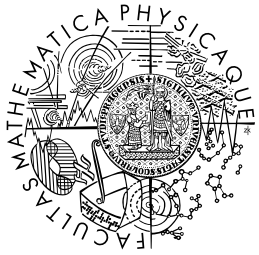


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Kateřina Vokálová

Node-attributed community detection

Computer Science Institute of Charles University

Supervisor of the bachelor thesis: Ing. David Hartman, Ph.D.
Study programme: Computer Science
Study branch: General Computer Science

Prague 2023

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank my supervisor Ing. David Hartman, Ph.D. for the consultations and interesting discussions about community detection. The next thanks are written in the native language of my family. Děkuji rodině za podporu v průběhu psaní a obecně při studiu. I also thank Borek Požár for the proofreading, grammatical corrections and support during the final month before the thesis submission. Thanks to Aneta Pokorná for the last-minute consultations of the results. The last thanks are to my friend Andrej Farkaš for motivating me to start writing; this work would not have been written this year without him.

Title: Node-attributed community detection

Author: Kateřina Vokálová

Institute: Computer Science Institute of Charles University

Supervisor: Ing. David Hartman, Ph.D., Computer Science Institute of Charles University

Abstract: Complex systems surround us in our everyday lives and their understanding can bring crucial insights into many fields. These systems consist of components (also known as communities) tied together. This thesis focuses on community detection in node-attributed networks, which are networks with some extra information about nodes.

At first, we introduced the necessary terminology and provided an overview of node-attributed benchmarks used for testing. Then we studied the setting of the benchmark and algorithm parameters and discussed the obtained results. The analysis was made on synthetic networks and focused on the impact of the benchmark and algorithm parameters on the results, which we then discussed. In particular, we have found that the algorithms are less influenced by mixing parameter when the size of the network is bigger. We also confirmed our expectation, that results will be negatively influenced by higher mixing and noise parameters.

Keywords: complex networks, community detection, node-attributed graphs

Contents

Introduction

Complex systems can be found all around us – relationships in society such as families or friendships, online communities on the Internet, communications infrastructure, protein-protein interactions, internet pages and their interconnectedness, functional modules in metabolic networks or activity of neurons in our brains ([6], [32]). The term *complex systems* originates from the observation that these systems cannot be studied separately by studying the system's components – these systems must be considered to be one complex component and studied accordingly ([6]). To study complex systems, we usually work with the complex systems in the form of complex networks.

Social networks are often mentioned in connection with the complex systems. On these networks, it can be easily seen that complex systems can consist of some densely connected parts. In the case of social networks, the parts can be groups of friends, families, etc. Inspired by the social groups, these parts are called *communities*. If we want to understand some complex network, it can be useful to uncover its community structure, bringing us more profound network knowledge.

This work focuses on finding the communities in networks with some extra information about nodes. These networks are called *node-attributed networks*. Having a social network, we could have information about the places the people like or the sports clubs they visit. In the first chapter, we will introduce community detection and present some essential terminology regarding networks in general, communities and node-attributed networks. Afterwards, we also present the division of algorithms into early fusion, late fusion and simultaneous fusion methods, which is helpful for the latter comparison analysis.

As community detection began to be the focus of more scientists, numerous community detection algorithms started to appear. It became necessary to distinguish the algorithms according to their accuracy, memory requirements or computation time. The second chapter describes the tools usually used when comparing the algorithms, mainly the graphs and graph models used in comparative studies and the metrics used for the evaluation of the algorithm results. The last part of the second chapter presents an overview of works related to the community detection algorithms comparison.

In the last two chapters, we carry out a comparison analysis of the algorithms used for community detection in node-attributed graphs. The third chapter introduces the benchmarks and algorithms we use, together with the values of their parameters. We also show some problems encountered during the algorithm analysis. The fourth chapter is devoted to the presentation of the obtained results and to the discussion of the results. We mainly discuss the influence of the benchmark parameters on the algorithms' results.

1. Community detection

The community detection problem probably developed around the 1960s from a problem known as *grouping problem* or *graph partitioning*. The goal of the grouping algorithms is to divide a large number of objects or persons into smaller mutually exclusive and collectively exhaustive groups with the condition that the members of each group are as similar as possible ([30], [87]). On the other hand, the graph partitioning problem focuses on partitioning the nodes of a weighted graph into some subsets of given sizes such that the weight of the edges among the subsets is minimal. The real-world use of the solution is, for example, in assigning the components of electronic circuits to circuit boards to minimise the number of connections between boards ([32]). Kernighan and Lin [46] worked on this problem with exceedingly good results – the Kernighan-Lin algorithm is one of the simplest and most frequently used algorithms for graph partitioning. According to Newman and Girvan [62] and Newman [61] (Section 11.4), the Kernighan-Lin algorithm is one of the best algorithms. However, Fortunato [32] suggests combining it with other techniques to improve its accuracy.

However, a significant difference exists between graph partitioning and *community detection*. The graph partitioning algorithms expect that the number of subgraphs/partitions is being decided a priori and that their sizes (in terms of the number of nodes) are equal. Some algorithms, such as Kernighan-Lin or Huff algorithm ([43], based on Kernighan-Lin), provide an option to partition into unequal-sized sets of nodes. On the contrary, community detection focuses on discovering the native community structure of the network (in this work, we consider graph and network synonyms, see Section 1.1). Therefore, the number of communities is not predetermined, and the algorithm must detect the communities only from the network structure ([61], Section 11.2.1).

Wong [88] was among the first to address an important question: "How many separate well-connected subgraphs are in the given graph?" He informally introduced the *clusters of nodes* as densely-connected subgraphs separated from other such subgraphs by relatively few cross-links. Using the formal definition, which regarded *high-density clusters*, Wong's algorithm can produce a graph partition into clusters without the number of clusters being known a priori, and the algorithm could be considered the predecessor of the community detection algorithms.

This chapter will introduce the community detection process and present the necessary terminology, starting with communities and their definitions in Section 1.1 and some information about the community sizes in Section 1.2. The following section (1.3) presents the *node-attributed graphs* – a subset of graphs providing extra information about the nodes. Moving into community detection algorithms, as many algorithms exist and new ones appear every year, it could be helpful to systematically divide them into some classes. The last section (Section 1.4) states some of the most used divisions.

1.1 Communities

First, it is essential to introduce the graph terminology used in this work. From now on, we will consider "network" and "graph" to be synonyms ([61], Section 6.1). The network (or graph) consists of some set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ (also known as vertices) connected with edges (also called links), $E = \{e_{ij}\}$ formally $G = (V, E)$. The overview of terminology in the community detection field can be found in the work of Bothorel et al. [11], Table 1.

The terminology change from subgraphs, partitions or clusters to communities occurred around 2000. Flake et al. [31] define *web communities* as a collection of densely connected web pages, and Radicchi et al. [74] state a more general definition of communities (distinguishing between strong and weak communities), which coincides with the definition in [31]. In all the above definitions, communities are mutually exclusive and collectively exhaustive groups of nodes.

Definition 1 (Community in a strong and weak sense by Radicchi et al.)**.** *Let $S \subset G$ be a subgraph of an unweighted and undirected graph G. Let $k_i^{in}(S)$ denote the number of edges connecting node i to other nodes belonging to S and $k_i^{out}(S)$ denote the number of edges from node i towards nodes in the rest of the graph $G \setminus S$.*

*S is a **community in a strong sense** if the following holds:*

$$\forall i \in S : k_i^{in}(S) > k_i^{out}(S).$$

*S is a **community in a weak sense** if the following holds:*

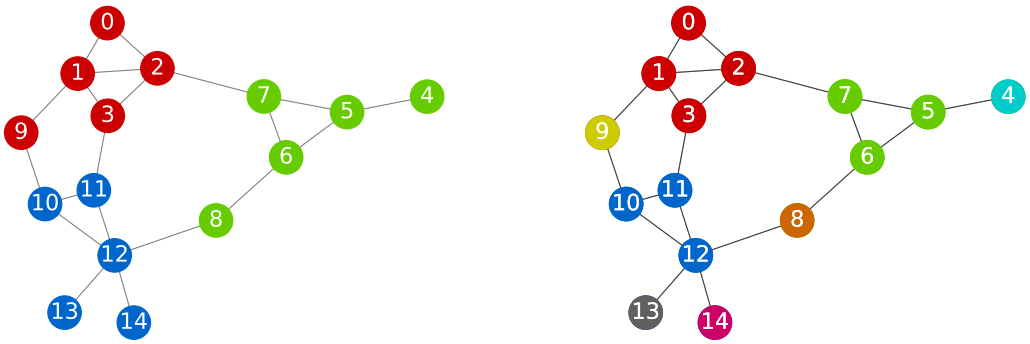
$$\sum_{i \in S} k_i^{in}(S) > \sum_{i \in S} k_i^{out}(S).$$

The idea of strong and weak communities is furtherly refined by Hu et al. [41], but in this case, the communities are defined using less strict constraints. However, Fortunato and Hric [33] found that the above definitions are not satisfactory for some specific settings and proposed an improved definition based on the probabilities of edges' existence. Fortunately, having the exact definition of a community is not necessary (and the canonic definition does not even exist). Most community detection techniques do not use a precise definition (but some algorithms do, see [74]). Nevertheless, defining the communities could be helpful when checking the accuracy of the final results of partitioning ([33]). In summary, communities can be defined in many ways, and the exact definition does not exist. For our purposes, we will loosely define a community according to Hu et al. [41] as a subset of nodes more densely connected to each other than to the rest of the graph.

Sometimes, the concept of communities as mutually exclusive subgroups is not suitable. For example, having a social network at a college, a person can be a member of a football and a baseball team, or a scientist can write articles in more than one field. Palla et al. [68] point this out and introduce the idea of overlapping communities (also described in [61], Section 14.7.1). Nonetheless, we will focus only on the non-overlapping communities in this work. For an interested reader, Mittal and Bhatia [58] distinguish more community types, such as dynamic, dense or isolated communities.

1.2 Community sizes and power-law distribution

As we have seen in the previous section, the definitions of communities are ambiguous. When constructing an algorithm for community detection, it is crucial to choose some condition for which will the algorithm finishes. Thus, various algorithms can detect communities differently, as shown in Figure 1.1.



(a) Similar sized communities

(b) Power-law distribution sized communities

Figure 1.1: Differences in community detection

However, communities of real-world networks usually look like in Figure 1.1b – many small communities and only a few bigger communities. The distribution of the community sizes often follows a power law (as does the distribution of node degrees). In the following section (1.2.1), we will present some information about the power-law distribution and the networks whose node degrees follow the distribution (*scale-free networks*). Section 1.2.2 analyses the distribution of community sizes and states some existing insights about the connection to power law.

1.2.1 Power-law distribution and scale-free networks

The distribution

$$p(x) = Cx^{-\tau}$$

with $C = \mathrm{e}^c$ is called a *power-law distribution* with exponent τ ([65], Equation (1)). The constant C is for fixed τ determined by the normalisation requirement ([65], Section III.A), so the power-law distribution can be denoted as

$$p(x) \sim x^{-\tau}$$

([6], Section 4.2). If we take a logarithm of both sides, it can be seen that power-law distribution scales linearly on a log-log plot (it is a straight line). Power-law distribution is also called a *scale-free distribution* because it is a distribution that is the same regardless of the scale we look at on it ([65]). For example, having values of node degrees, if 10 is half less common than 2, then 100 is also half less

common than 20. In fact, power-law distribution is the only distribution with a scale-free property (see [65], Section III.E for the proof).

In the following paragraphs, we will use a *(complementary) cumulative power-law distribution*. Generally, the cumulative distribution $F(x)$ shows the probability that X (random variable) has a value less than or equal to x. The complementary cumulative distribution equals $1 - F(x)$ and shows the probability that X has a value greater than or equal to x:

$$\Pr(X \geq x).$$

The complementary cumulative power-law distribution is denoted as (assuming $\tau > 1$):

$$P(x) = \Pr(X \geq x) = \int_x^\infty p(x')\mathrm{d}x' = C \int_x^\infty x'^{-\tau}\mathrm{d}x' = \frac{C}{\tau - 1} x^{-(\tau-1)}.$$

The cumulative distribution function also follows a power law with an exponent $\tau - 1$ ([65], Section II., [61], Section 8.4.1).

A network whose node degrees follow power-law distribution is called *a scale-free network* ([5], [7], [2], Section VII., [61], Section 10.4, [6], Section 4.2). Many studies ([76], [29], [59], [69], [65], [57], [80], an overview of networks in [61], Table 8.1 and Section 8.3) have reported real-world networks with the scale-free property, for example, Internet at the router level, protein interactions, e-mail network, citation network and many more ([6], Section 4.5, Figure 4.10). Most degree distributions observed are fat-tailed, meaning there are often nodes with very high degrees. These nodes are called *hubs* ([61], Section 10.3, [6], Section 4.3). If we consider fat-tailed distribution as a subset of heavy-tailed distribution, power-law distribution is the best-known approximation of a fat-tailed distribution. Many graph benchmark models, such as LFR and acMark (see Section 2.1), use the power-law node degree distribution.

1.2.2 Community size distribution

Moving on to communities, some of the graph benchmark models (Section 2.1) use power-law community size distribution. We have searched for some articles supporting the chosen community size distribution, and we have not found any systematic study. However, we have found a few reports examining community size distributions of real-world networks ([57], [92], [6], Section 9.2). The overview of the reports, which state some value for the cumulative distribution exponent, is shown in Table 1.1. The reports usually find a fat-tailed community size distribution which can be well described by a power-law distribution with an exponent ranging from 1 to 3 ([32], Section XVI.).

The following authors base their algorithms, benchmarks or results on the power law distribution property of community sizes: Lancichinetti et al. [50] (cite [40], [19], [68], [24]), Clauset et al. [19] (cite [4], [60]), Fortunato [32] (cites [74] [19], [64], [68], [24]). In Section 2.1, we will consider graph benchmarks that support power-law degree distribution. However, a further systematic study about community size distribution would be necessary to confirm or reject the connection with power-law distribution. Newman [63] states that several alternative

Network	**Exponent**	**Study**	**Notes** (s stands for community size)
E-mails at URV[1]	0.48[2]	[40], [4]	$2 < s < 100$
Jazz musicians	0.48[2]	[37], [4]	$200 < s < 1000$
ArXiv Physics E-Print C.M.[3]	≈ 1.6 or 2	[60], [74]	
Statistical Physics FisEs	1.07[2]	[4]	$s < 1000$
ArXiv Mathematical-Physics	1.07[2]	[4]	$s < 1000$
ArXiv Other Physics (3 networks)	0.97[2] 0.54[2]	[4]	$1 < s < 60$ $60 < s < 1000$
LA[4] E-Print collaboration	1.6[5]	[68]	
Word-association	1[5]	[68]	
Protein interaction	between 1 and 1.6[5]	[68]	
ArXiv Physics E-Print	0.5	[24]	

[1] University at Rovira i Virgili

[2] Value for cumulative community sizes – the sizes were taken over all levels of the *dendrogram*. A dendrogram is a hierarchical tree which represents the nested hierarchy of possible partitions of a network into communities (see [62] or [19], Section II). It usually shows a process of dividing a network into smaller and smaller parts (or a process of connecting smaller parts into bigger ones). Many algorithms detecting communities use dendrograms (see Greedy algorithm, Section 3.1.1).

[3] Condensed Matter

[4] Los Alamos

[5] The sizes of overlapping communities were taken.

Table 1.1: Exponents of cumulative power-law distribution

distributions can be confused with power-law and suggests using Kolmogorov-Smirnov or other goodness-of-fit tests for examining the power-law behaviour. (Newman [61] also presents an Equation (8.6) which could be used for computing the exponent τ directly from the data.)

1.3 Node-attributed graphs

The communities can be detected in many kinds of networks. In this work, we focus on community detection in node-attributed networks. The main difference compared to regular networks is that these networks carry some extra information

about the nodes in the form of attributes. For clarification, the information is stored in the nodes (usually as an attribute vector). Community detection in node-attributed graphs aims to find communities with homogenous structure (high edge density) and homogenous attributes (attribute values are similar). Some early inspiration for using the attributes to improve the algorithm accuracy can be found in graph mining ([11], [38]).

Bothorel et al. [11] in Section 3.1 explain two approaches to the representation of graphs with attributes – either the nodes have attribute vectors or the graph $G_s = (V_s, E_s)$ is augmented by a bipartite graph $G_a = (V_s \cup V_a, E_a), E_a \subseteq V_s \times V_a$ making an augmented graph $G = (V_s \cup V_a, E_s \cup E_a)$. Both these approaches appear in articles; the first one being used recently. Some of the first descriptions of attributed networks are from Zhou et al. [93] (attribute augmented graph), alternatively from Yin et al. [91] (Gong et al. [38] name it a *social-attribute network*). They suggest creating so-called *attribute nodes* apart from the *structural/person nodes* (using the second representation approach). Zhou et al. [93] further formally define an attributed graph with nodes having the attribute vectors (the first representation approach). A similar definition also appears in [20], and an extensive definition was introduced in [89]. Newer and recently used definitions can be found in articles by Falih et al. [27] and Chunaev [14] (also in [15] in a slightly modified form).

Definition 2 (Node-attributed social network)**.** *Node-attributed social network (or node-attributed graph) is a triple $G = (V, E, \mathcal{A})$, where*

- *$V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes (vertices),*
- *$E = \{e_{ij}\}$, $i, j \in \{1, \ldots, n\}$ is the set of edges (e_{ij} is the edge between nodes v_i and v_j, for undirected networks $e_{ij} = e_{ji}$),*
- *$\mathcal{A} = \{a_{ik}\}$, $i \in \{1, \ldots, n\}$, $k \in \{1, \ldots, d\}$ is the set of attribute sets associated with the nodes in V, the number of attributes is equal to d.*

The size of the set V is the number of nodes, $|V| = n$. The size of the set E is the number of edges, $|E| = m$.

The pairs (V, E), $(V, \mathcal{A})$ are called the structure *and the* attributes *of a node-attributed graph G, respectively.*

Node-attributed networks are not the only ones to provide additional information embedded in the network topology. Interdonato et al. [44] write about *feature-rich networks* in which they include node-attributed networks besides heterogeneous information networks, multilayer networks or probabilistic networks.

1.4 Fusion methods and algorithms

There exist many algorithms for node-attributed community detection. We will divide the algorithms into some categories or classes for better clarity. A brief overview of existing classifications can be found in [18]; we will also present some others. Bothorel et al. [11] propose a classification schema which includes weight-based, walk-based, distance-based and probabilistic methods. Vieira et al. [86] use relatively narrow descriptions of algorithms as classes (algorithms adjusting graph

before community detection, algorithms optimising a quality function, algorithms using a unified distance and lastly, the hybrid approaches). Finally, Mittal and Bhatia [58] classify the algorithms according to the technique used when detecting a community (4 types of algorithms: modularity, information theoretic, network algorithms and hierarchical).

All the previous classifications are similar – they present many classes which are pretty narrow (with only a few algorithms). They are missing some higher-level schema that would partition the algorithms into a couple of relatively broad categories (with some subclasses). We will now state two proposals for the division, which are done with respect to the techniques used in the algorithms but provide an extra higher level of hierarchy.

Falih et al. [27] distinguish algorithms into three categories based on their methodological principle:

- topological-based clustering (the attributes are used as additional structural information and can be used in order to change the topology of the graph),
- attributed-based clustering (the structure and the attributes are merged into a global similarity/distance and then processed by classical clustering algorithm),
- hybrid approach (the attributes and the structure are considered separately, clustered, and the results are merged by ensemble clustering method).

On the other hand, Chunaev [14] groups the algorithms according to a moment when the structure and the attributes are being fused in the community detection process:

- early fusion methods (fuse structure and attributes before the community detection process),
- simultaneous fusion methods (fuse structure and attributes simultaneously with the community detection process),
- late fusion methods (cluster structure and attributes separately and fuse the partitions obtained).

We can see that the classifications partly overlap ([18]) – hybrid approach overlaps with late fusion methods, topological-based clustering with simultaneous fusion methods and attributed-based clustering with early fusion methods.

We will divide the community detection methods into three classes according to Chunaev [14]. We chose this division because the survey is extensive and presents almost every algorithm until the middle of 2019. Therefore we believe the division is the most useful one to compare algorithms and create a tool for automatic comparison. With these classes, it is easier to create comparative studies. For example, having two early fusion algorithms, we can replace the fusion part of the first with the fusion part of the second. Then (supposing the output format of these parts is compatible), we can compare which of those fusion parts is better relatively to our input graph. This division is also used in [53] or [79].

2. Comparing algorithms

As more node-attributed community detection algorithms started to appear, the need to compare algorithms and choose the best among them arose. At first, it is necessary to have some networks on which we would test the algorithms. Section 2.1 provides insight into the node-attributed networks used for detecting communities (focusing on the synthetic networks generated by some models). The second essential part of comparing the algorithms is evaluating the given results – how accurate the detected communities are. The overview of such methods used for evaluation is given in Section 2.2. The last section (Section 2.3) presents comparison studies related to our work.

2.1 Graphs and graph models

To test the algorithms thoroughly, we need a relatively huge set of graphs with known communities. According to Chakraborty et al. [12], these graphs are called *graphs with ground truth communities* (also written as ground-truth). There exist real-world node-attributed graphs with ground truth (the most popular are mentioned in [14], Section 5.1), but the number of these graphs is not large (Chunaev [14] states about 20 datasets). Moreover, Peel et al. [70] recommend careful usage of real-world node-attributed networks for community detection. They comprehensively analyse problems related to treating the nodes' metadata (attributes) as a ground truth. The main problem is that the organisation of real-world networks and their division into communities can correlate with both observed and unobserved data. Nevertheless, the real-world networks are not condemned completely; Peel et al. suggest the metadata could be used to gain insights into real-world network organising principles.

Ceasing from real-world networks, we will move on to the artificially generated ones. These synthetic networks are generated by graph models (also called *benchmarks*), and their nodes are partitioned into ground truth communities (the partition is called *a planted partition* according to [70]). There as well exist some benchmarks which generate random graphs without community structure. However, we will not consider these, although it could be interesting to test the algorithms on these benchmarks to see how robust the algorithms are. This section will focus on the benchmarks generating graphs with some community structure (and providing the ground truth communities).

At first, we will look at a Girwan-Newman benchmark (GN, Section 2.1.1), which is likely the first benchmark massively used in the beginnings of community detection. Secondly, we will introduce the Lancichinetti-Fortunato-Radicchi (LFR, Section 2.1.2) benchmark, the most famous and most used benchmark in community detection. It produces networks with community structure, however, without node attributes. Both these benchmarks are realisations of planted l-partition model from Condon and Karp [22]. We do not consider other models such as the Erdős-Rényi model, Stochastic block model or Barabási-Albert model because they are missing community structure, power-law node degree distribution or power-law community size distribution. (More information about the power-law distribution and its relation to node degrees and community sizes can

be found in Section 1.2.) In Sections 2.1.3 and 2.1.4, we introduce two benchmarks producing node-attributed networks. Other benchmarks with node attributes are briefly summarised in Section 2.1.5, together with other methods of obtaining a node-attributed network.

2.1.1 Girwan-Newman benchmark

The first and the most famous benchmark graph is the Girvan-Newman (GN) by Girvan and Newman ([36], [62]). The generated graph consists of only 128 nodes divided into four equally sized communities (32 nodes each). The edges between nodes in the same community are generated with probability p_{in} and the edges between nodes belonging to different communities with probability p_{out}. It holds that $p_{in} > p_{out}$ and p_{in} is chosen to keep the average degree of node equal to 16. This graph model does not generate nodes with attributes, the structure is simple, the network size is relatively small, the expected node degrees do not differ, and the sizes of communities are equal. This model also does not satisfy the power-law distribution of community sizes.

2.1.2 LFR benchmark

The GN benchmark has long since been surpassed by the Lancichinetti-Fortunato-Radicchi benchmark (LFR) introduced by Lancichinetti et al. [50] (case of undirected and unweighted graphs) and by Lancichinetti and Fortunato [49] (case of directed and weighted graphs). The size of the generated network (in terms of the number of nodes) and the mixing parameter (which determines the fraction of edges connecting a node to nodes in different communities) can be chosen. Some of the parameters are the minimum and maximum node degree. It also has two other parameters – exponents of power-law distributions. The first is for the node degree distribution, and the second is for the community size distribution. It is clear that the LFR benchmark surpasses the GN benchmark with the variability of network and community sizes, but it still does not support node attributes. For an interested reader, the comparison of several community detection algorithms on the GN and LFR benchmarks can be found in [48].

2.1.3 acMark

acMark was introduced by Maekawa et al. [54] in 2019. It is implemented in Python, and its source code is available on GitHub.[1] It supports arbitrary distributions for the node degrees, the cluster sizes and the attribute values (power-law, uniform and normal distributions are mentioned in [54]). It also allows two kinds of attributes – discrete or continuous – and enables one to choose the ratio of continuous parameters following different distributions. Of course, there are parameters for the number of nodes, edges and attributes, balancing inter-edges and intra-edges (edges from node to other communities/to its community), and eventually, widths of uniform distribution or deviations of normal distribution if used. Finally, the runtime of acMark scales linearly with the number of edges. (A complete resume of parameters can be found in [54], Table 2.)

[1] `https://github.com/seijimaekawa/acMark`

2.1.4 X-Mark

The most recent model is X-Mark from 2021 by Citraro and Rossetti [18]. It is also implemented in Python with source code on GitHub.[2] It is based on the LFR benchmark (Section 2.1.2), so the node degrees and community sizes are taken from power-law distributions with user-specified exponents. Section 3.2.1 summarises the X-Mark benchmark parameters. There are parameters for the mixing, the number of nodes and the average degree of nodes. X-Mark supports creating attributes of one chosen type (categorical or continuous). If we choose categorical attributes, we must specify the number of values in the domain of the attributes and the noise parameter. The noise is used as a probability that a node will have a different categorical attribute than its community. If we choose continuous attributes, we must specify the parameter for the number of peaks of a multimodal distribution and the parameter for the standard deviation of a normal distribution. For each community, a community label is chosen uniformly randomly from the peaks of the multimodal distribution. Then, the community nodes' labels are taken from a normal distribution with the given standard deviation. The mean of the normal distribution is the community label.

2.1.5 Other benchmarks and methods

The acMark and X-Mark benchmarks are not the only ones able to generate node-attributed graphs. We would like to mention the following node-attributed benchmarks with community structure – **LFR-EA** by Elhadi and Agam [26] and **ANC** by Largeron et al. [51]. The LFR-EA generates attributes for LFR benchmark networks. The attributes are only categorical (discrete). All the nodes in a community share the same attribute value except for uniformly randomly chosen nodes which host the noise (the probability is controlled by the noise parameter, which can be different for each attribute). The LFR-EA was used for testing in some works, e.g. in [71], [72] and [8].

The ANC allows the choice of the network size, maximum number of edges within a community and between communities added to a node, minimum number of edges in the network, number of communities and maximum size of a community. Considering the attributes, we can set the standard deviation for each attribute (the attributes are continuous) and threshold for community attributes homogeneity, which causes a node to be assigned to a random community instead of a community with nodes having similar attributes. The main difference between ANC and the benchmarks mentioned above is that ANC first creates nodes with attributes and then creates communities according to the similarity of nodes. Therefore, it is impossible to control the community sizes and the positioning of structure and attributes. Despite that, the ANC was used for testing algorithms in [28] and [86].

Some authors (e.g. Chunaev et al. [16]) do not require a benchmark generating a node-attributed network. Instead of this, they generate two weighted graphs – the first as the structure graph (edges in this graph symbolise edges in a network) and the second as the attribute graph (edges and weights symbolise attributive closeness of nodes).

[2] `https://github.com/dsalvaz/XMark`

2.2 Result evaluation

Once the communities of the network are detected, we have to evaluate the algorithm's accuracy – how well has the algorithm detected the communities. For further purposes, we will suppose the ground truth communities are available. The comparison between found communities and ground truth is usually realised using some metrics. An exhaustive survey of metrics was carried out by Chakraborty et al. [12]. Many validation metrics for non-overlapping communities using the ground truth are summarised in [12], Section 3.1. An older survey of evaluation metrics (in this case, for partitioning and clustering) can be found in [35], Chapter 17. In this section, we will briefly present metrics that are used in most comparative studies – Adjusted Rand Index (ARI) and Normalised Mutual Information (NMI).

Notation

We will use a notation (see Table 2.1) corresponding to the ones in [12] and [55], Equation 16.1, supposing detection of communities is performed on a network with the set of nodes V.

Notation	Description
$\Omega = \{\omega_1, \omega_2, \cdots, \omega_K\}$	set of detected communities
ω_k	set of nodes in the detected community k
$C = \{c_1, c_2, \cdots, c_J\}$	set of ground truth communities
c_j	set of nodes in ground-truth community j
$N = \lvert V\rvert = \sum_{k=1}^{K} \lvert\omega_k\rvert = \sum_{j=1}^{J} \lvert c_j\rvert$	number of nodes in the network
$N_X = \lvert X\rvert$	number of nodes in the set $X \subseteq V$
$N_{XY} = N_{YX} = \lvert X \cap Y\rvert$	number of nodes in the intersection of the sets $X, Y \subseteq V$

Table 2.1: Notation used for result evaluation

2.2.1 Adjusted Rand Index

ARI was presented by Hubert and Arabie [42]. It is a "chance-corrected" version of Rand Index (RI) from Rand [75]. The "correction of chance" is meant in the sense of having a constant index value when an appropriate null model is used. In this case, a null model is considered to be a random partitioning to a fixed number of communities of fixed sizes.

ARI is defined as below ([12], Section 3.2, Equation 69),

$$ARI(\Omega, C) = \frac{\sum_{k,j}\binom{N_{\omega_k c_j}}{2} - \frac{\sum_k \binom{N_{\omega_k}}{2}\cdot\sum_j \binom{N_{c_j}}{2}}{\binom{N}{2}}}{\frac{1}{2}\left(\sum_k \binom{N_{\omega_k}}{2} + \sum_j \binom{N_{c_j}}{2}\right) - \frac{\sum_k \binom{N_{\omega_k}}{2}\cdot\sum_j \binom{N_{c_j}}{2}}{\binom{N}{2}}}.$$

It is a symmetric metric ($ARI(\Omega, C) = ARI(C, \Omega)$). Its maximum value is 1, showing that partitions are the same. When a value equals 0 or less, the partitions are as similar as two random partitions would be.

2.2.2 Normalised Mutual Information

NMI was introduced by Strehl and Ghosh [82] as $[0, 1]$-normalised mutual information criterion (a normalised version of mutual information from [83], Equation 5). NMI is defined as follows ([82], Equation 1):

$$NMI(\Omega, C) = \frac{\frac{1}{N}\sum_{k=1}^{K}\sum_{j=1}^{J} N_{\omega_k c_j} \log_{K\cdot J} \frac{N\cdot N_{\omega_k c_j}}{N_{\omega_k}\cdot N_{c_j}}}{\frac{1}{2}},$$

which can be also written as ([34], Equation 2, [55], Equation 16.2, [47], Section 3.4, ...):

$$NMI(\Omega, C) = \frac{I(\Omega, C)}{(H(\Omega) + H(C))/2},$$

where I is mutual information,

$$I(\Omega, C) = \sum_{k=1}^{K}\sum_{j=1}^{J} \frac{N_{\omega_k c_j}}{N} \log \frac{N \cdot N_{\omega_k c_j}}{N_{\omega_k} \cdot N_{c_j}},$$

and H is entropy,

$$H(\Omega) = -\sum_{k=1}^{K} \frac{N_{\omega_k}}{N} \log \frac{N_{\omega_k}}{N}$$

and

$$H(C) = -\sum_{j=1}^{J} \frac{N_{c_j}}{N} \log \frac{N_{c_j}}{N}.$$

The biggest problem of NMI is that it is not a true metric because it does not meet the triangle inequality ([12], Section 8.2). Its maximum value is 1, showing that the partitions are the same. The bigger NMI value is, the more similar the partitions are.

2.3 Related works

We would like to summarise the progress made in algorithm comparison and point out some interesting studies. This work focuses mainly on testing on artificial networks. One of the earliest attempts in this field was made by Danon et al. [25] who used the GN benchmark. Other algorithm comparisons followed later (mostly using GN and LFR), for example, [73], [48], [66], [67], [90] or [12].

However, all these studies are focused on networks without attributed nodes. As for node-attributed community detection, we found that new algorithms are usually tested just on real-world networks against chosen state-of-the-art algorithms ([93], [23], [26], [89], [21], [3], [1], [15]). Nevertheless, Chunaev [14] concludes that it is almost impossible to determine which algorithms are state-of-the-art.

Only some authors use synthetic networks in their studies. Combe et al. [21] use attributes following a normal distribution with a different mean for each group of nodes, and Akbas and Zhao [1] use node attributes following Gaussian and uniform distribution. Pizzuti and Socievole ([71], [72]) and Berahmand et al. [8] use LFR-EA, Meng et al. [56] use LFR with three specifically generated attributes and Sánchez et al. [84] use LFR with attributes following a Gaussian distribution. Next, we have X-Mark used by Citraro and Rossetti [18] and ANC used by Falih et al. [28]. We found only two truly comparative studies of node-attributed community detection algorithms on artificial networks – Falih et al. [27] (5 algorithms) and Vieira et al. [86] (7 algorithms) – and both are using the ANC benchmark.

3. Comparison analysis

In this chapter, we present a comparison analysis we have carried out. The comparison is done entirely on synthetic networks and focuses on the impact of different parameters of benchmarks and algorithms on the results. Section 3.1 presents the benchmarks and algorithms used for comparison, and Section 3.2 provides a view on the setting of the algorithm parameters – X-Mark benchmark parameters in Section 3.2.1, fusion algorithms parameters in Section 3.2.2, clustering algorithms parameters in Section 3.2.3 and simultaneous fusion parameters in Section 3.2.4. To provide a complete view, Section 3.3 roughly informs which algorithms have been faster than others and which have been slower. However, comparing the running times of algorithms was not our focus. Finally, Section 3.4 presents some problems we faced during the comparison analysis.

3.1 Benchmarks and algorithms

This section will present the benchmarks and algorithms used in our comparison analysis. We will also present the metrics used to evaluate the found communities and compare them to the ground truth.

Our comparison was done on synthetic networks. We have chosen the X-Mark benchmark because the model is as general as possible and easy to use. X-Mark can model the situations where attributes either align or do not align with the structure. Moreover, the X-Mark benchmark code is publicly available (see Section 2.1.4). We have slightly rewritten the code but maintained its functionality. The X-Mark benchmark was also chosen over the acMark benchmark because even though it has fewer parameters than acMark, the different parameter settings cover most of the interesting cases of structure and attributes relationship. We have used our testing framework for community detection written in Python (available on request) to automatise testing for various combinations of parameters. It provides methods for testing early fusion, simultaneous fusion and late fusion methods (division according to Chunaev [14], see Section 1.4).

3.1.1 Algorithms

We have decided to focus our work on the Python implementation of the algorithms. The algorithms were either already implemented (taken from NetworkX or igraph library or available by a respective author) or written by us.

For the fusion in the early fusion approach, we have used metrics which make a weighted graph without attributes from an unweighted attributed graph. Then, we ran a standard clustering algorithm for weighted graphs for obtaining communities. We have used the following metrics:

- **Node Attribute Similarity** (NAS) ([80], [81], [39] as *general similarity coefficient*, [45], Section 2.9, Equation (19) and (26) as *simple matching approach*, also [10], Section 4.1 only for the categorical attributes) – The similarity $s(u, v)$ between two nodes u, v is computed from the attributes of the nodes. In the beginning, $s(u, v) = 0$, and we compute the final

similarity value by comparing the attributes individually and modifying $s(u, v)$ accordingly. Having attribute values i, j (supposing the values of this attribute range from 1 to K) we proceed as follows:

- for categorical attributes, $s(u, v)$ += 1 if $i = j$,
- for continuous attributes, $s(u, v)$ += $1 - (1/K) \cdot |i - j|$.

Finally, the similarity is normalised to weight using the number of attributes a:

$$w_a(u, v) = \frac{s(u, v)}{a}.$$

If the nodes were not connected in the former graph, the weight is set to 0 independently of the attributes.

- **Node Path Similarity** (NPS) ([20], Section IV.B) – In this algorithm, the number of edges on the shortest path between all pairs of nodes u, v (denoted as $d(u, v)$) is computed. The maximum value of d (the maximum length of the shortest path between two nodes of the graph, also called the *diameter*) will be further denoted as $d_{\max}$. If there does not exist a path between the nodes u, v (they are in different network components), $d(u, v) = d_{\max} + 1$. Finally, the values of d are normalised and used as edge weights:

$$w_d(u, v) = \frac{1 + d_{\max} - d(u, v)}{d_{\max}} = 1 - \frac{d(u, v) - 1}{d_{\max}}.$$

The bigger the weight between nodes is, the closer the nodes are for the community detection process. If the nodes were directly connected in the former graph, then $w_d(u, v) = 1$. If they were not directly connected, then $1 > w_d(u, v) \geq 1/d_{\max}$, and if they were disconnected, then $w_d(u, v) = 0$

- **Mixed Similarity** (MS) (suggested with specific algorithms in [23], [20], generally in [93], [27]) – Our implementation of MS is based on NAS and NPS with parameter α:

$$w_m(u, v) = \alpha w_a(u, v) + (1 - \alpha) w_d(u, v).$$

There are many more possible metrics for attributes which could be used in the fusion process instead of NAS, the overview of many similarity measures for categorical data can be found in [10] and for both categorical and continuous data in [35], Section 6. Meng et al. [56] carried out experiments with different metrics for categorical attributes on real-world and synthetic networks. In this experiment, the simple matching coefficient (approach) surpassed the famous Jaccard similarity ([52]) and Cosine similarity ([13], Section 3.2). Therefore, we have chosen the simple matching coefficient (NAS) for the attribute similarity.

We have chosen the following algorithms for clustering in the early fusion approach. Many of these algorithms use *modularity*, described in Section 3.2.3.

- **Greedy modularity** (Greedy) ([19], taken from NetworkX library[1]) – The algorithm is based on the modularity maximalisation. It starts with all

[1] `https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html`

nodes in separate communities. Then, it repeatedly joins two communities whose union produces the largest increase of modularity. It obtains one big community after $n-1$ steps (for a network of size n). The process starting with many small communities and ending with one large community can be represented with a tree called *dendrogram*, where the nodes represent communities, and the edges represent the union of two communities. Finally, the algorithm chooses the partitioning according to the tree level with the highest modularity.

- **Louvain** ([9], taken from NetworkX library[2]) – The algorithm consists of two phases which are repeated until an objective function (modularity) is maximised. The algorithm starts with all nodes in separate communities. In the first phase, all nodes are moved to a neighbouring community if the move increases modularity. The algorithm starts the second phase if no nodes can be moved to increase the modularity. In this phase, every community is contracted into one node, and the edges between the new nodes are weighted with the sum of the edges between former communities. Then, both phases are repeated on the newly created network until no more nodes move.

- **Leiden** ([85], taken from igraph library[3]) – The algorithm is based on Louvain algorithm, but it adds a new phase between the original ones – refinement of the partition. The function that the algorithm optimises is called the Constant Potts Model.

- **Infomap** ([78], [77], taken from igraph library[4]) – The algorithm repeats the same two phases as Louvain algorithm until its objective function is maximised. In this case, the objective function is so-called *map equation*.

From the simultaneous fusion methods, we have used:

- **Eva** ([17], taken from the authors[5]) – The algorithm repeats the same two phases as Louvain algorithm until its objective function is maximised. In this case, the objective function is $Z = \lambda P + (1-\lambda)Q$ where Q is modularity, P is purity, and λ is a trade-off parameter between Q and P. More about the objective function can be found in Section 3.2.4.

3.1.2 Evaluation metric

For the evaluation, we have chosen NMI (see Section 2.2.2).

3.2 Parameters

This section presents the parameters used in benchmarks and algorithms – for X-Mark benchmark in Section 3.2.1, for fusing algorithms in Section 3.2.2, for

[2] `https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.louvain.louvain_communities.html`

[3] `https://python.igraph.org/en/stable/api/igraph.Graph.html#community_leiden`

[4] `https://python.igraph.org/en/stable/api/igraph.Graph.html#community_infomap`

[5] `https://github.com/GiulioRossetti/eva`

clustering algorithms in Section 3.2.3 and for simultaneous approach algorithms in Section 3.2.4. The sections describe the parameters, their meaning for the algorithm and the values used for testing.

3.2.1 X-Mark parameters

The names of the benchmark parameters are summarised in Table 3.1. We have omitted the parameters which were kept on the default values: the tolerance (default value 10^{-7}), the maximum number of iterations (default value 500) and seed parameters (by default using Python "random" module without seed).

Parameter	Description
n	number of nodes
k	average node degree[1]
$k_{\min}$	minimum node degree[1]
$k_{\max}$	maximum node degree
min_community	minimum community size
max_community	maximum community size
τ_1	power-law exponent for node degree distribution
τ_2	power-law exponent for community size distribution
μ	mixing parameter
type_attr	attributes type, either categorical or continuous
ℓ	a list of attribute domain sizes[2]
σ	standard deviation of continuous attributes
ε	distance of peaks for continuous attributes
θ	noise for categorical attributes

[1] Exactly one of the average degree and minimum degree must be specified.
[2] The domain size must be at least equal to two. "auto" as the domain value means the attribute will have a domain equal to the number of generated communities.

Table 3.1: X-Mark benchmark parameters

Basic parameters settings

We have generated 20 networks of size $n = 50$ and size $n = 100$, 15 networks of size $n = 250$ and 10 networks of size $n = 500$. For all networks, we have set $k = n/10$ and maximum community equal to $3n/5$. For the networks of size $n \leq 100$, we have chosen $k_{\max} = 2n/5$ and minimum community equal to $n/10$. For the networks of size $n \geq 250$, we have chosen $k_{\max} = 3n/8$ and minimum community equal to $7n/100$. As we have set k, the parameter $k_{\min}$ was left unset. The values of minimum community and $k_{\max}$ were explicitly chosen to be able to generate enough graphs because we wanted to keep the maximum number of iterations the same to keep the generating time reasonable. More information about the parameter settings used by other authors can be found at the end of this section.

Power-law exponents setting

We have used $\tau_1 \in \{2, 3\}$ and $\tau_2 \in \{2, 3\}$ because these power-law exponent values also appear in real-world networks (see Section 1.2).

Mixing parameter settings

The mixing parameter is defined as

$$\mu = \frac{\sum_u k_{u,out}}{\sum_u k_u},$$

where $u \in V$ for graph $G = (V, E)$, $k_{u,out}$ is the sum of weights of edges connecting node u to nodes in different communities and k_u is the sum of weights of edges incident to node u.

We have considered μ values 0.05, 0.3 (for smaller networks divided into 0.2 and 0.35) and 0.5. Values of μ above 1/2 would violate the definition of communities in a strong sense (Definition 1) presented in Section 1.1.

Attribute parameters settings

We have decided to use only categoric attributes with domain sizes equal to the number of communities $|C|$. It was not possible to test every combination of attributes, so we have created the following labels scenarios which should serve as a cross-section of the possible scenarios: $\ell_1 = [2, 2]$, $\ell_2 = [|C|, 2]$, $\ell_3 = [|C|, |C|]$ and $\ell_4 = [|C|, |C|, |C|, |C|, |C|]$. We have studied them on networks of size $n = 50$ and $n = 100$. The results were similar for ℓ_1, ℓ_2, and ℓ_3, so we have furtherly used labels ℓ_3. Labels ℓ_4 had consistently better results than ℓ_3, and the differences among algorithms or different parameter settings of one algorithm were less clearly visible than for ℓ_3.

We have generated networks of size $n \leq 250$ with $\theta \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. For networks of size $n = 500$, we have used $\theta \in \{0, 0.4, 0.7, 1\}$ to save the computation time.

Other network parameter settings

Because most of the X-Mark parameters are inherited from the LFR benchmark, we have done a small survey about the LFR benchmark parameter settings used for testing algorithms. Many authors use similar settings for k and $k_{\max}$. However, they usually use the settings for some fixed network size. Another problem is assigning communities to nodes and generating of edges because X-Mark uses iterative generation and randomness. In some edge cases, the network generating can take a long time (for more information about the generation, see Section 3.4).

Nevertheless, we propose a second possible X-Mark benchmark parameters setting which could have been used. The set has fixed $k = 15$. This value is based on the average degrees of the most popular datasets given by Chunaev [14]. The average degree of the small-sized datasets is approximately 14.5, and the average degree of medium-sized datasets is 14 (the large-sized datasets were not considered because the benchmark networks are small- and medium-sized). Some authors used lower values ($k = 5$ in [56], $k = 10$ in [18]) or higher values

($k = 20$ in [90] and [48], $k = 25$ in [26], [71], [72] and [8]). We have chosen minimum community size equal to the average degree, $k_{\max} = 50$ (used in [48], maximum degree of size 40 was used in [26], [71], [72] and [8]), and maximum community size equal to $2n/5$. Some authors use the maximum community size equal to $n/10$ for networks of sizes $n \geq 1000$, but this value is insufficient for smaller networks because of the maximum degree. For networks with 50 and 100 nodes, we propose $k = 10$, minimum community equal to 15, $k_{\max} = 30$ and maximum community equal to 40. The purpose of this parameter setting is to maintain the average degree and minimum community and to set the maximum degree as low as possible with respect to the average degree.

3.2.2 Fusing algorithms parameters

NAS and NPS do not have any parameters. For MS, we used $\alpha \in \{0.3, 0.5, 0.7\}$. For α values near 0, the algorithm behaves as NPS, and for values near 1, it behaves as NAS.

3.2.3 Clustering algorithms parameters

In many algorithms (Louvain, Leiden, Greedy), there is a parameter called resolution. Louvain and Greedy algorithms use a value called modularity, and the resolution γ is a parameter in the modularity computation. The modularity takes a value between -1 and 1, and it is a difference between the number of in-community edges according to the partition and a fraction of the expected number of in-community edges according to configuration model ([85]).

We will use the modularity definition from Clauset et al. [19] and Blondel et al. [9]:

Definition 3 (Modularity)**.** *Modularity is defined as*

$$Q = \frac{1}{2m} \sum_{v,w} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w),$$

where A_{vw} is the weight of the edge between nodes v and w, $k_v = \sum_w A_{vw}$ is the sum of the weights of edges incident with node v, c_v is the community of node v, $m = \frac{1}{2} \sum_{v,w} A_{vw}$ is the sum of weights of all edges, the function $\delta(c_v, c_w)$ is called Kronecker delta function and it is defined as 1 if $c_v = c_w$ and 0 otherwise.

The resolution parameter γ is used, for example, in [85] as

$$Q = \frac{1}{2m} \sum_{v,w} \left[A_{vw} - \gamma \cdot \frac{k_v k_w}{2m} \right] \delta(c_v, c_w).$$

It is a trade-off parameter which allows the user to give an advantage to smaller or bigger-sized communities. If the resolution is less than 1, bigger communities are favoured in the modularity computation. Resolution greater than 1 favours smaller communities. Louvain and Greedy were tested with res $\in \{0.5, 1, 2\}$.

Leiden algorithm uses resolution γ in the computation of an objective function called the Constant Potts Model. This quality function should overcome some limitations of modularity ([85]). The usage of γ in the Constant Potts Model is

slightly different compared to usage in modularity – communities should have the density of edges at least equal to γ, and the density of edges between communities should be lower than γ. However, as for modularity, lower resolution values support detecting bigger communities, while higher resolution values favour smaller communities during the function computation. Leiden algorithm was tested with $\gamma \in \{0.5, 0.7, 1, 2\}$ while using NPS as fusion method, with $\gamma \in \{0.3, 0.5, 1, 2\}$ for NAS as fusion method and with $\gamma \in \{0.5, 1\}$ for MS as fuaion method.

Infomap has a parameter called "trials" which is the number of attempts to partition the network. The default trials value is 10. We have tested Infomap for trials $\in \{1, 2, 4, 10, 15\}$.

The algorithms were run three times on each network of size $n \leq 250$. On networks of size $n \geq 500$, the algorithms were run twice on each network.

3.2.4 Simultaneous fusion algorithm parameters

Eva also has a resolution parameter inherited from the Louvain algorithm (mentioned in 3.2.3). We have tested the algorithm for res $\in \{0.5, 1, 2\}$. The second Eva parameter is λ (called α in the original text, but we have renamed it due to a naming conflict) – a trade-off parameter in the optimised score $Z = \lambda P + (1-\lambda)Q$ where Q is modularity, and P is purity. It is clear that for $\lambda = 0$, the Eva algorithm behaves the same way as the Louvain algorithm.

Purity is the average of the purities of the communities in the partitioning and takes a value between 0 and 1. The purity of a community is the product of the frequencies of the most frequent labels (attributes) carried by nodes in the community. The following definition of purity and purity of a community comes from Citraro and Rossetti [17]. We use notation according to Definition 2.

Definition 4 (Purity)**.** *Given a graph $G = (V, E, \mathcal{A})$ and a partitioning of nodes into communities $\mathcal{C}$, $\bigcup_{c \in \mathcal{C}} c = V$, the purity of community $c \subseteq V$ is defined as*

$$P_c = \prod_{k=1}^{d} \frac{\max_{a \in \mathcal{A}_k} \left(\sum_{v \in V} a(v)\right)}{|c|},$$

where $\mathcal{A}_k$ is the domain of attribute k for $k \in \{1, \ldots, d\}$, $a(v)$ is an indicator function defined for $a \in \mathcal{A}_k$, $k \in \{1, \ldots, d\}$ as

$$a(v) = \begin{cases} 1, & \text{if } a_{vk} = a \\ 0, & \text{otherwise} \end{cases}$$

The purity P is defined as

$$P = \frac{1}{|C|} \sum_{c \in C} P_c,$$

where C is the set of communities in given partition and P_c is the purity of community c.

The purity of a community is maximised if all the nodes in the community have the same attribute values. Purity is maximised if all community purities are maximised. We have tested the Eva algorithm for the default λ value ($\lambda = 0.5$) because the algorithm had a very long running time. We had chosen to

prioritise the testing on many networks and their parameters than focusing on the λ parameter. However, it could be interesting to analyse the impact of the λ parameter on the results. We refer an interested reader to an analysis by Citraro and Rossetti [17], which focuses on the impact of λ values on the detected number of communities.

On all network sizes, Eva was run two times on each network.

3.3 Running time of algorithms

For interested readers, we present some rough comparison of the running times of the algorithms.

Considering the fusion parts of algorithms, the MS was slower than separate NPS and NAS. As NAS only computes similarity and compares numbers and NPS must compute shortest paths, NAS was faster than NPS (even when the shortest paths were computed using the Floyd-Warshall algorithm).

If we move on to clustering parts of algorithms, the Greedy algorithm was the slowest, followed by Louvain and Infomap, and the Leiden algorithm was the fastest.

The running time of the Eva algorithm was similar to the running time of the Greedy algorithm.

3.4 Encountered problems

This section will introduce some problems we have encountered while analysing the algorithms.

X-Mark parameters setting

The first problem (already slightly mentioned in Section 3.2.1) was setting the X-Mark benchmark parameters. The problematic parameters were inherited from the LFR benchmark. LFR uses randomness in many steps of network creation – the most problematic part is assigning the communities to nodes. In the beginning, the expected node degrees and the community sizes are generated (randomly chosen from power-law degree distribution). The nodes are assigned an expected degree which is divided into in-community and out-community parts according to μ. The nodes are then assigned such community that the in-community degree is less than or equal to the community size. The community assignment is done in a few iterations because if we do not have an enough-sized community for a node, we randomly remove a node from the chosen enough-sized community, and we add our node. The last operation done is the creation of the edges between nodes. The edges are assigned with respect to the expected in- and out-community node degrees. The following problem can appear (especially for higher μ values): a node has some out-community degree, and there are not enough nodes outside the community to attach the edges.

The described problems, solved by the LFR algorithm via iterative generation, can be solved completely by solving an integer quadratic program. We will have the following constants:

- n for the number of nodes,
- s for the number of communities,
- $k_1, \ldots, k_n$ for the expected node degrees,
- $k'_1, \ldots, k'_n$ for the expected in-community node degrees,
- $n_1, \ldots, n_s$ for the community sizes.

We want to find the values of the following variables:

- $e_{uv} = e_{vu}$ for $u, v \in \{1, \ldots, n\}$

$$e_{uv} = \begin{cases} 1, & \text{if there is an edge in the graph between nodes } u \text{ and } v \\ 0, & \text{otherwise} \end{cases}$$

- c_{uj} for $u \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, s\}$

$$c_{uj} = \begin{cases} 1, & \text{if node } u \text{ is in the } j\text{-th community} \\ 0, & \text{otherwise} \end{cases}$$

- c'_{uv} for $u, v \in \{1, \ldots, n\}$

$$c'_{uv} = \begin{cases} 1, & \text{if nodes } u, v \text{ are in the same community} \\ 0, & \text{otherwise} \end{cases}$$

The constructed quadratic program is

$$\begin{aligned} & \max 1 \\ \forall j \in \{1, \ldots, s\} : & \sum_{u=1}^{n} c_{uj} = n_j \\ \forall u \in \{1, \ldots, n\} : & \sum_{j=1}^{s} c_{uj} = 1 \\ \forall u, v \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, s\} : & \; c_{uj} \cdot c_{vj} = c'_{uv} \\ \forall u \in \{1, \ldots, n\} : & \sum_{v=1}^{n} e_{uv} \cdot c'_{uv} = k'_u \\ \forall u \in \{1, \ldots, n\} : & \sum_{v=1}^{n} e_{uv} \cdot (1 - c'_{uv}) = k_u - k'_u \end{aligned}$$

However, we have not solved this integer quadratic program because finding an optimal solution for integer programs in general is hard. Nevertheless, even having the approximate solution of this program could be helpful because we could at least shorten the iterative generation used by the LFR benchmark. If we had known the exact solution, we could have generated all the possible networks for the given set of parameters and thoroughly tested the algorithms.

RAM size

Another problem we faced was the RAM size of the computer. All computing was done in WSL (Windows Subsystem for Linux) on Windows 10 with processor Intel Core i7-8565U and RAM size 16 GB. During the testing on graphs of size $n \geq 500$, the partitioning of a graph had to be saved into a file immediately after computation.

4. Results

This chapter introduces the results of the comparative analysis. The results for early fusion algorithms are presented in Section 4.1. We have divided the results according to the fusion method used. Each fusion method was run with Louvain, Leiden, Greedy and Infomap algorithms. Section 4.1.1 presents the results of NPS fusion, Section 4.1.2 contains the results of NAS fusion, and Section 4.1.3 shows the results of MS fusion. The last section regarding the early fusion approach, Section 4.1.4, compares the results for each clustering algorithm focusing on the performance of chosen fusion method.

The results for the simultaneous fusion approach are presented in Section 4.2. This section contains only one algorithm – Eva.

Before moving on to the results, we would like to recall the most important parameters of the X-Mark benchmark:

- n is the number of nodes,
- μ is the mixing parameter, lower μ values cause the communities to be almost disconnected, higher μ values cause the communities to be more interconnected,
- θ is the noise parameter, lower θ values cause the node attributes within a community to be more homogenous, higher θ values cause the node attributes within a community to differ,
- τ_1 is the power-law exponent for node degree distribution,
- τ_2 is the power-law exponent for community size distribution,

and the most important parameters of the algorithms:

- the resolution parameter of Louvain, Greedy, Leiden and Eva algorithms, lower res values mean the algorithm favours bigger communities, higher res values mean the algorithm favours smaller communities,
- α is the parameter of the MS algorithm, lower α values mean the algorithm uses the NPS weights more than the NAS weights, higher α values mean the algorithm uses the NAS weights more than the NPS weights,
- trials is the number of attempts to partition the network for the Infomap algorithm.

We will abbreviate "resolution" to "res". We show only results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$. The results of algorithms for other τ_1 and τ_2 values were similar.

We expected the results would get worse with higher μ values because the communities are more interconnected and thus more difficult to detect. We also expected (only for algorithms considering node attributes) that the results would worsen with higher θ values because the attributes in communities are less homogenous. The discussion of obtained results and the influence of the parameters can be found in Section 4.3.

4.1 Early fusion methods

4.1.1 Node path similarity

NPS does not consider node attributes, so we show the merged results for all θ values ($\theta \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ for $n \leq 250$ and $\theta \in \{0, 0.4, 0.7, 1\}$ for $n = 500$). To save running time, we have tested the NPS + Greedy algorithm for $n \leq 250$ only for networks with $\theta = 0$.

NPS + Louvain

NPS + Louvain with res = 0.5 detected only one large community consisting of all nodes, and NPS + Louvain with res = 2 assigned almost every node to its own community. Thus, we will show only results for res = 1.

For networks of size $n \leq 100$ (Figure 4.1), results for res = 1 are negatively influenced by higher μ values, the decrease of results is very high. However, the results for networks of size $n = 100$ are less influenced by μ than the results for networks of size $n = 50$.

For networks of size $n \geq 250$ (Figure 4.1), the results for res = 1 are decreasing with higher μ values which aligns with our expectations. The results for bigger networks are much better than those for smaller networks. The results have a relatively large variance, especially for $\mu = 0.5$.

NPS + Leiden

NPS + Leiden with res = 0.5 could not detect any communities – it detected one large community consisting of all nodes. NPS + Leiden with res ≥ 1 assigned every node to its own community. Therefore, we have used res = 0.7 to test the algorithm.

For all network sizes (Figure 4.2), the results are negatively influenced by higher μ values. However, the results for bigger networks are less influenced by μ than the results for smaller networks. Also, the results for $\mu \geq 0.3$ have a large variance.

NPS + Greedy

NPS + Greedy with res = 0.5 detected only one large community consisting of all nodes, and NPS + Greedy with res = 2 assigned almost every node to its own community. Thus, we will show only results for res = 1.

For networks of size $n \leq 100$ (Figure 4.2), results for res = 1 are negatively influenced by higher μ values.

For networks of size $n \geq 250$ (Figure 4.2), the results for res = 1 are decreasing with higher μ values which aligns with our expectations. In this case, we have tested networks of this size only for $\theta = 0$ because θ does not affect the NPS results, and it also saved some running time. Moreover, we supposed that for lower μ values, the Greedy algorithm will have better results. However, the best results the res = 1 achieves are for $\mu = 0.3$.

Overall, the results for bigger networks are much better than the results for smaller networks. As we observe, the results have a relatively large variance, especially for $\mu = 0.5$.

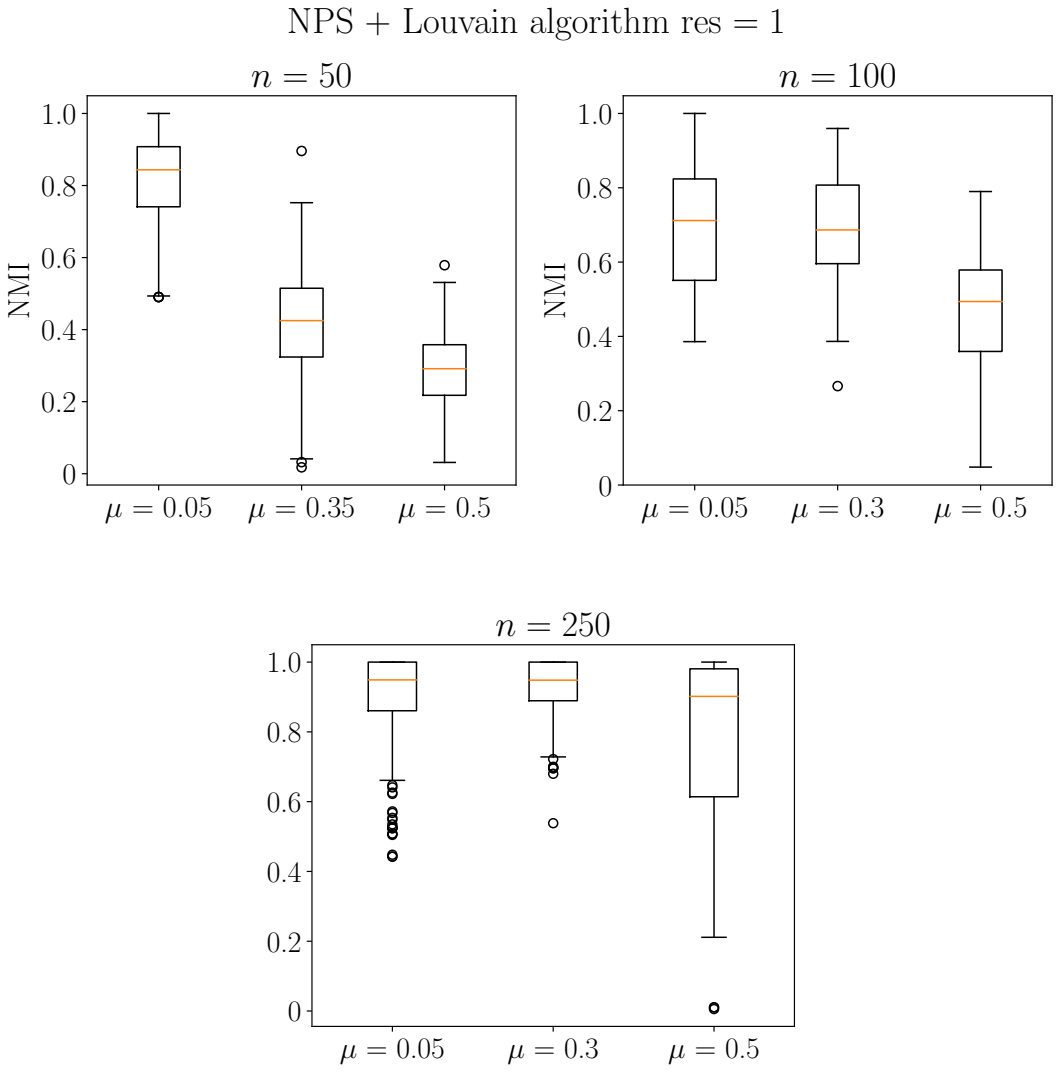


Figure 4.1: NPS + Louvain algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

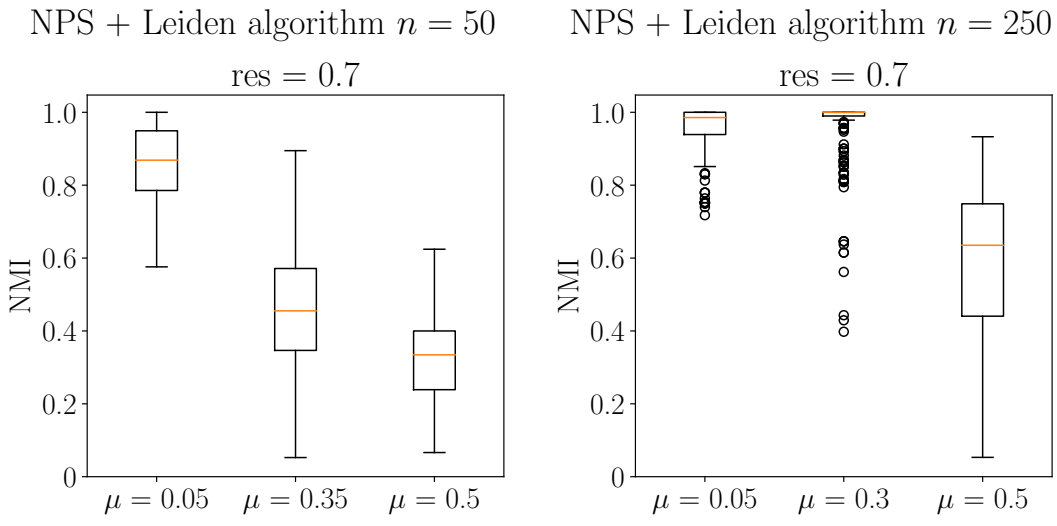


Figure 4.2: NPS + Leiden algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

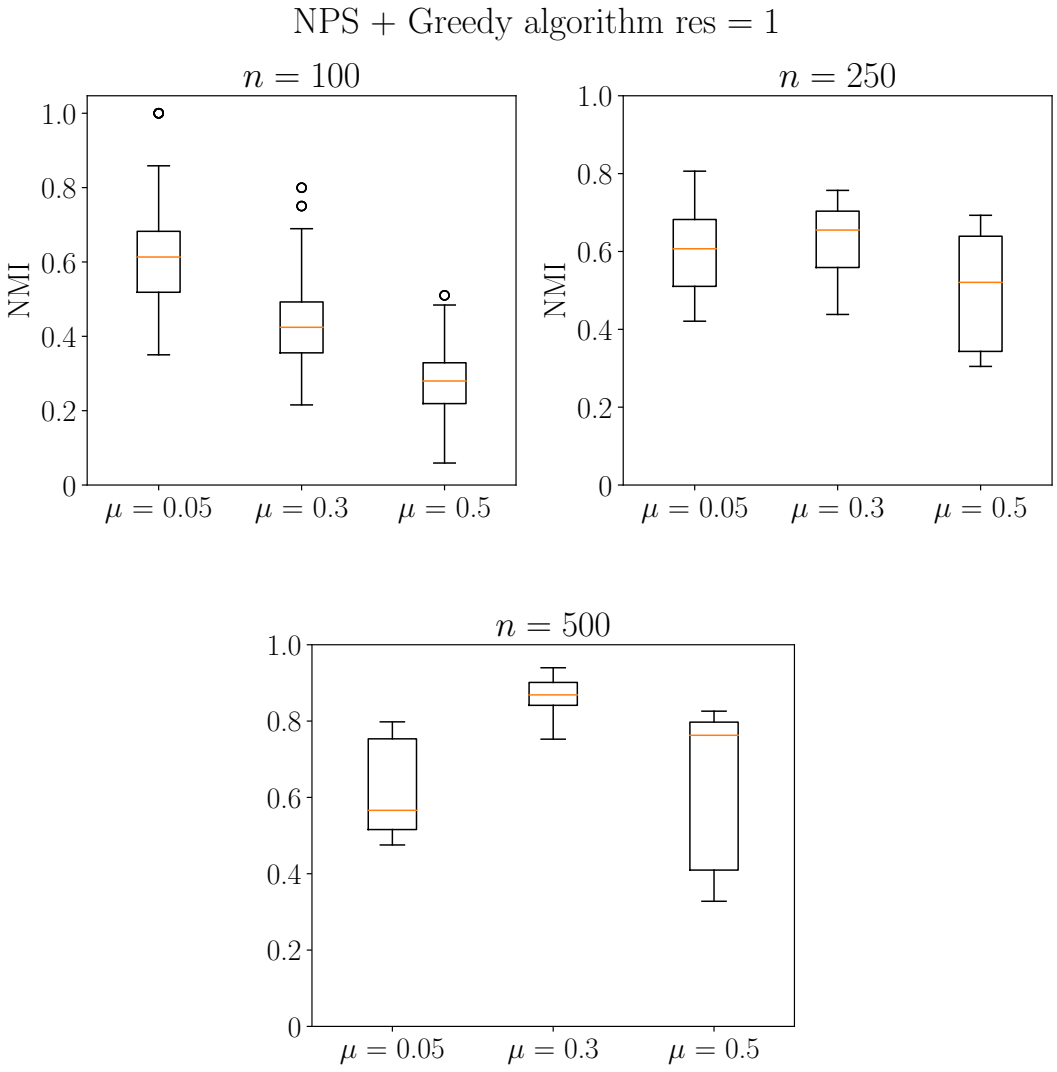


Figure 4.3: NPS + Greedy algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

NPS + Infomap

Infomap could not detect any communities independently of the μ values – it detected one large community consisting of all nodes.

NPS algorithms comparison

We will show results for NPS + Louvain and NPS + Greedy algorithms with res = 1 and NPS + Leiden algorithm with res = 0.7.

For networks of size $n \leq 100$ (Figure 4.4), the results for NPS + Leiden algorithm are very similar to the results for NPS + Louvain algorithm. NPS + Greedy algorithm has slightly worse results than NPS + Louvain.

For networks of size $n \geq 250$ (Figure 4.5), results for NPS + Greedy algorithm are always worse than results for NPS + Louvain. Overall, NPS + Louvain, despite its significant variance for $\mu = 0.5$, clearly outperforms Greedy and Leiden

for all μ values. It can also be seen that the results for NPS + Greedy algorithm are better than the results for NPS + Leiden algorithm for $\mu = 0.5$.

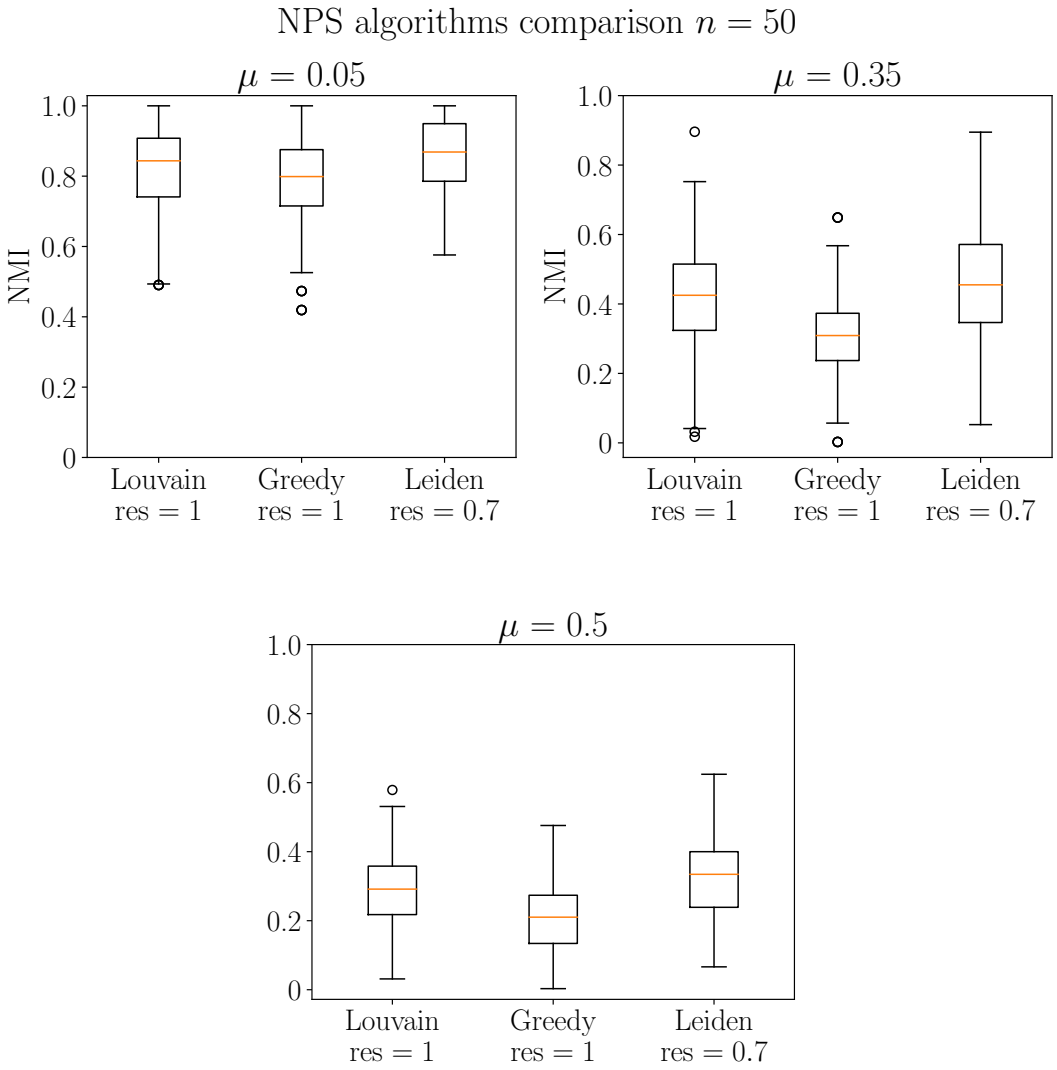


Figure 4.4: NPS + Louvain, Greedy and Leiden algorithms results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

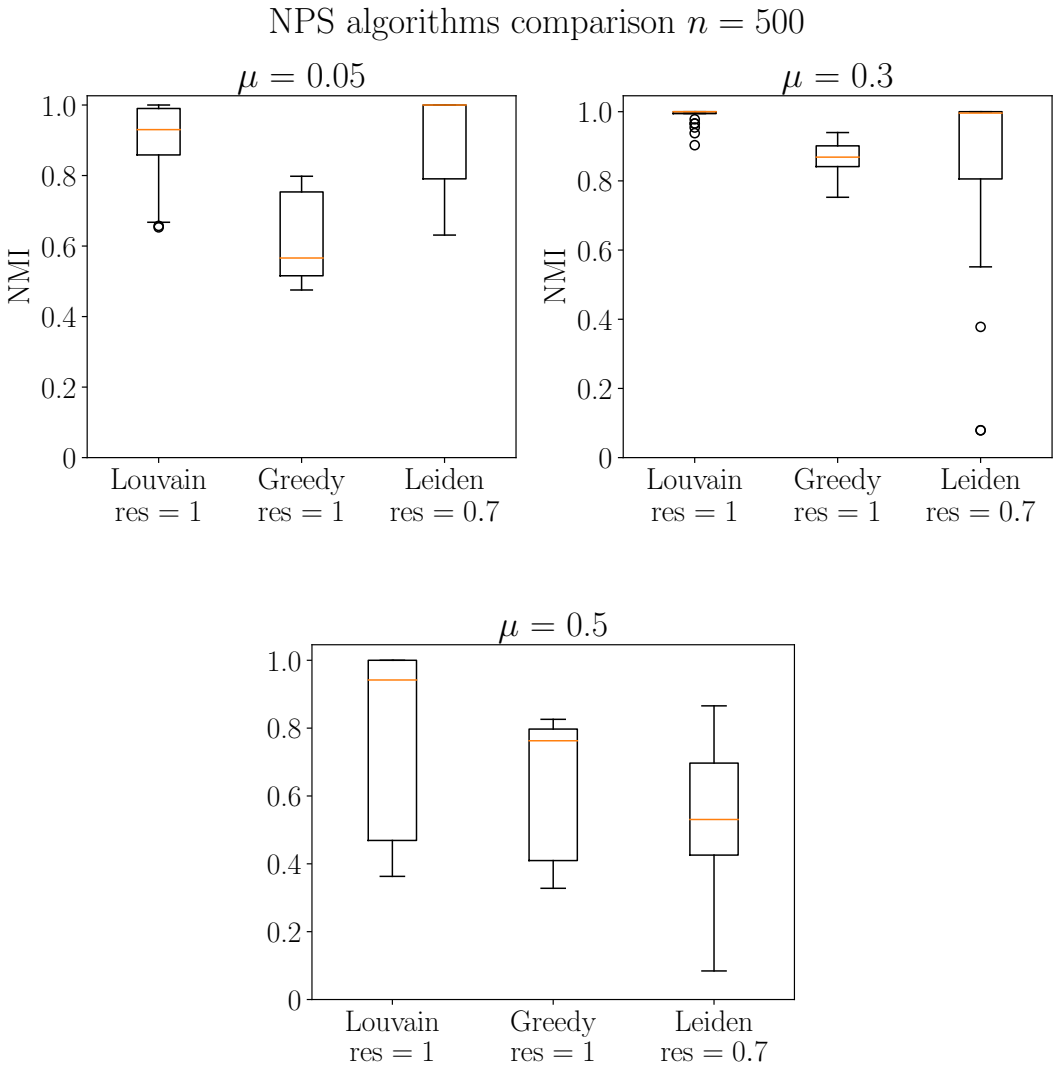


Figure 4.5: NPS + Louvain, Greedy and Leiden algorithms results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

4.1.2 Node attribute similarity

We have tested the algorithms on networks with $\theta \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ for $n \leq 250$ and $\theta \in \{0, 0.4, 0.7, 1\}$ for $n = 500$. However, we usually show only the most interesting results.

NAS + Louvain

Results for res = 1 are always placed between values for res = 0.5 and res = 2. For all networks, res = 0.5 is the best for $\mu \leq 0.05$ and also for the combination of $\mu = 0.3$ and $\theta \leq 0.4$. However, this resolution is the one which is most influenced by higher μ and higher θ values, so the results for $\mu = 0.5$ are the worst.

For networks of size $n \leq 100$ (Figure 4.6), results for $\mu = 0.05$ are very similar for all values of θ; they are only slightly decreasing. For $\mu \geq 0.3$, the best resolution results shift from res = 0.5 for $\theta \leq 0.4$ to res = 2 otherwise.

For networks of size $n \geq 250$ (Figure 4.7), results for $\mu = 0.05$ and a specific resolution are very similar for all θ values. The results are even the same for

res = 0.5. For other resolutions, the results are only slightly decreasing. For $\mu = 0.3$, the results are similar for all resolutions. However, the variance of results for res = 0.5 increases with increasing θ values. For $\mu = 0.5$, the results for all resolutions are close for small $\theta \leq 0.4$. The choice of res = 2 has the best results for $\theta \geq 0.4$ as it is the least influenced by the θ increase.

The results for bigger networks are better than those for smaller networks, except for high values of μ and θ (for example, $\mu = 0.5$ and $\theta = 1$). Overall, all results are decreasing for higher μ and θ values, which aligns with our expectations.

NAS + Leiden

NAS + Leiden algorithm with res ≥ 1 assigned every node to its own community. Therefore, we have added testing NAS + Leiden algorithm with res = 0.3.

For networks of size $n \leq 100$ (Figure 4.8), the results for res = 0.3 are better than results res = 0.5, except for the case $\mu = 0.5$ and $\theta \geq 0.8$. The variance of results for specific μ decreases with higher θ values.

For networks of size $n \geq 250$ (Figure 4.9), results for $\theta \geq 0.4$ and a specific resolution are very similar for all μ values. The results for res = 0.3 are better than the results for res = 0.5 (they are similar only in case $\mu = 0.5$ and $\theta \geq 0.8$). The variance of results for specific μ decreases with higher θ values.

The results for smaller networks are better than those for bigger networks. Overall, all results are decreasing for higher μ and θ values, which aligns with our expectations.

NAS + Greedy

Results for res = 0.5 and res = 1 are similar for all network sizes. Results for res = 2 are more stable than results for other resolutions – they are less negatively influenced by higher μ and θ values.

For networks of size $n \leq 100$ (Figure 4.10), res = 0.5 is better for smaller μ and θ values (for example $\mu = 0.05$ and θ arbitrary, $\mu = 0.5$ and $\theta \leq 0.2$). For $\mu = 0.5$ and $\theta \geq 0.4$, res = 0.5 is surpassed by res = 1, except for the case $\theta \geq 0.8$. For these θ values, the best results are obtained by res = 2.

For networks of size $n \geq 250$ (Figure 4.11), res = 0.5 and res = 1 have both very high results compared to the results of res = 2. The only case this resolutions are surpassed by res = 2 is when $\mu = 0.5$ and $\theta \geq 0.2$ (res = 2 surpasses res = 0.5) or $\mu = 0.5$ and $\theta \geq 0.6$ (res = 2 surpasses res = 1).

Overall, the results are negatively influenced by higher μ and θ values. When $\mu = 0.05$, the results are less influenced by θ than the results for $\mu = 0.5$.

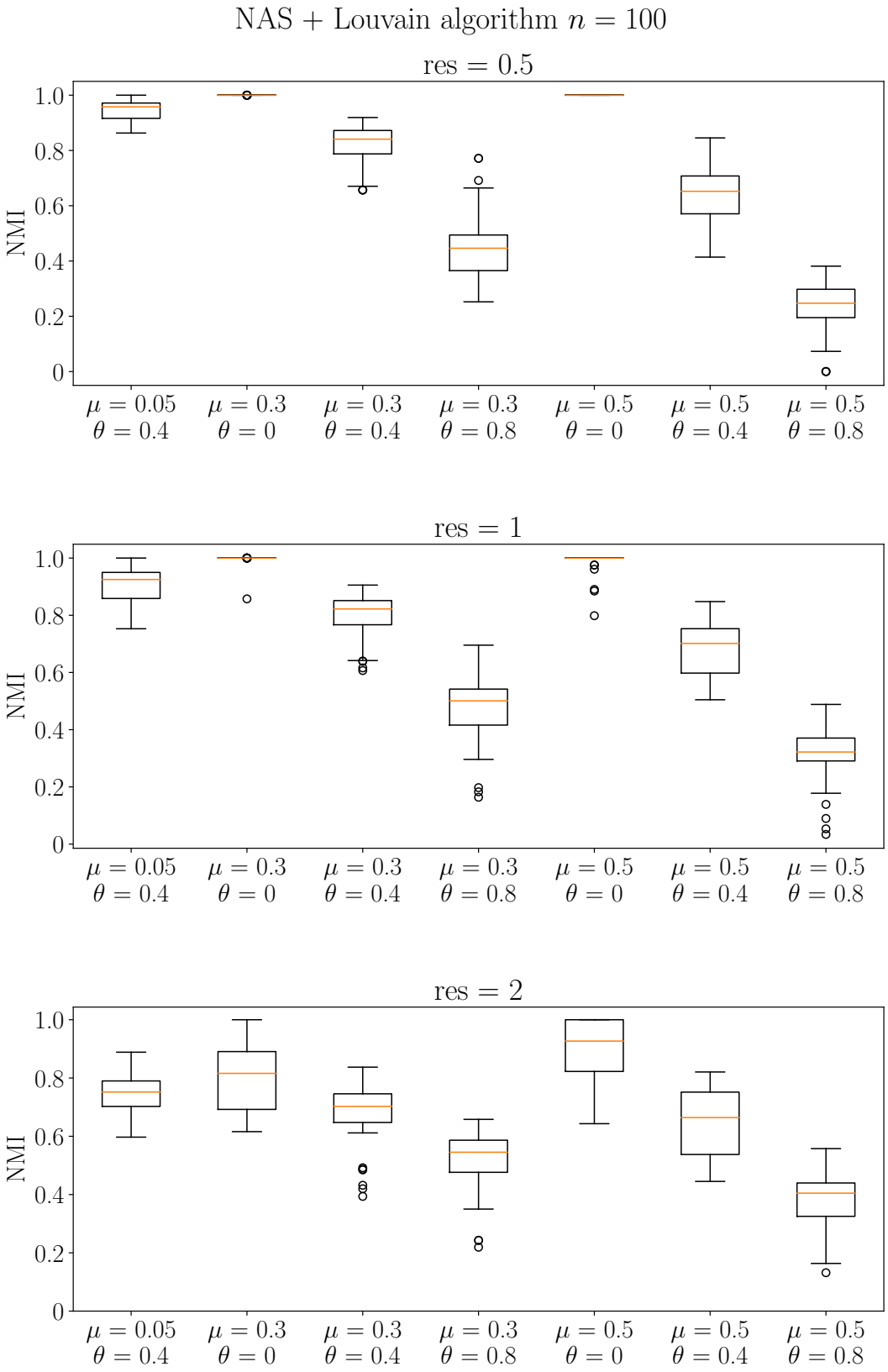


Figure 4.6: NAS + Louvain algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

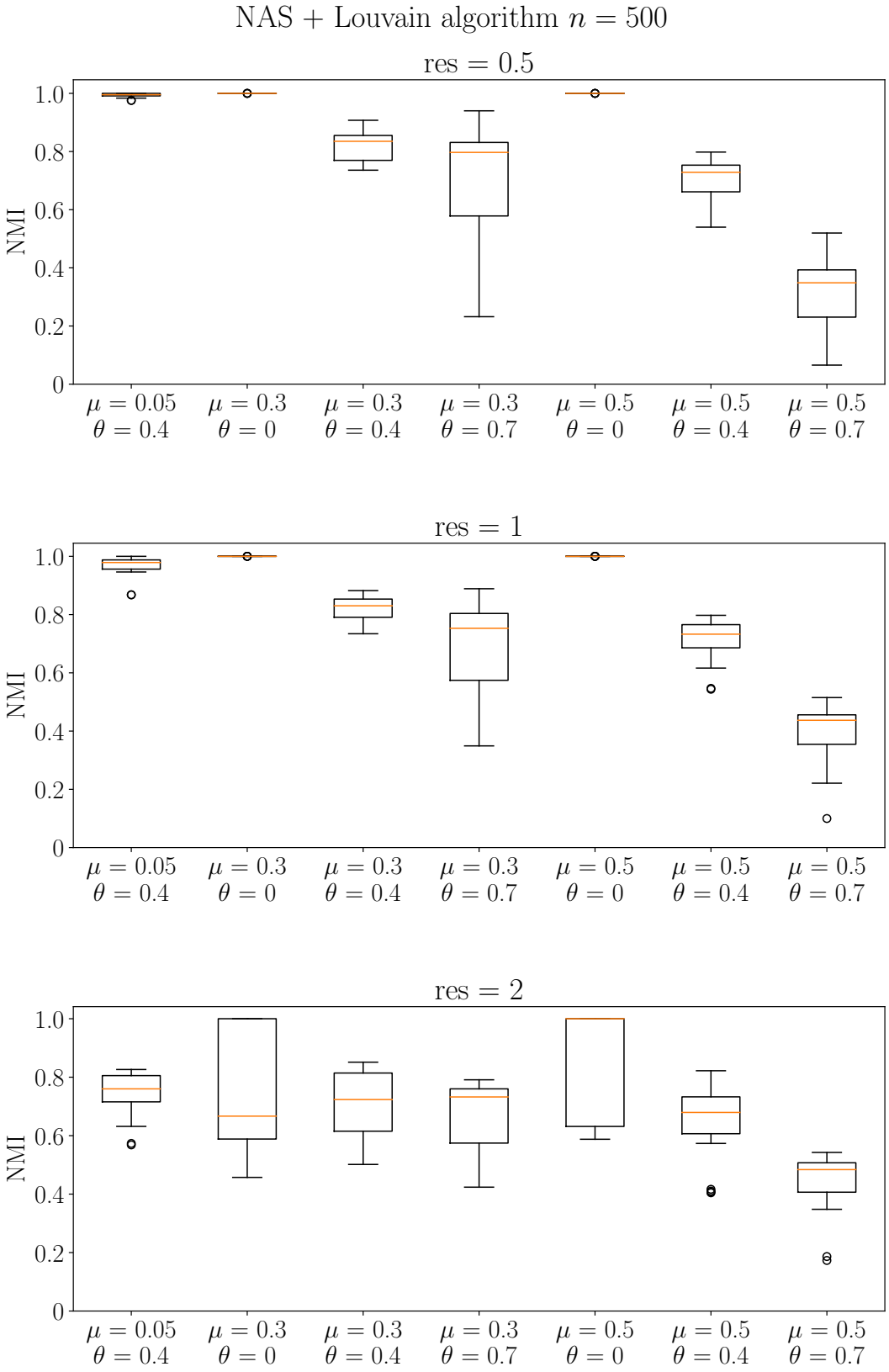


Figure 4.7: NAS + Louvain algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

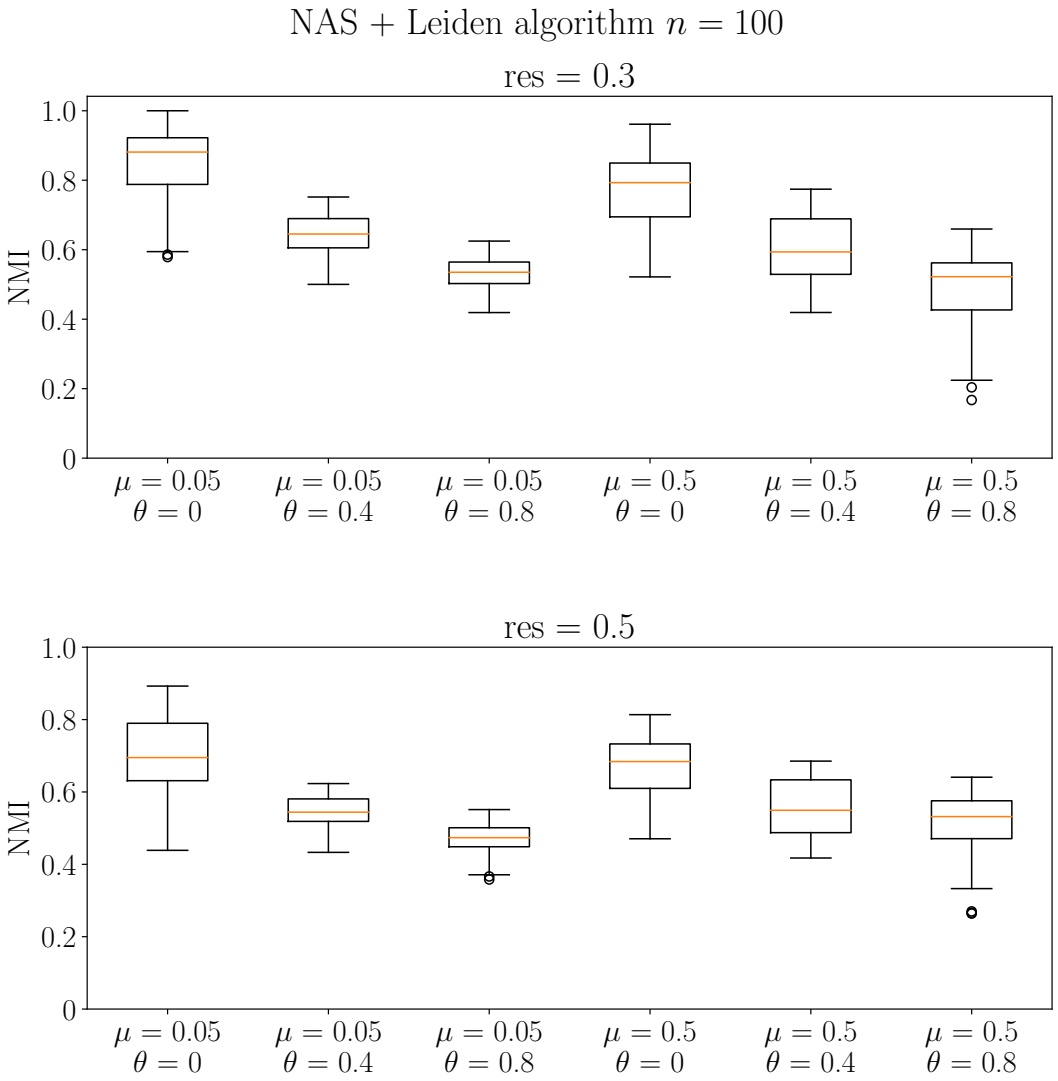


Figure 4.8: NAS + Leiden algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

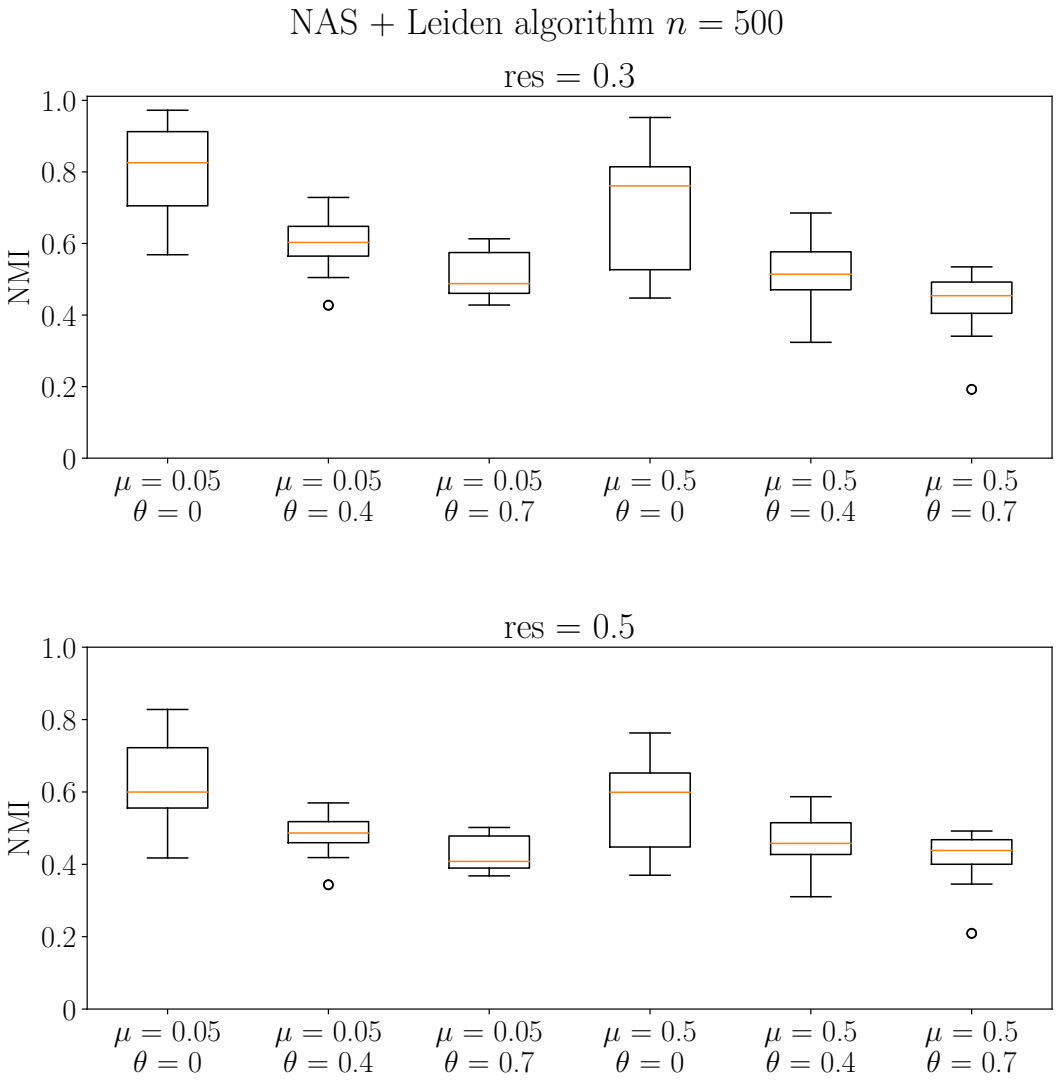


Figure 4.9: NAS + Leiden algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

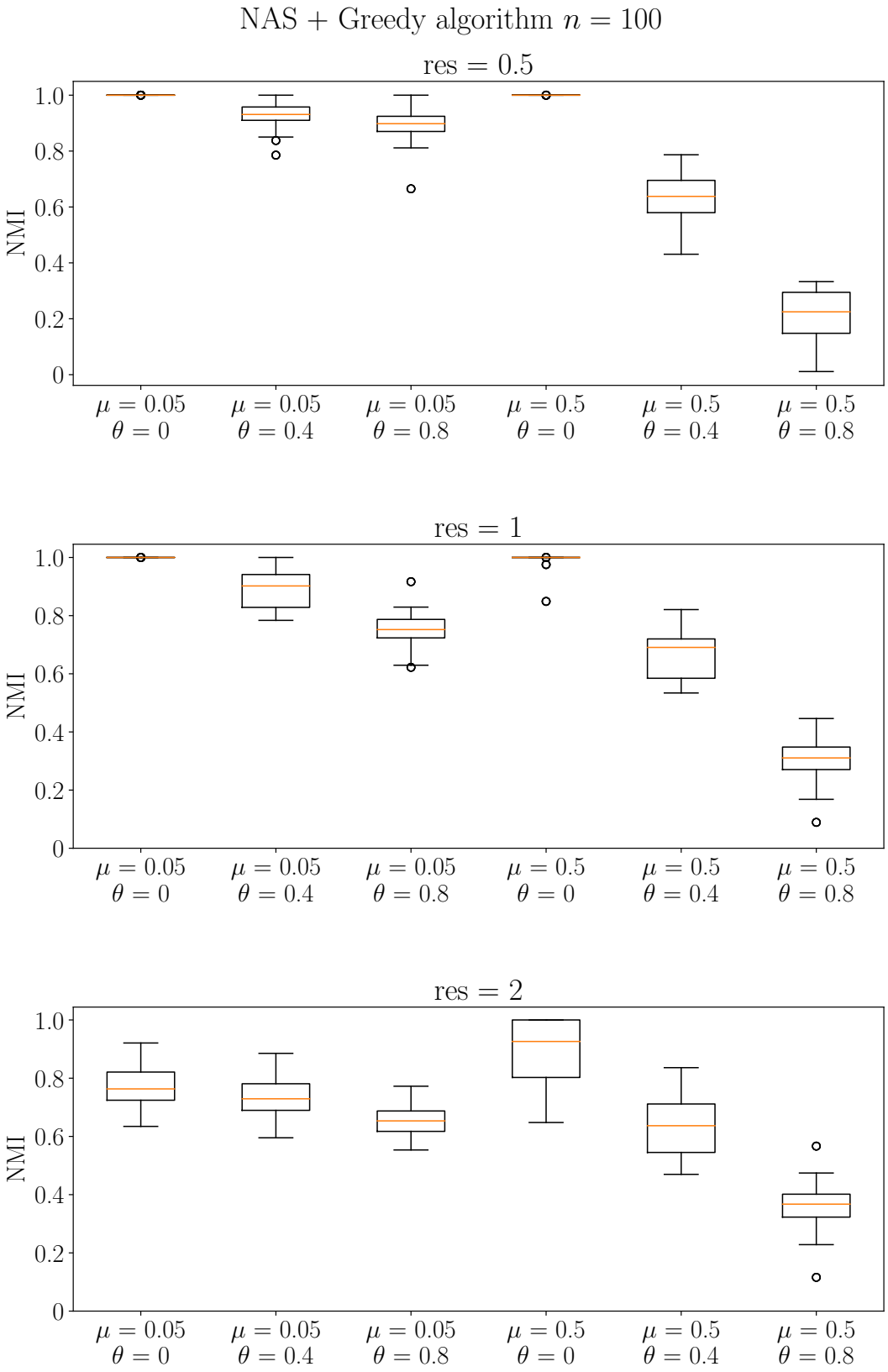


Figure 4.10: NAS + Greedy algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

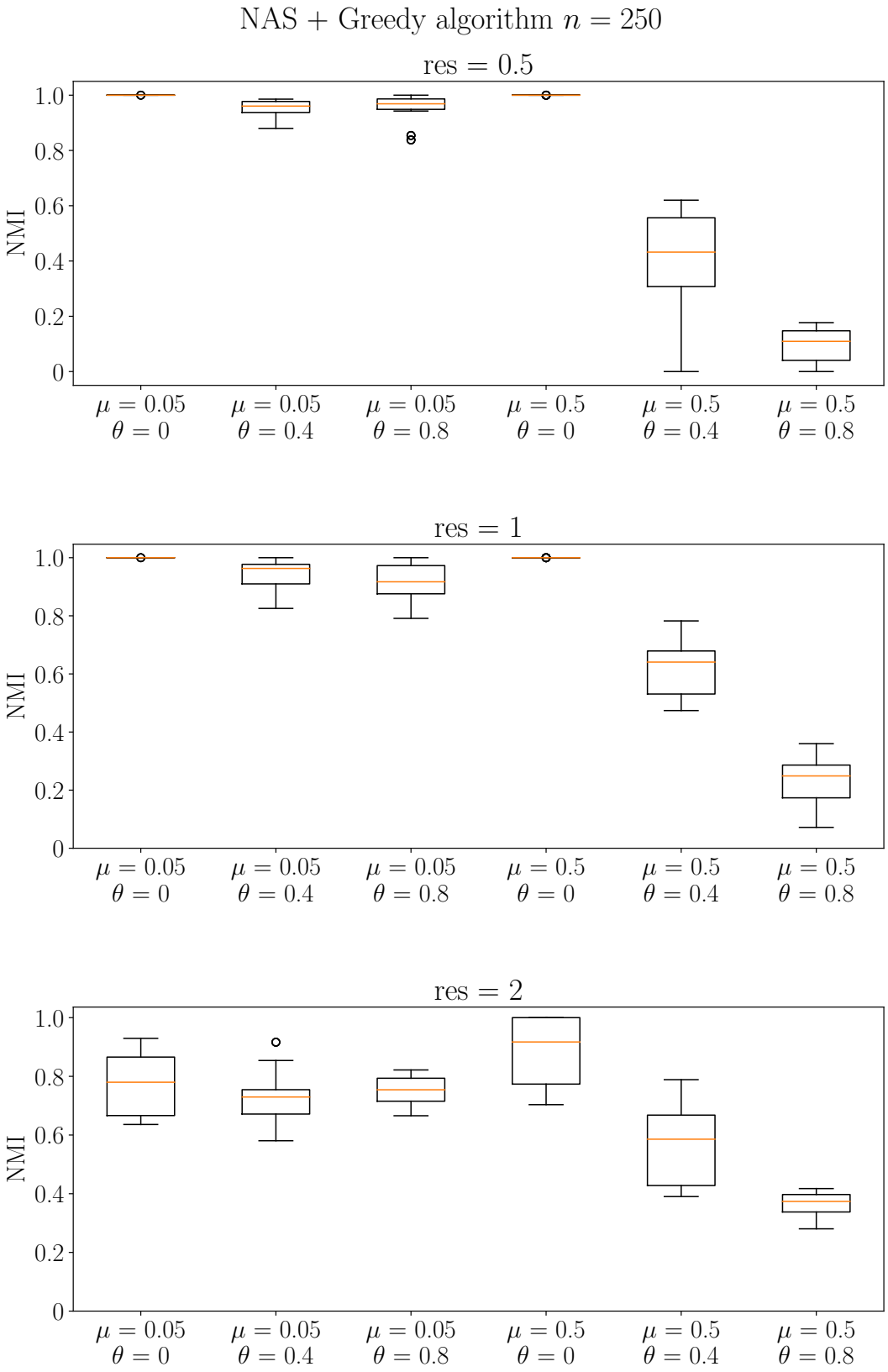


Figure 4.11: NAS + Greedy algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

NAS + Infomap

Results for different trial parameter values are similar, so we will show only results for trials equal to 10.

For networks of size $n \leq 100$ (Figure 4.12), results decrease with higher μ and θ values. When $\mu = 0.05$, the differences between the results for different θ values are less significant than for $\mu = 0.5$.

For networks of size $n \geq 250$ (Figure 4.13), results for $\mu = 0.05$ are very similar. The higher the μ value is, the more the results are negatively influenced by higher θ values.

Overall, having $\mu \leq 0.3$, the results for bigger networks are better than those for smaller networks. In case $\mu = 0.5$, results for $n \geq 250$ are better only for low θ values ($\theta \leq 0.6$). As we expected, the results are negatively influenced by higher μ and θ values.

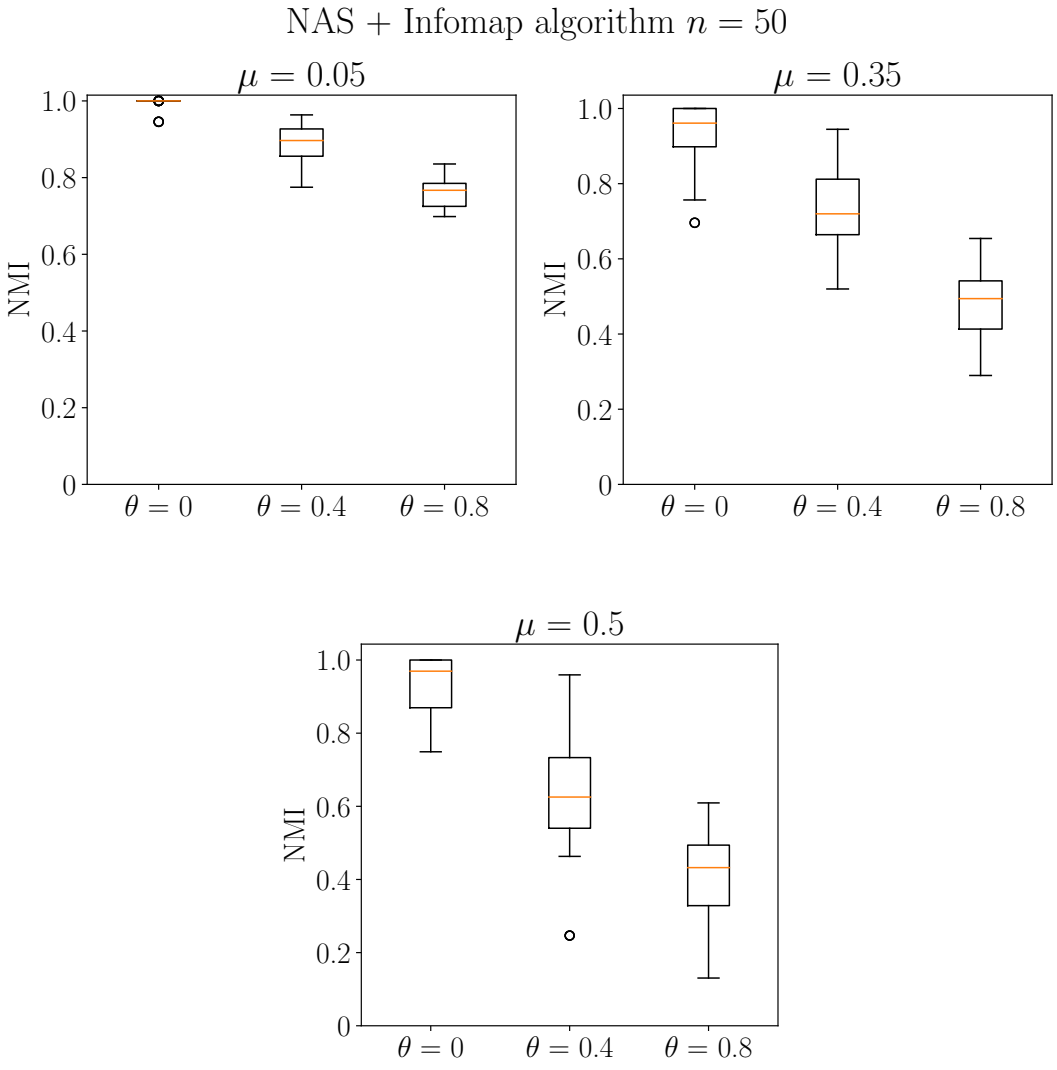


Figure 4.12: NAS + Infomap algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

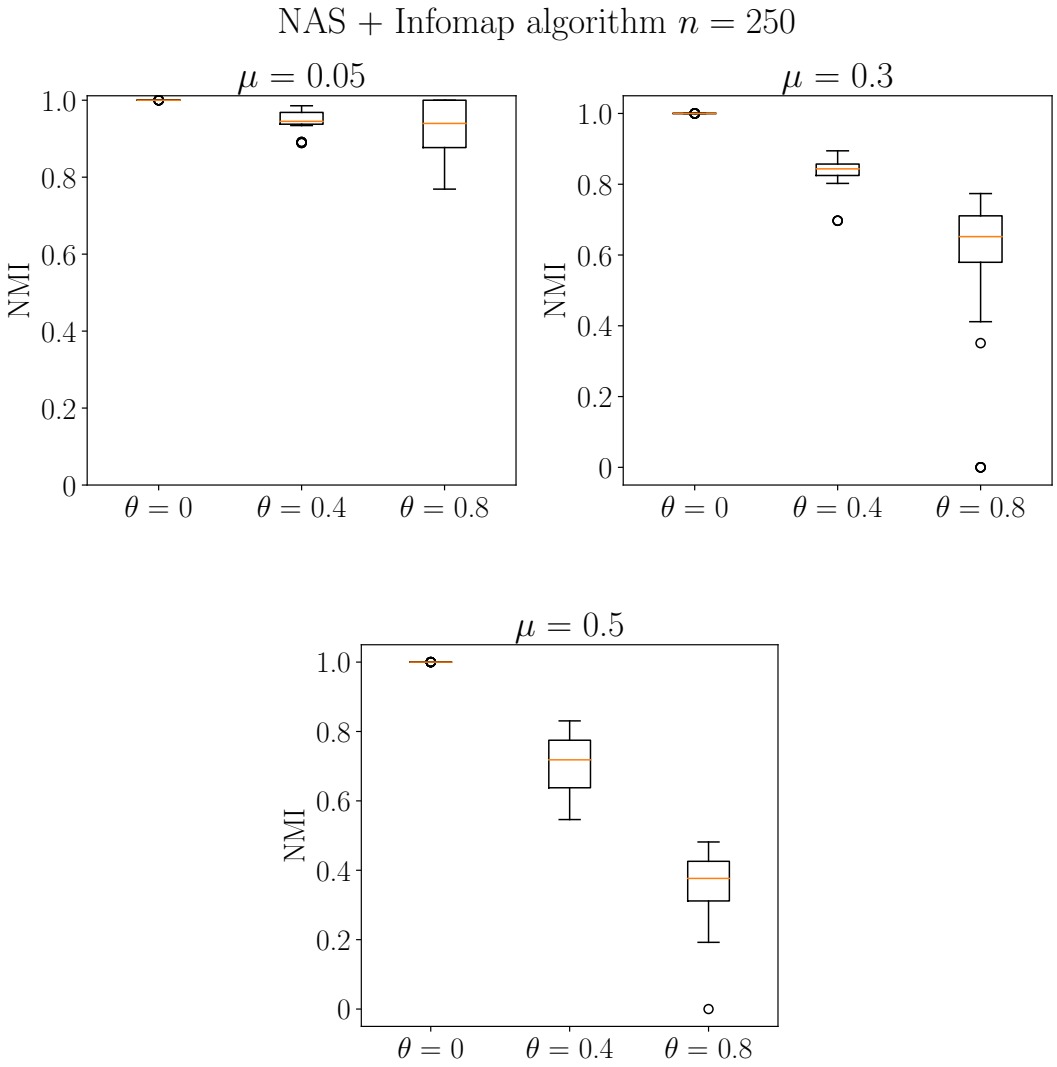


Figure 4.13: NAS + Infomap algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

NAS algorithms comparison

NAS + Louvain and NAS + Greedy algorithms had similar results for both resolutions res = 1 and res = 2 (Greedy had slightly worse results for all parameter values), so we will show only results for NAS + Louvain algorithm. We also show results for NAS + Leiden algorithm only with res = 0.3.

For networks of size $n \leq 100$ (Figure 4.14), Infomap and Louvain algorithms have better results for $\theta \leq 0.4$ than the Leiden algorithm. Leiden algorithm outperforms Infomap and Louvain in case $\mu \geq 0.35$ and $\theta > 0.4$.

For networks of size $n \geq 250$ (Figure 4.15), the Leiden algorithm outperforms Louvain and Greedy only in case $\mu = 0.5$ and $\theta \geq 0.6$.

Overall, the Infomap algorithm has similar results as Louvain res = 1, except for higher θ values. In that case, it has slightly better results, similar to the results of Louvain with res = 2. As shown in Figures 4.8 and 4.9, the Leiden algorithm is less sensitive to the increase of μ and θ values than the Infomap and Louvain algorithms. However, the results of the Leiden algorithm are lower for

larger networks compared to the results for smaller networks. On the contrary, Infomap and Louvain algorithms have better results for bigger networks (except for high μ and θ values). Therefore, for networks of size $n \geq 250$, the Leiden algorithm outperforms Infomap and Louvain algorithm only in the case $\mu = 0.5$ and $\theta \geq 0.6$.

4.1.3 Mixed similarity

MS + Louvain

Results for $\alpha = 0.5$ are always between results for $\alpha = 0.3$ and $\alpha = 0.7$, so we will not show results for $\alpha = 0.5$.

For networks of size $n \leq 100$ (Figure 4.16), $\alpha = 0.7$ has the best results for $\theta \leq 0.8$. The results for $\theta \geq 0.8$ are similar for all α values.

For networks of size $n \geq 250$ (Figure 4.17), results for all α values are similar in case $\mu \leq 0.3$. For $\mu = 0.5$, $\alpha = 0.3$ outperforms $\alpha = 0.5$ and $\alpha = 0.7$.

Overall, the results for $n \geq 250$ are always better than those for $n \leq 100$. Higher α values are preferable for $n \leq 100$. On the other hand, lower α values are better for $n \geq 250$.

MS + Leiden

MS + Leiden with res ≥ 1 assigned every node its own community. Thus, we will show only results for MS + Leiden with res $= 0.5$. Results for $\alpha = 0.5$ were always between results for $\alpha = 0.3$ and $\alpha = 0.7$, so we will not show results for $\alpha = 0.5$.

For networks of size $n \leq 100$ (Figure 4.18), $\alpha = 0.3$ has the best results for $\mu = 0.05$. The choice of $\alpha = 0.7$ is better for higher μ values, which we expected because $\alpha = 0.3$ uses more NPS weights. Thus, its results are more negatively influenced by higher μ values than the results for $\alpha = 0.7$. All results are negatively influenced by higher μ and θ values.

For networks of size $n \geq 250$ (Figure 4.19), results for $\alpha = 0.3$ are near one for $\mu \leq 0.3$. For $\mu = 0.5$, the results for $\alpha = 0.3$ are still better than those for $\alpha = 0.7$ despite the high variance for $\theta \geq 0.4$.

Overall, the results for $n \geq 250$ are always better than those for $n \leq 100$. Higher α values are preferable for $n \leq 100$. On the other hand, lower α values are better for $n \geq 250$.

MS + Greedy

We have not tested MS + Greedy algorithm because NAS + Greedy algorithm had results similar to NAS + Louvain algorithm. NPS + Greedy algorithm had slightly worse results than NPS + Louvain algorithm. Thus, we suppose the results for the MS + Greedy algorithm would be slightly worse than those for the MS + Greedy algorithm. In conclusion, MS + Greedy algorithm was not as interesting as other possible combinations of fusion and clustering algorithms.

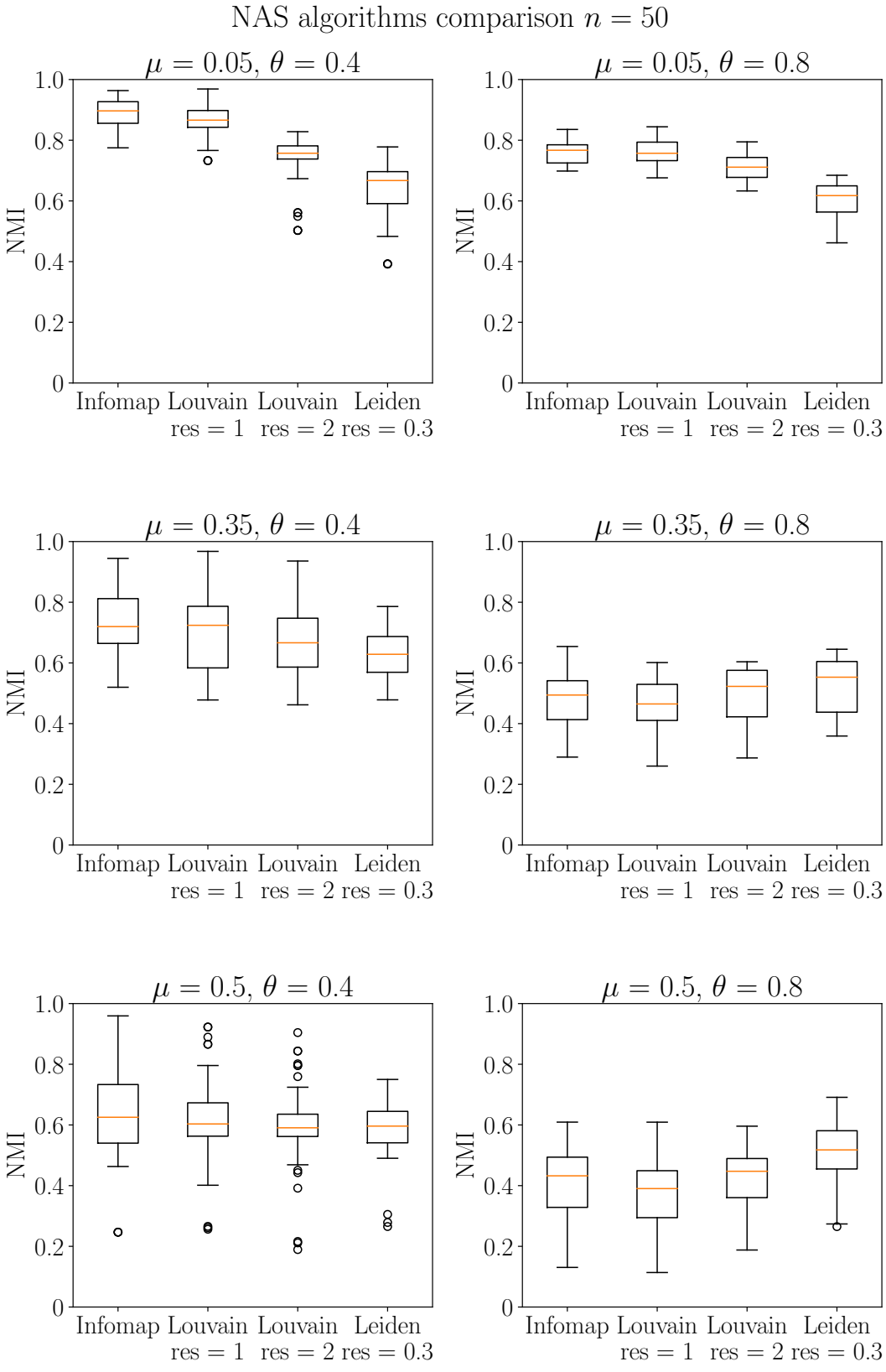


Figure 4.14: NAS + Louvain, Leiden and Infomap algorithms results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

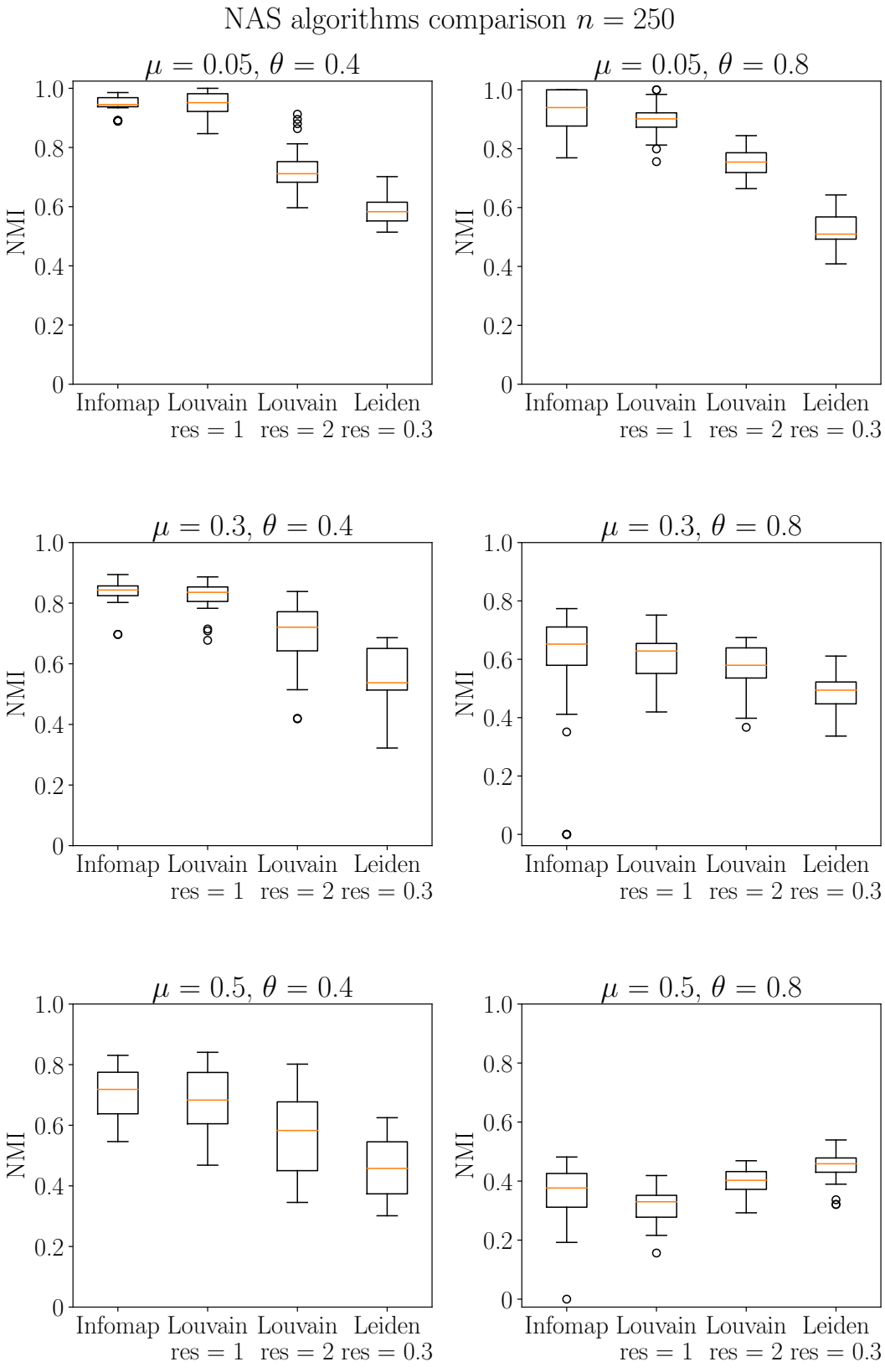


Figure 4.15: NAS + Louvain, Leiden and Infomap algorithms results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

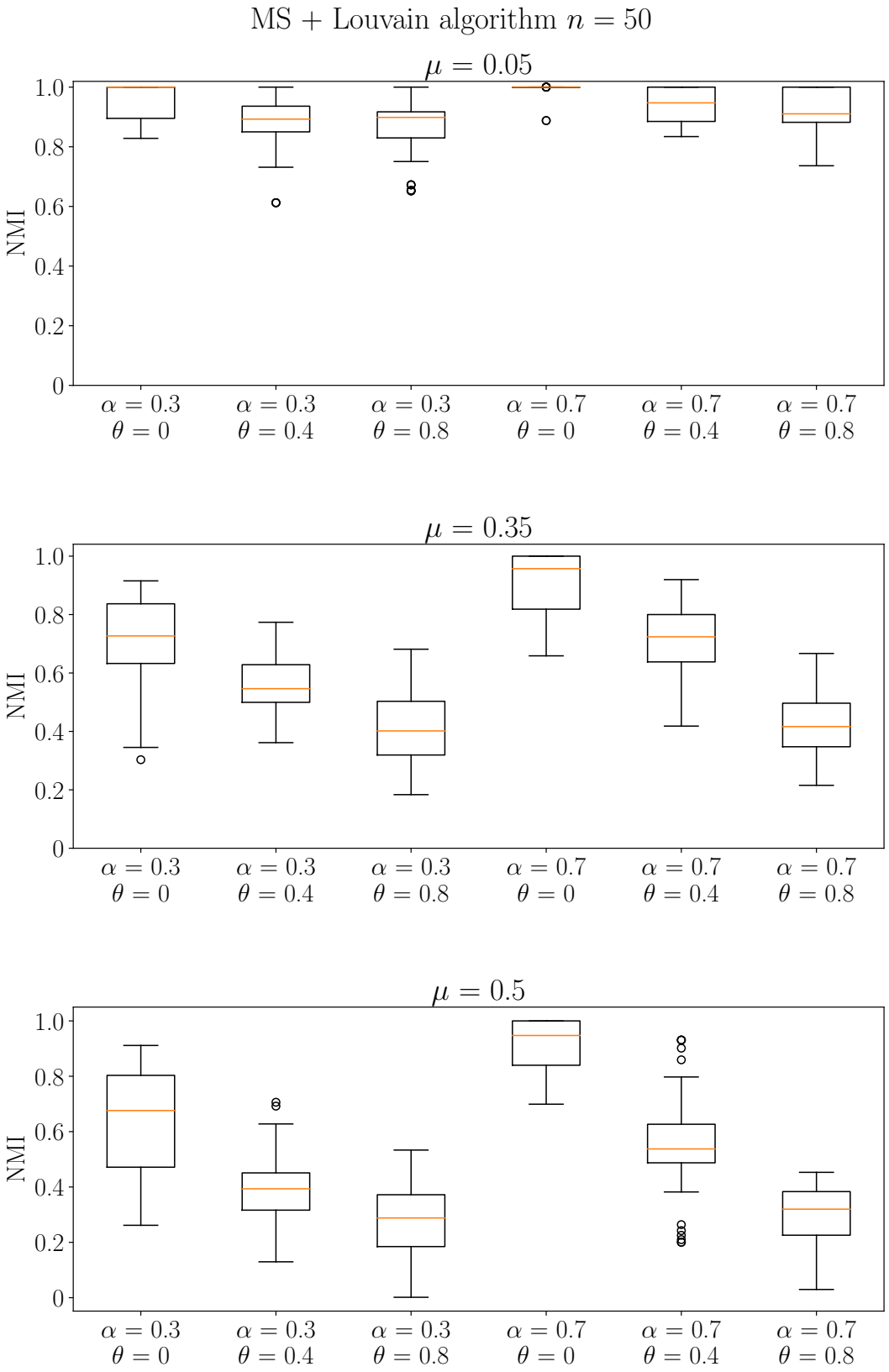


Figure 4.16: MS + Louvain algorithm results for res = 1 and power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

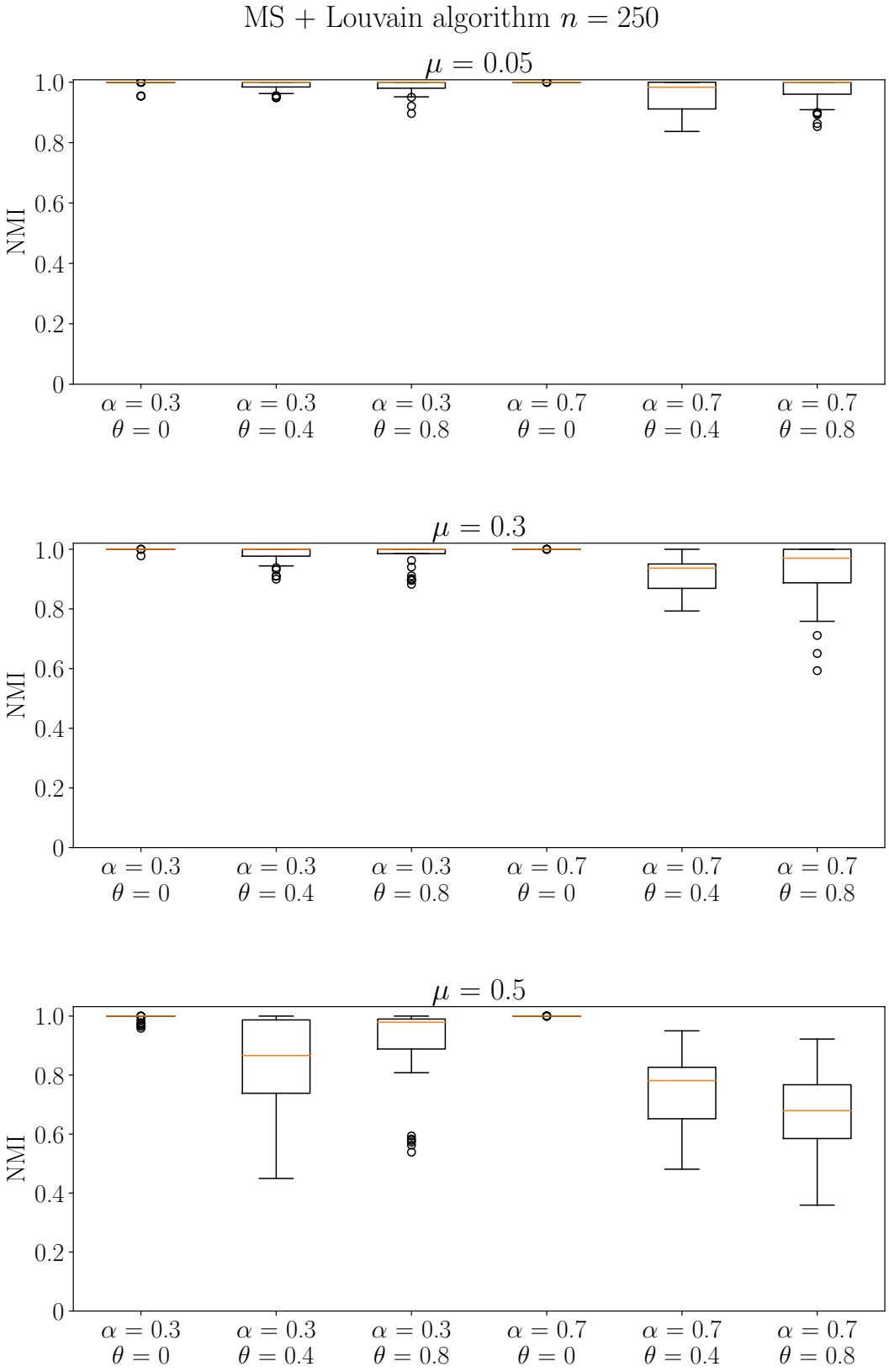


Figure 4.17: MS + Louvain algorithm results for res = 1 and power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

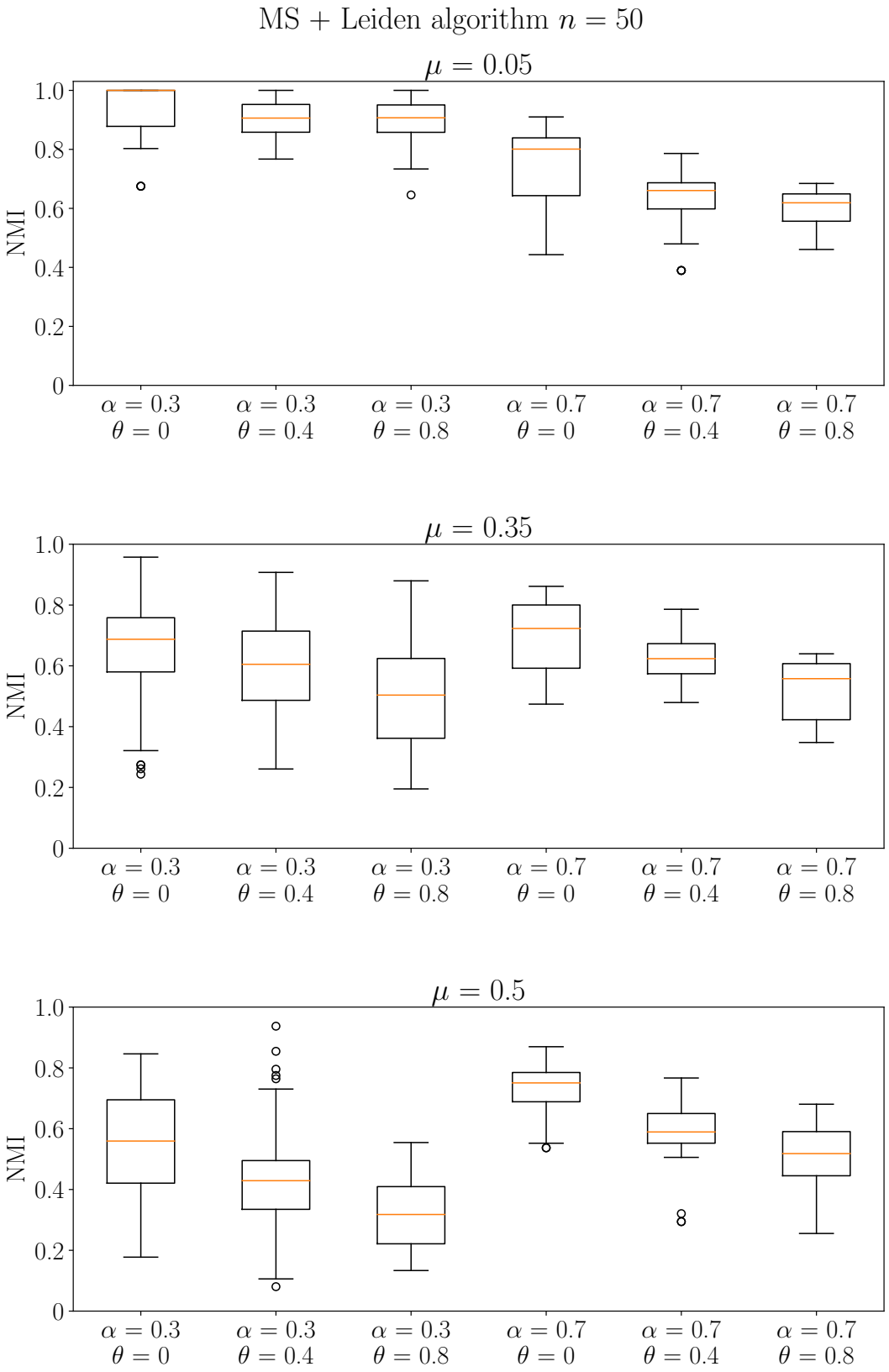


Figure 4.18: MS + Leiden algorithm results for res = 0.5 and power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

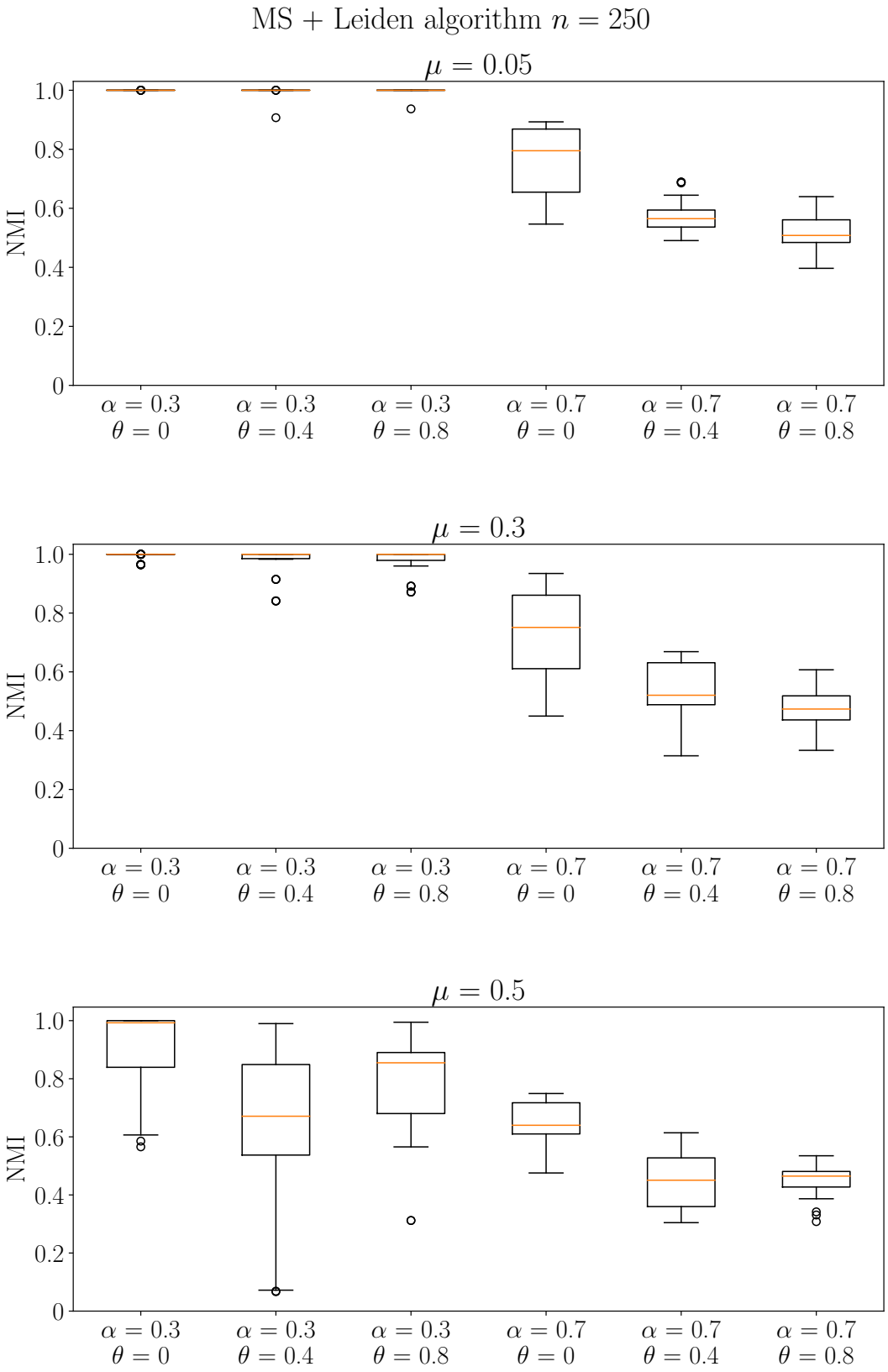


Figure 4.19: MS + Leiden algorithm results for res = 0.5 and power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

MS + Infomap

Results for MS + Infomap are equal to zero (only one large detected community) because of the usage of NPS + Infomap (see Section 4.1.1).

MS algorithms comparison

As we have not tested MS + Greedy algorithm and the results for MS + Infomap algorithm are near zero, we will discuss only the results for MS + Louvain algorithm and MS + Leiden algorithm.

For networks of size $n \leq 100$, MS + Leiden algorithm with $\alpha = 0.3$ outperforms MS + Louvain algorithm with $\alpha = 0.3$ in cases $\theta \geq 0.4$. However, MS + Louvain algorithm with $\alpha = 0.7$ is better than MS + Leiden algorithm with $alpha = 0.7$, except for cases when $\mu =\geq 0.35$ and $\theta \geq 0.8$.

For networks of size $n \geq 250$, MS + Louvain algorithm outperforms MS + Leiden algorithm for all μ and θ values. The results for $\alpha = 0.3$ and $\mu \leq 0.3$ are similar for both algorithms. However, the results differ for $\alpha = 0.7$. In this case, the results for MS + Louvain algorithm are better than those for MS + Leiden algorithm, and the difference between the algorithms is the most markable for $\mu = 0.5$ and $\theta \geq 0.8$.

Overall, MS + Louvain algorithm outperforms MS + Leiden algorithm for bigger networks and all α values. For smaller networks, MS + Louvain algorithm is better than MS + Leiden algorithm with the same α values for $\theta \leq 0.4$ (and for $\theta \leq 0.8$ in case $\alpha = 0.7$).

4.1.4 Comparison of NAS, NPS and MS

Louvain algorithm

For networks of size $n \leq 100$ (Figures 4.20 and 4.21), MS + Louvain has the best results for small $\mu = 0.05$. NAS + Louvain with all resolution values slowly surpasses MS + Louvain for higher μ values because NPS + Louvain with res = 1 negatively influences MS + Louvain. Results of other algorithms for $\theta \geq 0.8$ and higher μ are relatively similar.

For networks of size $n \geq 250$ (Figures 4.22 and 4.23), MS + Louvain outperforms NAS + Louvain and NPS + Louvain for all α values, especially MS + Louvain with $\alpha \leq 0.5$ has exceedingly good results. Only NAS + Louvain with res $\in \{0.5, 1\}$ can compete with MS + Louvain for $\mu = 0.05$, and NPS + Louvain with res = 1 can compete with the average MS + Louvain results, other algorithms drop behind. The difference between algorithms is particularly markable for $\mu = 0.5$ and $\theta \geq 0.8$, where only NPS + Louvain with res = 1 can cope with MS + Louvain. However, NPS + Louvain has a wide variance, so MS + Louvain with $\alpha \leq 0.5$ surpasses it.

Overall, MS + Louvain outperforms both NAS + Louvain and NPS + Louvain, especially for $\mu = 0.5$ and $\theta \geq 0.8$.

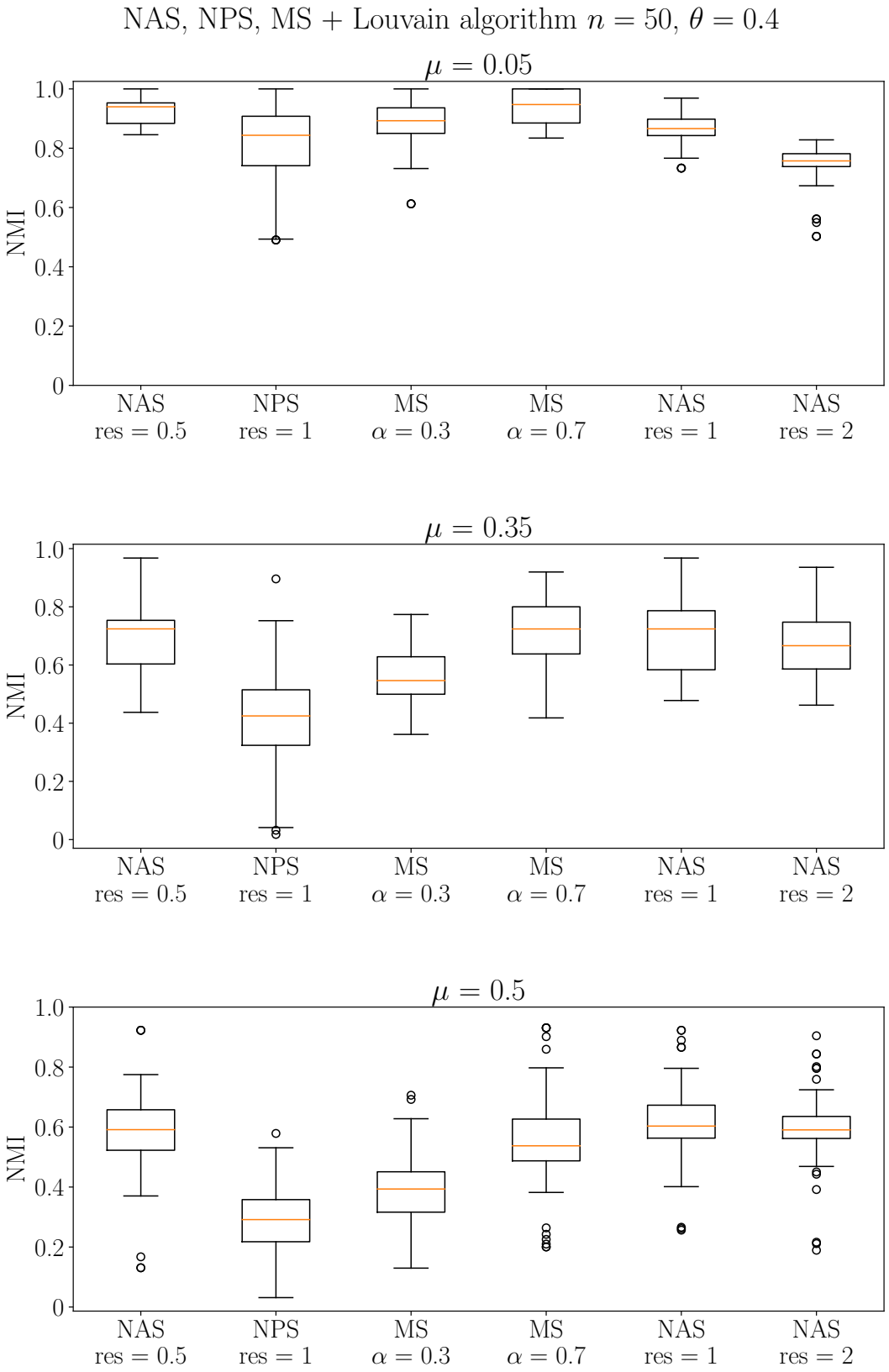


Figure 4.20: NAS, NPS, MS + Louvain algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

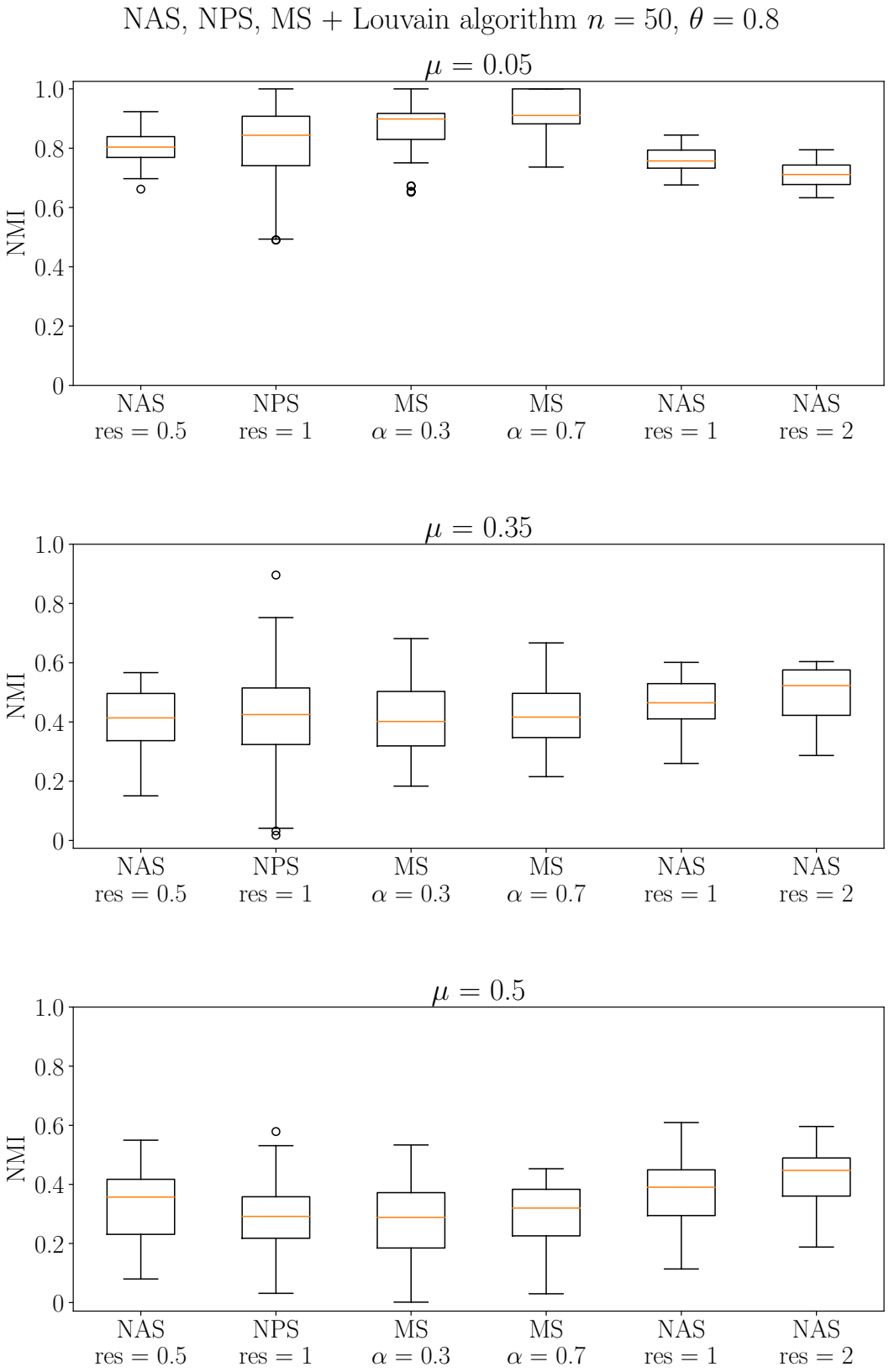


Figure 4.21: NAS, NPS, MS + Louvain algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

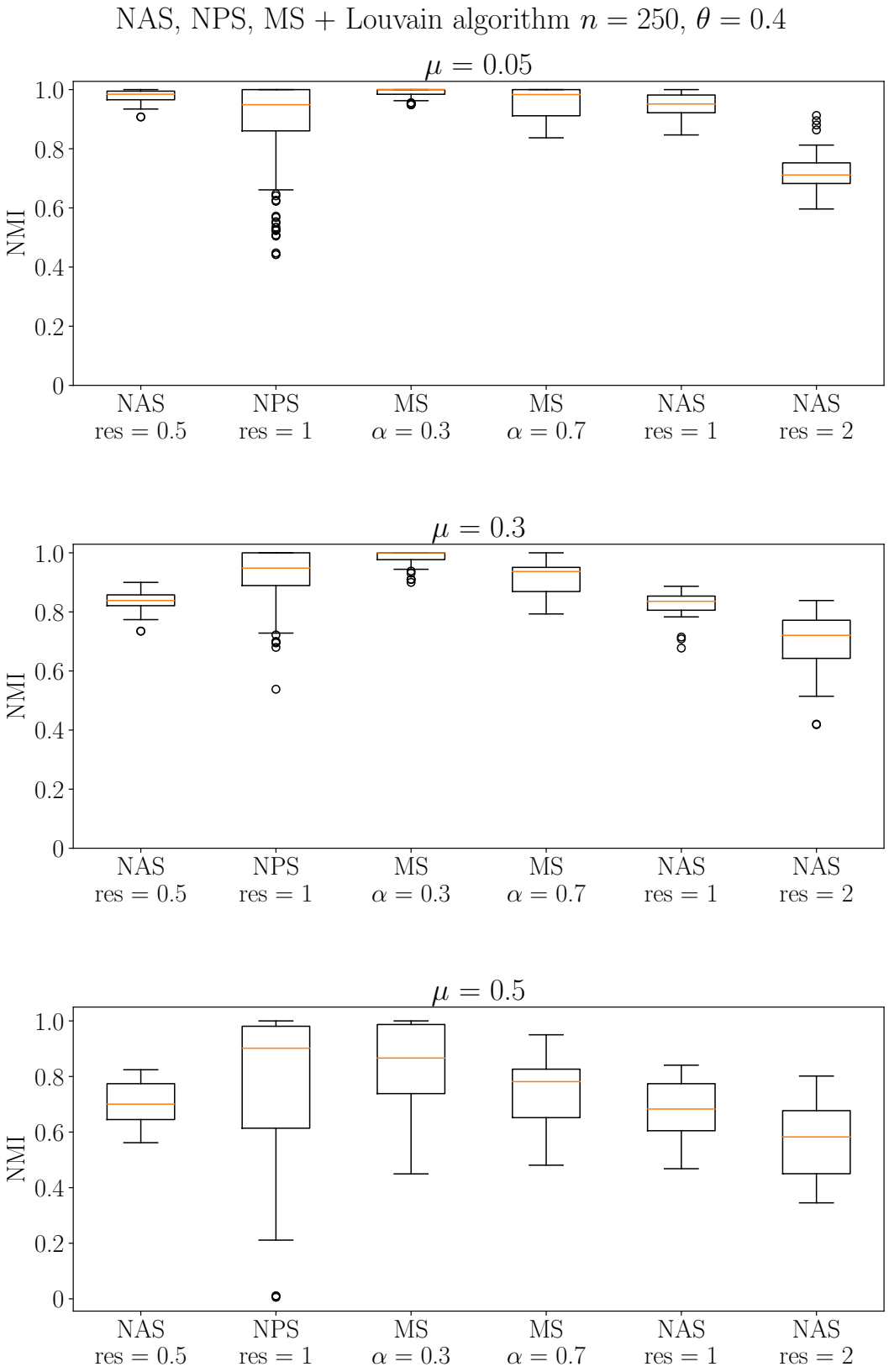


Figure 4.22: NAS, NPS, MS + Louvain algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

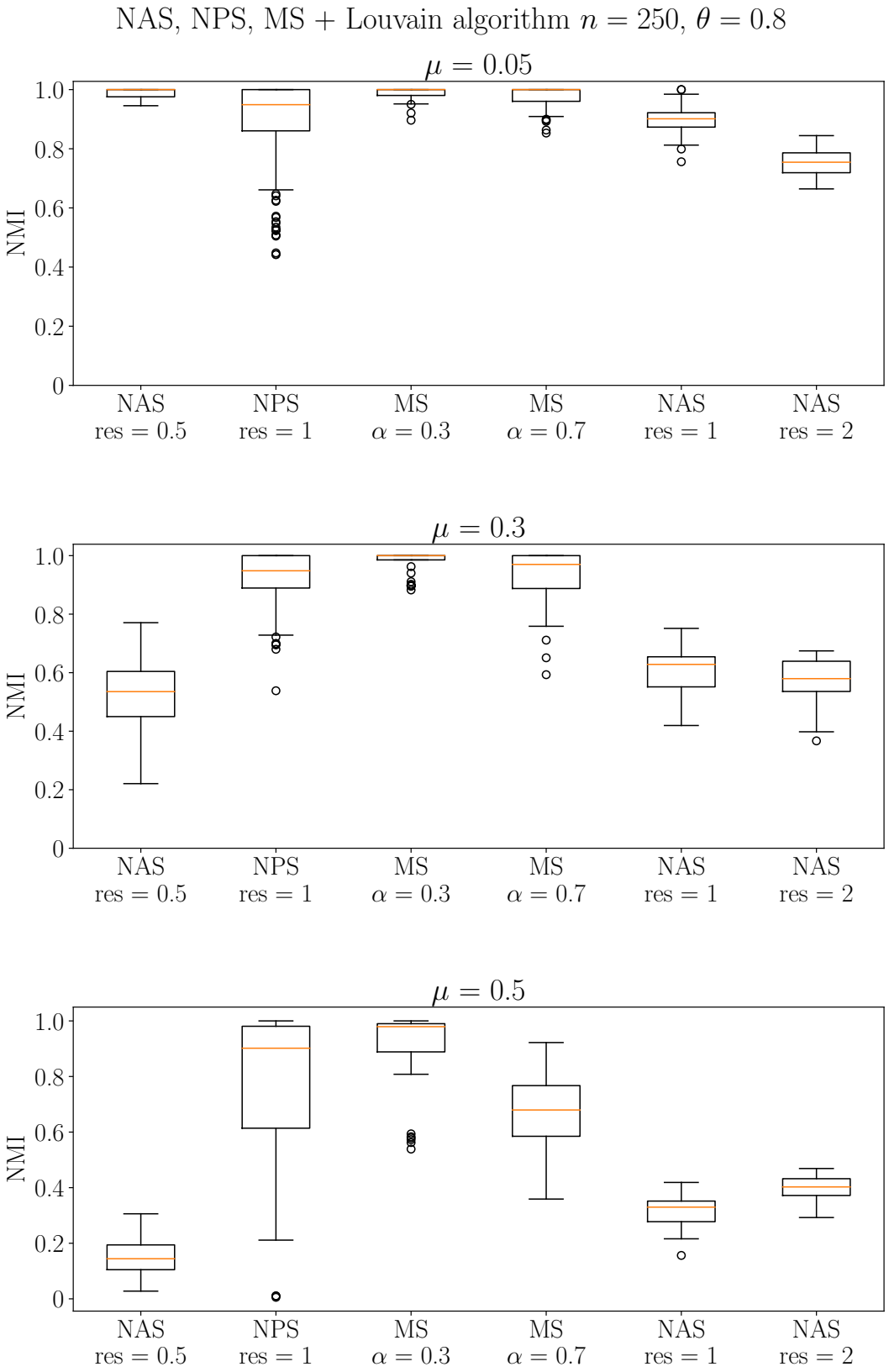


Figure 4.23: NAS, NPS, MS + Louvain algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

Leiden algorithm

Results for MS + Leiden algorithm with $\alpha = 0.7$ and NAS + Leiden algorithm with res = 0.3 were very similar, so we will only show results for MS + Leiden with $\alpha = 0.7$. Results for MS + Leiden with $\alpha = 0.3$ were always better than results for NPS + Leiden with res = 0.7.

For networks of size $n \leq 100$ (Figure 4.24), MS + Leiden with $\alpha = 0.3$ has the best results for small $\mu = 0.05$. For $\mu = 0.35$, NAS + Leiden with res = 0.3, MS + Leiden with $\alpha = 0.7$ and MS + Leiden with $\alpha = 0.3$ have similar results. However, MS + Leiden with $\alpha = 0.3$ has a large variance in results. MS + Leiden with $\alpha = 0.3$ is surpassed by MS + Leiden with $\alpha = 0.7$ and NAS + Leiden with res = 0.3 for $\mu = 0.5$.

For networks of size $n \geq 250$ (Figure 4.25), MS + Leiden with $\alpha = 0.3$ outperforms MS + Leiden with $\alpha = 0.7$ and NAS + Leiden with res = 0.3 for every μ value (even though it has a large variance for $\mu = 0.5$). NPS + Leiden with res = 0.7 is the only algorithm that can compete with MS + Leiden with $\alpha = 0.3$; its results are slightly lower and have larger variance.

For bigger networks, MS + Leiden with $\alpha = 0.3$ outperforms NAS, NPS and MS with $\alpha = 0.7$, especially for $\mu = 0.5$ and $\theta \geq 0.8$. For smaller networks, the results of MS + Leiden with $\alpha = 0.3$ are better for lower μ values, and the results of MS + Leiden with $\alpha = 0.7$ are better for higher μ values.

Greedy algorithm

The results for the Greedy algorithm were similar to the results for the Louvain algorithm.

For networks of size $n \leq 100$ (Figures 4.3 and 4.10), the results for NAS + Greedy with any resolution are better than the results for NPS + Greedy with res = 1.

For networks of size $n \geq 250$ (Figures 4.3 and 4.11), the results for NAS + Greedy with any resolution are better than the results for NPS + Greedy with res = 11, except for the case $\mu \geq 0.3$ and $\theta \geq 0.8$.

Overall, NAS + Greedy with any resolution outperforms NPS + Greedy with res = 1 in most cases. The difference is most markable for bigger networks and lower μ values.

Infomap algorithm

The only relevant results for Infomap are for NAS + Infomap because NPS + Infomap and MS + Infomap detected only one large community containing all nodes (and so the results are near zero for this fusion method).

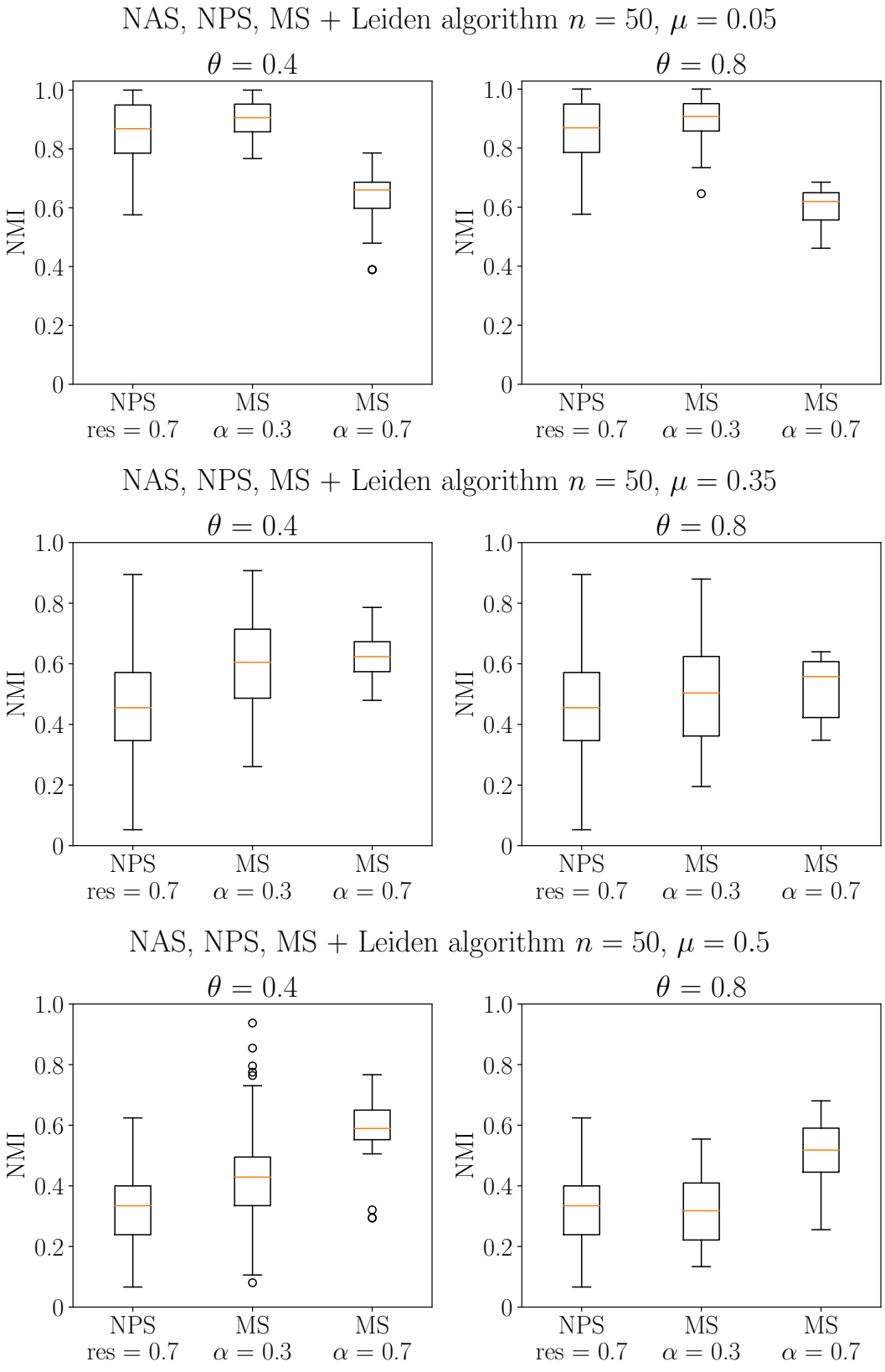


Figure 4.24: NAS, NPS, MS + Leiden algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

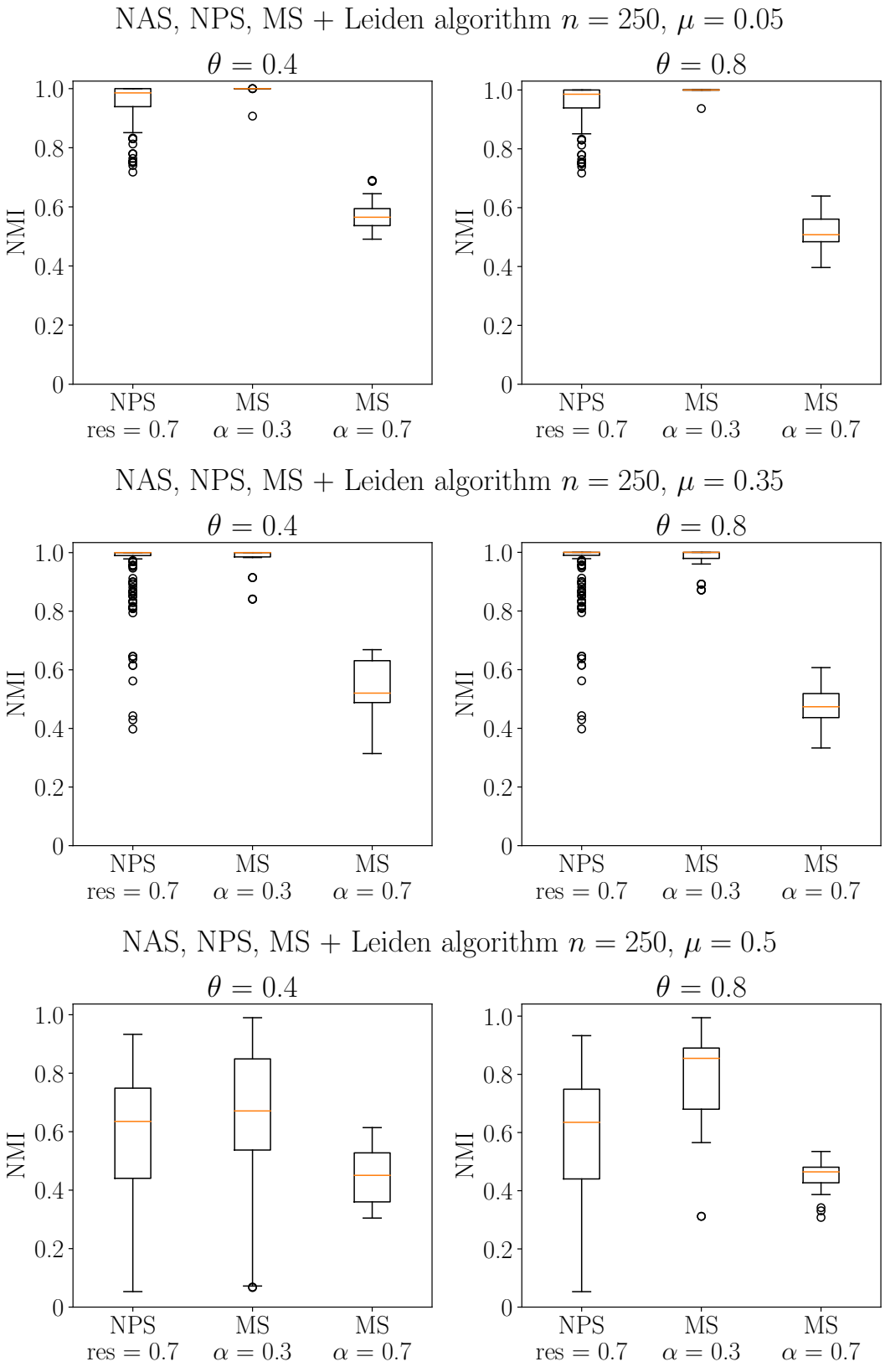


Figure 4.25: NAS, NPS, MS + Leiden algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

4.2 Simultaneous fusion methods

Eva

The results for res = 2 are relevant only for $n \leq 100$. For bigger networks, they dropped rapidly to zero. Since the results for res = 2 were consistently lower than or equal to the results for res = 1, we show only results for res = 1.

For networks of size $n \leq 100$ (Figure 4.26), res = 1 outperforms res = 0.5 for $\mu \leq 0.35$. In the case $\mu = 0.5$, the results for res = 0.5 are slightly better than for res = 1. All results are decreasing with higher μ and θ values.

For networks of size $n \geq 250$ (Figure 4.27), results for res = 1 are clearly better than the results for res = 0.5. The results for res = 0.5 are only slightly decreasing for higher μ and θ values. The choice of res = 1 is negatively influenced by μ and θ only in case $\mu = 0.5$.

Overall, the results for bigger networks are better than those for smaller networks. The difference between results is most markable for res = 1 and $\mu \geq 0.3$. All results are decreasing with higher μ and θ values, however, for networks of size $n \geq 250$ and res = 1, the results are around 1 even for $\mu = 0.3$ and $\theta = 0.8$. res = 1 clearly surpasses res = 0.5 for bigger networks, having exceedingly good results.

4.3 Discussion

This section presents a discussion of the impact of parameters on the algorithm's results.

4.3.1 Mixing parameter and noise parameter impact

Algorithms using NPS

For NPS, in general, the θ parameter had no impact on the results, as the NPS does not use the attributes. For NPS + Louvain, NPS + Leiden and NPS + Greedy algorithms with the choice res = 2, the results were not influenced by higher μ values. This resolution favours smaller communities. Thus the algorithm usually finds many small communities with only a few nodes. Such communities are unaffected by higher μ because of their small size – for higher μ values, the algorithm detects almost the same small communities.

NPS + Louvain and NPS + Greedy with the choice res = 1 were negatively influenced by higher μ values because the resolution parameter favours neither small nor large communities. With higher μ values, the communities are more interconnected, making community detection harder. For bigger networks and $\mu = 0.5$, the results for both algorithms had a large variance.

Also, NPS + Leiden with the choice res = 0.7 was negatively influenced by higher μ values, and its results had a large variance for $\mu \geq 0.3$. Considering the results for NPS + Louvain algorithm with res = 1 and NPS + Leiden algorithm with res = 0.7, for networks of size $n \geq 250$, the results were less influenced by μ than the results for networks of size $n \leq 100$.

It can be observed (Figure 4.3) that NPS + Greedy algorithm with the choice res = 1 achieved the best results for bigger networks for the μ settings $\mu = 0.3$. It

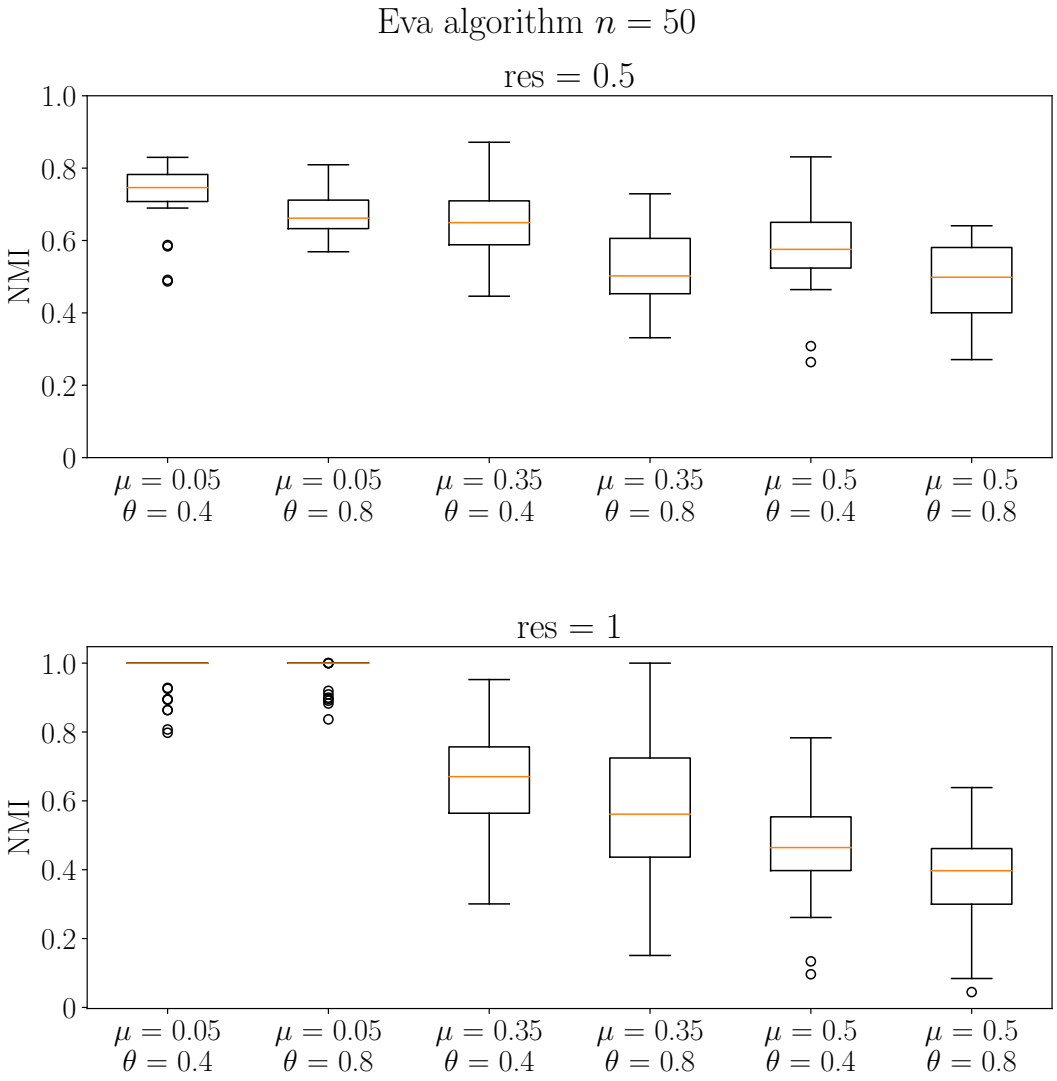


Figure 4.26: Eva algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

can be caused by the greediness of the algorithm but further experiments would be necessary to confirm it.

Algorithms using NAS

The results for NAS + Louvain, NAS + Greedy and NAS + Infomap were influenced by μ and θ in the same way, so we will describe only results for NAS + Louvain.

The results for NAS + Louvain with res = 0.5 were more negatively influenced by higher μ and θ values than results for higher resolutions because res = 0.5 favours bigger communities (see Figure 4.7). Higher μ values cause the communities to be more interconnected, and the algorithm had problems with their separation – the mixing of bigger communities causes a larger NMI decrease than the mixing of smaller communities. On the other hand, the results for NAS + Louvain with res = 2 were less negatively influenced by higher μ and θ values than results with lower resolutions because res = 2 favours smaller communities.

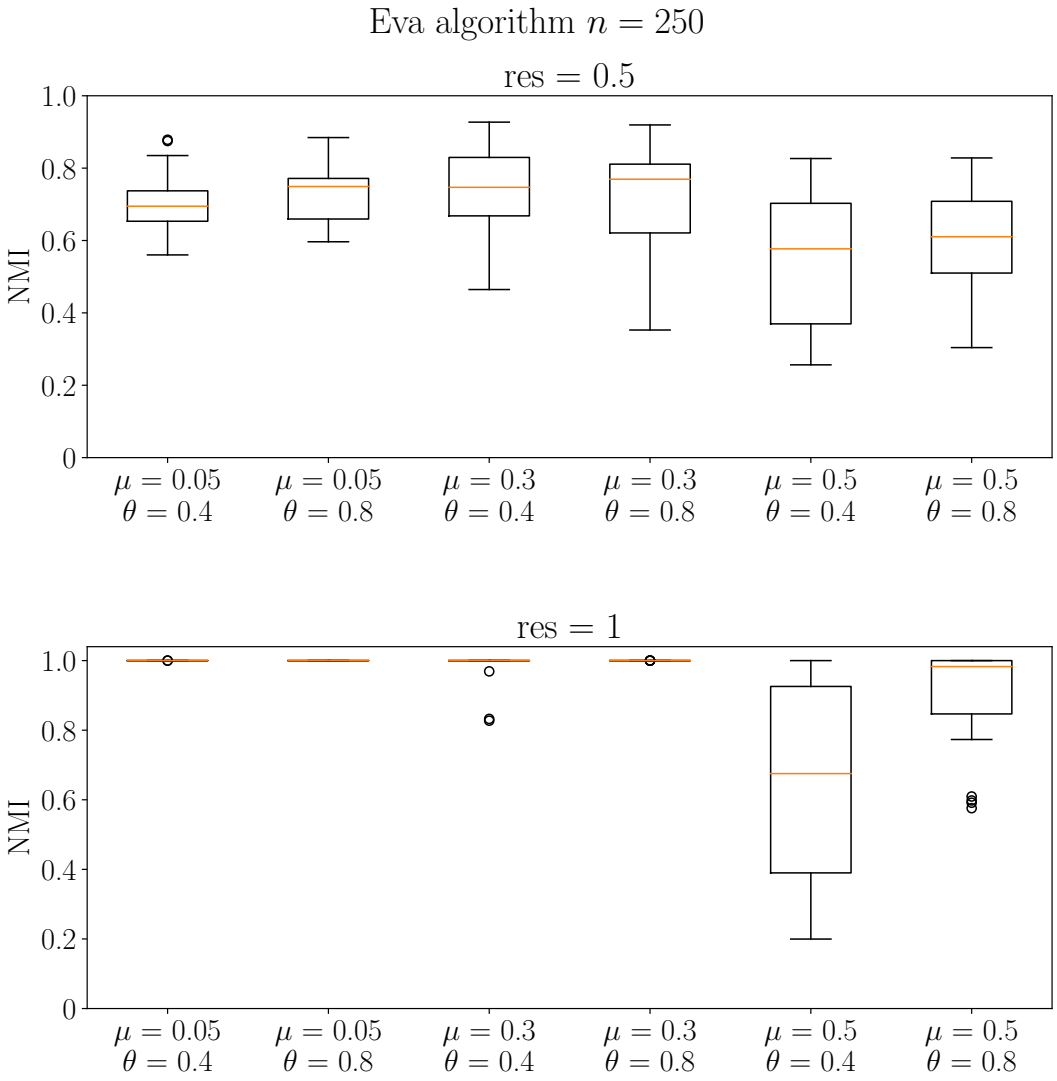


Figure 4.27: Eva algorithm results for power-law distribution exponents $\tau_1 = 2$ and $\tau_2 = 2$

Results for NAS + Louvain with res = 1 were negatively influenced by higher μ and θ values, too, and placed between results for res = 0.5 and res = 2.

For $\mu = 0.05$, results for NAS + Louvain with any resolution were less influenced by θ than the results for $\mu = 0.5$. This is caused by the fact that NAS uses attributes and adds weights only between nodes which were connected in the former graph. For $\mu = 0.05$, higher θ values caused fewer edges remained in the weighted graph (as the output of a fusion algorithm). If there was an edge with nonzero weight, it usually connected nodes in the same community because of the low μ value. On the other hand, having $\mu = 0.5$, the edges with nonzero weights often connected nodes in different communities and thus, the results got worse with higher θ values. Overall, all results were decreasing for higher μ and θ values, which aligns with our expectations.

NAS + Leiden algorithm with res = 0.3 was also more negatively influenced by higher μ and θ values than NAS + Leiden algorithm with res = 0.5 because res = 0.3 favours bigger communities (Figure 4.8). For networks of size $n \geq 250$ (Figure 4.9), results for $\theta \geq 0.4$ and a specific resolution are very similar for all

μ values. The variance of results for specific μ decreases with higher θ values. Overall, all results are decreasing for higher μ and θ values, which aligns with our expectations.

Algorithms using MS

For smaller networks, results for both MS + Louvain and MS + Leiden were negatively influenced by higher μ and θ values, which coincides with our expectations. For both algorithms, the choice of $\alpha = 0.7$ was better for higher μ values, which we expected because $\alpha = 0.3$ uses more NPS weights and thus, its results are more negatively influenced by higher μ values than the results for $\alpha = 0.7$. The results for MS + Louvain with any α value were similar for $\theta \geq 0.8$.

For bigger networks, the results for MS + Louvain with $\alpha = 0.3$ and MS + Leiden with $\alpha = 0.3$ were not influenced by higher μ and θ values (they were influenced only a little for $\mu = 0.5$). Also, $\alpha = 0.3$ outperformed $\alpha = 0.7$ for both algorithms and all μ values.

Simultaneous fusion algorithms

Results for Eva with any resolution were negatively influenced by higher μ and θ values. However, the results were not as influenced by higher θ values as results for NAS + Louvain. The results for Eva with res = 0.5 were less influenced by higher μ values than Eva with res = 1. However, for networks of size $n \geq 250$ and Eva with res = 1, the results are around 1 even for $\mu = 0.3$ and $\theta = 0.8$. This is caused by the bigger size of the network.

4.3.2 Summary

In conclusion, we have confirmed that higher μ values negatively influence all algorithms and that higher θ values negatively influence algorithms using NAS. The algorithms (mainly Leiden algorithm) are less influenced by μ when the size of the network is bigger. We observed, that the algorithms with NPS had larger variance of results for higher μ values. We conclude, that the combination of MS with Louvain algorithm can reach very good results and almost cope with Eva. Both these algorithms were resistant to the higher μ values, but only for larger networks.

Conclusion

Community detection is a helpful tool for deepening our knowledge of complex systems. The accuracy of community detection on complex networks can be improved by using networks with node attributes. In this work, we have compared several node-attributed community detection algorithms and the impact of the parameters of the results of the algorithms.

The theoretical part of this work introduced the terminology regarding communities and node-attributed graphs. We have found that the canonic definition of a community does not exist. However, the definition is usually not essential for community detection. Afterwards, we showed some interesting properties of real-world networks, such as the power-law node degree distribution. Furtherly, we have done a small survey about the community sizes and their connection with power law. We have not found any systematic study supporting the power-law distribution of community sizes; only a few articles observed this distribution on real-world networks. Notwithstanding, many authors use the power-law community size distribution in their works as a fact.

Moving onto the comparison of algorithms, we also presented some graph benchmarks – GN, LFR, acMark, X-Mark, LFR-EA, ANC – and evaluation metrics – Adjusted Rand Index, Normalised Mutual Information – which can be used to test the algorithms. We provided an overview of works related to comparing the algorithms and found only two truly comparative studies of node-attributed community detection algorithms carried out on synthetic networks.

In the practical part of this work, we carried out a comparative analysis of the node-attributed community detection algorithm. This comparison was made entirely on synthetic networks generated with the X-Mark benchmark, and we have used Normalised Mutual Information to evaluate the results. For simplification, we have divided the algorithms into three classes according to a moment when the structure and the attributes of the network are being fused in the community detection process – early fusion methods, simultaneous fusion methods and late fusion methods. We have tested one simultaneous fusion method (Eva algorithm) and several early fusion methods. The early fusion methods consist of a fusion and a clustering algorithm. We have used Node Attribute Similarity, Node Path Similarity and Mixed Similarity for the fusion and Louvain, Greedy modularity, Leiden and Infomap algorithms for the clustering.

We have also studied the setting of the benchmark and algorithms parameters, which was an essential part of the analysis. Finally, we have discussed the obtained results and how the benchmark and the algorithm parameters influence them. We confirmed that higher values of the mixing parameter negatively influence all algorithms and that higher noise parameter values negatively influence algorithms considering attributes. We have also observed that the algorithms are less influenced by the mixing parameters when the network size is bigger. The algorithms that used Node Path Similarity had a larger variance of results for higher mixing parameter values. The algorithms that used Mixed Similarity were resistant to higher values of mixing and noise parameters, but only for larger networks. Finally, we obtained very high Normalised Mutual Information values using Mixed Similarity and the Louvain algorithm. The Eva algorithm is the

only algorithm that outperforms Mixed Similarity with the Louvain algorithm. However, the Eva algorithm is slower than Mixed Similarity with Louvain.

During our work, we have found many topics for possible further studies. The first one is the analysis of the community size distributions of real-world networks. A thorough study is needed to support or confute the claim that community sizes follow power-law distribution (Section 1.2.2). As noted in the Related works section (Section 2.3), there are only a few truly comparative studies of node-attributed community detection algorithms. Although many small studies are carried out by the authors when introducing a new community detection algorithm, the studies are insufficient because they are not systematic, use genuinely different methodological approaches, and are usually done on a few networks. Further studies can also concern other node-attributed benchmarks, such as acMark and ANC. It could be interesting to examine their parameters and the impact on the accuracy of the algorithm.

Bibliography

[1] Esra Akbas and Peixiang Zhao. *Graph Clustering Based on Attribute-Aware Graph Embedding*, pages 109–131. 04 2019. ISBN 978-3-030-02058-3. doi: 10.1007/978-3-030-11286-8_5.

[2] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002. doi: 10.1103/RevModPhys.74.47. URL https://link.aps.org/doi/10.1103/RevModPhys.74.47.

[3] Esmaeil Alinezhad, Babak Teimourpour, Mohammad Mehdi Sepehri, and Mehrdad Kargari. Community detection in attributed networks considering both structural and attribute similarities: two mathematical programming approaches. *Neural Computing and Applications*, 32, 04 2020. doi: 10.1007/s00521-019-04064-5.

[4] A. Arenas, L. Danon, A. Díaz-Guilera, P. M. Gleiser, and R. Guimerà. Community analysis in social networks. *European Physical Journal B: Condensed Matter*, 38(2):373–380, March 2004. ISSN 1434-6028. doi: 10.1140/epjb/e2004-00130-1.

[5] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi: 10.1126/science.286.5439.509. URL https://www.science.org/doi/abs/10.1126/science.286.5439.509.

[6] Albert-László Barabási and Márton Pósfai. *Network Science*. Cambridge University Press, 2016. ISBN 9781107076266. URL https://books.google.cz/books?id=iLtGDQAAQBAJ.

[7] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, 272(1):173–187, 1999. ISSN 0378-4371. doi: https://doi.org/10.1016/S0378-4371(99)00291-5. URL https://www.sciencedirect.com/science/article/pii/S0378437199002915.

[8] Kamal Berahmand, Sogol Haghani, Mehrdad Rostami, and Yuefeng Li. A new attributed graph clustering by using label propagation in complex networks. *J. King Saud Univ. Comput. Inf. Sci.*, 34:1869–1883, 2020.

[9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008. doi: 10.1088/1742-5468/2008/10/p10008. URL https://doi.org/10.1088%2F1742-5468%2F2008%2F10%2Fp10008.

[10] Shyam Boriah, Varun Chandola, and Vipin Kumar. *Similarity Measures for Categorical Data: A Comparative Evaluation*, pages 243–254. doi: 10.1137/1.9781611972788.22. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611972788.22.

[11] Cecile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenkovã. Clustering attributed graphs: Models, measures and methods. *Network Science*, 3(3):408–444, September 2015. URL https://ideas.repec.org/a/cup/netsci/v3y2015i03p408-444_00.html.

[12] Tanmoy Chakraborty, Ayushi Dalmia, Animesh Mukherjee, and Niloy Ganguly. Metrics for community analysis: A survey, 2016. URL https://arxiv.org/abs/1604.03512.

[13] Theerasak Chanwimalueang and Danilo Mandic. Cosine similarity entropy: Self-correlation-based complexity analysis of dynamical systems. *Entropy*, 19(12):652, Nov 2017. ISSN 1099-4300. doi: 10.3390/e19120652. URL http://dx.doi.org/10.3390/e19120652.

[14] Petr Chunaev. Community detection in node-attributed social networks: A survey. *Computer Science Review*, 37:100286, 2020. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2020.100286. URL https://www.sciencedirect.com/science/article/pii/S1574013720303865.

[15] Petr Chunaev, Timofey Gradov, and Klavdiya Bochenina. Community detection in node-attributed social networks: How structure-attributes correlation affects clustering quality. *Procedia Computer Science*, 178:355–364, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.11.037. URL https://www.sciencedirect.com/science/article/pii/S1877050920324169. 9th International Young Scientists Conference in Computational Science, YSC2020, 05-12 September 2020.

[16] Petr Chunaev, Timofey Gradov, and Klavdiya Bochenina. The machinery of the weight-based fusion model for community detection in node-attributed social networks. *Social Network Analysis and Mining*, 11, 12 2021. doi: 10.1007/s13278-021-00811-6.

[17] Salvatore Citraro and Giulio Rossetti. Eva: Attribute-aware network segmentation. *CoRR*, abs/1910.06599, 2019. URL http://arxiv.org/abs/1910.06599.

[18] Salvatore Citraro and Giulio Rossetti. X-mark: a benchmark for node-attributed community discovery algorithms. *Social Network Analysis and Mining*, 11, 12 2021. doi: 10.1007/s13278-021-00823-2.

[19] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, Dec 2004. doi: 10.1103/PhysRevE.70.066111. URL https://link.aps.org/doi/10.1103/PhysRevE.70.066111.

[20] David Combe, Christine Largeron, Elöd Egyed-Zsigmond, and Mathias Géry. Getting clusters from structure data and attribute data. *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 710–712, 2012. doi: 10.1109/ASONAM.2012.123.

[21] David Combe, Christine Largeron, Mathias Géry, and Elod Egyed-Zsigmond. I-louvain: An attributed graph clustering method. 10 2015. ISBN 978-3-319-24464-8. doi: 10.1007/978-3-319-24465-5_16.

[22] Anne Condon and Richard M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2):116–140, mar 2001. ISSN 1042-9832.

[23] Anh Dang and Emmanuel Viennet. Community detection based on structural and attribute similarities. In *International Conference on the Digital Society*, 2012.

[24] Leon Danon, Jordi Duch, Alex Arenas, and Albert Díaz-Guilera. *Community Structure Identification*, pages 93–114. doi: 10.1142/9789812771681_0006. URL https://www.worldscientific.com/doi/abs/10.1142/9789812771681_0006.

[25] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, sep 2005. doi: 10.1088/1742-5468/2005/09/P09008. URL https://dx.doi.org/10.1088/1742-5468/2005/09/P09008.

[26] Haithum Elhadi and Gady Agam. Structure and attributes community detection: Comparative analysis of composite, ensemble and selection methods. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, SNAKDD '13. Association for Computing Machinery, 2013. ISBN 9781450323307. doi: 10.1145/2501025.2501034. URL https://doi.org/10.1145/2501025.2501034.

[27] Issam Falih, Nistor Grozavu, Rushed Kanawati, and Younès Bennani. Community detection in attributed network. In *Companion Proceedings of the The Web Conference 2018*, pages 1299–1306. International World Wide Web Conferences Steering Committee, 2018. ISBN 9781450356404. doi: 10.1145/3184558.3191570. URL https://doi.org/10.1145/3184558.3191570.

[28] Issam Falih, Nistor Grozavu, Rushed Kanawati, and Younès Bennani. Anca : Attributed network clustering algorithm. In Chantal Cherifi, Hocine Cherifi, Márton Karsai, and Mirco Musolesi, editors, *Complex Networks & Their Applications VI*, pages 241–252. Springer International Publishing, 2018. ISBN 978-3-319-72150-7.

[29] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '99, pages 251–262, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 1581131356. doi: 10.1145/316188.316229. URL https://doi.org/10.1145/316188.316229.

[30] Walter D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958. doi: 10.1080/

01621459.1958.10501479. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1958.10501479.

[31] Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans M. Coetzee. Self-organization and identification of web communities. *Computer*, 35(3): 66–71, mar 2002. ISSN 0018-9162. doi: 10.1109/2.989932. URL https://doi.org/10.1109/2.989932.

[32] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486 (3-5):75–174, feb 2010. doi: 10.1016/j.physrep.2009.11.002. URL https://doi.org/10.1016%2Fj.physrep.2009.11.002.

[33] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, nov 2016. doi: 10.1016/j.physrep.2016.09.002. URL https://doi.org/10.1016%2Fj.physrep.2016.09.002.

[34] Ana L. N. Fred and Anil K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II, 2003. doi: 10.1109/CVPR.2003.1211462.

[35] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics, 2007. doi: 10.1137/1.9780898718348. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898718348.

[36] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99 (12):7821–7826, jun 2002. doi: 10.1073/pnas.122653799. URL https://doi.org/10.1073%2Fpnas.122653799.

[37] Pablo M. Gleiser and Leon Danon. Community structure in jazz. *Advances in Complex Systems*, 06(04):565–573, 2003. doi: 10.1142/S0219525903001067. URL https://doi.org/10.1142/S0219525903001067.

[38] Neil Zhenqiang Gong, Ameet Talwalkar, Lester Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine, Shi, and Dawn Song. Jointly predicting links and inferring attributes using a social-attribute network (san), 2012.

[39] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, 1971. ISSN 0006341X, 15410420. URL http://www.jstor.org/stable/2528823.

[40] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68:065103, Dec 2003. doi: 10.1103/PhysRevE.68.065103. URL https://link.aps.org/doi/10.1103/PhysRevE.68.065103.

[41] Yanqing Hu, Hongbin Chen, Peng Zhang, Menghui Li, Zengru Di, and Ying Fan. Comparative definition of community and corresponding identifying algorithm. *Phys. Rev. E*, 78:026121, aug 2008. doi: 10.1103/PhysRevE.78.026121. URL https://link.aps.org/doi/10.1103/PhysRevE.78.026121.

[42] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. URL https://EconPapers.repec.org/RePEc:spr:jclass:v:2:y:1985:i:1:p:193-218.

[43] Sid L. Huff. Decomposition of Weighted Graphs Using the Interchange Partitioning Technique. Technical report, 1979. URL https://apps.dtic.mil/sti/citations/ADA069549.

[44] Roberto Interdonato, Martin Atzmueller, Sabrina Gaito, Rushed Kanawati, Christine Largeron, and Alessandra Sala. Feature-rich networks: going beyond complex network topologies. *Applied Network Science*, 4, 01 2019. doi: 10.1007/s41109-019-0111-x.

[45] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, 2005. ISBN 9780471735786.

[46] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell System Technical Journal*, 49(2):291–307, feb 1970. ISSN 8756-2324. doi: 10.1002/j.1538-7305.1970.tb01770.x.

[47] L.I. Kuncheva and S.T. Hadjitodorov. Using diversity in cluster ensembles. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 2, pages 1214–1219 vol.2, 2004. doi: 10.1109/ICSMC.2004.1399790.

[48] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5), nov 2009. doi: 10.1103/physreve.80.056117. URL https://doi.org/10.1103%2Fphysreve.80.056117.

[49] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1), jul 2009. doi: 10.1103/physreve.80.016118. URL https://doi.org/10.1103%2Fphysreve.80.016118.

[50] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78 (4), oct 2008. doi: 10.1103/physreve.78.046110. URL https://doi.org/10.1103%2Fphysreve.78.046110.

[51] Christine Largeron, Pierre-Nicolas Mougel, Reihaneh Rabbany, and Osmar R. Zaïane. Generating attributed networks with communities. *PLOS ONE*, 10(4):1–21, 04 2015. doi: 10.1371/journal.pone.0122777. URL https://doi.org/10.1371/journal.pone.0122777.

[52] Soojung Lee. Improving jaccard index for measuring similarity in collaborative filtering. pages 799–806, 03 2017. ISBN 978-981-10-4153-2. doi: 10.1007/978-981-10-4154-9_93.

[53] Chang Liu, Christine Largeron, Osmar R. Zaïane, and Shiva Zamani Gharaghooshi. A late-fusion approach to community detection in attributed networks. In *Advances in Intelligent Data Analysis XVIII: 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings*, pages 300–312. Springer-Verlag, 2020. ISBN 978-3-030-44583-6. doi: 10.1007/978-3-030-44584-3_24. URL https://doi.org/10.1007/978-3-030-44584-3_24.

[54] Seiji Maekawa, Jianpeng Zhang, George Fletcher, and Makoto Onizuka. *General Generator for Attributed Graphs with Community Structure*. 09 2019.

[55] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. doi: 10.1017/CBO9780511809071.

[56] Fanrong Meng, Xiaobin Rui, Zhixiao Wang, Yan Xing, and Longbing Cao. Coupled node similarity learning for community detection in attributed networks. *Entropy*, 20(6), 2018. ISSN 1099-4300. doi: 10.3390/e20060471. URL https://www.mdpi.com/1099-4300/20/6/471.

[57] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595939081. doi: 10.1145/1298306.1298311. URL https://doi.org/10.1145/1298306.1298311.

[58] Ruchi Mittal and M. P. S. Bhatia. Classification and comparative evaluation of community detection algorithms. *Archives of Computational Methods in Engineering*, 28:1417 – 1428, 2020.

[59] M. E. J. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Phys. Rev. E*, 64:016131, Jun 2001. doi: 10.1103/PhysRevE.64.016131. URL https://link.aps.org/doi/10.1103/PhysRevE.64.016131.

[60] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004. doi: 10.1103/PhysRevE.69.066133. URL https://link.aps.org/doi/10.1103/PhysRevE.69.066133.

[61] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, 03 2010. ISBN 9780199206650. doi: 10.1093/acprof:oso/9780199206650.001.0001. URL https://doi.org/10.1093/acprof:oso/9780199206650.001.0001.

[62] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), feb 2004. doi: 10.1103/physreve.69.026113. URL https://doi.org/10.1103%2Fphysreve.69.026113.

[63] Mark Newman. Power-Law Distribution. *Significance*, 14(4):10–11, 08 2017. ISSN 1740-9705. doi: 10.1111/j.1740-9713.2017.01050.x. URL https://doi.org/10.1111/j.1740-9713.2017.01050.x.

[64] Mark E. J. Newman. Detecting community structure in networks. *The European Physical Journal B*, 38:321–330, 2004.

[65] MEJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351, sep 2005. doi: 10.1080/00107510500052444. URL https://doi.org/10.1080%2F00107510500052444.

[66] Günce Keziban Orman and Vincent Labatut. A comparison of community detection algorithms on artificial networks. In *Proceedings of the 12th International Conference on Discovery Science*, DS '09, pages 242–256, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 9783642047466. doi: 10.1007/978-3-642-04747-3_20. URL https://doi.org/10.1007/978-3-642-04747-3_20.

[67] Günce Keziban Orman, Vincent Labatut, and Hocine Cherifi. Qualitative comparison of community detection algorithms. In *Communications in Computer and Information Science*, pages 265–279. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-22027-2_23. URL https://doi.org/10.1007%2F978-3-642-22027-2_23.

[68] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, jun 2005. doi: 10.1038/nature03607. URL https://doi.org/10.1038%2Fnature03607.

[69] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Using pagerank to characterize web structure. In *Proceedings of the 8th Annual International Conference on Computing and Combinatorics*, COCOON '02, pages 330–339, Berlin, Heidelberg, 2002. Springer-Verlag. ISBN 354043996X.

[70] Leto Peel, Daniel B. Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science Advances*, 3(5): e1602548, 2017. doi: 10.1126/sciadv.1602548. URL https://www.science.org/doi/abs/10.1126/sciadv.1602548.

[71] Clara Pizzuti and Annalisa Socievole. A genetic algorithm for community detection in attributed graphs. In Kevin Sim and Paul Kaufmann, editors, *Applications of Evolutionary Computation*, pages 159–170, Cham, 2018. Springer International Publishing. ISBN 978-3-319-77538-8.

[72] Clara Pizzuti and Annalisa Socievole. Multiobjective optimization and local merge for clustering attributed graphs. *IEEE Transactions on Cybernetics*, PP:1–13, 01 2019. doi: 10.1109/TCYB.2018.2889413.

[73] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10:191–218, 01 2006. doi: 10.7155/jgaa.00124.

[74] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663, 2004. doi: 10.1073/pnas.0400054101. URL https://www.pnas.org/doi/abs/10.1073/pnas.0400054101.

[75] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. doi: 10.1080/01621459.1971.10482356. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

[76] S. Redner. How popular is your paper? an empirical study of the citation distribution. *The European Physical Journal B*, 4(2):131–134, aug 1998. doi: 10.1007/s100510050359. URL https://doi.org/10.1007%2Fs100510050359.

[77] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, nov 2009. doi: 10.1140/epjst/e2010-01179-1. URL https://doi.org/10.1140%2Fepjst%2Fe2010-01179-1.

[78] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008. doi: 10.1073/pnas.0706851105. URL https://www.pnas.org/doi/abs/10.1073/pnas.0706851105.

[79] Annalisa Socievole and Clara Pizzuti. *Kernel-based Early Fusion of Structure and Attribute Information for Detecting Communities in Attributed Networks*, pages 141–151. 04 2023. ISBN 978-3-031-31182-6. doi: 10.1007/978-3-031-31183-3_12.

[80] Karsten Steinhaeuser and Nitesh Chawla. *Community Detection in a Large Real-World Social Network*, pages 168–175. 01 2008. ISBN 978-0-387-77671-2. doi: 10.1007/978-0-387-77672-9_19.

[81] Karsten Steinhaeuser and Nitesh V. Chawla. Identifying and evaluating community structure in complex networks. *Pattern Recognition Letters*, 31 (5):413–421, 2010. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2009.11.001. URL https://www.sciencedirect.com/science/article/pii/S0167865509003043.

[82] Alexander Strehl and Joydeep Ghosh. Cluster ensembles: A knowledge reuse framework for combining partitionings. In *Eighteenth National Conference on Artificial Intelligence*, pages 93–98, USA, 2002. American Association for Artificial Intelligence. ISBN 0262511290.

[83] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of the AAAI Workshop on AI for Web Search (AAAI 2000)*, pages 58–64, Austin, TX, USA, 2000.

[84] Patricia Sánchez, Emmanuel Müller, Uwe Korn, Klemens Böhm, Andrea Kappes, Tanja Hartmann, and Dorothea Wagner. *Efficient Algorithms for*

a Robust Modularity-Driven Clustering of Attributed Graphs, pages 100–108. 06 2015. ISBN 978-1-61197-401-0. doi: 10.1137/1.9781611974010.12.

[85] V. Traag, L. Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9:5233, 03 2019. doi: 10.1038/s41598-019-41695-z.

[86] Ana Rita Vieira, Pedro Campos, and Paula Brito. New contributions for the comparison of community detection algorithms in attributed networks. *Journal of Complex Networks*, 8(4), 12 2020. ISSN 2051-1329. doi: 10.1093/comnet/cnaa044. URL https://doi.org/10.1093/comnet/cnaa044.

[87] Joe H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, 1963. doi: 10.1080/01621459.1963.10500845. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845.

[88] M. Anthony Wong. A Graph Decomposition Technique Based on a High-Density Clustering Model on Graphs. Technical report, 1980. URL https://apps.dtic.mil/sti/citations/ADA090348.

[89] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. Gbagc: A general bayesian framework for attributed graph clustering. *ACM Trans. Knowl. Discov. Data*, 9(1), aug 2014. ISSN 1556-4681. doi: 10.1145/2629616. URL https://doi.org/10.1145/2629616.

[90] Zhao Yang, René Algesheimer, and Claudio J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6(1), aug 2016. doi: 10.1038/srep30750. URL https://doi.org/10.1038%2Fsrep30750.

[91] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. Linkrec: A unified framework for link recommendation with user attributes and graph structure. In *Proceedings of the 19th International Conference on World Wide Web*, pages 1211–1212. Association for Computing Machinery, 2010. ISBN 9781605587998. doi: 10.1145/1772690.1772879. URL https://doi.org/10.1145/1772690.1772879.

[92] Elena Zheleva, Hossam Sharara, and Lise Getoor. Co-evolution of social and affiliation networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 1007–1016, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557128. URL https://doi.org/10.1145/1557019.1557128.

[93] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.*, 2:718–729, 2009.

List of Figures

List of Tables

List of Abbreviations

- **LFR** Lancichinetti-Fortunato-Radicchi benchmark
- **NMI** Normalised Mutual Information
- **ARI** Adjusted Rand Index
- **NAS** Node Attribute Similarity algorithm
- **NPS** Node Path Similarity algorithm
- **MS** Mixed Similarity algorithm
- **Greedy** Greedy modularity algorithm
- **res** resolution (for Leiden, Louvain, Greedy and Eva algorithms)

A. Attachments

A.1 First Attachment

We attached files with the communities the algorithms detected and the files with the partitionings evaluated with NMI.

The files are in following folders according to the algorithms: eva, ms_infomap, ms_leiden, ms_louvain, nas_greedy, nas_infomap, nas_leiden, nas_louvain, nps_greedy, nps_infomap, nps_leiden, nps_louvain.

The names of the files either start with "eval" or "2eval" (which means the data are evaluated with NMI) or with "res" or "2res" (which means the files contain the detected communities). The number "2" before the filename means that the file contains other part of results or communities which was not created during the initial analysis.

Both types of files have the same format which can be easily read by the Python module "json". The file contain a dictionary. The key of the dictionary is a tuple of tuples. Each tuple corresponds to parameters of an algorithm. The first tuple corresponds to the benchmark parameters, the second tuple to the fusing algorithm parameters, the third tuple to the clustering algorithm parameters and the fourth to simultaneous approach algorithm parameters – ((graph), (fusing), (clustering), (simultaneous)). Each of the algorithm parameter tuples consists of tuples of specific parameters. The first value in the tuple is the parameter name, and the second value in the tuple is the parameter value. For example, (graph) tuple can be (("max_degree", 35), ("mu", 0.4)) and (clustering) tuple is for example (("resolution", 0.9)).

The value of the dictionary is a list of lists and the lists correspond to specific network (for each network, there is one list, so the number of lists equal to the number of generated networks for some benchmark parameter combination). The network results lists consist of results (the detected communities), so the length of such list is equal to the number of repeats of the algorithm.

The communities are stored in a list of length corresponding to number of nodes of the network. On each index in the list, there is the community id to which the corresponding node belongs.