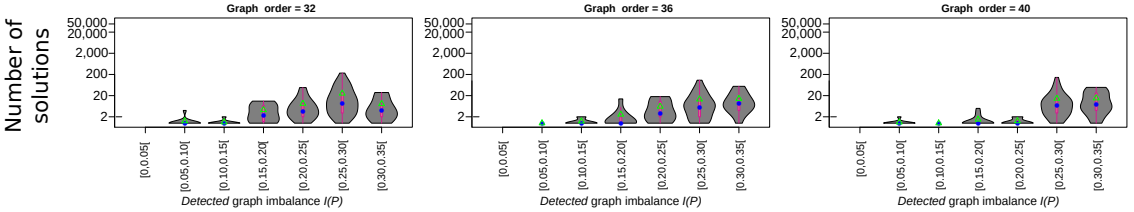


(c) Number of solutions for $d = 0.25$ and $q_n = 0.7$.

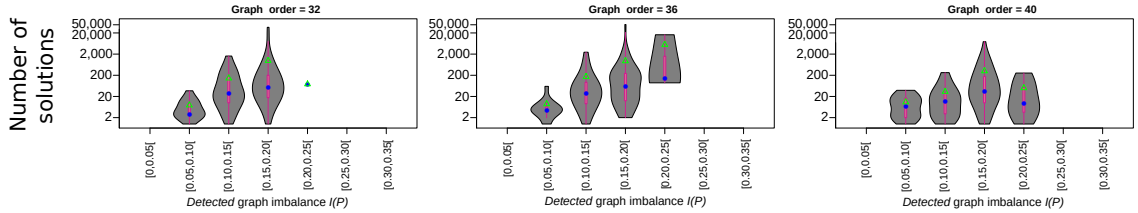
Figure D.1 – Number of solutions for $d = 0.25$ and $q_n = \{0.3, 0.5, 0.7\}$.



(a) Number of solutions for $d = 0.5$ and $q_n = 0.3$.



(b) Number of solutions for $d = 0.5$ and $q_n = 0.5$.



(c) Number of solutions for $d = 0.5$ and $q_n = 0.7$.

Figure D.2 – Number of solutions for $d = 0.5$ and $q_n = \{0.3, 0.5, 0.7\}$.

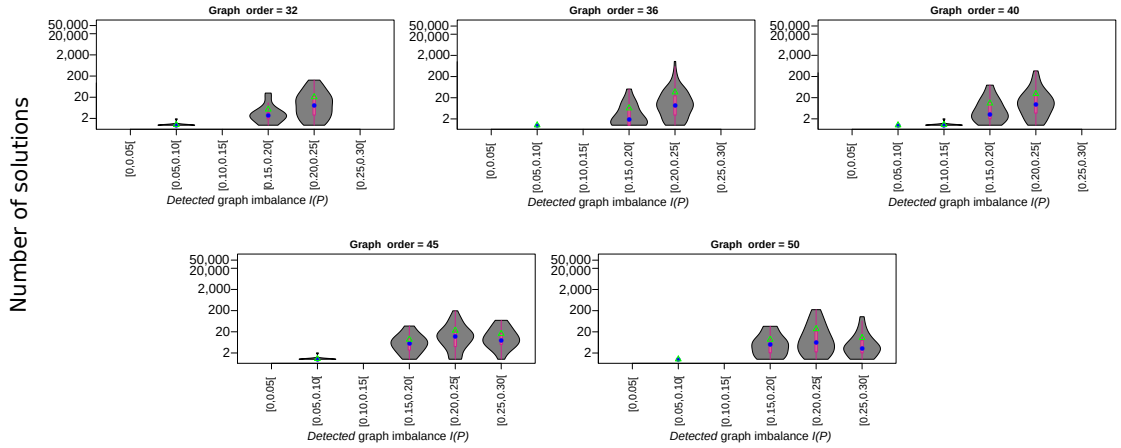


Figure D.3 – Number of solutions for $d = 1$ and $q_n \approx 0.7$.

Primary sources

[1] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen, « A survey of very large-scale neighborhood search techniques », *in*: *Discrete Applied Mathematics* 123.*1-3* (2002), pp. 75–102, DOI: 10.1016/s0166-218x(01)00338-9 (cit. on p. 112).

[2] A. N. Albatineh and M. Niewiadomska-Bugaj, « Correcting Jaccard and other similarity indices for chance agreement in cluster analysis », *in*: *Advances in Data Analysis and Classification* 5.*3* (2011), pp. 179–200, DOI: 10.1007/s11634-011-0090-y (cit. on pp. 51, 54).

[3] A. N. Albatineh, M. Niewiadomska-Bugaj, and D. Mihalko, « On Similarity Indices and Correction for Chance Agreement », *in*: *Journal of Classification* 23.*2* (Sept. 2006), pp. 301–313, DOI: 10.1007/s00357-006-0017-z (cit. on p. 49).

[4] Z. Ales, A. Knippel, and A. Pauchet, « Polyhedral Combinatorics of the K-partitioning Problem with Representative Variables », *in*: *Discrete Applied Mathematics* 211 (2016), pp. 1–14, DOI: 10.1016/j.dam.2016.04.002 (cit. on pp. 28, 29, 39).

[5] E. C. Alexopoulos, « Introduction to multivariate regression analysis », *in*: *Hippokratia* 14.*Suppl 1* (2010), pp. 23–28 (cit. on p. 57).

[8] A. Alush and J. Goldberger, « Ensemble Segmentation Using Efficient Integer Linear Programming », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.*10* (Oct. 2012), pp. 1966–1977, DOI: 10.1109/tpami.2011.280 (cit. on pp. 13, 23).

[9] A. Amelio and C. Pizzuti, « Correction for Closeness: Adjusting Normalized Mutual Information Measure for Clustering Comparison », *in*: *Computational Intelligence* 33.*3* (2016), pp. 579–601, DOI: 10.1111/coin.12100 (cit. on pp. 51–55, 72, 74, 75).

[10] B. Andres, T. Kroeger, K. L. Briggman, W. Denk, N. Korogod, G. Knott, U. Koethe, and F. A. Hamprecht, « Globally Optimal Closed-Surface Segmentation for Connectomics », *in*: *Computer Vision – ECCV 2012*, ed. by A. Fitzgibbon, S. Lazebnik, Pietro Perona, Y. Sato, and C. Schmid, Springer Berlin Heidelberg, 2012, pp. 778–791, ISBN: 978-3-642-33712-3, DOI: 10.1007/978-3-642-33712-3_56 (cit. on pp. 21, 24).

[11] G. Appa, « On the uniqueness of solutions to linear programs », *in*: *Journal of the Operational Research Society* 53.*10* (2002), pp. 1127–1132, DOI: 10.1057/palgrave.jors.2601320 (cit. on p. 139).

[12] S. Aref and M. C. Wilson, « Balance and frustration in signed networks », *in*: *Journal of Complex Networks* 7.*2* (2018), pp. 163–189, DOI: 10.1093/comnet/cny015 (cit. on p. 21).

[13] N. Arınık, R. Figueiredo, and V. Labatut, « Analysis of Roll-Calls in the European Parliament by Multiple Partitioning of Multiplex Signed Networks », *in*: *9ème Conférence Modèles & Analyse des Réseaux : Approches Mathématiques & Informatiques (MARAMI)*, 2018 (cit. on p. 17).

[14] N. Arınık, R. Figueiredo, and V. Labatut, « Characterizing measures for the assessment of cluster analysis and community detection », *in*: *11ème Conférence Modèles & Analyse de Réseaux : approches mathématiques et informatiques (MARAMI)*, 2020 (cit. on p. 18).

[15] N. Arınık, R. Figueiredo, and V. Labatut, « Efficient Enumeration of Correlation Clustering Optimal Solution Space (submitted) », *in*: *Journal of Global Optimization* (2021) (cit. on pp. 18, 105).

[16] N. Arınık, R. Figueiredo, and V. Labatut, « Exploiting Antagonistic Relations in Signed Graphs under the Structural Balance Hypothesis », *in*: *Programme Gaspard Monge Pour l'Optimisation (PGMO) Days*, 2018 (cit. on p. 18).

[17] N. Arınık, R. Figueiredo, and V. Labatut, « Multiple Optimal Solutions but Single Search: A Study of the Correlation Clustering Problem », *in*: *20ème congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Société Française de Recherche Opérationnelle et d'Aide à la Décision*, 2019 (cit. on p. 18).

[19] N. Arınık, R. Figueiredo, and V. Labatut, « Multiple partitioning of multiplex signed networks: Application to European parliament votes », *in*: *Social Networks* 60 (2020), pp. 83–102, DOI: https://doi.org/10.1016/j.socnet.2019.02.001 (cit. on pp. 17, 21, 81).

[20] N. Arınık, R. Figueiredo, and V. Labatut, « Multiplicity and Diversity: Analyzing the Optimal Solution Space of the Correlation Clustering Problem on Complete Signed Graphs », *in*: *Journal of Complex Networks* (2020), DOI: 10.1093/comnet/cnaa025 (cit. on pp. 17, 138).

[21] N. Arınık, R. Figueiredo, and V. Labatut, « Signed Graph Analysis for the Interpretation of Voting Behavior », *in*: *International Conference on Knowledge Technologies and Data-driven Business - International Workshop on Social Network Analysis and Digital Humanities*, 2017 (cit. on pp. 17, 81, 87, 92, 101).

[22] N. Arınık, R. Figueiredo, and V. Labatut, « Study of the European Parliament votes through the multiple partitioning of signed multiplex networks », *in*: *29th European Conference On Operational Research (EURO)*, 2018 (cit. on p. 17).

[23] N. Arınık, V. Labatut, and R. Figueiredo, « Characterizing and comparing external measures for the assessment of cluster analysis and community detection », *in*: *IEEE Access* (2021), pp. 1–22, DOI: 10.1109/access.2021.3054621 (cit. on pp. 17, 49).

[24] J. L. Arthur, M. Hachey, Sahr K., M. Huso, and A. R. Kiester, « Finding all optimal solutions to the reserve site selection problem », *in*: *Environmental and Ecological Statistics* 4.*2* (1997), pp. 153–165, DOI: 10.1023/a:1018570311399 (cit. on pp. 105–107, 110, 138).

[25] Javier Artiles, Julio Gonzalo, and Satoshi Sekine, « The SemEval-2007 WePS Evaluation: Establishing a Benchmark for the Web People Search Task », *in*: *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 64–69, URL: http://dl.acm.org/citation.cfm?id=1621474.1621486 (cit. on pp. 50, 67, 159).

[26] A. Bailoni, C. Pape, S. Wolf, T. Beier, A. Kreshuk, and F. A. Hamprecht, « A Generalized Framework for Agglomerative Clustering of Signed Graphs applied to Instance Segmentation », *in*: *arXiv e-prints*, arXiv:1906.11713 (June 2019), arXiv:1906.11713, arXiv: 1906.11713 (cit. on pp. 21, 33).

[27] N. Bansal, A. Blum, and S. Chawla, « Correlation Clustering », *in*: *43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 238–247, DOI: 10.1109/SFCS.2002.1181947 (cit. on pp. 11–14, 21).

[28] M. Barigozzi, G. Fagiolo, and G. Mangioni, « Identifying the community structure of the international-trade multi-network », *in*: *Physica A: Statistical Mechanics and its Applications* 390.*11* (June 2011), pp. 2051–2066, DOI: 10.1016/j.physa.2011.02.004 (cit. on p. 80).

[29] T. Beier, T. Kroeger, J. H. Kappes, U. Kothe, and F. A. Hamprecht, « Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning », *in*: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 73–80, DOI: 10.1109/CVPR.2014.17 (cit. on p. 33).

[30] Asa Ben-Hur, Andre Elisseeff, and Isabelle Guyon, « A stability based method for discovering structure in clustered data », *in*: *Pacific Symposium on Biocomputing 2002*, ed. by R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, World Scientific, 2001, pp. 6–17, DOI: 10.1142/9789812799623_0002 (cit. on pp. 66, 159).

[31] S. Benati, J. Puerto, and A. M. Rodríguez-Chía, « Clustering data that are graph connected », *in*: *European Journal of Operational Research* 261.*1* (2017), pp. 43–53, DOI: 10.1016/j.ejor.2017.02.009 (cit. on p. 152).

[32] J. Berg and M. Järvisalo, « Cost-optimal constrained correlation clustering via weighted partial Maximum Satisfiability », *in*: *Artificial Intelligence* 244 (Mar. 2017), pp. 110–142, DOI: 10.1016/j.artint.2015.07.001 (cit. on pp. 22, 108, 155).

[33] M. Berlingerio, F. Pinelli, and F. Calabrese, « ABACUS: frequent pAttern mining-BAsed Community discovery in mUltidimensional networkS », *in*: *Data Mining and Knowledge Discovery* 27.*3* (July 2013), pp. 294–320, DOI: 10.1007/s10618-013-0331-0 (cit. on p. 80).

[34] C. Blum and A. Roli, « Metaheuristics in combinatorial optimization », *in*: *ACM Computing Surveys* 35.*3* (Sept. 2003), pp. 268–308, DOI: 10.1145/937503.937505 (cit. on p. 112).

[35] S. Böcker, S. Briesemeister, and G. W. Klau, « Exact Algorithms for Cluster Editing: Evaluation and Experiments », *in*: *Algorithmica* 60.*2* (July 2009), pp. 316–334, DOI: 10.1007/s00453-009-9339-7 (cit. on pp. 13, 22).

[36] F. Bonchi, A. Gionis, F. Gullo, C. E. Tsourakakis, and A. Ukkonen, « Chromatic Correlation Clustering », *in*: *ACM Transactions on Knowledge Discovery from Data* 9.*4* (June 2015), pp. 1–24, DOI: 10.1145/2728170 (cit. on p. 155).

[37] M. J. Brusco and P. Doreian, « Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search », *in*: *Social Networks* 56 (Jan. 2019), pp. 70–80, DOI: 10.1016/j.socnet.2018.08.007 (cit. on pp. 34–36, 152).

[38] Michael Brusco and Douglas Steinley, « K-balance partitioning: An exact method with applications to generalized structural balance and other psychological contexts », *in*: *Psychological Methods* 15.*2* (2010), pp. 145–157, DOI: 10.1037/a0017738 (cit. on pp. 15, 21, 22, 103, 105, 107).

[39] D. Cartwright and F. Harary, « Structural balance: A generalization of Heider's theory », *in*: *Psychological Review* 63 (1956), pp. 277–293, DOI: 10.1037/h0046049 (cit. on pp. 11, 152).

[40] V. Černý, « Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm », *in*: *Journal of Optimization Theory and Applications* 45.*1* (Jan. 1985), pp. 41–51, DOI: 10.1007/bf00940812 (cit. on p. 33).

[41] Y.-T. Chang and D. Pantazis, « Multi-view network module detection », *in*: *Asilomar Conference on Signals, Systems and Computers*, 2013, pp. 975–979, DOI: 10.1109/ACSSC.2013.6810435 (cit. on p. 79).

[42] M. Charikar, V. Guruswami, and A. Wirth, « Clustering with qualitative information », *in*: *Journal of Computer and System Sciences* 71.*3* (2005), pp. 360–383, DOI: https://doi.org/10.1016/j.jcss.2004.10.012 (cit. on p. 31).

[43] Moses Charikar, Neha Gupta, and Roy Schwartz, « Local Guarantees in Graph Cuts and Clustering », *in*: *Integer Programming and Combinatorial Optimization*, Springer International Publishing, 2017, pp. 136–147, DOI: 10.1007/978-3-319-59250-3_12 (cit. on p. 122).

[44] S. Chawla, K. Makarychev, T. Schramm, and G. Yaroslavtsev, « Near Optimal LP Rounding Algorithm for CorrelationClustering on Complete and Complete k-partite Graphs », *in*: *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, ACM, June 2015, pp. 219–228, DOI: 10.1145/2746539.2746604 (cit. on p. 22).

[45] Y. Chen and J. W. Baker, « Community Detection in Spatial Correlation Graphs: Application to Non-stationary Ground Motion Modeling », *in*: *Computers and Geoscience* (2021), In press (cit. on p. 11).

[46] K.-Y. Chiang, C.-J. Hsieh, N. Natarajan, I. S. Dhillon, and A. Tewari, « Prediction and clustering in signed networks: a local to global perspective », *in*: *Journal of Machine Learning Research* 15.*1* (2014), pp. 1177–1213, URL: http://jmlr.org/papers/v15/chiang14a.html (cit. on p. 11).

[47] S. Chopra and M. R. Rao, « The partition problem », *in*: *Mathematical Programming* 59.*1-3* (Mar. 1993), pp. 87–115, DOI: 10.1007/bf01581239 (cit. on pp. 24, 26, 27).

[48] J. Cohen, « A Coefficient of Agreement for Nominal Scales », *in*: *Educational and Psychological Measurement* 20 (1960), pp. 37–46, DOI: 10.1177/001316446002000104 (cit. on p. 51).

[49] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*, 3rd Edition, Routledge, 2002, ISBN: 9780203774441, DOI: 10.4324/9780203774441 (cit. on pp. 64, 68).

[50] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd Edition, Wiley-Interscience, 2006, ISBN: 978-0-471-24195-9 (cit. on pp. 67, 160).

[51] M. Cucuringu, « Synchronization over Z_2 and community detection in signed multiplex networks with constraints », *in*: *Journal of Complex Networks* 3.*3* (2015), pp. 469–506, DOI: 10.1093/comnet/cnu050 (cit. on p. 11).

[52] M. Cucuringu, P. Davies, A. Glielmo, and H. Tyagi, « SPONGE: A generalized eigenproblem for clustering signed networks », *in*: *Proceedings of Machine Learning Research*, vol. 89, Proceedings of Machine Learning Research, PMLR, 2019, pp. 1088–1098 (cit. on p. 21).

[53] P. Damaschke, « Fixed-Parameter Enumerability of Cluster Editing and Related Problems », *in*: *Theory of Computing Systems* 46.*2* (July 2010), pp. 261–283, DOI: 10.1007/s00224-008-9130-1 (cit. on p. 107).

[54] E. Danna, M. Fenelon, Z. Gu, and R. Wunderling, « Generating Multiple Solutions for Mixed Integer Programming Problems », *in*: *International Conference on Integer Programming and Combinatorial Optimization*, Springer, 2007, pp. 280–294, DOI: 10.1007/978-3-540-72792-7_22 (cit. on pp. 107–109, 139).

[55] B. DasGupta, G. A. Enciso, E. Sontag, and Y. Zhang, « Algorithmic and complexity results for decompositions of biological networks into monotone subsystems », *in*: *Biosystems* 9.*1* (2007), pp. 161–178, DOI: 10.1016/j.biosystems.2006.08.001 (cit. on p. 11).

[56] J. A. Davis, « Clustering and structural balance in graphs », *in*: *Human Relations* 20.*2* (1967), pp. 181–187, DOI: 10.1177/001872676702000207 (cit. on pp. 11, 15, 103, 105, 106, 139).

[57] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica, « Correlation clustering in general weighted graphs », *in*: *Theoretical Computer Science* 361.*2-3* (Sept. 2006), pp. 172–187, DOI: 10.1016/j.tcs.2006.05.008 (cit. on pp. 13–15, 24, 103, 105).

[58] M. Deza, M. Grötschel, and M. Laurent, « Clique-Web Facets for Multicut Polytopes », *in*: *Mathematics of Operations Research* 17.*4* (Nov. 1992), pp. 981–1000, DOI: 10.1287/moor.17.4.981 (cit. on p. 27).

[59] G. Didier, C. Brun, and A. Baudot, « Identifying communities from multiplex biological networks », *in*: *PeerJ* 3 (Dec. 2015), e1525, DOI: 10.7717/peerj.1525 (cit. on pp. 80, 81).

[60] B. E. Dom, « An Information-theoretic External Cluster-validity Measure », *in*: *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, UAI'02, Morgan Kaufmann Publishers Inc., 2002, pp. 137–145, ISBN: 1-55860-897-4, URL: http://dl.acm.org/citation.cfm?id=2073876.2073893 (cit. on pp. 54, 56, 57).

[61] S. Dongen, *Performance Criteria for Graph Clustering and Markov Cluster Experiments*, tech. rep. 4, Amsterdam, The Netherlands, The Netherlands: National Research Institute For Mathematics and Computer Science, 2000, DOI: 10.5445/IR/1000011477 (cit. on pp. 50, 54).

[62] P. Doreian, V. Batagelj, and A. Ferligoj, *Generalized Blockmodeling*, Cambridge University Press, 2005, URL: https://www.cambridge.org/core/books/generalized-blockmodeling/E9B040215C13C1819EA98F2F932BE0CE (cit. on pp. 15, 103, 105, 106, 139).

[63] P. Doreian and A. Mrvar, « A partitioning approach to structural balance », *in*: *Social Networks* 18.*2* (1996), pp. 149–168, DOI: 10.1016/0378-8733(95)00259-6 (cit. on pp. 11, 13, 15, 103, 105, 138).

[64] P. Doreian and A. Mrvar, « Partitioning signed social networks », *in*: *Social Networks* 31.*1* (2009), pp. 1–11, DOI: 10.1016/j.socnet.2008.08.001 (cit. on p. 13).

[65] P. Doreian and A. Mrvar, « Structural balance and signed international relations », *in*: *Journal of Social Structure* 16 (2015), p. 1, DOI: 10.21307/joss-2019-012 (cit. on pp. 11, 37, 104).

[66] L. Drummond, R. Figueiredo, Y. Frota, and M. Levorato, « Efficient Solution of the Correlation Clustering Problem: An Application to Structural Balance », *in*: *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, ed. by Y. T. Demey and H. Panetto, vol. 8186, Full description of the ILS algorithm, Springer Berlin Heidelberg, 2013, pp. 674–683, ISBN: 978-3-642-41033-8, DOI: 10.1007/978-3-642-41033-8_85 (cit. on pp. 21, 35, 152).

[67] M. Elsner and W. Schudy, « Bounding and Comparing Methods for Correlation Clustering Beyond ILP », *in*: *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, 2009, pp. 19–27 (cit. on pp. 21, 32).

[68] P. Esmailian, S. E. Abtahi, and M. Jalili, « Mesoscopic Analysis of Online Social Networks: The Role of Negative Ties », *in*: *Physical Review E* 90.*4* (Oct. 2014), DOI: 10.1103/physreve.90.042817 (cit. on p. 11).

[69] P. Esmailian and M. Jalili, « Community Detection in Signed Networks: the Role of Negative ties in Different Scales », *in*: *Scientific Reports* 5.*1* (Sept. 2015), DOI: 10.1038/srep14339 (cit. on pp. 11, 154).

[70] J. Esteban, L. Mayoral, and D. Ray, « Ethnicity and Conflict: An Empirical Study », *in*: *American Economic Review* 102.*4* (2012), pp. 1310–1342, DOI: 10.1257/aer.102.4.1310 (cit. on pp. 37, 105).

[71] E. Estrada, « Rethinking structural balance in signed social networks », *in*: *Discrete Applied Mathematics* 268 (2019), pp. 70–90, DOI: 10.1016/j.dam.2019.04.019 (cit. on p. 152).

[72] E. Estrada and M. Benzi, « Walk-based measure of balance in signed networks: Detecting lack of balance in social networks », *in*: *Physical Review E* 90.*4* (2014), p. 042802, DOI: 10.1103/PhysRevE.90.042802 (cit. on p. 11).

[86] J. J. Faraway, « Practical regression and ANOVA using R », Accessed on 07/2020, https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf, 2002, URL: https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf (cit. on p. 68).

[87] T. A. Feo and M. G. C. Resende, « A probabilistic heuristic for a computationally difficult set covering problem », *in*: *Operations Research Letters* 8.*2* (Apr. 1989), pp. 67–71, DOI: 10.1016/0167-6377(89)90002-3 (cit. on p. 35).

[88] T. A. Feo and M. G. C. Resende, « Greedy Randomized Adaptive Search Procedures », *in*: *Journal of Global Optimization* 6.*2* (Mar. 1995), pp. 109–133, DOI: 10.1007/bf01096763 (cit. on p. 35).

[89] R. Figueiredo and Y. Frota, « The Maximum Balanced Subgraph of a Signed Graph: Applications and Solution Approaches », *in*: *European Journal of Operational Research* 236.*2* (July 2014), pp. 473–487, DOI: 10.1016/j.ejor.2013.12.036 (cit. on p. 86).

[90] R. Figueiredo and G. Moura, « Mixed Integer Programming Formulations for Clustering Problems Related to Structural Balance », *in*: *Social Networks* 35.*4* (2013), pp. 639–651, DOI: 10.1016/j.socnet.2013.09.002 (cit. on pp. 14, 21, 22, 24, 105, 107).

[91] R. M. V. Figueiredo, M. Labbé, and C. C. de Souza, « An exact approach to the problem of extracting an embedded network matrix », *in*: *Computers & Operations Research* 38.*11* (2011), pp. 1483–1492, DOI: 10.1016/j.cor.2011.01.003 (cit. on p. 12).

[92] Matteo Fischetti, Fred Glover, and Andrea Lodi, « The feasibility pump », *in*: *Mathematical Programming* 104.*1* (Mar. 2005), pp. 91–104, DOI: 10.1007/s10107-004-0570-3 (cit. on p. 109).

[93] S. Fortunato, « Community detection in graphs », *in*: *Physics Reports* 486.*3-5* (2010), pp. 75–174, DOI: 10.1016/j.physrep.2009.11.002 (cit. on p. 11).

[94] E. B. Fowlkes and C. L. Mallows, « A Method for Comparing Two Hierarchical Clusterings », *in*: *Journal of the American Statistical Association* 78.*383* (1983), pp. 553–569, DOI: 10.1080/01621459.1983.10478008 (cit. on pp. 52, 54, 67, 159).

[95] P. Fränti, M. Rezaei, and Q. Zhao, « Centroid index: Cluster level similarity measure », *in*: *Pattern Recognition* 47.*9* (2014), pp. 3034–3045, DOI: 10.1016/j.patcog.2014.03.017 (cit. on pp. 48, 52, 54).

[96] Michael L. Fredman and Robert Endre Tarjan, « Fibonacci heaps and their uses in improved network optimization algorithms », *in*: *Journal of the ACM (JACM)* 34.*3* (1987), pp. 596–615, DOI: 10.1145/28869.28874 (cit. on p. 171).

[97] Alexander J Gates, Ian B Wood, William P Hetrick, and Yong-Yeol Ahn, « Element-centric framework unifies overlaps and hierarchy », *in*: *Scientific Reports* 9.*1* (2017), ISSN: 2045-2322, DOI: 10.1038/s41598-019-44892-y (cit. on pp. 51–54, 56, 62, 63, 75, 77).

[98] Alexander J. Gates and Yong-Yeol Ahn, « The Impact of Random Models on Clustering Similarity », *in*: *Journal of Machine Learning Research* 18.*1* (2017), pp. 3049–3076, ISSN: 1532-4435, URL: http://dl.acm.org/citation.cfm?id=3122009.3176831 (cit. on pp. 51, 53, 54, 67, 159).

[100] M. Gong, B. Fu, L. Jiao, and H. Du, « Memetic algorithm for community detection in networks », *in*: *Physical Review E* 84.*5* (Nov. 2011), DOI: 10.1103/physreve.84.056101 (cit. on p. 36).

[101] B. H Good, Y.-A. de Montjoye, and A. Clauset, « Performance of modularity maximization in practical contexts », *in*: *Physical Review E* 81.*4* (2010), p. 046106 (cit. on pp. 138, 151).

[102] L. A. Goodman and W. H. Kruskal, « Measures of Association for Cross Classification », *in*: *Journal of the American Statistical Association* 49.*268* (1954), pp. 732–64, DOI: 10.2307/2281536 (cit. on p. 51).

[103] M. Grötschel and Y. Wakabayashi, « A cutting plane algorithm for a clustering problem », *in*: *Mathematical Programming* 45.*1-3* (Aug. 1989), pp. 59–96, DOI: 10.1007/bf01589097 (cit. on pp. 24, 104).

[104] M. Grötschel and Y. Wakabayashi, « Facets of the clique partitioning polytope », *in*: *Mathematical Programming* 47.*1-3* (May 1990), pp. 367–387, DOI: 10.1007/bf01580870 (cit. on p. 27).

[107] D. N. Gujarati, *Basic Econometrics*, ed. by A. Bright, McGraw-Hill, 2003 (cit. on pp. 65, 68).

[108] M. Hardy, *Regression with Dummy Variables*, Quantitative Applications in the Social Sciences, SAGE Publications, Inc., 1993, ISBN: 9780803951280, DOI: 10.4135/9781412985628 (cit. on pp. 64, 66).

[109] A. Hassan, A. Abu-Jbara, and D. Radev, « Detecting subgroups in online discussions by modeling positive and negative relations among participants », *in*: *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 59–70, URL: https://www.aclweb.org/anthology/D12-1006/ (cit. on p. 11).

[110] F. Heider, « Attitudes and cognitive organization », *in*: *Journal of Psychology* 21.*1* (1946), pp. 107–112, DOI: 10.1080/00223980.1946.9917275 (cit. on p. 11).

[111] H. van der Hoef and M. J. Warrens, « Understanding information theoretic measures for comparing clusterings », *in*: *Behaviormetrika* 46.*2* (2019), pp. 353–370, DOI: 10.1007/s41237-018-0075-7 (cit. on pp. 52, 54, 56).

[112] D. Horta and R. J. G. B. Campello, « Comparing Hard and Overlapping Clusterings », *in*: *Journal of Machine Learning Research* 16.*93* (2015), Editor: Marina Meila, pp. 2949–2997, URL: http://jmlr.org/papers/v16/horta15a.html (cit. on pp. 52, 54, 57, 77).

[113] Z. Huang and Y. Qiu, « A multiple-perspective approach to constructing and aggregating Citation Semantic Link Network », *in*: *Future Generation Computer Systems* 26.*3* (2010), pp. 400–407, DOI: 10.1016/j.future.2009.07.006 (cit. on p. 11).

[114] L. Hubert and P. Arabie, « Comparing partitions », *in*: *Journal of Classification* 2.*1* (1985), pp. 193–218, DOI: 10.1007/bf01908075 (cit. on pp. 48, 51, 54, 66, 158).

[115] G. Iacono, C. Altafini, N. Soranzo, and F. Ramezani, « Determining the distance to monotonicity of a biological network: a graph-theoretical approach », *in*: *IET Systems Biology* 4.*3* (May 2010), pp. 223–235, DOI: 10.1049/iet-syb.2009.0040 (cit. on p. 39).

[116] J. Iacovacci and G. Bianconi, « Extracting information from multiplex networks », *in*: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 26.*6* (June 2016), p. 065306, DOI: 10.1063/1.4953161 (cit. on p. 81).

[118] P. Jaccard, « Étude comparative de la distribution florale dans une portion des Alpes et des Jura », *in*: *Bulletin de la Société Vaudoise des Sciences Naturelles* 37.*142* (1901), pp. 547–579, DOI: 10.5169/seals-266450 (cit. on pp. 66, 159).

[119] P. Jain, R. Meka, and I. Dhillon, « Simultaneous Unsupervised Learning of Disparate Clusterings », *in*: *Statistical Analysis and Data Mining* 1.*3* (2008), pp. 195–210, DOI: 10.1002/sam.10007 (cit. on p. 139).

[120] P. Jensen, « Network-based predictions of retail store commercial categories and optimal locations », *in*: *Physical Review E* 74.*3* (2006), 035101(R), DOI: 10.1103/PhysRevE.74.035101 (cit. on p. 11).

[121] J. W. Johnson and J. M. Lebreton, « History and Use of Relative Importance Indices in Organizational Research », *in*: *Organizational Research Methods* 7.*3* (2004), pp. 238–257, DOI: 10.1177/1094428104266510 (cit. on pp. 65, 66).

[122] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd edition, Prentice-Hall, Inc., 2000 (cit. on p. 112).

[123] P. Kaniok and O. Mocek, « Roll Call Votes in the European Parliament: a good sample or a poisoned dead end? », *in*: *Parliaments, Estates and Representation* 37.*1* (2017), pp. 75–88, DOI: 10.1080/02606755.2016.1232994 (cit. on p. 87).

[124] T.-C. Kao and M. A. Porter, « Layer Communities in Multiplex Networks », *in*: *Journal of Statistical Physics* 173.*3-4* (Aug. 2017), pp. 1286–1302, DOI: 10.1007/s10955-017-1858-z (cit. on p. 81).

[125] T. D. Kaplan and S. Forrest, « A dual assortative measure of community structure », *in*: *arXiv* physics.data-an (2008), p. 0801.3290, URL: http://arxiv.org/abs/0801.3290 (cit. on p. 11).

[126] J. H. Kappes, M. Speth, G. Reinelt, and C. Schnörr, « Higher-order segmentation via multicuts », *in*: *Computer Vision and Image Understanding* 143 (Feb. 2016), pp. 104–119, DOI: 10.1016/j.cviu.2015.11.005 (cit. on pp. 27–29).

[127] D. Karaboga and C. Ozturk, « A novel clustering approach: Artificial Bee Colony (ABC) algorithm », *in*: *Applied Soft Computing* 11.*1* (Jan. 2011), pp. 652–657, DOI: 10.1016/j.asoc.2009.12.025 (cit. on p. 36).

[128] L. Kaufman and P. J. Rousseeuw, « Partitioning Around Medoids », *in*: *Finding Groups in Data: An Introduction to Cluster Analysis*, Hoboken, NJ, USA: John Wiley & Sons, 2009, 10.1002/9780470316801.ch2, DOI: 10.1002/9780470316801.ch2 (cit. on pp. 75, 85, 143).

[129] B. W. Kernighan and S. Lin, « An Efficient Heuristic Procedure for Partitioning Graphs », *in*: *Bell System Technical Journal* 49.*2* (1970), pp. 291–307, DOI: 10.1002/j.1538-7305.1970.tb01770.x (cit. on pp. 32, 33).

[130] M. Keuper, J. Lukasik, M. Singh, and J. Yarkony, « A Benders Decomposition Approach to Correlation Clustering », *in*: *The International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2020 (cit. on pp. 21, 24, 29, 39, 105).

[131] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, « Optimization by Simulated Annealing », *in*: *Science* 220.*4598* (May 1983), pp. 671–680, DOI: 10.1126/science.220.4598.671 (cit. on p. 33).

[132] M. Kivelä, A. Arenas, M. Barthélemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, « Multilayer Networks », *in*: *Journal of Complex Networks* 2.*3* (2014), pp. 203–271, DOI: 10.1093/comnet/cnu016 (cit. on pp. 15, 80).

[133] R. Kolluri, J. R. Shewchuk, and J. F. O'Brien, « Spectral surface reconstruction from noisy point clouds », *in*: *Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, 2004, pp. 11–21, DOI: 10.1145/1057432.1057434 (cit. on p. 11).

[134] H. W. Kuhn, « The Hungarian method for the assignment problem », *in*: *Naval Research Logistics Quarterly* 2.*1-2* (1955), pp. 83–97, DOI: 10.1002/nav.3800020109 (cit. on p. 170).

[135] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li, *Applied linear statistical models*, ed. by B. Gordon, 5th Edition, McGraw-Hill Irwin, 2005, ISBN: 0-07-238688-6 (cit. on pp. 65, 68).

[136] O. T. Kvålseth, « Entropy and Correlation: Some Comments », *in*: *IEEE Transactions on Systems, Man, and Cybernetics* 17.*3* (1987), pp. 517–519, DOI: 10.1109/tsmc.1987.4309069 (cit. on p. 160).

[137] O. T. Kvålseth, « On Normalized Mutual Information: Measure Derivations and Properties », *in*: *Entropy* 19.*11* (2017), pp. 631–645, DOI: 10.3390/e19110631 (cit. on pp. 50, 54).

[138] V. Labatut, « Generalised measures for the evaluation of community detection methods », *in*: *International Journal of Social Network Mining* 2.*1* (2015), pp. 44–63, DOI: 10.1504/ijsnm.2015.069776 (cit. on p. 77).

[139] D. Lai and C. Nardini, « A corrected normalized mutual information for performance evaluation of community detection », *in*: *Journal of Statistical Mechanics: Theory and Experiment* 2016.*9* (Sept. 2016), p. 093403, DOI: 10.1088/1742-5468/2016/09/093403 (cit. on p. 54).

[140] A. Lancichinetti and S. Fortunato, « Consensus clustering in complex networks », *in*: *Scientific Reports* 2 (2012), p. 336, DOI: 10.1038/srep00336 (cit. on pp. 85, 143).

[141] J.-H. Lange, « Multicut Optimization Guarantees & Geometry of Lifted Multicuts », PhD thesis, Saarland University, 2020 (cit. on pp. 13, 32).

[142] J.-H. Lange, A. Karrenbauer, and B. Andres, « Partial Optimality and Fast Lower Bounds for Weighted Correlation Clustering », *in*: *Proceedings of the 35th International Conference on Machine Learning*, ed. by J. Dy and A. Krause, vol. 80, 2018, pp. 2892–2901 (cit. on pp. 24, 32).

[143] L. M. Le Cam, *Asymptotic Methods in Statistical Decision Theory*, Springer-Verlag New York, 1986, ISBN: 978-1-4612-4946-7, DOI: 10.1007/978-1-4612-4946-7 (cit. on p. 75).

[145] J. Leskovec, D. Huttenlocher, and J. Kleinberg, « Signed networks in social media », *in*: *Proceedings of the SIGCHI conference on human factors in computing systems*, 2010, pp. 1361–1370 (cit. on p. 39).

[146] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford Large Network Dataset Collection*, http://snap.stanford.edu/data, June 2014 (cit. on p. 39).

[147] E. Levinkov, A. Kirillov, and B. Andres, « A Comparative Study of Local Search Algorithms for Correlation Clustering », *in*: *Lecture Notes in Computer Science*, Springer International Publishing, 2017, pp. 103–114, DOI: 10.1007/978-3-319-66709-6_9 (cit. on p. 33).

[148] M. Levorato, R. Figueiredo, Y. Frota, and L. Drummond, « Evaluating balancing on social networks through the efficient solution of correlation clustering problems », *in*: *EURO Journal on Computational Optimization* 5.*4* (Jan. 2017), Full description of the ILS algorithm, pp. 467–498, DOI: 10.1007/s13675-017-0082-6 (cit. on pp. 14, 21, 35, 36, 42, 86, 152).

[149] P. Li, H. Dau, G. Puleo, and O. Milenkovic, « Motif clustering and overlapping clustering for social network analysis », *in*: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, IEEE, May 2017, pp. 1–9, DOI: 10.1109/infocom.2017.8056956 (cit. on p. 155).

[150] P. Liu, T.-D. Nguyen, X. Cai, and X. Jiang, « Finding Multiple Optimal Solutions to Optimal Load Distribution Problem in Hydropower Plant », *in*: *Energies* 5.*5* (2012), pp. 1413–1432, DOI: 10.3390/en5051413 (cit. on p. 106).

[151] X. Liu, H.-M. Cheng, and Z.-Y. Zhang, « Evaluation of Community Detection Methods », *in*: *IEEE Transactions on Knowledge and Data Engineering* 32.*9* (2019), pp. 1736–1746, DOI: 10.1109/tkde.2019.2911943 (cit. on pp. 52–54, 62, 63, 170).

[152] H. R. Lourenço, O. C. Martin, and T. Stützle, « Iterated Local Search », *in*: *Handbook of Metaheuristics*, ed. by F. Glover and G. A. Kochenberger, Kluwer Academic Publishers, 2003, pp. 320–353, ISBN: 978-0-306-48056-0, DOI: 10.1007/0-306-48056-5_11 (cit. on p. 36).

[153] P. Luo, H. Xiong, G. Zhan, J. Wu, and Z. Shi, « Information-Theoretic Distance Measures for Clustering Validation: Generalization and Normalization », *in*: *IEEE Transactions on Knowledge and Data Engineering* 21.*9* (2009), pp. 1249–1262, DOI: 10.1109/tkde.2008.200 (cit. on p. 57).

[154] F. Ma and J.-K. Hao, « A multiple search operator heuristic for the max-k-cut problem », *in*: *Annals of Operations Research* 248.*1* (June 2016), pp. 365–403, DOI: 10.1007/s10479-016-2234-0 (cit. on p. 112).

[155] L. Ma, M. Gong, H. Du, B. Shen, and L. Jiao, « A memetic algorithm for computing and transforming structural balance in signed networks », *in*: *Knowledge-Based Systems* 85 (2015), pp. 196–209, DOI: 10.1016/j.knosys.2015.05.006 (cit. on pp. 36, 37, 42, 152).

[156] M. MacMahon and D. Garlaschelli, « Community detection for correlation matrices », *in*: *Physical Review X* 5.*2* (2015), p. 021006, DOI: 10.1103/PhysRevX.5.021006 (cit. on p. 11).

[157] K. T. Macon, P. J. Mucha, and M. A. Porter, « Community structure in the United Nations General Assembly », *in*: *Physica A* 391.*1-2* (2012), pp. 343–361, DOI: 10.1016/j.physa.2011.06.030 (cit. on pp. 80, 86).

[158] T. L. Magnanti and R. T. Wong, « Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria », *in*: *Operations Research* 29.*3* (June 1981), pp. 464–484, DOI: 10.1287/opre.29.3.464 (cit. on p. 30).

[159] E. Marczewski and H. Steinhaus, « On a certain distance of sets and the corresponding distance of functions », *in*: *Colloquium Mathematicum* 6.*1* (1958), pp. 319–327 (cit. on p. 159).

[161] M. Meilă, « Comparing Clusterings by the Variation of Information », *in*: *Learning Theory and Kernel Machines*, ed. by Bernhard Schölkopf, Springer Berlin Heidelberg, 2003, pp. 173–187, ISBN: 978-3-540-45167-9, DOI: 10.1007/978-3-540-45167-9_14 (cit. on p. 50).

[162] Marina Meilă, « Comparing clusterings—an information based distance », *in*: *Journal of Multivariate Analysis* 98.*5* (2007), pp. 873–895, DOI: 10.1016/j.jmva.2006.11.013 (cit. on pp. 48, 50, 52, 54, 55, 63, 73, 74).

[163] Marina Meilă, « Criteria for comparing clusterings », *in*: *Handbook of cluster analysis*, ed. by C. Hennig, M. Meila, F. Murtagh, and R. Rocci, 1st Edition, Chapman and Hall/CRC, 2015, chap. 27, pp. 619–635, ISBN: 9780367570408 (cit. on pp. 48, 50, 54, 55, 66, 71, 72, 75, 159).

[164] I. Mendonça, R. Figueiredo, V. Labatut, and P. Michelon, « Relevance of Negative Links in Graph Partitioning: A Case Study Using Votes From the European Parliament », *in*: *2nd European Network Intelligence Conference*, 2015, pp. 122–129, DOI: 10.1109/ENIC.2015.25 (cit. on pp. 11, 87).

[165] R. E. Miller, « Purpose-Driven Communities In Multiplex Networks: Thresholding User-Eengagd Layer Aggregatation », MA thesis, Naval Postgraduate School, 2016 (cit. on p. 80).

[166] G. W. Milligan and M. C. Cooper, « A Study of the Comparability of External Criteria for Hierarchical Cluster Analysis », *in*: *Multivariate Behavioral Research* 21.*4* (1986), pp. 441–458, DOI: 10.1207/s15327906mbr2104_5 (cit. on p. 52).

[167] B. Mirkin, « Mathematical Classification and Clustering: From How to What and Why », *in*: *Classification, Data Analysis, and Data Highways*, ed. by I. Balderjahn, Springer Berlin Heidelberg, 1998, pp. 172–181, ISBN: 978-3-642-72087-1, DOI: 10.1007/978-3-642-72087-1_20 (cit. on p. 54).

[168] A. Miyauchi, T. Sonobe, and N. Sukegawa, « Exact Clustering via Integer Programming and Maximum Satisfiability », *in*: *AAAI Conference on Artificial Intelligence* 32.*1* (2018) (cit. on pp. 22, 25, 26).

[169] N. Mladenović and P. Hansen, « Variable neighborhood search », *in*: *Computers & Operations Research* 24.*11* (Nov. 1997), pp. 1097–1100, DOI: 10.1016/s0305-0548(97)00031-2 (cit. on pp. 35, 36).

[170] A. Mrvar and P. Doreian, « Partitioning Signed Two-Mode Networks », *in*: *Journal of Mathematical Sociology* 33.*3* (2009), pp. 196–221, DOI: 10.1080/00222500902946210 (cit. on p. 11).

[171] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, « Community Structure in Time-Dependent, Multiscale, and Multiplex Networks », *in*: *Science* 328.*5980* (May 2010), pp. 876–878, DOI: 10.1126/science.1184819 (cit. on p. 80).

[172] L. L. Nathans, F. L. Oswald, and K. Nimon, « Interpreting multiple linear regression: A guidebook of variable importance », *in*: *Practical Assessment, Research and Evaluation* 17.*9* (2012), pp. 1–19 (cit. on pp. 65, 66).

[173] Z. Néda, R. Sumi, M. Ercsey-Ravasz, M. Varga, B. Molnár, and G. Cseh, « Correlation clustering on networks », *in*: *Journal of Physics A: Mathematical and Theoretical* 42.*34* (Aug. 2009), p. 345003, DOI: 10.1088/1751-8113/42/34/345003 (cit. on pp. 34, 152).

[174] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*, Wiley, 1999, ISBN: 0-471-35943-2 (cit. on pp. 26, 28, 29, 108).

[175] M. E. J. Newman, G. T. Cantwell, and J. G. Young, « Improved mutual information measure for classification and community detection », *in*: *Phys. Rev. E* 101.*4* (2020), p. 042304, DOI: https://doi.org/10.1103/PhysRevE.101.042304 (cit. on pp. 51, 54, 55, 63, 74, 75).

[176] S. Nowozin and S. Jegelka, « Solution stability in linear programming relaxations », *in*: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML*, ACM Press, 2009, DOI: 10.1145/1553374.1553473 (cit. on pp. 17, 23, 27, 38, 43, 136).

[177] J. J. O'Brien, M. T. Lawson, D. K. Schweppe, and B. F. Qaqish, « Suboptimal Comparison of Partitions », *in*: *Journal of Classification* 37.*2* (2019), pp. 435–461, DOI: 10.1007/s00357-019-09329-1 (cit. on pp. 50, 54).

[178] M. Oosten, J. H. G. C. Rutten, and F. C. R. Spieksma, « The clique partitioning problem: Facets and patching facets », *in*: *Networks* 38.*4* (2001), pp. 209–226, DOI: 10.1002/net.10004 (cit. on p. 27).

[179] G. K. Orman, V. Labatut, and H. Cherifi, « Towards realistic artificial benchmark for community detection algorithms evaluation », *in*: *International Journal of Web Based Communities* 9.*3* (2013), p. 349, DOI: 10.1504/ijwbc.2013.054908 (cit. on p. 154).

[180] E. E. Papalexakis, L. Akoglu, and D. Ience, « Do more views of a graph help? Community detection and clustering in multi-graphs », *in*: *Proceedings of the 16th International Conference on Information Fusion*, 2013, pp. 899–905 (cit. on pp. 80, 81).

[181] C. Pape, T. Beier, P. Li, V. Jain, D. D. Bock, and A. Kreshuk, « Solving Large Multicut Problems for Connectomics via Domain Decomposition », *in*: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, IEEE, Oct. 2017, pp. 1–10, DOI: 10.1109/iccvw.2017.7 (cit. on p. 33).

[182] Q. Paris, « Multiple Optimal Solutions in Linear Programming Models », *in*: *American Journal of Agricultural Economics* 63.*4* (1981), p. 724, DOI: 10.2307/1241218 (cit. on p. 106).

[183] T. P. Peixoto, « Inferring the mesoscale structure of layered, edge-valued, and time-varying networks », *in*: *Physical Review E* 92.*4* (Oct. 2015), DOI: 10.1103/physreve.92.042807 (cit. on p. 80).

[184] D. Pfitzner, R. Leibbrandt, and D. Powers, « Characterization and evaluation of similarity measures for pairs of clusterings », *in*: *Knowledge and Information Systems* 19.*3* (2008), pp. 361–394, DOI: 10.1007/s10115-008-0150-6 (cit. on pp. 48, 51, 52, 54–56, 62, 63, 76, 77).

[185] E. Queiroga, A. Subramanian, R. Figueiredo, and Y. Frota, « Integer programming formulations and efficient local search for relaxed correlation clustering », *in*: *Journal of Global Optimization* (Feb. 2021), DOI: 10.1007/s10898-020-00989-7 (cit. on p. 14).

[186] R. Rabbany, M. Takaffoli, J. Fagnan, O. R. Zaïane, and R. J. G. B. Campello, « Communities validity: methodical evaluation of community mining algorithms », *in*: *Social Network Analysis and Mining* 3.*4* (2013), pp. 1039–1062, DOI: 10.1007/s13278-013-0132-x (cit. on pp. 48, 51, 52, 54–57, 61, 63, 67, 72, 74, 75, 77, 159).

[187] W. M. Rand, « Objective Criteria for the Evaluation of Clustering Methods », *in*: *Journal of the American Statistical Association* 66.*336* (1971), pp. 846–850, DOI: 10.1080/01621459.1971.10482356 (cit. on pp. 50, 51, 54–56, 66, 158).

[188] R. Reichart and A. Rappoport, « The NVI Clustering Evaluation Measure », *in*: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, Boulder, Colorado: Association for Computational Linguistics, 2009, pp. 165–173, ISBN: 978-1-932432-29-9, URL: http://dl.acm.org/citation.cfm?id=1596374.1596401 (cit. on p. 55).

[189] M. Rezaei and P. Fränti, « Set Matching Measures for External Cluster Validity », *in*: *IEEE Transactions on Knowledge and Data Engineering* 28.*8* (2016), pp. 2173–2186, DOI: 10.1109/tkde.2016.2551240 (cit. on pp. 48, 52–56, 61, 63, 72, 74, 75).

[190] M. Rocklin and A. Pinar, « On Clustering on Graphs with Multiple Edge Types », *in*: *Internet Mathematics* 9.*1* (Jan. 2013), pp. 82–112, DOI: 10.1080/15427951.2012.678191 (cit. on p. 82).

[191] S. Romano, J. Bailey, N. X. Vinh, and K. Verspoor, « Standardized Mutual Information for Clustering Comparisons: One Step Further in Adjustment for Chance », *in*: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, Beijing, China: JMLR.org, 2014, pp. II-1143–II-1151, URL: http://dl.acm.org/citation.cfm?id=3044805.3045020 (cit. on pp. 50, 51, 53, 54, 56).

[192] S. Romano, N. X. Vinh, J. Bailey, and K. Verspoorn, « Adjusting for Chance Clustering Comparison Measures », *in*: *Journal of Machine Learning Research* 17.*134* (2016), pp. 1–32, URL: http://jmlr.org/papers/v17/15-627.html (cit. on pp. 51, 53, 54, 73).

[193] Julia Bell Rosenberg Andrewand Hirschberg, « V-Measure: A conditional entropy-based external cluster evaluation », *in*: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (2007), pp. 410–420, DOI: 10.7916/d80v8n84 (cit. on pp. 54, 55).

[194] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer, « Optimizing Binary MRFs via Extended Roof Duality », *in*: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, June 2007, pp. 1–8, DOI: 10.1109/cvpr.2007.383203 (cit. on p. 33).

[195] Charlotte Rotman, *Comment le FN élargit sa base électorale [How FN enlarges its electoral base]*, ed. by Libération, Sept. 9, 2014, URL: www.liberation.fr/france/2014/09/09/comment-le-fn-elargit-sa-base-electorale_1095996 (visited on 11/08/2018) (cit. on p. 98).

[196] P. J. Rousseeuw, « Silhouettes: A graphical aid to the interpretation and validation of cluster analysis », *in*: *Journal of Computational and Applied Mathematics* 20 (Nov. 1987), pp. 53–65, DOI: 10.1016/0377-0427(87)90125-7 (cit. on pp. 75, 85).

[197] F. J. C. T. de Ruiter, R. C. M. Brekelmans, and D. den Hertog, « The impact of the existence of multiple adjustable robust solutions », *in*: *Mathematical Programming* 160.*1-2* (2016), pp. 531–545, DOI: 10.1007/s10107-016-0978-6 (cit. on p. 106).

[198] P. Saikko, J. Berg, and M. Järvisalo, « LMHS: A SAT-IP Hybrid MaxSAT Solver », *in*: *Theory and Applications of Satisfiability Testing - SAT 2016*, ed. by N. Creignou and D. Le Berre, Springer International Publishing, 2016, pp. 539–546, DOI: 10.1007/978-3-319-40970-2_34 (cit. on p. 107).

[199] G. Santamaria and V. Gomez, « Convex inference for community discovery in signed networks », *in*: *NIPS Workshop: Networks in the Social and Information Sciences*, 2015, p. 15, URL: http://web.stanford.edu/~jugander/NetworksNIPS2015/papers/NIPS%5C_Networks%5C_2015%5C_paper%5C_15.pdf (cit. on p. 87).

[200] A. Santra, S. Bhowmick, and S. Chakravarthy, « Efficient Community Re-creation in Multilayer Networks Using Boolean Operations », *in*: *Procedia Computer Science* 108 (2017), pp. 58–67, DOI: 10.1016/j.procs.2017.05.246 (cit. on p. 80).

[201] P. C. Saxena and K. Navaneetham, « The Effect of Cluster Size, Dimensionality, and Number of Clusters on Recovery of True Cluster Structure Through Chernoff-Type Faces », *in*: *Journal of the Royal Statistical Society (the Statistician)* 40.*4* (1991), pp. 415–425, DOI: 10.2307/2348731 (cit. on pp. 57, 58).

[202] P. Schittekat and K. Sorensen, « OR Practice-Supporting 3PL decisions in the automotive industry by generating diverse solutions to a large-scale location-routing problem », *in*: *Operations Research* 57.*5* (2009), 10.1287/opre.1080.0633, pp. 1058–1067 (cit. on p. 136).

[203] P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, « An ant colony approach for clustering », *in*: *Analytica Chimica Acta* 509.*2* (May 2004), pp. 187–195, DOI: 10.1016/j.aca.2003.12.032 (cit. on p. 36).

[204] M. C.P. de Souto, A. L.V. Coelho, K. Faceli, T. C. Sakata, V. Bonadia, and I. G. Costa, « A Comparison of External Clustering Evaluation Indices in the Context of Imbalanced Data Sets », *in*: *2012 Brazilian Symposium on Neural Networks*, ed. by IEEE Computer Society Press, IEEE, Oct. 2012, pp. 49–54, DOI: 10.1109/sbrn.2012.25 (cit. on p. 75).

[205] A. Strehl and J. Ghosh, « Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions », *in*: *Journal of Machine Learning Research* 3 (2002), Editor: Claire Cardie, pp. 583–617, URL: https://www.jmlr.org/papers/v3/strehl02a.html (cit. on pp. 48, 54, 67, 80, 82, 160).

[206] Y. Sun, H. Du, M. Gong, L. Ma, and S. Wang, « Fast computing global structural balance in signed networks based on memetic algorithm », *in*: *Physica A: Statistical Mechanics and its Applications* 415 (Dec. 2014), pp. 261–272, DOI: 10.1016/j.physa.2014.07.071 (cit. on p. 36).

[207] P. Swoboda and B. Andres, « A Message Passing Algorithm for the Minimum Cost Multicut Problem », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, July 2017, pp. 4990–4999, DOI: 10.1109/cvpr.2017.530 (cit. on p. 31).

[208] P. Swoboda, J. Kuske, and B. Savchynskyy, « A Dual Ascent Framework for Lagrangean Decomposition of Combinatorial Problems », *in*: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, July 2017, pp. 4950–4960, DOI: 10.1109/cvpr.2017.526 (cit. on p. 31).

[209] M. Szell, R. Lambiotte, and S. Thurner, « Multirelational organization of large-scale social networks in an online world », *in*: *Proceedings of the National Academy of Sciences* 107.*31* (July 2010), pp. 13636–13641, DOI: 10.1073/pnas.1004008107 (cit. on p. 81).

[210] A. Tagarelli, A. Amelio, and F. Gullo, « Ensemble-based community detection in multilayer networks », *in*: *Data Mining and Knowledge Discovery* 31.*5* (2017), pp. 1506–1543, DOI: 10.1007/s10618-017-0528-8 (cit. on p. 80).

[211] S. Tan and J. Lü, « An evolutionary game approach for determination of the structural conflicts in signed networks », *in*: *Scientific Reports* 6.*1* (2016), p. 22022, DOI: 10.1038/srep22022 (cit. on p. 34).

[212] L. Tang, X. Wang, and H. Liu, « Community detection via heterogeneous interaction analysis », *in*: *Data Mining and Knowledge Discovery* 25.*1* (Aug. 2011), pp. 1–33, DOI: 10.1007/s10618-011-0231-0 (cit. on p. 80).

[213] W. Tang, Z. Lu, and S. I. Dhillon, « Clustering with Multiple Graphs », *in*: *2009 Ninth IEEE International Conference on Data Mining*, 2009, pp. 1016–1021, DOI: 10.1109/ICDM.2009.125 (cit. on p. 81).

[216] V. A. Traag and J. Bruggeman, « Community detection in networks with positive and negative links », *in*: *Physical Review E* 80.*3* (Sept. 2009), p. 036115, DOI: 10.1103/physreve.80.036115 (cit. on p. 34).

[217] V. A. Traag, G. Krings, and P. van Dooren, « Significant Scales in Community Structure », *in*: *Scientific Reports* 3.*1* (2013), DOI: 10.1038/srep02930 (cit. on p. 87).

[218] J. Trusty, B. Thompson, and J. V. Petrocelli, « Practical Guide for Reporting Effect Size in Quantitative Research in the Journal of Counseling & Development », *in*: *Journal of Counseling & Development* 82.*1* (2004), pp. 107–110, DOI: 10.1002/j.1556-6678.2004.tb00291.x (cit. on p. 65).

[219] S. Vega-Pons and J. Ruiz-Shulcloper, « A survey of clustering ensemble algorithms », *in*: *International Journal of Pattern Recognition and Artificial Intelligence* 25.*03* (May 2011), pp. 337–372, DOI: 10.1142/s0218001411008683 (cit. on p. 83).

[220] Nate Veldt, Anthony I. Wirth, and David F. Gleich, « Correlation Clustering with Low-Rank Matrices », *in*: *Proceedings of the 26th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, Apr. 2017, DOI: 10.1145/3038912.3052586 (cit. on p. 22).

[222] N. X. Vinh, J. Epps, and J. Bailey, « Information theoretic measures for clusterings comparison: Is a Correction for Chance Necessary? », *in*: *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, ACM Press, 2009, pp. 1073–1080, DOI: 10.1145/1553374.1553511 (cit. on pp. 51, 53, 54).

[223] N. X. Vinh, J. Epps, and J. Bailey, « Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance », *in*: *Journal of Machine Learning Research* 11.*95* (2010), pp. 2837–2854 (cit. on pp. 48–51, 54, 67, 71, 74, 75, 160).

[224] A. Vörös and T. A. B. Snijders, « Cluster analysis of multiplex networks: Defining composite network measures », *in*: *Social Networks* 49 (May 2017), pp. 93–112, DOI: 10.1016/j.socnet.2017.01.002 (cit. on p. 81).

[225] S. Wagner and D. Wagner, *Comparing clusterings: an overview*, tech. rep., Universität Karlsruhe, 2007 (cit. on pp. 48, 50–52, 54, 66).

[226] M. J. Warrens and H. van der Hoef, *Understanding partition comparison indices based on counting object pairs*, tech. rep., Groningen Institute for Educational Research, 2019, arXiv: 1901.01777 [stat.ML] (cit. on pp. 53, 54).

[227] A. S. Waugh, L. Pei, J. H. Fowler, P. J. Mucha, and M. A. Porter, « Party Polarization in Congress: A Network Science Approach », *in*: *arXiv e-prints*, arXiv:0907.3509 (July 2009), arXiv:0907.3509, arXiv: 0907.3509 (cit. on p. 86).

[228] L. A. Wolsey, *Integer Programming*, ed. by R. L. Graham, J. K. Lenstra, and R. E. Tarjan, Wiley-Interscience, 1998 (cit. on p. 28).

[229] J. Wu, H. Xiong, and J. Chen, « Adapting the right measures for K-means clustering », *in*: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, ACM Press, 2009, pp. 877–886, DOI: 10.1145/1557019.1557115 (cit. on pp. 50, 52, 54, 57).

[230] Q. Xiang, Q. Mao, K. M. A. Chai, H. L. Chieu, I. W.-H. Tsang, and Z. Zhao, « A Split-merge Framework for Comparing Clusterings », *in*: *Proceedings of the29th International Conference on Machine Learning*, ICML'12, Omnipress, 2012, pp. 1259–1266, URL: http://dl.acm.org/citation.cfm?id=3042573.3042735 (cit. on pp. 52, 54–56).

[231] J. Yarkony, T. Beier, P. Baldi, and F. A. Hamprecht, « Parallel Multicut Segmentation via Dual Decomposition », *in*: *New Frontiers in Mining Complex Patterns*, ed. by A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, and Z. W. Ras, Springer International Publishing, 2015, pp. 56–68, DOI: 10.1007/978-3-319-17876-9_4 (cit. on p. 24).

[232] P. Zhang, « Evaluating accuracy of community detection using the relative normalized mutual information », *in*: *Journal of Statistical Mechanics: Theory and Experiment* 2015.*11* (2015), P11006, URL: https://iopscience.iop.org/article/10.1088/1742-5468/2015/11/P11006 (cit. on pp. 51, 53, 54).

[233] S. Zhang, Z. Yang, X. Xing, Y. Gao, D. Xie, and H.-S. Wong, « Generalized Pair-Counting Similarity Measures for Clustering and Cluster Ensembles », *in*: *IEEE Access* 5 (2017), pp. 16904–16918, DOI: 10.1109/access.2017.2741221 (cit. on pp. 51, 54, 57).

Secondary sources

[6] Alliance of Liberals and Democrats for Europe (ALDE), *ALDE priorities for CAP reform post - 2013*, tech. rep., retrieved September, 2018, URL: www.meng-landwirtschaft.lu/fileadmin/files/meng-landwirtschaft/CAP_reform_ALDE.pdf (visited on 11/08/2018) (cit. on pp. 89, 99).

[7] Alliance of Liberals and Democrats for Europe (ALDE), *ASPARTAME: The European Health authorities play a fools game*, Mar. 2013, URL: www.alde.eu/press/press-and-release-news/press-release/article/alde-meps-call-for-an-investigation-into-serious-health-concerns-over-the-aspartame-40971/ (visited on 11/08/2018) (cit. on p. 98).

[73] European Commission, *CAP In Your Country: France*, Sept. 2016, URL: www.ec.europa.eu/info/sites/info/files/food-farming-fisheries/by_country/documents/cap-in-your-country-fr_en.pdf (cit. on p. 88).

[74] European Commission, *CAP In Your Country: Italy*, Sept. 2016, URL: www.ec.europa.eu/info/sites/info/files/food-farming-fisheries/by_country/documents/cap-in-your-country-it_en.pdf (cit. on p. 88).

[75] European Commission, *Direct payments*, retrieved september, 2018, URL: www.ec.europa.eu/agriculture/direct-support/direct-payments_en (cit. on p. 97).

[76] European Commission, *Export refunds for processed agricultural products*, retrieved september, 2018, URL: www.ec.europa.eu/growth/sectors/food/processed-agricultural-products/export-refunds_en (cit. on p. 99).

[77] European Commission, *Glossary of terms related to the Common Agricultural Policy*, Apr. 2015, URL: www.ec.europa.eu/agriculture/glossary_en (cit. on pp. 164, 165).

[78] European Commission, *Greening*, retrieved september, 2018, URL: www.ec.europa.eu/agriculture/direct-support/greening_en (cit. on pp. 99, 165).

[79] European Commission, *Overview of CAP Reform 2014 - 2020*, Dec. 2013, URL: www.eige.europa.eu/resources/05_en.pdf (visited on 11/08/2018) (cit. on p. 165).

[80] European Commission, *The end of milk quotas*, Mar. 2015, URL: www.ec.europa.eu/agriculture/milk-quota-end_en (cit. on pp. 98, 165).

[81] European Council, *Reform of the Common Agricultural Policy*, retrieved September, 2018, URL: www.consilium.europa.eu/en/policies/cap-reform/ (cit. on pp. 88, 164, 165).

[82] European Parliament, *Rules of procedure*, F, July 2010, URL: www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML+RULES-EP+20100705+0+DOC+PDF+V0//EN (cit. on p. 87).

[83] European Parliament, *The revised written translation of speeches retrieved from regulations related to 2011/0280(COD), 2011/0281(COD), 2011/0282(COD), 2011/0288(COD)*, Mar. 12, 2013, URL: www.europarl.europa.eu/sides/getDoc.do?type=CRE&reference=20130312&secondRef=ITEM-014&language=EN&ring=B7-2013-0079 (cit. on pp. 89, 98, 99).

[84] European United Left/Nordic Green Left (GUE-NGL), *Supporting small farmers, consumers and the environment: A plan for a fair CAP reform*, tech. rep., 2011, URL: www.guengl.eu/uploads/_old_cms_files/leaflet-agri-EN-web.pdf (visited on 11/08/2018) (cit. on p. 89).

[85] European Views, ed., *CAP Reform – Paolo De Castro MEP (S&D) [interview, video]*, Feb. 26, 2013, URL: www.european-views.com/videos/cap-reform-paolo-de-castro-mep-sd/ (visited on 11/08/2018) (cit. on p. 97).

[99] B. Gollnisch, *Oral intervention retrieved from the regulation related to 2011/0280(COD)*, ed. by European Parliament, Mar. 14, 2013, URL: http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+CRE+20130314+ITEM-009-10+DOC+XML+V0//EN&language=fr&query=INTERV&detail=4-223-750 (cit. on p. 89).

[106] Group of the European People's Party (EPP), *The new CAP: for more modern and efficient agriculture*, Nov. 2013, URL: www.eppgroup.eu/newsroom/news/the-new-cap-for-more-modern-and-efficient-agriculture (cit. on p. 89).

[117] IBM, *IBM ILOG CPLEX 12.8 User Manual IBM Corporation*, 2018 (cit. on pp. 107, 109).

[144] M. Le Pen, *Oral intervention retrieved from the regulation related to 2013/0117(COD)*, ed. by European Parliament, Nov. 20, 2013, URL: www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+CRE+20131120+ITEM-004+DOC+XML+V0//EN&language=fr&query=INTERV&detail=3-041-000 (cit. on p. 89).

[160] Alan Matthews, *More supply management demanded in COMAGRI single CMO report [Blog post]*, June 12, 2012, URL: www.capreform.eu/more-supply-management-demanded-in-comagri-single-cmo-report/ (visited on 11/08/2018) (cit. on p. 165).

[215] The International Federation of Organic Agriculture Movements, Europe (IFOAM EU), *The inside view of the CAP debate in the European Parliament*, Sept. 2012, URL: www.ifoam-eu.org/sites/default/files/event/files/ifoameu_event_biofach_martin_haeusling_presentation_201202.pdf (cit. on pp. 89, 99).

[221] P. de Villiers, *Oral intervention retrieved from the regulation related to 2011/0280(COD)*, ed. by European Parliament, Mar. 14, 2013, URL: www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+CRE+20130314+ITEM-009-10+DOC+XML+V0//EN&language=fr&query=INTERV&detail=4-222-000 (cit. on p. 89).

AVIGNON UNIVERSITÉ

Multiplicity in the Partitioning of Signed Graphs

by
Nejat ARINIK

LABORATOIRE INFORMATIQUE D'AVIGNON