



**TECHNISCHE
UNIVERSITÄT
DRESDEN**

Faculty of Computer Science, Chair of Privacy and Data Security

Graph-based Analysis of Dynamic Systems

Dissertation

Submitted to the Faculty of Computer Science, TU Dresden,
in Partial Fulfillment of the Requirements for the Degree of Dr.-Ing.

by

Dipl.-Inform., Dipl.-Math. **Benjamin Schiller**

born on 23 February 1983 in Frankfurt Höchst, Germany

Committee members:

Prof. Dr.-Ing. Thorsten Strufe
Prof. Dr.-Ing. Wolfgang Lehner
Prof. Dr. George Fletcher

TU Dresden, Dresden, Germany
TU Dresden, Dresden, Germany
Eindhoven University of Technology, Eindhoven, Netherlands

Dresden, October 2016

Abstract

The analysis of dynamic systems provides insights into their time-dependent characteristics. This enables us to monitor, evaluate, and improve systems from various areas. They are often represented as graphs that model the system’s components and their relations. The analysis of the resulting dynamic graphs yields great insights into the system’s underlying structure, its characteristics, as well as properties of single components. The interpretation of these results can help us understand how a system works and how parameters influence its performance. This knowledge supports the design of new systems and the improvement of existing ones.

The main issue in this scenario is the performance of analyzing the dynamic graph to obtain relevant properties. While various approaches have been developed to analyze dynamic graphs, it is not always clear which one performs best for the analysis of a specific graph. The runtime also depends on many other factors, including the size and topology of the graph, the frequency of changes, and the data structures used to represent the graph in memory. While the benefits and drawbacks of many data structures are well-known, their runtime is hard to predict when used for the representation of dynamic graphs. Hence, tools are required to benchmark and compare different algorithms for the computation of graph properties and data structures for the representation of dynamic graphs in memory. Based on deeper insights into their performance, new algorithms can be developed and efficient data structures can be selected.

In this thesis, we present four contributions to tackle these problems: A benchmarking framework for dynamic graph analysis, novel algorithms for the efficient analysis of dynamic graphs, an approach for the parallelization of dynamic graph analysis, and a novel paradigm to select and adapt graph data structures. In addition, we present three use cases from the areas of social, computer, and biological networks to illustrate the great insights provided by their graph-based analysis.

We present a new benchmarking framework for the analysis of dynamic graphs, the *Dynamic Network Analyzer* (DNA). It provides tools to benchmark and compare different algorithms for the analysis of dynamic graphs as well as the data structures used to represent them in memory. DNA supports the development of new algorithms and the automatic verification of their results. Its visualization component provides different ways to represent dynamic graphs and the results of their analysis.

We introduce three new stream-based algorithms for the analysis of dynamic graphs. We evaluate their performance on synthetic as well as real-world dynamic graphs and compare their runtimes to snapshot-based algorithms. Our results show great performance gains for all three algorithms. The new stream-based algorithm Stream_k , which counts the frequencies of k -vertex motifs, achieves speedups up to $19,043\times$ for synthetic and $2882\times$ for real-world datasets.

We present a novel approach for the distributed processing of dynamic graphs, called *parallel Dynamic Graph Analysis* (pDNA). To analyze a dynamic graph, the work is distributed by a partitioner that creates subgraphs and assigns them to workers. They compute the properties of their respective subgraph using standard algorithms. Their results are used by the collator component to merge them to the properties of the original graph. We evaluate the performance of pDNA for the computation of five graph properties on two real-world dynamic graphs with up to 32 workers. Our approach achieves great speedups, especially for the analysis of complex graph measures.

We introduce two novel approaches for the selection of efficient graph data structures. The compile-time approach estimates the workload of an analysis after an initial profiling phase and recommends efficient data structures based on benchmarking results. It achieves speedups of up to $5.4\times$ over baseline data structure configurations for the analysis of real-world dynamic graphs. The run-time approach monitors the workload during analysis and exchanges the graph representation if it finds a configuration that promises to be more efficient for the current workload. Compared to baseline configurations, it achieves speedups up to $7.3\times$ for the analysis of a synthetic workload.

Our contributions provide novel approaches for the efficient analysis of dynamic graphs and tools to further investigate the trade-offs between different factors that influence the performance.

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. This thesis was not previously presented to another examination board and has not been published in this form. I am aware that a false statement entails legal consequences.

Benjamin Schiller

City, Date

Acknowledgements

Over the past years, I was blessed with the continuous support and encouragement of a number of people. Without all of their help and support, this thesis would not have been possible.

First and foremost, I like to thank my adviser Thorsten Strufe. I learned a lot from him about doing research, managing projects, and transforming ideas into papers. He allowed me to pursue interesting new ideas and pushed me to do things the right way. He always made time to discuss new ideas and answered the many questions I had about everything. I am very thankful to George Fletcher who agreed to review this thesis as the external referee. I also like to thank Wolfgang Lehner for being my Fachreferent.

The contributions in this thesis would not have been possible without the help and input from collaborators and colleagues. For their input, the lively discussions, and the collaborations in various projects, I thank Jeronimo Castrillon, Clemens Deusser, Kay Hamacher, Irina Heimbach, Oliver Hinz, Sven Jager, Dirk Kohlweyer, Giang Nguyen, Stefanie Roos, Hani Salah, and Jan Seedorf.

I am grateful to the following colleagues and friends for taking the time to read parts of this thesis and helping me to improve it based on their comments: Clemens Deusser, Jan Reubold, Stefanie Roos, Sven Jager, Cornelius Stöhr, and René Wilmes.

Over the past years, I had the pleasure of working closely with a large number of students. Many of them extended the DNA framework with their own implementations. For their work, I thank Nico Haase, Benedict Jahn, Christoph Schott, and Marcel Wunderlich. A special thanks goes out to René Wilmes for implementing large parts of the framework's visualization and data processing components.

This thesis would not have been possible without $C_8H_{10}N_4O_2$ and $C_{25}H_{45}N_5O_{13}$ as well as the joy I got from the work of people like Tom Araya, Jack Black, Geezer Butler, Eric Clapton, Kyle Gass, David Gilmour, Greg Graffin, Toni Iommi, Jamey Jasta, Lemmy Kilmister, Ivan L. Moody, Ozzy Osbourne, John Petrucci, Mike Portnoy, Joey Ramone, Patti Smith, Michael Starr, Corey Taylor, Bill Ward, Steve Winwood, and Frank Zappa (just to name a few).

Last but not least, I thank my family and friends for being there for me, all the time. Thanks to my parents for always believing in me, enabling me to pursue my own path, and always encouraging me to do the things I love. Thanks to Thomäs for supporting me and standing by my side in countless situations. Thanks to Laura for being my Schnecksche.

Contents

1	Introduction	1
1.1	Research Questions	4
1.2	Contributions	5
1.3	Outline	7
2	Notation and Terminology	9
2.1	Graphs	9
2.1.1	Directed and Undirected Graphs	9
2.1.2	Weighted Graphs	10
2.1.3	Incidence and Adjacency Lists	10
2.1.4	Subgraphs	11
2.1.5	Paths	12
2.1.6	Connectivity	12
2.1.7	Trees, Forests, and Spanning Trees	12
2.2	Graph Representations	13
2.2.1	Incidence and Adjacency Lists	13
2.2.2	Adjacency Matrices	13
2.2.3	Storing a Graph in Memory	14
2.3	Graph Properties	15
2.3.1	Properties	15
2.3.2	Metrics	15
2.4	Dynamic Graphs	16
2.4.1	Atomic Updates	16
2.4.2	Compound Updates	16
2.4.3	Batches of Updates	17
2.4.4	Transitions of Dynamic Graphs	17
2.4.5	States of Dynamic Graphs	18
2.4.6	Example of a Dynamic Graph and its Transitions	18
2.5	Dynamic Graph Analysis - Problem Statement	18
2.6	Algorithms for Dynamic Graph Analysis	18
2.6.1	Algorithm Comparison	19
2.6.2	Snapshot-based Algorithms	19
2.6.3	Stream-based Algorithms	19
2.6.4	Batch-based Algorithms	20
2.6.5	Input-based Classification of Algorithms	21
3	Related Work	23
3.1	Data Processing	23
3.1.1	Data Processing Concepts	23
3.1.2	Data Processing Frameworks	24
3.2	Graph Analysis	25
3.2.1	Snapshot-based Graph Analysis Concepts	25
3.2.2	Separate Snapshot-based Graph Analysis Frameworks	25
3.2.3	Consecutive Snapshot-based Graph Analysis Frameworks	27
3.2.4	Stream-based Graph Analysis Frameworks	27
3.3	Algorithms for Dynamic Graph Analysis	28
3.3.1	Sequential Algorithms	28
3.3.2	Parallel Algorithms	29
3.3.3	Vertex-centric Algorithms	29

3.3.4	Streaming Algorithms	30
3.3.5	Dynamic Algorithms	30
3.4	Graph Representation	30
3.4.1	Efficient Graph Representation	31
3.4.2	Graph Databases	31
3.4.3	Profile-guided Selection of Data Structures	31
3.4.4	Adaptive Selection of Data Structures	32
4	DNA - Dynamic Network Analyzer	33
4.1	High-level Features	33
4.2	Requirements	34
4.3	Existing Frameworks	36
4.4	Design of DNA	37
4.4.1	Graph Management	38
4.4.2	Analysis	38
4.4.3	Benchmarking	39
4.4.4	Data Processing	39
4.4.5	Visualization	39
4.5	Implementation of DNA	40
4.5.1	Graph Management	40
4.5.2	Analysis	43
4.5.3	Benchmarking	44
4.5.4	Data Processing	44
4.5.5	Visualization	45
4.5.6	Series	46
4.6	Examples	46
4.6.1	Analyzing Dynamic Graphs	46
4.6.2	Implementing Metrics and Algorithms	47
4.6.3	Implementing Graph Generators	51
4.6.4	Implementing Batch Generators	51
4.6.5	Benchmarking and Comparing Graph Data Structures	52
4.6.6	Benchmarking and Comparing Algorithms	54
4.6.7	Benchmarking and Comparing the Initialization of Algorithms	57
4.6.8	Graph Visualization	58
4.6.9	Result GUI	58
4.7	Discussion	60
5	Algorithms	63
5.1	Degree Distribution	64
5.1.1	Snapshot-based algorithm DD_S	64
5.1.2	Stream-based algorithm DD_U	64
5.1.3	Performance comparison	64
5.1.4	Summary	67
5.2	Rich-club Coefficient	67
5.2.1	Snapshot-based algorithm RCC_S^k	67
5.2.2	Stream-based algorithm RCC_U^k	68
5.2.3	Performance comparison	68
5.2.4	Summary	70
5.3	Motif frequencies	71
5.3.1	Preliminaries	71
5.3.1.1	k -vertex Motifs	71
5.3.1.2	Counting Motifs in Dynamic Graphs	72
5.3.1.3	MD Trajectories	72
5.3.1.4	Modeling MD Trajectories as Dynamic Graph	73
5.3.2	Stream-based Algorithm $Stream_k$	73
5.3.2.1	Basic Insights	73
5.3.2.2	Processing Steps	74
5.3.2.3	Algorithm	75
5.3.2.4	Execution Complexity	75
5.3.3	Performance Evaluation	76
5.3.3.1	Implementation	76

5.3.3.2	Datasets	76
5.3.3.3	Setup	77
5.3.3.4	Evaluation using Artificial Graphs	78
5.3.3.5	Evaluation using MD Trajectory Graphs	79
5.3.4	Summary	80
5.4	Summary	81
6	Parallel Dynamic Network Analysis	83
6.1	Introduction	83
6.2	Approach	84
6.2.1	Partitioner	84
6.2.2	Worker	85
6.2.3	Collator	85
6.2.4	Workflow	85
6.3	Subgraph Types and Work Distribution	86
6.3.1	Separate Subgraphs	86
6.3.2	Overlapping Subgraphs	87
6.3.3	Full Subgraphs	87
6.4	Collation of Local Results	88
6.4.1	Degree Distribution	88
6.4.2	All-Pairs Shortest Paths	88
6.4.3	Betweenness Centrality	89
6.4.4	Clustering Coefficient	89
6.4.5	Weakly Connected Components	89
6.5	Evaluation	91
6.5.1	Implementation	91
6.5.2	Datasets	92
6.5.3	Graph Metrics	92
6.5.4	Evaluation Setup	92
6.5.5	Subgraph Characteristics	93
6.5.6	Performance of the Partitioner	93
6.5.7	Performance of the Worker	94
6.5.8	Performance of the Collator	95
6.5.9	Overall Performance of pDNA	96
6.6	Summary	98
7	Selection of Efficient Graph Data Structures	101
7.1	Problem Statement	102
7.1.1	Configuration of Graph Data Structures	102
7.1.2	Required Operations on Data Structures	102
7.1.3	Problem Definition	103
7.2	Compile-time Selection of Efficient Data Structures	103
7.2.1	Compile-time Approach	103
7.2.2	Benchmarking Results	104
7.2.3	Profiling Results	107
7.2.4	Evaluation	107
7.2.5	Summary of the compile-time approach	109
7.3	Run-time Selection of Efficient Data Structures	111
7.3.1	Run-time Approach	111
7.3.2	Performance analysis	112
7.3.3	Summary of the Run-time Approach	115
7.4	Summary	115
8	Use Cases	117
8.1	Growing a Web of Trust	117
8.1.1	Analysis of the PGP WoT	118
8.1.2	Existing Graph Models	120
8.1.3	WoT_{gr} - a WoT growth model	120
8.1.4	WoT_{com} - a community-based WoT model	121
8.1.5	Evaluation	123
8.1.6	Summary	128

8.2	Graph-based Intrusion Detection System	128
8.2.1	Preliminaries	129
8.2.2	Approach	130
8.2.3	Graph Models	131
8.2.4	Evaluation	131
8.2.5	Summary	136
8.3	Molecular Dynamics	136
8.3.1	Structural Properties of an Enzyme	137
8.3.2	Thermodynamics of Water Molecules	138
8.3.3	Summary	140
8.4	Summary	141
9	Conclusion	143
A	DNA - Dynamic Network Analyzer	175
A.1	Implementations of Graph Components	175
A.2	Implementations of Graph and Batch Generators	177
A.3	Implementations of Metrics	178
B	Algorithms	181
B.1	Degree Distribution	181
B.2	Rich-Club Coefficient	182
B.3	Motif Frequencies	184
C	Selection of Efficient Graph Data Structures	195
C.1	Estimation functions	195
C.2	Measurements and fitted functions	198
D	Parallel Dynamic Network Analysis	201
D.1	Implemented Strategies and Collation Algorithms	201
D.2	Subgraph Statistics	202
D.3	Runtimes of Performance Evaluation	203
E	Graph-based Intrusion Detection System	207
E.1	RFR Feature Ranking	207
E.2	Top-ranked features by LS for $c_{[0,1]}$	208
E.3	Top-ranked features by LS for $c_{[0,4]}$	209
E.4	Top-ranked features by RFR for $c_{[0,1]}$	210
E.5	Top-ranked features by RFR for $c_{[0,4]}$	211
E.6	Attack Detection Rates	212
F	Molecular Dynamics	213

List of Figures

1.1	General process of the <i>Analysis of Dynamic Systems</i>	1
1.2	Results of an analysis performed at different frequencies	2
1.3	General process of the <i>Graph-based Analysis of Dynamic Systems</i>	4
2.1	Examples of graphs with 5 vertices	10
2.2	Examples of undirected weighted graphs	11
2.3	Examples of tree graphs	12
2.4	\mathcal{A}_3 - all adjacency matrices of undirected 3-vertex graphs	14
2.5	Transitions of dynamic graph G from G_0 to G_5	18
2.6	Workflow of snapshot-based algorithms for dynamic graph analysis	19
2.7	Workflow of stream-based algorithms for dynamic graph analysis	20
2.8	Workflow of batch-based algorithms for dynamic graph analysis	20
2.9	Execution of stream-/batch-based algorithms before and after application of update/batch	21
4.1	Component groups, components, and workflow of DNA	38
4.2	Results from the analysis of the <i>Hypertext 2009</i> dataset	48
4.3	Size of V or E during benchmark using <i>WorkloadMetric</i>	53
4.4	Benchmarking results for the comparison of four graph data structures	54
4.5	Statistics recorded while benchmarking algorithms for <i>EdgeCounting</i>	55
4.6	Runtime measured while benchmarking algorithms for <i>EdgeCounting</i>	56
4.7	Results and heuristics qualities recorded while benchmarking algorithms for <i>EdgeCounting</i>	56
4.8	Benchmarking initialization: snapshot-based (R) vs. stream-based (U)	57
4.9	Visualization of graphs generated using basic generators	59
4.10	Visualization of analysis results using DNA's <i>Result GUI</i> component	59
4.11	Visualization of statistics and runtimes using DNA's <i>Result GUI</i> component	60
5.1	Basic properties of the dynamic graph in the first scenario (Edge Change)	65
5.2	Degree distribution: runtimes and results for the first scenario (Edge Change)	66
5.3	Basic properties of the dynamic graph in the second scenario (Random Growth)	66
5.4	Degree distribution: runtimes and results for the second scenario (Random Growth)	67
5.5	Rich-club coefficient: runtimes and results for the first scenario (Edge Change)	70
5.6	Rich-club coefficient: runtimes and results for the second scenario (Random Growth)	70
5.7	Examples for the set of motifs \mathcal{M}_k for different sizes	72
5.8	Unit-sphere model for the connection of representative atoms of an MD trajectory	73
5.9	Transitions between isomorph 4-vertex graphs	74
5.10	Steps of a stream-based algorithm for maintaining the motif frequencies $F_{\mathcal{M}_k}$	74
5.11	Speedup of <i>StreaM_k</i> for the analysis of synthetic dynamic graphs	78
5.12	Absolute runtimes for the analysis of <i>elec</i> and <i>roget</i>	78
5.13	Absolute runtimes for the analysis of <i>Complex</i> and <i>Pnb</i>	79
5.14	Absolute runtimes for the analysis of <i>Loops</i> and <i>Complex</i>	80
5.15	Speedup of <i>StreaM_k</i> for the analysis of <i>Complex</i> and <i>4AOA</i>	80
6.1	Workflow of pDNA, comprised of <i>Partitioner</i> , <i>Collator</i> , and two workers w_1 and w_2	86
6.2	Example graph G_i with partition $P_i = \{\{v_1, v_2, v_3, v_4\}, \{v_5, v_6, v_7, v_8\}\}$	86
6.3	Separate subgraphs for w_1 and w_2 (assigned vertices marked by thick border)	87
6.4	Overlapping subgraphs for w_1 and w_2 (assigned vertices marked by thick border)	87
6.5	Full subgraphs for w_1 and w_2 (assigned vertices marked by thick border)	88
6.6	Average subgraph sizes for <i>FB</i> , depending on the number of workers	93
6.7	<i>Partitioner</i> runtimes for generating subgraphs of different types	94

6.8	Worker runtimes for computing the metrics for FB	95
6.9	Worker runtimes for computing the metrics for Pnb_{12}	96
6.10	Collator runtimes for collating the results for FB	97
6.11	Runtimes of all components for the analysis of different metrics for FB	98
6.12	Runtimes of all components the analysis of different metrics for Pnb_{12}	98
7.1	Runtimes for different operations on lists of size k between 1,000 and 100,000	101
7.2	Compile-time optimization of data structures for dynamic graph analysis	103
7.3	Selected runtime estimations	105
7.4	Operation counts for graph modification ($o \in \{init, add_s, size, cont_f, get_s, rem_s\}$)	107
7.5	Operation counts for metric computation ($o \in \{size, iter, cont_s, cont_f\}$)	108
7.6	Dataset statistics (development of $ V $ and $ E $ over time)	108
7.7	Speedup of compile-time approach (for analysis of constant workload (MD))	110
7.8	Speedup of compile-time approach (for analysis of non-constant workload (FB))	111
7.9	Run-time optimization of data structures for dynamic graph analysis	112
7.10	List sizes (development of $ V $ and $ E $ during application of the synthetic workload)	113
7.11	Workload runtimes (execution of synthetic (non-constant) workload, round 3)	114
7.12	Overhead of run-time approach (consisting of hotswap and recommendation)	115
7.13	Speedup of run-time approach (for application of synthetic workload, 4 rounds)	115
8.1	Basic properties of WoT_{25k} , the analyzed instance of the PGP WoT	119
8.2	Adding vertex and edges in each round of WoT_{gr}	121
8.3	Interconnection of communities in the community-based WoT_{com} model	122
8.4	Parameter study for WoT_{com}	124
8.5	Comparison of relevant graph properties from our models and existing ones to WoT_{25k}	125
8.6	Comparison of our models with FF , FF_d , and WoT_* for increasing vertex count	126
8.7	Comparison of our models with FF , FF_d , and WoT_{55k}	128
8.8	Overview of the general workflow of intrusion detection systems	129
8.9	Overview of our approach, <i>graph-based Intrusion Detection System</i> (gIDS)	130
8.10	Example graphs generated for $t=13:20:05$ and lifetime $\lambda=5$ s	132
8.11	Examples for subgraphs of the Host-port-Host graph model (lifetime $\lambda=1280$ s)	133
8.12	Features computed during an <i>ipsweep</i> attack ($\nu=1$ s, $\lambda=320$ s)	133
8.13	Cumulative importance of features ranked by LS	135
8.14	Overall attack detection rate, depending on number of features used	136
8.15	Visualizations of <i>para Nitro Butyrate Esterase-13</i> (generated using PyMol [o70])	137
8.16	RMSD and frequencies of the 4-vertex motifs m_3 and m_4 over time	138
8.17	Co-occurrences of values for RMSD and frequencies of the 4-vertex motifs m_3 and m_4	138
8.18	Spatial measures of SPC/E water	139
8.19	Visualizations of <i>SPC/E water in a cubic box</i> (generated using PyMol [o70])	140
8.20	Frequencies of 3-vertex motifs and number of interactions for density of 0.9998 kg/m^3	140
8.21	Analysis of 3-vertex motifs of liquid and vapor	141
8.22	Convergence of the number of interactions for different temperatures of water simulations	141
B.1	Absolute runtimes for the analysis of synthetic dynamic graphs (grouped by k)	184
B.2	Speedup of Stream M_k for the analysis of synthetic dynamic graphs (grouped by k)	184
B.3	Absolute runtimes for the analysis of synthetic dynamic graphs (grouped by dataset)	185
B.4	Absolute runtimes for the analysis of MD traes (grouped by k)	188
B.5	Absolute runtimes for the analysis of MD trajectories (grouped by d)	189
C.1	Measurements and fitted functions for all operations and list type $t=t_v$	198
C.2	Measurements and fitted functions for all operations and list type $t=t_e$	199
D.1	Average subgraph sizes for Pnb_7 , depending on the number of workers	202
D.2	Average subgraph sizes for Pnb_{12} , depending on the number of workers	202
D.3	Worker runtimes for computing the metrics for Pnb_7	203
D.4	Collator runtimes for collating the results for Pnb_7	203
D.5	Collator runtimes for collating the results for Pnb_{12}	204
D.6	Runtimes of all components the analysis of different metrics for Pnb_7	204
D.7	Total runtimes for SSSP $_k$ and BC $_k$ for FB , $k \in \{32, 64, 128\}$	205
E.1	Cumulative importance of features ranked by RFR	207
E.2	Detection rate of attack classes, depending on number of features used	212

List of Tables

1.1	System-specific tasks to solve for the graph-based analysis of a dynamic system	3
1.2	Research questions investigated in this thesis	4
2.1	Examples of directed and undirected adjacency matrices	13
2.2	Notation and argument scope of topology and weight updates	16
2.3	Notation and argument scope for compound updates	16
2.4	Input-based classes of algorithms for processing a dynamic graph	21
4.1	High-level features of a benchmarking framework for dynamic graph analysis	33
4.2	Requirements for a benchmarking framework for dynamic graph analysis	35
4.3	Existing frameworks and their fulfillment of the requirements stated in Section 4.2	36
4.4	Generators for reading graphs from files (<i>dna.graph.generators.reading</i>)	41
4.5	Generators for graph models (<i>dna.graph.generators.model</i>)	41
4.6	Implementations of the different update types supported in DNA (<i>dna.updates.update</i>)	42
4.7	Batch generators for reading data from files (<i>dna.updates.generators.reading</i>)	42
4.8	Batch generators that model the evolution of a graph (<i>dna.updates.generators.evolving</i>)	42
4.9	Batch generators that contain random updates (<i>dna.updates.generators.random</i>)	42
4.10	Additional interfaces for batch- and stream-based algorithms	44
5.1	Statistics about adjacency matrices and motifs of different sizes	72
5.2	Properties of the synthetic dynamic graphs	76
5.3	Properties of the MD trajectory graphs for lowest and highest distance threshold	77
6.1	Problems to solve when distributing the analysis of dynamic graphs	84
6.2	Basic properties of the datasets used for the evaluation of pDNA	92
7.1	Estimation functions of get_s and get_f depending on data structure and element type	106
7.2	Fastest data structure according to our estimation for different list sizes	106
7.3	Recommendations for V , E , and adj depending on workload and computed metric	109
7.4	Recommended data structures (for workload and set size, <u>underlined: swap required</u>)	113
8.1	Basic properties of WoT_{25k} , the analyzed instance of the PGP WoT	118
8.2	Impact of parameter values on the properties of graphs generated using WoT_{com}	123
8.3	Parameters chosen for the evaluation of all models based on the properties of WoT_{25k}	124
8.4	Properties of graphs with 25,487 vertices generated with all models (baseline , <u>closest values</u>)	125
8.5	Example of a packet trace	130
8.6	Example for a netflow (generated from the packet trace in Table 8.5)	130
8.7	Example of a simplified netflow	132
8.8	Statistics about the attack classes in the DARPA'98 dataset	133
8.9	F_1 scores for the SVM and RF classifiers for <i>regular</i> and <i>attack</i> classes ($c_{[0,1]}$)	134
8.10	Selection of experimental entropies for the simulation of SPC/E water	139
A.1	Implementations of vertices (<i>dna.graph.nodes</i>)	175
A.2	Implementations of edges (<i>dna.graph.edges</i>)	175
A.3	Implementations of weights (<i>dna.graph.weights</i>)	176
A.4	Data structures implemented in DNA for the representation of graphs	176
A.5	Operations for the use as workloads (<i>dna.metrics.workload.operations</i>)	176
A.6	Generators for canonical graphs (<i>dna.graph.generators.canonical</i>)	177
A.7	Util graph generators (<i>dna.graph.generators.util</i>)	177
A.8	Implementations of batch generators for modeling sampling	177

A.9	Implementation of utility batch generators	178
A.10	Metrics implemented in DNA (R: snapshot-based, B: batch-based, U: stream-based)	180
B.1	Absolute runtimes for the analysis of synthetic dynamic graphs	186
B.2	Speedup of Stream M_k for the analysis of synthetic dynamic graphs	187
B.3	Absolute runtimes for the analysis of <i>Loops</i>	190
B.4	Speedup of Stream M_k for the analysis of <i>Loops</i>	190
B.5	Absolute runtimes for the analysis of <i>Complex</i>	191
B.6	Speedup of Stream M_k for the analysis of <i>Complex</i>	191
B.7	Absolute runtimes for the analysis of <i>4AOA</i>	192
B.8	Speedup of Stream M_k for the analysis of <i>4AOA</i>	192
B.9	Absolute runtimes for the analysis of <i>Pnb</i>	193
B.10	Speedup of Stream M_k for the analysis of <i>Pnb</i>	193
C.1	Estimation functions for <i>Array</i> depending on operation o	195
C.2	Estimation functions for <i>ArrayList</i> depending on operation o	195
C.3	Estimation functions for <i>HashSet</i> depending on operation o	196
C.4	Estimation functions for <i>HashMap</i> depending on operation o	196
C.5	Estimation functions for <i>HashTable</i> depending on operation o	196
C.6	Estimation functions for <i>LinkedList</i> depending on operation o	197
C.7	Estimation functions for <i>HashMap</i> depending on operation o	197
D.1	Implementations of partitioning strategies s_p (<i>dna.parallel.partitioning</i>)	201
D.2	Implementations of vertex assignment strategies s_v (<i>dna.parallel.nodeAssignment</i>)	201
D.3	Implementations of collation metrics (<i>dna.parallel.collation</i>)	201
E.1	15 top-ranked features by Lasso Scores for $c_{[0,1]}$ (HH, $\lambda = 5$ s)	208
E.2	15 top-ranked features by Lasso Scores for $c_{[0,1]}$ (HpH, $\lambda = 5$ s)	208
E.3	15 top-ranked features by Lasso Scores for $c_{[0,1]}$ (nFG, $\lambda = 5$ s)	208
E.4	15 top-ranked features by Lasso Scores for $c_{[0,4]}$ (HH, $\lambda = 5$ s)	209
E.5	15 top-ranked features by Lasso Scores for $c_{[0,4]}$ (HpH, $\lambda = 5$ s)	209
E.6	15 top-ranked features by Lasso Scores for $c_{[0,4]}$ (nFG, $\lambda = 5$ s)	209
E.7	15 top-ranked features by Random Forest Regressor for $c_{[0,1]}$ (HH, $\lambda = 5$ s)	210
E.8	15 top-ranked features by Random Forest Regressor for $c_{[0,1]}$ (HpH, $\lambda = 5$ s)	210
E.9	15 top-ranked features by Random Forest Regressor for $c_{[0,1]}$ (nFG, $\lambda = 5$ s)	210
E.10	15 top-ranked features by Random Forest Regressor for $c_{[0,4]}$ (HH, $\lambda = 5$ s)	211
E.11	15 top-ranked features by Random Forest Regressor for $c_{[0,4]}$ (HpH, $\lambda = 5$ s)	211
E.12	15 top-ranked features by Random Forest Regressor for $c_{[0,4]}$ (nFG, $\lambda = 5$ s)	211
F.1	Experimental entropies for the simulation of SPC/E water	213

List of Listings

4.1	Interface <i>IDataStructure</i> that are required to be implemented by all list representations	40
4.2	Interface <i>IGraphGenerator</i> that needs to be implemented by all graph generators	41
4.3	Interface <i>IBatchGenerator</i> that are required to be implemented by all batch generators	42
4.4	Interface <i>IMetric</i> that are required to be implemented by all metrics	43
4.5	Java code for the analysis of the <i>Hypertext 2009</i> dataset	47
4.6	<i>EdgeCounting</i>	49
4.7	<i>EdgeCountingR</i>	49
4.8	<i>EdgeCountingB</i>	49
4.9	<i>EdgeCountingU</i>	50
4.10	<i>EdgeCountingUH</i>	50
4.11	<i>EdgeCountingUIncorrect</i>	50
4.12	Graph generator <i>dna.graph.generators.model.RandomGraph</i>	51
4.13	Batch generator <i>dna.updates.generators.random.RandomEdgeAdditions</i>	52
4.14	Batch generator <i>dna.updates.generators.random.RandomEdgeRemovals</i>	52
4.15	Batch generator <i>dna.updates.generators.random.BatchRoundRobin</i>	53
4.16	Benchmarking graph data structures using five different workload operations	54
4.17	Code example for benchmarking different algorithms for edge counting	55
4.18	Log output of <i>Comparison</i> component while benchmarking algorithms for <i>EdgeCounting</i>	56
4.19	<i>weg iwegpewewjgopjew</i>	57
4.20	Minimal program to generate the graph visualizations shown in Figure 4.9	58
A.1	Batch generator <i>BatchFromGraph</i>	178
A.2	Interface <i>ISnapshotBased</i> for snapshot-based algorithms	178
A.3	Interface <i>IBatchBased</i> for snapshot-based algorithms	179
A.4	Interface <i>IStreamBased</i> for snapshot-based algorithms	179
A.5	<i>WorkloadMetric</i> for the generation of pre-defined workloads	179
B.1	Implementation of DD_S in <i>DNA</i> (<i>dna.metrics.degree.DegreeDistribution</i>)	181
B.2	Implementation of DD_U in <i>DNA</i> (<i>dna.metrics.degree.DegreeDistributionU</i>)	181
B.3	Implementation of $RCC_{>k,S}^U$	182
B.4	Implementation of $RCC_{>k,S}^U$	183

List of Algorithms

1	DD_U - a snapshot-based algorithm for computing the degree distribution	64
2	DD_U - a stream-based algorithm for computing the degree distribution	65
3	RCC_S^k - a stream-based algorithm for computing the rich-club coefficient	68
4	RCC_U^k - a stream-based algorithm for computing the rich-club coefficient	69
5	$Stream_k$ for maintaining F_{M_k} in dynamic graphs	75
6	Collation of the degree distribution metric for overlapping subgraphs	88
7	Collation of the all-pairs shortest paths metric for full subgraphs	89
8	Collation of the betweenness centrality metric for full subgraphs	89
9	Collation of the clustering coefficient metric for overlapping subgraphs	90
10	Collation of the connected components metric for overlapping subgraphs	90
11	Collation of the connected components metric for separate subgraphs	91
12	Adding a new vertex and edges with WoT_{gr}	121
13	Generating a complete graph with WoT_{com}	122

