Chapter 1

Introduction

An ever increasing demand exists for the timely analysis of large dynamic systems in order to monitor, evaluate, and improve them. These systems are characterized by their constantly changing properties and the need for their subsequent analysis. The in-depth analysis of systems like biological, computer, and traffic networks promises great insights into their working principles, key properties, and the time-dependent changes thereof. It allows us to monitor the development of fundamental characteristics over time and thereby to understand how a dynamic system works. Using the results of an analysis, we can judge the impact of parameters on a system's properties and performance. These insights assist the development of new systems and the optimization of existing ones. The detailed analysis of a system allows us to identify components that are important to a specific task. The timely analysis of dynamic systems enables us to quickly react to anomalies and thereby guarantee the correct functionality of a system.

Examples of dynamic systems can be found in many areas such as computer networks, power grid management, traffic systems, social networks, and biological systems. The identification of vulnerable points in computer networks, or power grids, enables us to protect them against random failures and targeted attacks [152, 144, 9, 8, 79]. The surveillance of air as well as road traffic is a great asset in the prevention of delays and traffic jams [229, 304]. Analyzing similarities of users in constantly changing online social networks like Facebook, LinkedIn, and Twitter enables intricate advertisement schemes and is an important resource for the service providers [150]. The stability and heat resistance of a protein can be improved by identifying relevant interactions between amino acids and strengthening their corresponding bonds [114, 38, 239].

Analyzing a system means to determine its characteristics. Analyzing a dynamic system means to track the change of its characteristics over time. A general workflow for this process is illustrated in Figure 1.1. The state of a dynamic system, which changes over time, is observed and subsequently analyzed at certain points in time. The interpretation of the analysis results provides insights into the development of the corresponding system over time. The question which properties are relevant, therefore depends on the system itself and the analyst's objectives.

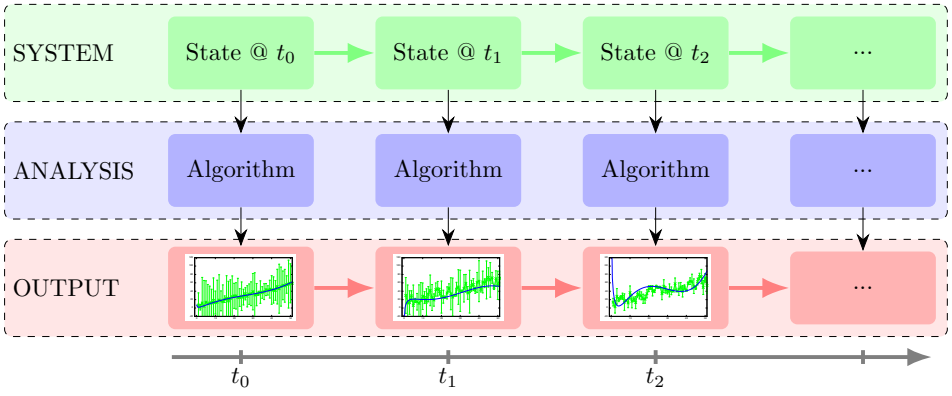

Figure 1.1: General process of the *Analysis of Dynamic Systems*

The analysis of a dynamic system is commonly performed with some frequency, e.g., every second, every minute, or every 24 hours. The system's state is analyzed for every corresponding point in time and a set of properties is output as a result. The duration between the analyzed points in time depends

on the analysis-frequency. A high frequency results in the output of more data points in a time interval compared to an analysis at low frequency.

The separate computation of relevant properties for each point in time is the basic way to analyze a dynamic system. The analysis takes the complete state as input and computes the properties of interest for each point in time. Increasing the frequency of an analysis therefore leads to the frequent re-computation of all properties based on the complete state. The performance of this approach does not scale well with an increasing frequency and is therefore not a good fit for analyses that require the frequent output of analysis results.

The event-based maintenance of a system's properties is another approach to analyze dynamic systems. Every event that occurs in the analyzed system is used to trigger an analysis that updates the current properties based on the implied change to the system. It takes advantage of the already computed properties and restricts the analysis to the single event. This approach is better suited in case an analysis at a high frequency is desired where only few changes occur between two points in time.

The frequency of an analysis has a high impact on its ability to describe the characteristics of a dynamic system accurately. Lowering the frequency of an analysis leads to the loss of information because fewer data points are output as results. This can be irrelevant in case a system's properties change rarely and steadily, like the room temperature. In case a system's properties change frequently and briefly, the expressiveness of analysis results decreases with the frequency. Interesting incidents may remain unobserved or be noticed too late, like the sudden increase of incoming traffic during a Denial-of-Service attack in a computer network. Therefore, it is commonly desirable to analyze a dynamic system at a high frequency.

The analysis results for a biological system at different frequencies are shown in Figure 1.2 as an example[1]. At the highest frequency, 1024 states of the system are analyzed while only 512, 256, 128, and 64 data points are contained in the results of the corresponding analyses with lower frequencies. Even for the second highest frequency, which outputs 512 data points, certain peaks are missed. The 64 data points, which result from the analysis at the lowest frequency, fail to depict the actual changes that occur in the system over time.

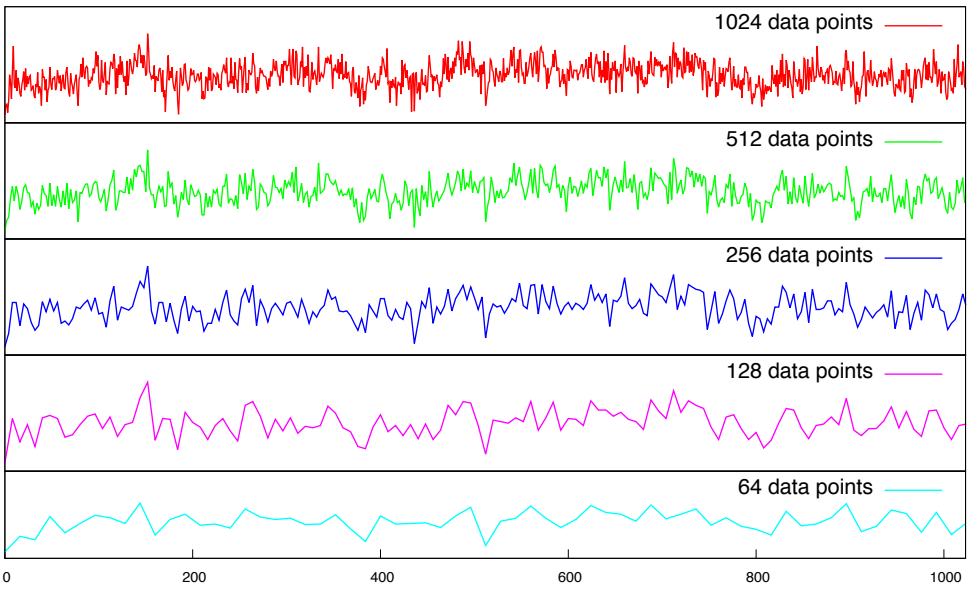

Figure 1.2: Results of an analysis performed at different frequencies

The characteristics of many systems are best described by the relationships between their components. Therefore, graphs are often used to model systems from various fields, including biology [6, 216], chemistry [153, 25], computer networks [110, 341], transportation networks [135], social sciences [54], and online social networks [280, 318]. The components of these systems are represented as vertices of a graph while their relations are modeled as edges that connect the corresponding vertices. These systems change over time as new components join and existing ones leave. Relations can be created and dissolved over time as well.

As examples, consider online social networks and air traffic systems. Users of an online social network are represented as vertices while their friendship relations are expressed as edges between them. Airports,

[1]Here, the frequencies of a single subgraph over time in the graph model of a molecular dynamics trajectory of the enzyme *para Nitro Butyrate Esterase-13* are shown. At the highest frequency of 2.5 ps, the system is analyzed at 1024 points in time. The lower analysis frequencies of 5 ps, 10 ps, 20 ps, or 40 ps result in the output of fewer data points.

the components of air traffic systems, are modeled as vertices, which are connected by edges that represent connecting flights. The creation of a new user profile in an online social network leads to the addition of a vertex to the graph while the shutdown of an airport results in the removal of the corresponding vertex. In case a new friendship relation is established between two users, the corresponding vertices are connected by an edge while the cancellation of the route between two airports leads to the removal of the edge connecting their respective vertices in the graph.

A large number of graph measures has been developed to analyze graphs modeled from arbitrary systems [108, 280]. Some measures compute global properties for the whole graph like, e.g., global clustering coefficient [237], degree distribution [99], and rich-club coefficient [348]. They reveal properties of an entire graph and thereby relate to overall system properties. Other measures express properties for each vertex like, e.g., local clustering coefficient [320], betweenness centrality [33], and PageRank [247]. Their values can be directly related to the corresponding component of the modeled system and reflect their characteristics.

The representation of a dynamic system as a graph and its subsequent analysis is a great way to utilize the expressiveness of graphs. The meaning and interpretation of analysis results depends on the analyzed system and the model used to represent it as a graph. It is therefore crucial to first determine the system's characteristics of interest. A system can then be modeled as a graph to reflect the desired properties. The graph measures that should be computed during analysis must then be selected to reflect the characteristics of interest.

Three system-specific tasks must therefore be solved to realize the graph-based analysis of a dynamic system: model a dynamic graph from the system (*T1*), compute graph properties relevant to the system (*T2*), and interpret the results to deduce characteristics of the underlying system (*T3*). An overview of these tasks is given in Table 1.1.

ID	Description
T1	Model the dynamic system as a graph
T2	Compute meaningful properties of the dynamic graph
T3	Interpret the analysis results and deduce properties of the dynamic system

Table 1.1: System-specific tasks to solve for the graph-based analysis of a dynamic system

The identification of influential users in online social networks is of great interest for recommendation systems [43], targeted advertisements [336], and the early detection of hate groups [241]. Systems like Twitter are often modeled as follower graphs where users are represented as vertices and their follower relations are modeled by directed edges from follower to followee. The number of incoming edges of a user's vertex, referred to as in-degree, is then considered as a measure of influence in the network. While this local measure denotes the influence of single users, the overall measure of the in-degree distribution represents the ratio between users of different influence levels.

The analysis of a dynamic system based on measures computed from its graph representation is a promising approach. A general workflow for the process of the *graph-based analysis of dynamic systems* is shown in Figure 1.3. A dynamic graph is modeled, which reflects the underlying system and all changes occurring to it over time. The dynamic graph's properties are analyzed at specific points in time, as for the general process for the analysis of dynamic systems. The analysis results are interpreted to reveal the system's characteristics.

Different approaches exist for the analysis of dynamic graphs. Snapshot-based approaches compute the graph properties of interest based on a single state of the graph [127], just like the separate computation introduced before. Stream-based approaches apply the idea of the event-based maintenance described earlier. They update the previously computed graph properties based on single updates to the graph instead of re-computing everything for each point in time [59].

The main issue that arises in this scenario is the performance of executing the analysis of a dynamic graph. Many algorithms have been developed for snapshot- as well as stream-based approaches to compute various graph measures. Unfortunately, there is no straight-forward way to predict which approach or algorithm performs best for the analysis of arbitrary dynamic graphs. The overall performance of dynamic graph analysis depends on two main factors: the approach and algorithm used for the computation of graph measures and the data structures used to represent the dynamic graph in memory. Other factors are the size, topology, and type of the analyzed graph, the type of changes that occur, and the analysis-frequency.

It poses a challenging problem to decide which approach or algorithm to use for the computation of certain graph properties for a specific graph. This creates the need for a general way to benchmark and compare different approaches and algorithms. Such a benchmarking approach should allow for the

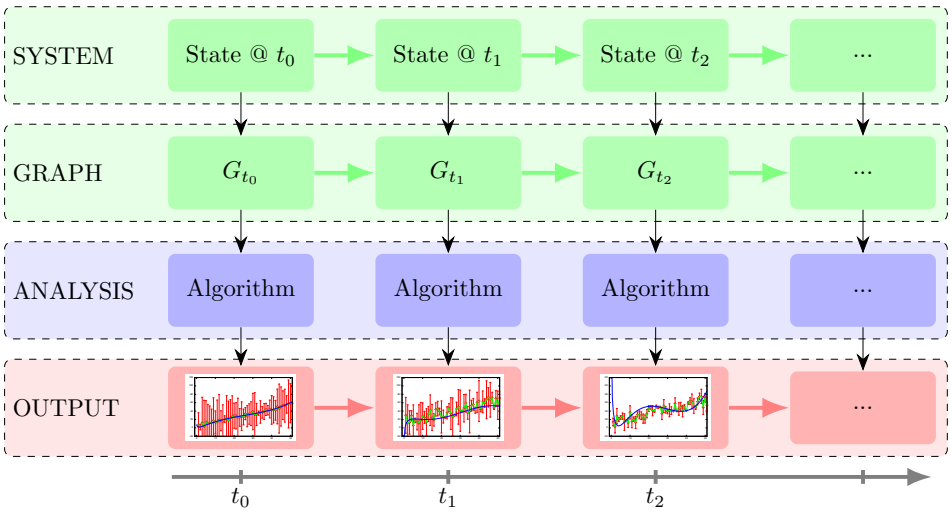

Figure 1.3: General process of the *Graph-based Analysis of Dynamic Systems*

comparison of approaches and algorithms for specific classes of dynamic graphs and analysis frequencies.

Many stream-based algorithms have been developed to compute certain graph measures. Still, there exist many graph measures for which no stream-based algorithms have yet been developed. It is therefore promising to investigate the development of new stream-based algorithms to speed up the analysis of dynamic graphs.

The performance of stream-based algorithms scales well when increasing the analysis-frequency. It does not scale well with the growth of the analyzed graph, especially for graph measures that investigate large fractions of the whole graph for each processed update. We should therefore investigate possibilities to distribute the analysis of dynamic graphs among multiple processing units.

The performance of dynamic graph analysis highly depends on the data structures used to represent and maintain the dynamic graph in memory. Which representation performs best depends on the patterns of read operations during analysis and write operations during graph maintenance as well as the number of elements stored. It is not always easy to foresee which data structure performs best for a given scenario. It highly depends on the structure and type of the dynamic graph, the algorithms used for its analysis, and the analysis-frequency. It is therefore necessary to provide a way to benchmark and compare different data structures for the representation of dynamic graphs.

The benchmarking and comparison of data structures provides insights into their corresponding benefits and drawbacks. Even with this knowledge, it cannot easily be decided when to use which data structure. We should therefore investigate the automatic selection of data structures for the analysis of dynamic graphs.

In the next Section, we state five research questions that are based on these general problems for the efficient analysis of dynamic graphs.

1.1 Research Questions

In this Section, we pose five research questions that should be investigated and answered in order to provide the efficient graph-based analysis of dynamic systems as mentioned before. An overview of these research questions is given in Table 1.2.

ID	Description
Q1	How can algorithms for dynamic graph analysis be benchmarked and compared?
Q2	How can graph measures be computed efficiently for dynamic graphs?
Q3	How can we speed up the analysis of dynamic graphs using distributed processing?
Q4	How can different graph representations for dynamic graphs be benchmarked and compared?
Q5	How can we determine the most efficient representation of a dynamic graph?

Table 1.2: Research questions investigated in this thesis

Q1 Different approaches exist to analyze dynamic graphs in general. For each approach and graph measure, a multitude of algorithms has been developed. Which algorithm performs best depends on many factors, like the size and topology of the graph, the number of changes between two analyzed states, and the type of changes that occur. It is not straight-forward to predict the best algorithm for a given scenario. We therefore require a way to benchmark and compare different algorithms in various scenarios to help analysts decide which algorithm they should use to achieve the best performance.

Q2 A large number of algorithms has been developed so far to efficiently compute various graph measures using snapshot-based approaches. For many measures, no stream-based algorithms have yet been developed, even though, they promise great performance gains over snapshot-based algorithms for the analysis of dynamic graphs at high frequency. This provides opportunities for the development of faster algorithms for the stream-based analysis of dynamic graphs. We should therefore investigate the efficient computation of graph measures in dynamic graphs.

Q3 The scalability of dynamic graph analysis at high frequencies can be achieved using stream-based algorithms. For large graphs and complex graph measures, the analysis using stream-based algorithms does not suffice to provide an efficient analysis of the corresponding graph. We should therefore investigate how to speedup the dynamic graph analysis using distributed processing.

Q4 The data structures used to represent a dynamic graph in memory have a big impact on the overall performance of dynamic graph analysis. It is therefore crucial to provide means for their proper benchmarking and comparison in order to judge their respective benefits and drawbacks for specific operations. We therefore require a way to benchmark and compare different data structures for the representation of dynamic graphs for different graph types and sizes as well as arbitrary combinations of algorithms for the computation of graph measures.

Q5 There exist many efficient representations of dynamic graphs but it is unclear which one performs best in a given scenario. Their performance is influenced by many factors which make it hard to foresee the best representation beforehand. Therefore, we should investigate how to determine the most efficient graph representation of a dynamic graph to use it during an analysis.

In this thesis, we investigate all these questions. In the next Section, we give an overview of our contributions to answer them.

1.2 Contributions

In this thesis, we investigate the five research questions posed in Section 1.1 and present four related contributions:

1. A benchmarking framework for dynamic graph analysis,
2. three novel algorithms that enable the efficient analysis of dynamic graphs,
3. an approach for the parallelization of dynamic graph analysis, and
4. a novel paradigm to select and adapt the data structures for dynamic graph analysis.

In addition, we present three use cases for the graph-based analysis of dynamic systems. They serve as examples for demonstrating the three system-specific tasks *T1*, *T2*, and *T3*. In the remainder of this Section, we give an overview of our four contributions and the three investigated use cases. Then, we detail collaborations and related publications that contribute to the content of this thesis.

Benchmarking Framework It is crucial to compare the performance of different algorithms for the analysis of dynamic graphs and their representation using various data structures. We therefore present a benchmarking framework for the analysis of dynamic graphs, called *Dynamic Network Analyzer* (DNA) in Chapter 4. It allows the generation of dynamic graphs of various types and arbitrary sizes to support the benchmarking and comparison of algorithms and data structures. The framework also supports the development of new algorithms of different types with verifications for their correctness and the automatic determination of the precision of their results. Its extensive visualization components for dynamic graphs and analysis results enables us to monitor them during their analysis. Overall, it provides means to benchmark and compare algorithms (*Q1*) and graph data structures (*Q4*). It supports the development

of new algorithms and their subsequent performance analysis. DNA thereby enables us to investigate new approaches for the efficient analysis of dynamic graphs (*Q2*).

Algorithms The performance of dynamic graph analysis highly depends on the algorithms used to compute the graph measures of interest for a dynamic graph. In Chapter 5, we present and evaluate new stream-based algorithms to speed up the analysis of dynamic graphs and thereby provide means to analyze them at high frequencies (*Q2*). We introduce novel stream-based algorithms for the computation of degree distribution, rich-club coefficient, and k-vertex motif frequencies. We evaluate and compare their performance to existing snapshot-based algorithms using synthetic as well as real-world dynamic graphs. Our results show that these novel algorithms are able to speed up the analysis of dynamic graphs compared to their snapshot-based counterparts. The stream-based algorithm for the computation of k-vertex motif frequencies, called StreaM$_k$, achieves great speedups compared to existing approaches and thereby enables the analysis of highly dynamic systems originating from biology.

Parallelization of Dynamic Graph Analysis The performance of dynamic graph analysis highly depends on the graph's size and the complexity of considered graph measures. The analysis does not scale well with an increase in graph size, especially for complex graph measures. We therefore present a novel approach for the distributed processing of dynamic graph analysis, called *parallel Dynamic Graph Analysis* (pDNA) in Chapter 6. The computational workload is distributed among workers based on a partition of the vertex set. Corresponding subgraphs are assigned to each worker, which computes the respective measure on its local graph view. The changes to the main graph are propagated accordingly to the workers. The results from all workers are then aggregated into the measures for the whole graph for each point in time in a collation step. We evaluate pDNA for the analysis of five graph measures on real-world dynamic graphs. Our results reveal great speedups for the distributed analysis of dynamic graphs using pDNA, especially for complex graph measures.

Data Structure Selection The data structures used to represent a dynamic graph in memory have a high impact on the performance of algorithm execution and the maintenance of the dynamic graph over time. The performance for the execution of single operations on data structures is well understood and can be investigated using benchmarks. The sizes of data structures and the frequencies of operations executed on them are not easy to foresee in the context of dynamic graph analysis. This makes it hard to predict which data structures perform best for the analysis of certain graph measures on a specific dynamic graph. We therefore present a novel paradigm for the selection and adaptation of the data structures for dynamic graph analysis in Chapter 7. Our approach determines efficient data structures to represent a dynamic graph in memory during analysis (*Q5*). It consists of two parts: a compile-time and a run-time selection of efficient data structures. The compile-time approach is applicable to dynamic graphs whose workload does not change significantly over time. The run-time approach enables the exchange of data structures during run-time and thereby supports scenarios in which the workload changes significantly during the analysis. We evaluate our approaches using synthetic workloads as well as workloads that originate from the analysis of real-world dynamic graphs. In both cases, our approaches achieve great speedups compared to baseline data structures.

Use Cases We show the general applicability of the graph-based analysis for dynamic systems that originate from three different fields: social networks, computer networks, and biological networks. We showcase the development of new graph models to grow Web-of-Trust graphs over time. We illustrate the expressiveness of dynamic graph analysis as feature generators for the detection of intrusions in a computer network. We present the analysis of dynamic graphs modeled from molecular dynamics trajectories and show that the graph-based analysis is more expressive than existing analysis approaches from that field.

Collaborations While I am the sole author of this thesis, its content is the result of extensive discussions with coworkers, collaborators, and my supervisor Prof. Dr. Thorsten Strufe. The DNA framework, presented in Chapter 4, has been used and extended by many students that I worked with. Various metrics and algorithms have been implemented by Tim Grube, Benedict Jahn, Bastian Laur, Christoph Schott, Maurice Wendt, Jan Wiese, and Marcel Wunderlich. Generators for dynamic graphs have been implemented by Benedict Jahn, Christoph Schott, Maurice Wendt, Jan Wiese, Réné Wilmes, and Marcel Wunderlich. The first version of the graph data structures component was implemented by Nico Haase and extended with interfaces to graph databases by Matthias Jordan. Large parts of the aggregation and visualization components have been implemented by Réné Wilmes. The algorithm StreaM$_k$, presented in Section 5.3, has been evaluated using trajectories from molecular dynamics simulations provided by

Sven Jager. These datasets are also used in the evaluation of pDNA, presented in Chapter 6, and the selection of efficient data structures, presented in Chapter 7. Jeronimo Castrillon and Clemens Deusser helped designing the evaluation of the compile- and run-time approaches for the selection of efficient data structures, presented in Chapter 7. Dirk Kohlweyer and Jan Seedorf co-developed the Web-of-Trust model, presented in Section 8.1. Matthias Fischer, Jan Reubold, and Réné Wilmes collaborated in the development and evaluation of gIDS, presented in Section 8.2. Kay Hamacher, Michael Vogel, and Sven Jager interpreted the results of the analyzed molecular dynamics simulations, presented in Section 8.3. I use these collaborative results in agreement with my collaborators. To indicate that part of the results presented in this thesis are the outcome of collaborations, I use the pronoun 'we' instead of 'I' throughout this thesis.

Related Publications An initial version of the DNA framework, presented in Chapter 4, was published and presented at SummerSim 2013 [s15]. The algorithm StreAM, was published and presented at WABI 2016 [s4]. StreaM$_k$, the algorithm presented in Section 5.3, was included in an article entitled "Motif-based Analysis of Molecular Dynamics Simulations" and submitted to the SIAM Interdisciplinary Journal of Multiscale Modeling and Simulation. The algorithm StreaM, a predecessor to StreaM$_k$, was published and presented at AlCoB 2015 [s11]. The compile-time approach for the selection of efficient data structures, presented in Section 7.2, was published and presented at ComplexNetworks 2015 [s9]. An extension of this approach as well as the run-time approach, presented in Section 7.3, was published in the Journal of Applied Network Science in September 2016 [s10]. The models for generating instances of the Web-Of-Trust, presented in Section 8.1, have been published and presented at LCN 2015 [s17]. The model has also been used in a demonstration that was published and presented at ICN 2014 [s18]. The framework GTNA, used for the development and analysis of the model was published and presented at SpringSim 2010 [s8] and an extension at SummerSim 2013 [s16]. The analyses results of molecular dynamics trajectories, presented in Section 8.3, were published and presented at AlCoB 2015 [s11] and are included in the article submitted to the Journal of Multiscale Modeling and Simulation.

1.3 Outline

The remainder of this thesis is structured as follows: In Chapter 2, we introduce the basic concepts and our notation of graphs, dynamic graphs, and their analysis. We describe and discuss work related to our research questions in Chapter 3. In Chapter 4, we present a new benchmarking framework for the analysis of dynamic graphs, called *Dynamic Network Analyzer* (DNA). We present and analyze new algorithms for the analysis of dynamic graphs in Chapter 5. We present *parallel Dynamic Network Analysis* (pDNA), an approach for the parallelization of dynamic graph analysis in Chapter 6. In Chapter 7, we present a compile-time and a run-time approach for the selection of efficient data structures for dynamic graph analysis. In Chapter 8, we present use cases for the graph-based analysis of dynamic systems from three areas: social networks, computer networks, and biological networks. Finally, we conclude this thesis and our contributions in Chapter 9.