DE GRUYTER OPEN

JAISCR, 2015, Vol. 5, No. 1, pp. 23 – 49
10.1515/jaiscr-2015-0017

ON MERGING AND DIVIDING SOCIAL GRAPHS

Pascal Held, Alexander Dockhorn and Rudolf Kruse

Department of Knowledge Processing and Language Engineering
Faculty of Computer Science, Otto von Guericke University Magdeburg
Universitaetsplatz 2, 39106 Magdeburg, Germany
e-mail: {pheld, kruse, dockhorn}@iws.cs.uni-magdeburg.de

Abstract

Modeling social interaction can be based on graphs. However most models lack the flexibility of including larger changes over time. The Barabási-Albert-model is a generative model which already offers mechanisms for adding nodes. We will extent this by presenting four methods for merging and five for dividing graphs based on the Barabási-Albert-model. Our algorithms were motivated by different real world scenarios and focus on preserving graph properties derived from these scenarios. With little alterations in the parameter estimation those algorithms can be used for other graph models as well. All algorithms were tested in multiple experiments using graphs based on the Barabási-Albert-model, an extended version of the Barabási-Albert-model by Holme and Kim, the Watts-Strogatz-model and the Erdős-Rényi-model. Furthermore we concluded that our algorithms are able to preserve different properties of graphs independently from the used model. To support the choice of algorithm, we created a guideline which highlights advantages and disadvantages of discussed methods and their possible use-cases.

1 Introduction

How can social interaction be modeled? This question is discussed in many fields such as psychology, sociology, and computer science. From an algorithmic point of view, graph-like structures influence our everyday life. Worldwide, people are interacting trough daily direct or technology-based communication. Investigating these structures uncovered patterns in the way people are connected to each other. A famous example is the phenomenon "six degrees of separation" which was first suggested by Frigyes Karinthy in 1929 and later popularized in a play written by John Guare in 1990âĂŹs. This phenomenon describes a global property of small-world networks such as the network people form through communication. It is suggested that every pair of humans is connected through a maximal chain of 6 steps. A proof for real networks of full-size scale could not be done, yet. Such a global property of a network could be used to predict the spreading of information or diseases.

However, while time is going on these structures are constantly changing. For instance some real-world communication examples, which are undergoing constant changes could be students making new friends, a group of classmates which is falling apart after graduation or two companies which are increasingly working together. Although these changes can be crucial for the lives of affected people, global properties should be largely unaffected.

These examples lead us to think about observed local changes. We can guess that some structures will still exist in the resulting networks. For example the group of classmates could lose track of each other and new groups will be forming. Extroverts will quickly find new friends, whereas introverts

could need some time to establish same amount of friends. Merging two cooperate networks will not be influenced by personal attitudes. For the scenario that company A is acquiring another company B it could be assumed that companies A's structure will undergo only little changes, while the latter could be fully restructured such that employees of company B will be integrated into company A's structure.

Those local effects are present throughout the whole network. While multiple connections will be lost, new connections will be established. The field of graph modeling offers us multiple models for the generation of such graphs like the Erdős-Rényi-model [1] and the Watts-Strogatz-model [2]. Additionally, the Barabási-Albert-model [3] is able to produce scale-free networks. Those are known for having few high connected nodes called hubs and much low connected ones. The latter model was already used to model the world wide web [4, 5] or movie co-occurrences between Hollywood actors [6]. Since social networks are claimed to be scale-free networks our initial analysis focuses on altering graphs following the Barabási-Albert-model.

The Barabási-Albert-model is build up by constantly adding nodes to an existing graph. Therefore an existing network can easily be expanded using the growing mechanism or shrinked by reverting previous expansions. Nevertheless this does not explain all observable features in real-world networks. For instance networks are able to split into separate networks when communication between two subgroups ceases e.g. a group of classmates which is falling apart after graduation or the well-known karate club data set presented by Zachary in [7]. The opposite effect, a merge of two previously distinct networks, is also plausible as described in the case of two joining enterprises. Since the general growing mechanism cannot be used to model those large changes, further algorithms need to be developed to meet all observable requirements.

The purpose of this paper is:

- to propose multiple algorithms for merging and dividing social network graphs based on networks generated using the Barabási-Albert-model. Algorithms will be focusing on different graph properties taking the characteristics of said real-world examples into account. See Section 3 for a full presentation of algorithms.
- to compare local and global influences of our merging and dividing approaches in Section 4
- to analyze the applicability of proposed algorithms to other graph models like the Erdős-Rényi-model and the Watts-Strogatz-model.

A short discussion of the results and ideas for future work will be presented in Section 7. Some of the results are already published in [8].

2 Related Work

Before we present our algorithms for merging and dividing graphs we shortly summarize graph related terms in the following sections. Afterwards we introduce typical graph models, which will be used to test proposed algorithms. We further highlight similarities and differences of used models in the final subsection.

2.1 Graphs

Our example of people staying connected to each other, as already being said, can be modeled as a graph. Here each person will be a node and two persons regularly communicating will be connected using an edge. If a person changes its communication behavior edges can be added or removed and therefore the graph can be altered over time. Persons joining in the group of people modeled in the graph can be added as additional nodes.

We will use the following graph notation for later sections. Let $G = (V,\ E)$ be a graph, with the set of nodes V and the set of edges E such that $E \subseteq \{(u,v) \mid u \neq v;\ u,v \in V\}$. For sake of simplicity we will assume that edges are always undirected and therefore the edges $e = (u,v)$ and $e' = (v,u)$ be the same. We will use index notations $V(g_i)$ and $E(g_i)$ to distinguish the nodes and edges of graphs g_i.

The number of links a node has is typically used to measure its connectivity. This is called the node degree and uses the notation $k_n = |\{e = (u,n) \mid e \in E;\ u \in V\}|$. Let $P(k)$ be the degree distribution of the network. While the individual node degree represents a local graph property, the degree distribu-

tion can be used as a global graph property. Note that for scale-free networks the degree distribution follows a power-law function.

Additionally a graph can consist of a set of connected components. A connected component is a maximal subgraph in which any two nodes are connected to each other by at least one path. Whereas two subgraphs are connected if a path from one to the other exists.

Merging of two formerly separated subgraphs $g1 = (V(g_1), E(g_1))$ and $g2 = (V(g_2), E(g_2))$ is defined by creating a new graph g such that $V(g) = V(g_1) \cup V(g_2)$, where $E(g)$ contains at least one edge $e = (u,v),\ u \in V(g_1),\ v \in V(g_2)$. Dividing a graph into two subgraphs works vice versa. The nodes of the divided graph will be distributed to the subgraphs g_1 and g_2 while holding the condition $V(g_1) \cap V(g_2) = \emptyset$.

Multiple generative graph models exist in the field of social networks. The following sections will introduce famous examples and takes a look at their properties. We will start with the Barabási-Albert-model since it forms the basis of our analysis.

2.2 Barabási-Albert-model

A global property of social networks is the scale-free property. It states that the node degree distribution follows nearly a power law function. Such a distribution could be observed in analyzing real world networks like the world wide web.

BarabÃąsi and Albert observed the scale-free property and searched for mechanisms explaining this resulting distribution. In their observations they found that new nodes favor the connection to well established nodes in the network. This mechanism is called preferential attachment and used for the generation of the model. Using the preferential attachment mechanism they were able to generate random scale-free networks. The procedure will be shortly explained in detail as follows [3].

The creation of the network starts with an initial set of m_0 nodes. Every new node will be connected to nodes in the graph using m edges, where $m \leq m_0$. The probability for a new node connecting to an existing node n is

$$p_n = \frac{k_n}{\sum_j k_j} \tag{1}$$

where k_n is the node-degree of node n, which is divided by the sum of all node-degrees. This results in the development of heavily linked nodes called hubs, which are linked to a great part of the graph. More generally the degree distribution of the full graph follows a power law function of the form

$$P(k) \sim k^{-\gamma}; \quad \gamma = 2.9 \pm 0.1 \quad . \tag{2}$$

By the definition of the preferential attachment strategy older nodes have higher chances to become hubs. In the case of $m = m_0$ we recommend to use a fully connected initial graph for the m_0 nodes. Otherwise the model will be biased to favor the $(m_0 + 1)$-node, because it has the maximal node-degree. We will make use of this throughout our proposed methods for merging and splitting Barabási-Albert-Graphs.

A drawback of the Barabási-Albert-model is that it is unlikely to result in multiple components. This can happen when the initial set of nodes consists of multiple connected components and further iterations do not connect those. If we choose to start with a complete graph, it will always result in one connected component. An extension of the original generative algorithm tries to increase the clustering capabilities. Holme and Kim proposed to add a fourth step to the generation process[9]. Adding a node with m edges will be done by choosing the first edge per preferential attachment. In variation to the standard process further edges can also be added with the alternative triad formation step. Here, a new edge is added such that the new node, the node from the first preferential attachment step and a third node form a triad. With probability p we chose between using a triad formation step instead of preferential attachment step. The authors conclude their alteration increases clustering while maintaining the power-law distribution of node-degrees. We will compare our algorithms for their use on pure Barabási-Albert-graphs and the alteration of Holme and Kim later referred as Extended-Barabási-Albert-graphs in in subsubsection 6.1.3.

The Barabási-Albert-model was a first attempt to explain the existence of node degree distributions following a power law function and the emergence of hubs. However, another drawback was that the model could not explain how new nodes could become hubs very fast. For instance, relating to the

world wide example, the rise of Google as one of the most linked web pages of the world wide web. The model was extended using a fitness model described in [10] to explain such a behavior. However such extensions will not be regarded in this paper.

2.3 Watts-Strogatz-model and Erdős-Rényi-model

Since we will compare our algorithms for the application on the Watts-Strogatz-model and the Erdős-Rényi-model, we will shortly summarize both below.

The Watts-Strogatz-model [2] constructs a random graph $G(n,k,\beta)$ by starting with N nodes each connected to K neighbors. As a second step all edges are rewired with probability β. This is done by replacing an edge (n_i,n_j) by (n_i,n_k), where $k \neq i$ and the edge does not already exist. Note that it is possible that the resulting graph consists of multiple connected components.

The Erdős-Rényi-model [1] is divided into two closely related variants. The first chooses one of all possible graphs $G(n,M)$ with M edges and n nodes, where each graph has an equal probability. This could be done by choosing N edges from the $\binom{n}{2}$ possible edges. Second variant $G(n,p)$ starts with an initial set of n unconnected nodes and includes edges with probability p [11]. It can easily be deduced that each graph with n nodes and M edges is equally likely with probability

$$p^M(1-p)^{\binom{n}{2}-M} \quad . \tag{3}$$

As in the Watts-Strogatz-model it is possible that the graph created by both variants consists of multiple connected components.

2.4 Comparison of the models

A limitation of the Watts-Strogatz-model and the Erdős-Rényi-model is that they are not able to produce a node degree distribution following a power law function. Also both do not provide a growing mechanism, which fixes them to the initial set of nodes.

Figure 1 shows examples of the four discussed graph models. The first one was created using the Erdős-Rényi-model. We can see that the graph consists of two connected components. The second graph is based on the Watts-Strogatz-model. The rewiring probability β was set to 0.3. Two edges were rewired in the generation process. Even if this graph shows one connected component it is also possible that two components would have developed. Both graphs to the right are based on the Barabási-Albert-model and its extension. Typical for the both versions is that it is always one connected component and some nodes have a much higher node-degree. The latter shows a much higher degree of clustering, which can be seen by the high amount of connected triads.

3 Altering Barabási-Albert-Graphs

We already discussed the process for generating a Barabási-Albert-Graph. Altering this graph by growing or shrinking the network can be implemented using the preferential attachment mechanism or reversing it. The Barabási-Albert-model itself is based on the approach of adding one node at a time and connecting this to other nodes. Growing the graph can be based on further iterations of this generation process.

Reversing last iteration steps results in shrinking the graph. If the order of adding nodes is unknown we need to estimate the order in which nodes were added to the graph. An implication of the preferential attachment mechanism is that older nodes probably have a higher node degree. For that reason we can guess that sorting the nodes increasingly by their node degree gives us a proper estimate for their amount of time being part of the graph. In a noise free network at least one node should have exactly m edges, which is a good candidate to be removed. Otherwise we simply remove the node with the smallest node-degree. However it cannot be assured that the model still holds true for the resulting graph.

We will use the following subsection to describe more complex algorithms for merging and dividing Barabási-Albert-Graphs. For some algorithms we will need to estimate the parameters a Barabási-Albert-Graph is based on. Therefor we will first explain how those can be estimated.

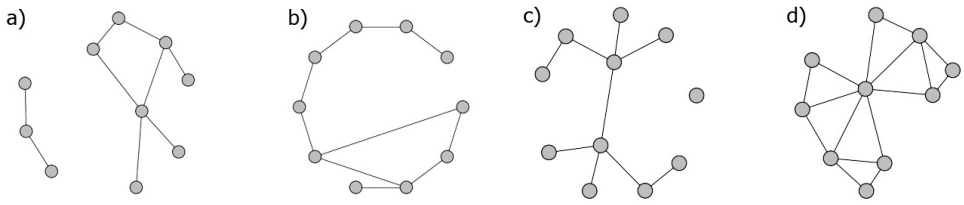

Figure 1. Comparison of graph models, a) Erdős-Rényi-model generated using the second variant $G(n,p)$ with $p = 0.25$, b) Watts-Strogatz-model with 0.3 probability for rewiring, c) Barabási-Albert-model with $m_0 = m = 1$, d) extended Barabási-Albert-model by Holme and Kai with $m_0 = m = 2,$ and $p = 0.5$

3.1 Estimating Barabási-Albert-Model Attributes

An existing Barabási-Albert-graph is based on the amount of initial nodes m and the number of edges each node gets connected to previous nodes m_0. For an existing Barabási-Albert-graph we need to estimate the attributes of the generation process. These are the amount of initial nodes m_0 and the number of edges per new node m.

Let n_t be the total number of nodes present in the graph and $n_a = n_t - m_0$ the number of nodes added in the growing phase. Similar we define e_t as the total number of edges in the graph, which is the sum of edges from the initial phase e_0 and the edges from the growing phase $e_a = n_a \cdot m$.

Depending on the assumption of the initialization e_0 is between 0 (starting with no edges at all) and $0.5 \cdot m_0 \cdot (m_0 - 1)$ (full-connected graph).

Based on this, we get:

$$n_a \cdot m \leq e_t \leq n_a \cdot m + \frac{m_0 \cdot (m_0 - 1)}{2} \quad (4)$$

$$\frac{e_t}{n_a} \geq m \geq \frac{e_t}{n_a} - \frac{m_0 \cdot (m_0 - 1)}{2n_a} \quad (5)$$

$$\frac{e_t}{n_t - m_0} \geq m \geq \frac{e_t}{n_t - m_0} - \frac{m_0 \cdot (m_0 - 1)}{2(n_t - m_0)} \quad (6)$$

Previously we assumed that the graph after the initialization phase is fully connected. For the case that $m = m_0$ and we are always starting with a full-connected graph of m nodes the equation can be reduced to:

$$e_t = (n_t - m) \cdot m + \frac{m \cdot (m-1)}{2} \quad (7)$$

$$0 = m^2 - 2 \cdot (n_t - \frac{1}{2}) \cdot m + 2 \cdot e_t \quad (8)$$

$$m_{1,2} = n_t - \frac{1}{2} \pm \sqrt{(n_t - \frac{1}{2})^2 - 2 \cdot e_t} \quad (9)$$

Experiments showed that subtracting the value of the square root in Equation 9 results in a correct estimation of m.

Furthermore we will need an estimation of the parameter m for a merge-graph g of two Barabási-Albert-Graphs g_1 and g_2 and the reverse operation of dividing graph g into two Barabási-Albert-Graphs g_1 and g_2. The latter case can be solved trivially by setting m_1 and m_2 equal to the estimates m of the divided graph. Deciding about an estimate for the value m of a merge-graph can be more complicated, except the trivial case of both graphs having an equal parameter m such that $m_1 = m_2$. In this case we can estimate m_1 and m_2 for both subgraphs separately and return $m = m_1 = m_2$. Otherwise we will have to find a value for m big enough to reach at least the same number of edges in the merge graph as the sum of edges in both subgraphs ($|E(g)| \geq |E(g_1)| + |E(g_2)|$). The estimation of the parameter m of a merge graph is described in the following algorithm.

```
1:  function ESTIMATEM(Graph: g1, g2)
2:      m1 ← estimateM(g1)
3:      m2 ← estimateM(g2)
4:      nt ← |V(g1)| + |V(g2)|
5:      m ← min(m1, m2)
6:      loop
7:          e ← (nt − m) · m + m·(m−1)/2
8:          if e ≥ |E(g1)| + |E(g2)| then
9:              return m
10:         else
11:             m ← m + 1
12:         end if
13:     end loop
14: end function
```

3.2 Merging of two Graphs

In the following we describe four strategies and their underlying motivation to merge two graphs.

3.2.1 Random Merge

To provide a baseline for later experiments we will use the random merge algorithm (RM). It is the simplest way of merging two graphs by picking up all their nodes and add them in random order to a new Barabási-Albert-Graph.

```
1: function RANDOMMERGE(Graph: g1, g2)
2:     m ← estimateM(g1, g2)
3:     graph ← EmptyBarabasiGraph(m)
4:     nodes ← V(g1) ∪ V(g2)
5:     nodes.shuffle()
6:     for all node ← nodes do
7:         graph.addNode(node)
8:     end for
9:     return graph
10: end function
```

The complexity of this algorithm is (n), where n is the number of nodes in both graphs.

While being the simplest way we could not expect the resulting graph to preserve properties of the input graphs. The only thing, which could be expected is, that the Barabási-Albert-model properties hold in the resulting graph.

3.2.2 Node-Degree-Order Merge

The second algorithm we propose focuses on preserving the node degree of every node. One exemplary use-case would be the differentiation of extroverts and introverts. Consider two groups from different schools merged by getting to know each other. Individual connectivity of a node should be preserved after the merge such that nodes representing extroverts will have more connections than introverts before and after the merge.

The preferential attachment strategy used in the Barabási-Albert-model leads to early added nodes having much more connections than later ones. Therefore we take all nodes of both graphs and put them in a combined list. Nodes will be added to a new Barabási-Albert-Graph in decreasing order of their node degree. With this strategy it is expected, that the node-degree distribution relating to the specific nodes is the same.

```
1: function NODEDEGREEMERGE(Graph:
   g1, g2)
2:     m ← estimateM(g1, g2)
3:     graph ← EmptyBarabasiGraph(m)
4:     nodes ← V(g1) ∪ V(g2)
5:     nodes.sortByNodeDegree('desc')
6:     for all node ← nodes do
7:         graph.addNode(node)
8:     end for
9:     return graph
10: end function
```

The complexity of this algorithm is $(n \cdot \log n)$, where n is the number of nodes in both graphs. The main complexity is a result of the sorting operation on almost presorted lists.

3.2.3 Preserving-Nodes Merge

Relating to our company example, where one enterprise acquired the other, we need a merge strategy which preserves the structure of one child graph as much as possible. Our observations resulted in the definition of the preserving-nodes merge (PNM). The main idea is to keep the full structure of one input graph and add the other node by node to the resulting merge-graph. Based on the node-degree-order merge, nodes from the second input graph are inserted in decreasing order of their node degree.

```
1: function PRESERVINGNODESMERGE(
   Graph: g1, g2)
2:     m ← estimateM(g1, g2)
3:     nodes ← V(g2)
4:     nodes.sortByNodeDegree('desc')
5:     for all node ← nodes do
6:         g1.addNode(node)
7:     end for
8:     return graph
9: end function
```

The complexity of this algorithm is $(n \cdot \log n)$, where n is the number of nodes in the second graph

From construction this strategy keeps all the information of the first graph. The inner structure of the second graph is lost, but the node-degree distribution relating to the specific nodes of the second graph is the same.

Since the first graph is used as the base for the merge, throughout the preferential attachment it is highly probable that new nodes will preferentially

be connected to nodes of the first graph. This leads to a domination of the nodes from the first graph and will increase their node degree much more as nodes from the second graph.

Based on our real-world example of two enterprises, in which one acquires the other, this could model rising connection between managers of the acquiring company and workers of the second company. The structure of the buying company will not change significant. Only few edges to nodes of the bought company will be added. Managers of the acquired company will be integrated first in the new company structure and therefor are more likely to be placed in higher positions with more influence and connections in the new merges company. Finally workers of the bought company will be integrated in the daily workflow.

3.2.4 Minimal-Merge

For our last merge strategy we again go back to the analysis of two real world examples. For instance we could model two groups of friends rarely having contact to people outside the group or two enterprises increasing their communication to each other. Both examples will experience nearly no changes in the base graphs when merged.

The minimal merge (MM) tries to model such an behavior and focuses on keeping most of the structure of both graphs. The main idea is to use both graphs and connect them with additional edges. To do so, we increase the estimated m. Now we have free edges, we can use to connect both graphs. Similar to the basic Barabási-Albert-approach we select nodes proportional to there node-degree.

```
1: function MINIMALMERGE(Graph: g1, g2)
2:     m ← estimateM(g1, g2) + 1
3:     g ← UnionBarabasiGraph(g1, g2, m)
4:     e_add = g.getMaxEdges() − |E(g)|
5:     while e_add > 0 do
6:         n1 ← V(g1).preferedSelect()
7:         n2 ← V(g2).preferedSelect()
8:         g.addEdge(n1, n2)
9:         e_add ← e_add − 1
10:    end while
11:    return graph
12: end function
```

The complexity of this algorithm is (n). This strategy keeps most of the structure, with the drawback of increasing the number of edges per node.

3.3 Dividing into two Graphs

After giving some examples for merging graphs, we want to add five strategies for dividing a graph in two subgraphs. Each algorithm will use the parameters graph (g) and the number of nodes expected in the first subgraph ($noNodes_1$).

3.3.1 Random-Divide

As in the case of merging two graphs we provide one algorithm as baseline for comparison in our evaluation experiments. The random-divide strategy (RD) is the simplest idea to divide a given graph. The basic idea is to create two sets of nodes, for each new graph one. Then create a new Barabási-Albert-graph from both of these sets.

```
1: function RANDOMDIVIDE(
   Graph: g, noNodes1)
2:     v1 ← V(g).randomSelect(noNodes1)
3:     v2 ← V(g) − v1
4:     m ← estimateM(g)
5:     g1 ← EmptyBarabasiGraph(m)
6:     g2 ← EmptyBarabasiGraph(m)
7:     for all node ← v1 do
8:         g1.addNode(node)
9:     end for
10:    for all node ← v2 do
11:        g2.addNode(node)
12:    end for
13:    return g1, g2
14: end function
```

The complexity of the random-divide strategy is (n), where n is the number of nodes. This simplest method does not care about the underlying structure of the graph, but it ensures the properties of a Barabási-Albert-Graph in the resulting graphs.

3.3.2 Random-Subgraph-Divide

An alteration of the Random-Divide lead us to the second strategy called Random-Subgraph-Divide (RSD). The set of nodes is randomly split into two subsets, which will be used to create two child graphs. Connections within those subgraphs will be preserved. However this will lead to graphs

violating the Barabási-Albert-properties. Therefor we have to include repairing steps, which will be described in detail in Subsection 3.4.

function RANDOMSGDIVIDE(Graph: g, $noNodes_1$)
  $v_1 \leftarrow V(g).randomSelect(noNodes_1)$
  $v_2 \leftarrow V(g) - v_1$
  $e_1 = \{(x_1,x_2) : \forall (x_1,x_2) \in E(g) \wedge x_1,x_2 \in v_1\}$
  $e_2 = \{(x_1,x_2) : \forall (x_1,x_2) \in E(g) \wedge x_1,x_2 \in v_2\}$
  $g_1 \leftarrow Graph(v_1,e_1)$
  $g_2 \leftarrow Graph(v_2,e_2)$
  $m_1 \leftarrow estimateM(g_1)$
  $m_2 \leftarrow estimateM(g_2)$
  $repairGraph(g_1,m_1)$
  $repairGraph(g_2,m_2)$
  return g_1,g_2
end function

The complexity of our second divide algorithm is (n^2), where n is the number of nodes. The increase in complexity results from our repairing mechanism.

3.3.3 Subgraph-Expansion-Divide (SED)

We already preserved some connections between nodes in the RSD. The main drawback of this method seems to be the extensive use of the repairing operation to reconnect each of the subgraphs to one connected component. To improve the Random-Subgraph-Divide we chose a node set, which forms one connected component. We achieve this by iteratively adding nodes to the current subgraph until the size of first graph is reached. For simplicity this step was implemented using a breadth-first search (BFS). Maximum-cardinality search could be used as an alternative search scheme. See [12] for a detailed description. We assume that the compactness of the resulting subgraph should be higher, but experiments still have to be performed.

However it cannot be guaranteed that both graphs approximate properties of the Barabási-Albert-model e.g. the second subgraph can be split into several components. In addition, both graphs could still lack some edges, so both need to be repaired using the repair-operator described in Subsection 3.4.

function SUBGRAPHEXPDIVIDE(Graph: g, $noNodes_1$)
  $startNode \leftarrow g.NodeWithLowestDegree()$
  $v_1 \leftarrow BFS(startNode, noNodes_1)$
  $v_2 \leftarrow V(g) - v_1$
  $e_1 = \{(x_1,x_2) : \forall (x_1,x_2) \in E(g) \wedge x_1,x_2 \in v_1\}$
  $e_2 = \{(x_1,x_2) : \forall (x_1,x_2) \in E(g) \wedge x_1,x_2 \in v_2\}$
  $g_1 \leftarrow Graph(v_1,e_1)$
  $g_2 \leftarrow Graph(v_2,e_2)$
  $m \leftarrow estimateM(g)$
  $repairGraph(g_1,m)$
  $repairGraph(g_2,m)$
  return g_1,g_2
end function

The complexity of our last divide is (n), where n is the number of nodes in the graph. The repairing step increases the complexity to (n^2).

3.3.4 Node-Degree-Divide A

As it was already modeled in the Node-Degree-Merge, we differentiated nodes by their node degree. The Node-Degree-Divide A (NDDa) is based on the same principle. First we order all nodes descending by their node-degree. Second we split this list and use the first part for the first subgraph and the second accordingly. Nodes are then added in order of their node degree. Edges between the nodes will not be preserved.

function NODEDEGREEDIVIDEA(Graph: g, $noNodes_1$)
  $m \leftarrow estimateM(g)$
  $g_1 \leftarrow EmptyBarabasiGraph(m)$
  $g_2 \leftarrow EmptyBarabasiGraph(m)$
  $nodes \leftarrow V(g)$
  $nodes.sortByNodeDegree('desc')$
  for all $node \leftarrow nodes[:noNodes]$ **do**
    $g_1.addNode(node)$
  end for
  for all $node \leftarrow nodes[noNodes:]$ **do**
    $g_2.addNode(node)$
  end for
  return g_1,g_2
end function

The complexity of this algorithm is $(n \cdot \log n)$, where n is the number of nodes in the graph. Since this strategy keeps the Barabási-Albert-properties

and also the node-degree distribution related to the nodes, no repairing operation is needed.

3.3.5 Node-Degree-Divide B (NDDb)

This strategy is closely related to NDDa and modifies the splitting strategy of the ordered list of nodes. A drawback of the first strategy is that the node degree distribution following a power law function results in two distinct distributed subsets. While the first includes nodes with a wide range of node degrees, the second subset could be nearly evenly distributed.

For that reason we do not split the list into two parts, but instead we pick alternating elements for every subgraph, with respect to the selected relation. The function *createIndexList*() returns all node indexes of nodes used for the first subgraph such that when possible they are equally spaced to each other.

This alternative picking strategy ensures that each subgraph includes some of the top connected nodes, while overall fairly distributing nodes of the originating node-degree distribution between both graphs. However, equal to the NDDa algorithm, edges between picked nodes will not be preserved.

```
1: function NODEDEGREEDIVIDEB(Graph: g, noNodes_1)
2:     m ← estimateM(g)
3:     g_1 ← EmptyBarabasiGraph(m)
4:     g_2 ← EmptyBarabasiGraph(m)
5:     nodes.sortByNodeDegree('desc')
6:     indexlist ← createIndexList(|V(g)|, noNodes_1)
7:     for all node ← nodes do
8:         if index(node) in indexlist then
9:             g_1.addNode(node)
10:        else
11:            g_2.addNode(node)
12:        end if
13:    end for
14:    return g_1, g_2
15: end function
```

The complexity of strategy NDDb is $(n \cdot \log(n))$ where n is the number of nodes. The increased effort for picking nodes does not influence the complexity class and therefor is equal to NDDa.

3.4 Repairing-Steps

Two of our divide algorithms, namely Random-Subgraph-Divide and Subgraph-Expansion-Divide, can produce graphs that do not hold typical properties of the Barabási-Albert-model. We created a repairing-operation which tries to achieve the three properties:

1 Barabási-Albert-graph is always one connected component

2 Every node has at least m edges, respectively each node $n \in V$ has a node-degree of $k_n \geq m$

3 The maximal number of edges is

$$n_a \cdot m + \frac{m \cdot (m_0 - 1)}{2} \tag{10}$$

This does not lead to a graph perfectly following the Barabási-Albert-model, but achieves the most properties with minimal manipulation of the graph. If the graph consists of multiple connected components, additional edges will be used to connect all to one component. Further on edges will be added to increase the node-degree of all nodes having a degree lower than m. For the case that the number of edges is not high enough an recursive run with $repairGraph(g, m+1)$ will be started. If first two steps did not use up all edges, remaining edges will be added using preferential attachment.

```
1: function REPAIRGRAPH(Graph: g, m)
2:     if |V(g)| ≤ m then
3:         return completeGraph(V(g))
4:     end if
5:     e_add = getMaxEdges(g) − |E(g)|
6:     while |g.components| > 1 do
7:         if e_add < 0 then
8:             return repairGraph(g, m + 1)
9:         end if
10:        source ← V(g).randomSelect(1)
11:        target ← V(g).randomSelect(1)
12:        Connect(source, target)
13:        e_add ← e_add − 1
14:    end while
15:    for all node ← {n : n ∈ V(g), k_n < m} do
16:        while node.degree < m do
17:            if e_add < 0 then
18:                return repairGraph(g, m + 1)
19:            end if
20:            target ← V(g).preferedSelect(1)
```

```
            g.addEdge(node, target)
            e_add ← e_add − 1
        end while
    end for
    while e_add > 0 do
        source ← V(g).preferedSelect(1)
        target ← V(g).preferedSelect(1)
        g.addEdge(source, target)
        e_add ← e_add − 1
    end while
    return g
end function
```

The repair operator has a complexity of (n^2) where n is the number of nodes.

The repairing process is based on the generative mechanisms of the original model. On the one hand it could be argued that a better approximation of the power-law function could be achieved by calculating the desired distribution beforehand and use the remaining edges to fit this distribution. One the other hand we want the power-law distribution to be a result of the preferential attachment strategy and not created by force. Therefor we focused on a minimal change of the start graph using the preferential attachment strategy to achieve properties listed above.

4 Experiments

The following subsections will describe our experiments for evaluating the behavior of proposed merge and divide algorithms for Barabási-Albert-Graphs. Subsection 4.3 introduces our comparison measures. Additionally we recorded run times of all algorithms. See Subsection 5 for results.

4.1 Merging of two Graphs

We used Barabási-Albert-Graphs of differing size and (n) and connectivity $(m_0 = m)$. Our four scenarios test the algorithm behavior for a merge of two graphs based on equal parameters and differences in one or both parameters. The following test scenarios formed the basis for our comparison of Barabási-Albert-graphs:

name	n_1	m_1	n_2	m_2
equal	5000	3	5000	3
diff_m	5000	3	5000	8
diff_size	5000	8	25000	8
diff_all	5000	8	25000	3

4.2 Dividing into two Graphs

Similar to the experiments for merging graphs we divided graphs in different ratios of nodes in the resulting subgraphs. Following test scenarios were created for dividing a Barabási-Albert-graph into two subgraphs:

name	n	m	$noNodes1$
10 : 90	10000	5	1000
20 : 80	10000	5	2000
30 : 70	10000	5	3000
40 : 60	10000	5	4000
50 : 50	10000	5	5000

4.3 Measurements

To measure the quality of each algorithm, we focus on three aspects derived from real world observations, namely edge preservation, node-degree rank and node-degree distribution.

4.3.1 Edge Preservation

One of our merging examples was the increased cooperation of two enterprises. In this case the inner structure of both graphs is nearly untouched. This means, that nodes which are connected before are also connected after the merge. Not connected nodes should be separate after the merge, as well.

We can measure such a behavior by calculating the percentage of preserved edges during the alteration process. Because of different usage of graph information during the merge and divide operations, e.g. PNM only preserves edges of the first graph, this has to be calculated for both graphs individually.

4.3.2 Rank-Correlation

Another property already discussed is the differentiation of extroverts and introverts. Nodes

could possibly establish a wide range of connections. We suggest that nodes which are more active, or stronger connected should also have a higher node-degrees in the resulting graphs.

To estimate this, we calculate the node-degree rank of every node in both graphs. Afterwards, we calculate the two well known rank correlation coefficients Spearman's ρ [13, Section 14.7] including tie-correction and Kendall's τ [14]. These measures have a range from -1 to $+1$, where $+1$ (-1) indicates that the order is completely preserved (reversed).

4.3.3 Node-Degree Distribution

Based on the Barabási-Albert-model the node-degree of all nodes in the graph should follow a power-law distribution, described in Equation 2.

We determine the node-degree distribution of the resulting graph and compare them with the theoretical node-degree distribution based on the presented formula and the number of edges in the graph. We calculate the root-mean-squared-error to check how good the distribution fits the node-degree distribution of our graph.

5 Results

In the following section we present the results of our experiments. First we give a brief comparison about the run times, and afterwards we show the in Subsection 4.3 described measures.

5.1 Merging of two Graphs

5.1.1 Computing Time

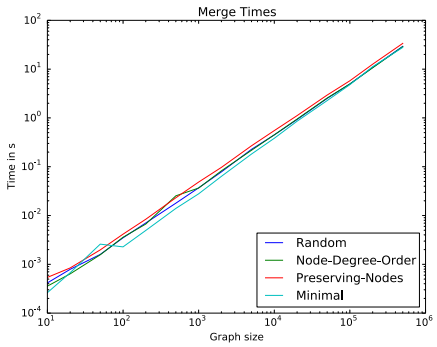

Figure 2. Computation time for merge operators with different graph sizes

In Figure 2 we show the run time for the presented merge algorithms. We used two graphs with the same number of nodes and a connectivity of $m = 4$. The measurement is based on 10 runs for each graph size. The runtime for all algorithms is almost linear in graph-size.

This is due to the fact that the constants of the logarithmic parts from the ordering are relative small in contrast to the constant factors of the merge process itself.

5.1.2 Edge Preservation

The edge preservation follows the design of the algorithms. The Minimal-Merge preserves all edges, and the Preserving-Nodes Algorithms the edges from the first graph. All others algorithms lose the inner structure. A more detailed view can be taken from Table 1. The first part of every column describes the percentage of preserved edges from the first resulting subgraph, and the second part the other subgraph respectively.

5.1.3 Rank Correlation

Recorded values were averaged over 100 iterations of specified experiments and are shown in Table 2 and Table 3. The baseline algorithm Random-Merge (RM) had always values around 0 which indicates that degree rank order before and after the merge stand in no correlation to each other. Preserving-Node-Merge (PNM) performed better in the merge of two equal graphs and on the same level as RM for graphs with differing m. The Node-Degree-Merge algorithm (NDM) ranked second best in first three experiments. For an equal merge both rank-correlation coefficients had much higher values ($\rho_{equal} = 0.726$ and $\tau_{equal} = 0.625$) than PNM. It reached even higher values for merging graphs with differing m ($\rho_{diff\_m} = 0.842$ and $\tau_{diff\_m} = 0.709$) and differing size ($\rho_{diff\_m} = 0.874$ and $\tau_{diff\_m} = 0.779$). However PNM performed better than NDM for graphs differing in size and connectivity ($\rho_{diff\_all} = 0.850$ and $\tau_{diff\_all} = 0.718$). The algorithm Minimal-Merge (MM) scored best with values near to +1 for both experiments. This is due to the minimal change by adding just a few edges.

Table 1. Average part of edges preserved after merge-operation in percent

Merge	equal	diff_m	diff_size	diff_all
RM	0.1 / 0.1	0.1 / 0.1	0.1 / 0.1	0.0 / 0.0
NDM	0.4 / 0.4	0.3 / 1.1	0.3 / 0.4	0.5 / 0.1
PNM	100 / 0.0	100 / 0.0	100 / 0.0	100 / 0.0
MM	100 / 100	100 / 100	100 / 100	100 / 100

Table 2. Measured values for Kendall's τ

Merge	ρ_{equal}	ρ_{diff_m}	ρ_{diff_size}	ρ_{diff_all}
RM	0.000	-0.001	0.000	0.001
NDM	0.726	0.842	0.874	0.779
PNM	0.475	-0.134	0.612	0.850
MM	1.000	0.979	1.000	0.985

Table 3. Measured values for Spearman's ρ

Merge	τ_{equal}	τ_{diff_m}	τ_{diff_size}	τ_{diff_all}
RM	0.000	0.000	0.000	0.001
NDM	0.625	0.709	0.745	0.661
PNM	0.406	-0.061	0.486	0.718
MM	1.000	0.927	1.000	0.964

5.1.4 Node-Degree Distribution

Table 4 shows the RMSE between the observed node-degree and the theoretical node-degree distribution. The upper part of the table shows the RMSE for the initial graphs, and below the results for all merged graphs for every method.

Table 4. RMSE of node-degree distribution before and after merge

RMSE	equal	diff_m	diff_size	diff_all
g_1	23.2	22.9	4.9	4.8
g_2	22.6	4.9	12.3	75.7
RM	38.2	11.7	14.0	53.9
NDM	39.4	12.0	14.3	54.2
PNM	38.3	38.2	14.1	14.3
MM	43.3	199.1	14.8	229.2

The Random-Merge (RM) algorithm generates the best results except for the diff_all dataset. Minimal-Merge (MM) leads to huge errors, especially with different m. This is based on the fact, that most of the inner structure is kept and no combined graph with all nodes is created from scratch. PNM and NDM generate results with RMSE between the initial graphs.

5.2 Dividing into two Graphs

5.2.1 Computing Time

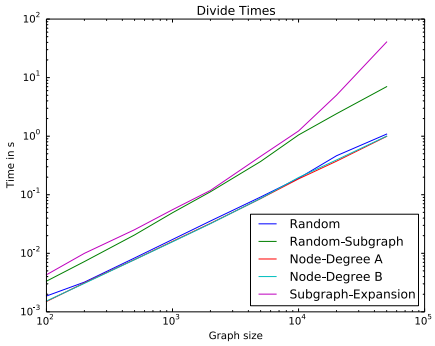

Figure 3. Computation time for divide operators with different graph sizes

Figure 3 shows the run time for the presented divide algorithms. We used a graph with a connectivity of $m = 4$. The divide operation divides the graph into two subgraphs with the same size. The measurement is based on 10 runs for each graph size.

The two algorithms using the repair operator are slower then the other ones. The Random-Divide (RM), and the Node-Degree-Divide (NDDa, and NDDb) algorithms are almost equal and linear in computation time. The Random-Subgraph-Divide (RSD) algorithms looks also linear, which is an indication that the repair operation is less used in this algorithm than in the Subgraph-Expansion-Divide (SED) algorithm.

Table 5. Average part of edges preserved after divide-operation in percent

Divide	10:90	20:80	30:70	40:60	50:50
RD	1.0 / 0.1	0.5 / 0.1	0.3 / 0.1	0.2 / 0.2	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	3.9 / 0.2	2.3 / 0.2	1.6 / 0.2	1.3 / 0.3	1.1 / 0.3
NDDb	4.0 / 0.7	2.5 / 0.8	1.8 / 0.9	1.4 / 1.0	1.1 / 1.2
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100

5.2.2 Edge Preservation

The RSD and the SED preserve all structure information from the selected subgraph. The NDD algorithms preserve some structure, $(2-4\%)$, while the Random Divide loses almost all inner structure. Detailed information is presented in Table 5. The first part of each column represents the portion of preserved edges from the first resulting subgraph, and the second portions represents the ratio of the other subgraph.

5.2.3 Rank Correlation

Measured rank-correlations of all divides are shown in Table 6 and Table 7. Recorded values were averaged over 100 iterations of specified experiments. It can be seen that our baseline algorithm Random-Divide (RD) shows no correlation between the degree rank order before and after the divide. So the order of nodes by their node degree before the divide has no effect on the order of nodes after the divide. The algorithms Node-Degree-Divide-A (NDDa) and Subgraph-Expansion-Divide (SED) scored nearly similar for uneven divides (see $\rho_{10:90}$ and $\tau_{10:90}$). However SED seems to be more resistant to a change of the node-ratio. The rank-correlation values of SED are slowly declining from $\rho_{10:90} = 0.675$ to $\rho_{50:50} = 0.516$ (decrease of $\approx$23%). Whereas rank-correlation values of NDDa were decreasing from $\rho_{10:90} = 0.661$ to $\rho_{50:50} = 0.385$ (decrease of $\approx$42%). Similar observations can be done comparing values for Kendall's τ. Random-Subgraph-Divide (RSD) was evaluated as second best algorithm. Both correlation coefficients have a high range of values, where distributing nodes in a ratio of 30:70 resulted in minimal values of $\rho_{30:70} = 0.632$ and $\tau_{30:70} = 0.520$. Higher values were reached for more equal divides with a ratio of 50:50 ($\rho_{50:50} = 0.722$ and $\tau_{50:50} = 0.612$) or more uneven divides with a ratio of 10:90 ($\rho_{10:90} = 0.771$ and $\tau_{10:90} = 0.709$). The Node-Degree-Divide-B (NDDb) algorithm scored best with nearly constant values of $\rho \approx 0.813$ and $\tau \approx 0.692$.

Table 6. Measured values for Spearman's ρ

Divide	$\rho_{10:90}$	$\rho_{20:80}$	$\rho_{30:70}$	$\rho_{40:60}$	$\rho_{50:50}$
RD	0.000	0.001	0.001	0.000	0.000
RSD	0.771	0.650	0.632	0.683	0.722
NDDa	0.661	0.541	0.456	0.401	0.385
NDDb	0.813	0.814	0.813	0.814	0.814
SED	0.675	0.568	0.525	0.511	0.516

Table 7. Measured values for Kendall's τ

Divide	$\tau_{10:90}$	$\tau_{20:80}$	$\tau_{30:70}$	$\tau_{40:60}$	$\tau_{50:50}$
RD	0.000	0.000	-0.001	0.000	0.002
RSD	0.709	0.558	0.520	0.569	0.612
NDDa	0.561	0.456	0.379	0.330	0.314
NDDb	0.692	0.693	0.692	0.693	0.693
SED	0.558	0.449	0.407	0.394	0.402

5.2.4 Node-Degree Distribution

The measured root-mean-squared-error for all dividing algorithms is presented in Table 8. The Random-Divide Algorithm (RD) as well as the Node-Degree-Divide algorithms (NDDa, and NDDb) generate subgraphs with a lower RMSE then the initial graph. On the contrary the Random-Subgraph-Divide (RSD) algorithm and the Subgraph-Expansion-Divide (SED) method leads to subgraphs with much higher RMSE. Both algorithms use much of the underlying structure and the repair method. That is the reason why the joined

Table 8. RMSE of node-degree distribution before and after divide

RMSE	10:90	20:80	30:70	40:60	50:50
g	16.3	16.4	16.2	16.3	16.5
RD(g_1)	3.8	5.7	7.2	8.9	10.2
RD(g_2)	15.3	14.1	12.8	11.6	10.1
RSD(g_1)	91.8	128.8	190.9	197.0	175.0
RSD(g_2)	309.6	310.8	283.2	234.2	170.6
NDDa(g_1)	3.9	5.7	8.8	8.8	9.8
NDDa(g_2)	15.4	14.1	11.8	11.8	10.1
NDDb(g_1)	3.9	5.5	8.7	8.7	10.4
NDDb(g_2)	15.3	14.2	11.5	11.5	10.1
SED(g_1)	32.0	60.8	93.6	127.9	167.5
SED(g_2)	442.6	464.0	444.4	412.4	374.9

node-degree distribution does not fit the theoretical node-degree distribution provided by the Barabási-Albert-model.

6 Altering other types of graphs

In this section we focus on graphs generated by other models. A brief introduction into these models could be find in Section 2. We performed the same experiments as for the Barabási-Albert-graphs by generating similar graphs.

In the following we present the experiments and the results of Erdős-Rényi, Watts-Strogatz, and Extended-Barabási-Albert graphs.

6.1 Experiments

In our case similar means, that the graphs have similar number of nodes and edges. For simplicity we assume $n_{edges} = n_{nodes} \cdot m$ for Barabási-Albert-graphs. This means that we ignore the special initialization step. As for the Barabási-Albert-graphs, we run every experiment 100 times.

6.1.1 Erdős-Rényi graphs

Our graph generator for Erdős-Rényi graphs uses the second variant, where p describes the probability of connecting two nodes. The number of edges in an Erdős-Rényi graph is binomial distributed with

$$n = n_{possible\_edges} = \frac{n_{nodes} \cdot (n_{nodes} - 1)}{2}.$$

The expected value is $E(edges) = n \cdot p$ and should be similar to Barabási-Albert-graphs $E(edges) = n_{nodes} \cdot m$. For a fixed n we can estimate p by

$$n_{nodes} \cdot m = \frac{n_{nodes} \cdot (n_{nodes} - 1)}{2} \cdot p \tag{11}$$

$$p = \frac{2m}{n_{nodes} - 1}. \tag{12}$$

We used the following test parameters for merging two Erdős-Rényi-graphs:

name	n_1	p_1	n_2	p_2
equal	5000	1.2 ‰	5000	1.2 ‰
diff_m	5000	1.2 ‰	5000	3.2 ‰
diff_size	5000	3.2 ‰	25000	0.64 ‰
diff_all	5000	3.2 ‰	25000	0.24 ‰

The split experiments were done with the following parameters:

name	n	m	$noNodes1$
10 : 90	10000	1 ‰	1000
20 : 80	10000	1 ‰	2000
30 : 70	10000	1 ‰	3000
40 : 60	10000	1 ‰	4000
50 : 50	10000	1 ‰	5000

6.1.2 Watts-Strogatz small-world graphs

In the Watts-Strogatz-graphs each node is connected to k neighbors. Every edge is connected with two nodes, so the number of edges in such a graph is $n_{edges} = 0.5 \cdot n \cdot k$. So we set $k = 2 \cdot m$ to get similar graphs to the Barabási-Albert-graphs. Additionally there is a parameter β which describes the randomness of the graph. To analyze the influence of β we did each experiment with $\beta \in \{0.2, 0.4, 0.6, 0.8\}$.

The merge experiment setup was the following:

name	n_1	k_1	n_2	k_2
equal	5000	6	5000	6
diff_m	5000	6	5000	16
diff_size	5000	16	25000	16
diff_all	5000	16	25000	6

Additionally we used the following parameters for splitting:

name	n	m	$noNodes1$
10 : 90	10000	10	1000
20 : 80	10000	10	2000
30 : 70	10000	10	3000
40 : 60	10000	10	4000
50 : 50	10000	10	5000

6.1.3 Extended-Barabási-Albert

The Extended-Barabási-Albert-graphs are an extension of Barabási-Albert-graphs with an additional parameter p. This parameter specifies a probability of creating a triangle with neighbor nodes instead of preferential attachment. We uses exact the same settings as for the Barabási-Albert-graph experiments. Additionally we vary $p \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$.

6.2 Results

All model produce similar test results, so we summarize them and point out special effects.

6.2.1 Edge Preservation

The edge preservation behavior of the presented algorithms, see Table 10, 11 and 12 for merge and Table 13, 14 and 15 for split, is similar to Barabási-Albert-graphs. The NDM, NDDa, and NDDb algorithm generate lower preservations for Erdős-Rényi, and Watts-Strogatz- graphs.

6.2.2 Rank Correlation

Also the rank correlation is comparable to Barabási-Albert-graphs. We figure out that the merging algorithm PNM produces high negative correlations for the *diff_m* setting for all alternative models, see Table 16, 17 and 18. All divide algorithms, beside RD generate lower correlations for Watts-Strogatz-graphs, especially for low β, Table 20. Higher β and Erdős-Rényi-graphs, Table 19, behave like Barabási-Albert-graphs. SED outperforms Extended-Barabási-Albert-graphs w.r.t. Barabási-Albert-graphs for *10:90*, and *20:80*, Table 21.

6.2.3 Node-Degree Distribution

The merging algorithms RM and NDM create new strict Barabási-Albert-graphs. So the RMSE describing the fitting to an optimal scale-free graph decreases dramatical for Erdős-Rényi-graphs, Table 22, and Watts-Strogatz-graphs, Table 23. The Extended-Barabási-Albert-graph is very similar to Barabási-Albert-graphs, so this effect is very low. But we can observe that MM produces better results then pure Barabási-Albert-graphs, see Table 24, and 25.

The splitting algorithms have a similar behavior.
RD, NDDa, and NDDb create new Barabási-Albert-graphs so the RMSE of the resulting graphs is very low, see Table 26, and 27. We can point out that Extended-Barabási-Albert-graphs have an higher RMSE as input graphs then Barabási-Albert-graphs, but the resulting graphs have a lower RMSE, Table 28, and 28 for RSD and SED..

7 Conclusion

Multiple use cases were present, where it could be necessary to merge and divide social graphs. Each theoretical use case had special properties, e.g. preserving node degrees, which we tried to model in proposed algorithms. It is not possible to determine one best algorithm for either merging or

dividing Barabási-Albert-Graphs. Every algorithm has its potential use cases and specific benefits in a subset of our evaluation measures. It is up to the user to decide, which algorithm fits best. A summary of our experimental evaluation results is shown in Table 9 and can be used as a guideline.

If the input algorithm fulfills the scale-free property also the resulting graphs will hold this property. So the algorithms could be used also for graph which are not created by a Barabási-Albert-model, but are scale-free. Due to the fact, that the resulting graphs have similar node-edge relations the sparsity could be assured.

The investigated other graph models generate similar results then the Barabási-Albert-graphs. The RM, and NDM merge as well as the RD, NDDa, and NDDb split transform the graphs into strict Barabási-Albert-graphs. This yields into loss of the input structure. The other algorithms could handle the alternative models in way the input structure would be kept. Naturally, the resulting graphs do not fulfill the properties of the alternative models.

Results will be included in our tool for event generation of dynamic social network simulations [15] in the next step of development. Furthermore the algorithms could be used to improve computational intelligence methods, for instance hierarchical clustering on Barabási-Albert-graphs. Also, our work on dynamic clusters [16][17] in social networks will benefit from these results.

Further research should be done towards the repairing function. With same more advanced repairing steps it would be possible to generate graphs that to fit more the theoretical node-degree distribution.

We provide a Python implementation of the presented algorithms at http://bitbucket.org/paheld/ dynamix.

References

[1] P. ErdÃűs and A. RÃl'nyi, "On random graphs, i," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.

[2] D. J. Watts and S. H. Strogatz, "Collective dynamics of âĂŸsmall-worldâĂŹ networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.

[3] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[4] R. Albert, H. Jeong, and A.-L. Barabasi, "The diameter of the world wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, Sep. 1999, arXiv:cond-mat/9907038.

[5] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Extracting large-scale knowledge bases from the web," in *Proceedings of the 25th VLDB Conference*, 1999, p. 639âĂŞ650.

[6] A.-L. Barabasi, R. Albert, and H. Jeong, "Mean-field theory for scale-free random networks," *Physica A: Statistical Mechanics and its Applications*, vol. 272, no. 1-2, pp. 173–187, Oct. 1999, arXiv:cond-mat/9907068.

[7] W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977.

[8] P. Held, A. Dockhorn, and R. Kruse, "On merging and dividing of barabasi-albert-graphs," in *Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on*. IEEE, Dec. 2014, pp. 17–24.

[9] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," *Physical review E*, vol. 65, no. 2, p. 026107, 2002.

[10] G. Bianconi and A.-l. BarabÃąsi, "Competition and multiscaling in evolving networks," *Europhysics Letters*, vol. 54, p. 436âĂŞ442, May 2001.

[11] E. N. Gilbert, "Random graphs," *The Annals of Mathematical Statistics*, pp. 1141–1144, 1959.

[12] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, and P. Held, *Computational Intelligence: A Methodological Introduction*, ser. Texts in Computer Science. New York: Springer, 2013.

[13] D. Zwillinger and S. Kokoska, *CRC standard probability and statistics tables and formulae*. CRC Press, 1999.

[14] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1-2, pp. 81–93, 1938.

[15] P. Held, A. Dockhorn, and R. Kruse, "Generating events for dynamic social network simulations," in *Proceedings of 15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, ser. Communications in Computer and Information Science, A. Laurent, O. Strauss, B. Bouchon-Meunier, and R. R. Yager, Eds. Switzerland: Springer International Publishing, 2014, pp. 46–55.

Table 9. Oberview of merge and divide algorithms $(-, \circ, +)$

Alg.	Run-Time	Edge Preservation	Rank-Correlation	Node-Degree-Distribution
RM	○	-	-	○
NDM	○	-	○	○
PNM	○	○	○	○
MM	○	+	+	-
RD	+	-	-	+
RSD	○	+	○	-
NDDa	+	-	○	+
NDDb	+	-	+	+
SED	-	+	○	-

[16] P. Held, C. Moewes, C. Braune, R. Kruse, and B. A. Sabel, "Advanced analysis of dynamic graphs in social and neural networks," in *Towards Advanced Data Analysis by Combining Soft Computing and Statistics*, ser. Studies in Fuzziness and Soft Computing, C. Borgelt, M. Á. Gil, J. M. C. Sousa, and M. Verleysen, Eds. Berlin Heidelberg: Springer, 2013, vol. 285, pp. 205–222.

[17] P. Held and R. Kruse, "Analysis and visualization of dynamic clusterings," in *2013 46th Hawaii International Conference on System Sciences*. Los Alamitos, CA, USA: IEEE Computer Society, Jan. 2013, pp. 1385–1393.

A Appendix - Detailed Results

In this appendix we present detailed results from experiments with other models then the Barabási-Albert graphs.

Table 10. Average part of edges preserved after merge-operation in percent for Erdős-Rényi-graphs

Merge	equal	diff_m	diff_size	diff_all
RM	0.1 / 0.1	0.1 / 0.1	0.1 / 0.1	0.0 / 0.0
NDM	0.1 / 0.1	0.0 / 0.3	0.1 / 0.1	0.2 / 0.0
PNM	100 / 0.0	100 / 0.0	100 / 0.0	100 / 0.0
MM	100 / 100	100 / 100	100 / 100	100 / 100

Table 11. Average part of edges preserved after merge-operation in percent for Watts-Strogatz graphs. The results are independent of β

Merge	equal	diff_m	diff_size	diff_all
RM	0.1 / 0.1	0.1 / 0.1	0.1 / 0.1	0.0 / 0.0
NDM	0.1 / 0.1	0.0 / 0.3	0.1 / 0.1	0.2 / 0.0
PNM	100 / 0.0	100 / 0.0	100 / 0.0	100 / 0.0
MM	100 / 100	100 / 100	100 / 100	100 / 100

Table 12. Average part of edges preserved after merge-operation in percent for Extended-Barabási-Albert graphs The results of RM, RNM and MM are independent of p.

Merge	equal	diff_m	diff_size	diff_all
p = 0.2				
RM	0.0 / 0.0	0.1 / 0.1	0.0 / 0.0	0.0 / 0.0
NDM	0.3 / 0.3	0.3 / 1.1	0.2 / 0.4	0.5 / 0.1
PNM	100 / 0.0	100 / 0.0	100 / 0.0	100 / 0.0
MM	100 / 100	100 / 100	100 / 100	100 / 100
p = 0.4				
NDM	0.3 / 0.3	0.3 / 1.1	0.3 / 0.4	0.5 / 0.1
p = 0.6				
NDM	0.3 / 0.3	0.3 / 1.1	0.3 / 0.4	0.5 / 0.2
p = 0.8				
NDM	0.3 / 0.3	0.3 / 1.0	0.3 / 0.3	0.5 / 0.2
p = 1.0				
NDM	0.3 / 0.3	0.4 / 1.0	0.3 / 0.3	0.4 / 0.2

Table 13. Average part of edges preserved after divide-operation in percent of Erdős-Rényi graphs

Divide	10:90	20:80	30:70	40:60	50:50
RD	1.0 / 0.1	0.5 / 0.1	0.3 / 0.1	0.2 / 0.2	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	1.0 / 0.1	0.6 / 0.2	0.4 / 0.2	0.3 / 0.2	0.2 / 0.2
NDDb	1.4 / 0.2	0.7 / 0.2	0.5 / 0.2	0.3 / 0.2	0.3 / 0.3
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100

Table 14. Average part of edges preserved after divide-operation in percent of Watts-Strogatz graphs

Divide	10:90	20:80	30:70	40:60	50:50
$\beta = \mathbf{0.2}$					
RD	1.0 / 0.1	0.5 / 0.1	0.3 / 0.1	0.3 / 0.2	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	1.0 / 0.1	0.5 / 0.2	0.4 / 0.2	0.3 / 0.2	0.2 / 0.3
NDDb	1.2 / 0.1	0.6 / 0.1	0.4 / 0.2	0.3 / 0.2	0.2 / 0.2
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
$\beta = \mathbf{0.4}$					
RD	1.0 / 0.1	0.5 / 0.1	0.3 / 0.1	0.3 / 0.2	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	1.1 / 0.1	0.6 / 0.2	0.4 / 0.2	0.3 / 0.2	0.2 / 0.3
NDDb	1.3 / 0.1	0.7 / 0.2	0.4 / 0.2	0.3 / 0.2	0.2 / 0.3
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
$\beta = \mathbf{0.6}$					
RD	1.0 / 0.1	0.5 / 0.1	0.3 / 0.1	0.3 / 0.2	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	1.1 / 0.1	0.6 / 0.2	0.4 / 0.2	0.3 / 0.2	0.2 / 0.3
NDDb	1.4 / 0.1	0.7 / 0.2	0.4 / 0.2	0.3 / 0.2	0.3 / 0.3
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
$\beta = \mathbf{0.8}$					
RD	1.1 / 0.1	0.5 / 0.1	0.3 / 0.1	0.3 / 0.2	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	1.2 / 0.1	0.6 / 0.2	0.4 / 0.2	0.3 / 0.2	0.2 / 0.3
NDDb	1.3 / 0.1	0.7 / 0.2	0.4 / 0.2	0.3 / 0.2	0.3 / 0.3
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100

Table 15. Average part of edges preserved after divide-operation in percent of Extended-Barabási-Albert graphs

Divide	10:90	20:80	30:70	40:60	50:50
p = 0.2					
RD	0.8 / 0.1	0.4 / 0.1	0.3 / 0.1	0.2 / 0.1	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	2.9 / 0.2	1.8 / 0.2	1.3 / 0.2	1.0 / 0.2	0.9 / 0.2
NDDb	3.4 / 0.6	2.1 / 0.6	1.5 / 0.7	1.1 / 0.8	0.9 / 1.0
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
p = 0.4					
RD	0.7 / 0.1	0.4 / 0.1	0.3 / 0.1	0.2 / 0.1	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	3.0 / 0.2	1.8 / 0.2	1.3 / 0.2	1.1 / 0.2	0.9 / 0.3
NDDb	3.5 / 0.6	2.1 / 0.6	1.5 / 0.7	1.2 / 0.8	0.9 / 1.0
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
p = 0.6					
RD	0.8 / 0.1	0.4 / 0.1	0.3 / 0.1	0.2 / 0.1	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	3.0 / 0.1	1.8 / 0.2	1.3 / 0.2	1.1 / 0.2	0.9 / 0.2
NDDb	3.4 / 0.6	2.1 / 0.6	1.5 / 0.7	1.2 / 0.8	0.9 / 1.0
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
p = 0.8					
RD	0.9 / 0.1	0.4 / 0.1	0.3 / 0.1	0.2 / 0.1	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	3.0 / 0.1	1.8 / 0.1	1.3 / 0.2	1.1 / 0.2	0.9 / 0.2
NDDb	3.3 / 0.6	2.0 / 0.7	1.5 / 0.7	1.1 / 0.8	0.9 / 1.0
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
p = 1.0					
RD	0.8 / 0.1	0.4 / 0.1	0.3 / 0.1	0.2 / 0.1	0.2 / 0.2
RSD	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100
NDDa	3.0 / 0.1	1.8 / 0.1	1.3 / 0.2	1.1 / 0.2	0.9 / 0.2
NDDb	3.2 / 0.6	1.9 / 0.6	1.5 / 0.7	1.1 / 0.8	0.9 / 1.0
SED	100 / 100	100 / 100	100 / 100	100 / 100	100 / 100

Table 16. Rank Correlation Measures for Spearman's ρ and Kendall's τ of Erdős-Rényi graphs

Merge	ρ_{equal}	ρ_{diff_m}	ρ_{diff_size}	ρ_{diff_all}	τ_{equal}	τ_{diff_m}	τ_{diff_size}	τ_{diff_all}
RM	-0.001	-0.002	0.000	0.000	-0.001	-0.001	0.000	0.000
NDM	0.728	0.844	0.875	0.785	0.612	0.703	0.743	0.655
PNM	0.405	-0.435	0.577	0.872	0.337	-0.222	0.459	0.749
MM	0.977	0.984	0.980	0.991	0.952	0.929	0.958	0.968

Table 17. Rank Correlation Measures for Spearman's ρ and Kendall's τ of Watts-Strogatz graphs

Merge	ρ_{equal}	ρ_{diff_m}	ρ_{diff_size}	ρ_{diff_all}	τ_{equal}	τ_{diff_m}	τ_{diff_size}	τ_{diff_all}
$\beta = \mathbf{0.2}$								
RM	0.001	0.000	-0.001	0.000	0.000	0.000	-0.001	0.000
NDM	0.708	0.840	0.864	0.771	0.622	0.713	0.751	0.665
PNM	0.266	-0.620	0.562	0.865	0.218	-0.379	0.464	0.760
MM	1.000	0.953	1.000	0.956	1.000	0.874	1.000	0.924
$\beta = \mathbf{0.4}$								
RM	0.000	-0.001	-0.001	0.000	0.000	-0.001	-0.001	0.000
NDM	0.724	0.842	0.870	0.779	0.627	0.710	0.750	0.664
PNM	0.311	-0.580	0.570	0.876	0.253	-0.333	0.465	0.765
MM	1.000	0.969	1.000	0.974	1.000	0.900	1.000	0.944
$\beta = \mathbf{0.6}$								
RM	-0.001	0.001	-0.001	0.001	-0.001	0.001	-0.001	0.000
NDM	0.728	0.843	0.872	0.781	0.626	0.708	0.749	0.662
PNM	0.334	-0.556	0.573	0.878	0.271	-0.310	0.465	0.765
MM	1.000	0.975	1.000	0.980	1.000	0.910	1.000	0.951
$\beta = \mathbf{0.8}$								
RM	-0.002	-0.001	0.000	-0.001	-0.001	-0.001	0.000	0.000
NDM	0.73	0.843	0.873	0.782	0.625	0.707	0.748	0.661
PNM	0.345	-0.543	0.575	0.879	0.280	-0.298	0.465	0.765
MM	1.000	0.977	1.000	0.982	1.000	0.914	1.000	0.955

Table 18. Rank Correlation Measures for Spearman's ρ and Kendall's τ of Extended-Barabási-Albert graphs

Merge	ρ_{equal}	ρ_{diff_m}	ρ_{diff_size}	ρ_{diff_all}	τ_{equal}	τ_{diff_m}	τ_{diff_size}	τ_{diff_all}
p = 0.2								
RM	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
NDM	0.656	0.842	0.859	0.778	0.566	0.710	0.730	0.661
PNM	0.319	-0.389	0.576	0.831	0.281	-0.225	0.458	0.699
MM	1.000	0.978	1.000	0.984	1.000	0.925	1.000	0.964
p = 0.4								
RM	-0.003	-0.001	0.000	0.000	-0.003	0.000	0.000	0.000
NDM	0.655	0.841	0.859	0.778	0.566	0.710	0.730	0.661
PNM	0.317	-0.391	0.576	0.830	0.280	-0.228	0.458	0.699
MM	1.000	0.977	1.000	0.984	1.000	0.924	1.000	0.963
p = 0.6								
RM	-0.001	0.001	0.000	0.001	0.000	0.000	0.000	0.000
NDM	0.655	0.842	0.859	0.778	0.566	0.710	0.730	0.661
PNM	0.315	-0.396	0.576	0.831	0.279	-0.231	0.458	0.700
MM	1.000	0.977	1.000	0.983	1.000	0.925	1.000	0.963
p = 0.8								
RM	-0.001	0.000	0.001	-0.001	-0.001	0.000	0.000	-0.001
NDM	0.655	0.842	0.859	0.778	0.567	0.710	0.729	0.661
PNM	0.316	-0.403	0.576	0.833	0.279	-0.234	0.457	0.703
MM	1.000	0.978	1.000	0.984	1.000	0.926	1.000	0.963
p = 1.0								
RM	0.000	-0.001	-0.001	-0.001	0.000	-0.001	-0.001	-0.001
NDM	0.655	0.842	0.859	0.778	0.566	0.709	0.728	0.661
PNM	0.316	-0.406	0.577	0.833	0.279	-0.236	0.457	0.704
MM	1.000	0.979	1.000	0.984	1.000	0.927	1.000	0.964

Table 19. Rank Correlation Measures for Spearman's ρ and Kendall's τ of Erdős-Rényi graphs

Divide	$\rho_{10:90}$	$\rho_{20:80}$	$\rho_{30:70}$	$\rho_{40:60}$	$\rho_{50:50}$	$\tau_{10:90}$	$\tau_{20:80}$	$\tau_{30:70}$	$\tau_{40:60}$	$\tau_{50:50}$
RD	0.001	-0.0	-0.001	0.001	-0.002	0.0	-0.0	-0.001	0.001	-0.002
RSD	0.768	0.601	0.563	0.634	0.688	0.694	0.508	0.446	0.5	0.551
NDDa	0.659	0.543	0.46	0.416	0.407	0.55	0.45	0.379	0.341	0.332
NDDb	0.804	0.803	0.803	0.804	0.803	0.675	0.674	0.674	0.675	0.674
SED	0.819	0.645	0.557	0.48	0.452	0.718	0.52	0.432	0.362	0.337

Table 20. Rank Correlation Measures for Spearman's ρ and Kendall's τ of Watts-Strogatz graphs

Divide	$\rho_{10:90}$	$\rho_{20:80}$	$\rho_{30:70}$	$\rho_{40:60}$	$\rho_{50:50}$	$\tau_{10:90}$	$\tau_{20:80}$	$\tau_{30:70}$	$\tau_{40:60}$	$\tau_{50:50}$
$\beta = \mathbf{0.2}$										
RD	-0.001	0.0	-0.002	0.001	-0.0	-0.0	0.0	-0.001	0.001	-0.0
RSD	0.623	0.409	0.31	0.332	0.37	0.541	0.336	0.247	0.264	0.297
NDDa	0.645	0.559	0.44	0.432	0.454	0.55	0.466	0.361	0.349	0.367
NDDb	0.8	0.8	0.801	0.8	0.8	0.695	0.695	0.696	0.696	0.695
SED	0.642	0.456	0.353	0.299	0.276	0.554	0.37	0.279	0.233	0.215
$\beta = \mathbf{0.4}$										
RD	-0.0	-0.001	0.002	0.0	0.001	-0.0	-0.001	0.001	0.0	0.001
RSD	0.689	0.481	0.391	0.424	0.471	0.603	0.396	0.308	0.332	0.374
NDDa	0.663	0.537	0.48	0.415	0.429	0.561	0.448	0.394	0.337	0.348
NDDb	0.811	0.811	0.811	0.81	0.809	0.696	0.696	0.696	0.696	0.695
SED	0.698	0.519	0.409	0.343	0.31	0.592	0.415	0.317	0.262	0.235
$\beta = \mathbf{0.6}$										
RD	0.0	-0.001	-0.001	-0.0	0.0	0.0	-0.001	-0.001	-0.0	0.0
RSD	0.713	0.513	0.429	0.471	0.522	0.629	0.424	0.338	0.369	0.414
NDDa	0.663	0.547	0.482	0.413	0.423	0.561	0.457	0.396	0.337	0.343
NDDb	0.813	0.813	0.813	0.813	0.813	0.695	0.695	0.695	0.694	0.695
SED	0.726	0.55	0.431	0.354	0.318	0.615	0.438	0.331	0.268	0.238
$\beta = \mathbf{0.8}$										
RD	0.0	0.0	0.0	-0.0	0.0	0.0	0.0	0.0	-0.0	0.0
RSD	0.723	0.527	0.449	0.497	0.548	0.64	0.437	0.354	0.389	0.434
NDDa	0.666	0.553	0.482	0.414	0.42	0.563	0.462	0.396	0.339	0.342
NDDb	0.814	0.814	0.814	0.814	0.814	0.694	0.694	0.694	0.694	0.694
SED	0.74	0.565	0.441	0.357	0.322	0.627	0.449	0.338	0.269	0.24

Table 21. Rank Correlation Measures for Spearman's ρ and Kendall's τ of Extended-Barabási-Albert graphs

Divide	$\rho_{10:90}$	$\rho_{20:80}$	$\rho_{30:70}$	$\rho_{40:60}$	$\rho_{50:50}$	$\tau_{10:90}$	$\tau_{20:80}$	$\tau_{30:70}$	$\tau_{40:60}$	$\tau_{50:50}$
p = 0.2										
RD	-0.0	-0.0	-0.0	0.002	0.0	-0.0	-0.0	-0.0	0.001	0.0
RSD	0.77	0.646	0.637	0.693	0.728	0.709	0.557	0.525	0.579	0.619
NDDa	0.635	0.52	0.437	0.382	0.366	0.537	0.437	0.364	0.314	0.299
NDDb	0.781	0.781	0.782	0.782	0.781	0.663	0.663	0.664	0.663	0.663
SED	0.801	0.632	0.529	0.513	0.502	0.704	0.518	0.419	0.407	0.403
p = 0.4										
RD	-0.001	-0.002	0.001	0.0	0.002	-0.001	-0.001	0.001	0.0	0.001
RSD	0.77	0.642	0.632	0.687	0.723	0.708	0.554	0.522	0.574	0.615
NDDa	0.634	0.519	0.436	0.378	0.362	0.537	0.436	0.363	0.312	0.296
NDDb	0.78	0.782	0.782	0.78	0.782	0.663	0.664	0.664	0.663	0.664
SED	0.805	0.646	0.549	0.528	0.521	0.709	0.532	0.438	0.421	0.42
p = 0.6										
RD	0.001	0.001	-0.001	0.002	0.0	0.001	0.001	-0.001	0.001	0.0
RSD	0.771	0.644	0.632	0.687	0.723	0.709	0.555	0.522	0.574	0.615
NDDa	0.635	0.519	0.436	0.379	0.364	0.537	0.436	0.363	0.312	0.297
NDDb	0.781	0.781	0.781	0.781	0.781	0.664	0.664	0.664	0.663	0.664
SED	0.816	0.677	0.585	0.549	0.552	0.722	0.561	0.47	0.439	0.446
p = 0.8										
RD	0.0	0.002	-0.001	0.001	0.0	0.0	0.002	-0.001	0.001	0.0
RSD	0.772	0.646	0.634	0.692	0.726	0.71	0.557	0.523	0.578	0.616
NDDa	0.635	0.52	0.439	0.383	0.37	0.537	0.437	0.365	0.316	0.302
NDDb	0.781	0.782	0.781	0.782	0.781	0.663	0.663	0.663	0.664	0.663
SED	0.822	0.714	0.656	0.605	0.603	0.73	0.598	0.535	0.488	0.49
p = 1.0										
RD	-0.001	-0.001	-0.001	0.001	-0.0	-0.001	-0.001	-0.001	0.001	-0.0
RSD	0.775	0.651	0.643	0.695	0.732	0.712	0.561	0.53	0.579	0.621
NDDa	0.636	0.521	0.44	0.387	0.372	0.538	0.437	0.365	0.318	0.304
NDDb	0.782	0.781	0.781	0.781	0.782	0.663	0.662	0.662	0.662	0.664
SED	0.823	0.783	0.742	0.73	0.729	0.733	0.675	0.625	0.613	0.615

Table 22. RMSE of node-degree distribution before and after merge of Erdős-Rényi graphs

RMSE	equal	diff_m	diff_size	diff_all
g_1	1025	1025	492	492
g_2	1025	485	2645	4905
RM	42	12	14	55
NDM	42	11	14	54
PNM	1709	1711	935	899
MM	1923	1394	3078	4229

Table 23. RMSE of node-degree distribution before and after merge of Watts-Strogatz graphs

RMSE	equal	diff_m	diff_size	diff_all
$\beta = \mathbf{0.2}$				
g_1	1114	1114	540	540
g_2	1116	543	2687	5343
RM	39	12	14	54
NDM	38	12	14	54
PNM	465	469	104	105
MM	2180	1116	3215	3555
$\beta = \mathbf{0.4}$				
g_1	894	894	459	459
g_2	895	458	2272	4262
RM	38	12	14	54
NDM	39	12	14	54
PNM	414	413	96	96
MM	1749	1028	2724	3082
$\beta = \mathbf{0.6}$				
g_1	792	792	424	424
g_2	791	427	2119	3753
RM	39	12	14	53
NDM	38	12	14	55
PNM	381	384	92	92
MM	1553	972	2542	2843
$\beta = \mathbf{0.8}$				
g_1	734	734	411	411
g_2	736	408	2004	3503
RM	38	12	14	54
NDM	39	12	14	54
PNM	365	362	89	88
MM	1435	947	2395	2706

Table 24. RMSE of node-degree distribution before and after merge of Extended-Barabási-Albert graphs

RMSE	equal	diff_m	diff_size	diff_all
$\mathbf{p = 0.2}$				
g_1	60	60	53	53
g_2	59	53	174	276
RM	67	12	19	55
NDM	68	12	19	54
PNM	119	119	10	11
MM	166	188	219	384
$\mathbf{p = 0.4}$				
g_1	96	96	53	53
g_2	97	54	167	291
RM	67	12	19	54
NDM	67	12	19	54
PNM	108	108	10	10
MM	269	193	201	394
$\mathbf{p = 0.6}$				
g_1	83	83	53	53
g_2	72	53	159	227
RM	66	12	19	54
NDM	66	12	19	54
PNM	105	105	10	10
MM	212	182	195	327

Table 25. RMSE of node-degree distribution before and after merge of Extended-Barabási-Albert graphs (continued)

RMSE	equal	diff_m	diff_size	diff_all
$\mathbf{p = 0.8}$				
g_1	57	57	47	47
g_2	53	48	144	207
RM	66	12	19	55
NDM	65	12	19	54
PNM	95	95	18	15
MM	153	167	181	283
$\mathbf{p = 1.0}$				
g_1	17	17	37	37
g_2	18	30	43	49
RM	67	12	19	54
NDM	67	12	18	54
PNM	95	95	25	29
MM	37	149	142	155

Table 26. RMSE of node-degree distribution before and after divide or Erdős-Rényi graphs

RMSE (g_1/g_2)	10:90	20:80	30:70	40:60	50:50
g	1768	1768	1768	1768	1768
RD	4 / 18	7 / 17	9 / 16	10 / 13	12 / 12
RSD	171 / 889	405 / 835	661 / 790	665 / 781	720 / 718
NDDa	5 / 18	6 / 17	9 / 15	10 / 14	12 / 12
NDDb	4 / 18	6 / 17	8 / 15	10 / 14	12 / 12
SED	62 / 746	117 / 582	178 / 520	225 / 435	286 / 368

Table 27. RMSE of node-degree distribution before and after divide or Watts-Strogatz graphs

RMSE (g_1/g_2)	10:90	20:80	30:70	40:60	50:50
$\beta = \mathbf{0.2}$					
g	1534	1534	1534	1534	1534
RD	4 / 15	6 / 14	7 / 13	9 / 12	10 / 10
RSD	180 / 1459	447 / 1276	774 / 1184	869 / 1076	964 / 958
NDDa	4 / 15	6 / 14	7 / 13	9 / 12	10 / 10
NDDb	4 / 15	6 / 14	7 / 13	9 / 11	10 / 10
SED	73 / 1224	142 / 943	221 / 725	300 / 573	381 / 429
381					
$\beta = \mathbf{0.4}$					
g	1291	1291	1291	1291	1291
RD	4 / 15	6 / 14	7 / 13	9 / 12	10 / 10
RSD	175 / 1269	440 / 1156	743 / 1079	791 / 971	899 / 904
NDDa	4 / 15	6 / 14	7 / 13	9 / 11	10 / 10
NDDb	4 / 15	6 / 14	7 / 13	9 / 11	10 / 10
SED	66 / 996	130 / 778	192 / 589	250 / 458	325 / 376
$\beta = \mathbf{0.6}$					
g	1169	1169	1169	1169	1169
RD	4 / 15	6 / 14	7 / 13	9 / 12	10 / 10
RSD	174 / 1169	432 / 1073	729 / 1015	789 / 956	865 / 867
NDDa	4 / 15	6 / 14	7 / 13	9 / 11	10 / 10
NDDb	4 / 15	6 / 14	7 / 13	9 / 11	10 / 10
SED	68 / 897	129 / 672	184 / 530	250 / 434	307 / 362
$\beta = \mathbf{0.8}$					
g	1087	1087	1087	1087	1087
RD	4 / 15	6 / 14	7 / 13	8 / 12	10 / 10
RSD	171 / 1114	423 / 1029	716 / 983	755 / 923	841 / 846
NDDa	4 / 15	6 / 14	7 / 13	9 / 11	10 / 10
NDDb	4 / 15	6 / 14	7 / 13	9 / 12	10 / 10
SED	66 / 836	120 / 650	178 / 517	238 / 420	308 / 344

Table 28. RMSE of node-degree distribution before and after divide or Extended-Barabási-Albert graphs

RMSE (g_1/g_2)	10:90	20:80	30:70	40:60	50:50
$\mathbf{p = 0.2}$					
g	138	138	138	138	138
RD	5 / 22	8 / 21	10 / 18	13 / 17	15 / 15
RSD	89 / 121	130 / 116	189 / 128	161 / 147	136 / 137
NDDa	5 / 23	8 / 20	10 / 19	13 / 17	14 / 15
NDDb	5 / 23	8 / 21	10 / 19	13 / 17	15 / 15
SED	16 / 130	25 / 199	26 / 215	32 / 256	26 / 283
$\mathbf{p = 0.4}$					
g	147	147	147	147	147
RD	5 / 22	8 / 21	10 / 19	12 / 16	15 / 14
RSD	83 / 102	124 / 113	176 / 114	151 / 116	121 / 121
NDDa	5 / 22	8 / 21	10 / 19	13 / 17	15 / 15
NDDb	5 / 22	8 / 20	10 / 19	12 / 17	15 / 15
SED	16 / 119	18 / 183	19 / 226	25 / 253	29 / 244

Table 29. RMSE of node-degree distribution before and after divide or Extended-Barabási-Albert graphs

RMSE (g_1/g_2)	10:90	20:80	30:70	40:60	50:50
$\mathbf{p = 0.6}$					
g	128	128	128	128	128
RD	5 / 23	8 / 20	10 / 19	13 / 17	15 / 15
RSD	83 / 93	115 / 117	163 / 102	119 / 120	110 / 114
NDDa	5 / 23	8 / 21	10 / 19	13 / 17	15 / 15
NDDb	5 / 23	8 / 21	10 / 19	13 / 17	15 / 14
SED	12 / 132	13 / 168	15 / 206	18 / 220	32 / 225
$\mathbf{p = 0.8}$					
g	122	122	122	122	122
RD	5 / 22	8 / 21	10 / 19	12 / 17	15 / 15
RSD	84 / 87	115 / 94	162 / 96	128 / 110	102 / 109
NDDa	5 / 23	8 / 21	10 / 18	13 / 17	15 / 14
NDDb	5 / 22	8 / 21	10 / 19	13 / 16	15 / 15
SED	11 / 138	13 / 178	21 / 192	28 / 210	36 / 194
$\mathbf{p = 1.0}$					
g	157	157	157	157	157
RD	5 / 22	8 / 20	10 / 19	13 / 17	14 / 14
RSD	86 / 84	131 / 88	166 / 96	158 / 100	107 / 109
NDDa	5 / 22	8 / 21	10 / 19	12 / 17	15 / 15
NDDb	5 / 22	8 / 21	10 / 19	13 / 16	15 / 15
SED	9 / 170	19 / 167	28 / 191	40 / 189	54 / 168