

DEPARTMENT OF INFORMATICS
UNIVERSITY OF FRIBOURG (SWITZERLAND)

**Unsupervised and Parameter-Free Clustering
of Large Graphs for Knowledge Exploration
and Recommendation**

THESIS

presented to the Faculty of Science and Medicine of the University of Fribourg (Switzerland)
in consideration for the award of the academic grade of
Doctor of Philosophy in Computer Science

by

ARTEM LUTOV

from

UKRAINE

Thesis No: 2192

UniPrint

2020

Accepted by the Faculty of Science and Medicine of the University of Fribourg (Switzerland)
upon the recommendation of Prof. Dr. Claudio J. Tessone, Dr. Mourad Khayati and Prof. Dr.
Ulrich Ultes-Nitsche (president of the jury).

Fribourg, May 13, 2020

Thesis supervisor



Prof. Dr. Philippe Cudré-Mauroux

Dean



Prof. Dr. Gregor Rainer

Declaration of Authorship

Title: Unsupervised and Parameter-free Clustering of Large Graphs for Knowledge Exploration and Recommendation.

I, Artem Lutov, declare that I have authored this thesis independently, without illicit help, that I have not used any other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Signed:



Date: 17.03.2020

We cannot eliminate inequality or abuse of power, but through technological inclusion
we can help transfer power into the hands of individual people.

— Eric Schmidt and Jared Cohen [210]

Acknowledgements

A doctoral thesis is an amazing journey, which provides a unique opportunity to study and enrich the field of your interest rather than following a path chosen by others in the industry. This journey assumes and even encourages multiple failures on the selected path beside the achievements that you collect along the way, which are just the tip of the iceberg. I would not have gone far along this way without the people around me, who helped me overcome all those pitfalls, who supervised and supported me. I would like to thank them all.

First, I would like to thank my supervisor, Prof. Dr. Philippe Cudré-Mauroux, who made this experience possible by welcoming me to his laboratory, encouraging my work, and organizing a perfect environment to accomplish it. I am especially grateful to Philippe for the provided opportunity to complete the thesis despite the long time it took. He always had great patience, an individual approach to each of his students, and guided us along the way without forcing us to change the direction we chose. Philippe has organized a very welcome and friendly atmosphere in our lab, always supported us far beyond our daily research activities, making it possible to overcome all the issues along the way and to enjoy our research.

I'm very grateful to Mourad and Dingqi, who devoted significant efforts to help me in my research and, especially, in the presentation of my results. I learned mostly from Mourad that the perceived information by a reader may substantially differ from the intended meaning if insufficient attention is paid to ambiguity elimination in the presented work. Mourad, Philippe and Dingqi have devoted an enormous amount of time polishing my ideas into human-readable and concise representations. I would not have been able to attend even a single conference where our works were published without their immense support.

Many thanks to the former XI members, Elisabeth, Sylviane and other administrative staff with whom I started my research and who helped me at the hardest time in various aspects. Gianluca, Roman, Michael and Alberto advised and helped me in various living aspects, which are one of the major pre-condition to be productive at work and, especially, in research. I am very grateful for the support and engaging philosophical discussions at the XI space with the current XI members. And I am sorry for missing too many great parties and social gatherings within the second half of my PhD study; it was the price to pay to start developing startups in addition to the research conducted at the university.

Acknowledgements

I'm thankful to Dr. Alexey Boyarsky, who literally paved the way for my PhD in Switzerland, and who had a major impact on the research direction that I took. I liked the idea of conducting research in the field of Big Data analysis and clustering in particular, but would not have assumed that it goes beyond the development of the required clustering algorithm. My arrival in Switzerland would not have been possible without the support of Alexey and Oleg. Also, I would like to thank Dr. Paolo De Los Rios, Alessio, Andrea, Alex, Vasil, Andrii and the rest of the ScienceWISE team for their advice and mutual exchanges at the very beginning of my PhD.

I would like to thank everybody who helped and accompanied me on this fantastic journey. Especially, I appreciate the support of my wife, Iaroslava, and of my family and relatives. Above all, the continuous efforts of my grandparents made it possible for me to obtain a great education which, finally, led to me coming to Switzerland. They passed away during my studies here but they live forever in my heart.

Fribourg, 24 February 2020

Artem Lutov

Abstract

We live in an Information Age, facing a rapid increase in the amount of information that is exchanged. This permanently growing amount of data makes the ability to store, analyze, and act upon information a primary concern (in addition to the obvious privacy, legal and ethical issues that are related), raising the question: “How can one consume Big Data and transform it into actionable knowledge?”. Knowledge in the field of Information Systems can be characterized as explicit and processible information in a given context; information can be defined as data organized in a meaningful way, whereas data represents raw facts. Any complex system can be represented as a graph, which is human-readable for individuals without any mathematical background. Graph clustering provides a way to automatically identify complex structures and build a taxonomy, transforming information into knowledge. Taxonomies enable searching for specific knowledge items by navigating a hierarchy of concepts (i.e., topics) without knowing the exact title or original terminology of the item itself. However, neither the availability of comprehensive but also overwhelming information by itself nor its taxonomy are sufficient for effective information search, knowledge exploration and recommendation. The crucial aspect here is the *approach to transform the abundance of the retrieved information in a usable (i.e., human-adapted) and efficient way*.

In this thesis, we first overview existing research on human perception and visual analytics in Chapter 1 and Chapter 2, which leads to a set of criteria according to which the information should be transformed into knowledge to become human-readable while reflecting multiple viewpoints. Then, in Chapter 3, we discuss existing approaches to construct such a transformation and propose our novel clustering algorithm, DAOC. Our method efficiently produces a human-adapted hierarchy of clusters (i.e., a taxonomy) for any input graph without any manual tuning and is on average 25% more accurate than the state-of-the-art analogs. The former feature is especially important for its direct applicability and seamless integration into various data mining pipelines as shown in Chapter 6 and 7. To ensure that the results are accurate besides being human-readable, in Chapter 4 we analyze existing approaches of the clustering results validation, taking into account the criteria formulated in Chapter 2. In addition, we extend the existing accuracy measures in Chapter 4, yielding both a) the implementation of the Mean F1 score evaluation that works faster on a single CPU than state-of-the-art implementations running on high-performance servers and b) implementations of the Generalized NMI and Omega Index that address certain constraints of the original versions. In Chapter 5, we introduce Clubmark, a new framework for benchmarking and profiling clustering algorithms on NUMA architectures. Clubmark is designed for a long-term

and crash-resistant benchmarking, and provides both console and web interfaces for process inspection. Our framework includes a dozen of state-of-the-art clustering algorithms and evaluates them on standard real-world and synthetic datasets avoiding biases (in terms of both the order of items and particular statistical distributions in a dataset). In Chapter 6, we propose an effective and efficient approach, StaTIX, to infer and complete entity types in Linked Open Data (including knowledge graphs) based on our DAOC clustering algorithm. In Chapter 7, we bridge a gap between graph clusters (also known as network communities) and a low-dimensional representation of the graph nodes, presenting our novel graph embedding technique, DAOR. DAOR is a highly efficient and parameter-free graph embedding technique producing metric space-robust, compact and interpretable embeddings without any manual tuning. Compared to a dozen state-of-the-art graph embedding algorithms, DAOR yields competitive results on diverse graph analysis tasks, while being several orders of magnitude more efficient. Our approach has the ambition to greatly simplify and speed up data analysis tasks involving graph representation learning (e.g., information recommendation and search). We conclude the thesis by suggesting the integration of our innovative clustering method, DAOC, into Visual Analytics frameworks and data search interfaces in Chapter 8. In the future, our approach may bridge a gap in Visual Analytics tools between a coarse-grained view on data (e.g., using the UMAP or t-SNE techniques) and a fine-grained inspection of selected points (e.g., GLC techniques). In the context of search interfaces, our approach might provide a higher diversity of the results without sacrificing their accuracy, making users more comprehensively informed on a topic (i.e., considering multiple view-points on the issue) and, hopefully, reducing social polarization.

Key words: Cluster Analysis, Cluster Stability, Interpretable Graph Embedding, Clustering Evaluation, Accuracy Metrics, Community Structure Discovery, Network Clustering, Visual Analytics, Knowledge Exploration.

Résumé

Nous vivons une ère où la quantité d'informations échangées augmente rapidement. Cette quantité croissante de données fait que la capacité de stocker, d'analyser et d'agir sur les informations une préoccupation principale (en plus des problèmes apparents de confidentialité, juridiques et éthiques qui y sont liés), soulevant la question : "Comment peut-on consommer des Big Data et la transformer en connaissances exploitables?". Les connaissances dans le domaine des systèmes d'information peuvent être caractérisées comme des informations explicites et traitables dans un contexte donné ; les informations peuvent être définies comme des données organisées de manière significative, tandis que les données représentent des faits bruts. Tout système complexe, y compris les informations elles-mêmes, à une représentation universelle sous forme de graphique, qui est à la fois mathématiquement strict et lisible par l'Homme, même sans aucune compétence préalable. Le clustering de graphes permet d'identifier automatiquement des structures complexes et de construire une taxonomie, transformant les informations de traitement en connaissances. Les taxonomies permettent de rechercher des éléments de connaissances spécifiques en parcourant une hiérarchie de concepts (c'est-à-dire des sujets) sans connaître le titre exact ou la terminologie originale de l'élément lui-même. Cependant, ni la disponibilité d'informations complètes ni sa taxonomie ne sont suffisantes pour une recherche d'informations, une exploration des connaissances et des recommandations efficaces. L'aspect crucial ici est *l'approche pour transformer l'abondance des informations récupérées d'une manière utilisable* (c'est-à-dire adaptée à la perception humaine) *et efficace*.

Dans cette thèse, nous commençons par résumer les recherches récentes sur la perception humaine et l'analyse visuelle. Ces recherches formulent les critères, selon lesquels l'information doit être transformée en connaissances pour devenir lisibles par l'homme tout en reflétant plusieurs points de vue. Ensuite, dans le chapitre 3, nous discutons des approches existantes pour construire une telle transformation et proposons notre nouvel algorithme de clustering, DAOC. Notre méthode produit efficacement une hiérarchie de groupes adaptée à la perception humaine (c'est-à-dire une taxonomie) pour tout graphe sans aucun réglage manuel et est en moyenne 25% plus précis que les méthodes analogues. Le fait que notre approche ne nécessite aucune intervention manuelle est particulièrement important pour son applicabilité et son intégration dans divers pipelines d'exploration de données, comme indiqué dans les chapitres 6 et 7. Pour garantir l'exactitude des résultats en plus d'être lisibles par l'homme, au chapitre 4, nous analysons les approches de la validation des résultats du clustering, en tenant compte des critères formulés au chapitre 2. De plus, nous étendons les mesures de précision

existantes au chapitre 4, produisant à la fois a) la mise en œuvre de l'évaluation du score moyen F1 qui fonctionne plus rapidement sur un seul processeur que des implémentations de pointe fonctionnant sur des serveurs hautes performances et b) des implémentations de NMI généralisé et Omega Index qui répondent à certaines contraintes des versions originales. Dans le chapitre 5, nous présentons Clubmark, un nouveau cadre pour l'analyse comparative et le profilage des algorithmes de clustering sur les architectures NUMA. Clubmark est conçu pour une analyse comparative à long terme et résistante aux chocs, et fournit à la fois des interfaces de console et Web pour l'inspection des processus. Notre cadre comprend une douzaine d'algorithmes de clustering de pointe (les implémentations originales par les auteurs d'algorithmes) et les évalue sur des ensembles de données standard et synthétiques évitant les biais (en termes d'ordre des éléments et de distributions statistiques particulières dans un ensemble de données). Dans le chapitre 6, nous proposons une approche efficace, StaTIX, et rapide pour déduire et compléter des types d'entités dans des données ouvertes liées (y compris des graphiques de connaissances) sur la base de notre algorithme de clustering DAOC. Dans le chapitre 7, nous comblons un fossé entre les groupes de graphes (également appelées communautés de réseaux) et une représentation à basse dimension des nœuds de graphe, présentant notre nouvelle technique d'intégration de graphes, DAOR. DAOR est une technique d'intégration graphique très efficace et sans paramètres produisant des intégrations métriques robustes, compactes et interprétables sans aucun réglage manuel. Comparé à une douzaine d'algorithmes d'intégration de graphiques de pointe, DAOR donne des résultats compétitifs sur diverses tâches d'analyse de graphiques, tout en étant plus efficace de plusieurs ordres de grandeur. Notre approche a pour ambition de simplifier et d'accélérer considérablement les tâches d'analyse de données impliquant l'apprentissage de la représentation graphique (par exemple, la recommandation et la recherche d'informations).

Nous concluons la thèse suggérant l'intégration de notre méthode innovante de clustering, DAOC, dans les frameworks Visual Analytics et les interfaces de recherche de données. À l'avenir, notre approche pourrait combler un fossé dans les outils Visual Analytics entre la vue à granularité grossière sur les données (par exemple, les techniques UMAP et t-SNE) et l'inspection à grain fin des points sélectionnés (par exemple, les techniques GLC). Dans le contexte des interfaces de recherche, notre approche fournira une plus grande diversité des résultats de livraison qui ne s'échangent pas avec leur précision, ce qui informera les utilisateurs de manière exhaustive (c'est-à-dire en considérant plusieurs points de vue sur le problème) et, espérons-le, réduira la polarisation sociale.

Mots clefs : Analyse de Cluster, Stabilité de Cluster, Intégration de Graphiques Interprétables, Évaluation de Clustering, Mesures de Précision, Découverte de la Structure de la Communauté, Clustering Réseau, Analyse Visuelle, Exploration des Connaissances.

Contents

Acknowledgements	i
Abstract (English/Français)	iii
List of figures	xi
List of tables	xiii
1 Introduction	1
1.1 Research Questions	6
1.2 Contributions	6
1.2.1 Stable clustering of large networks without manual tuning	6
1.2.2 Accuracy measures for overlapping and multi-resolution clustering . . .	7
1.2.3 Benchmarking of stable, overlapping and multi-resolution clustering . .	7
1.2.4 Entity type completion in knowledge graphs via clustering	8
1.2.5 Fully automatic and interpretable graph embedding via clustering	8
1.2.6 What this Thesis is Not About	9
1.2.7 Thesis Outline	9
2 Background	11
2.1 Human Perception of Information	11
2.2 Knowledge Representation	12
2.3 Clustering for Information Granulation	13
2.3.1 Network Clustering	13
2.3.2 Clustering Techniques	14
2.3.3 Clustering Objective	17
2.3.4 Clustering Validation	19
2.4 Information Visualization and Analytics	20
2.4.1 Visualization	20
2.4.2 Visual Analytics	20
2.5 Visualization Techniques for Multidimensional Data	21
2.5.1 Lossless Techniques for Multi-dimensional Data Visualization	22
2.5.2 Lossy Techniques for Multidimensional Data Visualization	23
2.6 Research Focus	27
	vii

2.7	Notations	28
3	DAOC: Stable Clustering of Large Networks Without Manual Tuning	31
3.1	Introduction	31
3.2	Related Work	32
3.3	Method	34
3.3.1	Identification of the Clustering Candidates	36
3.3.2	Clusters Formation with Overlap Decomposition	38
3.3.3	Complexity Analysis	42
3.4	Experimental Evaluation	42
3.4.1	Evaluation Environment	42
3.4.2	Stability Evaluation	43
3.4.3	Effectiveness and Efficiency Evaluation	44
3.5	Conclusions	48
4	Xmeasures: Accuracy Measures for Overlapping and Multi-resolution Clustering	49
4.1	Introduction	49
4.2	Related Work	51
4.3	Accuracy Metrics for Overlapping Clustering	52
4.3.1	Omega Index	53
4.3.2	Mean F1 Score	56
4.3.3	Generalized NMI	59
4.3.4	Discussion	62
4.4	Conclusions	63
5	Clubmark: Benchmarking of Stable, Overlapping and Multi-resolution Clustering	65
5.1	Introduction	65
5.2	Related Work	66
5.3	System Description	68
5.3.1	Framework Core	68
5.3.2	Input Data	69
5.3.3	Clustering Algorithms	70
5.3.4	Web Interface	71
5.3.5	Evaluation Measures	73
5.3.6	Evaluation & Results	73
5.4	Discussion and Conclusions	74
6	StaTIX: Entity Type Completion in Knowledge Graphs via Clustering	77
6.1	Introduction	77
6.2	Related Work	78
6.3	Method Overview	80
6.3.1	Problem Statement	80
6.3.2	Unsupervised Statistical Type Inference	80

6.3.3	Similarity Matrix Construction	81
6.4	Type Inference	82
6.4.1	Weight-Preserving Reduction of the Similarity Matrix	82
6.4.2	Clustering	85
6.4.3	Representative Clusters Identification at Multiple Scales	87
6.5	Evaluation	88
6.5.1	Metrics	89
6.5.2	Datasets	89
6.5.3	Results and Discussion	90
6.6	Conclusions and Future Work	93
7	DAOR: Fully Automatic and Interpretable Graph Embedding via Clustering	95
7.1	Introduction	95
7.2	Related Work	97
7.2.1	Graph embedding	97
7.2.2	Community detection for graph embedding	98
7.3	Transforming Clusters into Node Embeddings	99
7.3.1	Feature Extraction from Clusters	99
7.3.2	Dimension Formation from Features	101
7.4	Community Detection	102
7.4.1	Hierarchical multi-resolution clustering	103
7.4.2	Bounding the number of clusters	106
7.5	Experimental Evaluation	107
7.5.1	Experimental Setup	107
7.5.2	Node Classification Task	108
7.5.3	Link Prediction Task	109
7.5.4	Robustness to the Metric Space	110
7.5.5	Runtime Performance	111
7.6	Conclusions	112
8	Conclusions	113
8.1	Summary of Findings	113
8.2	Future Work	115
8.2.1	Outlook	115
8.2.2	Addressing the Limitations of our Research Findings	116
8.2.3	DAOC Extension for Attributed and Hyper-Graphs	117
8.2.4	Visual Exploration and Analytics Based on a Hierarchy of Clusters	118
8.2.5	Social Impact of Fairness and Diversity in Recommendation Systems	122
	Bibliography	123
	Curriculum Vitae	

List of Figures

1.1	The evolution of knowledge management	2
1.2	Digital knowledge discovery and exploration systems	3
1.3	The visual knowledge exploration interface of ScienceWISE	5
2.1	Multi-perspective and context-dependent human perception	12
2.2	K-Means clustering of non-convex shapes	16
2.3	Shapes with various densities	16
2.4	Formal constraints for the accuracy measures.	19
2.5	Hybrid visualization of n D data	22
2.6	Big Data in parallel coordinates	23
2.7	Mapping from parallel coordinates to GLCs	24
2.8	Visualization pitfalls of t-SNE	26
2.9	Visualization by UMAP vs t-SNE	27
2.10	Unsupervised UMAP visualization	27
3.1	Overlap decomposition	40
3.2	Stability and sensitivity evaluation.	45
3.3	F1h of the deterministic algorithms on the synthetic networks.	46
3.4	Performance on the real-world networks.	47
4.1	Formal constraints for the accuracy measures.	52
4.2	Xmeasures MF1 vs ParallelComMetric F1-Measure	59
5.1	Clubmark architecture.	68
5.2	Top utility listing the executing processes during benchmarking.	70
5.3	WebUI quired from a modern browser	72
5.4	WebUI quired from a console browser	72
6.1	Unsupervised Type Inference Process in StaTIX	80
6.2	Reduction candidates identification	85
6.3	Representative vs filtered out clusters	88
6.4	Type inference accuracy evaluation by F1h	91
6.5	Impact of the similarity matrix reduction on the accuracy	91
6.6	Memory consumption of type inference algorithms	93

List of Figures

6.7	Execution time of type inference algorithms	93
6.8	Impact of the similarity matrix reduction on the execution time	93
7.1	Transformation of clusters into node embeddings	99
7.2	Overlapping hierarchical clustering vs multi-resolution clustering	104
8.1	InFoGViC visualization	119
8.2	InFoGViC solving t-SNE pitfalls	120
8.3	IntEVA framework for VA of Big Data	121

List of Tables

2.1	Our objectives: mapping human perception onto clustering requirements . . .	15
2.2	Notations	29
3.1	Evaluating clustering algorithms.	42
3.2	Peak memory consumption (RSS) on the real-world networks, MB.	47
4.1	Omega Index vs Soft Omega Index	55
4.2	Formal Constraints for Soft and original Omega Index.	55
4.3	Formal Constraints for MF1 metric family.	58
4.4	Formal Constraints for NMI, original GNMI and our GNMI.	61
4.5	Formal Constraints for Omega Index, MF1 and GNMI.	63
5.1	Clustering algorithms included in Clubmark.	71
6.1	Evaluation Datasets.	90
6.2	Accuracy of the labeled types	92
7.1	Characteristics of the experimental graphs	107
7.2	Node classification performance	109
7.3	Link prediction performance	110
7.4	Node embedding robustness to the metric space	111
7.5	Learning time of node embedding	111

1 Introduction

We live today in an *Information Age* [178, 210] facing a rapid increase in the amount of information that is exchanged. One of the effects of this abundance is known as *information explosion*. The proliferation of computation and communication technologies causing that explosion shape our daily life affecting not only human interactions but also our current work landscape, healthcare, education, human rights and government, among others. Our digital universe doubles at least every two years, yielding a 50-fold growth within the past decade [58]. This overwhelming yet still growing amount of data makes the ability to store, analyze, and act upon information a primary concern (in addition to the obvious privacy, legal and ethical issues that are related), raising the question: “*How can one consume Big Data and transform it into actionable knowledge?*”. To answer this question, first we clarify the terminology and provide a brief introduction to past prevailing knowledge management techniques, illustrated in Figure 1.1, and to their evolution in the digital realm focusing on knowledge exploration techniques shown in Figure 1.2; then, we give an overview of more recent paradigms and outline their shortcomings. This work aims to address some of the fundamental shortcomings on the current state-of-the-art knowledge exploration platforms (i.e., related to search, recommendation and visual analytics systems) and their underlying knowledge graphs.

Knowledge in the field of Information Systems can be characterised as explicit and processible information in a given context; information can be defined as data organized in a meaningful way, whereas data represents raw facts [223, 280]. In retrospect, humanity has tackled the knowledge management issue for millennia, by storing available knowledge (e.g., clay tablets, papyrus scrolls, manuscripts, books and other artifacts) in centralized archives (e.g., libraries and museums). All knowledge artifacts were in the past manually sorted alphabetically, by time, and cataloged. With the growth of available knowledge, humanity invented *taxonomies*, whose origins can be traced to ancient China around 3000 BC [149]. Taxonomies enable searching for specific knowledge items by navigating a hierarchy of concepts (i.e., topics) without knowing the exact title or original terminology of the item itself. The first evidences on *glossaries*, giving series of definitions of terms in a particular domain of knowledge, are dated

Chapter 1. Introduction



Figure 1.1 – The evolution of knowledge management: 1. The Babylonian glossary “Urra=hubullu” of Sumerian and Akkadian lexical lists, 2nd ML BC; 2. A fragment of Chinese herbs taxonomy from Pen-Tsao Kang-mu, 1578 AD (the origins are traced to 3rd ML BC); 3. Papyrus scrolls cataloging in the Library of Alexandria, 250 BC; 4. Encyclopædia Britannica, 1768; 5. A catalogue of books and prints, 1779 (published since 1564); 6. A contemporary physical cataloging system of libraries: 6.1. Rules for a printed dictionary catalogue, 1875; 6.2. The book “Card Catalog” listing books, cards, and literary treasures from the Library of Congress, 2017; 6.3. A sheaf catalogue commonly used till the 20th century.

around the same time. Glossaries were the precursors of encyclopedias, which summarize the state of existing knowledge in a structured manner [40]. The evolution of legacy knowledge management systems is briefly illustrated in Figure 1.1.

With the emergence of the World Wide Web (WWW)¹ [16, 115] in the 1990s, the dawn of the Internet era and the digitization of the existing knowledge (e.g., project Gutenberg [81], Wikipedia [6]), the historical methodology for knowledge management evolved as follows. Initially, various digital catalogs known as web directories were created, where existing web-sites were manually registered, indexed, taxonomically classified and visually provided for exploration and navigation (e.g., through websites such as VLib, Yahoo! Directory, Craigslist, or DMOZ). Manual exploration of multiple web directories were time consuming, ineffective and hardly efficient to find any specific information. The need for a more efficient approach propelled the emergence of the first a) search engines for global knowledge discovery (e.g.,

¹<https://www.w3.org/History/1989/proposal.html>

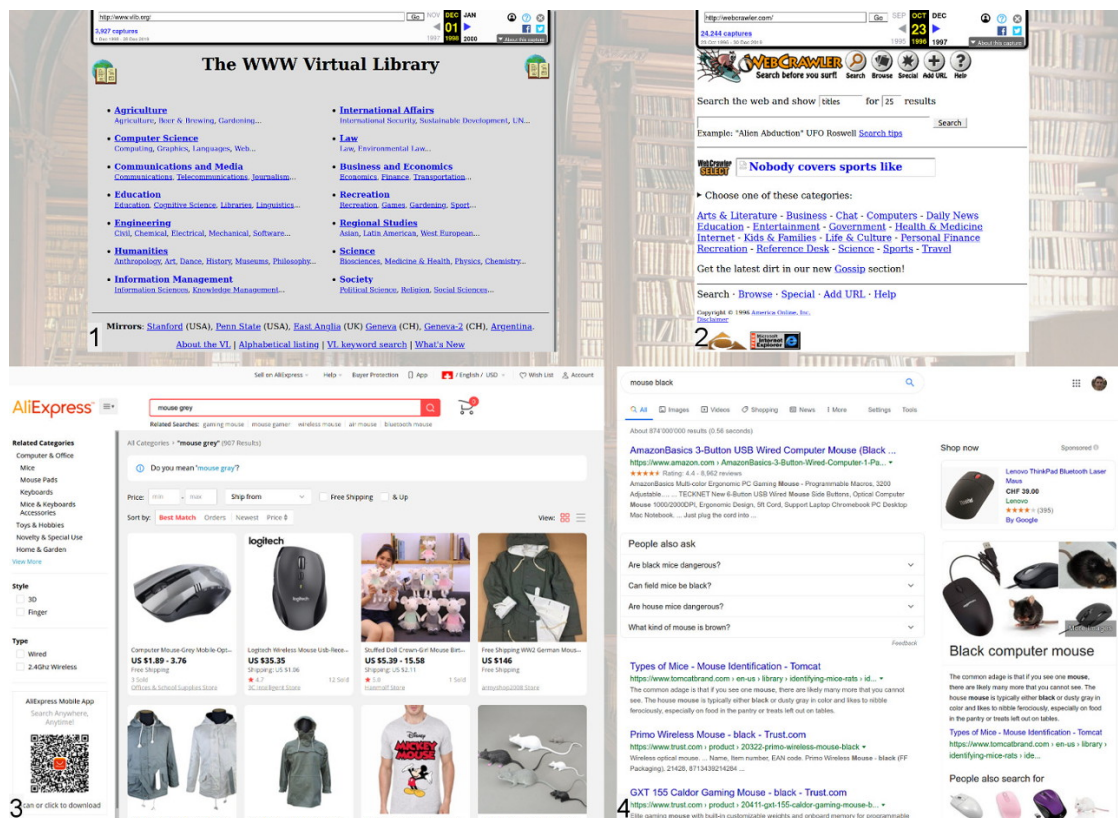


Figure 1.2 – Evolution of knowledge discovery and exploration systems: 1. One of the first website catalogs, VLib, founded in 1992; 2. The first automatic web search engine, WebCrawler, created in 1994; 3. A present e-commerce recommender system, AliExpress, responding to the query “mouse grey”, 4. A present web search engine, Google, responding to the query “mouse black”.

Archie² for FTP, Veronica [155] and Jughead for Gopher, or the W3Catalog³ for the World Wide Web) and b) recommender systems for domain-specific knowledge exploration and filtering (e.g., Tapestry [71] for email filtering, NewT [215] and GroupLens [200] for news filtering and recommendation, Ringo/FireFly [213] for music recommendation). The first search engines relied on preexisting and manually curated website catalogs resources, which a) could not promptly reflect new or updated information and b) contained only pieces of meta information about the target resources (e.g., titles, taxonomic information and, sometimes, manually-defined keywords for each web page). To overcome these limitations, search engines were latter extended with automatic crawling and indexing capabilities (e.g., such capabilities were partially present in Aliweb [110] and JumpStation⁴, and fully implemented in WebCrawler [192], shown in Figure 1.2#2) providing up-to-date and comprehensive full-text indexing of the large portions of the World Wide Web.

²<http://www1.chapman.edu/gopher-data/archives/Internet%20Information/whatis.archie>

³<http://scg.unibe.ch/archive/software/w3catalog/>

⁴<http://www.bbc.co.uk/news/technology-23945326>

However, the availability of overwhelming comprehensive information by itself is insufficient for knowledge exploration and especially for information search. The crucial aspect there is the *approach to transform the abundance of the retrieved information in a usable* (i.e., human perception-adapted) and efficient way. The emerged search engines in the middle of the 90s ordered results (i.e., a list of web page references) by the number of exactly matching keywords [15, 242]. Such an ordering has a low utility, failing short of differentiating reliable information from irrelevant results (e.g., when retrieving an irrelevant page matching the query topic by being extended with irrelevant information). Moreover, semantically similar statements expressed using slightly different terminology could not be matched, negatively affecting the relevance of the retrieved results. These issues were partially addressed in 1996 via the groundbreaking approach proposed by Google (called BackRub at that time) and led to the rise of their search engine, which became the most tool on the web [15]. Search results generated by Google were ranked by their *importance* respective to the user query using the PageRank algorithm, dismissing irrelevant or unimportant pages [181]. To further improve the relevance of the produced results, modern search engines provide personalized results [95] and operate with contextual awareness [241, 128]. Search engines have also since then adopted various recommendation approaches blurring the boundaries between knowledge search and recommendation services as shown in Figure 1.2#3-4.

Still, present knowledge exploration systems cannot always provide highly relevant and contextually meaningful results for a given user query and struggle with fundamental issues relating to the contextual processing of synonyms and homonyms [234], cold start personalization [209, 208, 12], reliance on evolving (and, often, manually curating) knowledge graphs (KGs) suffering from consistency issues, the use of machine learning (ML) models lacking explainability features and requiring supervision and manual tuning. Moreover, personalized and contextualized processing have high computational requirements, often limiting their applicability beyond domain-specific knowledge exploration and recommender systems. These issues boosted research in semantic text representations and led the emergence of the word and graph embedding techniques (e.g., word2vec [161] and GloVe [188], DeepWalk [190] and Node2Vec [76]). These techniques perform automatic feature extraction from an input dataset and identify items (e.g., words in a text, nodes in a graph, points in space, etc.) having similar semantic meaning or tight mutual relations, yielding a representation that can be used to tackle many of the aforementioned points (e.g., cold-start recommendations [208], enrichment and refinement of KGs [137, 275], recommendation assisted by KGs [275]).

While providing many benefits, state-of-the-art embedding techniques mostly work as black box ML models lacking *explainability* of the yielded results and relying on multiple parameters that still have to be *manually tuned for* each dataset exhibiting statistically distinct properties. The explainability of black box ML models is typically achieved with the help of Visual Analytics (VA) [111, 39, 2, 70, 273] and often relies on embedding methods themselves to visualize n -dimensional space in 2D (e.g., t-SNE [146, 109, 67]). VA can also support the interactive exploration of knowledge typically relying on hierarchical or multi-scale aggregation (e.g., taxonomies [29, 253, 158], ontologies [1, 197], treemaps [136, 94] or summaries [139, 189, 5]) to

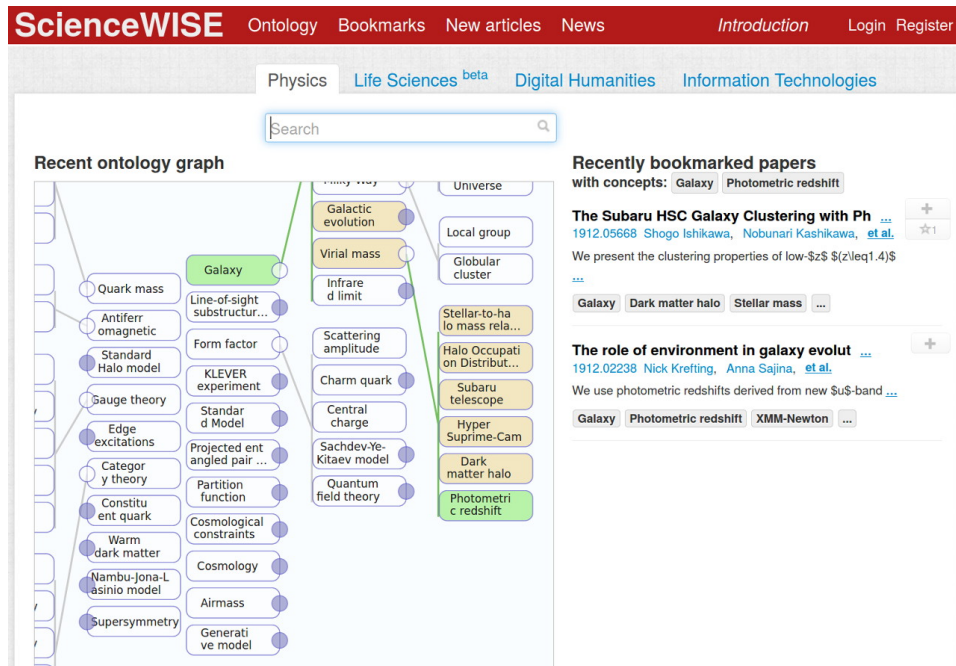


Figure 1.3 – The visual knowledge exploration interface of ScienceWISE, a collaborative web service for scientific papers exploration.

condense and structure the underlying information [53, 117, 103], as illustrated in Figure 1.3. However, knowledge evolves in time, imposing certain requirements on the visualization including the *stability and coherence* of the resulting layout [118, 86, 219] preserving the mental map of the user (which represents his/her understanding of the visualized information [51, 164, 7]) to benefit from his/her reliance on past interactions and avoid confusion or misperception.

The research conducted in the scope of this thesis aims at overcoming the aforementioned issues. In particular, we fully automatically build a human-readable representation of information that is both suitable for visual exploration and facilitates further data analysis tasks. For example, graph embeddings are crucial components of present search and recommender systems, but typically require manual tuning of several parameters while often lacking interpretability features. Based on our custom representation of information, we propose novel graph embedding techniques that generate interpretable embeddings without any manual tuning while being orders of magnitude faster compared to state-of-the-art solutions. Our research contributions resulted in a series of data mining tools and frameworks, which have been made freely available as open source software. The novel techniques and frameworks described in the following chapters do not rely on hyper-parameters and do not require any manual tuning, yet provide customizable and interactive results with the ambition to greatly simplify and speed up various data analysis tasks. In terms of their social impact, our results can be helpful in reducing social polarization by diminishing “filter bubbles” in search and recommender services, and in combating disinformation and rumors as outlined at the end of this thesis.

1.1 Research Questions

Research conducted in the scope of this thesis tackles the following questions:

1. How to *structure* large amounts of (evolving) information without any manual tuning to return a *human-readable representation* that is suitable for visual exploration?
2. What are the *measures* that are suitable to evaluate the accuracy of the resulting representation?
3. How to *benchmark* our structuration process and compare it to the existing solutions?
4. How can the resulting representation facilitate the *refinement of knowledge graphs* to infer missing information?
5. How to build *interpretable graph embeddings* without any manual tuning relying on our representation?

We introduce these questions in more detail in the following chapter (Background), describing formal and measurable research objectives for each of them.

1.2 Contributions

We made a series of contributions related to the research questions listed above. Our contributions are broadly related to the overall goal of representing information summaries in a human-readable way and suitable for visual exploration. Our main representation in that context corresponds to a *stable hierarchy of overlapping clusters* formed on various resolution levels for a given information domain. Our contributions are introduced below and are described in more detail in the rest of this thesis. We also mention the research papers we published in the context of each contribution.

1.2.1 Stable clustering of large networks without manual tuning

In Chapter 3, we present a novel clustering method, DAOC, designed for stable (i.e., deterministic and robust) clustering of large networks. DAOC is a Deterministic and Agglomerative Overlapping Clustering algorithm that leverages a new technique called Overlap Decomposition to identify fine-grained clusters in a deterministic way capturing multiple optima. In addition, it leverages a novel consensus approach, Mutual Maximal Gain, to ensure robustness and further improve the stability of the results while still being capable of identifying micro-scale clusters. DAOC does not require any manual tuning and produces a hierarchy of overlapping clusters as a result. Our empirical results on both synthetic and real-world networks show that DAOC yields stable clusters while being on average 25% more accurate than state-of-the-art deterministic algorithms.

DAOC is a versatile clustering technique, which besides the generation of human-readable taxonomies for large networks, also facilitates various data mining tasks. For example, we adopt DAOC for knowledge graph refinement in Chapter 6 while a novel graph embedding technique based on DAOC is introduced in Chapter 7. Moreover, the avoidance of manual tuning in DAOC enables to dynamically cluster arbitrary input networks in real time, and to support complex applications down the processing pipeline. Our approach has the ambition to greatly simplify and speed up data analysis tasks involving iterative processing (need for determinism) as well as data fluctuations (need for robustness) and to provide accurate and reproducible results.

Publication: Artem Lutov, Mourad Khayati, and Philippe Cudré-Mauroux. “DAOC: Stable Clustering of Large Networks”. IEEE International Conference on Big Data (BigData’19), 2019, Los Angeles, CA, USA [142].

1.2.2 Accuracy measures for overlapping and multi-resolution clustering

In Chapter 4, we identify accuracy measures that are suitable for the evaluation of overlapping and multi-resolution (in particular, nested) clusters in large networks. We analyze how these metrics satisfy a set of formal constraints and propose several optimizations and extensions. More specifically, we introduce a new indexing technique to reduce both the runtime and the memory complexity of Mean F1 score evaluation. Our technique can be applied on large datasets and is faster on a single CPU than state-of-the-art implementations running on high-performance servers. In addition, we propose several other extensions to improve the effectiveness of the evaluation and to satisfy formal constraints related to the accuracy metrics without affecting their efficiency.

Publication: Artem Lutov, Mourad Khayati, and Philippe Cudré-Mauroux. “Accuracy Evaluation of Overlapping and Multi-Resolution Clustering Algorithms on Large Datasets”. IEEE International Conference on Big Data and Smart Computing (BigComp’19), Kyoto, Japan, 2019 [141].

1.2.3 Benchmarking of stable, overlapping and multi-resolution clustering

Chapter 5 is dedicated to the comprehensive evaluation of clustering algorithms on large networks taking into account various peculiarities. To the best of our knowledge, there does not exist any uniform benchmarking framework yet that is publicly available and suitable for the parallel benchmarking of diverse clustering algorithms on a wide range of synthetic and real-world datasets. In Chapter 5, we hence introduce Clubmark, a new extensible framework that aims to fill this gap by providing a parallel isolation benchmarking platform for clustering algorithms evaluation on NUMA servers. Clubmark allows for fine-grained control over various execution variables (timeouts, memory consumption, CPU affinity and cache policy) and supports the evaluation of a wide range of clustering algorithms including multi-level,

hierarchical and overlapping clustering techniques on both weighted and unweighted input networks with built-in evaluation of several extrinsic and intrinsic measures. Our framework is open-source and provides a consistent and systematic way to execute, evaluate and profile clustering techniques considering a number of aspects that are often missing in state-of-the-art frameworks and benchmarking systems.

Publication: Artem Lutov, Mourad Khayati, and Philippe Cudré-Mauroux. “Clubmark: A Parallel Isolation Framework for Benchmarking and Profiling Clustering Algorithms on NUMA Architectures”. IEEE International Conference on Data Mining Workshops (ICDMW’18), Singapore, 2018 [140].

1.2.4 Entity type completion in knowledge graphs via clustering

In Chapter 6, we elaborate on entity type completion for Linked Data (i.e., knowledge graphs) in a fully unsupervised manner, introducing a novel statistical type inference method called StaTIX. Our inference technique leverages our new hierarchical clustering algorithm, DAOC, which is robust, highly effective, and scalable. In addition, we propose a novel approach to reduce the processing complexity of the similarity matrix specifying the relations between various instances in a knowledge base. This approach speeds up the inference process while also improving the correctness of the inferred types due to the noise attenuation in the input data. We further optimize the clustering process by introducing a dedicated hash function that speeds up the inference process by orders of magnitude without negatively affecting its accuracy. Finally, we describe a new technique to identify representative clusters from the multi-scale output of our clustering algorithm to further improve the accuracy of the inferred types. We empirically evaluate our approach on several real-world datasets and compare it to the state of the art. Our results show that StaTIX is more efficient than existing methods (both in terms of speed and memory consumption) as well as more effective. StaTIX reduces the F1-score error of the predicted types by about 40% on average compared to the state of the art and improves the execution time by orders of magnitude.

Publication: Artem Lutov, Soheil Roshankish, Mourad Khayati, and Philippe Cudré-Mauroux. “StaTIX - Statistical Type Inference on Linked Data”. IEEE International Conference on Big Data (BigData’18), Seattle, WA, USA, 2018 [143].

1.2.5 Fully automatic and interpretable graph embedding via clustering

Finally, we introduce in Chapter 7 a new graph embedding technique, DAOR, which overcomes several shortcomings of the respective state-of-the-art approaches. Current graph embedding approaches either sample a large number of node pairs from a graph to learn node embeddings via stochastic optimization or factorize a high-order node proximity/adjacency matrix via computationally intensive matrix factorization techniques. These approaches typically require significant resources for the learning process and rely on multiple parameters,

which limits their applicability in practice. Moreover, most of the existing graph embedding techniques operate effectively in one specific metric space only (e.g., the one produced with cosine similarity), do not preserve the higher-order structural features of the input graph and cannot automatically determine a meaningful number of dimensions for the embedding space. Typically, the produced embeddings are not easily interpretable, which complicates further analyses and limits their applicability. To address these issues, we propose DAOR, a highly efficient and parameter-free graph embedding technique producing metric space-robust, compact and interpretable embeddings without any manual tuning. Compared to a dozen state-of-the-art graph embedding algorithms, DAOR yields competitive results on both node classification (which benefits from high-order proximity) and link prediction (which relies on low-order proximity mostly). Unlike existing techniques, however, DAOR does not require any parameter tuning and improves the embeddings generation speed by several orders of magnitude. Our approach has hence the ambition to greatly simplify and speed up data analysis tasks involving graph representation learning.

Publication: Artem Lutov, Dingqi Yang, and Philippe Cudré-Mauroux. “Bridging the Gap between Community and Node Representations: Graph Embedding via Community Detection”. IEEE International Conference on Big Data (BigData’19), 2019, Los Angeles, CA, USA [144].

1.2.6 What this Thesis is Not About

The research described in the context of this thesis is performed with the aim to *construct a human-readable representation of information that is suitable for visual exploration and facilitates further data analysis tasks*, including recommendation and knowledge graph refinement. This ambitious and broad goal is decomposed into several concrete, formal and measurable tasks improving on state-of-the-art results, where further aspects of the whole data mining pipeline, such as *information retrieval, or data preparation and visualization* are left aside. The input information we focus on is expected to be provided as a network, i.e., a weighted graph specified by pairwise relations. Though we describe possible ways to convert a knowledge graph into a network when solving specific tasks (e.g., in Chapter 6), the overall methodology to convert the required information into a network is left outside of our work. Similarly, even though some of the tools we developed for the construction of information summaries support specific output formats (e.g., for visual exploration with the ScienceWISE [1] interface shown in Figure 1.3), we typically do not focus on such technical details that are left outside of the scope of this thesis.

1.2.7 Thesis Outline

We conclude this first chapter by giving an outline of the overall thesis. This work is organized in eight chapters. Background information is presented in Chapter 2, which gives information on the state-of-the-art research from various fields that became the basis for our work. This background also allows us to break down the formulated research questions into more specific

Chapter 1. Introduction

and measurable objectives. In particular, it gives additional information on human perception and information representation, justifying some of the representations and processing methods that we selected. Chapters 3-7 are devoted to the comprehensive presentation of our scientific contributions, as listed in Section 1.2. The thesis concludes with Chapter 8, which summarizes our main findings and gives an outlook on future research directions.

2 Background

In this chapter we present the main concepts required to understand the rest of this thesis. The information is given here in a concise manner omitting unnecessary technical details and concentrating on the most important concepts only. The following information is accompanied with numerous references to further sources should the reader be interested in learning more about a particular subject.

In what follows, we start with a brief on the aspects of human perception from various scientific disciplines, then discuss a unified representation of information and a specific kind of its granulation, i.e., clustering. The latter is described in more detail as it is an important building block for the rest of this thesis. This chapter concludes with a concise description of state-of-the-art approaches in information visualization and their shortcomings, which we have the ambition to overcome as part of our future work.

2.1 Human Perception of Information

Substantial multidisciplinary research has been conducted to shed light on the aspects of human cognition related to information perception. The major aspects required for effective information perception by humans, which affect the information representation and, specifically, its visualization can be summarized as follows. First, humans rely on their experience, habits and employ unconscious movements when operating in a familiar environment, which dramatically boosts their operational performance while reducing their cognitive load [78]. The *stability* and *coherence* of an individual's surroundings [118, 86, 219] preserves his/her *mental map* [51, 164, 7]. Hence, such stability is also expected from an information visualization or layout. Second, human perception is individualized, meaning that the same piece of information can be perceived, described and processed from different perspectives. Moreover, those perspectives are also dependent on the ongoing activities of an individual and of his/her environment (i.e., *context*) [239, 103]. A well-known image on human perception is given Figure 2.1, where several blind people touching an elephant guess what they touch, and come up with very different interpretations based on their local context. Third, in spite of being highly

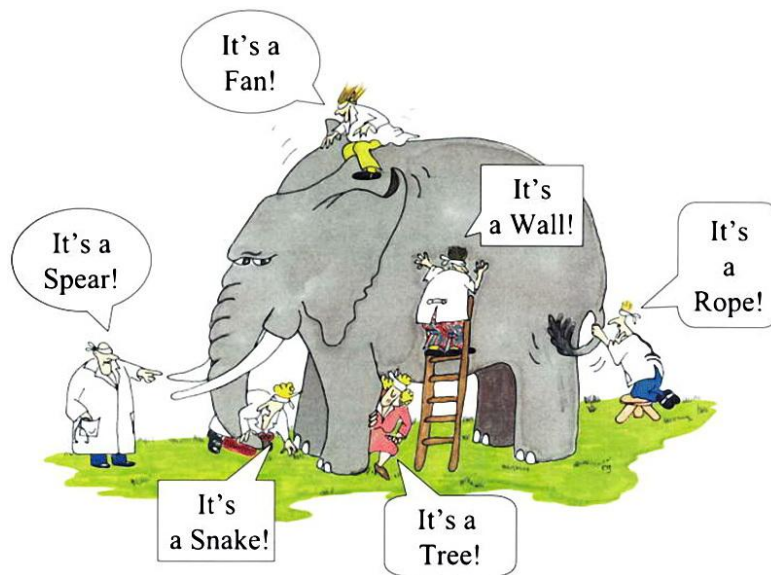


Figure 2.1 – An illustration of a multi-perspective and context-dependent human perception.

complex, our perception also implies strict limits on our capacity of processing information. In particular, humans seem to perceive well only up to 7 ± 2 items simultaneously, which was termed the “Magic number 7” or “Miller’s Law” [162]. Therefore, to explore large amounts of information, an individual should obtain it in a structured way with only a limited number of items in each (sub)structure. In essence, such an organization corresponds to a hierarchical representation, which can also be called taxonomy or ontology [29, 253, 1] depending on the context.

To summarize, human perception dictates that information be represented respecting the following criteria for efficient interaction:

- Stability;
- Multiple perspectives;
- Fine-grained hierarchical representation.

2.2 Knowledge Representation

Knowledge in the field of Information Systems can be characterised as explicit and processible information in a given context; *information* can be defined as data organized in a meaningful way, whereas data represents raw facts [223, 280]. Organizing pieces of data as a set of items is performed by imposing relations between them. Such relations can be defined either implicitly by specifying some properties (also known as attributes or features) on the items, or explicitly by specifying pairwise relations (i.e., links) [278, 246]. Formally, an abstract set of items with

relations specified between them is called a *graph*. The graph construction process from a real-world complex system (e.g., a physical, social, biological or information system) is called *network modelling* and is studied by network science [127]. A graph, where the item (i.e., vertex or node) relations are (implicitly) specified by attributes (e.g., vertices specified by coordinates in a multi-dimensional space) is called an attributed graph. A graph specified by pairwise relations is often called a *network*. It consists of nodes (i.e., graph vertices) and links (i.e., graph edges), where the links can be a) directed (i.e., arcs) or undirected (i.e., edges) and b) either weighted or unweighted. An attributed graph can be converted to a network [278, 246, 152] and vice versa [87] but the conversion process is unambiguous, at least unless one agrees on some customized and specific conversion function. A *Knowledge Graph (KG)* is a (knowledge) base containing semi-structured information represented as a hyper graph [235].

Thus, a formal and unified representation of information can be achieved via *network* modelling, yielding a network (i.e., a graph) of information. However, humans are not capable of handling large amounts of information simultaneously, as it was outlined in the previous section. So, for the effective interactive exploration of large amounts of information, some non-straightforward representation of the corresponding graph is required respecting the constraints outlined in Section 2.1. A general approach yielding such a representation is discussed in the following section.

2.3 Clustering for Information Granulation

The finite capacity of the human mind to handle information (see “Miller’s Law” [162] in Section 2.1) calls for information *granulation*. Granulation involves decomposing information into subparts, called granules [270]. In general, granulation is hierarchical in nature and may be viewed as a form of (lossy) data compression [270]. Information Granulation is studied in the field of Granular Computing, which can be seen as a superset of fuzzy information, rough set theory, interval computations, and clustering [270, 266, 267].

2.3.1 Network Clustering

A clustering process is applied to a network (i.e., a graph) to produce groups of similar nodes that are called clusters (also known as communities, granules, modules, partitions or covers). This process is also known as community detection or community structure discovery [127, 84]. Unlike a classification process, clustering is an unsupervised technique, where the resulting groups (i.e., clusters corresponding to the classes, labels or categories) are not known in advance. The similarity of nodes in a network is often related to the strength or density of the connections between the nodes, though in general various similarity functions can be used (see Section sec:clsobj). Thus, clusters represent groups of tightly-coupled nodes with loosely inter-group connections [171], where the group structure is defined by a clustering optimization function. The resulting clusters can be *overlapping*, which happens in case they share some common nodes, the overlap. If all nodes in the overlap are equally shared

between the clusters, then the overlap is called *crisp* otherwise it is called *fuzzy* [75]. Also, the clusters can be nested, forming a hierarchical structure inherent to many complex real-world systems [218, 121]. Each cluster represents a coarse-grained view on (i.e., an approximation of) its member nodes or subclusters and is also called a granule [267]. The input network (graph) can be weighted and directed, where a node (vertex) weight is represented as a weighted link (edge) to the node itself (a self-loop).

2.3.2 Clustering Techniques

In this section, we first describe several popular classifications of clustering algorithms, and then narrow them down with respect to a) essential characteristics with respect to the research questions stated in Section 1.1 and b) aspects of human perception outlined in Section 2.1. In this way, we identify the main properties of a clustering algorithm for constructing a human-readable representation of information that is suitable for visual exploration and can facilitate further data analysis tasks.

Clustering algorithms can be classified through various criteria [163, 68, 84]. Depending on the kind of input data they accept, clustering algorithms can operate on a) graphs specified by pairwise relations (i.e., networks), b) attributed graphs (e.g., vertices specified by coordinates in a multi-dimensional space), or c) or hybrid graphs containing both links and attributes for both the nodes and links. In terms of cluster formation, the algorithms can produce a) hard (a.k.a crisp) or b) overlapping clusters. If all nodes in the overlap are equally shared between their clusters, then the overlap is called crisp otherwise it is called fuzzy [75]. Also, the resulting clusters can be either flat or nested, forming complex hierarchical structures. Depending on data availability, the clustering algorithms can operate on the whole dataset or incrementally in a stream mode. There are many distinct classifications of clustering algorithms [68, 84]; a general and concise one is given by [163] and classifies clustering techniques as follows: single cluster clustering techniques (fitting input data into a specified number of clusters), conceptual clustering techniques (relying on learning decision trees), separating surface and neural network clustering techniques, and probabilistic and statistical clustering techniques. Each of those techniques can be further classified by the optimization function the use [28, 68, 257]. In-depth classifications of clustering algorithms along with their overview can be found in several dedicated books [163, 68, 84] and surveys [93, 257, 60, 79, 256, 49].

Formally, the clustering task can be represented by a stochastic block model and exactly solved using maximum likelihood method or column generation methods for integer programming [3]. However, the exact solution is NP-hard even in its simplest form (i.e. for disjoint partitions) [69], such that any practical clustering algorithm employs some heuristics or approximations [84]. This tradeoff between applicability and optimality resulted in the emergence of a vast number of clustering algorithms designed for the specific tasks [57]. In this thesis, we are only considering solutions applicable to large graphs, i.e. having a *near-linear computational complexity* at least on sparse graphs. In addition, since knowledge exploration

2.3. Clustering for Information Granulation

in general does not impose any assumption on the processed information and skills of the user, the clustering algorithm should be a) applicable on any input graph and b) applicable by any user. These two additional constraints lead to avoiding any manual tuning. Therefore, in our context a clustering algorithm should be *unsupervised* and *parameter-free*.

Mapping the aspects of human perception specified in Section 2.1 to properties of the clustering algorithm lead to the following requirements:

- Stability requires *determinism* and *robustness*, as explained in more detail in Chapter 3.
- Multi-perspective views on input information leads to *overlapping* and *multi-resolution* clustering techniques, where each node of the input graph can be shared between several clusters at the same resolution (i.e., stale or granularity) and belong to several nested clusters corresponding to various resolutions. Overlaps and resolutions are discussed in more detail in Chapters 4 and 7.
- A browsable tree-like representation is naturally constructed by an iterative *hierarchical* clustering, which is a special case of multi-resolution clustering techniques. Hierarchical clustering is performed using either divisive (i.e., iteratively dividing groups of items into subgroups) or agglomerative (i.e., iteratively merging groups of items into supergroups) approaches. Among them, agglomerative clustering approaches fit aforementioned computational requirements better, since they reduce the number of items to be processed at every iteration.

Our objectives, along with the constraints imposed on the clustering algorithms we consider, can be summarized as follows: we target a human-readable representation of information that is suitable for visual exploration and facilitates data analysis through a taxonomy that can be refined for evolving networks. Table 2.1. Summarizes our objectives.

Table 2.1 – Our objectives: mapping criteria of a human perception-adapted taxonomy onto clustering requirements. The clustering properties highlighted with italics are present in the Louvain clustering algorithm.

Taxonomy construction for evolving networks	by incremental clustering
<ul style="list-style-type: none"> • <u>Stable</u> • Multi-viewpoint • Without any manual tuning • Browsable • Large • Narrow (7 ± 2 rule) levels 	<ul style="list-style-type: none"> • <u>Robust and deterministic</u> • Overlapping and Multi-resolution • <i>Unsupervised and parameter-free</i> • <i>Hierarchical</i> • <i>Near-linear runtime</i> • Fine-grained clusters

Louvain

To select or design an appropriate clustering algorithm fitting our objectives, several intrinsic characteristics should be considered in addition to the aforementioned properties. In particular, the required processing of any input network implies that the clustering algorithm

should be applicable to items that a) may form both convex and concave shapes, and b) may form multiple complex distributions with various statistical properties (e.g., density). The first constraint filters out many popular clustering algorithms including K-Means [82, 135, 21], which are not capable to meaningfully cluster items forming non-convex shapes, as shown in Figure 2.2. This issue is addressed by density-based clustering algorithms, which can correctly segment any shapes consisting of uniformly distributed items. In addition, the second constraint further filters out many algorithms including the popular DBSCAN [55], where the density is parameterized, such that clusters having distinct item densities (see for example Figure 2.3) cannot be fully detected. One approach that fully addresses the first constraint and detects nested clusters in the second constraint, while exhibiting some properties from Table 2.1 and being computationally efficient is called *Louvain* [17]. This clustering algorithm is based on Modularity [174] (see Section 2.3.3.1) and is described in more detail in Chapter 3.

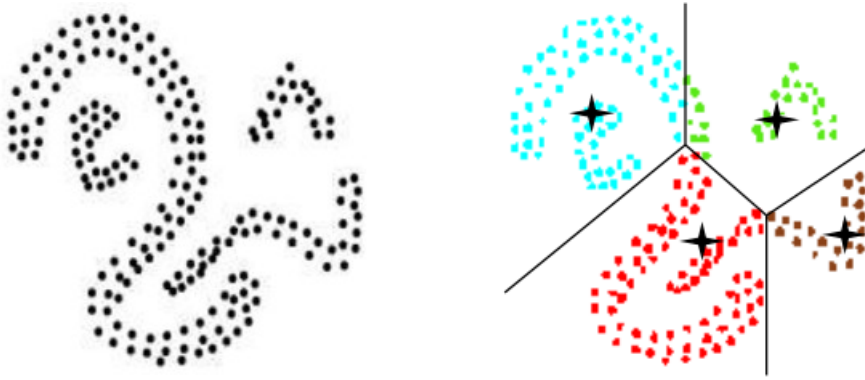
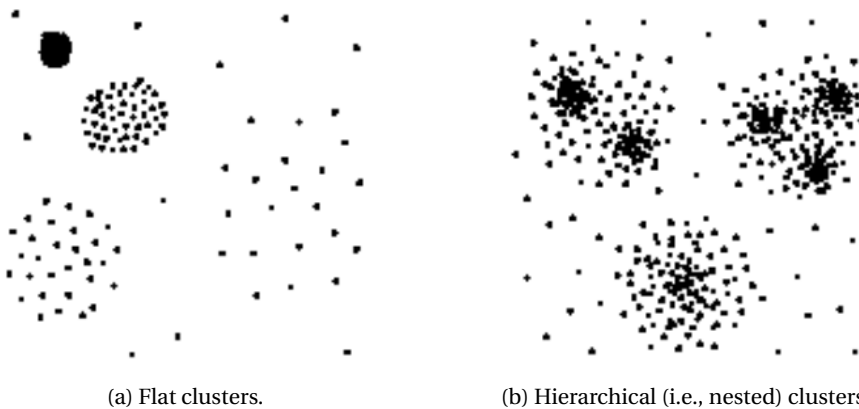


Figure 2.2 – A typical K-Means clustering of items forming a non-convex shape yielding meaningless clusters [167].



(a) Flat clusters.

(b) Hierarchical (i.e., nested) clusters.

Figure 2.3 – Items forming multiple complex distributions with various densities [56].

2.3.3 Clustering Objective

Clusters are groups of similar items (i.e., nodes, vertices, points), where the similarity value for each pair of items is defined by some (dis)similarity measure (also known as a distance criteria) [256]. A clustering objective is formally specified via an optimization function, which optimizes the aggregated value of a given similarity measure for all input items. Typically, some intrinsic quality measure (also known as internal quality indicator, internal validity index or goodness of the clustering) is used as the optimization function. Popular intrinsic quality measures include modularity [174], conductance [98], silhouette index [204], betweenness centrality [65], and the clustering coefficient [251], among others. A comprehensive overview of various similarity and quality measures is given in [28, 68]. In what follows, we describe the intrinsic quality measure used in the main clustering algorithm we describe in this thesis, DAOC. In addition, we discuss the phenomenon of degeneracy, which is inherent to any optimization function, and which is mitigated in DAOC as explained in Chapter 3.

Modularity

Modularity (Q) [174] is a standard measure of clustering quality that is equal to the difference between the density of the links in the clusters and their expected density:

$$Q = \frac{1}{2w} \sum_{i,j} \left(w_{i,j} - \frac{w_i w_j}{2w} \right) \delta(C_i, C_j), \quad (2.1)$$

where $w_{i,j}$ is the accumulated weight of the arcs between nodes $\#i$ and $\#j$; w_i is the accumulated weight of all arcs of $\#i$; w is the total weight of the graph; C_i is the cluster to which $\#i$ is assigned; and the Kronecker delta $\delta(C_i, C_j)$ equals to 1 when $\#i$ and $\#j$ belong to the same cluster (i.e., $C_i = C_j$), and 0 otherwise.

Modularity is applicable to non-overlapping cases only. However, there exist modularity extensions that handle overlaps [75, 36]. The main intuition behind such modularity extensions is to quantify the degree of a node membership among multiple clusters either by replacing a Kronecker δ (see (2.1)) with a similarity measure $s_{i,j}$ [168, 214] or by integrating a belonging coefficient [175, 129, 138] directly into the definition of modularity. Although both old and new measures are named modularity, they generally have different values even when applied to the same clusters [74], resulting in incompatible outcomes. Some modularity extensions are equivalent to the original modularity when applied to non-overlapping clusters [168, 214, 138]. However, the implementation of these extensions introduce an excessive number of additional parameters [168, 214] and/or boost the computational time by orders of magnitude [138], which significantly complicates their application to large networks.

Modularity gain (ΔQ) [17] captures the difference in modularity when merging two nodes $\#i$ and $\#j$ into the same cluster, providing a computationally efficient way to optimize Modularity:

$$\Delta Q_{i,j} = \frac{1}{2w} \left(w_{i,j} - \frac{w_i w_j}{w} \right). \quad (2.2)$$

We use modularity gain (ΔQ) as an underlying optimization function for our meta-optimization function MMG (introduced in Chapter 3.3.1).

Modularity is commonly used as a global optimization criterion but suffers from the *resolution limit* problem [61, 72], which corresponds to its inability to detect clusters smaller than a certain size. To address this problem, *generalized modularity* was proposed with a resolution parameter γ [8, 120]:

$$Q = \frac{1}{2w} \sum_{i,j} \left(w_{i,j} - \gamma \frac{w_i w_j}{2w} \right) \delta(C_i, C_j). \quad (2.3)$$

The optimal value of the resolution parameter is $\hat{\gamma}$ [145]:

$$\hat{\gamma} = \frac{\check{p} - \hat{p}}{\log \check{p} - \log \hat{p}}, \quad (2.4)$$

$$\check{p} = \frac{2\check{w}}{\sum_c \dot{w}_c^2 / (2w)}, \quad \hat{p} = \frac{2w - 2\check{w}}{2w - \sum_c \dot{w}_c^2 / (2w)},$$

where \check{w} is the total internal weight of all clusters (i.e., accumulated weight of all intra-cluster links), and \dot{w}_c is the internal weight of the cluster c . The generalized modularity is equivalent to the standard modularity when $\gamma = 1$; it tends to find larger (macro-scale) clusters if $\gamma \in [0, 1)$ and smaller clusters otherwise. We use the *generalized modularity gain* (ΔQ) as an underlying optimization function for the meta-optimization strategy MMG of the DAOC [142] clustering algorithm (see Chapter 3) on top of which our framework, DAOR [142], is built as presented in Chapter 7.

Degeneracy of the Optimization Function

Degeneracy is a phenomenon linked to the clustering optimization function, appearing when multiple distinct clusterings (i.e., the results of the clustering process) share the same globally maximal value of the optimization function while being structurally different [72]. This phenomenon is inherent to any optimization function and implies that a network node might yield the maximal value of the optimization function while being a member of multiple clusters, which is the case when an overlap occurs. This prevents the derivation of accurate results by deterministic clustering algorithms that do not consider overlap. To cope with degeneracy, typically multiple stochastic clusterings are produced and combined, which is called an ensemble or consensus clustering and provides robust but coarse-grained results [72, 14]. Degeneracy of the optimization function, together with the aforementioned computational drawback of modularity extensions, motivated us to introduce a new overlap decomposition technique, OD (see Chapter 3).

2.3.4 Clustering Validation

Clusters are validated using quality measures (also known as validity indices). There are two kinds of such measures: intrinsic (a.k.a. internal) and extrinsic (a.k.a. external). Intrinsic quality measures evaluate how the elements of each cluster are similar to each other and how they differ from elements in other clusters given some similarity measure. Extrinsic quality measures (also known as accuracy measures) evaluate instead how the generated clusters are similar to the ground-truth (i.e., gold standard) clusters. The latter measures allow to identify clustering algorithms producing expected results rather than the ones just having some statistical properties suitable for clusters in theory. That is why clustering effectiveness evaluations typically involve accuracy measures. A comprehensive overview of various extrinsic quality measures is performed in [28, 199, 30, 277]. It is worth noting that clustering validation also makes use of clustering interpretations (e.g., through functional descriptions of the obtained cluster structure). Although validation and interpretation are not coincident, finding a good interpretation is a part of validation; conversely, if the clusters are invalid, the interpretation appears unnecessary [163].

Given the existence of a large variety of accuracy measures, a question that arises is which of the measures should be used, when and why. Four formal constraints for the accuracy measures were introduced in [4] and shed light on which aspects of the quality of a clustering are captured by different measures. Two of these constraints (Homogeneity, see Figure 2.4a and Completeness, see Figure 2.4b) were originally proposed in [203] while the other two (Rag Bag, Figure 2.4c and Size vs Quantity, Figure 2.4d) were newly introduced in [4]. Besides being intuitive, these four constraints were developed to be formally provable and to clarify the limitations of each measure. We list these constraints in the way they were presented in [4] and use them in Chapter 4 to discuss various accuracy measures. Each constraint is written as

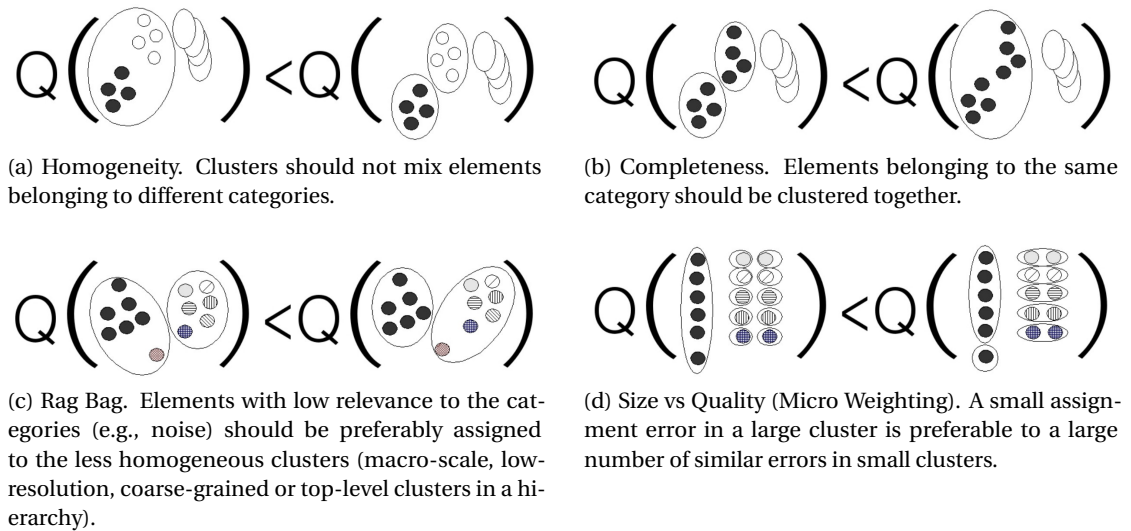


Figure 2.4 – Formal constraints for the accuracy measures.

an inequality applied to a pair of clusterings (i.e., sets of clusters), where a quality measure Q (some accuracy measure in our case) from the clustering on the right-hand side is assumed to be better than the one from the left-hand side. Ground-truth clusters are called categories for short in the following.

2.4 Information Visualization and Analytics

In this section, we first give an introduction to information visualization and its objectives. Then, we describe several modern and popular visualization techniques for representing multidimensional data. An overview of further visualization techniques for clustered data can be found in [68, 84]. Comprehensive studies on more general techniques for information and knowledge visualization are given in [105, 217, 42, 23].

2.4.1 Visualization

Visualization can be defined as a graphical representation of information whose aim is to make it easy to interpret the information and gain knowledge or insight [221]. Visualization systems depict relationships among information items in a human comprehensible form [261] and serve to guide the viewer's attention, communicate data to the viewer, let him/her process the data and assist in interactive exploration [244].

Information visualization has developed methods for the visualization of abstract data where no explicit spatial references are given (e.g., business data or social networks) [222]. Such information is often characterized by both large volumes, high dimensionality and complex data types (e.g., graphics, video and sound) that cannot be naturally mapped onto a low-dimensional space for straightforward visualization using conventional techniques (e.g., line, bar or pie charts) [103]. Therefore, novel visualization techniques have been developed including graph-based approaches for networks [166], geo-visualization for spatial and temporal data, techniques to reduce display clutter [52] and to visualize clustered data (e.g., *parallel coordinates*, *treemaps*, *glyph* and *pixel-based*) [217, 68, 84, 111]. The latter techniques make use of automatic data analysis methods such as clustering and dimensionality reduction as a preprocessing step prior to visualization. Cluster visualization is also a major interpretation tool for cluster analysis and is helpful in cluster validation [163].

2.4.2 Visual Analytics

Visual analytics (VA) combines automated analysis techniques with interactive visualizations to support effective understanding, reasoning and decision making on data [103]. VA evolved from *confirmatory* data analysis relying on results presentation to *exploratory* data analysis [240] employing intrinsic human abilities to find patterns in data while interacting with the visual representation [11, 103]. VA involves visual data exploration and is a part of visual data

mining [104]. Visual data mining is a field of computer science that aims to discover implicit knowledge from large data sets using knowledge visualization techniques [42].

The conventional workflow of VA is presented by Shneiderman's Visual Information Seeking Mantra: "overview first, zoom and filter, then details on demand" [216]. In terms of purpose, VA involves tools and techniques facilitating [103]:

- Information synthesis and insight derivation from massive, dynamic, ambiguous, and often conflicting data;
- Detection of the expected and discovery of the unexpected patterns;
- Provision of timely, defensible, and understandable assessments;
- Communication of the performed assessments effectively to take respective actions.

2.5 Visualization Techniques for Multidimensional Data

Visualization techniques for multidimensional data can be subdivided into *lossless*, *lossy* and *hybrid*.

Lossless techniques (e.g., parallel coordinates [91]) provide a straight-forward visualization of multidimensional data preserving the original properties and relations. These techniques typically represent each n D item (i.e., a node, vertex, or point) with n points in a 2D space, visualized as a polygonal curve (e.g., line segments). Thereby, this visualization limits both a) the amount of visualized data and b) the number of dimensions that can be effectively perceived, explored and analysed visually by a human (see Section 2.5.1) [68, 111].

Lossy techniques (e.g., Sammon's Mapping [207], t-SNE [146], or LargeVis [227]) transform the original data by projecting each n D item onto their respective 2D or 3D space, which significantly reduces the cluttering compared to the lossless techniques and provides effective coarse-grained view on the data. However, since properties and relations between the data items are distorted, lossy visualization may disregard some important characteristics of the visualized data and hence lead to misleading interpretations [244, 112], which we discuss in Section 2.5.2.

Hybrid techniques are either a) a combination of lossless and lossy visualizations applied to the same data items as shown in Figure 2.5, or b) a visualization of the iterative multi-level granulation [270, 267], which yields a lossy coarse-grained representation on the top levels and a lossless one on the bottom. An approximated representation of such granulation is a hierarchy of clusters, which typically can be efficiently built using agglomerative clustering methods (see Section 2.3.2) and visualized with folding trees (e.g., dendrograms and tree maps) [68, 84] as shown in Figure 1.3. Non-leaf nodes of the hierarchy are coarse-grained and lossy summaries of the underlying leaf nodes. The leaf nodes represent original data items, whose relations are losslessly represented (and visualized) with a graph as described in Section 2.2.

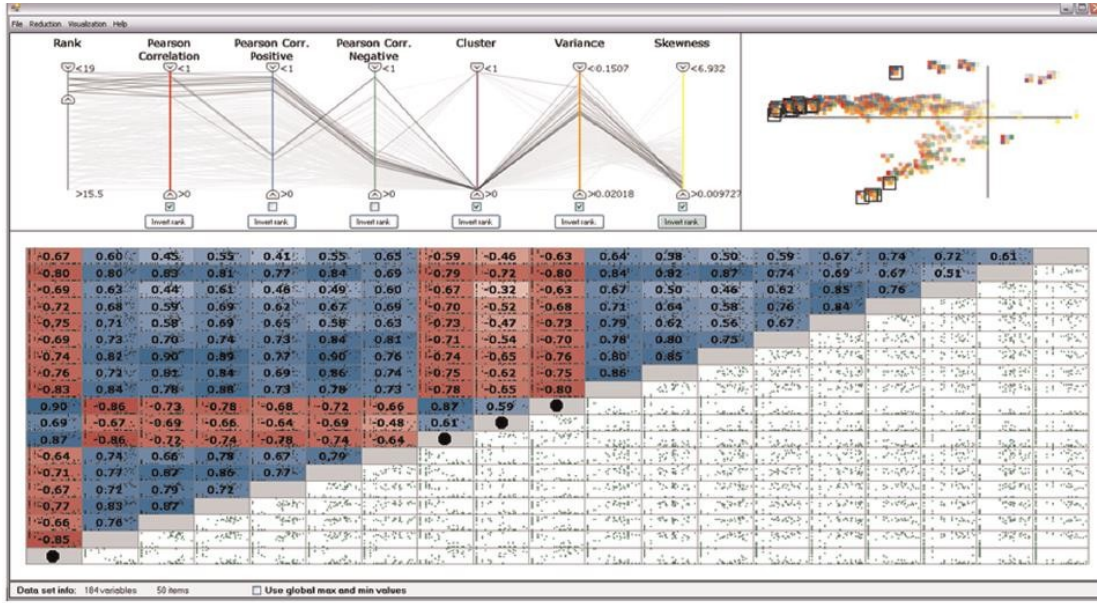


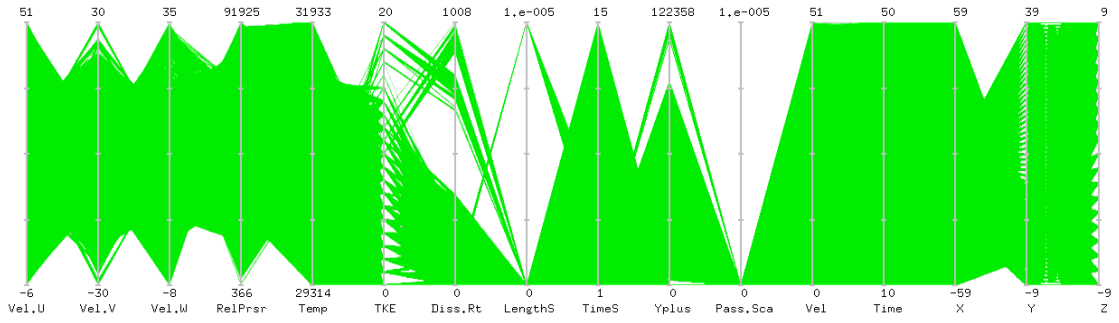
Figure 2.5 – Hybrid visualization of n D data using both parallel coordinates (top left), PCA-based 2D embedding of square glyphs (top right) and scatter plot matrix visualized as a heat map [112].

2.5.1 Lossless Techniques for Multi-dimensional Data Visualization

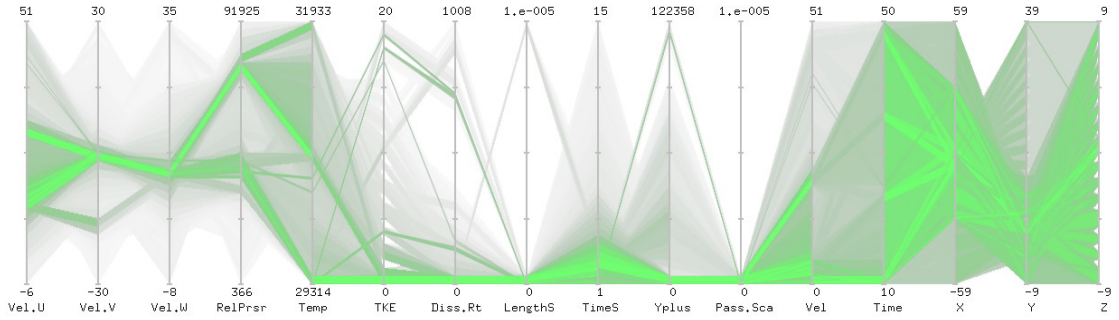
A straightforward and lossless visualization of multi-dimensional data in 2D or 3D space using *Parallel Coordinates* was proposed in [91]. This technique maps each n D item (i.e., node, vertex, or point) into n points, each one located in their respective parallel line (i.e., axis) on a 2D space, where an item is visualized as a polygonal line as shown in Figure 2.6. This representation preserves all relations between each data item but becomes ineffective when for the number of dimensions is larger than a dozen (see Section 2.1) or the number of visualizing items is large [68]. To address these shortcomings, several complementary techniques have emerged. To reduce the clutter when visualizing large datasets, sampling and semi-transparent lines are leveraged [177, 52] as shown in Figure 2.6b. To further improve human-readability of the visualization, a generalization of parallel coordinates, *General Line Coordinates*, was introduced in [111, 114]; its application with other visualization techniques is discussed in [113].

General Line Coordinates (GLC) is a class of reversible lossless 2D and 3D visualization methods preserving the original n D information of the visualizing dataset [111]. GLC comprises more than a dozen related methods including Parallel and Radial Coordinates, Collocated Paired Coordinates (CPC), Shifted Paired Coordinates (SPC), Paired Crown Coordinates (PWC) and others [111, 112, 113]. GLC has been designed to improve human-readability for multi-dimensional data visualization compared to Parallel Coordinates as shown in Figure 2.7 for several popular GLC instances. However, GLC is not a silver bullet, since they still retain the fundamental constraints of Parallel Coordinates: complex visual data analysis for a large

2.5. Visualization Techniques for Multidimensional Data



(a) Original visualization in PC.



(b) Visualization in PC applying clutter reduction techniques (sampling and semi-transparent line drawing).

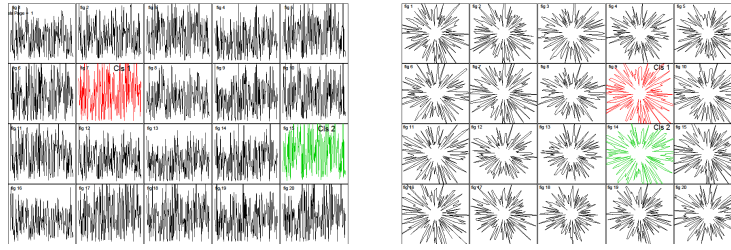
Figure 2.6 – Visualization of several millions of data items in parallel coordinates (PC) [177].

number of data items and dimensions, which is caused by cluttering and occlusions. Some GLCs yield the occlusion of several coordinates of the same data point. The latter is often attained intentionally to reduce visual clutter as it shown in Figure 2.7d, but on the other hand, it also may complicate visual analysis in some cases (e.g., when an inspection of colliding shapes of overlapping clusters is required).

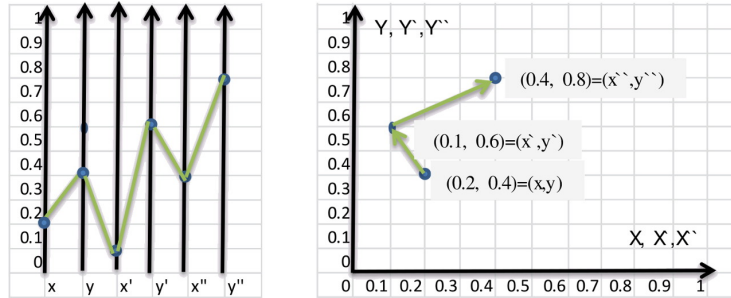
2.5.2 Lossy Techniques for Multidimensional Data Visualization

Multidimensional data may involve thousands of dimensions (e.g., knowledge graphs such as Wikidata, DBpedia, Freebase, Probase and others), the lossless visualization of which is typically ineffective for visual analysis performed by humans [68]. To cope with this issue, *lossy* visualization techniques have emerged, which provide a coarse-grained human-readable representation of nD data in 2D or 3D space via dimensionality reduction or clustering (e.g., *class-preserving projections*, *self-organizing maps*, *t-SNE*, *glyph* and *pixel-based visualizations*) [68, 217, 84, 105, 217]. However, for correctly interpreting the data when interacting with a lossy visualization of nD data in 2D space, it is critical to always keep in mind that the interaction is performed with some non-linear transformation rather than on the original data. This transformation may not reflect some of the original properties and relations that are inherent to the original data and that might be crucial for the purpose of the conducted analysis, causing misleading interpretations [244, 112]. Therefore, multiple pitfalls should

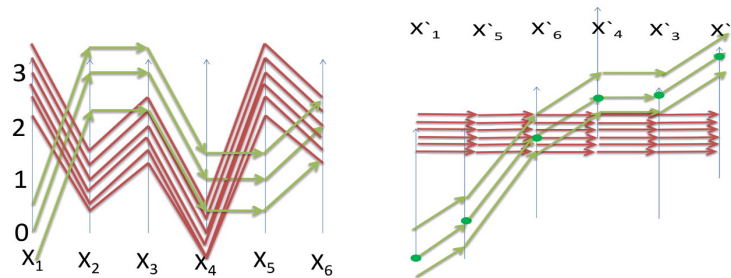
Chapter 2. Background



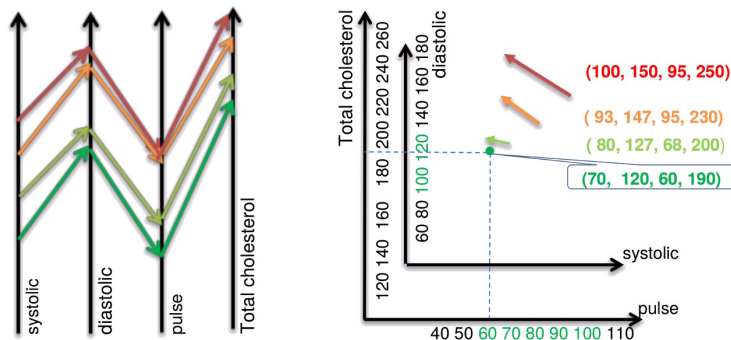
(a) Mapping of 20 160D points of 2 classes from Parallel Coordinates shown on the left to Radial Coordinates shown on the right. Data patterns are more easily perceived by humans in Radial Coordinates.



(b) Mapping of a 6D point from Parallel Coordinates shown on the left to Collocated Paired Coordinates shown on the right. The number of visualized lines including axes (i.e., cluttering) is reduced more than twice, but occlusions may happen when visualizing several points (e.g., in case $\{X_1, Y_1\} = \{X_2', Y_2'\}$) complicating the interpretation.



(c) Adjustable GLC for occlusion decreasing. A Parallel Coordinates plot with two 6D points shown on the left is adjusted by the axes shifting and reordering to the representation with decreased occlusions shown on the right.



(d) Mapping of 4 4D points from Parallel Coordinates shown on the left to the Parameterized Shifted Paired Coordinates (PSPC) shown on the right. The visual interpretation of distances to the desired state (e.g., person health) shown in green becomes more intuitive until occlusions occur.

Figure 2.7 – Transformation of visualizations from Parallel Coordinates to several popular kinds of General Line Coordinates (GLC) [112].

be taken into account when interacting with high-dimensional data projected into 2D or 3D space. In particular, often [244]:

- two random vectors become perpendicular,
- a standard Gaussian distribution becomes just a uniform distribution on a sphere,
- a random matrix becomes a scalar multiple of an orthogonal matrix,
- random walks all have the same shape.

The theoretical background grounding the limitations of dimensionality reduction-based methods for accurate representation of multi-dimensional data is given by the Johnson-Lindenstrauss lemma [63]. This lemma implies that only a small number of arbitrary n D points can be mapped onto k D points of a smaller dimension k that preserve n D distances with relatively small deviations [111]. In particular, keeping the errors within about 30% requires [112]:

- over 1900 dimensions for just 10 arbitrary high-dimensional points and
- over 4500 dimensions for 300 arbitrary points.

Lossy visualization methods projecting multi-dimensional data onto 2D space do not meet these requirements. To address these problems, a combination of lossy and lossless techniques can be applied as shown in Figure 2.5.

The two most popular state-of-the-art lossy visualization techniques are t-SNE (t-distributed Stochastic Neighbor Embedding) [146, 67] and UMAP (Uniform Manifold Approximation and Projection) [157]. Both of them are non-linear algorithms. They adapt to the underlying data, performing different transformations on different regions to reveal data structures at various scales, and feature a supervised mode in addition to the unsupervised one. Those differences in region transformation can be a major source of confusion when interpreting the generated visualization [250]. In particular, both t-SNE and UMAP are prone to the following problems when visualizing data and interpreting it [157, 244, 113] ¹:

- distortion of the global structure trying to preserve the local one,
- visible outliers may not actually be present in the original data (if n D data similarity is not preserved in 2D, which typically happens as explained below),
- visible densities of clusters might differ a lot compared to the ones in the original data,

¹<https://towardsdatascience.com/tsne-vs-umap-global-structure-4d8045acba17?sk=765c722b79cd4c12a584d20e321b8c83>

- distances between the visualizing clusters may not have meaning and differ a lot from the ones in the original data,
- some visual shapes might be spurious, being formed from noisy data.

In addition, both visualization techniques have tunable (hyper-)parameters that significantly affect the visualization results: the factor balancing attention between the local and global aspects of in the data (perplexity) in t-SNE and both the number of neighbors (k) and the desired separation between close points (min-dist) in UMAP. Therefore, to make a reasonable interpretation, it is desirable to analyze multiple plots being built varying the algorithm parameters [250].

Several examples of shape and topology distortions when projecting 3D data into 2D using t-SNE are shown in Figure 2.8. UMAP, being constructed from a theoretical framework based on Riemannian geometry and algebraic topology, typically better captures global data structure and is more stable compared to t-SNE, as shown in Figure 2.9 [157]. The computation complexity of UMAP is bounded by the approximate nearest neighbor search complexity, which is empirically $O(n^{1.14})$ for n data items [48], and scales linearly with the number of dimensions unlike exponential dependency in t-SNE. Both the computational and the memory complexity of t-SNE are $O(n^2)$, so random sampling is used to visualize large datasets [146]. Unlike t-SNE, UMAP supports a wide variety of distance functions (including non-metric distance

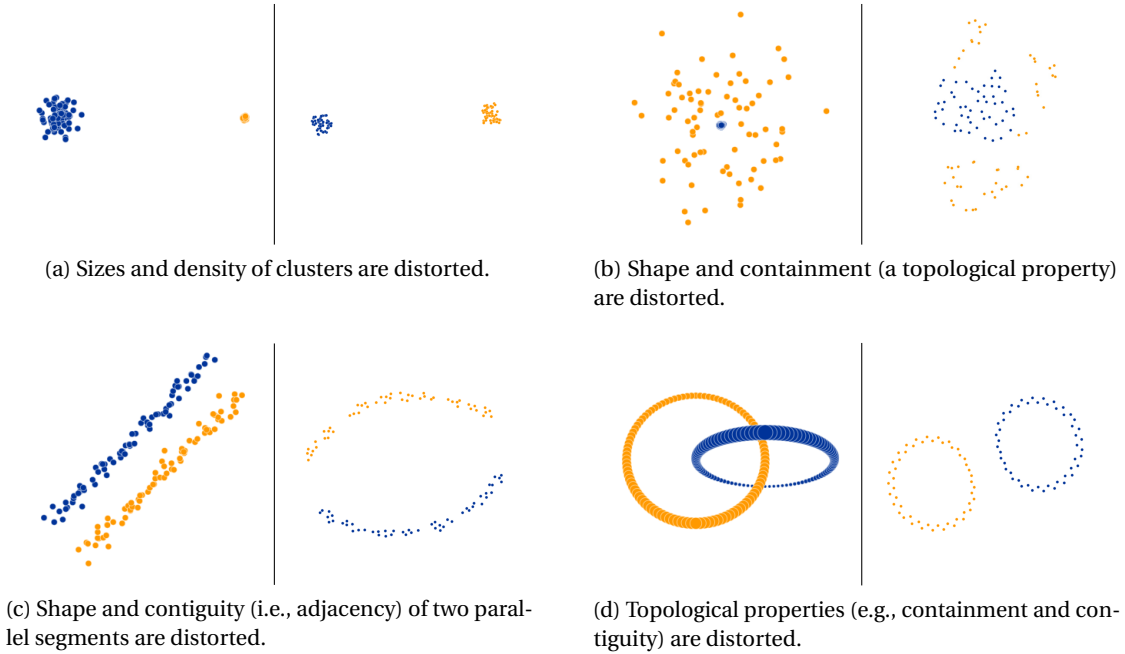
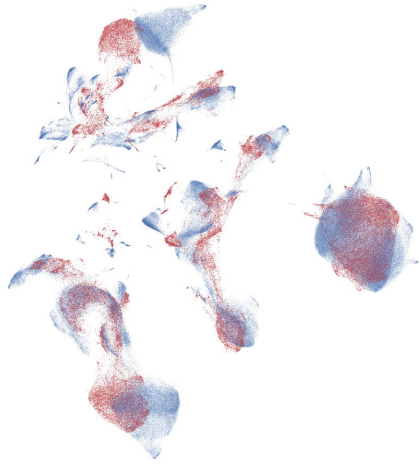
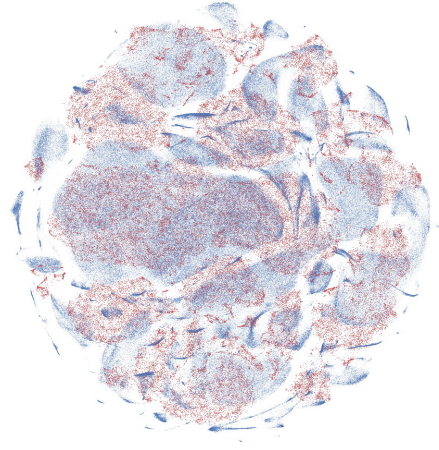


Figure 2.8 – Visualization pitfalls of t-SNE, where the original data is displayed on the left hand side and the visualization with t-SNE on the right hand side using the perplexity within the middle of the recommended range: 0.15 .. 0.30 [250].

functions) and iterative extensions when building the embeddings with new points. UMAP (and also t-SNE) used in unsupervised mode, despite providing stable and human-readable visualizations, does not necessarily visualize the clusters as expected when comparing them to the respective ground-truth as shown see Figure 2.10. Therefore, lossy visualizations should be analyzed carefully and validated with other means (such as lossless visualization techniques) when correct interpretation of the data is essential, as shown in Figure 2.5.



(a) UMAP visualization.



(b) t-SNE visualization.

Figure 2.9 – Procrustes based alignment of a 10% subsample (red) against the full dataset (blue) for the flow cytometry dataset [157].

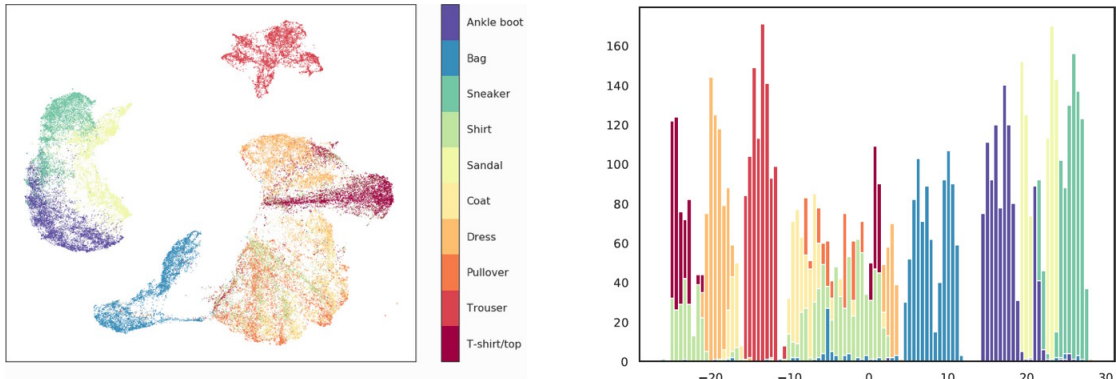


Figure 2.10 – Unsupervised UMAP visualization in 2D and 1D of the Zalando Fashion MNIST, where unexpected occlusions of some ground-truth categories are clearly visible [Mci20].

2.6 Research Focus

Based on the information provided in this chapter, we narrow down each research question stated in Section 1.1 to more formal and measurable research objectives as follows.

1. How to *structure* large amounts of (evolving) information without any manual tuning to return a *human-readable representation* that is suitable for visual exploration? As discussed in Section 2.3, the representation of information adapted to human perception involves a taxonomy having specific properties, which are mapped onto specific requirements imposed to the clustering algorithm. Our algorithm, DAOC, which is designed to conform to the outlined requirements, is presented in Chapter 3.
2. What are the *measures* that are suitable to evaluate the *accuracy* of the resulting representation? The resulting representation is built via some specific clustering, whose evaluation requires specific metrics taking into account both the a) properties of the clustering algorithm and b) formal constraints for the measures themselves (see Section sec:clsobj). The respective measures are discussed both in terms of efficiency of the evaluation process and compliance with the formal constraints in Chapter 4.
3. How to *benchmark* our structuration process and compare it to the existing solutions? Since our structuration process is a kind of clustering, a specific benchmarking process for clustering is elaborated to take into account the imposed constraints that were formulated in Section 2.3. Both the methodology of the benchmarking and its corresponding open-source framework are introduced in Chapter 5.
4. How can the resulting representation facilitate the *refinement of knowledge graphs* to infer missing information? The resulting representation preserves statistical properties of the underlying graph simplifying the identification and inference of missing information, which we formalize as an entity type completion problem for Knowledge Graphs in Chapter 6.
5. How to build *interpretable graph embeddings* without any manual tuning relying on our representation?

The generated representation, i.e., a stable and fine-grained hierarchy of overlapping and multi-resolution clusters, can be converted to interpretable graph embeddings in a fully automatic manner, which is presented in Chapter 7 and opens the door to wider applications as discussed in Chapter 8.

2.7 Notations

The main notations used in this paper are listed in Table 2.2.

Table 2.2 – Notations

$\#i$	Node $\langle i \rangle$ of the graph (network) \mathcal{G}
c_i	Cluster $\langle i \rangle$ of the graph \mathcal{G}
$j \sim i$	Items i and j (nodes or clusters) are linked
n	The number of nodes in the graph \mathcal{G}
m	The number of links in the graph \mathcal{G}
k	The number of clusters (communities) in the graph \mathcal{G}
w	Weight of the graph \mathcal{G}
w_i	Weight of $\#i$
\dot{w}_i	Weight of c_i : $\dot{w}_i = w_{c_i}$
Q	<i>Modularity</i>
$\Delta Q_{i,j}$	<i>Modularity Gain</i> between $\#i$ and $\#j$
ΔQ_i	<i>Maximal Modularity Gain</i> for $\#i$: $\Delta Q_i = \{\max \Delta Q_{i,j} \mid \#j \sim \#i\}$
γ	<i>Resolution</i> parameter
$MMG_{i,j}$	<i>Mutual Maximal Gain</i> : $MMG_{i,j} = \{\Delta Q_{i,j} \mid \Delta Q_i = \Delta Q_j, \#j \sim \#i\}$
ccs_i	Mutual clustering candidates of $\#i$ (by MMG_i)
s	The number of salient clusters (features), $s \leq k$
d	The number of embedding dimensions: $d = D , d \leq s$

3 DAOC: Stable Clustering of Large Networks Without Manual Tuning

In this chapter, we introduce our novel clustering algorithm, DAOC, which is designed to build a stable hierarchy of clusters without any manual tuning for any input network, addressing our requirements for a *human-readable representation* of information (see Chapter 2.1).

3.1 Introduction

Clustering is a fundamental part of data mining with a wide applicability to statistical analysis and exploration of physical, social, biological and information systems. Modeling and analyzing such systems often involves processing large complex networks [127]. Clustering large networks is intricate in practice, and should ideally provide *stable* results in an efficient way in order to make the process easier for the data scientist (see Chapter 2.1).

Stability is pivotal for many data mining tasks since it allows to better understand whether the results are caused by the evolving structure of the network, by evolving node ids (updated labels, coordinates shift or nodes reordering), or by some fluctuations in the application of non-deterministic algorithms. Stability of the results involves both determinism and robustness. We refer to the term *deterministic* in the strictest sense denoting algorithms that *a)* do not involve any stochastic operations and *b)* produce results invariant of the nodes processing order. *Robustness* ensures that clustering results gracefully evolve with small perturbations or changes in the input network [100]. It prevents sudden changes in the output for dynamic networks and provides the ability to tolerate noise and outliers in the input data [45].

Clustering a network is usually not a one-off project but an iterative process, where the results are visually explored and refined multiple times. The visual exploration of large networks requires to consider the specificities of human perception [19, 162] which is good at handling *fine-grained hierarchies* of clusters. In addition, those hierarchies should be stable across iterations such that the user can compare previous results with new results. This calls for results that are both stable and fine-grained.

In this chapter, we introduce a novel clustering method called DAOC to address the afore-

mentioned issues. To the best of our knowledge, DAOC¹ is the first parameter-free clustering algorithm that is simultaneously deterministic, robust and applicable to large weighted networks yielding a fine-grained hierarchy of overlapping clusters. More specifically, DAOC leverages *a*) a novel consensus technique we call *Mutual Maximal Gain (MMG)* to perform a robust and deterministic identification of node membership in the clusters, and *b*) a new technique for *overlap decomposition (OD)* to form fine-grained clusters in a deterministic way, even when the optimization function yields a set of structurally different but numerically equivalent optima (see *degeneracy* in Section 2.3.3). We empirically evaluate the stability of the resulting clusters produced by our approach, as well as its efficiency and effectiveness on both synthetic and real-world networks. We show that DAOC yields stable clusters while being on average 25% more accurate than state-of-the-art deterministic clustering algorithms and more efficient than state-of-the-art overlapping clustering algorithms without requiring any manual tuning. In addition, we show that DAOC returns on average more accurate results than any state-of-the-art clustering algorithm on complex real-world networks (e.g., networks with overlapping and nested clusters). We foresee DAOC to represent an important step forward for clustering algorithms as: *a*) deterministic clustering algorithms are usually not robust and have a lower accuracy than their stochastic counterparts, and *b*) robust methods are typically not deterministic and do not provide fine-grained results as they are insensitive to micro-scale changes, as described in more detail in the following section.

3.2 Related Work

A great diversity of clustering algorithms can be found in the literature. Below, we give an overview of prior methods achieving robust results, before describing deterministic approaches and outlining a few widely used algorithms that are neither robust nor deterministic but were inspirational for our method.

Robust clustering algorithms typically leverage *consensus* or *ensemble* techniques [64, 243, 125, 148]. They identify clusters using consensus functions (e.g., majority voting) by processing an input network multiple times and varying either the parameters of the algorithm, or the clustering algorithm itself. However, such algorithms typically *a*) are unable to detect fine-grained structures due to the lack of consensus therein, *b*) are stochastic and *c*) are inapplicable to large networks due to their high computational cost. We describe some prominent and scalable consensus clustering algorithms below.

Order Statistics Local Optimization Method (OSLOM) [124] is one of the first widely used consensus clustering algorithms, which accounts for weights of the network links and yields overlapping clusters with a hierarchical structure. It is based on the local optimization of a fitness function expressing the statistical significance of clusters with respect to random fluctuations. OSLOM scales near linearly on sparse networks but has a relatively high compu-

¹<https://github.com/eXascaleInfolab/daoc>

tational complexity at each iteration, making it inapplicable to large real-world networks (as we show in Section 3.4).

Core Groups Graph Clustering Randomized Greedy (CGGC[i]_RG) [180] is a fast and accurate ensemble clustering algorithm. It applies a generic procedure of ensemble learning called Core Groups Graph Clustering (CGGC) to determine several weak graph (network) clusterings and then to form a strong clustering from their maximal overlap. The algorithm has a near linear computational complexity with the number of edges due to the sampling and local optimization strategies applied at each iteration. However, this algorithm is designed for unweighted graphs and produces flat and non-overlapping clusters only, which limits its applicability and yields low accuracy on large complex networks as we show in Section 3.4.

Fast Consensus technique was recently proposed and works on top of state-of-the-art clustering algorithms including Louvain (FCoLouv), Label Propagation (FCoLPM) and Infomap (FCoIMap) [226]. The technique initializes a consensus matrix and then iteratively refines it until convergence as follows. First, the input network is clustered by the original algorithm multiple times. The consensus values $D_{i,j} \in [0, 1]$ of the matrix are evaluated as the fraction of the runs in which nodes i and j belong to the same cluster. The consensus matrix is formed using pairs of co-clustered adjacent nodes and extended with closed triads instead of all nodes in the produced clusters, which significantly reduces the amount of computation. The formed matrix is filtered with a threshold τ and then clustered n_p times by the original clustering algorithm, producing a refined consensus matrix. This refinement process is repeated until all runs produce identical clusters (i.e., until all values in the consensus matrix are either zero and one) with precision $1 - \delta$. The Fast Consensus technique however lacks a convergence guarantee and relies on three parameters having a strong impact on its computational complexity.

Deterministic clustering algorithms and, in general, non-stochastic ones (i.e., algorithms relaxing the determinism constraint) are typically not robust and are sensitive to both *a*) initialization [224, 27, 232, 85] (including the order in which the nodes are processed) and *b*) minor changes in the input network, which may significantly affect the clustering results [45, 125]. Non-stochastic algorithms also often yield less precise results getting stuck on the same local optimum until the input is updated. Multiple local optima often exist due to the *degeneracy* phenomenon, which is explained in Section 2.3.3 and has to be specifically addressed to create deterministic clustering algorithms that are both robust and accurate. We describe below some of the well-known deterministic algorithms.

Clique Percolation method (CPM) [46] is probably the first deterministic clustering algorithm supporting overlapping clusters and capable of providing fine-grained results. *Sequential algorithm for fast clique percolation (SCP)* [119] is a CPM-based algorithm, which detects k -clique clusters in a single run and produces a dendrogram of clusters. SCP produces deterministic and overlapping clusters at various scales, and shows a linear dependency of the computational complexity with the number of k -cliques in the network. However, SCP relies

on a number of parameters and has an exponential worst case complexity in dense networks, which significantly limits its practical applicability.

pSCAN [31] is a fast overlapping clustering algorithm for “exact structural graph clustering” (i.e., it is deterministic and input-order independent). First, it identifies core graph vertices (network nodes), which always belong to exactly one cluster, forming initially disjoint clusters. The remaining nodes are then assigned to the initial clusters, yielding overlapping clusters. *pSCAN* relies on two input parameters, $0 < \epsilon \leq 1$ and $\mu \geq 2$. The results it produces are very sensitive to those parameters, whose optimal values are hard to guess for arbitrary input networks.

Inspirational algorithms for our method

Louvain [17] is a commonly used clustering algorithm that performs modularity optimization using a local search technique on multiple levels to coarsen clusters. It introduces modularity gain as an optimization function. The algorithm is parameter-free, returns a hierarchy of clusters, and has a near-linear runtime complexity with the number of network links. Moreover, Louvain is one of the most well-studied and accurate clustering algorithms to discover non fine-grained communities² in large networks [225, 62, 264]. However, the resulting clusters are not stable and depend on the order in which the nodes are processed. Similarly to Louvain, our method is a greedy agglomerative clustering algorithm, which uses modularity gain as optimization function. However, the clusters formation process in DAOC differs a lot, addressing the aforementioned issues of the Louvain algorithm.

DBSCAN [55] is a density-based clustering algorithms suitable to process data with noise. It regroups points that are close in space given the maximal distance between the points ϵ and the minimal number of points *MinPts* within an area. DBSCAN is limited in discovering a large variety of clusters because of its reliance to a density parameter. It has a strong dependency on input parameters, and lacks a principled way to determine optimal values for these parameters [33]. We adopt however the DBSCAN idea of clusters formation based on the extension of the densest region to prevent early coarsening and to produce a fine-grained hierarchical structure.

3.3 Method

We introduce a novel clustering algorithm, DAOC, to perform a stable (i.e., both *robust* and *deterministic*) clustering of the input network, producing a fine-grained hierarchy of overlapping clusters. DAOC is a greedy algorithm that uses an agglomerative clustering approach with a local search technique (inspired by Louvain [17]) and extended with two novel techniques. Namely, we first propose a novel (micro) consensus technique called *Mutual Maximal Gain (MMG)* for the robust identification of nodes membership in the clusters, which is performed

²Relying on modularity, Louvain suffers from the *resolution limit* (see Chapter 2 and Chapter 7), which complicates detection of fine-grained clusters [61, 72, 151]

in a deterministic and fine-grained manner. In addition to MMG, we also propose a new *overlap decomposition (OD) technique* to cope with the degeneracy of the optimization function. OD forms stable and fine-grained clusters in a deterministic way from the nodes preselected by MMG.

Algorithm 1 gives a high-level description of our method. It takes as input a directed and weighted network with self-loops specifying node weighs. The resulting hierarchy of clusters is built iteratively starting from the bottom level (the most fine-grained level). One level of the hierarchy is generated at each iteration of our clustering algorithm. A clustering iteration consists of the following steps listed on lines 4–5:

1. Identification of the clustering candidates ccs_i for each node $\#i$ using the proposed consensus approach, MMG, described in Section 3.3.1 and
2. Cluster formation considering overlaps, described in Section 3.3.2.

Algorithm 1 DAOC Clustering Algorithm.

```

1: function CLUSTER(nodes)
2:   hier  $\leftarrow$  [] ▷ List of the hierarchy levels
3:   while nodes do ▷ Stop if the nodes list is empty
4:     identifyCands(nodes) ▷ Initialize nd.ccs
5:     cls  $\leftarrow$  formClusters(nodes)
6:     if cls then ▷ Initialize the next-level nodes
7:       for all cl  $\in$  cls do
8:         initCluster(cl)
9:       end for
10:      for all nd  $\in$  nodes do ▷ Consider propagates
11:        if nd.propagated then
12:          initNode(nd)
13:          cls.append(nd)
14:        end if
15:      end for
16:      hier.append(cls) ▷ Extend the hierarchy
17:    end if
18:    nodes  $\leftarrow$  cls ▷ Update the processing nodes
19:  end while
20:  return hier ▷ The resulting hierarchy of clusters
21: end function

```

At the end of each iteration, links are (re)computed for the formed clusters (*initCluster* procedure) and for the non-clustered nodes (*propagated* nodes in the *initNode* procedure). Both the non-clustered nodes and the formed clusters are treated as input nodes for the following iteration. The algorithm terminates when the iteration does not produce any new cluster.

The clustering process yields a hierarchy of overlapping clusters in a deterministic way independent of the nodes processing order, since all clustering operations *a*) consist solely of non-stochastic, uniform and local operations, and *b*) process each node independently, relying on immutable data evaluated on previous steps only. The algorithm is guaranteed to converge since *a*) the optimization function is bounded (as outlined in Section 3.3.1) and monotonically increasing during the clustering process, and *b*) the number of formed clusters does not exceed the number of clustered nodes at each iteration (as explained in Section 3.3.2).

3.3.1 Identification of the Clustering Candidates

The *clustering candidates* are the nodes that are likely to be grouped into clusters in the current iteration. The clustering candidates are identified for each node (*nd.ccs*) in two steps as listed in Algorithm 2. First, for each node *nd* the adjacent nodes ($\{link.dst \mid link \in nd.links\}$) having the maximal non-negative value *nd.gmax* of the optimization function *optfn* are stored in the *nd.ccs* sequence, see lines 3–11. Then, the preselected *nd.ccs* are reduced to the mutual candidates by the *mcands* procedure, and the filtered out nodes are marked as propagated. The latter step is combined with a cluster formation operation in our implementation to avoid redundant passes over all nodes. The *mcands* procedure implements our *Maximal Mutual Gain (MMG)* consensus approach described in Section 3.3.1, which is a

Algorithm 2 Clustering Candidates Identification

```
1: function IDENTIFYCANDS(nodes)
2:   for all nd  $\in$  nodes do                                ▷ Evaluate clustering candidates
3:     nd.gmax  $\leftarrow$  -1                                       ▷ Maximal gain
4:     for all ln  $\in$  nd.links do
5:       cgain  $\leftarrow$  optfn(nd, ln)                        ▷ Clustering gain
6:       if cgain  $\geq$  0 and cgain  $\geq$  nd.gmax then
7:         if cgain > nd.gmax then
8:           nd.ccs.clear()                                     ▷ Reset cand
9:           nd.gmax  $\leftarrow$  cgain
10:        end if
11:        nd.ccs.append(ln.dst)                                ▷ Extend cand
12:      end if
13:    end for
14:  end for

15:  for all nd  $\in$  nodes do                                ▷ Reduce the candidates using the consensus approach,
    propagate remained nodes
16:    if nd.gmax < 0 or not mcands(nd) then
17:      nd.propagated  $\leftarrow$  true
18:    end if
19:  end for
20: end function
```

meta-optimization technique that can be applied on top of any optimization function that satisfies a set of constraints described in the following paragraph.

Optimization Function

In order to be used in our method, the optimization function optfn should be *a)* applicable to pairwise node comparison, i.e. $\exists \text{optfn}(\#i, \#j) \mid \#i \sim \#j$ (adjusted pair of nodes); *b)* commutative, i.e. $\text{optfn}(\#i, \#j) = \text{optfn}(\#j, \#i)$; and *c)* bounded on the non-negative range, where positive values indicate some quality improvement in the structure of the forming cluster. There exist various optimization functions satisfying these constraints besides modularity and inverse conductance (see the list in [28], for instance).

Our DAOC algorithm uses *modularity gain*, ΔQ (see (2.2)), as an optimization function. We chose modularity (gain) optimization because of the following advantages. First, modularity maximization (under certain conditions) is equivalent to the provably correct but computationally expensive methods of graph partitioning, spectral clustering and to the maximum likelihood method applied to the stochastic block model [173, 145]. Second, there are known and efficient algorithms for modularity maximization, including the Louvain algorithm [17], which are accurate and have a near-linear computational complexity.

MMG Consensus Approach

We propose a novel (micro) consensus approach, called *Mutual Maximal Gain (MMG)* that requires only a single pass over the input network, is more efficient and yields much more fine-grained results compared to state-of-the-art techniques.

Definition 1 (Mutual Maximal Gain (MMG)) *MMG is a value of the optimization function (in our case modularity gain) for two adjacent nodes $\#i$ and $\#j$, and is defined in cases where these nodes mutually reach the maximal value of the optimization function (i.e., reach consensus on the maximal value) when considering each other:*

$$\text{MMG}_{i,j} = \{\Delta Q_{i,j} \mid \Delta Q_i = \Delta Q_j, \#j \sim \#i\} \quad (3.1)$$

where \sim denotes adjacency of $\#i$ and $\#j$, and ΔQ_i is the maximal modularity gain for $\#i$:

$$\Delta Q_i = \{\max \Delta Q_{i,j} \mid \#j \sim \#i\} \quad (3.2)$$

where $\Delta Q_{i,j}$ is the modularity gain for $\#i$ and $\#j$ (see (2.2)).

MMG exists in any finite network, which can be easily proven by contradiction as follows. The nonexistence of MMG would create a cycle with increasing $\max \Delta Q$ and results in $\Delta Q_i < \Delta Q_i$ considering that $\forall \#i \exists \Delta Q_i: (\Delta Q_i = \Delta Q_{i,j}) < (\Delta Q_j = \Delta Q_{j,k}) < \dots < (\Delta Q_t = \Delta Q_{t,j} = \Delta Q_j)$, i.e. $\Delta Q_j < \Delta Q_j$, which yields a contradiction. MMG evaluation is deterministic and the result-

ing nodes are *quasi-uniform* clustering candidates, in the sense that inside each connected component they share the same maximal value of modularity gain. MMG takes into account fine-grained clusters as it operates on pairs of nodes, unlike conventional consensus approaches, where micro-clusters either require lots of reexecutions of the consensus algorithm, or cannot be captured at all. MMG does not always guarantee optimal clustering results but reduces degeneracy due to the applied consensus approach. According to (3.1), all nodes having MMG to #i have the same value of ΔQ_i , i.e., form the overlap in #i. Overlaps processing is discussed in the following section.

3.3.2 Clusters Formation with Overlap Decomposition

Clusters are formed from candidate nodes selected by MMG as listed in Algorithm 3: *a*) nodes having a single mutual clustering candidate (*cc*) form the respective cluster directly as shown on line 8, *b*) otherwise the overlap is processed. There are three possible ways of clustering an overlapping node in a deterministic way: *a*) split the node into *fragments* to have one fragment

Algorithm 3 Clusters Formation

```

1: function FORMCLUSTERS(nodes)
2:   cls  $\leftarrow$  [] ▷ List of the formed clusters
3:   for all nd  $\in$  nodes do
4:     if nd.propagated then ▷ Prefilter nodes
5:       continue
6:     end if

7:     if nd.ccs.size = 1 then ▷ Form a cluster
8:       merge(cls, nd, nd.ccs)
9:     else if odAccept(nd) and gainEach(nd) > gainAll(nd) then ▷ Form overlapping
        clusters
10:      for all cand  $\in$  nd.ccs do
11:        mergeOvp(cls, nd, cand)
12:      end for
13:    else ▷ DBSCAN inspired aggregation
14:      rccs  $\leftarrow$  maxIntersectOrig(nd)
15:      if rccs.size  $\geq$  1 then ▷ Form a single cluster
16:        merge(cls, nd, rccs)
17:      else if gainAll(nd)  $\geq$  0 then ▷ Form a cluster
18:        merge(cls, nd, nd.ccs)
19:      else
20:        nd.propagated  $\leftarrow$  true
21:      end if
22:    end if
23:  end for
24:  return cls ▷ Resulting clusters
25: end function
  
```

per each *cc* of the node and group each resulting fragment with the respective *cc* into the dedicated cluster (see lines 10–11), or *b*) group the node together with all its *nd.ccs* items into a single cluster (i.e. coarsening on line 18), or *c*) propagate the node to be processed on the following clustering iteration if its clustering would yield a negative value of the optimization function. Each node fragment created by the overlap decomposition is treated as a virtual node representing the belonging degree (i.e., the fuzzy relation) of the original node to multiple clusters. Virtual nodes are used to avoid the introduction of the fuzzy relation for all network nodes (i.e., to avoid an additional complex node attribute) reducing memory consumption and execution time, and not affecting the input network itself. In order to get the most effective clustering result, we evaluate the first two aforementioned options and select the one maximizing the optimization function, ΔQ . Then, we form the cluster(s) by the merge or merge0vp procedures as follows. The merge0vp procedure groups each node fragment (i.e., the virtual node created by the overlap decomposition) together with its respective mutual clustering candidate. This results in either *a*) an extension of the existing cluster to which the candidate already belongs to, or *b*) the creation of a new cluster and its addition to the *cls* list. The merge procedure groups the node with all its clustering candidates either *a*) merging together the respective clusters of the candidates if they exists, or *b*) creating a new cluster and adding it to the *cls* list.

Node splitting is the most challenging process, which is performed only if the accumulated gain from the decomposed node fragments to each of the respective mutual clustering candidates, *nd.ccs*, (*gainEach* procedure) exceeds the gain of grouping the whole node with all *nd.ccs* (*gainAll* procedure). The node splitting involves: *a*) the estimation of the node fragmentation impact on the clustering convergence (*odAccept* procedure given in Section 3.3.2) and *b*) the evaluation of the weights for both the forming fragment and for the links between the fragments of the splitting node as described in Section 3.3.2.

Overlap Decomposition (OD)

An overlap occurs when a node has multiple mutual clustering candidates (*ccs*). To evaluate ΔQ when clustering the node with each of its K *ccs*, the node is split into K identical and fully interconnected fragments sharing the node weight and original node links. However, since the objective of the clustering is the maximization of ΔQ : *a*) the node splitting itself is acceptable only in case the resulting $\Delta Q \geq 0$, and *b*) the decomposed node can be composed back from the fragments only in case $\Delta Q \leq 0$. Hence, to have a reversible decomposition without affecting the value of the optimization function for the decomposing node, we end up with $\Delta Q = 0$.

The outlined constraints for an isolated node, which does not have any link to other nodes,

can formally be expressed as:

$$\begin{cases} w = \sum_{k=1}^K w_k + \sum_{k=1}^K \sum_{t=1}^{K-1} \frac{w_{k,t}}{2} \\ \sum_{k=1}^K w_k - \sum_{k=1}^K \frac{(w_k + \sum_{t=1}^{K-1} \frac{w_{k,t}}{2})^2}{w} = 0 \end{cases}, \quad (3.3)$$

where w is the weight of the original node being decomposed into K fragments, w_k is the weight of each node fragment $k \in K$ and $w_{k,t}$ is the weight of each link between the fragments. $\Delta Q = Q_{split} - Q_{node} = Q_{split}$ since the modularity of the isolated node is zero (see (2.1)). The solution of (3.3) is:

$$w_k = \frac{w}{K^2}, \quad w_{k,t} = 2 \frac{w}{K^2}. \quad (3.4)$$

Nodes in the network typically have links, which get split equally between the fragments of the node:

$$\forall k, t \in \{1..K\} \mid \#j \sim \#i: \begin{cases} w_{ik} = \frac{w_i}{K^2} \\ w_{ik,it} = 2 \frac{w_i}{K^2} \\ w_{ik,j} = \frac{w_{i,j}}{K} \end{cases}, \quad (3.5)$$

where w_{ik} is the weight of each fragment $\#ik$ of the node $\#i$, $w_{i,j}$ is the weight of the link $\{\#i, \#j\}$.

Example 1 (Overlap Decomposition) The input network on the left-hand side of Figure 3.1 has node C with neighbor nodes $\{A, B, D\}$ being ccs of C . These neighbor nodes form the respective clusters $\{C1, C2, C3\}$ overlapping in C . C is decomposed into fragments $\{C_A, C_B, C_D\}$

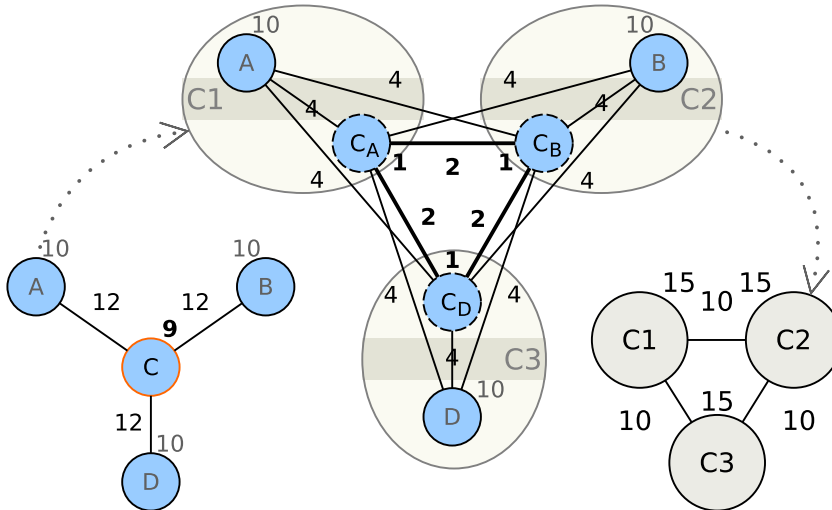


Figure 3.1 – Decomposition of the clusters $C1(A, C)$, $C2(B, C)$, $C3(D, C)$ overlapping in node C of the input network $\{A, B, C, D\}$ with weights on both nodes and links.

to evaluate the overlap. Node C has an internal weight equal to 9 (which can be represented via an additional edge to itself) and three edges of weight 12 each. The overlapping clusters are evaluated using (3.5) as equivalent and virtual non-overlapping clusters formed using the new fragments of the overlapping node:

$$\forall k, t \in \{1 \dots K\} \mid \#j \sim \#i : \begin{cases} w_{ik} = \frac{9}{3^2} = 1 \\ w_{ik,it} = 2 \frac{9}{3^2} = 2 \\ w_{ik,j} = 2 + \frac{12}{3} = 6 \quad (\forall j \neq k) \\ w_{ik,k} = \frac{12}{3} = 4 \end{cases}$$

$$w_{CA} = w_{CB} = w_{CD} = w_{ik} = 1,$$

$$w_{C1} = w_{C2} = w_{C3} = w_k + w_{ik} + w_{ik,k} = 10 + 1 + 4 = 15,$$

$$w_{C_k, C_t} = w_{ik,t} + w_{it,k} - w_{ik,it} = 6 + 6 - 2 = 10.$$

Constraining Overlap Decomposition

Overlap decomposition (OD) does not affect the value of the optimization function for the node being decomposed ($\Delta Q = 0$), hence it does not affect the convergence of the optimization function during the clustering. However, OD increases the complexity of the clustering when the number of produced clusters exceeds the number of clustered nodes decomposed into multiple fragments. This complexity increase should be identified and prevented to avoid indefinite increases in terms of clustering time.

In what follows, we infer a formal condition that guarantees the non-increasing complexity of OD. We decompose a node of degree d into k fragments, $2 \leq k \leq d$. Each forming cluster that has an overlap in this node owns one fragment (see Figure 3.1) and shares at most $d - k$ links to the non-*ccs* neighbors of the node. The number of links between the k fragments resulting in the node split is $k \times \frac{k-1}{2}$. The aggregated number of resulting links should not exceed the degree of the node being decomposed to retain the same network complexity, therefore:

$$\begin{cases} k(d - k) + k \frac{k-1}{2} \leq d \\ 2 \leq k \leq d \end{cases} \quad (3.6)$$

The solution of (3.6) is $2 \leq k \leq d \leq 3$, namely: $k = 2, d = \{2, 3\}$; $k = 3, d = 3$.

If a node being decomposed has a degree $d \geq 3$ or a node has more than k *ccs* then, before falling back to the coarse cluster formation, we apply the following heuristic inspired by the DBSCAN algorithm [55]. We evaluate the intersection of *nd.ccs* with each $\{c.ccs \mid c \in nd.ccs\}$ (`maxIntersectOrig` procedure on line 14 of Algorithm 3) and group the node with its clustering candidate(s) yielding the maximal intersection if the latter contains at least half of the *nd.ccs*. In such a way, we try prevent an early coarsening and obtain more fine-grained

and accurate results.

3.3.3 Complexity Analysis

The computational complexity of DAOC on sparse networks is $O(m \cdot \log m)$, where m is the number of links in the network. All links of each node ($d \cdot n = m$) are processed for $\log m$ iterations. In the worst case, the number of iterations is equal to the number of nodes n (instead of $\log m$) and the number of mutual candidates is equal to the node degree d instead of 1. Thus, the theoretical worst-case complexity is $O(m \cdot d \cdot n) = O(m^2)$ and occurs only in a hypothetical dense symmetric network having equal MMG for all links (and, hence, requiring overlap decomposition) on each clustering iteration and in case the number of clusters is decreased at each iteration by one only. The memory complexity is $O(m)$.

3.4 Experimental Evaluation

3.4.1 Evaluation Environment

Our evaluation was performed using an open-source parallel isolation benchmarking framework, Clubmark³ [140], on a Linux Ubuntu 16.04.3 LTS server with the Intel Xeon CPU E5-2620 v4 @ 2.10GHz CPU (16 physical cores) and 132 GB RAM. The execution termination constraints for each algorithm are as follows: 64 GB of RAM and 72 hours max per network clustering. Each algorithm is executed on a single dedicated physical CPU core with up to 64 GB of guaranteed available physical memory.

Table 3.1 – Evaluating clustering algorithms.

Features \ Algs	<i>Daoc</i>	<i>Scp</i>	<i>Lvn</i>	<i>Fcl</i>	<i>Osl2</i>	<i>Gnx</i>	<i>Psc</i>	<i>Cgr</i>	<i>Cgri</i>	<i>Scd</i>	<i>Rnd</i>
Hierarchical	+		+		+						
Multi-scale	+	+	+		+	+					
Deterministic	+	+					+				
Overlapping clusters	+	+			+	+	+				
Weighted links	+	+	+	◦	+	+					+
Parameter-free	+!		+	*	*	*		*	*	*	+
Consensus/Ensemble	+			+	+			+	+		

Deterministic includes input-order invariance;

+! the feature is available, still the ability to force custom parameters is provided;

* the feature is partially available, parameters tuning might be required for specific cases;

◦ the feature is available in theory but is not supported by the original implementation of the algorithm.

We compare DAOC against almost a dozen state-of-the-art clustering algorithms listed in Table 3.1 (the original implementations of all algorithms except Louvain are included into Clubmark and are executed as precompiled or JIT-compiled applications or libraries) and

³<https://github.com/eXascaleInfolab/clubmark>

described in the following papers: SCP [119], Lvn(Louvain⁴ [17]), Fcl (Fast Consensus on Louvain: FCoLouv [226]), Osl2(OSLOM2 [124]), Gnx(GANXiS also known as SLPA [255]), Psc(pSCAN [31]), Cgr[i](CGGC[i]_RG [180]), SCD [196] and Rnd(Randcommuns [140]). We have not evaluated a well known CPM-based overlapping clustering algorithm, CFinder [182], because *a*) GANXiS outperforms CFinder in all aspects by several accuracy metrics [255, 254] and *b*) we do evaluate SCP, a fast CPM-based algorithm. For a fair accuracy evaluation, we uniformly sample up to 10 levels from the clustering results (levels of the hierarchical / multilevel output or clusterings produced uniformly varying algorithm parameters in the operational range) and take the best value.

3.4.2 Stability Evaluation

We evaluate stability in terms of both robustness and determinism for the consensus (ensemble) and deterministic clustering algorithms listed in Table 3.1. Determinism (non-stochasticity and input order independence) evaluation is performed on synthetic and real-world networks below, where we quantify the standard deviation of the clustering accuracy. To evaluate stability in terms of robustness, we quantify the deviation of the clustering accuracy in response to small perturbations of the input network. The clustering accuracy on each iteration is measured relative to the clustering yielded by the same algorithm at the previous perturbation iteration. For each clustering algorithm, the accuracy is evaluated only for the middle level (scale or hierarchical level), since it is crucial to take the same clustering scale to quantify structural changes in the forming clusters of evolving networks. Robust clustering algorithms are expected to have their accuracy gradually evolving (without any *surges*) relative to the previous perturbation iteration. In addition, the clustering algorithms sensitive enough to capture the structural changes are expected to have their accuracy *monotonically decreasing* since the relative network reduction (perturbation) is increased at each iteration: X deleted links on iteration i represent a fraction of X/N_i , but on the following iteration this fraction is increased to $X/(N_i - X)$.

We evaluate robustness and sensitivity (i.e., the ability to capture small structural changes) on synthetic networks with nodes forming overlapping clusters generated by the LFR framework [123]. We generate a synthetic network with ten thousand nodes having an average degree of 20 and using the mixing parameter $\mu = 0.275$. This network is shuffled (the links and nodes are reordered) 4 times to evaluate the input order dependence of the algorithms. Small perturbations of the input data are performed gradually reducing the number of links in the network by 2% of the original network size (i.e., $10 \times 1000 \times 20 \times 0.02 = 4000$ links) starting from 1% and ending at 15%. The links removal is performed *a*) randomly to retain the original distributions of the network links and their weights but *b*) respecting the constraint that each node retains at least a single link. This constraint prevents the formation of disconnected regions. Our perturbation does not include any random modification of the link weights or the creation of new links since it would affect the original distributions of the network links

⁴<http://igraph.org/c/doc/igraph-Community.html>

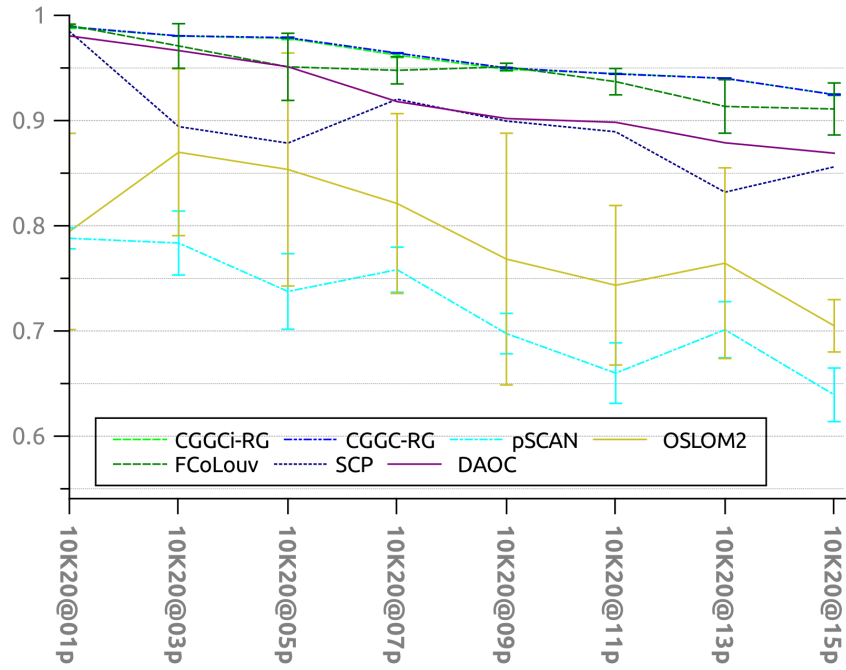
and their weights, causing surges in the clustering results.

The evaluations of stability in terms of robustness (absence of surges in response to small perturbation of the input network) and sensitivity (ability to capture small structural changes) are shown in Figure 3.2. Absolute accuracy values relative to the previous link reduction iteration are shown in Figure 3.2a. The results demonstrate that all deterministic clustering algorithms except DAOC (i.e., pSCAN and SCP) yield surges and hence are not robust, confirming our expectations. We also obtain some unexpected results. First, pSCAN, which is nominally “exact” (i.e., non-stochastic and input-order independent), actually shows significant deviations in accuracy. Second, the clusterings produced by OSLOM2 using default parameters and by FCoLouv using a number of consensus partitions $n_p = 5$ are prone to surges. Hence, OSLOM2 and FCoLouv cannot be classified as robust algorithms according to the obtained results in spite of being a consensus clustering algorithms. Figure 3.2b illustrates the sensitivity of the algorithms, where the relative accuracy values compared to the previous perturbation iteration are shown. Sensitive algorithms have monotonically decreasing results for the subsequent link reduction, which corresponds to positive values on this plot. The stable algorithms (CGGC-RG, CGGCi-RG and DAOC) are highlighted with a bolder line width on the figure. These results demonstrate that being robust, CGGC-RG and CGGCi-RG are not always able to capture structural changes in the network, i.e., they are less sensitive than DAOC. Overall, the results show that only our algorithm, DAOC, is stable (it is deterministic, including input-order independence, and robust) and at the same time is able to capture even small structural changes in the input network.

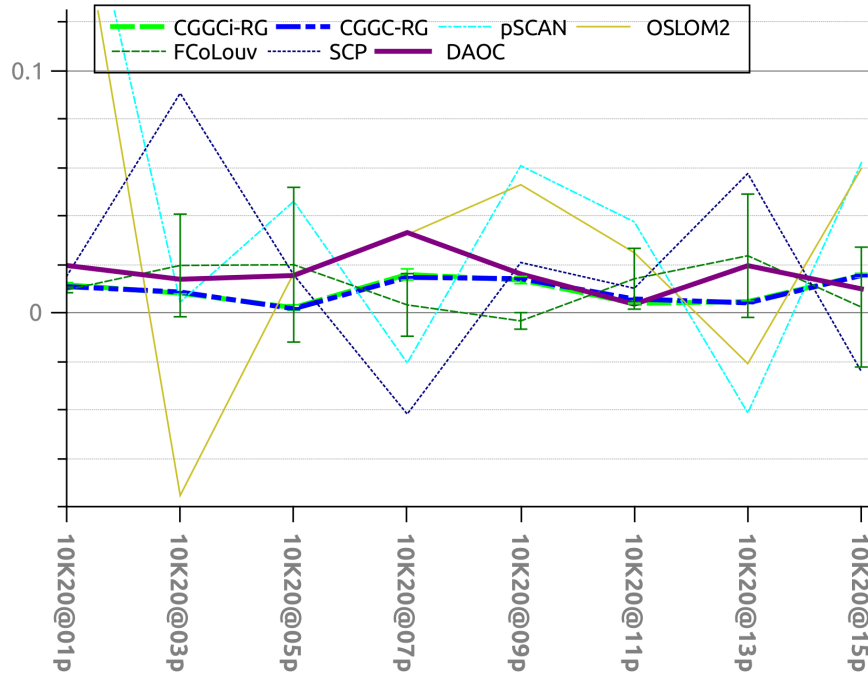
3.4.3 Effectiveness and Efficiency Evaluation

Our performance evaluation was performed both *a)* on weighted undirected synthetic networks with *overlapping* ground-truth clusters produced by the LFR framework integrated into Clubmark [140] and *b)* on large real-world networks having overlapping and nested ground-truth communities⁵ [263]. It worth noting that the RB-LFR framework [265] is superior to LFR for the generation of synthetic networks with *non-overlapping* hierarchical structure of ground-truth clusters. RB-LFR framework generates a clear hierarchical structure of the Ravasz-Barabási model, being extendable to an arbitrary number of hierarchical levels. However, we fell back to the LFR framework for synthetic network generation because of evaluating overlapping clusters. The synthetic networks were generated with 1, 5, 20 and 50 thousands nodes, each having an average node degrees of 5, 25 and 75. The maximal node degree is uniformly scaled from 70 on the smallest networks up to 800 on the largest ones. Synthetic networks generation parameters are taken by default as provided by Clubmark. The real-world networks contain from 334,863 nodes with 925,872 links (amazon) up to 3,997,962 nodes with 34,681,189 links (livejournal). The ground-truth communities of real-world networks were pre-processed to exclude duplicated clusters (communities having exactly the same nodes). Each network is shuffled (reordered) 4 times and the average accuracy value along with its

⁵<https://snap.stanford.edu/data/#communities>



(a) F1h (average value and deviation) for subsequent perturbations (link removals). Stable algorithms are expected to have a gracefully decreasing F1h without any surges.



(b) $\Delta F1h$ relative to the previous perturbation iteration. Stable and sensitive algorithms are highlighted with bolder line width and have positive $\Delta F1h$ evolving without surges. Standard deviation is shown only for the consensus algorithms but visible only for FCoLouv and CGGCI-RG.

Figure 3.2 – Stability and sensitivity evaluation.

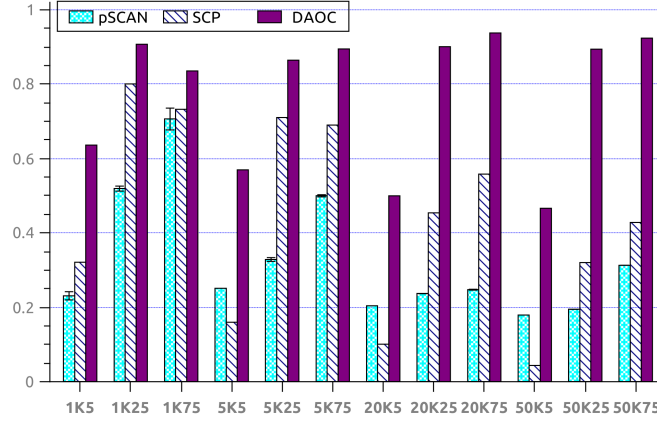


Figure 3.3 – F1h of the deterministic algorithms on the synthetic networks.

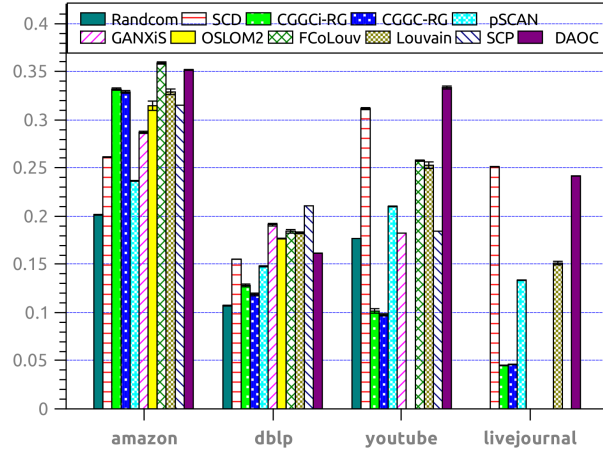
standard deviation are reported.

A number of accuracy measures exist for overlapping clusters evaluation. We are aware of only two families of accuracy measures applicable to large overlapping clusterings, i.e. having a near-linear computational complexity with the number of nodes: the F1 family [141] and generalized NMI (GNMI) [54, 141]. However, mutual information-based measures are biased to a large numbers of clusters while GNMI does not have any bounded computational complexity in general. Therefore, we evaluate clustering accuracy with *F1h* [141], a modification of the popular *average F1-score (F1a)* [262, 196] providing indicative values in the range $[0, 0.5]$, since the artificial clusters formed from all combinations of the input nodes yield $F1a \rightarrow 0.5$ and $F1h \rightarrow 0$.

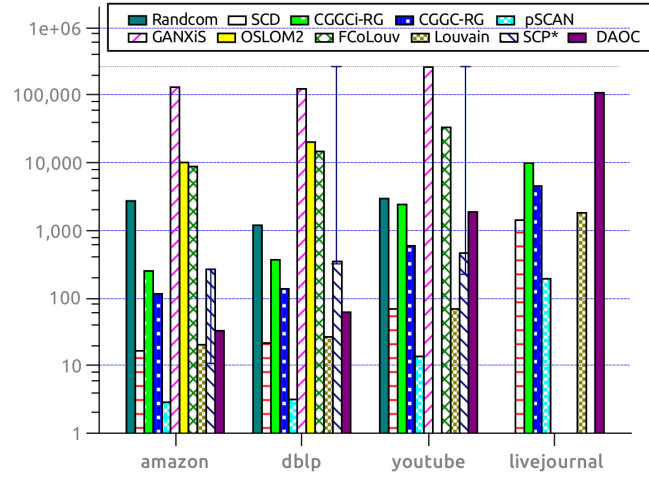
First, we evaluate accuracy for all the deterministic algorithms listed in Table 3.1 on synthetic networks, and then evaluate both accuracy and efficiency for all clustering algorithms on real-world networks. Our algorithm, DAOC, shows the best accuracy among the deterministic clustering algorithms on synthetic networks, outperforming others on each network and being more accurate by 25% on average according to Figure 3.3. Moreover, DAOC also has the best accuracy on average among all evaluated algorithms on large real-world networks as shown in Figure 3.4a. Being parameter-free, our algorithm yields good accuracy on *both* synthetic networks and real-world networks, unlike some other algorithms having good performance on some datasets but low performance on others.

Besides being accurate, DAOC consumes the least amount of memory among the evaluated hierarchical algorithms (Louvain, OSLOM2) as shown in Table 3.2. In particular, DAOC consumes 2x less memory than Louvain on the largest real-world evaluated network (livejournal) and 3x less memory than OSLOM2 on dblp, while producing much more fine-grained hierarchies of clusters with almost an order of magnitude more levels than other algorithms. Moreover, among the evaluated overlapping clustering algorithms, only pSCAN and DAOC are able to cluster the livejournal network within the specified execution constraints, the missing

3.4. Experimental Evaluation



(a) F1h (average value and deviation).



(b) Execution time for a single algorithm run on a single and dedicated CPU core, sec. The range in SCP shows the execution time for $k > 3$.

Figure 3.4 – Performance on the real-world networks.

Table 3.2 – Peak memory consumption (RSS) on the real-world networks, MB.

Nets\Algs	Daoc	Scp*	Lvn	Fcl	Osl2	Gnx	Psc	Cgr	Cgri	Scd	Rnd
amazon	238	3,237	339	3,177	681	3,005	155	247	1,055	37	337
dblp	225	3,909	373	3,435	717	2,879	167	247	1,394	36	373
youtube	737	4,815	1,052	–	–	8,350	508	830	3,865	131	1,050
livejournal	5,038	–	10,939	–	–	–	4,496	4,899	11,037	761	–

– denotes that the algorithm was terminated for violating the execution constraints;

* the memory consumption and execution time for SCP are reported for a clique size $k = 3$ since they grow exponentially with k on dense networks, though accuracy was evaluated varying $k \in 3..7$.

bars in Figure 3.4b corresponding to the algorithms that we had to terminate.

3.5 Conclusions

In this chapter, we presented a new clustering algorithm, DAOC, which is at the same time stable and provides a unique combination of features yielding a fine-grained hierarchy of overlapping clusters in a fully automatic manner (to conform the requirements for the human-readable information representation formulated in Table 2.1 of Chapter 2). We experimentally compared our approach on a number of different datasets and showed that while being parameter-free and efficient, it yields accurate and stable results on *any* input networks. DAOC builds on a new (micro) consensus technique, MMG, and a novel overlap decomposition approach, OD, which are both applicable on top of non-overlapping clustering algorithms and allow to produce overlapping and robust clusters. DAOC is released as an open-source clustering library implemented in C++ that includes various cluster analysis features not mentioned in this chapter and that is integrated with several data mining applications (StaTIX [143], or DAOR [144] embeddings). In future work, we plan to design an approximate version of MMG to obtain near-linear execution times on dense networks, and to parallelize DAOC taking advantage of modern hardware architectures to further expand the applicability of our method.

4 Xmeasures: Accuracy Measures for Overlapping and Multi-resolution Clustering

In this chapter, we answer the question “What are the *measures* that are suitable to evaluate the *accuracy* of our specific clustering algorithm, DAOC (presented in Chapter 3)?”. In particular, we first discuss existing accuracy metrics suitable for the evaluation of overlapping and multi-resolution clustering algorithms on large datasets. Then, we propose several optimizations and extensions of these metrics to improve their effectiveness and satisfaction to formal constraints (described in Chapter 2.3.4) without affecting their efficiency.

4.1 Introduction

As mentioned in Chapter 2.3, clustering is a key component of many data mining systems with numerous applications including statistical analysis and the exploration of physical, social, biological and informational systems. This diversity of potential applications spawned a wide variety of network (graph) clustering algorithms proposed in the literature. It also led to specialized clustering algorithms in particular domains. Hence, the need to find the most suitable and best performing clustering algorithms for a given task became more dire, and the evaluation of the resulting clustering through proper metrics more important. Moreover, as modern systems often operate on very large datasets (consisting of billions of items potentially), the computational properties of the evaluation metrics become more important. In particular, performance-related constraints rapidly emerge when sampling the original large datasets is not desirable or not possible for a given use-case.

Clustering quality metrics can formally be categorized into two types: intrinsic and extrinsic metrics. Intrinsic quality metrics (e.g., modularity [174], conductance [98], silhouette index [204] or betweenness centrality [65]) evaluate how the elements of each cluster are similar to each other and how they differ from elements in other clusters. Extrinsic quality metrics (also known as *accuracy* metrics) evaluate instead how the clusters are similar to the *ground-truth* (gold standard) clusters. In this chapter, we focus on extrinsic metrics since

Chapter 4. Xmeasures: Accuracy Measures for Overlapping and Multi-resolution Clustering

they allow to identify clustering algorithms producing expected results, which is in practice often more useful than measuring the formation of optimal clusters by a given similarity metric. More specifically, we evaluate the similarity (proximity) between two collections of overlapping clusters (unordered sets) of elements, where *a*) the collections have the same number of elements, *b*) each element may be a member of multiple clusters and *c*) each cluster may have several, non-mutual, best (in terms of the similarity value) matches in the other collection.

We call *clustering* the set of clusters resulting from an algorithm. Clusterings can be categorized as non-overlapping (crisp clustering, hard partitioning), overlapping (soft, fuzzy clustering) or, in some cases, multi-resolution (including hierarchical). Multi-resolution clusterings are considered when there is a need to simultaneously compare multiple resolutions (hierarchy levels) of the results against the ground-ground, where each resolution contains non/overlapping clusters as discussed further in Section 4.3.4. Non-overlapping clusterings can be seen as a special case of overlapping clusterings.

A large number of accuracy metrics were proposed in the literature to measure the clustering quality [4, 41, 44, 203, 262, 202]. Evaluation frameworks [35, 211, 268, 153, 140] and surveys [247, 80, 83, 264, 205] were also introduced. Despite the large number of accuracy metrics proposed, very few metrics are applicable to overlapping clusters, causing many issues when evaluating such clusters (e.g., Adjusted Rand Index is used to evaluate overlapping clusters in [159], even though it is applicable to non-overlapping clusters only). Moreover, most of the quality metrics for overlapping clusters are not comparable to similar metrics for non-overlapping clusters (e.g., standard NMI [44] or modularity [174] versus some overlapping NMI [154] or overlapping modularity [274, 175, 129, 36] implementations), which complicates the direct comparison of the respective clustering algorithms.

Therefore, further research is required to develop accuracy metrics that are applicable to overlapping (and multi-resolution) clusterings, that satisfy tight performance constraints and, ideally, that are compatible with the results of standard accuracy metrics used for non-overlapping clustering. Finally, producing a single, easy to interpret value for the final clustering is of importance also, in order to help the user pick the most suitable clustering for a particular use-case and for potentially several accuracy metrics. This issue has been tackled through the formal constraints, introduced for example in [4, 203], and is further discussed in Chapter 2.3.4.

To the best of our knowledge, this is the first work discussing all state-of-the-art accuracy metrics applicable to overlapping clustering evaluation on large datasets (i.e., with more than 10^7 elements). Being able to evaluate metrics on large datasets means—in our context—that the evaluation process should be at most: *a*) *quadratic* in terms of runtime complexity and *b*) *quasilinear* in terms of memory complexity with the number of elements considered. In addition, we also introduce in this chapter a novel indexing technique to reduce the runtime complexity of the Mean F1 score (and a similar metric, NVD [35]) evaluation from $O(N \cdot (|C| +$

$|C'|$) to $O(N)$, where N is the number of elements in the processed clusters, $|C|$ is the number of resulting clusters and $|C'|$ is the number of ground-truth clusters. Finally, we propose extensions to the state-of-the-art accuracy metrics to satisfy more formal constraints (i.e., to improve their effectiveness) without sacrificing their efficiency. Efficient C++ implementations of *a)* all discussed accuracy metrics and *b)* their improved versions are freely available online as open source utilities as listed in the sections devoted to each metric. All our accuracy metrics are also integrated into an open source benchmarking framework for clustering algorithms evaluation, Clubmark¹ [140], besides being available as dedicated applications.

4.2 Related Work

Related efforts can be categorized into three main groups, namely: *a)* accuracy metrics for overlapping clustering evaluation (satisfying the complexity constraints mentioned above), *b)* frameworks providing efficient implementations for accuracy evaluation and *c)* formal constraints for accuracy metrics (discussed in Chapter 2.3.4).

Accuracy Metrics for Overlapping Clusters The *Omega Index* [41] is the first accuracy metric that was proposed for overlapping clustering evaluation. It belongs to the family of *Pair Counting Based Metrics*. It is a fuzzy version of the Adjusted Rand Index (ARI) [88] and is identical to the Fuzzy Rand Index [89]. We describe the Omega Index in Section 4.3.1.

Versions of *Normalized Mutual Information (NMI)* suitable for overlapping clustering evaluation were introduced as Overlapping NMI (ONMI)² [154] and Generalized NMI (GNMI) [54] and belong to the family of *Information Theory Based Metrics*. The authors of ONMI suggested to extend Mutual Information with approximations (introduced in [126]) to find the best matches for each cluster of a pair of overlapping clusterings. This approach allows to compare overlapping clusters, but unlike GNMI we introduce in Section 4.3.3, it yields values that are incompatible with standard NMI [44] results.

The *Average F1 score* is introduced in [262, 196] and a similar metric, NVD, is introduced in [35]. The Average F1 score belongs to the family of *Cluster Matching Based Metrics* and is described in Section 4.3.2.

Accuracy Measurement Frameworks A toolkit³ for the parallel measurement of the quality of non-overlapping clusterings on both distributed and shared memory machines is introduced in [35]. This toolkit performs the evaluation of several accuracy metrics (Average F1 score, NMI, ARI and JI) as well as some intrinsic quality metrics, and provides highly optimized parallel implementations of these metrics in C++ leveraging MPI (the Message Passing Interface) and Pthreads (POSIX Threads). Among its accuracy metrics, only Average F1 score

¹<https://github.com/eXascaleInfolab/clubmark>

²<https://github.com/eXascaleInfolab/OvpNMI>

³<https://github.com/chenmingming/ParallelComMetric>

is applicable to overlapping clusterings.

WebOCD [211] is an open-source RESTful web framework for the development, evaluation and analysis of *overlapping* community detection (clustering) algorithms. It comprises several baseline algorithms, evaluation metrics and preprocessing utilities. However, since *WebOCD* (including all its accuracy metrics) is implemented in pure Java as a monolithic framework, many existing implementations of evaluation metrics cannot be easily integrated into *WebOCD* without either being reimplemented in Java or modifying the framework architecture. A reimplementation of existing metrics is not always possible without a significant performance drop (especially when linking native, high-performance libraries such as Intel TBB, STL or Boost) and time investment.

CoDAR [268] is a framework for community detection algorithm evaluation and recommendation providing user-friendly interfaces and visualizations. Based on this framework, the authors also introduced a study of *non-overlapping* community detection algorithms on *unweighed undirected* networks [247]. Unfortunately, the framework URL provided in the paper refers to a forbidden page, i.e. the implementation is not available to the public anymore.

Circulo [153] is a framework for community detection algorithms evaluation. It executes the algorithms on preliminary uploaded input networks and then evaluates the results using several accuracy metrics and multiple intrinsic metrics.

4.3 Accuracy Metrics for Overlapping Clustering

Accuracy metrics indicate how much one clustering (i.e., set of clusters) is similar to another (ground-truth) clustering. For each presented metric, we first give its original definition before proposing our extensions and optimizations. Then, we empirically evaluate four formal constraints described in Chapter 2.3.4 on samples from [4], which are shown in Figure 4.1, denoting the left clustering as *Low* and the right clustering as *High* for each sample. The constraints that are satisfied are marked in *italics* in the results tables for each discussed metric in the respective subsection. Cluster elements belonging to the same category (ground-truth cluster) in these figures are colored with the same color and texture, while the formed clusters (results) are shown with oval shapes.

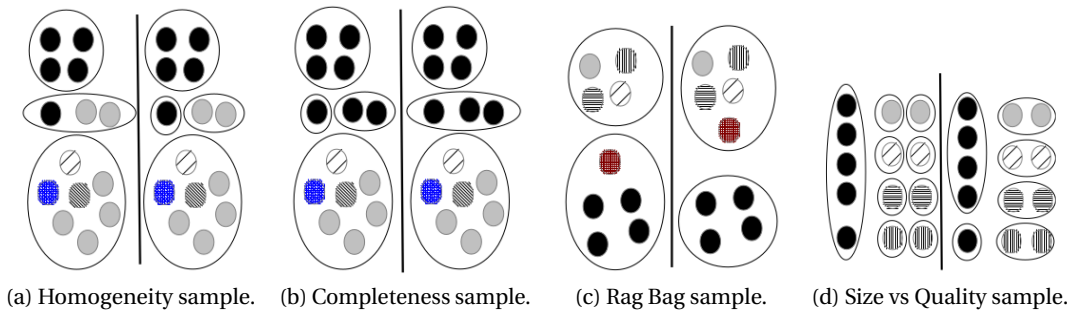


Figure 4.1 – Formal constraints for the accuracy measures.

4.3.1 Omega Index

Preliminaries

The Omega Index [41] is an ARI [88] generalization applicable to overlapping clusters. It is based on counting the number of pairs of elements occurring in exactly the same number of clusters as in the number of categories and adjusted to the expected number of such pairs. Formally, given the ground-truth clustering C' consisting of categories $c'_i \in C'$ and formed clusters $c_i \in C$:

$$\text{Omega}(C', C) = \frac{\text{Obs}(C', C) - \text{Exp}(C', C)}{1 - \text{Exp}(C', C)}. \quad (4.1)$$

The observed agreement is:

$$\text{Obs}(C', C) = \sum_{j=0}^{\min(J', J)} A_j / P, \quad (4.2)$$

where J' (J) is the maximal number of categories (clusters) in which a pair of elements occurred, A_j is the number of pairs of elements occurring in exactly j categories and exactly j clusters, and $P = N \cdot (N - 1) / 2$ is the total number of pairs given a total of N elements (nodes of the network being clustered).

The expected agreement is:

$$\text{Exp}(C', C) = \sum_{j=0}^{\min(J', J)} P'_j P_j / P^2, \quad (4.3)$$

where P'_j (P_j) is the total number of pairs of elements assigned to exactly j categories (clusters).

Proposed Extension (Soft Omega Index)

The Omega Index evaluates overlapping clusterings by counting the number of pairs of elements present in exactly the same number of clusters as in the number of categories, which does not take into account pairs present in slightly different number of clusters. We propose to fix this issue by normalizing smaller number of occurrences of each pair of elements in all clusters of one clustering by the larger number of occurrences in another clustering as outlined on line 9 of Algorithm 4. The input data consists of two clusterings (grs, cls), and the *rels* hashmap relating the clusters to their elements (nodes) for each clustering. The updated computation of the observed agreement of pairs requires also to correct the expected agreement, which is performed on line 19.

Thus, *OmegaSoft* has the same definition as Eq. 4.1, except the observed agreement number is

Chapter 4. Xmeasures: Accuracy Measures for Overlapping and Multi-resolution Clustering

Algorithm 4 Soft Omega Index

```

1: procedure OMEGASOFT(rels, grs, cls)
2:   nobs = 0 ▷ Observed number
3:   ngs = vector(size(grs)) ▷ Ranked pairs counts for grs
4:   ncs = vector(size(cls)) ▷ Ranked pairs counts for cls
5:   for ir in range(begin(rels), end(rels)) do
6:     for jr in range(++ir, end(rels)) do
7:       gn = mutualnum(grs(ir), grs(jr))
8:       cn = mutualnum(cls(ir), cls(jr))
9:       nobs += 1 if gn == cn else min(gn, cn) / max(gn, cn)
10:      ++ngs[gn]; ++ncs[cn]
11:     end for
12:   end for
13:   nexp = 0; szmin = min(size(ngs), size(ncs))
14:   for i in range(szmin) do
15:     nexp += ngs[i] · ncs[i]
16:   end for
17:   rns = ngs if size(ngs) > szmin else ncs
18:   for i in range(szmin, size(rns)) do
19:     nexp += rns[i]
20:   end for
21:   nps = size(nodes(rels)); nps *= (nps - 1) / 2
22:   nexp /= nps
23:   return (nobs - nexp) / (nps - nexp)
24: end procedure

```

evaluated as:

$$Obs_{soft}(C', C) = \sum_{j=0}^{\max(J', J)} Anorm_j / P, \quad (4.4)$$

where $Anorm_j$ is the number of pairs of elements occurring in exactly j' and j clusters of the clusterings and being weighted by $\min(j', j) / \max(j', j)$.

The expected agreement is:

$$Exp_{soft}(C', C) = \left(\sum_{j=0}^{Jmin} P'_j P_j + \sum_{j=Jmin+1}^{\max(J', J)} Prem_j \right) / P^2, \quad (4.5)$$

where $Jmin = \min(J', J)$, $Prem_j = P_j$ if $\min(J', J) = J'$ and $Prem_j = P'_j$ otherwise.

Note that for non-overlapping clusterings (i.e., when the membership in clusters equals to 1 $\implies J' = J = 1$), the Soft Omega Index is equivalent to the original Omega Index, which is equivalent to ARI.

Evaluation and Constraints Matching

A counterexample outlining the issue of the Omega Index when discarding partially matching pairs of elements is shown in Table 4.1, where each of the four categories (C1'-C4') consists of 3 elements and the total number of elements is 4 (#1-#4). The first clustering algorithm has a *Low* accuracy and discovers only two clusters as shown in Table 4.1. The second clustering algorithm (*High*) performs much better discovering all four clusters but not all elements of the respective categories as shown in Table 4.1. The original Omega Index fails to discriminate these cases yielding 0 for both cases, whereas the Soft Omega Index clearly differentiates the more accurate solution.

Table 4.1 – Accuracy evaluation of *Low* and *High* vs *Ground-truth* clusterings by Omega Index and Soft Omega Index.

Metrics \ Clusterings	Ground-truth	
	C1': 1 2 3	
	C2': 2 3 4	
	C3': 3 4 1	
	C4': 4 1 2	
	Low	High
	C1: 1 2	C1: 1 2
	C2: 3 4	C2: 2 3
		C3: 3 4
		C4: 4 1
Omega Index	0	0
Soft Omega Index	0	0.33

The empirical satisfaction of the formal constraints for both versions of Omega Index is given in Table 4.2 and discussed below in Section 4.3.4. The computational complexity $O(N^2)$ and the memory complexity is $O(|C| + |C'|) \approx^4 O(\sqrt{N})$ for both implementations, where N is the number of elements in the clustering. Implementations of both the original and Soft Omega Index are provided in the open source *xmeasures*⁵ utility and are available for free.

Table 4.2 – Formal Constraints for Soft and original Omega Index.

Metrics	Clusterings		Homogen.		Comple.		RagBag		SzQual.	
	low	high	low	high	low	high	low	high	low	high
[Soft] Omega Index	0.247	0.282	0.244	0.311	0.4	0.4	0.804	0.804		

⁴As a rule of thumb, the number of clusters in most real-world networks consisting of N nodes can be estimated as \sqrt{N} [150].

⁵<https://github.com/eXascaleInfolab/xmeasures>

4.3.2 Mean F1 Score

Preliminaries

The *Average F1 score (F1a)* is a commonly used metric to measure the accuracy of clustering algorithms [262, 196, 263]. F1a is defined as the average of the weighted F1 scores [201] of *a)* the *best matching* ground-truth clusters to the formed clusters and *b)* the *best matching* formed clusters to the ground-truth clusters. Formally, given the ground-truth clustering C' consisting of clusters $c'_i \in C'$ (called categories) and clusters $c_i \in C$ formed by the evaluating clustering algorithm:

$$F1a(C', C) = \frac{1}{2}(F_{C',C} + F_{C,C'}), \quad (4.6)$$

where

$$F_{X,Y} = \frac{1}{|X|} \sum_{x_i \in X} F1(x_i, g(x_i, Y)), \quad g(x, Y) = \{\operatorname{argmax}_y F1(x, y) \mid y \in Y\}, \quad (4.7)$$

where $F1(x, y)$ is the F1 score of the respective clusters.

Proposed Extensions (F1h and F1p)

The F1a definition yields *non-indicative* values of $F1a \in [0, 0.5]$ when evaluating a large number of clusters. In particular, for clusters formed by taking all possible combinations of the nodes, $F1a > 0.5$ ($F1_{C',C} = 1$ since for each category there exists the exactly matching cluster, $F1_{C,C'} \rightarrow 0$ since majority of clusters have low similarity to the categories). To address this issue, we suggest to use the harmonic mean instead of the arithmetic mean (average). We introduce the *harmonic F1 score (F1h)* as:

$$F1h(C', C) = \frac{2F_{C',C}F_{C,C'}}{F_{C',C} + F_{C,C'}}. \quad (4.8)$$

$F1h \leq F1a$ since the harmonic mean cannot be larger than the arithmetic mean. In case of F1h, $F_{C',C}$ can be interpreted as recall and $F_{C,C'}$ as precision for the evaluated clustering C and the ground-truth clustering C' .

Intuitively, a more straightforward approach to avoid low non-indicative value, seems to be just the normalization of $F1a$ by the total (or maximal) number of clusters:

$$F1ax(C', C) = \frac{1}{|C'| + |C|} \left(\sum_{x_i \in C'} F1(x_i, g(x_i, C)) + \sum_{x_i \in C} F1(x_i, g(x_i, C')) \right). \quad (4.9)$$

However, this approach would be biased towards small numbers of clusters. For example, if a large dataset has only two ground-truth clusters ($|C'| = 2$) and the clustering strategy intentionally forms only a single cluster ($|C| = 1$) then $F1ax(C', C) \gtrsim 1/3$ ($F1_{C',C} = 1$ since for each category there exists an exactly matching cluster, $F1_{C,C'} \rightarrow 0$ since the formed cluster has

a low similarity to the categories).

$F1h$ is more indicative than $F1a$ but both measures do not satisfy the Homogeneity constraint, as they penalize local best matches too severely. We propose to evaluate the probability of the local best matches rather than the F1 score to address the outlined issue. Our new metric $F1p$ is the harmonic mean (i.e. F1 measure) of the average over each clustering of the best local probabilities for each cluster. $F1p$ corresponds to the expected probability of the best match of the clusterings unlike $F1h$, which corresponds to the expected worst-case of the best match in the clusterings. Formally, $F1p$ is evaluated similarly to $F1h$, except that the local matching function $pprob$ given in Eq. 4.10 replaces $f1$ given in Eq. 4.11.

$$pprob(m, c', c) = m/|c'| * m/|c| = \frac{m^2}{|c'| * |c|}, \quad (4.10)$$

$$f1(m, c', c) = 2 \frac{m/|c'| * m/|c|}{m/|c'| + m/|c|} = 2 \frac{m}{|c'| + |c|}, \quad (4.11)$$

where m is the *contribution* of matched elements between the cluster c and category c' . The notations of contribution and $|x|$ (size of the cluster x) vary for the overlapping and multi-resolution clusterings and are discussed in Section 4.3.4. For multi-resolution and non-overlapping clusterings, $|x|$ is simply equal to the number of elements in the cluster x , and the contribution of each element is equal to 1. For overlapping clusterings, $|x|$ is equal to the total contribution of elements in cluster x , where each element x_i contributes the value $1/\text{shares}(x_i)$ given that x_i is a member of the number $\text{shares}(x_i)$ of (overlapping) clusters.

Optimizations (Efficient Indexing) for the F1 Metric Family

We propose an efficient indexing technique to reduce the computational complexity of computing the *Mean F1 score* metric family ($F1a$, $F1h$ and $F1p$). Our technique is based on dedicated data structures described below. When loading the clusterings, we create a *rels* hashmap relating the clusters to their elements (nodes) for each clustering. Besides the member nodes, our cluster data structure holds also: *a*) an accumulator *ctr* for the matching contributions of the member nodes together with the pointer to the originating cluster from which these contributions are formed, and *b*) the local contributions *cont* for all members nodes of the cluster. This data structure allows to evaluate the metrics using a single pass over all members of all clusters. The content of the *ctr* attribute is reset on line 9 in Algorithm 5 when adding a value of the contribution together with a distinct cluster pointer from the already stored one. The main procedure to evaluate the aforementioned best matches for the clustering *cls* is listed in Algorithm 5 considering *a*) the *fmatch* matching function ($f1$ or $pprob$ given in Eq. 4.10-4.11) parameterized as *prob* argument, and *b*) the overlapping or multi-scale clusters evaluation semantics parameterized as *ovp*.

Chapter 4. Xmeasures: Accuracy Measures for Overlapping and Multi-resolution Clustering

Algorithm 5 Best matches evaluation using efficient indexing technique

```

1: procedure BMATCHES(grels, crels, cls, prob, ovp)
2:   bms = vector()
3:   fmatch = pprob if prob else f1                                ▷ Matching function
4:   for c in cls do                                                ▷ Traverse each cluster
5:     bmt = 0                                                        ▷ Value of the best match
6:     for nd in members(c) do
7:       ndcls = grels[nd]                                         ▷ Node clusters
8:       for cn in ndcls do                                          ▷ Evaluate node clusters
9:         cn.ctr.add(c, 1 / max(size(grels[nd]), size(crels[nd])) if ovp else 1)
10:        mt = fmatch(cn.ctr, cn.cont, c.cont)
11:        if bmt < mt then
12:          bmt = mt
13:        end if
14:      end for
15:    end for
16:    if prob then
17:      bmt =  $\sqrt{bmt}$ 
18:    end if
19:    bms.append(bmt)
20:  end for
21:  return bms
22: end procedure

```

Evaluation and Constraints Matching

Our new indexing technique reduces the computational complexity of Mean F1 Score evaluation from $O(N \cdot (|C| + |C'|))$ [35] $\simeq^4 O(N\sqrt{N})$ to $O(N \cdot s) \simeq O(N)$, where N is the number of elements in the processing clusters and the constant s is the average membership of the elements, which typically is $\in [1, 2)$ for overlapping clusterings. Implementations of all *MF1* metrics (*F1a*, *F1h*, *F1p*) are provided in the open source *xmeasures*⁵ utility and are available for free. Such a drop in computational complexity allows to evaluate large datasets on a single CPU hundreds times faster than what state-of-the-art parallel implementations [35] do on high-performance servers. The effectiveness of *F1h* and *F1p* on several clusterings of large real-world networks from [263] is given in Figure 4.2. The empirical satisfaction of the formal constraints for all *MF1* metrics is given in Table 4.3 and discussed in Section 4.3.4.

Table 4.3 – Formal Constraints for MF1 metric family.

Clusterings Metrics	Homogen.		Comple.		RagBag		SzQual.	
	low	high	low	high	low	high	low	high
F1a	0.646	0.646	0.639	0.663	0.641	0.630	0.795	0.936
F1h	0.646	0.646	0.639	0.660	0.639	0.630	0.795	0.935
F1p	0.665	0.672	0.686	0.703	0.693	0.693	0.819	0.942

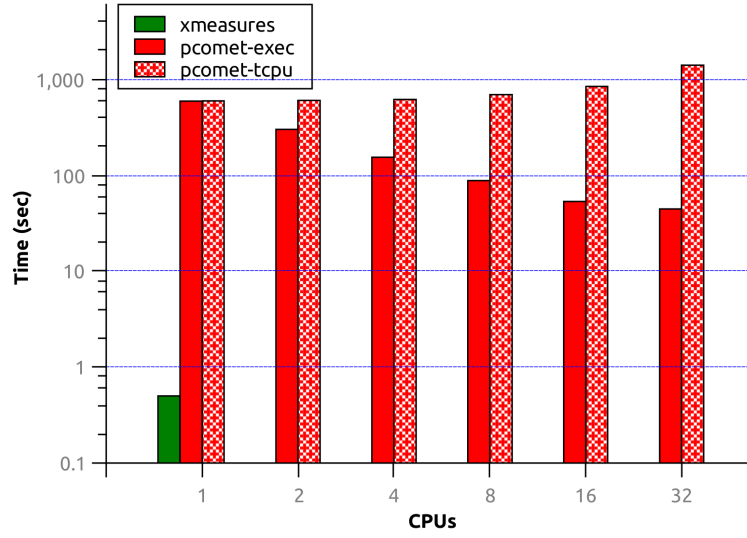


Figure 4.2 – Xmeasures MF1 vs ParallelComMetric F1-Measure. Even on a single CPU, our approach is faster by orders of magnitude compared to parallelized ParallelComMetric.

4.3.3 Generalized NMI

Preliminaries

Generalized NMI (GNMI)⁶ [54] uses a stochastic process to compare overlapping clusterings and feeds the random variables of the process into the standard definition of mutual information (MI) [116]. MI is evaluated by taking all pairs of clusters from the formed and ground-truth clusterings respectively and counts the number of common elements in each pair. Formally, given the ground-truth clustering C' consisting of clusters $c' \in C'$ and the formed clusters $c \in C$, mutual information is defined as:

$$I(C' : C) = \sum_{c' \in C'} \sum_{c \in C} p(c', c) \log_2 \frac{p(c', c)}{p(c')p(c)}, \quad (4.12)$$

where $p(c', c)$ is the normalized number of common elements in the pair of (*category*, *cluster*), $p(c')$ and $p(c)$ is the normalized number of elements in the categories and formed clusters respectively. The normalization is performed using the total number of elements in the clustering, i.e. the number of nodes in the input network.

Normalized Mutual information (NMI) [44] performs normalization of MI by *maximum value*, *arithmetic* or *geometric mean* of the unconditional entropies $H(C')$ and $H(C)$ of the clusterings. Normalization by the maximum value of the entropies is the standard approach, which is also considered as the most discriminative one [54, 154]:

$$NMI(C', C) = \frac{I(C' : C)}{\max(H(C'), H(C))}. \quad (4.13)$$

⁶<https://github.com/eXascaleInfolab/GenConvMI>

Chapter 4. Xmeasures: Accuracy Measures for Overlapping and Multi-resolution Clustering

The unconditional entropy $H(X)$ [212] of clusters $x \in X$ is:

$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x). \quad (4.14)$$

Proposed Extension and Optimizations

The GNMI approach of using a stochastic process provides the only known way to evaluate *standard NMI* for overlapping clusterings. The original GNMI implementation⁶ uses Intel's TBB library (lightweight threads) to execute the stochastic process on all available CPUs efficiently. However, the original implementation has several shortcomings, which makes it inapplicable to large datasets:

- its hard-coded maximal number of stochastic events (successful samples) `EVCOUNT_THRESHOLD` is not adequate for handling both small and large datasets. Moreover, the original value is too small for large datasets;
- its fully random sampling of the cluster elements has a too high computational complexity on large datasets to produce results considering a reasonably small evaluation error (default value is 0.01);
- the two aforementioned points cause significant errors on small datasets and loss of convergence on large datasets while consuming significant computational resources.

We optimized and extended the original version addressing these issues in the following ways. First, instead of the hard-coded `EVCOUNT_THRESHOLD`, we dynamically evaluate the maximal number of stochastic events as:

$$evsmax = \max(\min(mbs', mbs), \frac{1}{rerr\sqrt{rrisk}}), \quad (4.15)$$

where mbs' and mbs are the total membership of the elements in the categories and clusters respectively, $rerr$ is the admissible error and $rrisk$ is the complement of the resulting confidence; $rerr$ and $rrisk$ are specified as input arguments with a default value equal to 0.01.

Second, we extended the original `try_get_sample` procedure with the weighed adaptive sampling given in Algorithm 6. The original version randomly takes the first node (cluster element) among all nodes in the clusterings as shown on line 2 and then applies *mixer* until it returns `false` or all nodes are traversed in a randomized order. We traverse the nodes located in the same cluster of the formed (*c*) or ground-truth (*g*) clusterings on line 7 and weight the shared nodes inversely to their membership. The weighting is performed to discount the contribution (importance) of frequent nodes compared to rare nodes since we traverse only a fraction of all nodes and they may have varying membership ≥ 1 . Indexes on the matched ground-truth and formed clusters are stored in the *mixer* and their matching probability is returned explicitly as *importance*.

Algorithm 6 Adaptive sampling of elements in GNMI

```

1: procedure TRY_GET_SAMPLE(nodes, rrisk, mixer)
2:   node = nodes[rand(size(nodes))]
3:   g, c = clsPair(node)
4:   attempts = (size(g) + size(c)) / (2 · rrisk)
5:   importance = 1 / max( $\sqrt{\text{size}(\mathbf{g}) \cdot \text{size}(\mathbf{c})}$ , 1)
6:   adone = 1
7:   while mixer.apply(g, c) and ++adone ≤ attempts do
8:     cm = rand(g, c)
9:     ndm = cm[rand(size(cm))]
10:    g, c = clsPair(ndm)
11:    importance += 1 / max( $\sqrt{\text{size}(\mathbf{g}) \cdot \text{size}(\mathbf{c})}$ , 1)
12:  end while
13:  return importance / adone
14: end procedure

```

In addition, we performed some technical optimizations to reduce the number of memory allocations and copies, calls to external functions, etc. Our *extended implementation of GNMI*⁷ is open source and available for free.

Evaluation and Constraints Matching

The empirical satisfaction of the formal constraints for *NMI*, the original *GNMI_{orig}* and our *GNMI* implementations is given in Table 4.4 and discussed below in Section 4.3.4. Since GNMI-s yields stochastic results, we report the median value over 5 runs with the same default values of the error and risk arguments. As the table clearly shows, our GNMI implementation is much more accurate than the original one and yields values equal to the original NMI within the specified admissible error (0.01) on non-overlapping clusterings. The empirical evaluation on *a*) the synthetic datasets formed using the LFR⁸ [123] framework and *b*) the large real-world networks with ground-truth⁹ introduced in [263] show that our implementation is one order of magnitude faster on datasets with 10⁴ nodes and two orders of magnitude faster on datasets

Table 4.4 – Formal Constraints for NMI, original GNMI and our GNMI.

Clusterings Metrics	Homogen.		Comple.		RagBag		SzQual.	
	low	high	low	high	low	high	low	high
NMI	0.450	0.555	0.546	0.546	0.434	0.434	0.781	0.888
GNMI _{orig}	0.512	0.598	0.572	0.632	0.417	0.397	0.808	0.877
GNMI	0.448	0.557	0.546	0.547	0.434	0.436	0.781	0.888

⁷<https://github.com/eXascaleInfolab/GenConvNMI>⁸https://github.com/eXascaleInfolab/LFR-Benchmark_UndirWeightOvp⁹<https://snap.stanford.edu/data/#communities>

Chapter 4. Xmeasures: Accuracy Measures for Overlapping and Multi-resolution Clustering

with 10^6 nodes than the original GNMI implementation. The actual computational complexity of both GNMI implementations depends on the structure of the clusters and on the number of overlaps in the clusterings. Moreover, for very dissimilar clusterings the evaluation might not converge at all, which is a disadvantage of the stochastic evaluation. The worst case computational complexity for our GNMI implementation is $O(N \cdot s \cdot |\bar{c}| \cdot |C|) \approx O(N \cdot |\bar{c}| \cdot |C|)$ while it is $O(\text{EVCOUNT_THRESHOLD} \cdot N \cdot |C|)$ for the original GNMI, where N is the number of elements in the evaluating clusterings, $|C|$ is the number of clusters and $|\bar{c}|$ is the average size of the clusters.

4.3.4 Discussion

When evaluating clusterings, it is important to *a)* distinguish overlapping from multi-resolution (including hierarchical) clusterings and *b)* consider the limitations of the applied accuracy metric in order to produce meaningful results. First, we discuss the differences when handling overlapping clusterings versus multi-resolution clusterings having non-overlapping clusters on each resolution and how they affect accuracy evaluations.

In the case of overlapping clusterings, a node x_i can be shared between s clusters and has equal membership in each of them (e.g., a person may spend time for several distinct hobbies but the more hobbies are involved the less time the person devotes to each one). Thus, the membership contribution x_i to each of the clusters is $1/s$. In case of non-overlapping, multi-resolution clusterings, a node x_i may be a member of s nested clusters taken at different resolutions (granularities), but here x_i fully belongs to each of these clusters having a membership contribution equal to 1 (e.g., a student tacking a course can be a full member of the course, as well as of the department offering the course and of the whole university). These distinct semantics for the elements shared in a clustering are represented by the *ovp* argument in our *xmeasures*⁵ utility for all MF1 metrics.

We summarize the advantages and limitations of the various metrics discussed in this chapter as follows:

Omega Index pros: its values are not affected by the number of clusters (unlike NMI) and have an intuitive interpretation (0 means equal quality to a random clustering).

Omega Index cons: it performs purely when applied to multi-resolution clusterings and has the highest computational complexity among all considered metrics ($O(N^2)$).

MF1 pros: it has the lowest computational complexity (linear when using our indexing technique).

MF1 cons: it evaluates the best-matching clusters only, which gives an unfair advantage to the clusterings with larger numbers of clusters (which is partially addressed by the application of the harmonic mean in *F1h* and *F1p* instead of the average).

GNMI pros: it parallelizes very well and inherits NMI's pros, namely it evaluates the full matching of all clusters (unlike MF1). Also, it is well-grounded formally in Information Theory.

GNMI cons: the convergence in the stochastic implementation is not guaranteed (though

loss of convergence typically indicates a relevance close to zero); the stochastic process yields non-deterministic results and is computationally heavy; its execution time is hard to estimate. In addition, GNMI inherits the NMI's cons, namely the results depends on the number of clusters being evaluated and increase up to ≈ 0.3 for large numbers of clusters.

Empirical evaluation of the formal constraints satisfaction for all discussed metrics on the original intuitive samples from [4] is given in Table 4.5. As it shown in the table, none of the metrics satisfies the RagBag constraint, which is essential in practice when evaluating the multi-resolution or hierarchical clusterings since the more fine-grained (higher resolutions) are expected to have less noisy structure. Otherwise, the proposed $F1p$ metric performs the best according to the empirical satisfaction of the formal constraints having the lowest computational complexity (linear on the number of elements being clustered) compared to GNMI and Omega Index.

Table 4.5 – Formal Constraints for Omega Index, MF1 and GNMI.

Metrics\Clusterings	Homogen.	Comple.	RagBag	SzQual.
[Soft] Omega Index	+	+		
F1h		+		+
F1p	+	+		+
GNMI	+			+

4.4 Conclusions

In this chapter, we discussed the state-of-the-art accuracy metrics applicable to overlapping clustering evaluations on large datasets and introduced several optimizations and extensions of the discussed metrics. In particular, we introduced an efficient indexing technique to speedup the evaluation of Mean F1 score from $O(N\sqrt{N})$ to $O(N)$, where N is the number of elements in the clustering. We proposed an adaptive sampling strategy for *GNMI*, which not only speeds up the evaluation by orders of magnitude but also improves the precision of the metric. In addition, we proposed two extensions of the Average F1 score ($F1a$), namely $F1h$ and $F1p$. $F1h$ addresses the issues of the loss of indicativity of $F1a$ in the range $[0, 0.5]$ while $F1p$ empirically improves the satisfaction of the formal constraints we consider without sacrificing efficiency. We also proposed an extension of the Omega Index called *Soft Omega Index*, which is equivalent to the original Omega Index evaluating non-overlapping clusterings and yields more discriminative results for overlapping clusterings due to the fact that it considers partial matches of pairs of elements.

Besides the proposed optimizations and extensions of the accuracy metrics, we discussed formal constraints for each metric, as well as their applicability and limitations for specific cases in overlapping and multi-resolutions clusterings. Our analysis of the metrics should provide insight for their future usage and should hopefully help identify the best performing

Chapter 4. Xmeasures: Accuracy Measures for Overlapping and Multi-resolution Clustering

clustering algorithm for particular user's needs or tasks.

We freely provide implementations of all discussed metrics, and are also integrating them into an open source benchmarking framework for clustering algorithms evaluation, Clubmark¹.

5 Clubmark: Benchmarking of Stable, Overlapping and Multi-resolution Clustering

In this chapter, we investigate the benchmarking process of diverse clustering algorithms on large datasets and propose our novel benchmarking framework, Clubmark. Our framework takes advantage of modern NUMA architectures for the efficient and fair (i.e., avoiding various biases) evaluation of resource-intensive and long term running clustering algorithms (including DAOC presented in Chapter 3).

5.1 Introduction

Clustering is a key component of many data mining systems with many applications encompassing statistical analysis and the exploration of physical, social, biological and informational systems as it outlined in Chapter sec:bg:clsgran. A wide variety of graph algorithms have been proposed in the literature aiming to improve the efficiency and/or accuracy of the clustering. An extensive evaluation of these algorithms typically includes both real-world graphs with a ground truth as well as synthetic networks of varying parameters. Moreover, the evaluation on synthetic networks should fulfill the following desiderata:

- the evaluation should be performed on various types of synthetic networks, i.e., generating networks with diverse parameters is required to check for bias in the clustering algorithm;
- the evaluation should take into account multiple instances of each type of synthetic network to avoid occasional bias due to particular structures in the network;
- the evaluation should consider multiple shuffles of a given network instance to avoid bias toward a particular ordering of the input network.

The consideration of the aforementioned requirements can increase the number of input networks by orders of magnitude (i.e., by $network_types * instances * shuffles$), which is

Chapter 5. Clubmark: Benchmarking of Stable, Overlapping and Multi-resolution Clustering

hardly practical when using sequential execution frameworks even when considering a single clustering algorithm. In addition, parallel executions of the algorithm on multiple networks having diverse structures may affect the benchmarking results. In particular, *a)* the growing memory consumption of parallel processes may utilize almost all the available physical memory and cause swapping, which significantly affects execution time; *b)* execution of cache-intensive processes may result in conflicting CPU cache evictions and increasing page faults, negatively affecting the execution time; *c)* a bug in one algorithm or a high computational complexity on a particular network may result in interminable process executions.

Besides the practical considerations described above, there are also a number of important theoretical aspects that are missing in most existing frameworks:

- hierarchical and multi-level algorithms may produce various numbers of output levels (resolutions), whose fair evaluation is not straightforward since an algorithm with a larger number of output levels is more likely to score high on effectiveness metrics on one of its levels;
- very few extrinsic quality measures are applicable to overlapping clusters, which causes some frameworks to apply improper measures (e.g. ARI is used for overlaps in [159]) or let end-users apply improper measures;
- most of the quality measures for overlapping clusters are not comparable to similar measures for non-overlapping clusters (e.g. standard NMI [44] or modularity [174] VS some overlapping NMI [154] or overlapping modularity [274, 175, 129] implementations), which prevents direct comparison of the respective clustering algorithms.

Finally, besides comparing to the state of the art, a benchmarking framework could be extremely useful for the online and iterative development of new clustering algorithms given interactive profiling capabilities. To the best of our knowledge Clubmark is the first benchmarking framework that addresses all of the aforementioned issues and provides a unified and fully automatic solution for the comprehensive benchmarking and profiling of diverse clustering algorithms on a wide variety of synthetic and real-world networks.

5.2 Related Work

WebOCD [211] is an open-source RESTful web framework for the development, evaluation and analysis of *overlapping* community detection (clustering) algorithms. It comprises several baseline algorithms, evaluation metrics and input data pre-processing utilities for the fast development of new clustering algorithms inside the framework. However, WebOCD being implemented in pure Java, is designed to execute and evaluate algorithms implemented solely in Java with specific interfaces tightly integrated into the framework. Moreover, the existent implementations of evaluation metrics can not be easily integrated into WebOCD without

being reimplemented in Java, which is not always possible without a significant performance drop and time loss.

CoDAR [268] is a framework for community detection algorithm evaluation and recommendation providing a user-friendly interface and visualizations. The framework monitors the real-time structural changes of the network during the clustering process, adopts multiple metrics and builds a rating model for algorithm performance evaluation. Based on this framework, the authors also introduced a study of *non-overlapping* community detection algorithms on *unweighed undirected* networks [247]. The evaluated algorithms are reimplemented in a common code base inside the framework, which is convenient for the uniform evaluation but limits the applicability of the framework to the existing algorithms. Unfortunately, the framework URL provided in the paper refers to a forbidden page, i.e. the implementation is not available to the public anymore.

LDBC Graphalytics [92] is a benchmark for large-scale graph analysis platforms such as Giraph and GraphX. It comprises several parallel algorithms, standard datasets, synthetic dataset generators, reference output and evaluation of various metrics to quantify multiple kinds of system scalability and performance variability. This benchmark provides comprehensive evaluations of graph analysis platforms on various algorithms and datasets rather than an evaluation of the algorithms themselves (i.e., evaluating the accuracy of the algorithms themselves is outside the scope of this platform).

Several frameworks and toolkits have been presented to measure the quality of the clustering, which can not be qualified as full-fledged benchmarking frameworks but are related to benchmarking. *Circulo* [153] is a framework for community detection algorithms evaluation. It executes the algorithms on preliminary uploaded input networks and then evaluates the results with multiple intrinsic and a few extrinsic measures. The framework executes the algorithms on the input datasets in parallel; however, the execution is performed without any isolation and no measures are taken to prevent mutual impact of the running processes. For example, if one of the processes consumes most of the available physical memory, others will be swapped by the operating system affecting the execution time. Multi-treaded processes in *Circulo* also affect the execution time of the remaining running algorithms by occupying the shared computational resources, which is unacceptable for a fair benchmarking. A *toolkit for the parallel measurement* of quality in non-overlapping clusters on both distributed and shared memory machines is presented in [35]. This toolkit performs exclusively the evaluation of several intrinsic and extrinsic quality measures without executing the clustering algorithms themselves. The evaluation of multiple algorithms using various measures is performed in several surveys and studies on clustering algorithms [80, 263, 83]. However, the corresponding evaluation frameworks have not been publicly shared.

5.3 System Description

Clubmark¹ is an industrial-grade benchmarking framework for parallel isolation benchmarking of clustering algorithms. It can be applied on a wide variety of real-world and synthetic networks, and evaluates both the efficiency and the effectiveness of the algorithms. This framework is implemented in Python to be cross-platform and easily extensible, and is available as a free and open source package. Clubmark has a modular architecture with pluggable external clustering algorithms and utilities for the evaluation and data pre/post-processing. Additionally, we provide a ready-to-use, containerized execution environment² for benchmarking with pre-installed dependencies for all algorithms and utilities since most of them are implemented in compilable languages (C++ and C) with dependencies on external libraries (Boost, Intel TBB, etc.). The containerization is performed on Docker for Linux Ubuntu 16.04 LTS x64. The overall Clubmark architecture is depicted in Figure 5.1.

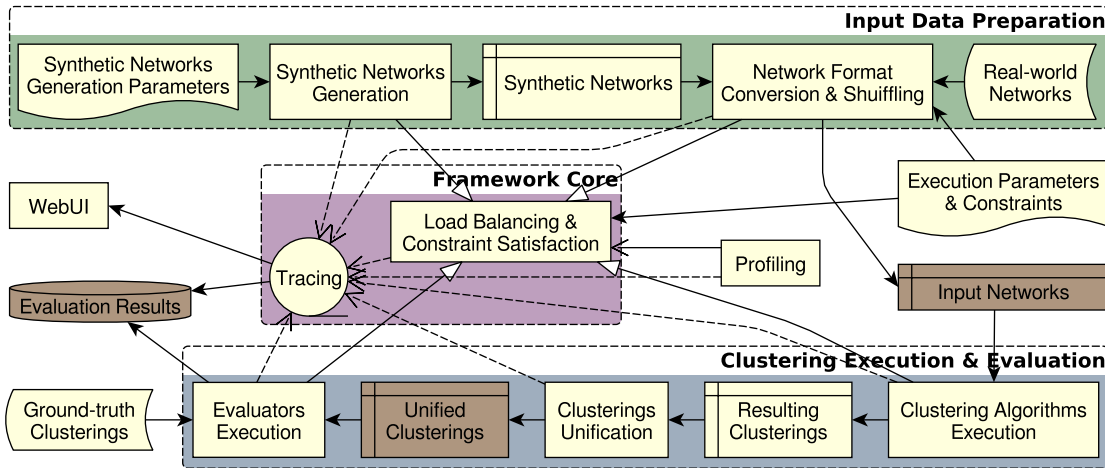


Figure 5.1 – Clubmark architecture.

5.3.1 Framework Core

The framework is based on the PyExPool³ multi process execution pool with a constraint-aware load balancer, which isolates each executing process. PyExPool provides a number of primitives like Job, Task and ExecPool. Each application (clustering algorithms, evaluation utilities, etc.) is scheduled for execution as a *Job* instance and is transparently started under an external tiny profiler, exectime⁴, to trace resource consumption (execution time, processing time, peak RAM consumption) and return code. A Job instance includes the execution arguments, a descriptor of the executing process, a symbolic name, an optional timeout, and provides the optional capability to restart the process on timeout. Jobs can be wrapped

¹<https://github.com/eXascaleInfolab/clubmark>

²<https://hub.docker.com/r/luaxi/clubmark-env/>

³<https://github.com/eXascaleInfolab/PyExPool>

⁴<https://bitbucket.org/lumais/exectime/>

into a hierarchy of *Tasks* for the intuitive management of related jobs. For example, a task for executing a clustering algorithm on a specific type of synthetic networks may include subtasks with several instances of this network type, where each instance may include jobs with network shuffles (randomly reordered nodes and links) of the instance. Each Task and Job provide callbacks for the start and finish events, which can be used for the pre/post-processing activities such as notification of external services about a process crash.

All jobs are scheduled and executed by the *ExecPool*. The execution pool performs load balancing to adjust the number of workers (executing processes, i.e. running jobs) in a way as to maximize the utilization of CPU and memory resources as much as possible within the specified constraints. ExecPool optionally provides isolation of the executing processes on the processing units according to the specified policy. Portable Hardware Locality utility (hwloc)⁵ is used to identify the hierarchical topology of the underlying NUMA system architecture to provide several policies for the maximization of the dedicated CPU L1/L2/L3 cache (process execution on the physical CPU core / node) vs parallelization (execution on a logical CPU, i.e. hardware thread). By default, *a*) each clustering algorithm is executed on the dedicated physical CPU core to maximize CPU L1 cache usage and to provide equal computational resources for all algorithms *b*) each single-threaded evaluation utility is executed on the dedicated logical CPU to maximize parallelization *c*) each instance of the multi-threaded evaluation utility (gecmi) is executed on the dedicated CPU node to maximize L1/L2/L3 cache usage and thread parallelization. The number of worker processes in the pool is defined dynamically to satisfy *a*) the isolation policy and *b*) considering the available hardware resources to prevent system swapping while executing the processes. For example, if the isolation policy allows n worker processes but $k \leq n$ workers consume more than the automatically defined low memory condition⁶. Then, the shortest running task among the heaviest workers (in terms of memory consumption) is killed and postponed (if $k \geq 2$) as shown in Figure 5.2. The execution pool also takes care of cleaning tables for the terminating processes to avoid zombies and catches, logs and handles all exceptions occurred during the execution (including system signals), handles global timeouts and feeds the WebUI component.

5.3.2 Input Data

Clubmark includes the LFR benchmark [126] for undirected weighted synthetic networks generation with ground-truth overlapping clusters and a script to download real-world networks with ground-truth communities from SNAP⁷. Clubmark includes default parameters to generate a wide range of diverse synthetic networks with varying numbers of nodes, density of links and mixing parameters for nodes membership in overlapping clusters. The input datasets are represented in a widely used `ncol` format. Additionally, an accessory PyNetCon-

⁵<https://www.open-mpi.org/projects/hwloc/>

⁶The low memory condition is defined to be a bit larger than the amount triggering a system swap, `vm.swappiness` is set to 5 by default.

⁷<https://snap.stanford.edu/data/#communities>

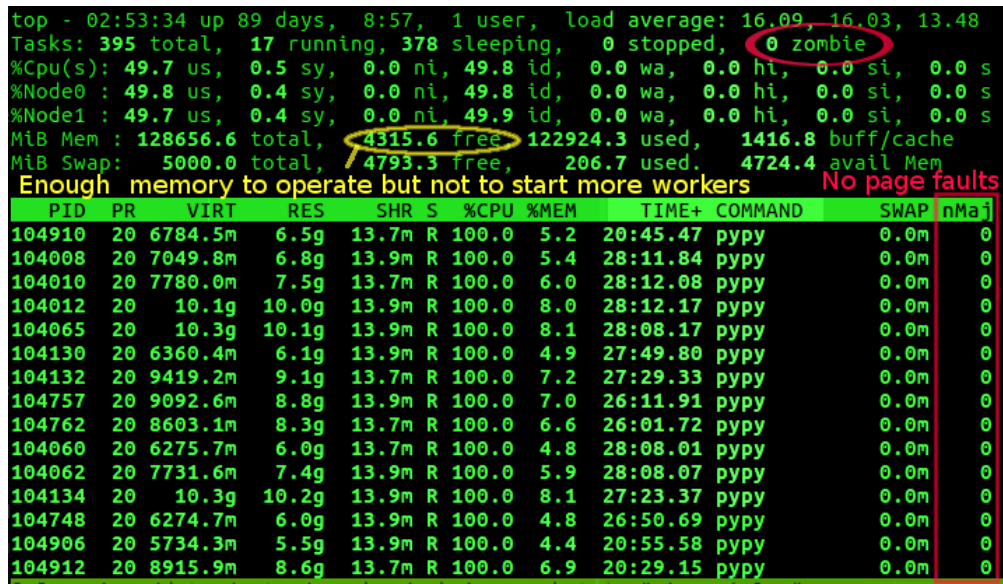


Figure 5.2 – Top utility listing the executing processes during benchmarking.

vert⁸ utility is included to convert custom input networks from pajek and metis into the ns1⁹ format (generalization of ncol). A user may include any additional un/weighted, un/directed networks with/out ground-truth for the benchmarking from any custom location by simply specifying them in the arguments on the benchmark execution. Optionally, the specified number of shuffles is produced from each input network by reordering nodes and links to avoid bias of the algorithms toward the input order of the data.

5.3.3 Clustering Algorithms

Clustering algorithms can be classified by their input data type, i.e. operating on *a*) attributed graphs or *b*) networks (graphs) specified by pairwise relations. These two types of inputs cannot be unambiguously converted into each other. Thus, the respective two types of clustering algorithms cannot be executed on the same input data and, hence, are not (directly) comparable. Clubmark includes a dozen of diverse clustering algorithms processing networks specified by pairwise relations. The included algorithms are listed in Table 5.1: SCP¹⁰ [119], Louvain¹¹ [17], Oslom2¹² [124], GANXiS¹³ (also known as SLPA) [255],

⁸<https://github.com/eXascaleInfolab/PyNetConvert>

⁹<https://github.com/eXascaleInfolab/clubmark/blob/master/formats>

¹⁰<http://www.lce.hut.fi/research/mm/complex/software/>

¹¹<http://igraph.org/c/doc/igraph-Community.html>

¹²<http://www.oslom.org/software.htm>

¹³<https://sites.google.com/site/communitydetectionslpa/>

pSCAN¹⁴ [31], CGGC[i]_RG¹⁵ [180] and SCD¹⁶ [196]. We also included the *Randcommuns*¹⁷ algorithm, which takes a number of clusters and their sizes from the ground-truth and randomly fills each formed template with connected nodes fetched from the input network. Each node is fetched only once, so some templates might end up empty and become omitted if the ground-truth clusters contain overlaps. Clusters formed by the Randcommuns represent a useful baseline for all algorithms providing interpretable and intuitive values of each evaluation measure. Additional algorithms can be added just by wrapping the call of the respective application into the `execAlgName` Python function located in the `benchapps.py` module.

Table 5.1 – Clustering algorithms included in Clubmark.

Features \ Algorithms	DA	D	SP	L	O	G	P	C	CI	SD	R
Hierarchical	+	+		+	+						
Multi-scale	+	+	+	+	+	+					
Deterministic	+	+	+				+				
Overlapping clusters	+	+	+		+	+	+				
Weighted links	+	+	+	+	+	+					+
Parameter-free	+!	+!		+	*	*		*	*	*	+
Consensus/Ensemble	+	+			+			+	+		

Algorithms: DA(DaocA), D(Daoc), SCP(SP), L(Louvain), O(Oslom2), G(GANXiS), P(pSCAN), C(CGGC_RG), CI(CGGCi_RG), SD(SCD), R(Randcommuns);

Deterministic means here that the algorithm is deterministic and input-order invariant;

+! the feature is available and does not require any tuning, still the ability to force a manual value is provided;

* the feature is parameterized and the default value is available, however a tuning might be required to obtain good results for the given network.

5.3.4 Web Interface

Clubmark comprises a RESTful web interface (*WebUI*) for the interactive profiling of the clustering algorithms and monitoring of the system resources. The WebUI provides three endpoints, each of them showing a common summary on the system resources and execution state, and also provides the following endpoint-specific information: *a) /failures* lists hierarchies of the failed tasks with their jobs *b) /jobs* lists all executing and scheduled jobs *c) /tasks* lists hierarchies of the executing and scheduled tasks with their jobs. Each endpoint has an API for queries described at `/api/info` and provides features such as *a) queries* filtering tasks and jobs using multiple fields and supporting range and optional values *b) snapshots* and live continuous listing of the execution state *c) adjustment* of the displayed columns for tasks and jobs *d) selection* of the output format (`html` or `json` for integration with other services). The WebUI is built using pure HTML/CSS without any JavaScript to provide identical

¹⁴<https://github.com/eXascaleInfolab/pSCAN>

¹⁵<https://github.com/eXascaleInfolab/CGGC>

¹⁶<https://github.com/eXascaleInfolab/SCD>

¹⁷<https://github.com/eXascaleInfolab/clubmark/blob/master/algorithms/>

Chapter 5. Clubmark: Benchmarking of Stable, Overlapping and Multi-resolution Clustering

appearance and functionality for both the graphical and the terminal web browsers as shown in Figure 5.3-5.4.

clubmark webui

Failures | Jobs | Tasks | API Manual

Summary

RSS RAM usage: 68.53% (84.32 / 123.039 GB)

CPU loading: 51.30% (32 lcpus, 16 cores, 2 nodes)

Workers: 16 / 16

Failed | Remained | Completed Jobs: 0 | 197 | 75

Failed Root Tasks: 0 / 8

Failed Tasks: 0 / 31

Remained Tasks with Jobs

name	tstart	tstop	numadded	numdone	numterm	task	duration		
DaocX	529967.486	-	4	1	0	-	166271.911		
DaocX-com-dblp	529967.649	-	6	0	0	DaocX	166271.748		
DaocX/com-dblp%5	-	-	-	-	-	0.000	1		
DaocX-com-lj	529967.603	-	6	3	0	DaocX	166271.793		
DaocX/com-lj%3	-	-	-	-	-	0.000	1		
DaocX/com-lj%1	-	-	-	-	-	0.000	1		
DaocX/com-lj%5	-	-	-	-	-	0.000	1		
DaocAR	529967.489	-	4	1	0	-	166271.908		
DaocAR-com-dblp	529967.651	-	6	0	0	DaocAR	166271.746		
DaocAR/com-dblp%5	-	-	-	-	-	0.000	1		
DaocAR-com-lj	529967.605	-	6	0	0	DaocAR	166271.789		
DaocAR/com-lj%4	55209	-	665336.295	-	4.702	1	DaocAR-com-lj	DaocAR	30903.100
DaocAR/com-lj	48624	-	544685.350	-	4.865	1	DaocAR-com-lj	DaocAR	151554.044
DaocAR/com-lj%2	47869	-	531048.072	-	4.831	1	DaocAR-com-lj	DaocAR	165191.323
DaocAR/com-lj%3	-	-	-	-	0.000	1	DaocAR-com-lj	DaocAR	-
DaocAR/com-lj%1	-	-	-	-	0.000	1	DaocAR-com-lj	DaocAR	-
DaocAR/com-lj%5	-	-	-	-	0.000	1	DaocAR-com-lj	DaocAR	-
DaocA	529967.486	-	4	1	0	-	166271.911		

Figure 5.3 – WebUI queried from the Firefox browser as `http://host/tasks`.

name	pid	code	tstart	tstop	memsize	memkind	task	category	duration
DaocAR/com-lj%4	55209	-	665336.295	-	4.702	1	DaocAR-com-lj	DaocAR	31530.841
DaocAR/com-lj	48624	-	544685.350	-	4.865	1	DaocAR-com-lj	DaocAR	152181.785
Daoc/com-lj	48064	-	533579.601	-	4.830	1	Daoc-com-lj	Daoc	163287.535
DaocA/com-lj%4	55126	-	663557.118	-	4.718	1	DaocA-com-lj	DaocA	33310.018
Randcomms/com-lj	54831	-	657822.718	-	7.031	1	Randcomms-com-lj	Randcomms	39044.418
DaocAR/com-lj%2	47869	-	531048.072	-	4.831	1	DaocAR-com-lj	DaocAR	165819.064
Randcomms/com-lj%4	55260	-	666130.694	-	7.111	1	Randcomms-com-lj	Randcomms	30736.442
Daoc/com-lj%3	55344	-	667651.183	-	4.715	1	Daoc-com-lj	Daoc	29215.953
Daoc/com-lj%2	47835	-	530514.679	-	4.879	1	Daoc-com-lj	Daoc	166352.458
DaocR/com-lj%2	47839	-	530547.793	-	4.894	1	DaocR-com-lj	DaocR	166319.344
Daoc/com-lj%4	54836	-	657940.324	-	4.713	1	Daoc-com-lj	Daoc	38926.813
DaocA/com-lj%2	47854	-	530696.153	-	4.797	1	DaocA-com-lj	DaocA	166170.983
DaocA/com-lj	48304	-	539316.774	-	4.871	1	DaocA-com-lj	DaocA	157550.362
DaocR/com-lj%4	55062	-	662582.817	-	4.721	1	DaocR-com-lj	DaocR	34364.320
DaocR/com-lj	48188	-	535996.379	-	4.883	1	DaocR-com-lj	DaocR	166870.758
Randcomms/com-lj%2	47998	-	532699.701	-	7.121	1	Randcomms-com-lj	Randcomms	164167.436

Figure 5.4 – WebUI queried from the w3m console browser using a simple filtering query: `$ w3m http://host/jobs?flt=tstart`.

5.3.5 Evaluation Measures

Clubmark evaluates both the efficiency and the effectiveness of the clustering algorithms. The following *efficiency measures* are evaluated for each algorithm: *a)* peak consumption of resident memory (RAM) *b)* execution time *c)* processing time (total amount of time spent by the algorithm on each processing unit). The *effectiveness measures* comprise most common intrinsic and extrinsic clustering quality measures. We include only the *unified* measures suitable for both overlapping and non-overlapping clusters evaluation and having a reasonably low complexity for large datasets evaluation, that is: *a)* having at most near linear complexity on the number of links and *b)* having at most square complexity on the number of nodes. Providing unified measures only prevents the improper use of measures by the end-user, facilitating a fair and direct comparison of diverse clustering algorithms.

The provided *intrinsic measures* include conductance [98] and modularity [174]. As the standard modularity measure is not directly applicable to overlaps evaluation, we extend it to overlapping cases using a method for the virtual decomposition of overlaps as described in Chapter 3.3.2. Our decomposition technique retains the total weight and structure of the network yielding values equal to the standard modularity when the overlap is not present. The latter provides a fair comparison of both overlapping and non-overlapping clusters even if the non-overlapping clusters are evaluated with the standard modularity being reported in other papers. The provided *extrinsic measures* include all known extrinsic quality measures for overlaps (fuzzy partitions) [75] satisfying our complexity requirements. In particular, Omega Index¹⁸ [41] (which is a fuzzy version of the Adjusted Rand Index [88] and is identical to the Fuzzy Rand Index [89]), an NMI version for overlaps [54] compatible with standard NMI for disjoint clustering and harmonic mean of F1-Score (F1h)¹⁸. The average of F1-Score (F1a) is a commonly used evaluation measure of clustering accuracy [262, 196] but the resulting values lower than 0.5 are non-indicative since the artificial clusters formed from all permutations of the input nodes yield $F1a \rightarrow 0.5$. To make all resulting values indicative we use F1h presented in Chapter 4.3.2. Also, we extended the original implementation of NMI for overlaps¹⁹ with adaptive sampling and specific optimizations to speed up its execution, in order to apply it on large datasets. Additional quality measures can be added by wrapping the call of the respective application into the `execMeasureName` Python function located in the `benchevals.py` module.

5.3.6 Evaluation & Results

The uniform evaluation and fair comparison of the resulting clusterings requires some post-processing to unify their structures. In particular, multi-level and hierarchical algorithms returning multiple levels of clusters. The more levels the algorithm produces, the more likely it is to have a higher evaluation. Therefore, we unify the expected number of output levels to a

¹⁸<https://github.com/eXascaleInfolab/xmeasures>

¹⁹<https://github.com/eXascaleInfolab/GenConvNMI>

Chapter 5. Clubmark: Benchmarking of Stable, Overlapping and Multi-resolution Clustering

fixed parameter L (10 by default). If the produced number of levels is larger than L , then the original results are moved to another repository (-orig/) and symbolic links are created to the L output levels sampled uniformly from those results. Algorithms producing a single output level typically have a resolution parameter (density, clique rank, etc.), which can optionally be leveraged to produce L output clusterings.

Thus, each clustering algorithm produces up to L clusterings for each shuffle of each network instance of each network type forming the following structure of output directories:

```
<algnam>/  
  <nettype>[~<instance>]/  
    <shuffle>/
```

with up to L clusterings in the <shuffle>/. The *ground-truth* is specified per network instance. The quality evaluation of the resulting clusterings is performed by the measures specified by the user and saved into an HDF5 hierarchical structure. Afterwards, the aggregated final results are computed for each measure: 1) the average value and variance are calculated for all shuffles of each instance and 2) these results are averaged for all instances of each network type. The efficiency evaluations reported by the `exectime` for each clustering algorithm execution on each shuffle are aggregated similarly to the quality evaluations.

5.4 Discussion and Conclusions

Clubmark benchmarking framework is open sources and is available for free for both non-commercial and commercial purposes from <https://github.com/eXascaleInfolab/clubmark>. Clubmark is designed to facilitate its users in: *a)* running comprehensive evaluations of emerging clustering algorithms, and *b)* speeding up their development thanks to fast and convenient profiling tools on diverse input networks.

Clubmark Deployment. Clubmark is designed for flexible deployment on cloud-hosted servers, both: *a)* directly on the host and *b)* using a containerized environment from the Docker hub.

Benchmarking Execution. A typical execution of our benchmarking is performed for the clustering algorithms selected by the user on both synthetic and a predefined set of real-world networks. The synthetic networks can be generated using default framework configurations. The benchmarking framework allows: *a)* to apply optional pre-processing steps (i.e. synthetic networks generation and shuffling) *b)* to specify execution constraints for the benchmarking *c)* to produce a uniform input from distinct datasets, considering the availability of distinct instances and shuffles for each network *d)* to execute diverse clustering algorithms, storing their results *e)* to unify the results of the diverse clustering algorithms for the subsequent evaluation *f)* to perform the parallel evaluation considering both single and multi-threaded applications *g)* to aggregate the evaluation results, store and output them for the user.

Clubmark includes several built-in utilities for *a*) identification of the server loading during benchmarking process *b*) structuring of the input and intermediate results *c*) structuring of the generating logs *d*) structuring of the efficiency results produced by an external application (`exectime`) *e*) structuring of the final results in the HDF5 storage.

Algorithms Profiling. Our benchmark features a Web interface facilitating profiling of the evaluating algorithms. In particular, the web interface allows to *a*) experience the REST WebAPI of the framework *b*) perform a simple profiling of the algorithms being executed *c*) inspect how load-balancing and isolation work for clustering algorithms executed in parallel *d*) monitor for potential issues and failed tasks when executing clustering algorithms.

Framework Extensions. Clubmark is designed keeping in mind its seamless extension with *a*) new clustering algorithms, *b*) multiple versions of a single clustering algorithm (which is a useful feature when designing a new clustering algorithm) and *c*) new evaluation measures implemented as external applications and in various languages.

Thus, our benchmarking framework facilitates profiling and provides comprehensive evaluation of diverse clustering algorithms, avoiding common pitfalls including various biases (e.g., bias to the order of input data, specific instance of a synthetic network, or single stochastic parameters of a clustering algorithm). Currently, Clubmark relies on the LFR framework [123] for generating synthetic networks with ground-truth clusters, which has some limitations when generating hierarchical structures. However, we are going to integrate also RB-LFR framework [265] to address those limitations by generating a more clear hierarchical structure that is extendable to an arbitrary number of hierarchical levels.

6 StaTIX: Entity Type Completion in Knowledge Graphs via Clustering

In this chapter, we show how our specific clustering algorithm, DAOC (introduced in Chapter 3), can facilitate the *refinement of knowledge graphs*. Based on DAOC, we propose a novel method, called StaTIX, for the automatic identification and inference of missing information, which we formalize as an entity type completion problem for Linked Data (e.g., Knowledge Graphs).

6.1 Introduction

A significant fraction of the data available in knowledge bases today are stored as *Linked Open Data (LOD)*. Large Linked Data projects such as the Linked Open Data Cloud ¹ or DBpedia [10] are often collaborative and contain data that has been extracted semi-automatically or that come from different sources. Hence, Linked Data often does not have a single maintainer or a strict unified schema for structuring the instances and as such typically includes noisy and/or incomplete data [271]. In particular, type information is often missing [186], which is particularly problematic as types are crucial for correctly handling many integration and post-processing tasks such as semantic search [236], federated query processing [176], linked data integration [50], or knowledge graph partitioning [130].

In this chapter, we propose a novel method, called StaTIX ² (for Statistical Type Inference), for automatically inferring instance types from Linked Data. Most methods in this context use supervised learning that leverage large, pre-labeled training sets (see the Related Work section). This can often turn out as a severe limitation, as such pre-labeled data are difficult and costly to acquire (or produce) for non-specialists or smaller entities, and as new labels are required for every new domain or dataset where type inference has to be applied. Our method instead is unsupervised and fully automated, and can be readily applied on any Linked Data source irrespective of its size or content.

¹<http://lod-cloud.net/>

²<https://github.com/eXascaleInfolab/StaTIX>

StaTIX performs link-based statistical type inference leveraging a dedicated clustering algorithm that significantly improves type inference accuracy compared to the state of the art. Our link-based type inference technique takes as input weighted statistics from multiple attributes (properties) of each instance and avoids the propagation of errors from isolated erroneous axioms, similar to [185], which allows it to operate on noisy data. In particular, we propose a novel approach to simplify (*reduce*) the processing complexity of the similarity matrix specifying the similarity between the instances. This reduction technique can speedup the clustering process by orders of magnitude, allowing to cluster larger datasets. Moreover, it can improve the overall accuracy of the results on noisy data. Also, we introduce a new optimization of the clusters formation process, using a dedicated hash function to speedup execution time by orders of magnitude without negatively affecting the resulting accuracy. Finally, we propose a novel technique to identify representative clusters from the resulting hierarchy, which further improves the accuracy of the type inference.

We perform an extensive empirical evaluation of our technique on real data and show that StaTIX significantly outperforms other unsupervised type inference approaches in terms of both effectiveness and efficiency. StaTIX reduces the accuracy error by about 40% on average comparing to other evaluated methods. In addition, StaTIX improves the execution speed by orders of magnitude and consumes less memory than the state of the art.

6.2 Related Work

The classical way to perform type inference is the application of logical reasoning, e.g., via RDFS/OWL entailment regimes [99, 233]. The resulting accuracy is highly dependent on the cleanliness and correctness of the statements in the knowledge base, though a number of works have attempted to reason on noisy semantic data [96]. Reasoning-based techniques are generally speaking considered as not suitable for cases where the knowledge base contains erroneous or conflicting statements [185]. In addition, logical reasoning only allows to infer information from the facts that are present in the dataset; it is unsuited to infer types when most of the *rdf:type* values are missing.

Several unsupervised type inference techniques have been proposed in the literature. In [107] the authors introduce a statistical method, which we refer to as *SDA*, to compare the conformity of a dataset against its schema using statistical type inference. The proposed technique is based on the concept of probabilistic type profiles consisting of a set of properties p and related probabilities α encoding the probability of an instance having p as a property. In addition to the type profiles, a profile is assigned to each class in the schema to assess the completeness of the dataset and its conformity to the schema. Paulheim et al. [185] proposed a link-based classification technique, called *SDType*, to infer missing types. *SDType* uses the statistical distribution of each link from an instance to assign types to instances. The statistical distribution is computed using a weighted voting approach, where a distribution of type votes is assigned to each link. The proposed technique outputs the confidence of

each instance-type pair. SDType is implemented on top a relational database and achieves a quasilinear runtime complexity with the number of statements in the dataset. It is important to outline that SDType requires some supporting database with ground-truth types (DBpedia is used by default), whose types are then assigned to the target dataset. Therefore, SDType aims solely at discovering types that are present in the supporting dataset, even if those types have very little statistical significance in the target dataset, which conceptually differs from the semantics of the SDA results.

Both SDA and SDType are directly related to our present effort and are evaluated against our approach in the following.

A number of supervised techniques have been proposed in this context as well. Klieger et al [108] proposed a supervised type inference technique called LHD 2.0 that extends the Linked Hypernyms Dataset (LHD) framework to extract types from DBpedia graphs. The proposed technique uses a statistical type inference (STI) technique to leverage the similarity between graphs by mapping classes appearing in one source knowledge graph, namely DBpedia, to another target knowledge graph, LHD. Together with the STI technique, the authors introduce an ontology-aware fusion approach based on hierarchical SVM to perform the assignment of instance types. LHD 2.0 combines a lexico-syntactic pattern analysis with supervised classification to assign the most probable types to the terms in the input text. Zhang et al [276] introduced a data mining type prediction technique for Linked Data. The proposed technique is based on a text classification algorithm and boils down to a three-step procedure. First, a maximum entropy estimation is applied to find bags of words (BOW) in an RDF graph. Then, a weighted virtual document of type information (VDT) is computed. VDT consists of sub-BOW of words from URI of object o_i , sub-BOW of literals from o_i 's annotation properties and a sub-BOW of URIs of o_i 's properties related to/from other objects. Each sub-BOW is represented using word frequencies. This technique requires some *a priori knowledge* on the number of target clusters and trains two classifiers to infer types. Böhmann et al. [22] introduced a system called DL-Learner to perform inductive learning on semantic web data. Their system provides an OWL-based machine learning tool to solve supervised learning tasks. It also supports knowledge engineers in constructing knowledge and learning about the data they created. A major component of this system is the induction process, which can be applied to infer types in a knowledge graph. Melo et al. [160] introduced a type prediction method called SLCN to tackle type incompleteness in Semantic Web knowledge bases with an ontology defining a type hierarchy. The authors formulate the type prediction problem as a hierarchical multi classification, where the class labels are types. The SLCN approach is based on a local classifier per node and performs feature selection, instance sampling, and class balancing. SLCN is applicable to large-scale RDF datasets with high-dimensional features. The aforementioned supervised techniques were not designed to be applicable on cases where we do not expect any prior information on the schemas and instance types (e.g., when we do not know the number of types a priori and do not have any training set with corresponding labeled types), which is the focus of this chapter (see Section 6.3.1 for more detail).

6.3 Method Overview

6.3.1 Problem Statement

We consider Linked Data statements specified as RDF triples (s, p, o) . Each triple is composed of three distinct components: a *subject* (a URI or blank node), a *predicate* (URI) and an *object* (a URI, blank node or literal). In the following, we denote as an *instance* V_i all triples sharing a given subject i . The predicates belonging to an instance correspond to attributes having a literal, a URI or a blank node as value. A special property defines the *rdf:type* of i , indicating that i is an instance of the class specified by the property value. Each instance may have multiple *rdf:type* properties, i.e., may belong to multiple classes. Our *objective* is to infer missing *rdf:type* values for all instances in the dataset G , considering the following: *a*) the schema is incomplete; hence, both *rdf:type* statements as well as class definitions may be missing; *b*) classes can themselves be organized as to create a hierarchy (e.g., through *rdfs:subClassOf* properties); *c*) the dataset may be noisy (hence, G may include incorrect statements). In other words, we aim to induce types (*rdf:type* values) for all instances, and classify each instance with respect to the discovered types for realistic scenarios where the data are both incomplete and noisy.

6.3.2 Unsupervised Statistical Type Inference

We now turn to a high-level description of our approach. We focus on a technique that is fully unsupervised and does not rely on any third-party knowledge base, and, therefore, can be readily applied to any Linked Data without any preparation or parameter tuning. The fundamental assumption behind our approach is that the more properties the instances share, the more likely they have the same types. Basically, we define the similarity between instances by matching their properties and then apply a dedicated clustering algorithm to infer the type clusters as shown in Figure 6.1.

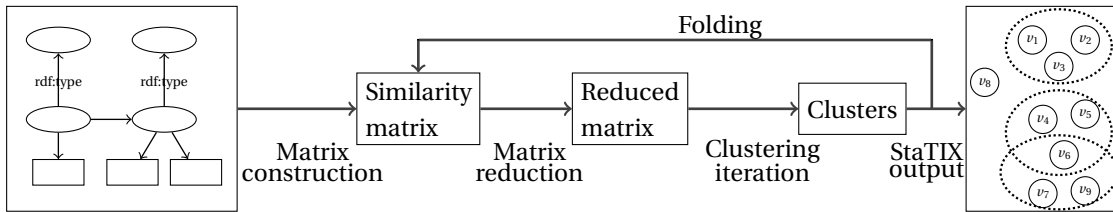


Figure 6.1 – Unsupervised Type Inference Process in StaTIX, where frames denote forms of the processing data and applied actions are displayed with arrows.

Our type inference technique, StaTIX, takes a LOD dataset as input, where some (or all) type information and class definitions are missing. From this input dataset, a *Similarity Matrix* capturing the similarity among the instances is constructed, as explained below in

Section 6.3.3. From there on, the type inference process is iterative, as the Similarity Matrix is reduced (see Section 6.4.1) and clustered (see Section 6.4.2) iteratively to infer clusters of types.

At the end of each iteration, weights for the resulting clusters as well as inter-cluster links are computed by aggregating the weight of the nodes in each cluster and the respective links (see *folding* on Figure 6.1). The resulting (*clustering*) graph forms a new Similarity Matrix and is used as a new input by the next iteration of the clustering algorithm. The clustering process terminates as soon as an iteration does not produce any new cluster. Clusters produced by this process form a hierarchy, where each level can be seen as representing the input dataset at a given level of granularity [126]. Clusters on the top (final) level of the hierarchy represent the inferred instance types. Inferring subtypes in addition to the coarse-grained types requires considering the non-top level clusters, which is described in Section 6.4.3 and which improves the accuracy of the type inference.

The computational complexity of StaTIX varies from $O(m)$ on sparse clustering graphs³ up to $O(m\sqrt{\frac{m}{n}})$ on dense noisy graphs, where m is the number of links in the graph and n is the number of nodes. The theoretical worst case corresponds to an extremely dense graph with weights yielding an excessive number of overlapping clusters and never occurs in practice thanks to our reduction technique (see below Section 6.4.1). The memory complexity of our approach depends on the same factors, and varies from linear on sparse graphs to $O(m \cdot n)$ on the same worst case. We show that our technique is both space and time efficient in practice in Section 6.5.3.

6.3.3 Similarity Matrix Construction

The input of the clustering algorithm is a (*clustering*) graph, which formally can be represented by a similarity matrix. The matrix stores pairwise similarities between the instances in the input (RDF) dataset. The similarities are computed like in [185] by applying a similarity function on vectors representing the set of properties attached to each instance. Both the property vectors and the similarity function are described below.

Property Vectors

Each instance in the input dataset is represented as a vector of its weighted properties. The weight w_i of property p_i expresses the importance of the property for the type inference and, intuitively, decreases for frequently occurring properties: $w_i = \frac{1}{\sqrt{freq_i}}$, where $freq_i$ is the number of occurrences of p_i in the dataset. We introduce the square root as the statistical distribution of links in real-world networks is typically heavy tailed [13] and as we want to take into account a large number of properties (beyond the head of the distribution). This

³Note that a *clustering* graph corresponds to the similarity matrix formed on the first iteration, which is typically much smaller than the input RDF graph

weighting function yields better results than equal or frequency weighting in practice.

Similarity Function

Various functions, such as Cosine or Jaccard, can be used to evaluate the similarity between the property vectors. We use the cosine similarity as it is known to be highly effective [131] and since it allows us to operate on weighted properties.

6.4 Type Inference

We now turn to the core of our method. We describe below the three main steps of our type inference pipeline: Reduction of the Similarity Matrix (Section 6.4.1), Clustering (Section 6.4.2), and Cluster Identification at Multiple Scales (Section 6.4.3).

6.4.1 Weight-Preserving Reduction of the Similarity Matrix

Our novel similarity matrix reduction technique is applied before each clustering iteration to reduce the cost of the subsequent processing and to improve the accuracy of the results thanks to the link denoising. The similarity matrix can be seen as an input graph for the clustering consisting of nodes (instances) and weighed links (pairwise similarities between the instances). The number of links is in the worst case equal to the squared number of nodes—which only occurs when all pairs of instances share some property. However, many links are insignificant in practice (as a given instance is typically related to a subset of the graph only) and as such can be omitted. Yet, carelessly removing lower-weight links can negatively impact the clustering process in two ways: *a*) a link connecting nodes *A* and *B* might be insignificant for node *A* but significant for *B*, and *b*) the total weight of the graph should stay constant in order not to affect the clustering of the remaining nodes (that are not adjacent to the nodes being reduced) when the *global* optimization function (e.g., modularity [174]) is applied. The following reduction approach takes care of both cases.

Our reduction technique consists of two steps. First, we *identify* insignificant links and then *convert* them to weights of their respective nodes to retain the total weight of the input graph. A link is considered as insignificant when it has no impact on the clustering of its incident nodes, which is defined by the optimization function of the clustering. In the scope of this chapter, we present a lightweight reduction approach, which is independent from the optimization function and operates on the link weights of the input graph directly. Our approach is inspired by the empirical observation that clusters formed by picking the minimum-weight link a node rarely maximize the optimization function. Moreover, the higher the number of links connected to a node, the lower the probability that the minimum-weight link impacts cluster formation. Hence, two key values should be taken into account when reducing the graph: *a*) the minimal number of links a node should have to be qualified for the reduction without

negatively affecting the clustering accuracy) and *b*) the maximal number of links that can be reduced in a node to not (significantly) affect the clustering.

The minimal number of links a node should have to be eligible for the link reduction is defined formally considering the following aspects.

- The performed reduction should not cause the formation of disconnected clusters (not linked to any node outside of the cluster). A cluster regroups together nodes with the most relevant relations, which roughly corresponds to the heaviest link weights. Therefore, the non-reducible (head) links of the node should include the heaviest links with at least two distinct weights.
- A node link can be considered for the reduction only if its weight is insignificant, i.e. the weight is closer to zero than to the heaviest link weight of the node: $w_i < \frac{w_o}{2}$.

The reducible (tail) links of the node effectively consist of at least one link. Therefore, a node being eligible for link reduction includes at least two remaining links with distinct weights and one lightweight link being reduced, which strictly defines the hard threshold, $l_{smin} \geq 3$. However, nodes having only three and even four links often do not satisfy our outlined restrictions. So, empirically we select $l_{smin} = 5$ considering that the reduction for nodes having at most four links does not yield any speedup for our applied clustering algorithm. The actual number of reducible links is defined automatically as follows for the nodes having at least l_{smin} links.

Our reduction technique is outlined in Algorithm 7 and illustrated in Figure 6.2. First, we order the links of each node by descending weight on line 4 of Algorithm 7. Then, we initialize the head links of the node by accumulating all the links having the two heaviest weights (lines 5-15). During the head link weight accumulation, the `rankedWeight` function (line 9) adopts an increasing ratio $rw_i \in (0, 1]$ starting from the second heaviest link ($i = 1$): $rw_i = \frac{2i}{ndlsnum-2}$, where $1 \leq i < \frac{ndlsnum}{2}$ and $ndlsnum \geq l_{smin}$ is the number of links in the node. Afterwards, we iteratively aggregate the reduction candidates in the tail and the remained links in the head (lines 16-28) till *a*) the tail has a lower weight than the head, *b*) the tail can be expanded with links not assigned to the head and *c*) the weight of each tail link is less than a half of the weight of the first head link $\frac{w_o}{2}$. Each iteration of the aggregation results in the addition of a single Head Weight bar and several Tail Weight bars until convergence as shown in Figure 6.2. The tail links being picked as reduction candidates are marked on line 35 for each node in the graph. The links marked from both of their incident nodes are identified as insignificant and removed on lines 29-36 transferring their weights to their respective nodes to retain the total weight of the graph.

Through this reduction, the size of the respective similarity matrix remains constant but the number of null values is increased, providing opportunities for more efficient storage and processing (only the non-null values are actually stored and processed by our system).

Algorithm 7 Weight-preserving Similarity Matrix Reduction

```

1: procedure REDUCEDENSITY(graph)
2:   lsmin = 5
3:   for node in graph where count(links(node)) ≥ lsmin do           ▷ Identify reduction
   candidates
4:     order(links(node))
5:     els = end(links(node)); bls = begin(links(node))
6:     ih = bls; wh = weight(ih); wc = wh
7:     for i in range(2) do
8:       while ++ih ≠ els and wc ≤ weight(ih) do
9:         wh += rankedWeigh(ih, node)
10:      end while
11:      wc = weight(ih)
12:    end for
13:    if ih == els then
14:      continue
15:    end if
16:    -- ih; wtlmax = weight(bls) / 2
17:    -- (it = els); wt = weight(it)
18:    while it ≠ ih and weight(it) < wtlmax do
19:      while wt < wh and weight(it) < wtlmax and it ≠ ih do
20:        wt += weight(-- it)
21:      end while
22:      if weight(it) < wtlmax and it ≠ ih then
23:        wh += rankedWeigh(ih++, node)
24:      end if
25:    end while
26:    wc = weight(it)
27:    while ++it ≠ els and wc ≤ weight(it) do
28:      end while
29:    for ln in range(it, els) do
30:      if marked(ln) then           ▷ Convert insignif. links
31:        addWeight(srcNode(ln), weight(ln) / 2)
32:        addWeight(dstNode(ln), weight(ln) / 2)
33:        remove(graph, ln)
34:      end if
35:      mark(ln)
36:    end for
37:  end for
38: end procedure

```

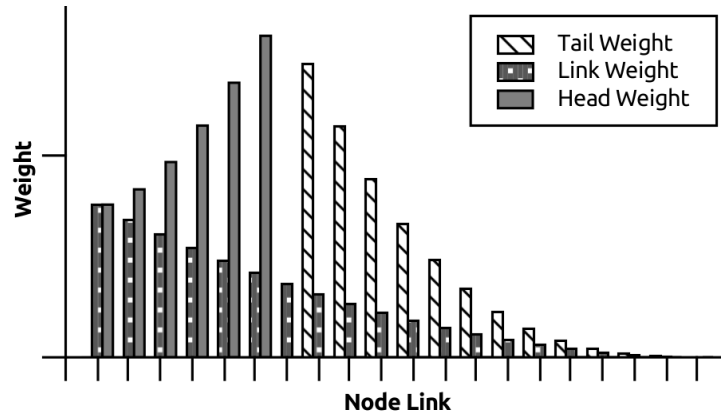


Figure 6.2 – Reduction candidates identification via the accumulation of weights of node links from the tail.

Moreover, the reduction implicitly acts as a noise filtering step, which often improves the accuracy of the subsequent clustering as we show in Section 6.5.3.

6.4.2 Clustering

We now turn to the unsupervised technique we use to infer clusters of instances sharing similar types. The problem definition that we consider imposes a number of requirements to the clustering algorithm. First, as an instance may have multiple types, we need an *overlapping* (also called fuzzy or soft) clustering algorithm to allow instances to belong to several clusters (i.e., types). Second, as types may also form hierarchies, using a *hierarchical* (or multi-scale) clustering technique would be desirable. Third, as we aim to infer types for any dataset without any manual labeling or tuning, the clustering algorithm should be *parameter-free* (without any parameter to tune). Finally, as the input dataset might be noisy, the clustering algorithm should be *robust*. In addition to those criteria, the clustering technique should be efficient and scalable; both its time and space complexity should be lower than quadratic to be applicable to large datasets in practice. In Chapter 3, we introduced the dedicated clustering algorithm, DAOC, to meet all those criteria. We recall the distinguishing features of DAOC that are essential in the context of type inference as follows.

Support for overlaps: Overlaps occur when a node is shared by several clusters and has an equally good value with each of them in terms of the optimization function. To support overlaps, we transform the original graph to represent such cases explicitly in the network by *decomposing* the node into (virtual) sub-nodes that can be processed independently in different clusters. This weight-preserving transformation does not influence the optimization function (as we ensure that the global modularity value stays constant), and allows to explicitly consider the fact that a single node can belong to several clusters.

Robustness: Robustness implies stable results even if the input data are shuffled or are subject

to minor perturbations (e.g., in the presence of noisy statements). Robust algorithms typically leverage some form of consensus (e.g. majority voting) to infer clusters by processing an input network multiple times and varying either the parameters of the clustering, the optimization function, or even the clustering algorithms in the meta-algorithm (e.g. OSLOM [125]). To avoid the high computational costs of such methods, we devised a new consensus approach in scope of DAOC as discussed in Chapter 3.3.1. The basic idea behind this consensus approach is simple: to cluster a pair of adjacent nodes together, we consider the (*mutual*) maximal modularity gain from *each* of the nodes instead of the maximal modularity gain from any of them. Thus, we apply a lightweight consensus approach, which yields robust (due to the consensus [165]) and fine-grained clusters at each level of the hierarchy.

Efficiency: A computationally heavy analysis has to be performed to decide whether a single or multiple overlapping clusters should be formed when a node has multiple clustering candidates (i.e., neighbors having the same mutual maximal value for the optimization function). This analysis includes the identification of the minimal subset of clustering candidates (of the origin node) being also clustering candidates between each other. If such a subset exists, then a single solid cluster is formed comprising inter-node relations. This analysis is the most computationally expensive step in our clustering process. However, there is a frequent special case, which can be identified and processed extremely efficiently. In semantic networks, a node often has multiple clustering candidates and all of them are also clustering candidates between each other. This case corresponds to a clear fine-grained subtype of some actual type (note that existing noise in the input data gets attenuated by the Similarity Matrix Reduction approach described above).

To identify such special cases with a reasonable efficiency (i.e. without having to order the clustering candidates and then comparing ordered sets) Bloom, Quotient Filter or any other (approximate) membership structures could be leveraged. However, we propose a different, much more efficient approach, which is theoretically optimal and yields a linear time complexity on the number of clustering candidates of the node. Our approach is a new *history-independent* (i.e., independent of the input order) hash function, AggHash. Generally, a history-independent hashing of a set can be achieved by applying any standard hash function to each item in the set with a subsequent commutative aggregate operator, which maintains the distribution of the hash values (e.g., xor for the uniform distribution). However, the resulting hashing is required to minimize the number of collisions, since each collision causes omission of the respective fine-grained clusters and results in merging them into a single super cluster. The latter requirement implies the application of some efficient cryptographic hash function, which is at least an order of magnitude slower than an effective non-cryptographic one. Also, a non-cryptographic hash function typically results in less uniform distributions of the hash values, which boosts the number of collisions yielded by the aggregating xor. To strike an ideal balance between accuracy and efficiency, we designed a dedicated hash function, AggHash.

AggHash is a cache-friendly, history-independent and aggregating hashing function for un-

ordered sets. Being cache-friendly and history-independent, AggHash is an order of magnitude faster on CISC architectures than `xor` operations on generic non-cryptographic hash functions (e.g., MurmurHash from the C++ standard library). Given a set A of unique node ids a_i of size N_A , AggHash is defined as:

$$\text{AggHash}(A) \rightarrow \{\{N_A, \sum_{i=1}^{N_A} a_i, \sum_{i=1}^{N_A} a_i^2\} \mid i = \{1, \dots, N_A\} \wedge a_i \in \mathbb{N}_0\} \quad (6.1)$$

The pseudo-code of AggHash is given in Algorithm 8. In line 5, AggHash performs a *correction* of the input values to prevent collisions. This correction increments each input value by the square root of the maximal estimated input value (the largest possible node id in the input graph). The intuition behind this transformation is the introduction of a lower bound for the partial values of the aggregating fields, since the probability of a collision drops quadratically with the increase of the smallest hashed value. The experiments we performed confirmed that this correction does not yield any collision on all datasets we processed. In lines 8-9, AggHash uses the addition operator (which is commutative), yielding an input nodes order-invariant result, i.e. performing a history-independent hashing for the clustering candidates (`clscands`) of the node.

Algorithm 8 AggHash Hashing

```

1: procedure HASHNODE(node)
2:   if hashed(node) then
3:     return
4:   end if
5:   nid = corr(id(node))                                ▷ Corrected node id
6:   hash = {1, nid, nid · nid}                         ▷ Init(num, sum, sum2)
7:   for cnode in clscands(node) do
8:     hash.num += 1; nid = corr(id(cnode))
9:     hash.sum += nid; hash.sum2 += nid · nid
10:  end for
11:  node.hash = hash
12:  for cnode in clscands(node) do
13:    cnode.hash = hash
14:  end for
15: end procedure

```

6.4.3 Representative Clusters Identification at Multiple Scales

We propose a new technique to identify *representative* clusters at multiple scales. We call a cluster representative if it is likely to represent an actual type, which happens only with a fraction of all clusters in the resulting hierarchy. The representative clusters include *a*) the top level of the resulting hierarchy, which corresponds to coarse-grained types, and *b*) some

clusters on the lower levels (smaller scales), which correspond to fine-grained subtypes. The intuition behind our selection technique is explained below.

We illustrate our idea through an example. Figure 6.3 depicts a few clusters of nodes, where the similarity between the nodes is represented by their spatial closeness (density). Cluster C_2 consists of several sub-clusters (C_{21}, C_{2k}, \dots) produced at lower levels of the hierarchy. Many sub-clusters, including C'_{2k} , have a density (i.e., strength of the pairwise instance similarity) lower than the average density of their super cluster C_2 . Such a density variation is typical from real-world networks because of the heavy tailed distributions of node degrees and link weights [13], which result in a heavy-tailed distribution of cluster size and densities. The sub-clusters having a lower density than their super cluster typically do not represent groups that are statistically significantly different from their super cluster. Therefore, we select sub-clusters with a higher density of nodes only, which are likely to represent actual subtypes in the Linked Data.

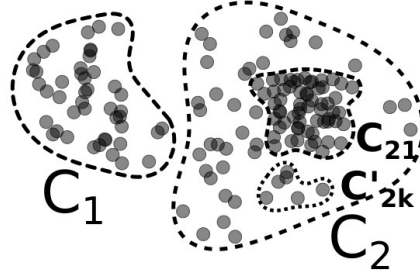


Figure 6.3 – C_1 , C_2 and C_{21} are representative clusters at various scales, C'_{2k} -like clusters are filtered out.

In particular, we apply the following technique to retain only the most representative clusters, i.e., the inferred subtypes. Starting from the top level of the hierarchy, we evaluate the density of each cluster as its weight divided by its number of nodes. All sub-clusters having a density lower than their direct super clusters or having a weight close to either *a*) 0 or *b*) the weight of their super cluster, are filtered out as non-representative clusters. As a result, we end up with high-density clusters only, which have distinct statistical properties and are more likely to represent actual subtypes as we empirically show in the following section.

6.5 Evaluation

In this section, we first describe our experimental environment, including the evaluation metrics and datasets we used. Then, we present and discuss our results. The evaluation was performed on a Linux Ubuntu 16.04.3 LTS server with an Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz (8 cores) and 32 GB RAM. Our evaluation framework (including all scripts and datasets) is open-source and available online ⁴.

⁴<https://github.com/eXascaleInfolab/TInfES>

6.5.1 Metrics

We initially assume that some (or all) type labels might be missing. We measure the accuracy of the resulting unlabeled types, which are represented by clusters of instances, using *F1h*, which was introduced in Chapter 4. In case some type labels are available, we assign each label to the *best-matching inferred cluster* using a weighted F1-score [201] as matching criterion. Note that we assume that the labels might be missing and that the total number of types is unknown.

We measure the efficiency of the algorithms by measuring their runtime (in seconds) and their peak main memory consumption (in MB).

Weighted F1-score of Labeled Types (LF1)

The F1-score together with Precision (P) and Recall (R) are a commonly used metrics for measuring the accuracy of labeled types. The weighted F1-score of the labeled types (LF1) represents the average F1-score of each labeled type ($g(l, C)$) weighted by the number of instances in the label ($|l|$):

$$LF1(C', C) = \frac{1}{|C'|} \sum_{l \in C'} |l| \frac{2P_{l,g(l,C)}R_{l,g(l,C)}}{P_{l,g(l,C)} + R_{l,g(l,C)}}. \quad (6.2)$$

Note that each label can be assigned to several inferred types, that some inferred types might not have any assigned label, and that some types might have multiple labels. Thus, LF1 measures the accuracy of only the *labeled types*, where F1h measures the accuracy of *all resulting types*.

6.5.2 Datasets

We consider three distinct categories of real-world Linked Open Data (LOD) datasets from various domains to evaluate and ensure a wide applicability of our unsupervised type inference. It is worth outlining that a variety of data relations exist besides the distinct categories in the selected datasets. Each dataset contains multiple types, some datasets contain extremely diverse granularity of types while the granularity varies only slightly in others. Some of the types may contain a single instance even, which makes them statistically indistinguishable from noise. Instance types can be fully or partially overlapping with other types and some instances may not be attached to any type at all. The first category of datasets are samples of DBpedia used for the SDA evaluation [106]. This category is extended with *mixen*, representing the union of the samples of the category datasets belonging to the English DBpedia. The second category are biomedical datasets⁵ while the third category are open government datasets⁶. Some statistics about each dataset are listed in Table 6.1. The link density of the

⁵<http://download.bio2rdf.org>

⁶<https://opendata.swiss/en/dataset/>, <https://data.gov.uk/dataset/schools2>

input graph is evaluated as the number of links (edges) divided by the maximal possible number of links ($nodes \cdot (nodes - 1) / 2$).

Table 6.1 – Evaluation Datasets.

Dataset	Triples	Types	Nodes	Links	Density
museum	1418	84	178	7143	0.453
soccerplayer	2654	172	272	11008	0.299
country	2273	65	453	22176	0.217
politician	3783	200	523	32977	0.242
film	6334	5	1303	822557	0.970
mixen	10128	475	1426	244408	0.241
gendrgene	5651	7	532	140888	0.997
lsr	56507	11	5767	7621860	0.458
bauhist	9022	2	861	186460	0.504
schools	15347	3	2256	847320	0.333
histmunic	119151	14	12132	73380691	0.997

Note that not all those datasets define ground-truth types for all their instances. In case the ground-truth is missing for a given instance, we simply discard the instance when evaluating the F1 score.

6.5.3 Results and Discussion

We compare both the effectiveness and the efficiency of our method against two state of the art unsupervised statistical type inference methods: SDA [107] and SDType [185]. Additionally, we evaluate the impact of each proposed technique; *StaTIX-rm-m-f* denotes the final results below, where *rm* stands for similarity matrix reduction, *m* for representative clusters identification, and *f* for fast clusters formation using AggHash. Since one of our main objectives is unsupervised type inference *without any manual tuning*, the algorithms we evaluate are executed without any modification or parameters tuning. The latter is important for SDType, which is the only algorithm we evaluate requiring a supporting dataset. Moreover, SDType being tuned by default for DBpedia, it only considers incoming links for types discovery, while StaTIX and SDA consider all available links. However, in order to not penalize SDType for types present in DBpedia but missing in the ground-truth, we discard such cases from the SDType results.

Effectiveness

Effectiveness results (in terms of *F1h* score) are shown in Figure 6.4. Our approach outperforms SDA reducing the *F1h* error by 37% on average. On the last dataset, *histmunic*, the results are absent for SDA as it throws a Java heap space error after 30 hours of evaluation. SDA

yields noticeably more accurate result than StaTIX does on a single dataset (*gendrgene*) only, which has a ground-truth with heavily overlapping clusters of significantly varying sizes. Both StaTIX and SDA inferred the same number of types in this case, but StaTIX resolved overlaps more strictly by pruning some types that were correctly retained by SDA. However, SDA has a relatively large variance in accuracy, which can be explained by the parameterized clustering algorithm (DBSCAN) used for the inference having default parameter values. In particular, SDA fails to detect large types with relatively weak relations between member instances, i.e. coarse-grained clusters of medium density, in half of the evaluated datasets. On the contrary, StaTIX yields a good accuracy with only small deviations from the ground-truth for all datasets independently of their size or density, thanks to our dedicated clustering algorithm.

The impact of our techniques on accuracy is shown in Figure 6.5, where StaTIX-rm-m-f and StaTIX-rm-m are displayed with a single bar since the resulting clusters are exactly the same (AggHash does not affect the structure of the resulting clusters). Our similarity matrix reduction approach significantly improves the accuracy on several datasets, which can be explained by the denoising effect, and in particular by filtering out a number of noisy properties that caused the original clustering to get stuck on local optima. The technique for representative clusters identification does not significantly impact the accuracy in terms of F1h, but improves the F1-score of the labeled types as shown in Table 6.2. Most of the representative clusters correspond to fine-grained ground-truth labels, which otherwise are assigned to larger clusters and negatively impact recall. However, not all statistically representative clusters are present in the ground-truth, which penalizes F1h. But even the statistically representative clusters that are absent in the ground-truth can be useful as they can help identify candidates for fine-grained types or outliers.

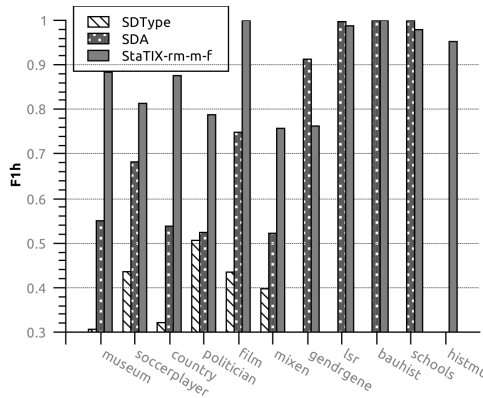


Figure 6.4 – Accuracy of the unsupervised statistical type inference algorithms by F1h measure.

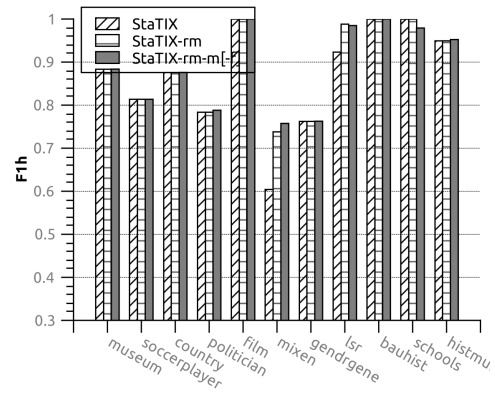


Figure 6.5 – Impact of the similarity matrix reduction technique on StaTIX accuracy by F1h measure.

Chapter 6. StaTIX: Entity Type Completion in Knowledge Graphs via Clustering

Table 6.2 – Accuracy of the labeled types by the weighted F1-score, the positive impact of the proposed techniques is outlined in bold.

Dataset	StaTIX	StaTIX-rm	StaTIX-rm-m[-f]			SDA			SDType		
	F1	F1	F1	P	R	F1	P	R	F1	P	R
museum	0.866	0.866	0.866	1.00	0.76	0.539	0.38	0.93	0.209	0.12	0.79
soccerplay	0.789	0.789	0.789	1.00	0.65	0.695	0.57	0.88	0.447	0.34	0.66
country	0.840	0.840	0.840	1.00	0.73	0.632	0.48	0.93	0.249	0.16	0.63
politician	0.732	0.732	0.756	0.98	0.62	0.704	0.59	0.87	0.471	0.40	0.57
film	1.000	1.000	1.000	1.00	1.00	0.839	0.72	1.00	0.435	0.28	1.00
mixen	0.505	0.723	0.751	0.87	0.66	0.559	0.41	0.87	0.378	0.36	0.40
gendrgene	0.806	0.806	0.806	0.76	0.86	0.889	0.99	0.81			
lsr	0.912	0.990	0.990	1.000	0.981	0.998	0.99	1			
bauhist	1.000	1.000	1.000	1.00	1.00	1.000	1.00	1.00			
schools	1.000	1.000	1.000	1.00	1.00	1.000	1.00	1.00			
hismunic	0.950	0.950	0.958	1.00	0.92						

Efficiency

In terms of space efficiency StaTIX consumes 1.5x-20x less memory than its competitors as shown in Figure 6.6. Memory consumption for SDType was not evaluated for the datasets where it produced empty results. Note that SDA throws a heap space error on the last dataset. StaTIX consumes slightly more memory than SDA on the *film* dataset only, which is the smallest dataset having a density of links close to the theoretical maximum with a relatively small variation in terms of the link weights. The memory overhead itself is caused by the deferred garbage collection in the JVM of StaTIX resulting in the storage of two similarity matrices in memory when the matrix is streamed to the underlying native clustering library. Our similarity matrix reduction technique speeds up subsequent clustering steps but does not affect the peak memory consumption since the matrix is generated and loaded in memory before being reduced. To execute StaTIX on large LOD datasets, each column and row of the similarity matrix could be stored as a memory-mapped file on an SSD drive.

The execution time of the algorithms is shown in Figure 6.7, except for the cases where SDType produced empty results. StaTIX and SDType are implemented as single-threaded applications, whereas SDA takes advantage of multiple CPU cores. Nevertheless, StaTIX performs type inference for the largest dataset (*hismunic*) within 150 seconds, while SDA spends about 30 hours on up to 8 cores throwing an *OutOfMemoryError* exception in the end. In addition, we note that the type inference of StaTIX takes 35 seconds only, while most of the execution time is spent by StaTIX for I/O and RDF processing (consuming several GBs of memory also). As shown in Figure 6.8, our similarity matrix reduction technique results in orders of magnitude speedups on several input datasets (e.g., 40x on the *film* dataset). Fast clusters formation by AggHash speeds up execution on all remaining datasets by a similar magnitude. Essentially, the execution time of StaTIX is bound by the I/O throughput and RDF conversion.

In conclusion, we improve over the state of the art by reducing the accuracy error by 40%

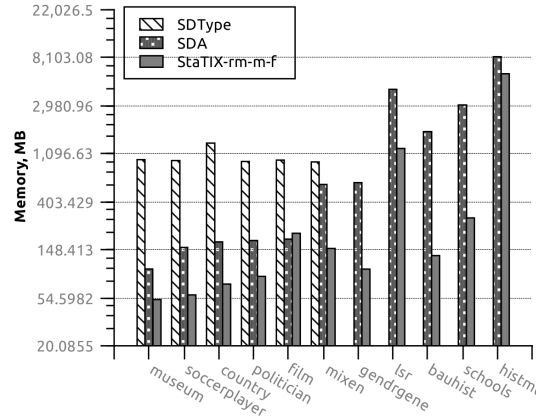


Figure 6.6 – Memory consumption of the unsupervised statistical type inference algorithms.

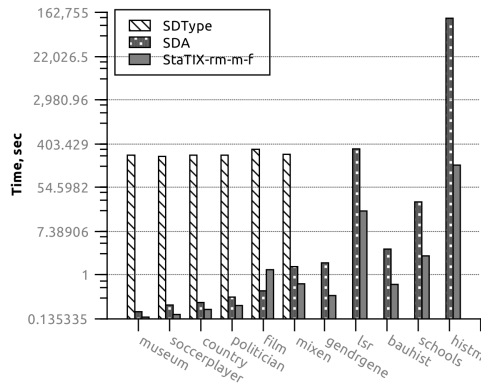


Figure 6.7 – Execution time of the unsupervised statistical type inference algorithms.

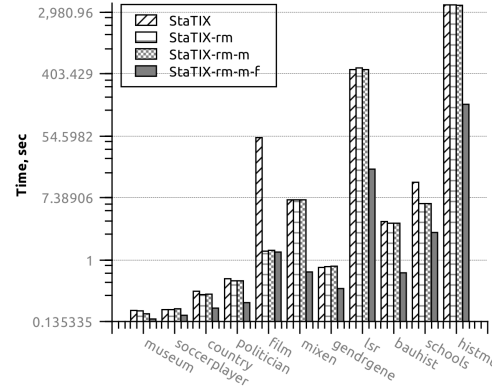


Figure 6.8 – Impact of the similarity matrix reduction technique on StaTIX execution time.

on average and by reducing the execution time by up to three orders of magnitude on the evaluated datasets while requiring considerably less memory.

6.6 Conclusions and Future Work

In this chapter, we introduced a new statistical type inference method, called StaTIX, to infer instance types in Linked Data in a fully automatic manner without requiring any prior knowledge about the dataset. Our method is based on a new clustering technique, DAOC (presented in Chapter 3), which infers (overlapping) types in a robust and efficient manner. As part of our method, we also presented novel techniques to *a*) reduce the similarity matrix representing relationships between the instances, *b*) speed up the clusters formation using a dedicated, history-independent hash, AggHash, and *c*) identify representative clusters at multiple scales. We empirically compared our approach on a number of different datasets and showed that it is at the same time considerably more effective and orders of magnitude more

efficient than state-of-the-art techniques.

In the future, we plan to extend StaTIX with additional semantic analysis leveraging both logical reasoning and embedding techniques to better grasp the differences and relationships between various instances. We also plan to add support for automatically borrowing type labels from third-party knowledge bases whenever available. In terms of implementation-specific aspects, we plan to parallelize our algorithm to take advantage of modern multi-core CPU architectures.

7 DAOR: Fully Automatic and Interpretable Graph Embedding via Clustering

In this chapter, we show how our specific clustering algorithm, DAOC (introduced in Chapter 3), can facilitate the construction of *graph embeddings* without any manual tuning. Based on DAOC, we propose a novel method, called DAOR, for the automatic construction of embeddings, which are interpretable and robust to various metric spaces. In addition, we identify the key features of clustering algorithms required to efficiently produce effective graph embeddings.

7.1 Introduction

Representation learning has become a key paradigm to learn low-dimensional node representations from graphs. These automatically generated representations can be used as features to facilitate downstream graph analysis tasks such as node classification and link prediction. The main idea behind graph embedding techniques is to project graph nodes onto a low-dimensional vector space such that the key structural properties of the graph are preserved. The most commonly preserved property in this context is the *proximity* between nodes in the graph [24]. For example, DeepWalk [191] and Node2vec [77] preserve up-to- k -order node proximity by sampling random walk sequences from an input graph using a context window of a certain size; HOPE [179] and NetMF [198] capture high (up-to-infinite)-order node proximity by factorizing high-order node proximity matrices, measured by the Katz index [102], for example. The resulting embedding vectors from those techniques, capturing node proximity in graphs, are known to achieve good results on many downstream graph analysis tasks, such as node classification, node clustering (a.k.a. community detection) and link prediction.

Among various graph analysis applications, community detection is one of the most popular tasks. Network communities (i.e., node clusters or granules [266]) represent groups of nodes that are densely connected inside each group and loosely connected between different groups [171]. In essence, community detection techniques intrinsically capture node

proximity in the graph to generate such node clusters. In the context of graph embeddings, community detection naturally implies that nodes in a cluster should be projected closer to each other than to the nodes from other clusters in the vector space. For example, DeepWalk [191] performs node sampling using random walks, such that nodes from the same cluster (intra-cluster nodes) are linked tighter together than nodes from different clusters (inter-cluster nodes) and have a higher probability to be closer in random walk sequences. In the case of HOPE [179], the node proximity is defined by the Katz index [102] that computes the weighted sum of all paths between two nodes. There, intra-cluster nodes also have a higher proximity measure than inter-cluster nodes, since there are more paths linking nodes in the same cluster than paths linking nodes between different clusters.

In this chapter, by revisiting existing graph embedding techniques, we raise the following question: *“Can we generate node embeddings from clusters produced via community detection?”* More precisely, we explore how to generate node embeddings in a graph leveraging the latest advances in community detection techniques, analyzing and addressing emerging issues in the embedding learning process.

In the current literature, the graph embedding problem has also been investigated to specifically preserve community structures in a graph by applying hybrid approaches [26, 248]. These approaches consist in specific graph embedding models jointly learning node embeddings and performing community detection, where the two steps are performed iteratively until convergence. However, such hybrid methods preserving community structures lose other inherent advantages from modern community detection techniques:

- **Parameter-free** community detection does not require any manual tuning, while current embedding techniques (hybrid or not) impose significant human efforts. Parameter-free processing could significantly simplify the application of graph embeddings in practice compared to current graph embedding techniques, which require manual tuning of multiple parameters (including the number of embedding dimensions).
- **Metric-Robustness:** community detection typically does not rely on any specific metric space (e.g., cosine, Jaccard, or Hamming spaces), which provides an opportunity to obtain metric-robust node embeddings. More precisely, as existing graph embedding techniques are designed to learn node embeddings in a specific metric space (e.g., cosine [191] or Hamming [258]), they are often limited to the specified metric space. To ensure the applicability of learned embeddings in a wide variety of settings, it might hence be beneficial to learn embeddings that are robust to different metric spaces.
- **Efficient** community detection techniques usually have linear or near-linear runtime complexity and are able to handle large graphs consisting of billions of nodes [17], which may significantly speedup the embedding learning process. Specifically, existing graph embedding techniques usually either sample a large number of node pairs from a graph to learn node embeddings via stochastic optimization, or factorize a high-order proximity/adjacency matrix, which requires significant computational resources. Therefore, it might be desirable to let graph embedding techniques benefit from the high-efficiency of state-of-the-art community detection techniques.

In this chapter, we bridge the gap between community detection and node embeddings. More specifically, our main contributions can be formulated as follows: *a)* we propose a mechanism to generate node embeddings from given community structures (i.e., a set of clusters) and *b)* we identify the key features of community detection algorithms required to efficiently produce effective graph embeddings. Both of these contributions are applied on top of the Louvain [17]-based DAOC¹ [142] clustering algorithm and constitute our novel graph embedding framework called DAOR².

7.2 Related Work

7.2.1 Graph embedding

Graph embedding techniques project graph nodes onto a low-dimensional vector space such that the key structural properties of the graph are preserved [24]. Existing techniques can be classified into four categories. First, graph-sampling based techniques design specific embedding models to learn node embeddings from sampled node pairs from an input graph. The node pairs are often sampled by scanning random walk sequences from an input graph using a context window of size k to capture up-to- k -order node proximity [191, 259, 90], or directly sampled to capture 1st- and 2nd-order node proximities [228]. Second, factorization-based techniques decompose specific node proximity matrices, such as high-order transitional matrices [25], high-order proximity matrices measured by the Katz index, personalized PageRank or Adamic-Adar [179], to output node embeddings. Third, hashing-based techniques resort to similarity-preserving hashing techniques [252, 260] to create node embeddings, capturing high-order common neighbors between nodes in a graph. Due to the high efficiency of the hashing process, these techniques show significant speedup in the embedding learning process compared to the techniques of the two previous categories [260]. Fourth, a few meta techniques are designed to preserve higher-order structural features by hierarchical coarsening of the input graph prior to the embedding process [34]. As they capture higher-order node proximity in the graph, modern graph embedding techniques have shown good performance on various graph analysis tasks, including node classification, node clustering and link prediction.

In the current literature, the graph embedding problem has also been investigated to specifically preserve community structures in a graph [26, 248]. These hybrid techniques combine the objectives of both node embedding learning and community detection. More precisely, community detection and node embedding learning are tightly coupled and are performed alternatively to enhance each other in the learning process. The resulting node embeddings preserve the community structures of the input graph. However, such hybrid approaches are not able to directly reuse results of existing community detection techniques, losing their advantages as outlined in the introduction, i.e., they typically are not parameter-free,

¹<https://github.com/eXascaleInfolab/daoc>

²<https://github.com/eXascaleInfolab/daor>

metric-robust, or particularly efficient (as we show in Section 7.5).

In this chapter, instead of jointly learning node embedding and detecting communities in a graph, we take an alternative solution to bridge the gap from detected communities to node embeddings. Specifically, we *a)* design a new mechanism to generate node embeddings directly from given community structures (i.e., clusters) and *b)* analyze the various aspects of community detection techniques required for effective node embeddings generation from the clusters. We implement these contributions in a novel graph embedding framework called DAOR².

7.2.2 Community detection for graph embedding

To the best of our knowledge, only matrix factorization-based (MF) community detection methods have been used to directly generate graph embeddings. Spectral clustering is a matrix factorization approach [77], which was the first community detection method used for graph embedding. It can be applied in two variations: *a)* conventional spectral clustering [231] (introduced in [59, 195] and operating on a Laplacian matrix) and *b)* spectral optimization of modularity [229, 231] (introduced in [169] and operating on a modularity matrix). Node embeddings in this context are represented as the top- d eigenvectors (i.e., latent dimensions) of the respective matrix. Conventional spectral clustering is equivalent to nonnegative matrix factorization (NMF) [47]. The latter is another community detection method [262], which is applied jointly with spectral optimization of modularity to learn node embeddings [248].

The Gaussian mixture model (GMM) is a statistical inference-based community detection method [272], which can be used jointly with a conventional node representation learning (e.g., Deepwalk [191] and SDNE [245]) to perform graph embedding [26]. It is worth noting that GMM by itself explicitly learns the “random mixtures over latent communities variables” [272] (i.e., node embeddings) but suffers from a large number of parameters and does not take into account the low-order proximity of the nodes when generating the graph embeddings.

In essence, community detection by modularity maximization [172], statistical inference [220], normalized-cut graph partitioning [269] and spectral clustering are equivalent under certain conditions [173]. According to [145], community detection through generalized modularity maximization is equivalent to the provably correct but computationally expensive maximum likelihood method applied to the degree-corrected stochastic block model [101, 187]. The latter inspired us to develop a novel graph embedding framework, which is able to generate node embeddings directly from the detected communities, and to extend an efficient community detection method based on parameter-free optimization of generalized modularity to produce effective embeddings.

7.3 Transforming Clusters into Node Embeddings

Community detection algorithms only generate clusters as groups of nodes, which hence requires some post-processing to produce node embeddings, namely to: *a)* form latent embedding dimensions from the clusters (i.e., extract features) and *b)* quantify node membership in each dimension (i.e., generate the embedding vector for a node). In addition, it is often desirable to manually control the number of embedding dimensions d . These aspects are described in the following and are illustrated in Figure 7.1.

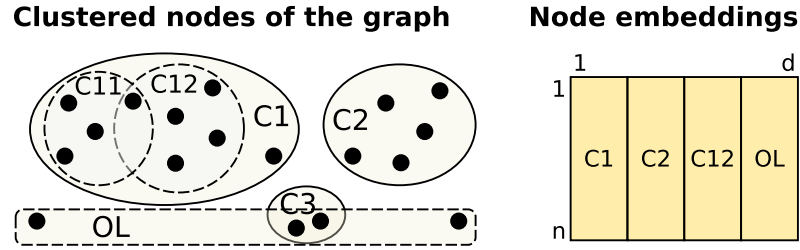


Figure 7.1 – Transformation of the hierarchy of overlapping clusters consisting of n nodes and $k = 5 + 2$ clusters (2 nodes-outliers, which are grouped as OL together with the outlier cluster C3) into $d = 4$ dimensional node embeddings.

7.3.1 Feature Extraction from Clusters

Intuitively, a straightforward approach for feature extraction from clusters is to consider each cluster as a dedicated feature representing one embedding dimension [229]. This approach can be used to discriminate between the nodes (given a fine-grained structure of clusters), but on the other hand, may not correlate with the graph embedding objective of providing a *low-dimensional* representation of the nodes. Fixing the number of clusters to the required number of dimensions d is possible when considering flat clustering [229] but not for hierarchical cases. For hierarchical clustering, producing d dimensions can be expressed with an inequality in the general case, i.e. producing *at most* d clusters at the top level. Therefore, some technique to identify a small number $s \geq d$ of *salient* clusters, which are then regarded as features, is required. This number s can either be parameterized or, ideally, inferred as the optimal tradeoff between the number of clusters and their quality. In addition, there exists a fundamental constraint on the structure of the clusters used to create the embedding space. Namely, each graph node should be connected to at least one feature to be present in the embeddings.

The exact way salient clusters (features) are identified depends on the structure of the clusters (e.g., hierarchical, multi-resolution, overlapping, non-overlapping) and on the ability to control the number of formed clusters $k \geq s$ by the community detection algorithm. In the following, we discuss salient clusters identification for the most general case, i.e., for clustering techniques that are multi-resolution, hierarchical and overlapping. Features extraction is presented in Algorithm 9. We traverse all clusters on each level of the formed hierarchy of

Algorithm 9 Features extraction

```

1: function FEATURES(hier)
2:   scls  $\leftarrow$  [] ▷ Salient clusters (features), dynamic array
3:   clsts  $\leftarrow$  {} ▷ Ancestor cluster statistics, hash map
4:   for all lev  $\in$  hier do
5:     for all cl  $\in$  lev do
6:       res  $\leftarrow$  count(cl.ances) = 0 ▷ Salient flag
7:       dens  $\leftarrow$  cl.weight/count(cl.nodes) ▷ Density
8:       savdens  $\leftarrow$  0; savwgh  $\leftarrow$  0 ▷ Statistics to be stored
9:       if not res then
10:        hits  $\leftarrow$  0 ▷ Saliency hits
11:        for all ac  $\in$  cl.ances do ▷ Traverse ancestors
12:          ast  $\leftarrow$  clsts[ac] ▷ Ancestor statistics
13:          if savdens < ast.dens and (not savwgh or savwgh > ast.wgh) then
14:            savdens  $\leftarrow$  ast.dens
15:            savwgh  $\leftarrow$  ast.wgh
16:          end if
17:          if not res and dens  $\geq$  ast.dens and cl.weight  $\leq$  ast.wgh then
18:            hits  $\leftarrow$  hits + 1
19:          end if
20:          ast.reqs  $\leftarrow$  ast.reqs + 1
21:          if ast.reqs = count(ac.des) then
22:            clsts.erase(cl) ▷ Remove outdated
23:          end if
24:        end for
25:      end if
26:      if not res and hits = count(cl.ances) then
27:        res  $\leftarrow$  true
28:      end if
29:      savwgh  $\leftarrow$  cl.weight * wrstep
30:      clsts[cl]  $\leftarrow$  (savdens, savwgh) ▷ dens,wgh,reqs=0
31:      if res then
32:        scls.add(cl)
33:      end if
34:    end for
35:  end for
36:  return scls ▷ The resulting salient clusters
37: end function

```

clusters starting from the top level, and fetch as salient clusters: *a*) the top-level clusters of the hierarchy (line 6) (to cover all nodes) and *b*) all the nested clusters satisfying the following requirements (line 17):

- having a higher weight density than each of their direct super-clusters (ancestors), since the nested salient clusters are expected to represent the mutual core structure [18] of their ancestors, and
- weighting less than the most lightweight ancestor (discounted by a factor).

The weight discounting factor r_w (line 29) is required to prevent fetching too many similar clusters that are nested into each other. The factor $r_w \in (0.5, 1)$ retains an approximation of the nested cluster to its ancestor, while $r_w \rightarrow r_{w_{min}} = 0.5$ reduces the number of salient clusters. However, considering the availability of overlapping clusters and known weight of the overlap $w_{C_{ovp}} < w_C$ for the cluster C , we refine r_w as $r_w = 0.5 + \frac{(b-1) \times w_{C_{ovp}}}{2b \times w_c}$, where $b \geq 2$ is the overlap factor equal to the number of clusters sharing the overlapping nodes in C . Also, the number of overlapping clusters can be estimated according to the Pareto principle [183, 20], which makes reasonable to take $r_w = 0.5 \times 1.2 = 0.6$ when the exact value cannot be evaluated.

7.3.2 Dimension Formation from Features

Embedding dimensions are formed from the salient clusters extracted above, which implicitly yields the recommended number of dimensions for the input graph based on its statistical properties. The embedding vector $v_i \in V$ of size $d = |D|$ for each graph node $\#i$ is then produced by quantifying the degree of belonging of the node to each dimension D_j as the node weight w_{i,D_j} belonging to this dimension. This weight corresponds to the aggregated weight of the node links to other nodes being members of all salient clusters composing the dimension:

$$V = \left\{ \frac{w_{i,D_j}}{w_i} \mid i = 1..n, j = 1..d \right\}, \quad (7.1)$$

$$w_{i,D_j} = \sum_{k \in nodes(D_j)} w_{i,k}.$$

We perform this embedding vectors generation efficiently by resorting to a single scan over the members of each salient cluster. The node weights are then aggregated to form the dimension values. The embedding vectors are obtained by transposing the resulting matrix: $V = D^T$.

Constraining the Number of Dimensions In the case of a connected graph without clusters-outliers, the most salient clusters d are fetched from the $t \leq d \leq s$ top level clusters of the hierarchy and from the $d - t$ densest of the remaining $s - t$ salient clusters. When the clustering algorithm does not allow to control the number of top level clusters or when the graph is disconnected and the number of components is larger than d resulting in $t > d$, then the dimensions are formed as follows. According to the so-called ‘‘Rag Bag’’ formal constraint of clustering quality [141, 4], the $t - (d - 1)$ most lightweight clusters should be grouped together

to the last dimension and the $d - 1$ heaviest clusters fill the remaining dimensions. However, the presence of outliers on the top levels prevents to effectively generate dimensions from the salient clusters. To solve this issue, the outliers can be separated from the valid clusters based on their weights, which are either evaluated from the statistical distributions or approximately estimated as follows. In case there is no prior information about the data, a rule of thumb is to take the estimated minimal size of clusters as the square root of the number of nodes [150]. Generalizing the rule of thumb to the weighted graphs, the number of $z < t$ root clusters, each having a weight less than the square root of the graph weight w can be moved to the “outliers” dimension. The resulting dimensions are composed of the $t - z$ top level clusters of the hierarchy, the $d - 1 - (t - z)$ densest remaining salient clusters, each having a weight $w_i \geq \sqrt{w}$, and a single dimension of outliers to cover all remaining graph nodes (see Figure 7.1 for an illustration).

Dimension Interpretability The resulting embedding space is inherently easier to interpret than spaces derived from other techniques, as its dimensions are taken from (salient) clusters representing ground-truth semantic categories, with accuracy being evaluated using extrinsic quality metrics [141]. This opens the door to new features and applications in the fields of granular and privacy-preserving computations. For example, only a subset of the dimensions having some required semantics can be fetched for evaluation, which has the advantage of reducing the amount of processing data while avoiding to leak or share information beyond the required features.

7.4 Community Detection

In this section, we first discuss the properties of a community detection algorithm that are required to perform an effective and efficient graph embedding. Then, we select one of the most suitable state-of-the-art community detection algorithms for our task and propose its extension to satisfy the required properties.

As an effective graph embedding technique should preserve both low- and high-order node proximities, the community detection algorithm used for the graph embedding should be able to produce clusters with various resolutions (i.e., at different granularities). Moreover, the more resolutions are covered, the wider the range of node proximity orders that can be captured, since each embedding dimension consists of at least one cluster as described in Section 7.3.2. A low-dimensional representation of graph nodes implies a *small* number t of coarse-grained (macro-scale) clusters, since the number of generated dimensions $d \geq t$ (see Section 7.3). This number t should be a parameter of the technique when a specific number d of embedding dimensions is required, with a default value defined according to the statistical properties of the input graph.

In addition, the following properties of community detection algorithms are also required to generate high-quality embeddings, to simplify and speedup the generation process:

- Each graph node should potentially belong to multiple features (i.e., should be represented in multiple dimensions), which requires the clustering to be soft (i.e., fuzzy or *overlapping*).
- It is desirable to have graph embedding techniques applicable to any input graph without any manual tuning; hence, the clustering should be *parameter-free* and *efficient* (to be applicable to large graphs).
 - An unsupervised parameter-free processing is sensitive to the quality of the input data, so a *robust* clustering algorithm is required. Robustness is typically reached by applying a consensus (also called ensemble) clustering method [64, 243, 125].
 - From a practical perspective, it is desirable to have consistent embeddings for the same input graph irrespective of the order in which the nodes are processed or whether their IDs are modified. The clustering algorithm should hence be *deterministic* and input-order invariant.
 - Considering the hierarchical nature of complex networks modeling real-world systems [218, 13], the effective clustering algorithm should be hierarchical. In particular, an agglomerative hierarchical clustering addresses also the efficiency criterion by reducing the number of processed items at each iteration, since each hierarchy level is built using clusters from the previous level directly.

Following the above requirements, DAOC¹ [142] is, to the best of our knowledge, the only parameter-free clustering algorithm that is simultaneously deterministic, input order invariant, robust (as it uses a consensus approach) and applicable to large weighted networks yielding a fine-grained hierarchy of overlapping clusters [140]. Moreover, it is based on a MMG meta-optimization function, where generalized modularity gain can be used as the target optimization function to perform clustering at the required resolution. However, DAOC *a*) yields a hierarchy of clusters only for a single value of the resolution parameter ($\gamma = 1$ operating with the non-generalized modularity), treating the hierarchy levels as scales (i.e., resolutions in terms of nested clusters rather than distinct values of γ) similar to [206, 17], and *b*) does not bound the size of the top (root) level of the forming hierarchy to produce the required number $t \leq d$ of clusters as described in Section 7.3.1. Therefore, we propose two extensions addressing these issues in the following.

7.4.1 Hierarchical multi-resolution clustering

Even though a multi-resolution structure of non-overlapping clusters is not necessary hierarchical [9, 122] (i.e., a node might be a member of several clusters that are not nested into each other), it can be represented as a *hierarchy of overlapping clusters* [126] (where the overlap addresses the case of a node shared by non-nested clusters). However, a hierarchical overlapping structure created with a single resolution may substantially differ from the respective multi-resolution structure (i.e., the produced clusters may differ vastly) as illustrated in Figure 7.2, where the strength on nodes interaction is represented with the width of the

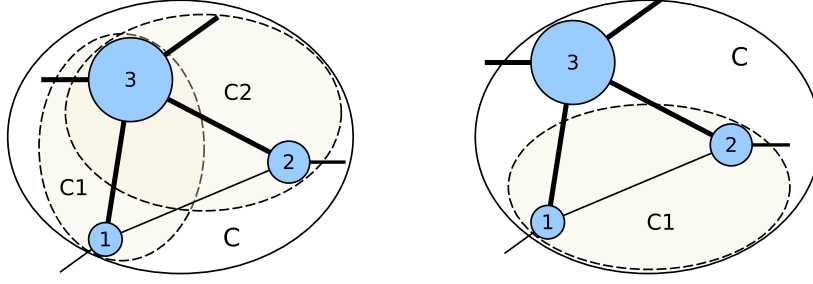


Figure 7.2 – A possible overlapping hierarchical clustering with the fixed resolution parameter of the weighed subgraph consisting of three nodes is shown on the left-hand side of the figure. A possible multi-resolution clustering for the same subgraph is shown on the right-hand side. The size of the nodes and the width of the links correspond to their weights.

links and the number of interactions with the size of the nodes (i.e., their weight). A large value of the resolution parameter can penalize heavily-weighted nodes according to Eq. (2.3), resulting in grouping together linked lightweight nodes. Such a behavior makes sense in many real-world cases, for example when employees working on the same project interact more frequently with their supervisor than between each other but the supervisor may not be a core of the group, participating also in other projects. Therefore, it is essential to incorporate the resolution parameter when generating the hierarchy levels, similar to [9, 194, 73].

In [9, 73], the authors operate with the analog of the resolution parameter called resistance parameter, which can not be transformed to the resolution parameter γ according to [8] and, hence, cannot be used with the generalized modularity defined in Eq. (2.3). In [194], the scale factor α is proposed, which can be directly transformed to the resolution parameter: $\gamma = (1 - \alpha)/\alpha$. However, the computational complexity of the proposed method is $O(n\sqrt{n}) \times O_0$ in average and $O(n^2) \times O_0$ in the worst case, where n is the number of nodes in the graph and O_0 is the complexity of the original clustering algorithm without the multi-scaling technique. This boosting of the computational complexity makes this technique inappropriate for large graphs. Hence, we propose our own approach to produce a hierarchy of multi-resolution clusters for large graphs based on DAOC.

The main idea behind our efficient multi-resolution hierarchical clustering is the dynamic variation of the resolution parameter γ during the hierarchy construction process. More precisely, DAOC is an agglomerative clustering algorithm, which builds the hierarchy bottom-up with micro-scale clusters on the bottom levels. These clusters should be constructed on a high value of the resolution parameter ($\gamma \gg 1$) according to Eq. (2.3). The resolution should then gradually decrease to the lowest value $\gamma > 0$ yielding macro-scale clusters located on the top (root) level of the hierarchy. The bounds for γ can be estimated based on the resolution limit analysis conducted in [61], where the equation is defined for the marginal case when all clusters have a perfect balance between the internal and external degrees being on the limit of detectability. That equation takes the following shape when adapted to the generalized

modularity:

$$\check{m} < \check{m}_{max} = \frac{m}{4\gamma}, \quad (7.2)$$

where \check{m} is the number of internal links in a cluster and m is the total number of links in the graph. To estimate the upper bound of γ , we need to bind \check{m} and m , which could be done relying on the two following heuristics. First, in case there is no prior information about the data, a rule of thumb is to take the estimated maximal number of clusters as the square root of the number of nodes [150]: $\tilde{k}_{max} = \sqrt{n}$. Then, considering that the number of internal links constitutes half of all the links of a cluster in the marginal case described by Eq. (7.2):

$$2m = 2\check{m} \times k, \quad k \lesssim \tilde{k}_{max} \implies m \lesssim \check{m}\sqrt{n}. \quad (7.3)$$

Second, most real-world systems are modeled by sparse graphs [13]. The number of links in a sparse graph does not exceed $n^{3/2}$. Thereby, Eq. (7.2) extended with Eq. (7.3) takes the following shape:

$$\check{m} < \frac{\check{m}\sqrt{n}}{4\gamma} \implies \gamma < \frac{\sqrt{n}}{4} \lesssim \frac{\sqrt[3]{m}}{4} \leq \frac{\sqrt[3]{w/\check{w}_{min}}}{4} = \gamma_{max}. \quad (7.4)$$

where \check{w}_{min} is the minimal weight of a link. Eq. (7.4) provides a definition for the upper bound of γ in a weighted graph. The lower bound of gamma is evaluated dynamically based on the optimal value of the resolution parameter given in Eq. (2.4) for each level of the hierarchy: $\gamma_{min} = \hat{\gamma}$. Typically, $\hat{\gamma} \geq 1$ for large real-world networks [145], so $\gamma_{min} = 0.5 \dots 1$ is taken for the first iteration before $\hat{\gamma}$ is evaluated.

In summary, γ is decreased from γ_{max} to γ_{min} with a fixed ratio r_γ and represents a trade-off between the number of captured resolutions when $r_\gamma \rightarrow 1$ versus a lower number of iterations (i.e., higher efficiency) and higher quality on coarse resolutions when $r_\gamma \rightarrow 0$. The quality of the forming macro-clusters is affected by the amount and size of the fine-grained clusters since the former are constructed iteratively from the latter, and the actual multi-resolution structure is not necessary hierarchical [9, 122]. In theory, r_γ should allow the growth of the cluster weight by a factor less than two to still retain the cluster semantics (super/sub-cluster ordering). Otherwise, the super/sub-cluster association is transformed into an overlapping clusters relation, losing the hierarchical structure. The limitation of the cluster weight growth to a factor 2 corresponds to $r_\gamma > 2^{-2} = 0.25$ according to Eq. (2.3), which is a hard theoretical bound. In practice, a larger value of r_γ is required considering the possibility of multiple mutually overlapping clusters as discussed in Section 7.3.1: $r_\gamma \gtrsim r_{w_{min}}^2 = 0.36$. The upper bound of r_γ corresponds to the lower bound of the cluster weight growth factor, which can be taken as 10 – 20% according to the Pareto principle [183, 20] or 10/90 gap [43]: $r_\gamma \leq 1.1^{-2} = 0.826$. Thus, the operational range of the gamma ratio is as follows: $r_\gamma \in [0.36, 0.826]$, and we pick $r_\gamma = 0.6$. This selected value is not supposed to be tuned by the users. Higher values of r_γ yield a larger number of salient clusters, which is not desirable, and lower values may cause the loss of some

important salient clusters. The exact theoretical r_γ depends on the number of overlapping clusters on the next iteration, which is generally speaking unknown.

7.4.2 Bounding the number of clusters

DAOC generates a hierarchy of clusters as long as the value of the optimization function is improving (i.e., $\Delta Q \geq 0$), which might result in any number of clusters at the top level of the hierarchy. If a specific number of clusters is required (e.g., for a fair comparison of various graph embedding techniques on the same number of dimensions), we propose the following extensions of the hierarchy generation process:

- The hierarchy generation is interrupted early if the number of clusters at level i , $|h_i|$ reaches the required number d .
- The hierarchy generation is forced to continue until the number of clusters reaches the required number d even if the value of the optimization function ΔQ becomes negative.

Algorithm 10 Hierarchical multi-resolution clustering with optionally bounded number of clusters.

```

1: function CLUSTER(nodes, d)
2:   rg  $\leftarrow$  0.6                                     ▷ Gamma ratio
3:   gamma  $\leftarrow \sqrt[3]{\text{weight}(\text{nodes})/\text{min\_weight}\ln(\text{nodes})/4}$ 
4:   gmin  $\leftarrow$  0.5                                     ▷ Min gamma
5:   hier  $\leftarrow$  []                                     ▷ List of the hierarchy levels
6:   while nodes do                                     ▷ Stop if the nodes list is empty
7:     cls  $\leftarrow$  daocHierLev(nodes, gamma, d)
8:     if gamma * (rg + 1) / 2  $\geq$  gmin and (not cls or count(cls)  $\leq \sqrt{\text{count}(\text{nodes})}$ ) then
      ▷ Decrease gamma
9:       gamma  $\leftarrow$  gamma * rg
10:    else
11:      nodes  $\leftarrow$  cls                                     ▷ Consider early termination
12:    end if
13:    if cls then                                     ▷ Initialize the next-level nodes
14:      hier.append(cls)                                     ▷ Extend the hierarchy
15:      gmin  $\leftarrow$  gammaOptim(cls)                         ▷ Update min gamma
16:      if not d or count(cls) > d then
17:        nodes  $\leftarrow$  cls                                     ▷ Update the processing nodes
18:      else
19:        nodes  $\leftarrow$  []                                     ▷ Early termination
20:      end if
21:    end if
22:  end while
23:  return hier                                     ▷ The resulting hierarchy of clusters
24: end function

```

In that case, the clustering objective becomes the minimal loss of the optimization function value: $\max \Delta Q < 0$.

The proposed extensions of the clustering algorithm are summarized in Algorithm 10. The early termination of the hierarchy construction process happens on lines 11, 19. The forced continuation of the hierarchy construction corresponds to a boolean parameter $d \neq 0$ for the original DAOC clustering on line 7. This parameter prevents the completion of the clustering process when the optimization function ΔQ cannot be further improved.

7.5 Experimental Evaluation

In this section, we conduct an extensive set of experiments to evaluate our proposed method on two typical graph analysis tasks, i.e., node classification and link prediction. We start by introducing our experimental setup including the datasets and baselines we use before presenting our experimental results.

7.5.1 Experimental Setup

Datasets

We conduct experiments on the following five graphs, which are widely used in the current literature for evaluating graph embedding techniques. Table 7.1 summarizes the key characteristics of the graphs.

Table 7.1 – Characteristics of the experimental graphs

tbl:daor:dataset	Blog	PPI	Wiki	DBLP	YouTube
Nodes	10,312	3,890	4,777	13,326	1,138,499
Links	333,983	76,584	184,812	34,281	2,990,443
Labels	39	50	40	2	47

- *BlogCatalog (Blog)* [229] is a social network of bloggers. Each node represents a user, while the labels of a node represent the topics the corresponding user is interested in.
- *Protein-Protein Interactions (PPI)* [77] is a graph of the PPI network for Homo Sapiens. The labels of a node refer to its gene sets and represent the corresponding biological states.
- *Wikipedia (Wiki)* [77] is a co-occurrence network of words appearing in a sampled set of the Wikipedia dump. The labels represent part-of-speech tags.
- *DBLP* [263] is a collaboration network capturing the co-authorship of writers. Each node represents an author, and the labels of a node refer to the publication venues of the corresponding author.
- *YouTube* [230] is a social network of users on YouTube. Each node represents a user, and the labels of a node refer to the groups (e.g., “anime”) that the corresponding user is interested in. This graph is used only to evaluate the efficiency of the embedding techniques, since the

ground-truth categories include only 3% of the graph (as opposed to a 100% coverage for the other graphs).

Baselines

We compare our proposed technique against ten state-of-the-art graph embedding techniques from three categories.

Graph-sampling based techniques **DeepWalk** [191], **Node2Vec** [77], **LINE** [228] and **VERSE** [238]. For DeepWalk and Node2Vec, we set the walk length to 40, the number of walks per node to 80, and the context window size to 10. For Node2Vec, we also tune the return parameter p and the in-out parameter q with a grid search over $p, q \in \{0.25, 0.05, 1, 2, 4\}$. For LINE, we set the total number of samples to 1 billion for Blog, PPI, Wiki and DBLP and to 10 billions for YouTube. For VERSE, we tune the damping factor α of personalized PageRank using the method suggested by the authors.

Factorization-based techniques **GraRep** [25], **HOPE** [179] and **NetMF** [198]. For GraRep, we search the optimal k over $\{1, 2, 3, 4, 5, 6\}$. When d/k is not an integer, we learn the first $k - 1$ sets of $\lceil d/k \rceil$ -dimension embeddings, and the last set of embeddings of dimension $d - (k - 1)\lceil d/k \rceil$. For HOPE, we search the optimal decay parameter β from 0.1 to 0.9 with a step of 0.2. For NetMF, we tune the implicit window size T within $\{1, 10\}$.

Similarity-preserving hashing based techniques **INH-MF** [134], **NetHash** [252] and **NodeSketch** [260]. For INH-MF, we set the ratio for subspace learning to 100%, to let it achieve optimal performance w.r.t. the quality of the learnt node embeddings. For NetHash, as suggested by the authors, we search the optimal tree depth in $\{1, 2, 3\}$. For NodeSketch, we search the optimal order of proximity k up to 6 and the optimal decay parameter α from 0.0001 to 1 on a log scale.

Meta techniques **HARP** [34]. We configure HARP to learn from the embeddings of DeepWalk (HARP-DWalk) and LINE (HARP-LINE) using the following parameter settings. For HARP-DWalk, we set the walk length to 10, the number of walks per node to 40, the context window size to 10 and the sampling ratio to 0.1. For HARP-LINE, we set the context window size to 1, the number of iterations to 50 and the sampling ratio to 0.001.

7.5.2 Node Classification Task

Node classification tries to predict the most probable label(s) for some nodes based on other labeled nodes. In this experiment, we consider a multi-label setting, where a node is assigned

Table 7.2 – Node classification performance using kernel SVM, where the top-3 results for each dataset are highlighted with bold numbers

Method	Micro-F1 (%)				Macro-F1 (%)			
	Blog	PPI	Wiki	DBLP	Blog	PPI	Wiki	DBLP
·DeepWalk	39.60	17.24	46.05	83.46	21.93	10.28	6.62	83.16
·Node2Vec	37.95	16.04	50.32	93.25	20.22	9.57	9.86	93.12
·LINE	35.49	15.01	48.22	86.83	16.60	8.70	8.47	86.54
·VERSE	39.61	15.90	41.39	92.79	22.85	9.76	4.14	92.66
·GraRep	36.21	5.83	56.22	91.41	16.91	1.52	12.14	91.25
·HOPE	31.37	14.69	56.68	91.47	11.74	8.13	13.30	91.30
·NetMF	40.04	15.03	57.62	93.59	23.43	8.74	14.35	93.46
·INH-MF	36.13	15.50	45.03	93.27	18.88	9.55	6.90	93.16
·NetHash	35.80	18.85	47.57	97.61	18.72	12.91	8.05	97.57
·NodeSketch	38.16	21.04	59.07	98.83	21.84	15.55	16.31	98.81
·HARP-DWalk	36.52	15.46	43.06	92.66	19.56	9.04	5.59	92.53
·HARP-LINE	30.27	12.67	42.79	88.07	13.06	6.25	5.38	87.84
DAOC	21.3	12.56	42.43	89.24	6.47	7.25	5.66	89.03
DAOR	33.05	19.07	53.24	87.86	17.25	13.94	15.97	87.64

· the algorithm meta parameters are *tuned* once for all datasets to maximize accuracy
: the algorithm parameters are *tuned for each dataset* to maximize accuracy

one or multiple labels. Our evaluation was performed using an open-source graph embeddings evaluation framework, GraphEmbEval³, which uses the same evaluation scheme as in [191, 77, 25]. More precisely, we randomly pick a set of nodes as labeled nodes for training, and use the rest for testing. To fairly compare node embeddings with different similarity measures, we train a one-vs-rest kernel SVM classifier with a pre-computed kernel (cosine, Jaccard or Hamming kernel according to the embedding techniques) to return the most probable labels for each node. We report the average Macro-F1 and Micro-F1 scores from 5 repeated trials. A higher value of these metrics implies better performance.

Our method, DAOR, shows competitive results without requiring any tuning (unlike conventional embedding techniques, which require extensive tuning, as described above). We also evaluated embeddings generated using DAOC without our proposed extension for multi-resolution clustering. The improvement of DAOR over DAOC verifies the effectiveness of our proposed extensions for graph embedding.

7.5.3 Link Prediction Task

Link prediction is a typical graph analysis task that predicts potential (or missing) links between nodes in a graph. For this task, we use the same setting as in [179]. Specifically, we randomly sample 20% of the links out of the graph as test data, and use the rest of the graph for training.

³<https://github.com/eXascaleInfolab/GraphEmbEval>

Chapter 7. DAOR: Fully Automatic and Interpretable Graph Embedding via Clustering

Table 7.3 – Link prediction performance, where the top-3 results for each dataset are highlighted with bold numbers

Method	Precision@100				Recall@100			
	Blog	PPI	Wiki	DBLP	Blog	PPI	Wiki	DBLP
·DeepWalk	0.0200	0.0159	0.0090	0.0423	0.0301	0.2227	0.0493	0.6749
:Node2Vec	0.0927	0.0137	0.0267	0.0321	0.1378	0.1958	0.1514	0.5174
·LINE	0.0070	0.0073	0.0031	0.0392	0.0103	0.0923	0.0167	0.6186
·VERSE	0.0404	0.0206	0.0212	0.0436	0.0602	0.2723	0.1118	0.6906
:GraRep	0.0014	0.0011	0.0054	0.0001	0.0020	0.0118	0.0286	0.0011
:HOPE	0.0023	0.0073	0.0027	0.0248	0.0035	0.0960	0.0149	0.4034
:NetMF	0.0175	0.0174	0.0084	0.0218	0.0266	0.2287	0.0474	0.3126
·INH-MF	0.0158	0.0158	0.0084	0.0252	0.0227	0.2209	0.0454	0.4052
:NetHash	0.0015	0.0134	0.0020	0.0387	0.0022	0.1899	0.0101	0.5958
:NodeSketch	0.0729	0.0250	0.0176	0.0462	0.1080	0.3331	0.0942	0.7595
·HARP-DWalk	x	0.0142	0.0101	0.0407	x	0.1978	0.0536	0.6459
·HARP-LINE	x	0.0026	0.0021	0.0309	x	0.0331	0.0117	0.5029
DAOC	0.0001	0.0099	0.0059	0.0027	0.0001	0.1301	0.0314	0.0444
DAOR	0.0958	0.0175	0.0164	0.0032	0.1438	0.2345	0.0892	0.0548

- the algorithm meta parameters are *tuned* once for all datasets to maximize accuracy
- : the algorithm parameters are *tuned for each dataset* to maximize accuracy
- x the algorithm is crashed on coarsening small disconnected components

After learning the node embeddings based on the training graph, we predict the missing links by generating a ranked list of potential links. For each pair of nodes, we use the cosine, Jaccard or Hamming similarity (according to the embedding techniques) between their embedding vectors to generate the ranked list. As the number of possible pairs of nodes is too large, we randomly sample 0.1% pairs of nodes for evaluation. We report the average precision@N and recall@N from 10 repeated trials.

Table 7.3 shows the results of the link prediction task. Our proposed method, DAOR, is among the top-3 best-performing techniques being unsupervised and parameter-free. The impact of our proposed multi-resolution clustering is especially visible on this task, where DAOR significantly outperforms DAOC.

7.5.4 Robustness to the Metric Space

Node embedding robustness to different metric spaces is shown in Table 7.4 on the node classification task for our method DAOR versus the two other best-performing methods from Table 7.2 that technically can be evaluated with another metric space (NodeSketch is omitted because it uses a non-linear Hamming metric space, where cosine distance cannot be formally evaluated). For *all* input graphs, DAOR shows the best worst-case performance, i.e., DAOR yields much more accurate results in its least accurate non-primary metric space than the other methods do. Moreover, Hamming distance is directly applicable to DAOR without any

preliminary binarization, unlike the algorithms operating in the cosine metric space.

Table 7.4 – Node embedding robustness to the metric space, where the native metric space for each algorithm is highlighted in bold

Method	Metric	Micro-F1 (%)			
		Blog	PPI	Wiki	DBLP
DAOR	jaccard	33.05	19.07	53.24	87.86
	cosine	30.41	13.62	47.20	87.42
	hamming	23.87	14.13	45.52	86.98
	binham	23.89	10.25	41.36	87.17
NetMF	cosine	40.04	15.03	57.62	93.59
	hamming	17.54	6.55	40.93	70.32
	binham	19.82	7.05	42.85	74.91
HOPE	cosine	31.37	14.69	56.68	91.47
	hamming	17.02	5.95	40.89	59.94
	binham	18.23	6.21	40.89	74.36

binham - Hamming metric applied after a binarization of each dimension using its median value

7.5.5 Runtime Performance

In this experiment, we investigate the efficiency of the graph embedding learning process. Our evaluation was performed using an open-source graph embeddings evaluation framework,

Table 7.5 – Node embedding learning time (in seconds), where the top 3 results for each dataset are highlighted in bold

Method	Blog	PPI	Wiki	DBLP	YouTube
DeepWalk	3375	1273	1369	4665	747060
Node2Vec	1073	383	1265	504	-
LINE	2233	2153	1879	2508	29403
VERSE	1095	203	276	1096	245334
GraRep	3364	323	422	10582	-
HOPE	239	100	78	283	15517
NetMF	487	124	708	213	-
INH-MF	509	39	98	378	-
NetHash	721	201	134	35	12708
NodeSketch	70	8	17	8	2439
HARP-DWalk	1436	299	483	958	336200
HARP-LINE	1274	106	189	90	13951
DAOC	7.9	0.3	1.8	0.2	4893
DAOR	1.6	0.2	0.4	0.2	57.1

- the algorithm was terminated by timeout

GraphEmbEval³, on a Linux Ubuntu 16.04.3 LTS server with an Intel Xeon CPU E5-2620 v4 @ 2.10GHz CPU (16 physical cores) and 132 GB RAM. The training and execution termination constraints for each algorithm were set to 64 GB of RAM and 240 hours CPU time (we terminate the process when either of those thresholds are met).

Table 7.5 shows the end-to-end embedding learning time. To discount the impact of the multi-threaded implementation of some of the methods, we dedicate a single logical CPU per each method implementation and report the total CPU time. Our method, DAOR, is faster than existing state-of-the-art techniques by several orders of magnitude; it exhibits near-linear scaling when increasing the number of links in the graph. DAOR is also much more scalable than DAOC (on which DAOR is built) due to its specific multi-scaling approach that boosts the number of nodes reaching consensus of the optimization function early on lower levels of the hierarchy. Moreover, unlike other graph embedding techniques, DAOR execution time decreases when increasing the number of embedding dimensions, which is due to the early termination of the clustering as described in Section 7.4.2.

7.6 Conclusions

In this chapter, we presented a novel highly efficient and parameter-free graph embedding technique, DAOR², which produces metric-robust and interpretable embeddings without requiring any manual tuning. Compared to a dozen state-of-the-art graph embedding algorithms, DAOR yields competitive results on diverse graph analysis tasks (node classification and link prediction), while being several orders of magnitude more efficient.

In future work, we plan to recommend the minimal, maximal and optimal number of embedding dimensions, and conduct a comprehensive study on their quality and interpretability. Also, we plan to integrate further state-of-the-art community detection algorithms in addition to DAOC.

8 Conclusions

We conclude this dissertation by discussing the main findings and contributions of our research along with potential extensions, implications and future research directions. The latter constitute our outlook on the potential evolution of fully automatic data analytics in order to assist humans in decision making.

8.1 Summary of Findings

We first recall that one of the key challenges of transforming Big Data into actionable knowledge is the identification of transformations that can efficiently generate a human perception-adapted representation. This challenge was decomposed in Chapter 1 and Chapter 2 into formal and measurable research objectives that were addressed in Chapters 3-7. All our research findings are implemented as open-source applications and are available freely as specified in their respective chapters.

We started addressing the formulated research objectives in Chapter 3 by introducing our novel clustering algorithm, DAOC, conforming to the requirements for information structuring as formulated in Chapter 2. DAOC is at the same time stable and provides a unique combination of features yielding a fine-grained hierarchy of overlapping clusters in a fully automatic manner. We experimentally compared our approach on a number of different datasets and showed that while being parameter-free and efficient, it yields accurate and stable results on any input networks (i.e., graphs specified by pairwise relations). In particular, DAOC is on average 25% more accurate than state-of-the-art deterministic algorithms while not requiring any tuning. DAOC builds on a new (micro) consensus technique, MMG, and a novel overlap decomposition approach, OD, which are both applicable on top of non-overlapping clustering algorithms and allow to produce overlapping and robust clusters. DAOC is released as an open-source clustering library implemented in C++, which includes various cluster analysis features that greatly simplify and speed up data analysis tasks involving iterative processing as well as data fluctuations to provide accurate and reproducible results. Several data mining applications were built on DAOC, including StaTIX [143] presented in Chapter 6 and DAOR [144] introduced

in Chapter 7.

Subsequently, we turned our attention to the performance evaluation of the proposed information representation approach, namely, of our clustering technique. Clustering accuracy measures were discussed in Chapter 4, taking into account both a) the properties of the clustering algorithm and b) the formal constraints for the measures themselves (see Section sec:clsobj). In particular, we discussed the state-of-the-art accuracy metrics applicable to overlapping clustering evaluations on large datasets and introduced several optimizations and extensions of the discussed metrics. More specifically, we introduced an efficient indexing technique to speedup the evaluation of Mean F1 score from the near-linear to linear on the number of items in the clustering. We proposed an adaptive sampling strategy for GNMI to speedup the evaluation by orders of magnitude while improving the metric precision. In addition, we proposed two extensions of the Average F1 score (F1a), namely F1h and F1p, to address the poor performance of F1a in the range $[0, 0.5]$ while improving the satisfaction of the formal constraints and without sacrificing efficiency. We also proposed an extension of the Omega Index called Soft Omega Index, which is equivalent to the original Omega Index evaluating non-overlapping clusterings and yields more discriminative results for overlapping ones. Also, we discussed formal constraints for each metric, as well as their applicability and limitations for specific cases in overlapping and multi-resolutions clustering. Our analysis on metrics provides additional insight for their future use and should hopefully help identify the best performing clustering algorithm for particular user's needs or tasks.

In Chapter 5, we introduced both a) a benchmarking methodology for various clustering algorithms that addresses the biases in the clustering results, and b) our open-source benchmarking framework, Clubmark. Clubmark is an isolating benchmarking platform for clustering evaluation on a wide range of synthetic and real-world datasets that runs on NUMA servers. It allows for fine-grained control over various execution variables (timeouts, memory consumption, CPU affinity and cache policy) and supports the evaluation of a wide range of clustering algorithms including multi-level, hierarchical and overlapping clustering techniques on both weighted and unweighted input networks (i.e., graphs) with built-in evaluation of several extrinsic and intrinsic measures. Our framework is open-source and provides a consistent and systematic way to execute, evaluate and profile clustering techniques considering a number of aspects that are often missing in state-of-the-art frameworks and benchmarking systems.

Finally, we turned to a number of applications of our proposed approach for information representation in the context of data mining tasks in Chapters 6 and 7. In particular, in Chapter 6, we introduced a new statistical type inference method, called StaTIX, to infer instance types in Linked Data (e.g., Knowledge Graphs) in a fully automatic manner without requiring any prior knowledge about the dataset. This method is based on our clustering technique, DAOC, which infers (overlapping) types in a robust and efficient manner as discussed in Chapter 3. As part of StaTIX, we also presented novel techniques to a) reduce the similarity matrix representing relationships between the instances, b) speed up the clusters formation using a dedicated, history-independent hash, AggHash, and c) identify representative clusters at

multiple scales. We empirically compared our approach on a number of different datasets and showed that it is at the same time considerably more effective and orders of magnitude more efficient than state-of-the-art techniques.

In Chapter 7 we presented a novel, highly efficient and parameter-free graph embedding technique, DAOR, based on our specific clustering technique, DAOC, described in Chapter 3. DAOR produces metric-robust and interpretable embeddings without requiring any manual tuning. Compared to a dozen state-of-the-art graph embedding algorithms, DAOR yields competitive results on diverse graph analysis tasks (node classification and link prediction), while being several orders of magnitude more efficient. In the scope of DAOR, we: a) proposed a new mechanism to generate node embeddings from a given hierarchy of clusters, b) identified the key features of community detection (i.e., clustering) algorithms for the efficient construction of effective node embeddings, and c) proposed efficient extension of the community detection algorithms to form communities that effectively represent embedding dimensions.

8.2 Future Work

8.2.1 Outlook

Living in an Information Age and facing a rapid increase in the amount of information that is exchanged, we envision an ever greater automation of information processing. This also implies a proliferation of unsupervised, weak- and semi-supervised data mining processes. Weak- and semi-supervised techniques typically rely on clustering techniques [32, 279], which belong to the field of unsupervised machine learning. Therefore, we expect data clustering to become one of the fundamental components of future data processing pipelines. However, to be widely useful in practice, a clustering method should meet the criteria outlined in Table 2.1 of Chapter 2 besides just being an unsupervised technique. In particular, crucial characteristics for the assistance of human decision making are *interpretability* (i.e., what does each piece of the constructed results mean and how does it confirm the existing relationships in the information being processed) and *explainability* (i.e., why are such results generated and not others, which can be seen as mixing interpretation, formal inference and prediction¹) of the automated process, which also involves the stability of the produced results: “Nothing can be improved upon unless it’s stable and you’re able to understand what might be influencing the results”². In this thesis, we tackled a number of technical aspects of data clustering to facilitate its anticipated ubiquitous integration into diverse data processing systems.

In the rest of this section, we first outline some limitations of our research and propose approaches to address them in Section 8.2.2. In Section 8.2.3, we discuss possible extensions of DAOC to process attributed and hyper-graphs. Then, we propose a novel visualization technique based on our clustering method and briefly discuss how to enhance state-of-the-

¹<https://medium.com/inventing-intelligent-machines/is-strong-ai-inevitable-f4ed58c05293>

²<https://spectrum.ieee.org/the-institute/ieee-member-news/ten-on-tech-spotlight-on-christopher-sanderson>

art Visual Analytics systems through our interactive visualization in Section 8.2.4. Finally, in Section 8.2.5, we give a brief outlook on the social impact of automatic data processing systems that assist human decision making, focusing on information search and recommendation systems. We propose extending the latter systems with the approaches presented in this thesis, aiming at providing awareness to the user in response to his/her information queries and, thus, reducing social polarization.

8.2.2 Addressing the Limitations of our Research Findings

The research conducted in the scope of this thesis presents a number of trade-offs and limitations, which can be addressed in future work as follows.

Our *DAOC clustering* algorithm has a near-linear execution time on sparse input networks (i.e., graphs specified by pairwise relations) but up to a quadratic run-time complexity on dense networks, which can be significantly improved using approximated evaluations as we discuss in Section 8.2.3. In addition, we plan to parallelize DAOC taking advantage of modern hardware architectures to further expand the applicability of our method. Also, it is worth noting about the common limitation of network representation, which also affects the applicability of DAOC; namely, the “*curse of dimensionality*” may occur when multidimensional data is transformed into a network representation. However, this issue can be partly overcome as outlined in Section 8.2.3, where we discuss DAOC extensions for its applicability on attributed graphs and hypergraphs.

Accuracy metrics in our *Xmeasures framework* are implemented for crisp overlaps but not for the fuzzy overlaps. Fuzzy overlaps could however be supported by introducing weight attributes when forming node fragments instead of using a node sharing degree that is uniform for each resulting fragment.

Our *Clubmark benchmarking framework* is designed to run on a single NUMA server but can be extended in future work to support federated benchmarking in a cluster of machines in order to speedup the benchmarking of a large number of algorithms. We note that using identical machines in the cluster is essential in this context to ensure fairness of the evaluation, since the execution time of the evaluated applications depends on many facets of the hardware architecture (e.g., CPU, memory and storage frequencies; CPU and storage cache sizes; CPU instruction set). Clubmark relies on the LFR framework [123] for generating synthetic networks with ground-truth clusters, but the latter is limited in generating hierarchical clusters compared to RB-LFR [265] when overlaps are not essential. Thus, we are going to integrate also RB-LFR framework into Clubmark in the future.

Our *StaTIX framework for statistical type inference on Linked Open Data (LOD)* suffers from an execution time bottleneck when forming its input matrix, which could be avoided with the direct processing of attributed graphs as we propose in Section 8.2.3. In addition, StaTIX can be extended with additional semantic analysis leveraging both logical reasoning and embedding

techniques to better grasp the differences and relationships between various instances. Also, it would be beneficial in this context to add support for automatically borrowing type labels from third-party knowledge bases whenever available. In terms of implementation-specific aspects, it would also make sense to parallelize our algorithm to take advantage of modern multi-core CPU architectures.

Our *DAOR graph embedding framework* provides very high performance for the evaluation of moderate-size graphs (up to millions of nodes and dozens millions links) when embedding them into high-dimensional spaces (i.e., larger than 100 dimensions), but would perform poorly for low-dimensional spaces like 2D or 3D spaces. The latter can be addressed by replacing one dimension (which is currently redundant since its value is equal to the total weight of the node subtracted by the values of all other dimensions of this node) with some metainformation that identifies the node position in the hierarchy of clusters or in the input graph. This metainformation should be in a form of a real value and can be evaluated, for example, as a node centrality, special locality-sensitive indexing of the hierarchy (e.g., based on arithmetic coding) or approximated locality-sensitive hash of the node position in the resulting hierarchy. In addition, a comprehensive study can be conducted on the minimal, maximal and optimal number of embedding dimensions by DAOR, their quality and interpretability. Our DAOR framework could be extended to rely on further state-of-the-art clustering algorithms besides DAOC: Louvain [17], Leiden [237] or HDBSCAN* [156]. Finally, we propose integrating our DAOR/DAOC method into modern search and recommender systems to improve the diversity of the results delivered as well as improve their exploration, having the ambition to reduce social polarization as discussed in in Section 8.2.5.

8.2.3 DAOC Extension for Attributed and Hyper-Graphs

Real-world data often has multiple attributes (e.g., a multi-dimensional point in space, or a product having multiple characteristics) that can naturally be stored as an attributed graph. Any attributed graph can be converted to a respective graph with pairwise relations (i.e., a network) and vice versa, however such a conversion is resource-intensive, lossy and ambiguous as we mentioned in Section 2.2 of Chapter 2. Moreover, incremental updates of the input data are most efficient on the original format. Therefore, it is desirable to support both input graphs specified by pairwise relations, as well as by attributes when processing Big Data.

Data items are compared to each other when being clustered, which requires to identify pairwise relations anyway. However, the number of all possible pairwise relations between items is quadratic, which yields both computational and memory bottlenecks for large datasets. Moreover, many if not most of the relations have no significant impact when being clustered and can be pruned as shown in Chapter 6. Thus, the question arises of how to prune insignificant relations before actually evaluating them and constructing a sparse graph specified by pairwise relations on the fly from the attributed data. In this context, one needs to with approximate evaluations and sampling strategies. Among the most effective existing methods

applicable to large attributed datasets are:

- NN-Descent [48] constructing k -nearest neighbor graph for generic similarity measures, which has an empirical computational complexity of $O(n^{1.14})$;
- HNSW [147] performing efficient and robust approximate nearest neighbor (ANN) search using Hierarchical Navigable Small World graphs, having an empirical computational complexity of $O(n \cdot \log n)$;
- RobustIQ [37] (based on FAISS [97]) performing a robust ANN search of billion-node graphs on GPUs; and
- NSG [66] approximating Monotonic Relative Neighborhood Graph for k -nearest neighbor search, having a computational complexity of $O(k \cdot n^{(1+d)/d} \log n^{1/d} + n^{1.16})$ for the initial graph indexing and $O(n^{1/d} \log n^{1/d})$ for the nearest neighbors search itself.

It is worth noting that sampling strategies are also useful to reduce the computational complexity when clustering dense networks given that the approximate results are sufficient.

However, mixing multiple attributes when constructing a pairwise similarity relation between a pair of items raises two conceptual issues: 1) some required homogeneity for the item relations when clustering and 2) the curse of dimensionality. Homogeneity of network links does not imply that the similarity space should be metric or the links should be constructed using exactly the same attributes, but that a *consistent semantic meaning* should be reflected which the relation strength (i.e., link weight): real-world networks have a property called assortativity, or assortative mixing, which is the tendency of network nodes to attach to other nodes that are similar in *some* way [170]. The curse of dimensionality is the problem of exponential growth of the volume space with the number of dimensions. Thus, a network should not be directly constructed based on the multi-dimensional input data, instead a hyper-network (hyper-graph) should be constructed for such data, overcoming the curse of dimensionality. Then, to cluster a hypergraph either a) a subset of the links having their types corresponding to the user intention (i.e., user query, which defines the clustering objective) can be fetch and converted to an ordinary network, or b) the links can be reweighted on the fly and pruned as proposed in Chapter 2. Anyway, the conversion to the respective flat network requires reweighting the links proportionally to their correlation with the user intent.

8.2.4 Visual Exploration and Analytics Based on a Hierarchy of Clusters

Typically, a hierarchy of clusters can be visualized using folding trees (e.g., dendrograms and tree maps) [68, 84]. Compared to dendrograms, threemaps includes some visualization of the cluster size. They are however less expressive and not that convenient for interacting with large hierarchies. Moreover, none of these approaches is directly suitable for the visualization of overlapping clusters or non-hierarchical relations between the clusters and the nodes of

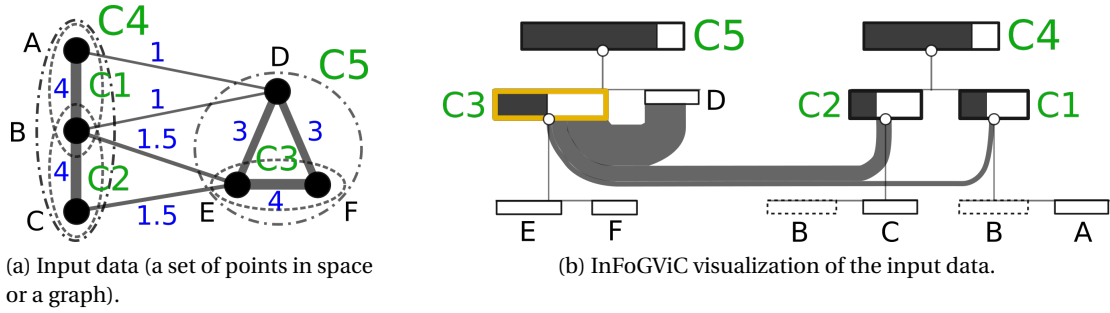


Figure 8.1 – InFoGVic visualization for a hierarchy of nested clusters with overlap(s).

the input network. To address these shortcomings, we propose to build InFoGVic (spel.: in fog we see), an Interactive Folding Graph-based Visualization with Clustering, which is inspired by folding trees [193], clutter reduction techniques [177, 52], approaches for relations visualization (e.g., Arc Diagrams [249] and In-Line Coordinates [111]). A high-level example of this visualization technique for nested clusters with overlap(s) is shown in Figure 8.1 and builds on the following principles:

- Each item (i.e., a node of the input network or a cluster of nodes) is visualized with a *rectangle*: non-leaf nodes of the hierarchy (i.e., clusters) are visualized with a higher rectangle than leaf nodes (i.e., nodes of the input network).
- Visualizing Rectangles:
 - The *filled* part of each rectangle corresponds to the internal weight of the respective item (i.e., the aggregated weight of the internal links) while the remaining part corresponds to the weight of the relations to other items on the same hierarchical level (i.e., the aggregated weight of the external links);
 - The relative *size* of the rectangle corresponds to the item weight with optional scaling (e.g., logarithmic) along both levels of the hierarchy;
 - The rectangles are *ordered*³ with respect to the item centrality in the respective hierarchical level, and, optionally considering some further attributes⁴ (e.g., vector length for a point in space, or total weight for more abstract item);
 - Optionally, the standard deviation (SD) of the specified links (e.g., external, internal, or those being visualized only) can be visualized by morphing the original rectangle into *trapeze* on the bottom, where the morphing ratio is $1 / \text{SD}$.
- Hierarchical relations are visualized via lines attaching all descendant items (i.e., sub-clusters). When the number of descendant items is larger than 10, only the heaviest of

³The order of the items is essential for the interpretation of the whole visualization as explained below.

⁴Additional ordering criteria might be required for the deterministic visualization of symmetric shapes to preserve the mental map (see Chapter 2.1).

them are visualized by default, and the omitted ranges are replaced with the unfoldable “...” placeholder.

- Most of the item relations are visualized only on-demand (i.e., when selecting the required items and specifying to visualize their mutual, in-level or all relations) to prevent excessive cluttering:
 - Relations between the items located on the same hierarchical level and belonging to the same cluster;
 - Indirect relations between some items and the clusters of its neighbour (i.e., linked) items;
 - Only the *significant* (also known as salient or representative) relations as defined in Chapter 6 are visualized by default, which can be optionally changed to a) only the *strongest*⁵ or b) *all* relations.
- Items belonging to an overlap are visualized as singletons attaching to their first unfolded ascendant.

This proposed visualization is scalable to arbitrary large datasets due to its hierarchical nature, while allowing the inspection of fine-grained relations, hence addressing the shortcomings of state-of-the-art visualization methods as shown in Figure 8.2. For example, the containment in a (multidimensional) space is preserved by InFoGVic being visually identifiable as a specific

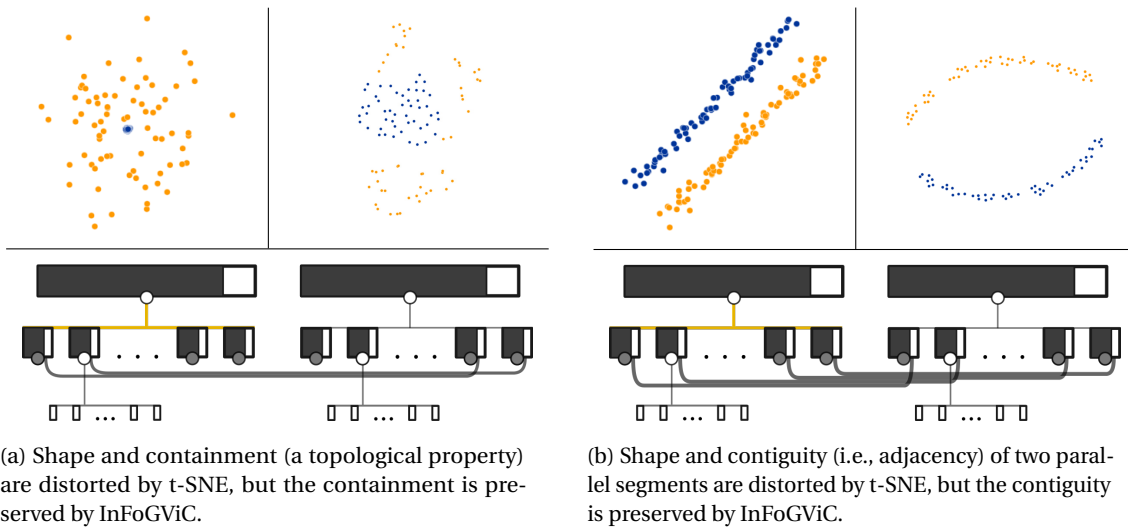


Figure 8.2 – Visualization pitfalls of t-SNE being solved with InFoGVic, where the original data is displayed on the top-left hand side, the visualization with t-SNE on the top-right hand side (as shown in Figure 2.8) and the respective InFoGVic visualizations are on the bottom.

⁵The inspection of the strongest relations is often convenient to analyze topological properties of the visualizing data (i.e., input graph) as discussed below.

pattern of the strongest external relations between the contained and encapsulating clusters as shown in Figure 8.2a. Namely, the containment is visualized as the strongest external relations of the contained cluster's items having the lowest centrality (i.e., located at the end of the respective unfolded cluster) to the items of the encapsulating cluster having the largest centrality (i.e., located at the begin). Moreover, typically, such clusters are grouped into a single super-cluster reflecting their compositionality (i.e., containment in this case). Similarly, InFoGVic preserves contiguity, as it visualizes two parallel segments as the strongest external relations being mutual and pairwise from the beginning to the end of the level of each cluster as it shown in Figure 8.2b.

We conclude this section by proposing IntEVA, an integrated framework for visual analytics (VA) of Big Data addressing the shortcomings of state-of-the-art solutions as follows. We propose to visualize data with both lossy, lossless and InFoGVic techniques in the same framework in a way that items (e.g., points) selected in any of these visualizations are also selected

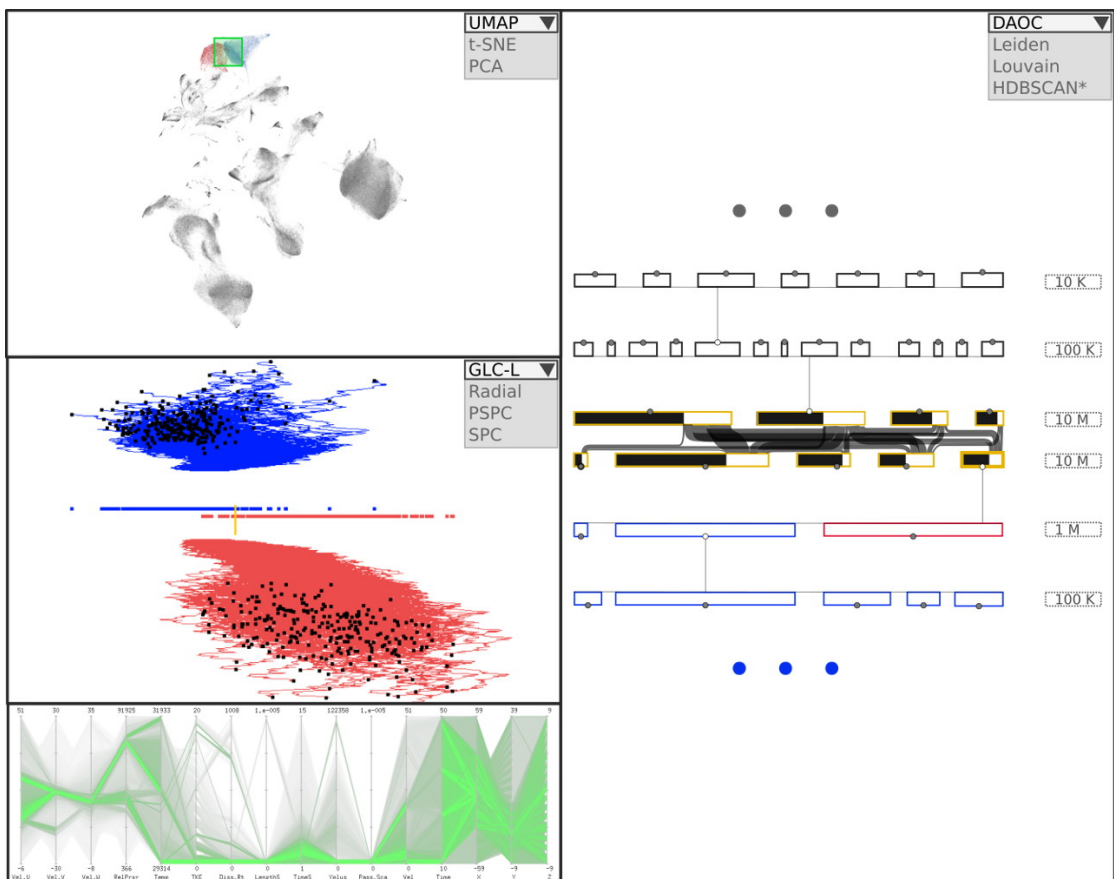


Figure 8.3 – IntEVA framework for VA of Big Data, providing seamless integration of lossy, lossless and InFoGVic visualization techniques for in-depth multimodal data analysis. The numbers on the right of the InFoGVic view correspond to the weight scales of the visualized items.

in others. This integration provides the possibility of in-depth multi-modal inspection as shown in Figure 8.3, addressing the shortcomings of each visualization technique by taking advantage of the others. In particular, most cultures first look at the top left corner of an image, where the coarse-grained summary of the whole dataset is shown with a lossy techniques (i.e., UMAP, t-SNE or PCA, which are implemented in Embedding Projector⁶ of TensorFlow). Any confusing parts of the visualization (i.e., diffusing groups of points) can be selected for further inspection with InFoGVic (located on the right) by iteratively resolving most of the compositionality-related concerns using the drill-down capabilities of our proposed hierarchical visualization. Specific items, for which the original (multi-dimensional) relations ought to be inspected, are selected in the InFoGVic view and studied with a lossless visualization (i.e., GLCs including Parallel Coordinates) located on the bottom left side. Each selected item can be augmented with the visualization of the original object (e.g., an image or a textual serialization of the object) in the upper visualization layer as performed by Divvy [132]. Thus, our proposing hybrid visualization framework, IntegVA, aims to simplify, speed up and provide more in-depth capabilities for visual analytics of Big Data by seamless integration of complementary state-of-the-art VA techniques extended with our proposed visualization, InFoGVic.

8.2.5 Social Impact of Fairness and Diversity in Recommendation Systems

Modern search services are based on the idea of delivering relevant and popular content in response to a user query. However, these services often lack explicit content segregation and do not account much for content diversity in the delivered results. These two flaws often create so-called information *filter bubbles* isolating people from a diversity of viewpoints [184], whereas truth discovery ought to deal with multiple sources of information considering their completeness, accuracy and reliability [133]. In other words, users of a modern search and recommender systems (including most of the social network services) mostly receive information that already complies with the user preferences. The propaganda spreading via social networks and the social polarization are to a large degree caused by such incomplete information [38]. These issues could be tackled by providing multi-perspective summaries to user queries using drill-down interface (e.g., relying on our method, DAOC, as discussed in Section 8.2.4) to easily perceive distinct viewpoints on the same event and conveniently analyzing them. Such an approach has the ambition to decrease social polarization by conveying information for each viewpoint in an intuitive and explorative way. Unlike modern state-supported censorship systems, this approach does not restrict or control information from any single side and provides comprehensive and intuitive information in response to user queries.

⁶<https://projector.tensorflow.org/>

Bibliography

- [1] K. Aberer, A. Boyarsky, P. Cudré-Mauroux, G. Demartini, and O. Ruchayskiy. Sciencewise: A web-based interactive semantic platform for scientific collaboration. In *ISWC 2011 - Demo*, 2011.
- [2] Y. Ahn and Y. Lin. Fairsight: Visual analytics for fairness in decision making. *IEEE T Vis Comput Gr*, 26(1):1086–1095, Jan 2020.
- [3] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, and L. Liberti. Column generation algorithms for exact modularity maximization in networks. *Phys. Rev. E*, 82:046112, Oct 2010.
- [4] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retr.*, 12(4), Aug. 2009.
- [5] S. E. Amiri, B. Adhikari, A. Bharadwaj, and B. A. Prakash. Netgist: Learning to generate task-based network summaries. In *ICDM*, pages 857–862, Nov 2018.
- [6] Annys, Shin. Wikipedia was born in 2001. and the world got a bit truthier, 2017.
- [7] D. Archambault and H. C. Purchase. Mental map preservation helps user orientation in dynamic graphs. In W. Didimo and M. Patrignani, editors, *Graph Drawing*, pages 475–486. Springer, 2013.
- [8] A. Arenas, A. Fernandez, and S. Gomez. Analysis of the structure of complex networks at different resolution levels. *New J. Phys.*, 10(5):053039, 2008.
- [9] A. Arenas, A. Fernández, and S. Gómez. Multiple resolution of the modular structure of complex networks. *New J. Phys*, 10, 2008.
- [10] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735, 2007.
- [11] F. Badjio and F. Poulet. Dimension reduction for visual data mining. In *Proceedings of International symposium on applied stochastic models and data analysis*, ASMDA, 2005.
- [12] N. Banovic and J. Krumm. Warming up to cold start personalization. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4):124:1–124:13, Jan. 2018.
- [13] A.-L. Barabási and M. Pósfai. *Network science*. Cambridge Press, 2016.
- [14] D. S. Bassett, M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson, and P. J. Mucha. Robust detection of dynamic community structure in networks. *Chaos*, 23(1):013142, 2013.
- [15] J. Battelle. *The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture*. Portfolio, 2005.
- [16] T. Berners-Lee, R. Cailliau, J. Groff, and B. Pollermann. World-wide web: The information universe. *Electronic Networking: Research, Applications and Policy*, 2(1):74–82, 1992.

Bibliography

- [17] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J Stat Mech.*, 2008(10):P10008, oct 2008.
- [18] S. P. Borgatti and M. G. Everett. Models of core/periphery structures. *Social Networks*, 21(4):375 – 395, 2000.
- [19] M. A. Borkin. *Perception, Cognition, and Effectiveness of Visualizations with Applications in Science and Engineering*. PhD thesis, Harvard University, 2014.
- [20] G. E. Box and R. D. Meyer. An analysis for unreplicated fractional factorials. *Technometrics*, 28(1):11–18, 1986.
- [21] M. Brusco and D. Steinley. A comparison of heuristic procedures for minimum within-cluster sums of squares partitioning. *Psychometrika*, 72(4):583–600, 2007.
- [22] L. Bühmann, J. Lehmann, and P. Westphal. DL-learner - A framework for inductive learning on the semantic web. *J. Web Sem.*, 39:15–24, 2016.
- [23] R. A. Burkhard. *Knowledge Visualization: The Use of Complementary Visual Representations for the Transfer of Knowledge. A Model, a Framework, and Four New Approaches*. PhD thesis, ETH Zurich, 2005.
- [24] H. Cai, V. W. Zheng, and K. Chang. A comprehensive survey of graph embedding: problems, techniques and applications. *TKDE*, 2018.
- [25] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *CIKM'15*, pages 891–900. ACM, 2015.
- [26] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria. Learning community embedding with community detection and node embedding on graphs. In *CIKM*, pages 377–386. ACM, 2017.
- [27] M. E. Celebi and H. A. Kingravi. Linear, deterministic, and order-invariant initialization methods for the k-means clustering algorithm. In *Partitional Clustering Algorithms*, pages 79–98. 2015.
- [28] P. C. Céspedes and J. F. Marcotorchino. Comparing different modularization criteria using relational metric. In *GSI*, pages 180–187, 2013.
- [29] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, 7(3):163–178, Aug. 1998.
- [30] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly. Metrics for community analysis: A survey. *ACM Comput. Surv.*, 50(4), Aug. 2017.
- [31] L. Chang, W. Li, X. Lin, L. Qin, and W. Zhang. pscan: Fast and exact structural graph clustering. In *ICDE'16*, pages 253–264, 2016.
- [32] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006.
- [33] B. Chen and K. M. Ting. Neighbourhood contrast: A better means to detect clusters than density. In *PAKDD*, 2018.
- [34] H. Chen, B. Perozzi, Y. Hu, and S. Skiena. Harp: Hierarchical representation learning for networks. In *AAAI*, 2018.
- [35] M. Chen, S. Liu, and B. K. Szymanski. Parallel toolkit for measuring the quality of network community structure. In *ENIC*, pages 22–29, 2014.

- [36] M. Chen and B. K. Szymanski. Fuzzy overlapping community quality metrics. *SNAM*, 5(1):40:1–40:14, 2015.
- [37] W. Chen, J. Chen, F. Zou, Y.-F. Li, P. Lu, and W. Zhao. Robustiq: A robust ann search method for billion-scale similarity search on gpus. ICMR '19, page 132–140, New York, NY, USA, 2019. Association for Computing Machinery.
- [38] U. Chitra and C. Musco. Analyzing the impact of filter bubbles on social network polarization. WSDM '20, page 115–123, New York, NY, USA, 2020. Association for Computing Machinery.
- [39] J. Choo and S. Liu. Visual analytics for explainable deep learning. *IEEE Computer Graphics and Applications*, 38(4):84–92, Jul 2018.
- [40] M. Civil and L. Feliu. *Studies in Sumerian Civilization: Selected Writings of Miguel Civil*. Barcino monographica orientalia. Edicions de la Universitat de Barcelona, 2017.
- [41] L. M. Collins and C. W. Dent. Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions. *Multivariate Behavioral Research*, 23(2):231–242, 1988.
- [42] N. Cunningham. Investigation of the visual aspects of business intelligence. Master's thesis, Technological University Dublin, 2008.
- [43] L. Currat, A. Francisco, S. Al-Tuwaijri, A. Ghaffar, and S. Jupp. volume 18. Global Forum for Health Research, 2004.
- [44] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 9:8, Sept. 2005.
- [45] R. N. Davé and R. Krishnapuram. Robust clustering methods: a unified view. *IEEE Transactions on Fuzzy systems*, 5(2):270–293, 1997.
- [46] I. Derényi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Phys. Rev. Lett.*, 94:160202, Apr 2005.
- [47] C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*, pages 606–610, 2005.
- [48] W. Dong, C. Moses, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. WWW '11, page 577–586. Association for Computing Machinery, 2011.
- [49] A. Drif and A. Boukerram. Taxonomy and survey of community discovery methods in complex networks. *International Journal of Computer Science and Engineering Survey*, 5(4):1, 2014.
- [50] A. Dutta, C. Meilicke, and S. P. Ponzetto. A probabilistic approach for integrating heterogeneous knowledge sources. In *ESWC*, pages 286–301. Springer, 2014.
- [51] P. Eades, W. Lai, K. Misue, and K. Sugiyama. Preserving the mental map of a diagram. Technical report, Technical Report IAS-RR-91-16E, Fujitsu Laboratories, 1991.
- [52] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE T Vis Comput Gr*, 13(6):1216–1223, Nov. 2007.
- [53] N. Elmqvist and J. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE T Vis Comput Gr*, 16(3):439–454, May 2010.
- [54] A. V. Esquivel and M. Rosvall. Comparing network covers using mutual information. *CoRR*, abs/1202.0425, 2012.

Bibliography

- [55] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.
- [56] M. Ester and J. Sander. *Knowledge Discovery in Databases: Techniken und Anwendungen*. Springer Berlin Heidelberg, 2000.
- [57] V. Estivill-Castro. Why so many clustering algorithms: A position paper. *SIGKDD Explor. Newsl.*, 4(1):65–75, June 2002.
- [58] P. ffoulkes. Insidebigdata guide to the intelligent use of big data on an industrial scale. Technical report, Hewlett Packard Enterprise, 2017.
- [59] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.
- [60] S. Fortunato. Community detection in graphs. *Phys. Rep.*, 486(3-5):75–174, 2010.
- [61] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *PNAS*, 104(1):36–41, 2007.
- [62] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics reports*, 659:1–44, 2016.
- [63] P. Frankl and H. Maehara. The johnson-lindenstrauss lemma and the sphericity of some graphs. *J. Comb. Theory Ser. A*, 44(3):355–362, June 1987.
- [64] A. L. N. Fred and A. K. Jain. Robust data clustering. In *CVPR*, pages 128–136, 2003.
- [65] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.
- [66] C. Fu, C. Xiang, C. Wang, and D. Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proc. VLDB Endow.*, 12(5):461–474, Jan. 2019.
- [67] C. Fu, Y. Zhang, D. Cai, and X. Ren. Atsne: Efficient and robust visualization on gpu through hierarchical optimization. *KDD '19*, page 176–186, New York, NY, USA, 2019. ACM.
- [68] G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*. ASA-SIAM. Society for Industrial and Applied Mathematics, 2007.
- [69] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, CA, 1979.
- [70] R. Goebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, and A. Holzinger. Explainable ai: The new 42? In *CD-MAKE 2018*, volume 11015, 2018.
- [71] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, Dec. 1992.
- [72] B. H. Good, Y.-A. de Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81(4):046106, 2010.
- [73] C. Granell, S. Gómez, and A. Arenas. Hierarchical multiresolution method to overcome the resolution limit in complex networks. *Int. J. Bifurc. Chaos*, 22(07):1250171, 2012.
- [74] S. Gregory. A fast algorithm to find overlapping communities in networks. In *ECML PKDD*, pages 408–423, 2008.
- [75] S. Gregory. Fuzzy overlapping communities in networks. *J Stat Mech.*, 2011(02):P02017, 2011.

- [76] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. KDD '16, pages 855–864. ACM, 2016.
- [77] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD'16*, pages 855–864. ACM, 2016.
- [78] A. M. Haith and J. W. Krakauer. The multiple effects of practice: skill, habit and reduced cognitive load. *Current opinion in behavioral sciences*, 20:196–201, April 2018.
- [79] S. Harenberg, G. Bello, L. Gjeltrema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova. Community detection in large-scale networks: a survey and empirical evaluation. *Wiley Interdiscip Rev Comput Stat.*, 6(6):426–439, dec 2014.
- [80] S. Harenberg, G. Bello, L. Gjeltrema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova. Community detection in large-scale networks: A survey and empirical evaluation. *WIREs Comput. Stat.*, 6:426–439, Nov. 2014.
- [81] Hart, Michael S. Gutenberg:the history and philosophy of project gutenber, 1992.
- [82] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
- [83] M. Hascoët and G. Artignan. Evaluation of Clustering Algorithms: a methodology and a case study. Technical Report RR-14008, Sept. 2014.
- [84] C. Hennig, M. Meila, F. Murtagh, and R. Rocci, editors. *Handbook of Cluster Analysis*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, 12 2015.
- [85] J. Hou and W.-X. Liu. Clustering based on dominant set and cluster expansion. In *PAKDD*, 2017.
- [86] Y. Hu, S. G. Kobourov, and S. Veeramoni. Embedding, clustering and coloring for dynamic maps. In *2012 IEEE Pacific Visualization Symposium*, pages 33–40, Feb 2012.
- [87] D. Huang, J.-H. Lai, and C.-D. Wang. Robust ensemble clustering using probability trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1312–1326, 2016.
- [88] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec 1985.
- [89] E. Hullermeier and M. Rifqi. A fuzzy variant of the rand index for comparing clustering structures. IFSA-EUSFLAT 2009.
- [90] R. Hussein, D. Yang, and P. Cudré-Mauroux. Are meta-paths necessary?: Revisiting heterogeneous graph embeddings. In *CIKM*, pages 437–446, 2018.
- [91] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. *VIS '90*, page 361–378. IEEE, 1990.
- [92] A. Iosup, T. Hegeman, W. L. Ngai, S. Heldens, A. Prat-Pérez, T. Manhardto, H. Chafio, M. Capotă, N. Sundaram, M. Anderson, I. G. Tănase, Y. Xia, L. Nai, and P. Boncz. Ldbc graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms. *Proc. VLDB Endow.*, 9:1317–1328, Sept. 2016.
- [93] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, Sept. 1999.
- [94] W. Javed and N. Elmqvist. Stack zooming for multi-focus interaction in skewed-aspect visual spaces. *IEEE T Vis Comput Gr*, 19(8):1362–1374, 2013.

Bibliography

- [95] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 271–279, New York, NY, USA, 2003. ACM.
- [96] Q. Ji, Z. Gao, and Z. Huang. Reasoning with noisy semantic data. In *8th Extended Semantic Web Conference, Part II*, pages 497–502, 2011.
- [97] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, pages 1–1, 2019.
- [98] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, May 2004.
- [99] Z. Kaoudi, I. Miliaraki, and M. Koubarakis. Rdfs reasoning and query answering on top of dhds. In *ISWC*, pages 499–516, 2008.
- [100] B. Karrer, E. Levina, and M. E. J. Newman. Robustness of community structure in networks. *Phys. Rev. E*, 77:046119, Apr 2008.
- [101] B. Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83, Jan 2011.
- [102] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.
- [103] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann. *Mastering the Information Age - Solving Problems with Visual Analytics*. Eurographics Association, 2010.
- [104] D. A. Keim. Visual exploration of large data sets. *Commun. ACM*, 44(8):38–44, Aug. 2001.
- [105] D. A. Keim and H.-P. Kriegel. Visualization techniques for mining large databases: a comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):923–938, Dec 1996.
- [106] K. Kellou-Menouer and Z. Kedad. Evaluating the gap between an rdf dataset and its schema. In *Advances in Conceptual Modeling*, pages 283–292. Springer, 2015.
- [107] K. Kellou-Menouer and Z. Kedad. Schema discovery in RDF data sources. In *Conceptual Modeling, ER 2015, Stockholm*, pages 481–495, 2015.
- [108] T. Kliegr and O. Zamazal. LHD 2.0: A text mining approach to typing entities in knowledge graphs. *J. Web Sem.*, 39:47–61, 2016.
- [109] D. Kobak and P. Berens. The art of using t-sne for single-cell transcriptomics. *Nature Communications*, 10(1), 2019.
- [110] M. Koster. Aliweb–archie-like indexing in the web. In *WWW*, pages 175–182, 1994.
- [111] B. Kovalerchuk. *Visual Knowledge Discovery and Machine Learning*. Intelligent Systems Reference Library. Springer International Publishing, 2018.
- [112] B. Kovalerchuk. Interpretable knowledge discovery reinforced by visual methods. KDD '19, page 3219–3220. ACM, 2019.
- [113] B. Kovalerchuk. Intelligible machine learning and knowledge discovery boosted by visual means. WSDM '20, page 881–883. ACM, 2020.
- [114] B. Kovalerchuk and V. Grishin. Adjustable general line coordinates for visual knowledge discovery in n-d data. *Information Visualization*, 18(1):3–32, 2019.
- [115] E. Krol. *The Whole Internet User's Guide and Catalog*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1992.

- [116] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.
- [117] G. Kumar and M. Garland. Visual exploration of complex time-varying graphs. *IEEE T Vis Comput Gr*, 12(5):805–812, Sept. 2006.
- [118] G. Kumar and M. Garland. Visual exploration of complex time-varying graphs. *IEEE T Vis Comput Gr*, 12(5):805–812, Sep. 2006.
- [119] J. M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki. Sequential algorithm for fast clique percolation. *Phys. Rev. E*, 78(2):026109, 2008.
- [120] Kumpula, J. M., Saramäki, J., Kaski, K., and Kertész, J. Limited resolution in complex network community detection with potts model approach. *Eur. Phys. J. B*, 56(1):41–45, 2007.
- [121] M. D. König, C. J. Tessone, and Y. Zenou. Nestedness in networks: A theoretical model and some applications. *Theoretical Economics*, 9(3):695–752, 2014.
- [122] R. Lambiotte. Multi-scale modularity in complex networks. In *WiOpt*, pages 546–553, May 2010.
- [123] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, 80, Jul 2009.
- [124] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Phys. Rev. E*, 84:066122, Dec 2011.
- [125] A. Lancichinetti and S. Fortunato. Consensus clustering in complex networks. *Sci. Rep.*, 2, 2012.
- [126] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New J. Phys.*, 11(3):033015, 2009.
- [127] A. laszlo Barabasi. *Linked: The New Science of Networks*. apr 2002.
- [128] S. Lawrence. Context in web search. *IEEE Data Engineering Bulletin*, 23:25–32, 2000.
- [129] A. Lázár, D. Abel, and T. Vicsek. Modularity measure of networks with overlapping communities. *EPL*, 90(1):18001, 2010.
- [130] J. Lehmann, G. Sejdin, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.-C. Ngonga Ngomo, and H. Jabeen. Distributed semantic analytics using the sansa stack. In *ISWC*, pages 147–155, 2017.
- [131] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225, 2015.
- [132] J. M. Lewis, V. R. De Sa, and L. Van Der Maaten. Divvy: Fast and intuitive exploratory data analysis. *J. Mach. Learn. Res.*, 14(1):3159–3163, Jan. 2013.
- [133] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2):1–16, Feb. 2016.
- [134] D. Lian, K. Zheng, V. W. Zheng, Y. Ge, L. Cao, I. W. Tsang, and X. Xie. High-order proximity preserving information network hashing. In *KDD’18*, pages 1744–1753. ACM, 2018.
- [135] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.
- [136] J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom. Viztree: A tool for visually mining and monitoring massive time series databases. *VLDB ’04*, page 1269–1272, 2004.

Bibliography

- [137] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. AAAI'15, pages 2181–2187. AAAI Press, 2015.
- [138] J. Liu. Fuzzy modularity and fuzzy community structure in networks. *Eur. Phys. J. B*, 77(4):547–557, 2010.
- [139] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Comput. Surv.*, 51(3), June 2018.
- [140] A. Lutov, M. Khayati, and P. Cudré-Mauroux. Clubmark: a parallel isolation framework for benchmarking and profiling clustering algorithms on numa architectures. In *ICDMW*, pages 1481–1486, 2018.
- [141] A. Lutov, M. Khayati, and P. Cudré-Mauroux. Accuracy evaluation of overlapping and multi-resolution clustering algorithms on large datasets. In *BigComp*, pages 1–8, 2019.
- [142] A. Lutov, M. Khayati, and P. Cudré-Mauroux. Daoc: Stable clustering of large networks. In *IEEE BigData*, 2019.
- [143] A. Lutov, S. Roshankish, M. Khayati, and P. Cudré-Mauroux. Statix — statistical type inference on linked data. In *BigData*, BigData'18, pages 2253–2262, 2018.
- [144] A. Lutov, D. Yang, and P. Cudré-Mauroux. Bridging the gap between community and node representations: Graph embedding via community detection. In *IEEE BigData*, 2019.
- [145] M. E. J. Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Phys. Rev. E*, 94, 2016.
- [146] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9:2579–2605, Nov 2008.
- [147] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.
- [148] D. Mandaglio, A. Amelio, and A. Tagarelli. Consensus community detection in multilayer networks using parameter-free graph pruning. In *PAKDD*, 2018.
- [149] M. Manktelow. History of taxonomy. Uppsala, 2010.
- [150] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate analysis*. Academic Press, 1979.
- [151] E. L. Martelot and C. Hankin. Multi-scale community detection using stability optimisation. *Int. J. Web Based Communities*, 9(3):323–348, June 2013.
- [152] L. J. Martin. *Robust and Efficient Data Clustering with Signal Processing on Graphs*. PhD thesis, École polytechnique fédérale de Lausanne (EPFL), 2018.
- [153] P. Mazzuca and Y. T. Circulo: A Community Detection Evaluation Framework. Feb. 2015.
- [154] A. F. McDaid, D. Greene, and N. J. Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *CoRR*, abs/1110.2515, 2011.
- [155] L. McGillis. Gopher searching using veronica. *The Reference Librarian*, 19(41-42):25–35, 1994.
- [156] L. McInnes and J. Healy. Accelerated hierarchical density based clustering. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, nov 2017.
- [157] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, Feb. 2018.

-
- [158] O. Medelyan, S. Manion, J. Broekstra, A. Divoli, A.-L. Huang, and I. H. Witten. Constructing a focused taxonomy from a document collection. In P. Cimiano, O. Corcho, V. Presutti, L. Hollink, and S. Rudolph, editors, *ESWC*, pages 367–381, 2013.
- [159] V. Melnykov and R. Maitra. Carp: Software for fishing out good clustering algorithms. *J. Mach. Learn. Res.*, 12:69–73.
- [160] A. Melo, H. Paulheim, and J. Völker. Type prediction in rdf knowledge bases using hierarchical multilabel classification. In *Web Intelligence, Mining and Semantics*, pages 14:1–14:10, 2016.
- [161] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *NIPS’13*, pages 3111–3119, 2013.
- [162] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychol. Rev.*, 63(2), 1956.
- [163] B. Mirkin. *Mathematical Classification and Clustering*. Nonconvex Optimization and Its Applications. Springer US, 1996.
- [164] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183 – 210, 1995.
- [165] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, 52(1):91–118, 2003.
- [166] J. Moreno and H. Jennings. *Who Shall Survive? A New Approach to the Problem of Human Interrelations*. Nervous and mental disease monograph. 1934.
- [167] E. Müller. Big data analytics: Clustering, February 2016.
- [168] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó. Fuzzy communities and the concept of bridge-ness in complex networks. *Phys. Rev. E*, 77:016107, Jan 2008.
- [169] M. Newman, A. Iaszlo Barabasi, and D. J. Watts. *The Structure and Dynamics of Networks*. 2006.
- [170] M. E. J. Newman. Mixing patterns in networks. *Phys. Rev. E*, 67:026126, Feb 2003.
- [171] M. E. J. Newman. The structure and function of complex networks. *SIAM Rev.*, 45(2), 2003.
- [172] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004.
- [173] M. E. J. Newman. Spectral methods for network community detection and graph partitioning. *Phys. Rev. E*, 88(4):042822, 2013.
- [174] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, 2004.
- [175] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *J Stat Mech.*, 3:24, Mar. 2009.
- [176] A. Nolle, M. W. Chekol, C. Meilicke, G. Nemirovski, and H. Stuckenschmidt. Automated fine-grained trust assessment in federated knowledge bases. In *ISWC’17*, pages 490–506.
- [177] M. Novotny and H. Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE T Vis Comput Gr*, 12(5):893–900, Sep. 2006.

Bibliography

- [178] N. A. of Engineering. *Information Technologies and Social Transformation*. The National Academies Press, Washington, DC, 1985.
- [179] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. Asymmetric transitivity preserving graph embedding. In *KDD'16*, pages 1105–1114. ACM, 2016.
- [180] M. Ovelgönne and A. Geyer-Schulz. An ensemble learning strategy for graph clustering. volume 588 of *Contemp. Math.*, pages 187–206, 2013.
- [181] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
- [182] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.
- [183] V. Pareto. *Manual of political economy (manuale di economia politica)*. Kelley, 1971 (1906).
- [184] E. Pariser. *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Group, The, 2011.
- [185] H. Paulheim and C. Bizer. Type inference on noisy RDF data. In *The Semantic Web - ISWC 2013*, pages 510–525, 2013.
- [186] H. Paulheim and C. Bizer. Improving the quality of linked data using statistical distributions. *IJSWIS*, 10(2):63–86, 2014.
- [187] T. P. Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Phys. Rev. E*, 89, Jan 2014.
- [188] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. EMNLP'14, pages 1532–1543, Doha, Qatar, Oct. ACL.
- [189] B. Perozzi and L. Akoglu. Discovering communities and anomalies in attributed graphs: Interactive visual exploration and summarization. *Trans. Knowl. Discov. Data*, 12(2), Jan. 2018.
- [190] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. KDD'14, pages 701–710. ACM, 2014.
- [191] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD'14*, pages 701–710. ACM, 2014.
- [192] B. Pinkerton. Finding what people want: Experiences with the webcrawler. In *Proc. of the Second International WWW Conference*, 1994.
- [193] C. Plaisant, J. Grosjean, and B. B. Bederson. Spacetreel: Supporting exploration in large node link tree, design evolution and empirical evaluation. INFOVIS '02, page 57, USA, 2002. IEEE.
- [194] P. Pons and M. Latapy. Post-processing hierarchical community structures: Quality improvements and multi-scale view. *Theor. Comput. Sci.*, 412(8-10):892–900, Mar. 2011.
- [195] A. Pothen, H. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIMAX*, 11(3):430–452, 1990.
- [196] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey. High quality, scalable and parallel community detection for large real graphs. WWW '14, pages 225–236, 2014.
- [197] R. Prokofyev, A. Boyarsky, O. Ruchayskiy, K. Aberer, G. Demartini, and P. Cudré-Mauroux. Tag recommendation for large-scale ontology-based information systems. ISWC'12, 2012.

-
- [198] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM'18*, pages 459–467. ACM, 2018.
 - [199] R. Rabbany, M. Takaffoli, J. Fagnan, O. R. Zaïane, and R. J. G. B. Campello. Communities validity: methodical evaluation of community mining algorithms. *Social Netw. Analys. Mining*, 3(4):1039–1062, 2013.
 - [200] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. *CSCW '94*, pages 175–186. ACM, 1994.
 - [201] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.
 - [202] H. Rosales-Méndez and Y. Ramírez-Cruz. Cice-bcubed: A new evaluation measure for overlapping clustering algorithms. *CIARP'13*.
 - [203] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. *EMNLP-CoNLL'07*, pages 410–420.
 - [204] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.
 - [205] W. Saelens, R. Cannoodt, and Y. Saeys. A comprehensive evaluation of module detection methods for gene expression data. *Nat. Commun.*, 9:12, 2018.
 - [206] M. Sales-Pardo, R. Guimerà, A. A. Moreira, and L. A. N. Amaral. Extracting the hierarchical organization of complex systems. *PNAS*, 104(39):15224–15229, Sept. 2007.
 - [207] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.
 - [208] M. Saveski and A. Mantrach. Item cold-start recommendations: Learning local collective embeddings. *RecSys '14*, pages 89–96. ACM, 2014.
 - [209] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. *SIGIR '02*, pages 253–260, New York, NY, USA, 2002. ACM.
 - [210] E. Schmidt and J. Cohen. *The new digital age: reshaping the future of people, nations and business / by Eric Schmidt and Jared Cohen*. John Murray London, 2013.
 - [211] M. Shahriari, S. Krott, and R. Klamma. Webocd: A restful web-based overlapping community detection framework. *i-KNOW '15*. ACM.
 - [212] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.
 - [213] U. Shardanand. *Social information filtering for music recommendation*. PhD thesis, Massachusetts Institute of Technology, 1994.
 - [214] H.-W. Shen, X.-Q. Cheng, and J.-F. Guo. Quantifying and identifying the overlapping community structure in networks. *J Stat Mech.*, 2009(07):P07042, 2009.
 - [215] B. Sheth and P. Maes. Evolving agents for personalized information filtering. In *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*, pages 345–352, March 1993.
 - [216] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *VL '96*, page 336. IEEE, 1996.

Bibliography

- [217] B. Shneiderman. Extreme visualization: Squeezing a billion records into a million pixels. SIGMOD '08, page 3–12, New York, NY, USA, 2008. ACM.
- [218] H. A. Simon. The architecture of complexity. *Proc. Am. Phil. Soc.*, 106(6):467–482, 1962.
- [219] P. Simonetto, D. Archambault, and S. Kobourov. Drawing dynamic graphs without timeslices. In F. Frati and K.-L. Ma, editors, *GD*, pages 394–409. Springer, 2018.
- [220] T. A. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14(1):75–100, Jan 1997.
- [221] T. Soukup and I. Davidson. *Visual Data Mining: Techniques and Tools for Data Visualization and Mining*. John Wiley & Sons, Inc., USA, 1st edition, 2002.
- [222] R. Spence. *Information Visualization: Design for Interaction (2nd Edition)*. Prentice-Hall, Inc., USA, 2007.
- [223] D. Stenmark. The relationship between information and knowledge. In *IRIS 24*, pages 11–14, 2001.
- [224] T. Su and J. G. Dy. In search of deterministic methods for initializing k-means and gaussian mixture clustering. *Intelligent Data Analysis*, 11(4):319–338, 2007.
- [225] P. Symeon, K. Yiannis, V. Athena, and S. Ploutarchos. Community detection in social media, performance and application considerations. *Data Min. Knowl. Discov.*, 24(3):515–554, 2012.
- [226] A. Tandon, A. Albesri, V. Thayanathan, W. Alhalabi, and S. Fortunato. Fast consensus clustering in complex networks. *Phys. Rev. E*, 99:042301, Apr 2019.
- [227] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. WWW '16, page 287–297, 2016.
- [228] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.
- [229] L. Tang and H. Liu. Relational learning via latent social dimensions. In *ACM SIGKDD*, pages 817–826. ACM, 2009.
- [230] L. Tang and H. Liu. Scalable learning of collective behavior based on sparse social dimensions. In *CIKM'09*, pages 1107–1116. ACM, 2009.
- [231] L. Tang and H. Liu. Leveraging social media networks for classification. *Data Min. Knowl. Discov.*, 23:447–478, Nov. 2011.
- [232] M. Tepper, P. Musé, A. Almansa, and M. Mejail. Automatically finding clusters in normalized cuts. *Pattern Recognition*, 44(7), 2011.
- [233] H. J. ter Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Web Semant.*, 3(2-3):79–115, oct 2005.
- [234] T. Tian, J. Geller, and S. A. Chun. Improving web search results for homonyms by suggesting completions from an ontology. In F. Daniel and F. M. Facca, editors, *Current Trends in Web Engineering*, pages 175–186. Springer, 2010.
- [235] A. Tonon. *Leveraging Entity Types and Properties for Knowledge Graph Exploitation*. PhD thesis, University of Fribourg (Switzerland), 2017.
- [236] A. Tonon, G. Demartini, and P. Cudré-Mauroux. Combining inverted indices and structured search for ad-hoc object retrieval. In *ACM SIGIR*. ACM, 2012.

- [237] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. In *Scientific Reports*, volume 9, 2018.
- [238] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller. Verse: Versatile graph embeddings from similarity measures. In *WWW'18*, pages 539–548, 2018.
- [239] N. Tsuchiya and J. Van Boxtel. *Attention and consciousness in different senses*. Frontiers Research Topics. Frontiers E-books, 2013.
- [240] J. W. Tukey. *Exploratory data analysis*, volume 2. Reading, Mass., 1977.
- [241] P. Vakkari. Information seeking in context: A challenging metatheory. ISIC '96, pages 451–464, 1997.
- [242] A. D. Vanberg. From archie to google: Search engine providers and emergent challenges in relation to eu competition law. *European Journal of Law and Technology*, 3(1), 2012.
- [243] S. Vega-Pons and J. Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *Int. J. Pattern Recogn.*, 25(03):337–372, 2011.
- [244] F. Viégas and M. Wattenberg. Tutorial: Data visualization for machine learning, December 2018.
- [245] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. *ACM SIGKDD*, pages 1225–1234, 2016.
- [246] F. Wang, C. Ding, and T. Li. *Integrated KL (K-means – Laplacian) Clustering: A New Clustering Approach by Combining Attribute Data and Pairwise Relations*, pages 38–48. 2009.
- [247] M. Wang, C. Wang, J. X. Yu, and J. Zhang. Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework. *Proc. VLDB Endow.*, 8(10):998–1009, June 2015.
- [248] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang. Community preserving network embedding. In *AAAI*, 2017.
- [249] M. Wattenberg. Arc diagrams: visualizing structure in strings. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pages 110–116, Oct 2002.
- [250] M. Wattenberg, F. Viégas, and I. Johnson. How to use t-sne effectively. *Distill*, 2016.
- [251] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.
- [252] W. Wu, B. Li, L. Chen, and C. Zhang. Efficient attributed network embedding via recursive randomized hashing. In *IJCAI*, 2018.
- [253] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. *SIGMOD '12*, page 481–492. ACM, 2012.
- [254] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Comput. Surv.*, 45(4):1–35, aug 2013.
- [255] J. Xie, B. K. Szymanski, and X. Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *ICDM 2011 Workshop on DMCCI*, pages 344–349.
- [256] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 8 2015.

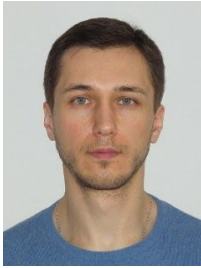
Bibliography

- [257] R. Xu and D. Wunsch. Survey of clustering algorithms. *Trans. Neur. Netw.*, 16(3):645–678, May 2005.
- [258] D. Yang, B. Li, L. Rettig, and P. Cudré-Mauroux. Histosketch: Fast similarity-preserving sketching of streaming histograms with concept drift. In *ICDM'17*, pages 545–554. IEEE, 2017.
- [259] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. In *WWW'19*, pages 2147–2157. ACM, 2019.
- [260] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *WWW*, pages 2147–2157. ACM, 2019.
- [261] D. Yang, Z. Xie, E. A. Rundensteiner, and M. O. Ward. Managing discoveries in the visual analytics process. *SIGKDD Explor. Newsl.*, 9(2):22–29, Dec. 2007.
- [262] J. Yang and J. Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *WSDM'13*, pages 587–596.
- [263] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.*, 2015.
- [264] Z. Yang, R. Algesheimer, and C. J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, 6:30750, 2016.
- [265] Z. Yang, J. I. Perotti, and C. J. Tessone. Hierarchical benchmark graphs for testing community detection algorithms. *Physical review E*, 96(5), 2017.
- [266] Y. Y. Yao. Granular computing: basic issues and possible solutions. In *5th Joint Conference on Information Sciences*, pages 186–189, 2000.
- [267] Y. Y. Yao. Information granulation and rough set approximation. *Int J Intell Syst*, 16(1):87–104, 2001.
- [268] X. Ying, C. Wang, M. Wang, J. X. Yu, and J. Zhang. Codar: Revealing the generalized procedure & recommending algorithms of community detection. *SIGMOD '16*, pages 2181–2184. ACM, 2016.
- [269] L. Yu and C. Ding. Network community discovery: Solving modularity clustering via normalized cut. *MLG '10*, pages 34–36. ACM.
- [270] L. A. Zadeh. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Systems*, 90(2):111 – 127, 1997. Fuzzy Sets: Where Do We Stand? Where Do We Go?
- [271] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.
- [272] H. Zhang, C. L. Giles, H. C. Foley, and J. Yen. Probabilistic community discovery using hierarchical latent gaussian mixture model. volume 1 of *AAAI*, pages 663–668, 2007.
- [273] Q.-s. Zhang and S.-c. Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, Jan 2018.
- [274] S. Zhang, R.-S. Wang, and X.-S. Zhang. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A*, 374(1):483–490, 2007.
- [275] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, and H. Chen. Interaction embeddings for prediction and explanation in knowledge graphs. *WSDM '19*, pages 96–104. ACM, 2019.

- [276] X. Zhang, E. Lin, and S. Pi. Predicting object types in linked data by text classification. In *CBD*, pages 391–396, 2017.
- [277] Q. Zhao. *Cluster Validity in Clustering Methods*. PhD thesis, University of Eastern Finland, 2012.
- [278] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.*, 2(1):718–729, Aug. 2009.
- [279] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017.
- [280] C. Zins. Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology*, 58(4):479–493, 2007.

Curriculum Vitae

Artem Vitalievich Lutov



Home Address: Fribourg, Switzerland
Year of birth: 1984
Citizenship: Ukrainian
Email: artem@exascale.info
Skype: luartvi
Actual CV: <https://bit.ly/lavCVpub>

OBJECTIVE	CTO, Team Leader, R&D Engineer (AI: Data Science, Business Intelligence, Machine Learning, NLP; CS/IT), AI&IT Consulting, C++/Python/JS Development
Preferable fields:	<p>Symbiotic Autonomous Systems, AI, Data Analytics, Business Intelligence, IoT/E, Wearable and embedded devices, Human-Computer Interaction, Bionics.</p> <p>I'm looking for a team-leading/research/consulting position in the intersection of big data analytics, business intelligence, software design & development for cloud and edge devices (e.g., IoT, robotics), and human-machine interfaces.</p>
Employment type:	Full day / Part time / Remote

Skills summary:

Core technologies:	<ul style="list-style-type: none"> • <u>Data Analysis</u> (during work at UNIFR, eXascale Infolab and ScienceWISE) Clustering and classification, machine learning, statistical inference, community structure discovery, graph embedding, automated building of ontologies / taxonomies; semantic data (graphs) visualization, navigation and editing; benchmarking of clustering algorithms, collections similarity measures. • <u>Embedded and Distributed Systems, System and Network programming</u> (during work at Samsung Electronics, Symantec PCTools and Startups) ARM, OMAP, TI MSP devices on Linux/Android/ThreadX/VxWorks platforms, distributed systems • <u>Human-Computer Interaction, Computer Vision & Image Processing</u> (during work at Samsung Electronics) Human gesture recognition for Augmented Reality environments on 2D/3D (Primesense, Kinect, Optrima, Canesta) cameras, objects tracking • <u>Team Leading & Project Management</u> (during work at Samsung Electronics) OOD/OOA, RUP, BDD/TDD, Agile, Kanban, Lean, Six Sigma DMAIC Led teams of 8-21 developers + QAEs and designers, cooperation with remote co-teams on the customer side (UX dep. in Korea), infrastructure establishment for the projects from scratch. Have commercialized projects (Gesture Engine for Samsung MV900 photo camera: http://www.samsung.com/global/mv900f/, see Gesture Shot).
Platforms:	<ul style="list-style-type: none"> • <u>Nix (Linux/Unix)</u> User mode development, Network programming, embedded systems development (ARM CPUs, TI MSP430 chips), kernel rebuilding (including custom kernel modules for Android), GCC, CLang, ccache, build automation

	<p>(make, qmake, ANT) and Python / bash scripting; administration (Linux/FreeBSD OS and services: SCMs, Issue/Bugs tracking, communication services)</p> <ul style="list-style-type: none"> • <u>Android</u> User mode API, Services, NDK (JNI bindings, kernel modules building, rooting) • <u>Windows</u> (98, NTx86-Win7x64) WinAPI, System and Network programming, basics of Kernel driver development (Symantec PCTools Firewall porting to x64), WDK; GUI: MFC/WTL, GDI+ • <u>DOS</u> (knowledge of architecture, system programming) • <u>Embedded HW platforms</u> Android based mobile devices, TI Beagle Board, Arduino, TI MSP430 • <u>Computer Vision platforms</u>: OpenCV & proprietary frameworks (at Samsung) • <u>Web</u> HTML/CSS/JS, D3.JS, Vue.JS, Flask, Bottle, Jinja, SSI, F/P CGI, GQL/NDB, PostgreSQL, MySQL, XML (DOM, SAX, XPath; DTD, XSD, RelaxNG) • <u>Big Data Processing Systems & NoSQL DBs</u> Google Cloud Platform; ELK Stack; Cloudera Hadoop, pySpark; ArangoDB (usage, administration and tutorials preparation) • <u>Cloud Platforms & Services</u> Google Cloud Platform (BigQuery ML, Data Studio, AutoML, Perception API, AI Platform & Hub, Kubeflow, App Engine including NDB); AWS (EC2, S3, Beanstalk, Amplify, Amazon SageMaker); Azure Machine Learning Studio (data cleansing); IBM Cloud/Bluemix, CloudFoundry (web services deployment); Docker containers (cross-platform deployment of computational services)
Tools:	<ul style="list-style-type: none"> • <u>IDEs</u> VSCode & Geany, Code::Blocks (C++), Komodo Edit (Python), Jupyter Notebooks (Python, Julia), MSVS (VC), Eclipse, BlueFish (HTML), Energiya (Arduino, TI MSP430), Qt Creator, IntelliJ IDEA, BCB, Delphi • <u>Debugging</u> WinDbg (Win Kernel), IDA & OllyDbg (Win User Mode), gdb • <u>VCS</u> Git, Mercurial and Subversion (usage & administration), Seapine Surround SCM, Perforce • <u>Issue Tracking</u> YouTrack, Redmine, Trac (usage & administration); GitHub and Bitbucket; be • <u>Planning</u> YouTrack, Bitrix24, Zoho Projects (full-fledged solutions), Favro (Kanban), Ganttproject and ProjectLibre (Gantt diagrams, RBS and PERT charts), VYM and FreeMind (mind maps) • <u>Build automation</u> and continuous integration make, qmake, Ant; basics of CruiseControl.NET, Jenkins/Hudson; Docker containers based deployment • <u>Data Analysis Tools & Machine Learning Frameworks</u> RapidMiner; basics of KNIME; igraph, NetworKit; basics of Minitab, SAS; Scikit-learn, TensorFlow; NLTK, FastText; basics of pySpark with MLlib • <u>Accessory Tools</u> for Presentations preparation Google Docs/Sheets/Slides, MS Office (including Visio), Open/Libre Office, TeXstudio, DIA, <i>Gimp, Inkscape</i>, Movie Maker, OpenShot, Blender, <i>FreeCAD</i>; basics of Photoshop, 3DSMax, Corel Draw, Autocad
Expert in:	<ul style="list-style-type: none"> • C/C++ and STL
Good in:	<ul style="list-style-type: none"> • Python (including IPython with Pandas, NumPy and Cython, binding with C++17)

	libs via SWIG, Cython and Boost-Python) <ul style="list-style-type: none"> • JavaScript / HTML5 / CSS3, D3.JS • Java (Android development) • ASM x86, reverse code engineering and debugging MASM, TASM. Worked with IDA, GDB, WinDbg, OllyDbg
Worked with:	<ul style="list-style-type: none"> • SQL (PostgreSQL, MySQL, SQLite), GQL/NDB/BigQuery, SPARQL and RDF • Scikit-learn, TensorFlow; RapidMiner; NLTK, FastText; igraph; Minitab, SAS • Elasticsearch, Hadoop, pySpark; Graph DBs (mainly ArangoDB) • Google AI Platform, Amazon Sagemaker • Julia, Nim, Golang, Octave 3.x / Matlab, Pascal, Ada; VHDL • UML; SOAP & XML processing • ...

EDUCATION	Doctorate in higher education: <u>Doctor of Philosophy in Computer Science</u>
02.2014 - 05.2020	<u>Université de Fribourg Suisse</u> (University of Fribourg, Switzerland) PhD Research in the field of Big Data analysis at <u>eXascale Infolab</u> PhD Thesis: " <u>Unsupervised and Parameter-free Clustering of Large Graphs for Knowledge Exploration and Recommendation</u> ", 2020, nominated for the award
09.2005 – 06.2007	<u>NTUU "Kiev Polytechnic Institute"</u> , Kiev, Ukraine Master. Specialty: 8.091501 'Computer systems and networks', ICEF Faculty MSc Thesis: "Data backup system based on the virtual file system", 2007
09.2001 – 06.2005	<u>NTUU "Kiev Polytechnic Institute"</u> , Kiev, Ukraine Faculty of Informatics and Computer Engineering (ICEF). Bachelor of 'Computer engineering' Bachelor Thesis: "Incremental work scheduling method in parallel computing systems", 2005
09.1997 – 06.2001	<u>"Lyceum of Foreign Languages"</u> , Lugansk, Ukraine Specialty: Mathematics, Physics and English

LANGUAGES	Reading	Speaking	Writing
English	fluent	good	good
Russian (native)	native	native	good
Ukrainian	fluent	good	good

Experience summary:

- 02.2014-03.2020 - PhD Researcher. **Big Data analysis**, development of **clustering and recommender algorithms** (libraries and their integrations), **unsupervised machine learning**; **graph embedding**, automation of taxonomies construction, visualization and editing, information discovery and search assistance; benchmarking of clustering and recommending algorithms. Lectures and labs preparation in the fields of Graph DBs, **Social Media Analysis** (Communities Detection), Big Data Infrastructures (Elasticsearch labs) and parts of some other courses. I worked at the eXascale Infolab and participated in the ScienceWise development (scientific papers' ontology construction and visualization).

My projects at the eXascale Infolab (XI):

- [libdaor](#) - the first framework, which fully automatically build **graph embeddings** for any input network/graph without any manual tuning (was presented at BigData'19 in LA, USA)
- [libdaoc](#) - a new generation **clustering** library for the stable clustering of large networks (was presented at BigData'19 in LA, USA);
 - [hirecs](#) - a basic ancestor of the libdaoc, deterministic overlapping clustering algorithm;
- [StaTIX](#) / [TInfES](#) - statistical types inference for **semantic datasets** and its evaluation;
- [Clubmark](#) / [PyCaBeM](#) - a parallel isolation framework for benchmarking and profiling of clustering (community detection) algorithms considering overlaps (covers), specifying fine-grained control policies for timings, employed CPU cores, cache, RAM, etc.
- [PyExPool](#) - a lightweight multi-process execution pool with load balancing, customizable resource consumption constraints and tasks monitoring;
- [xmeasures](#) - extrinsic measures for overlapping multiresolution clustering evaluation.
- Accessory tools: [resmerge](#), [PyNetConvert](#), ...

Contributions to the FOSS projects used during the work at XI:

- [python-igraph](#) porting to pypy JIT, minor contributions to the [lgraph](#) **network analysis** lib;
 - [LFR benchmark](#) I/O extension and bugs fixing;
 - [Gecmi](#) (GenConvNMI) generalized *overlapping* NMI evaluation (I/O extension, major algorithm optimizations and bugs fixing);
 - [ONMI](#) (OvpNMI) extension and bugs fixing;
 - Assistance of [SWIG](#) (C++ binding framework for almost any other language) extension for C++11/14 support, specification of the [core issues](#);
 - Python [RDFlib](#) bugfix;
 - [Mettle](#) C++14 unit test framework (minor extension and fixes);
 - Open source graph embedding algorithms and related libraries: [HARP](#) and [magic-graph](#), [DeepWalk](#), [NodeHash](#), ...
 - Open source clustering algorithms: [Leiden](#), [pSCAN](#), [CGGC](#), [OSLOM2](#), ...
 - Minor contribution to the official C++ code guidelines: [CppCoreGuidelines](#), [Nim](#) language, [nim-argparse](#) package, [Code::Blocks](#), ...
- **06.2012-today** - [Entrepreneur](#). Own and collaborative projects & startups. My former website: <http://lumais.com>. [My posts](#) about software development related activities.
07-10.2012 Developed technical part of the Petcube project prototype, implemented its embedded version that was represented on IDCEE-2012, read <http://habrahabr.ru/post/159763/>.
03.2013 - 11.2016 Collaborative activities with <http://sciencewise.info/> in infographics and some components of their **semantic web** service.
03.2015-2016 Founder of Amisens keypad, a portable keypad that provides capability to work on smartphones, smart TVs and Virtual Reality Glasses with the same productivity as on laptops. CEO & CTO and Team Leader, coordinating the team of 6 members and external partners of the project. The project has been frozen.
2019-today Founder of [Armofab](#) composite material (features dynamic flexibility) and the first consumer product based on that material (wristguards for snowboarders and rollers). We are going to start the ground-funding in the spring 2020 and then build a whole IoT infrastructure around our smart wearables, respecting privacy of our consumers. Our equipment provides outstanding protection being lightweight, flexible (becoming rigid in response to hits and punches) and comfortable to wear.
05.2020-today Sole Proprietorship consulting services (<https://lutan.ch>) in AI (**business intelligence, data analysis, machine learning**), **software architecture design**, and **cloud integration**. Clients: [Dydon AG](#) (Business Intelligence Platform), UNIFR [Lab of Social Fluids](#).

- 04.2009-06.2012 – Samsung Ukraine Research Center, R&D. Position: *Project Leader, Senior Software Engineer (contractor)*. I started as a Lead developer in 12.2009 and became a Senior developer and Project Leader (both Team Leader and Project Manager for human **gesture recognition engines** in Augmented Reality environments on embedded devices) since 01.2010: our team started with 6 and grown to 20 members in 2010; we presented the LFD project at CES-2011 in Las Vegas in a private section by Samsung DMC center, Visual Displays lab; another our project (Gesture Shot function: <http://www.samsung.com/global/mv900f/>) was commercialized in South Korea in August 2012.
04.2009-12.2009 – Design and development of prototypes for markerless gesture recognition on a single web camera with the aim of a remote control. Head and face features detection and tracking; motion-based hand gesture detection and human tracking in non lab environment; distance calculation using single camera (all 3 projections).
01.2010-06.2012 - **Team leading** and project management. Communication with customers and requirements specification clarification, project Architecture design and prototyping, project planning and reporting, work decomposition and delegation to the team members. Interviewing and hiring process. Risk management and team management, algorithms prototyping, project infrastructure establishment from scratch (SCM, I/BTS, CI systems; automation of the team workflow considering its security).
Besides being a team leader, I developed proposals for the visionary projects based on the Samsung Electronics infrastructure and participated in the hardware selection for our prototyping of upcoming products. In particular, our team has evaluated and recommended 3D cameras for human gesture recognition since 2010. Also, our team has developed an internal **computer vision framework** that is optimized for the specific embedded devices (e.g., Samsung photo cameras) having a small amount of RAM and slow CPUs (350 MHz+) in 2010th.
- 10.2008–04.2009 – “Symantec Corporation”. “PC Tools” company (were acquired by Symantec), position: *Software Engineer* (C++/Windows kernel developer).
PCTools Firewall driver porting to Windows x64 from x86.
- 04.2008–09.2008 – Security software company “PC Tools”, position: *Software Engineer* (C++/Windows kernel developer).
PCTools firewall development (Windows drivers) and testing, analyzed kernel crash dumps.
- 09.2006–03.2008 - Outsourcing software company Zoral Labs. QA department, position: *Software Engineer* (C++ developer / QA).
Testing automation of the antimalware system.
Development of the automated testing system for a distributed system of business process management.
- Summers of 2003-2005 internship at Softline company. Worked as SQA engineer in JavaEE and Web projects.

Publications and conference talks:

- Artem Lutov, Dingqi Yang, and Philippe Cudré-Mauroux. [“Bridging the Gap between Community and Node Representations: Graph Embedding via Community Detection”](#). IEEE International Conference on Big Data (BigData), Los Angeles, USA, 2019, Dec.
- Artem Lutov, Mourad Khayati, and Philippe Cudré-Mauroux. [“DAOC: Stable Clustering of Large Networks”](#). IEEE International Conference on Big Data (BigData), Los Angeles, USA, 2019, Dec.
- Artem Lutov, Mourad Khayati, and Philippe Cudré-Mauroux. [“Accuracy Evaluation of Overlapping and Multi-Resolution Clustering Algorithms on Large Datasets.”](#) IEEE International Conference on Big Data and Smart Computing (BigComp), Kyoto, Japan, 2019, Feb.
- Artem Lutov, Soheil Roshankish, Mourad Khayati, and Philippe Cudré-Mauroux. [Statix — statistical type inference on linked data](#). IEEE International Conference on Big Data (BigData 2018), Seattle, WA, USA, 2018, Dec.

- Artem Lutoy, Mourad Khayati, and Philippe Cudré-Mauroux. [Clubmark: a parallel isolation framework for benchmarking and profiling clustering algorithms on NUMA architectures](#). IEEE International Conference on Data Mining Workshops (ICDMW'18), Singapore, 2018.
- Andrea Martini, Artem Lutoy, Valerio Gemmetto, Andrii Magalich, Alessio Cardillo, Alex Constantin, Vasyl Palchykov, Mourad Khayati, Philippe Cudré-Mauroux, Alexey Boyarsky, Oleg Ruchayskiy, Diego Garlaschelli, Paolo De Los Rios, Karl Aberer. [ScienceWISE: Topic Modeling over Scientific Literature Networks](#). CoRR abs/1612.07636
- *System Analysis and Information Technologies*, Ukraine, Kiev, June 2005
Лутов А.В., Симоненко В.П. (Национальный технический университет "Киевский политехнический институт"). Метод пошагового конструирования при распределении работ в параллельных вычислительных системах (*Stepwise method of the tasks scheduling in parallel computing systems*). 28 июня - 2 июля 2005. Седьмая международная научно-техническая конференция САИТ (Системный анализ и информационные технологии) 2005 <http://sait.org.ua/>
<http://iee.org.ua/ua/conference/63/>
- *Issues of IT usage in the Economy*, Ukraine, Irpen, May 2004
Захариодакис Л., Лутов А.В., Широчин В.П. Повышение безопасности программ реального времени на основе анализа сетевых моделей (*Real time applications security improvement based on the analysis of network models*). Тр.5-ой міжнародної науково-практичної конференції "Проблеми впровадження інформаційних технологій в економіці". ДПА, Ірпень, травень 2004. – с. 238-239.
<http://lib.convdocs.org/docs/index-217064.html?page=2>

Patents:

- Application in Korea 10-2010-0043058 from 7/05/2010 and correspondent patent application in USA #20110274316: "Method and Apparatus for recognizing of user" patent #8396253, <http://patents.justia.com/inventor/artem-lutoy>
Related to content interaction in 3D using a single color 2D camera; 1 of 4 authors with 50% invention quota, Samsung Electronics patent
- Invention Application № a 2010 10142 from 16.08.2010, Patent № 98529 registered 25.05.2012, Ukraine: "The optical display system being controlled by touch gestures and the method for detecting surface touch by shadow analysis". Relates to remote control and smart networks; 1st of 2 Authors, Own private patent
- Declarative Patent (I didn't perform fixes required by the substantive examination to pass it for the Invention patent) № a 2011 03019 from 15.03.2011 related to optical display systems and picoprojectors, Ukraine; 1st of 5 Authors
- Coauthor of Patent #72913 from 27.08.2012, Ukraine.
- Invention Application and Declarative Patent (I didn't perform fixes required by the substantive examination to pass it for the Invention patent) № a 2012 08013 from 27.06.2012, related to services personalization and automation systems; 1st of 4 Authors, Own private patent

Completed courses, visited conferences:

- **Smart Analytics, Machine Learning, and AI on GCP**
(<https://coursera.org/share/7596d3631d741c1e79868a77c7026622>), 04.2020
- CUSO, UniFR: Social Signal and Multimodal Processing, 11.2018
- CUSO, UniFR: Film-making for scientists, 09.2016
- **CTI Business Creation** <http://startuptraining.ch/>, 05.2016
- VentureLab Startup Essential Workshops
(http://www.venturelab.ch/index.cfm?page=132978&event_id=4805), 02-05.2016
- Writing a successful business plan (<https://novoed.com/writing-business-plan-2016>), 03-04.2016

- CUSO, UniFR: Visual Thinking for Doctoral Researchers, 06.2016
- CUSO, UniFR: Bringing Your Presentation Skills to the Next Level, 04-05.2016
- CUSO, UniFR: Project Management for Research, 10.2015
- CUSO, UniFR: Conference & Seminar Skills, 09.2015
- Web Applications Engineering, CS 253 (<https://www.udacity.com/course/cs253>, completed after the due date), 2013
- Visitor of CVPR-2012, 06-2012, USA, RI.
- **Programming a Robotic Car**, CS 373 (<https://www.udacity.com/course/cs373/>), 04.2012
- MBA Harvard ManageMentor: Project Management, 03.2012, provided by Samsung Electronics
- Visitor of Augmented Reality conference held by Microsoft in Kiev, 17 of March 2012
- Visitor of MWC-2012, Spain, Barcelona 02.27-03.02
- MBA Harvard ManageMentor: Hiring, 01. 2012, provided by Samsung Electronics
- **Machine Learning**, 10.2011-12.2011: <http://www.ml-class.org> (<https://www.coursera.org/course/ml>)
- **Artificial intelligence**, 10.2011-12.2011: <https://www.ai-class.com> (<https://www.udacity.com/course/cs271>)
- MBA Harvard ManageMentor: Business plan development, 10.2011, provided by Samsung Electronics
- Agile Base Camp, Kiev, 04.2011
- Visitor of local conferences on Agile and MSVS, 2010
- Samsung Electronics: Open source licenses, 08.2009
- Symantec: Secure coding, 09-13.02.2009

Own projects:

- Own projects and pre-seed startups:
 - My website: <http://www.lumais.com/>
 - Armofab composite material featuring dynamic flexibility and the first consumer product, which is built using this material: <https://armofab.com/>
 - Temporal outdated projects:
 - [DOE for AR Glasses](#)
 - [Consuming estimation and discounting system](#)
 - Amisens keypad: <https://amisens.com/>
- Took part in the startup that was presented in IDCEE-2012 (Petcube), 10..2012: <http://idcee.org/>, habr post: <http://habrahabr.ru/post/159763/>
- Marriage website project: <http://artem-i-masha.appspot.com/>, target browsers: Firefox, Chrome, Opera (IE7/8 not supported)

Peculiarities:

- Analytical mentality
- Responsible, executive, assertive, well-balanced
- Hardworking, assiduous, easily master new technologies; lifelong learner
- Sociable, purposeful, and active; have experience in both communication and management

Additional information:

- Member of IEEE in 2011-2012, 2018-today
- PC member of ESWC2015 P&D
- Business trips and Visas:
 - 2014-20: Swiss residence permit B
 - 2019-20: USA Business Visa (B1/B2): BigData19 conference presenter, Los Angeles
 - 2019: Japan single-entry Visa: BigComp19 conference presenter, Japan, Kyoto

- 2018-9: USA Business Visa (B1/B2): BigData18 conference presenter, Seattle
- 2018, Nov: Singapore multi-entry Visa: ICDM18 conference presenter, Singapore
- 2012-17: USA Business Visa (B1/B2): business trip to CVPR12, Rhode Island
- 2012: Spain single-entry Visa: business trip to MWC12 exhibition, Barcelona
- 2009-11: 4 single-entry visas to South Korea: business trips to Samsung HQ DMC R&D center, Suwon
- I have a driving license of category B (Ukrainian & International, valid till 2022)
- I'm fond of Ballroom dances, skiing, diving and some other sports including gymnastics
- I was a trade-union of the group in the university (NTUU KPI), headman and a member of the student council there. Also, I organized a LAN (computer network) at our hostel there, connected it to other hostels and provided network access for the students

Profile last updated: 24.07.2020