EXPLORING BEHAVIORAL PATTERNS IN COMPLEX ADAPTIVE SYSTEMS

by

Andrii Cherniak

B.S. in applied physics 2004,

M.S. in radio physics and electronics 2006,

Taras Shevchenko National University of Kyiv,

Kyiv, Ukraine

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This thesis was presented

by

Andrii Cherniak

It was defended on

April 22nd, 2014

and approved by

Vladimir I. Zadorozhny, PhD, Associate Professor

Marek J. Druzdzel, PhD, Associate Professor

Konstantinos Pelechrinis, PhD, Assistant Professor

Paul Munro, PhD, Associate Professor

Jesse Bridgewater, PhD, Principal Data Scientist / Director of Data Science, eBay Inc

Thesis Director: Vladimir I. Zadorozhny, PhD, Associate Professor

EXPLORING BEHAVIORAL PATTERNS IN COMPLEX ADAPTIVE SYSTEMS

Andrii Cherniak, PhD

University of Pittsburgh, 2014

Many phenomenons in real world can be characterized as complex adaptive systems (CAS). We are surrounded with a huge number of communicating and interacting agents. Some of those agents may be capable of learning and adapting to new situation, trying to achieve their goals. E-commerce, social media, cloud computing, transportation network and real-time ride sharing, supply chain are a few examples of CAS. These are the systems which surround us in every days life, and naturally we want to make sense of those systems and optimize systems behavior or optimize our behavior around those systems. Given the complexity of these systems, we want to find a set of simplified patterns out of the seeming chaos of interactions in a CAS, and provide more manageable means of analysis for such systems.

In my thesis I consider a few example problems from different domains: modeling human behavior during fire evacuation, detection of notable transitions in data streams, modeling finite resource sharing on a computational cluster with many clients, and predicting buyer behavior on the marketplace. These (and other) seemingly different problems demonstrate one important similarity: complex semi-repetitive or semi-similar behavior. This semi-repetitive behavior poses a challenge to model such processes. This challenge comes for two major reasons: 1) state-space explosion and sparsity of data 2) critical transitions and precision of process modeling

I show, that the analysis of smilingly different CAS coming from different domains, can be performed by following the same recipe.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I still cannot believe I am finishing writing my thesis. It seems like it was a week ago when I started the program, and now I have a bit over 160 pages document, summarizing what I have done in the last five and a half years.

This work was not possible without many great people, who came into my life, who helped me to undertake this endeavor. Those are people I would like to take a moment to thank them.

First of all, I would like to say thank you to my adviser, who helped me in making sense of where my dissertation is going, helped to summarize the results and finally helped to crystalize ROCAS model, which is the central part of my thesis.

Secondly, many thanks go to eBay Inc for supporting my research via a number of internships. I thank Huma Zaidi and Tony Thrall for the support during my first internship. I thank Jesse Bridgewater for his support and ideas during my second internship. And finally I thank Andy Edmonds for ”adopting” me into his team and Sudha Jamthe for financial support during my third internship.

Thirdly, I thank my roommate, Vicente Ordonez Roman, for all those all night long discussions on different aspects of machine learning and computer vision : it helped me to complete the last chapter of my thesis.

And last, but not least, I say thank you to the members of my committee for actually taking time and being on my committee, and providing me with valuable feedback.

1.0 INTRODUCTION

Many phenomenons in real world can be characterized as complex adaptive systems **(CAS)** Bullock and Cliff [2004], Miller and Page [2007]. We are surrounded with a huge number of communicating and interacting agents. Some of those agents may be capable of learning and adapting to new situation, trying to achieve their goals. E-commerce Hao et al. [2010], social media Kanter [2009], cloud computing Jhawar and Piuri [2013], transportation network Hulsmann et al. [2009] and real-time ride sharing Amey [2010], supply chain Choi et al. [2001] are a few examples of CAS. These are the systems which surround us in every day's life, and naturally we want to make sense of those systems and optimize systems' behavior (e. g. more efficient supply chain) or optimize our behavior around those systems (e.g. avoid price surge Berkovici [2014]). Given the complexity of these systems, we want to find a set of simplified patterns out of the seeming chaos of interactions in a CAS, and provide more manageable means of analysis for such systems.

1.1 DEFINITION OF A COMPLEX ADAPTIVE SYSTEM

Literature sources differ in the exact wording of what to consider as a complex adaptive system **(CAS)** Bullock and Cliff [2004], Miller and Page [2007]. Often, the definition for a CAS is provided through examples of some real-life systems followed by a set of rules or properties, defining what a CAS is Miller and Page [2007]. Other sources define a CAS as a complex, self-similar collection of interacting adaptive agents MacLennan [2007], Faucher et al. [2008], Holland [2006].

This definition of a CAS, which I am using in my thesis, is similar to the definition of a multi-agent system **(MAS)**. What distinguishes a CAS from a MAS is the focus on the top-level properties of the system, including:

- self-similarity: when an object is exactly or approximately similar to a part of itself Mandelbrot [1965]
- emergent behavior can be observed when a number of simple entities (agents) operate in an environment, forming more complex behaviors as a collective. The common characteristics are: radical novelty; coherence or correlation; a global or macro "level"; it is the product of a dynamical process Goldstein [1999]
- self-organization Bak [1999] is a process where some form of global order or coordination arises out of the local interactions between the components of an initially disordered system. This process is spontaneous: it is not directed or controlled by any agent or subsystem inside or outside of the system. It is often triggered by random fluctuations that are amplified by positive feedback. The resulting organization is completely decentralized or distributed over all the components of the system. It is typically very robust and able to survive and self-repair substantial damage or perturbations.

Instead of trying to analyze the behavior of every individual agent in a multi-agent system, we are looking for a more simplified analysis of complex, emergent and macroscopic properties of the system.

Emergence is the way complex systems and patterns arise out of relatively simple interactions between interacting agents. The concept of "emergence" may generally be subdivided into "weak emergence" and "strong emergence" Clayton and Davies [2008].

We can say that a high-level phenomenon is weakly emergent with respect to a low-level domain when the high-level phenomenon arises from the low-level domain. It is a type of emergence in which the emergent property is reducible to its individual constituents.

Strong emergence implies that if systems can have qualities not directly traceable to the system's components, but rather to how those components interact. These new qualities are irreducible to the system's constituent parts. The whole is greater than the sum of its parts. The systems can have qualities not directly traceable to the system's components, but rather to how those components interact. These system properties are important for CAS analysis.

1.2 CAS ANALYSIS: REALITY VS. IDEAL WORLD

In my thesis I consider four example problems from different domains: modeling human behavior during fire evacuation, detection of notable transitions in data streams, modeling finite resource sharing on a computational cluster with many clients, and predicting buyer behavior on the marketplace. These (and other) seemingly different problems demonstrate one important similarity: complex semi-repetitive or semi-similar behavior. This semi-repetitive behavior poses challenges to model and analysis of such processes. These challenges comes for two major reasons:

- **state-space explosion and sparsity of data**: when we analyze buyer activity on the marketplace or resource sharing on a computational cluster between many users, we are dealing with data, coming from multiple users. Humans do not repeat themselves precisely. Moreover, they react and adapt to the change in the environment. Thus their actions may be very similar to what they have been doing before, yet a tiny bit different. If we want to build a model for a CAS which captures every nuance in their behavior, we may need to build a **very large state-space representation** for these problems. However, as state-space grows, we may experience **data sparsity problem**, that we do not have enough data to train the obtained model. From the state-space perspective, **we want to reduce the size of state-space Chen et al. [2008] as much as we can**.
- **critical transition detection** : at the same time, complex dynamic systems demonstrate transitions between completely different types of the behavior, known as phase transitions Saitta et al. [2011]. To detect such transitions as soon as possible or to model these characteristic changes, we want to have very detailed state-space model. Moreover, CAS tend to be open systems, which allow external factors to play an important role (e.g. if we model buyer behavior, ideally we want to know how deep a buyer's pocket is). From this perspective, **we want to gather and put in a model as much information as possible, and ultimately increase the state-space**.

These are contradicting requirements for the analysis. However we want to utilize both of these properties in a standard methodology for CAS analysis.

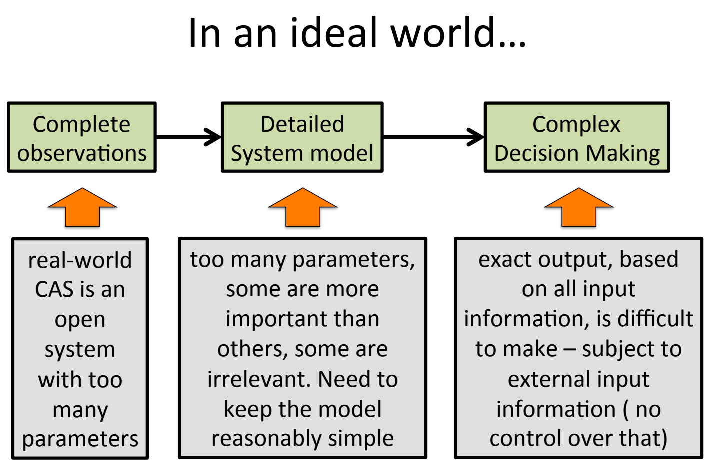

In an ideal world…
Complete observations
Detailed System model
Complex Decision Making
real-world CAS is an open system with too many parameters
too many parameters, some are more important than others, some are irrelevant. Need to keep the model reasonably simple
exact output, based on all input information, is difficult to make – subject to external input information (no control over that)

Figure 1.1: Decision making for a CAS in ideal world

Probably one of the most obvious methodological schemas for CAS analysis can be shown as **Figure 1.1**. In this ideal world we would rely on being able to collect complete observations from the system, build a pretty detailed system model, and use some sort of complex decision making (which takes into consideration all system parameters) to come up with the best result or strategy for system optimization. However, this is an idealized schema, and in real life we face some of the fundamental problems associated with this schema:

- CAS quite often is an open system: the system is influenced with a number of factors, which we cannot observe and / or control. Consider a an example of online shopping. Each buyer has his own agenda and requirements: budget, item delivery time range, item selection diversity and quality, etc. We have no means to collect this extra information, and cannot explicitly include in the model
- Quite often we cannot build a detailed system model. we may have too many system parameters and not enough training data to learn the model. Thus we have to make certain assumptions about parameter relationships which we cannot learn and essentially simplify the model
- Often complex decision making process is not feasible either. This happens because CAS

are often a subject to stability issues, when even small changes in one parameter may have a dominant influence

It seemed at first, that each CAS analysis is a unique task and methods, developed to analyze one system cannot be applied to analyze another one. However, at the end of my thesis, when I considered for different examples of CAS, I noticed, that the analysis of these different systems follows the same methodology, which I present in **Section 1.3**.

1.2.1 CAS optimization and analysis procedures can resemble a search for a solution for an NP-hard problem

It is worth mentioning, that some examples of CAS resemble NP-hard problems Arora and Barak [2009]. However, this similarity was not obvious at the beginning of my research study, and this was not the premise on what I started my research. For instance, the task of fire evacuation, which I talk about in **Chapter 2**, looks very similar to supply-chain optimization problem Arthur F. Veinott [2005]. It does not resemble one-to-one similarity, of course. We do not see dedicated producers and consumers, and the routing graphs. However, because the building has limited capacity, when we try to move people in this environment, we need to free enough space for them to be able to move. Thus the whole environment resembles a distributed producer-consumer relationship, with a central consumer (exit). Another example is the task of resource distribution in a computational cluster between multiple processes. This task is quite similar to flow shop scheduling problem Varadharajan and Rajendran [2005]. Again, there are some differences here as well. For instance, the task I was trying to solve possesses an element of uncertainty, when at completely arbitrary time we can expect to receive yet another set of interdependent jobs to be executed.

While I mention that some CAS resemble NP-hard problems, **my goal is not to provide a formal proof that these problems are indeed NP-hard**, or can be reduced to one of NP-complete problems. However, there are two implications, emerging from this similarity.

The first implication is that we should not always expect an easy (exact) solution to the problem of CAS analysis. And instead we might want to consider other alternatives, like methods to obtain approximate solutions.

The second implication is that we can re-use some of the techniques from the NP-hard problems solutions toolbox. However, it became clear only at the end of my thesis that I indeed use methods, similar to those used for NP-hard problems solutions. Particularly, I use relaxation techniques and constraint programming to build **simplified patterns or simplified features**. Since we do not need to re-invent the terminology, it is easier just to refer to the established names.

1.2.2 Agent-based models vs. systems of differential equations for CAS analysis

Before proceeding to the description of the proposed methodology for CAS analysis in **Section 1.3**, it is important to mention agent-based modeling **ABM**, which can be used for the same purpose. ABM is a class of computational models for simulating the actions and interactions of autonomous agents with a view to assessing their effects on the system as a whole. The goal of ABM is to find explanations and insights about the collective behavior of a system (emergent properties) which consists of agents following simple rules. The models simulate the simultaneous operations and interactions of multiple agents, in an attempt to re-create and predict the appearance of complex phenomenaEpstein and Axtell [1996]. Successful applications of such modeling can be found in different domains, like: economics Tesfatsion [2002], Axtell [2005], Axelrod [1997], electric power grid analysis Sueyoshi and Tadiparthi [2008], Macal et al. [2005], Wu et al. [1998], Wehinger [2010], Lamparter et al. [2010], diseases spread predictionLaroum and Tighiouart [2011] and others.

Interactions between agents form very complex patterns and dynamics. Changing dynamics Epstein and Axtell [1996] is a phenomenon which is very difficult to model. And it requires to capture very complex interactions between agents. This interactions can be spotted while performing multi-agent simulations.

Alternative methods, based on systems of differential equations, were used to simulate the development of social systems. For instance, predator-prey model Mehlum et al. [2003] and Bellman equationMiranda and Fackler [2002] are widely used in economics for a wast variety of purposes, including pricing models, future income, capital gains, taxation, economic policies and many others. As opposed to multi-agent simulation, systems of differential equations rely on predicting a particular objective assuming known interactions between other factors.

A natural question is: which method is better? And how can we determine that? The beauty of agent-based simulation is the simplicity. We assign some basic behavioral parameters to each agent (often - the same behavioral model), and let the system run, and we collect the required observations from the system, as it develops. Even by using simple rules, multi-agent simulation produces results, very close to real-world observations Epstein and Axtell [1996]. Sometimes simulation is capable to produce more accurate results Laroum and Tighiouart [2011] about the dynamics in the system, than the system of differential equations. Also, simulation can predict very complicated events, like California electricity crisis in 2000 - 2001 Sueyoshi and Tadiparthi [2008]. From this point, simulation may be seen as the ultimate solution to the analysis of CAS and multi-agent societies.

However, agent-based simulation produces good results under one condition: the behavior of the agents is correctly modeled. If during the design of agent behavior we do not capture all important aspects, the multi-agent simulation may produce wrong results. For instance, Macal et al. [2005] mentions verification procedure as a mandatory step in multi-agent system design. This assures that the simulated development of a multi-agent system actually reflects the real system dynamics. Agent behavior model selection actually can be as complicated as establishing relationship between system parameters for the differential equations equivalent. For instance, Sueyoshi and Tadiparthi [2008] presents numerical results to explain fluctuations and spikes in electricity wholesale prices during California crisis in 2001. Despite the fact, that this approach operates under multi-agent model, individual agent behavioral model takes into consideration a long list of equations to modify and parametrize agent behavior. Wehinger [2010] uses a regression-based approach to tune agents' behavior, which implies that we establish the relationship between environmental factors and agent behavior via the help of equations, very similar to the systems of differential equations.

The corollary of this comparison is: agent-based simulation approach may seem easier than establishing a system of differential equations to describe the dynamics of the system. Often we can describe complex phenomenons via simulation using simple agent behavioral models. Yet, if we care to calibrate and validate a multi-agent model on extreme cases or to use it to model previously unseen behavior, this process requires complex model tuning and essentially incorporating some external knowledge. This process closely resembles writing differential equations for agent behavior. Moreover, we cannot always learn every aspect of agent behavior, just because we are dealing with a huge mix of agents pursuing different goals. I consider an example of such CAS in **Chapter 4**.

We cannot advocate that the system of differential equations is better or worse than the agent-based simulation: it all depends on the application, and the accuracy of the results which we expect. Differential equations are often used to find equilibrium conditions in the system development. However, usually this analysis does not include time component: the assumption is that the system will reach its equilibrium before its major parameters change. That this is not always true, however: if it takes too long for the system to reach the equilibrium, and a system parameter change happens before that, we may never observe our expected result. In **Chapter 3** I start with the example, which demonstrates such scenario.

When we use differential equations to describe system dynamics, we may need to establish the applicability limits, and possibly to augment those equations with additional conditions. Moreover, for equations we need to define system parameters upfront, and make sure that we have captured all important behavioral features, which may not always be true. Especially if we are dealing with a new problem, and there is not much prior research done on the topic.

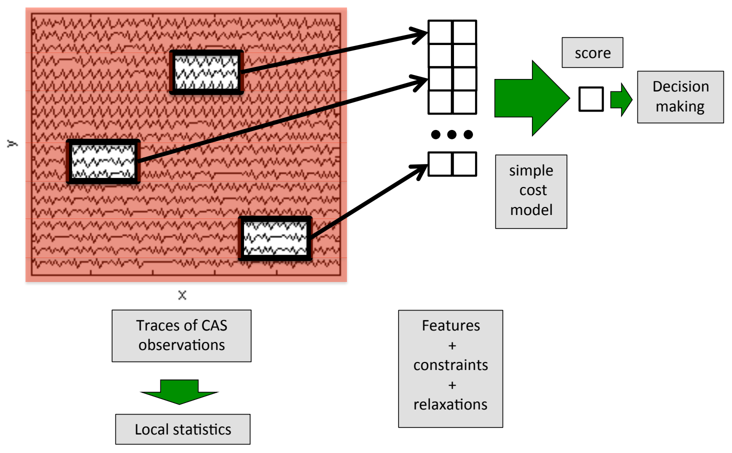
In this comparison, there is no clear winner. I utilize both approaches in my proposed method, which I present in the following section.

1.3 RELAXED OPTIMIZATION IN CAS (ROCAS)

Given all the complexity of interactions in CAS, as I mentioned in the previous sections, we do not want to chase every single trend Chen et al. [2008] in CAS development. Instead, we would want to utilize the concept of emergence to reduce the complexity of the analysis, and derive a new set of features, which can better capture the complex behavior and utilized for optimization of varies processes within the system.

In my thesis I consider examples from four different CAS. After the analysis of these

CAS, it was quite surprising to arrive to the conclusion, that the analysis of these systems can be performed by following exactly the same steps. A visual schema of the approach is shown in **Figure 1.2**. I will go over each of the components to provide more details, as well



y
x
Traces of CAS observations
Local statistics
Features + constraints + relaxations
simple cost model
score
Decision making

Figure 1.2: A proposed schema to find approximate solution to CAS

as some common sense reasoning behind them. Since two out of four CAS scenarios resemble NP-hard problems, I will also provide the equivalence between some of the steps with the equivalent techniques from the NP-hard problem solution toolbox.

- **Local system observations**. There are many parameters, which we can measured from CAS. However, quite often CAS are open systems. If we want to analyze buyer behavior on the marketplace, we understand that there many input signals, which come into play, many of which are external ones. In no way we can record every external event which may be relevant to a buyer's decision right at this moment: we need to rely on a smaller observable piece of information. The same is true if we are trying to conclude if a data stream is experiencing a significant change. We do not really need to record every data sample from it: it is enough to periodically to record small samples from it, make summaries from it, and compare, if local stream summaries deviate from other summaries much.
- **Statistics from raw data streams**. We want to use the notion of emergence, mentioned in the previous section, that certain properties of CAS emerge from the local

interactions of its components. Thus, we need to introduce new (derived) features, which capture the cumulative effect of individual contributions. For this purpose, we add all sorts of summarization statistics over the raw data streams, together with the gradients of those statistics. This approach helps to capture stationary properties of CAS, and properties, which emphasize the change in stationary conditions. This is very similar to the concept of "emergent leveling" Wilensky and Resnick [1999] and hierarchy of patterns Kurzweil [2012].

- **System constraints**. This component of the analysis comes directly from constraint programming Arora and Barak [2009] paradigm, and resembles the finite resources inside of a CAS. For instance, when we consider a scenario of fire evacuation in **Chapter 2**, we understand that the building has limited throughput and occupational capacity, and we can move only finite number of people at any given time. Thus, the evacuation strategy must take this aspect into account, if we want to move a big group of people away from the source of fire. When we try to solve the task of optimal resource sharing on a computational cluster in **Chapter 4**, we need to understand that when a cluster is not in use, we can request almost all its resources. However, if the cluster is heavily used, the amount of the resources provided is a fraction of the requested amount. The model must incorporate the notion of constraints: we cannot get more than physically possible.

- **Reasonably simple cost model**. It is clear, that we can introduce a large number of derived features, which can help us to analyze and describe different aspects of CAS behavior. But the challenge we face is to combine all these different signals into one, to answer certain questions. For instance, in **Chapter 3** I show that i need to combine two different properties of stream components, because only a proper combination of those two defines a transition. We do care about one event: transition, and we need one score, which would say, how likely it is to detect a transition. Thus, we can compute statistics on the input stream separately, and then make a weighted sum of those statistics, to get a score.

- **Detecting notable changes and relaxation**. It was shown, for instance, in Miller and Page [2007], that predicting the exact trajectory of CAS development, is the most difficult task in CAS analysis. And often, we do not need and do not care about the exact development of the processes in a CAS, and we relax Arora and Barak [2009] the need for exact process development, and instead pay attention to upper or lower bounds of the problem. For instance, with the fire evacuation we care only to maximize rescuing performance, but not to achieve the best possible optimization for the equivalent of ”supply chain” condition inside the building. When we perform optimization of resource sharing between concurrent jobs on a computational cluster, we do not really care, how many resources ”shadow jobs” receive, as long as they do not extent the total execution time. More in detail I describe this issue in **Chapter 4**. When we analyze online buyer behavior, we do not care about exact sequence of buyer actions. But we do care, whether a buyer made a purchase or not, or whether he resumed his item search activity within the items, he visited in the nearest past.

In **Section 1** I mentioned, that we have two conflicting requirements on how detail state-space we should consider. The list of simplifications techniques, presented in this section, is targeted to mitigate this conflict. For instance, by applying relaxation principle, we equivalently reduce the complexity of the state-space model. Yet, by adding a set of system constraints, we still keep in place the requirement for early detection of critical transitions in CAS, or to build a better cost model.

1.4 RESEARCH QUESTIONS

The question, which I am trying to answer in my thesis, can be formulated close to ”Where do we start, and what do we do when we analyze a CAS?”. I am looking for a recipe, which will be general enough, to be applicable to a variety of CAS analysis scenarios. In **Section 1.3** I presented a method, based on relaxed optimization. The applicability of this method relies on the following questions:

- **Question 1:** Can we analyze global system properties based on observable local interactions?

- **Question 2:** How to capture behavioral patterns that correspond to notable transitions in system dynamics?
- **Question 3:** How to use CAS behavioral patterns to perform relaxed large-scale optimization?

By the end of this thesis I will provide the answers to those questions.

1.5 THESIS OUTLINE

This thesis consists of five chapters, and the results, presented here, were published in Cherniak and Zadorozhny [2010] and Cherniak and Zadorozhny [2013], Cherniak et al. [2013] and Cherniak and Bridgewater [2013]. The first two chapters started as a proof of concept that indeed using the notion of emergence we can derive features, which can capture the complex dynamics of a CAS, and reduce the complexity of the state-space. Also I demonstrate a way how to build a simple cost model, which is sufficient to capture the nature of the processes inside a CAS. The next two chapters is the extension of these ideas, applied to domains with very high number of dimensions. Here the concepts of constrained satisfaction and relaxation are the most articulated. And finally, in the last chapter I present the obtained results from my proposed work, with the strongest point towards automated feature generation, based on the concepts of relaxation. Despite the fact, that certain techniques from the optimization list in **Section 1.3** are most clearly demonstrated for specific domain examples, all of those techniques can be applied to other domains.

The first chapter (**Chapter 2**) is a case study on how to design an application to assist people with fire evacuation from the enclosed spaces. Obviously, human dynamics is very complex during a life-threatening event like fire evacuation, and we cannot model all its aspects. Yet, it appears, that the complexity of the dynamics can be greatly reduced to the task of modeling the most critical parts of the evacuation process, like human behavior very close to fire and very close to the exit. Thus local interactions between people in those critical ares have the highest impact on the fire evacuation process, and the global properties in such CAS can be expressed via the individual interactions between the system agents in the critical points. In this example scenario we demonstrate, how we use principles of locality and relaxation to perform the analysis of a dynamic system. I also show the deign of a simplified cost model, which helps in decision making.

In the first case study we were able to identify those critical points in the functioning of a CAS by collecting and analyzing the logs of the interactions between agents. However, what if we do not have a way to collect the information about local interactions? For instance, we can measure how flu spreads in the population, but we do not really know precisely how human interactions lead to the virus spread. So can we detect the change in the flu spread dynamics only from the total count of causes of flue? Note, that we are looking for transitions in a CAS which happen very rarely, so we cannot have a substantial history of those transitions. I explore this problem in the second part of my thesis (**Chapter 3**). This problem can also be characterized as "black swan detection" Taleb [2010]. Like in biology, we can suspect that black swans exist, but until we get one, we do not really know if it is true and how exactly it looks like. Similarly, if the transitions happen rarely, this does not leave much room for statistical learning of these phenomenons. I propose a solution based on deriving a signature of a transition in a data stream. I obtain this signature by using the principles of locality, when we decompose the time series into a set of individual components, and build statistics from these individual components. Then I show, how I apply a cost function with the tuning parameters to combine different statistics from stream components into a single score, which helps to identify a transition.

In the third part of my thesis in **Chapter 4**, I consider a scenario for CAS optimization, when the number of the optimization parameters is huge. I consider a hadoop cluster with its resources shared between many users. The major objective for every user is to have his task to be computed as fast as possible. This typically implies that each user would want to obtain as many resources as possible for his job(s). Yet, given that many users compete for finite resources, I discovered that task computation acceleration has a saturation point. A hadoop cluster has hundreds of tuning parameters and system settings. However, a cluster is not a deterministic system. Since users submit their tasks, which differ in the input size, at arbitrary time, it is virtually impossible to capture every single aspect of their behavior. Here I propose to use the approach based on the notion of strong emergence in a CAS.

Strong emergence implies that we observe properties in a CAS, which cannot be attributed to the individual properties of the components. Instead of concentrating on low-level system parameters, like CPU load, memory usage, etc, I propose to use higher-order properties of the cluster as a whole. In this example scenario, I show how locality principle (queue scheduling, whole cluster resource allocation) and constraint satisfaction (we cannot obtain more resources than a fixed limit) help to simplify this multi-criterion optimization task. This approach also provides a substantial performance improvement exactly when its needed the most: when cluster reaches its maximum resource usage capacity.

In the last part of my proposal, I consider the task of analysis of buyer behavior on eBay marketplace. This task differs from the previous ones in two ways. At first, it has many more tuning parameters, than any other task considered in the previous chapters. And at second, online shoppers have multiple goals. The social component in this task makes us operate with less defined or less formalized parameters, than what we had in the case of cluster resource optimization. In **Chapter 5** I present the first approach to address the issue of modeling online buyer behavior. The results of this attempt clearly demonstrate the importance on relaxation techniques for this task, when we relax the need for precise modeling. In **Chapter 6**, I present my second approach to the task of buyer behavior modeling. This approach utilizes locality of the data stream decomposition, and automated feature generation with the help of statistics over the local stream components. I show the benefits of automated feature generation, and show the improvements in the obtained results.

In the next chapter, I start with the first example CAS, maybe a bit of a toy example or proof-of-concept scenario. I describe the problem domain, and show how some of the CAS analysis steps from **Section 1.3** emerge in this problem, and can be applied to help in the solution.

2.0 ADAPTIVE SENSOR DATA MANAGEMENT FOR DISTRIBUTED FIRE EVACUATION INFRASTRUCTURE

Aviation industry was probably the first one, which paid attention to the whole complexity of passenger behavior during an emergency evacuation procedure from a commercial plane. Partially this happens because we have a mixture of passengers, who differ in age, body shape, physical and emotional fit. If a less fit person blocks the exit, some people may try to climb over seats, or be pushed away by others. Often in the most narrow parts of the path, they would experience extra competition, when physically stronger people pass over weaker passengers.

We can expect similar competitive behavior in other fire evacuation setups. For instance, when people leave a subway station or a skyscraper on fire. If we would like to help those people with evacuation, how should we handle this very complex mixture of behavior? The solution has to be capable of recognizing different types of behavior and react accordingly in real time. Otherwise, people would simply ignore it and would rely on their own judgement. Another challenge is how to make this solution robust. This system has to function even if being partially damaged by fire, thus it should be fully distributed, and without a central point of information processing. We also need an algorithm, capable to adapt to the change in environment. And this change may come not only from the fire spread and human dynamics, but also from the damages to the system, caused by fire.

CAS description: To address the issue of system survivability, we explored the feasibility of using wireless sensor networks (WSNs) as the building block to address this problem. This architecture does not have a central point of failure. Each wireless node was using the same algorithm, capable of identifying its neighbors, making sense of the environment from its own measurements combined with the measurements, obtained from the neighboring nodes. Our approach is based on considering the WSN as a complex adaptive system where decisions made locally by individual sensors can efficiently converge into desirable information processing patterns. Thus even if the network gets partitioned in the very small pieces, each of those pieces will perform its functionality.

A problem of fire evacuation optimization is similar to supply-chain optimization. This happens, because we have physical constraints, one of which is limited space capacity. We cannot pack unlimited number of people in a limited location. If we want to move people from one place to another (assuming they follow our suggestions), we need to free enough space for them prior to make such move. Supply-chain optimization is an NP-hard problem Arora and Barak [2009], thus we should not expect an easy solution for this problem either.

What we are trying to optimize: There are two major optimization objectives in this problem:

- evacuate as many people as possible, while causing as minimum health damage as possible
- for those people, who cannot be evacuated (due to limited exit throughput, building partitioning or others), direct them to the safest area within the building

The distributed nature of a WSN poses a challenge in the form of communication updates and memory limits. We cannot store the information about the entire network in each wireless node - it will not fit in tiny sensor memory. Also, we cannot send too many network updates, because we risk to overload the network, which can lead to delays in communications, buffer overflow and (potentially critical) message dropping, and premature battery depletion Some sort of approximation is required to address these problems, which I will be talking in the next section.

2.1 APPLYING ROCAS SCHEMA TO THE PROBLEM

Some of the solution steps, discussed in **Section 1.3** can be applied to the task of fire evacuation to make it more tractable. And more importantly, some of the bullet points from that list emerge from this task. Let us go over those points one by one.

- **Statistics from raw observations**. When we look at the problem, the first question which we get is : what are the features here, how should we describe the process? In **Section 2.3** we start with the exact solution, which assumes that all information is known about the environment. We show, that this solution is simply not scalable. After a substantial number of simulations, we observed that only two types of locations inside the building have the strongest influence on the number of the survivors: very close to the fire and very close to the exit. Sub-optimality in decision making for people close to the fire, will cost lives. Less-than-optimal decisions around the exit will slow down evacuation performance. The rest of the building can tolerate sub-optimalities. Thus, instead of keeping precise information about the location of fire, exit, and human density, and perform complex optimization using that information, we can operate with more simple statistics: (distance to fire $\leq$ threshold or not; distance to exit $\leq$ threshold or not?).
- **System constraints** one of the constraints in the setup is the throughput: we cannot evacuate more people, than it is physically possible through the exit for the time, while the exit is still functioning. This allows us to assess the quality of the evacuation performance not as the absolute number of people evacuated through the exit, but as the ratio of those people we rescued to the theoretical maximum of what we could have possibly done. The same reasoning goes towards assessing the performance of life time saving for people who are trapped inside the building. Theoretically you can prolong their lives only till the moment while the last piece of building is not in fire.
- **Relaxation techniques** As I mentioned, one of the challenges in distributed fire evacuation is the need to maintain updates about the environment and human motion in this environment. One of the possible solutions to this issue is to spread regular updates from every sensor in the network. However, this process is very costly: we can flood the network with messages. Moreover, we do not always need to have the latest updates from each sensor. As I mentioned before, we have different tolerance to sub-optimal decisions depending on the distance to fire and exit. Thus, we can divide the environment into different states, and reason about decision making in all those states. The process of evacuation becomes more simple: each sensor can detect which state it belongs to using

some local observations, and adjust the optimality of its actions, based on pre-assigned optimality parameters.

  Since we can be sub-optimal in states, where we are far from the source of fire, and from the exit, we can reduce the frequency of network updates in those states, as long as the total performance does not degrade much. Thus we relax the need for the optimal decision making, as long as the total performance does not suffer.

- **Cost model** in a relaxed state-space, mentioned in the previous bullet point, we can also simplify decision making. Let us consider a couple of constraints, which help to cover corner cases. If a human is very close to the source of fire, the best decision for this person is to go as far as he can from the fire. If the person is very close to the exit, then the best decision is to go right straight to the exit, without any other considerations. For cases in between, some sort of weighted decision making is expected. We can propose a pareto-like cost function, which is a weighted sum of combining of two actions: go away from fire and move towards the exit. And since we can partition the whole space into different states, we can adjust the weights appropriately for each state

The rest of the chapter is organized as follows. **Section 2.2** provides an e-WSN system model for emergency evacuation. **Section 2.3** elaborates on our proposed e-WSN evacuation strategies. **Section 2.4** reports on experimental analysis, demonstrating the efficiency of our approach. **Section 2.5** considers related works, followed by our conclusions in **Section 2.6**.

2.2 BACKGROUND AND SYSTEM MODEL

Each sensor in an *e-WSN* maintains a system model that supports decision-making for emergency evacuation. According to this model, the evacuation space is represented as a 2D space, split into square cells of equal sizes **Figure** 2.1. In a real situation, those cells can correspond to actual locations in a building, or simply zones of responsibilities controlled by sensors. This space is populated with people, and there is at least one source of fire and at least one exit. People are allowed to move from a current cell to its four non-diagonal neighboring cells. Every cell has a wireless sensor ***S*** in it that monitors condition and controls evacuation in that cell (denoted ***cell-S***). System time is discreet: one unit is denoted as one tick. Every sensor ***S*** maintains the information required to make evacuation decisions. Part of this information is received via communications with other sensors in the e-WSN. This information includes the following items:

- ***Occupancy*** – an estimated maximum number of people, which the cell controlled by sensor ***S*** (i.e., ***cell-S***) can accommodate.
- ***Throughput*** – an estimated number of people who will be allowed to move from ***cell-S*** within one tick.
- ***Time_to_ignite*** - time required for the ***cell-S*** to catch on fire, assuming that one non-diagonal neighboring cell is completely on fire.
- ***Hazard_time*** – estimated time before fire reaches ***cell-S***.
- ***Delay*** – estimated time which a person needs to leave the building, starting from ***cell-S***, or to reach another specific point inside the building.
- ***Waiting*** – estimated amount of time a person needs to cross one virtual cell.

It is assumed in the model that each person possesses ***life level*** variable. The value of this variable is initialized with 1. Health impact is caused to people in the vicinity of fire. Thus every time tick *life level* is being reduced on delta, which is equal to health impact. We assume ***health reduction*** to be an exponentially decreasing function of ***hazard_time***:

$$f_{health\_reduction} = e^{-hazard\_time} \tag{2.1}$$

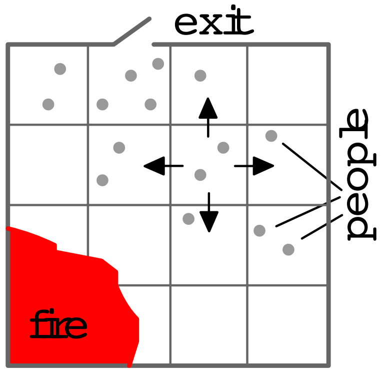


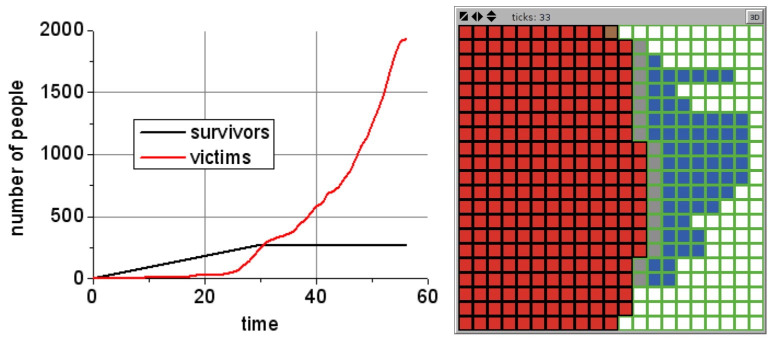
Figure 2.1: 2Dspace

A person dies, when *life level* reaches zero. A major objective of the e-WSN is to avoid reducing life levels as much as possible while evacuating the maximal number of people.

To illustrate this, consider **Figure 2(b)** which shows a snapshot from simulated fire evacuation scenario in a building with one source of fire, one exit, 441 cells, and 2205 people inside. We deliberately set the parameters in a way, that not every person can escape the building. Details of our simulation environment are explained in **Section 2.3**. At the beginning, all people go towards the exit, which is located in the middle of the top border of the 2D space **Figure 2(b)**. The dynamics of evacuation performance is shown in **Figure 2(a)**. When the exit is available, people can leave the building. At time tick 33, fire reaches the exit, and no more people can be evacuated. A snapshot of the environment at time tick #33 is given on **Figure 2(b)**. Red cells correspond to fire; white cells show the location of people. Starting from that point, the only available option for people is to go away from fire to the safest place. There are no more exits left in the system. After time tick # 33, the number of rescued people remains constant and the number of victims increases **Figure 2(a)**.

We use several metrics to estimate the performance of the e-WSN. They include ***normalized rescuing (NR)***, ***life time (LT),*** and ***data cost (DC)***.

Normalized rescuing is the ratio of the number of successfully evacuated people to the maximum number of people who could have been evacuated before fire reaches the exit, given in **Equation 2.2**. The maximum number of people, who could have been evacuated,



(a) (b)

Figure 2.2: Simulation dynamics: (a)rescued and dead people over time; (b)[aaaa]a snapshot of the environment

is equal to the throughput of the exit multiplied by the time before the fire reaches the exit $hazard\_time_{exit}$. This ratio shows how fast the *e-WSN* drives people to the exit, avoiding suboptimal decisions on their way.

$$NR = \frac{people\_evaluated}{throughput_{exit} * hazard\_time_{exit}} \tag{2.2}$$

Life time shows how well the algorithm helped people who were not able to leave the building, to stay alive. *Life time* is the ratio of how long a person stayed alive to the maximum possible time, which is the time when the last cell caught on fire $hazard\_time_{max}$. If D is a set of victims, $|D|$ the number of victims, and $hazard\_time_i$ - time, when the i-th person died, then we express the ratio as **Equation 2.3**:

$$LT = \frac{\sum_{i \in D} hazard\_time_i}{|D| * hazard\_time_{max}} \tag{2.3}$$

Finally, ***data cost*** shows how much information (messages) is exchanged between sensors in the *e-WSN* during the evacuation procedure. Higher *data cost* means higher data processing overhead, network load and congestion, as well as depleting the sensor's battery. In **Section 2.4**, we will provide formal definition of *data cost*.

Our major objective in this thesis is to show how to minimize *data cost* while maintaining the highest possible values of *NR* and *LT*. In the next section, we will introduce our adaptive approach to solve this problem. As a comparison baseline, we also implemented a non-adaptive technique called information diffusion, which we also consider in the next section.

2.3 USING WSN FOR EMERGENCYEVACUATION

2.3.1 Information diffusion

The main idea of the information diffusion ***InD*** approach is to lay out the fastest route to the exit, where the ***health impact*** value is below the given threshold. This method assumes that for the best assessment performance, information about the whole route, or a significant portion of it, should be available to a sensor for decision-making. This information is a part of the global knowledge that should be maintained by the *e-WSN*. The *InD* method uses *health reduction* function as defined in **Equation 2.1**, which shows how the proximity to the fire impacts *life level*. It is obvious that keeping complete information about each possible evacuation route for each sensor in an *e-WSN* does not scale. This is why *InD* maintains the information about the limited number points, which have the most significant impact on human life over a chosen route – so-called critical points. It defines *health reduction* in a critical point as defined in **Equation 2.4**:

$$health\_reduction = f_{threat}(hazard\_time - delay - waiting) \tag{2.4}$$

It means that, in light of current knowledge, fire will reach a critical point in *hazard_time* ticks; from a person's current location, it will take *delay* ticks to reach that critical point. A person will spend *waiting* ticks in the cell, crossing it; thus, he will stay in a cell until $t = hazard\_time - delay - waiting$ ticks remain before fire reaches the cell. **Algorithm 1** outlines the *InD* algorithm with line-by-line explanations. The explanations refer to an example of a fire evacuation route with three critical points as shown in **Figure 2.3**. Because of the brevity of the algorithm schema, here we add some explanations to it:

- for the exit node, we set delay = 0, because if someone is in exit cell, he does not need any extra time to reach this cell. Send this tuple to all neighbor sensors (in our case to S1)
- when we generate own tuple $\{hazard\_time, delay, waiting\}$, we start with the updates from the neighboring nodes, and add local information. Whatever time was needed to reach a particular critical point, a person will need more time, because he needs to cross

```
input  : sensor neighbors,
output: ($hazard\_time, delay, waiting$) tuple for each sensor
begin
    1: for exit node: set ($hazard\_time_{exit}, 0, waiting_{exit}$)
    foreach $sensor \leftarrow network$ do
        1:  receive $\{hazard\_time, delay, waiting\}$ from neighbors
        2:  set $delay_{new} = delay_{old} + waiting_{own}$.
        if the number of tuples in the message is less than the maximal then
            1:  add own tuple to the message
        else
            1:  compute health_reduction value for all the tuples in the message
            2:  find the lowest health_reduction value
            if found value bigger than own health_reduction then
                do nothing
            else
                replace the tuple, which corresponds to this lowest value, with the local
                tuple.
            end
        end
    end
end
```

Algorithm 1: Information diffusion procedure

the current cell. Following the example, S1 after modification of the received tuple, will obtain: $\{hazard\_time_{exit}, waiting_{S1}, waiting_{exit}\}$

- In the example, S1 will add its own tuple, and this is what will be in the result: $\{\{hazard\_time_{exit}, waiting_{S1}, waiting_{exit}\}$,$\{hazardtimeS1, 0, waiting_{S1}\}\}$
- This is what will happen on S2, when it receives a message from S1. Having assumed, that there may be maximum 2 tuples in a message, S2 has to verify, which health reduction (hr) is the lowest: hr(S_{exit}), hr(S1), hr(S 2) . In our example, hr(S_{exit})is the lowest, because S_{exit} is much further away from fire, that S1, S2, S3. Then, S2 will

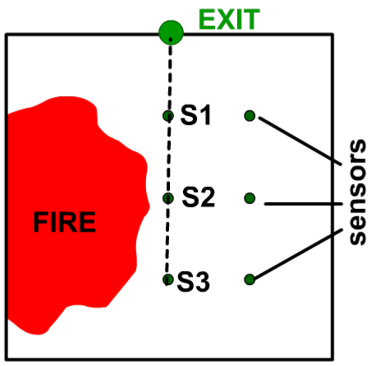


Figure 2.3: Critical points

replace that tuple with its own record. After modification, the message on S2 will be $\{\{hazard\_time_{S1}, waiting_{S2}, waiting_{S1}\}, \{hazard\_time_{S2}, 0, waiting_{S2}\}\}$

For the best route selection, a sensor computes the *health_reduction* value for all of its neighboring cells, and sorts out those with total *health_reduction >=1*. If the resulting set is not empty, it selects the fastest route from the elements in that set. Otherwise, it selects a sensor with the highest value of *hazard_time* as the successor and navigates people towards that cell. This sensor will not send any message to its neighbors, because there is no safe route to the exit, according to the current status.

2.3.2 Adaptive State/Action Strategy

In contrast to the information diffusion approach, the state/action method does not require any global information: all of the actions will be taken based on the local state of a sensor and its direct neighbors. In a state-action algorithm, the whole set of sensors is partitioned into a finite number of states as in **Figure** 2.4 and **Table** 2.1.

A sensor takes those actions, which were assigned to execute in its current state. To justify the state selection in **Figure** 2.4, consider how the influence of the fire depends on the distance between a person and fire. Fire tends to behave stochastically with spikes in spreading. Assuming that spread of the fire follows a log-normal distribution, **Figure** 2.5 shows the probability for a person to be caught by the fire in the next time tick as a function of distance to fire.

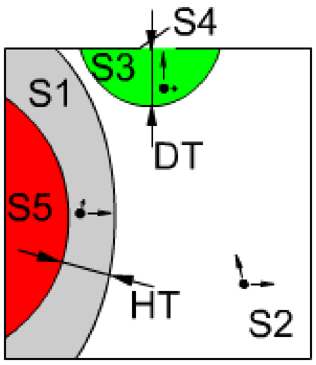


Figure 2.4: State-action visualized

Table 2.1: States definitions and transitions

State	Definition	Transitions
S1	$0 \leq hazardtime \leq HT, delay_{exit} \geq DT$	$S1 \rightarrow S2, S3, S5$
S2	$hazardtime \geq HT, delay_{exit} \geq DT$	$S2 \rightarrow S1, S3$
S3	$delay_{exit} \leq DT$	$S3 \rightarrow S1, S2, S4$
S4	$hazardtime \leq 0$	$S4 \rightarrow x$
S5	$delay_{exit} = 0$	$S5 \rightarrow x$

The plot can be separated into two parts: one with steep slopes and one with more graceful slopes. A suboptimal decision in the steep-slope zone may result in a notable increase of the probability to be captured by the fire. Meanwhile, the graceful-slope zone is much less sensitive to suboptimal decisions. Thus, in the vicinity of the fire, it seems to be a good strategy to go further from the fire and then head towards the exit. In terms of the state diagram in **Figure 2.4**, such behavior would be desirable in state *S1*. The density of people near the exit is expected to be one of the highest in the building. Thus, any suboptimal decisions such as overreacting to fire dynamics when the fire is far away will severely reduce the evacuation performance. A good strategy here would be to be more persistent in the decision to go to the exit. In **Figure 2.4**, this corresponds to state *S3*. People who are far from both the fire and the exit would be willing to consider both the distance to the exit and to the fire in their decision-making. During the evacuation process, sensors may change

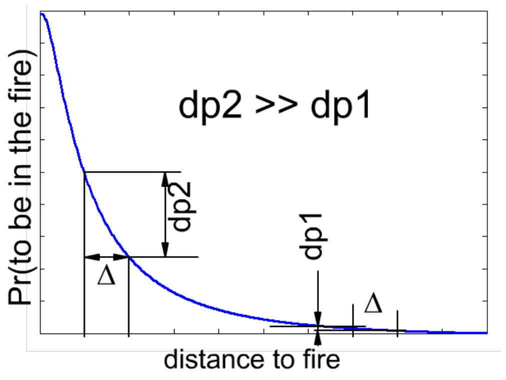


Figure 2.5: Effect of suboptimal decisions

their states, as defined formally in **Table 2.1**. Here HT stands for *hazard_time* threshold, and DT for *distance_to_exit* threshold.

Thus, if a sensor assigns itself to a specific state, this state may change; however, because of the environmental constraints, not every state transition is possible. For example, the system can only transit from S2 to S5 via S1, not directly. Feasible state transitions are shown in **Table 2.1**.

Every sensor uses local information to assess its state and to perform a set of actions. In general, the primary task of a successful evacuation is to take action in order to simultaneously increase *hazard_time* and decrease *delay* to exit. However, in many cases it is impossible to choose a neighboring cell which satisfies both conditions. This is why each sensor uses Pareto-optimality criteria by weighting both the change in *hazard_time* and *delay* values, while choosing the neighboring cell to direct the evacuation. Namely, each sensor estimates a successor cell as follows:

$$successor = argmax_{Si}[\alpha * \Delta hazard\_time + \beta * \Delta delay] \tag{2.5}$$

where $\Delta hazard\_time = hazard\_time_{own} - hazard\_time_{Si}$, $\Delta delay = delay_{Si} - delay_{own}$ and Si - direct neighboring sensors. A Pareto-optimal policy for individual sensor decision-making would correspond to a pair of coefficients $\{\alpha, \beta\}$ for each state.

2.4 EXPERIMENTAL RESULTS

We performed experiments using the Netlogo simulator for complex adaptive systems Netlogo. The evacuation space is set as in 2: it is a square area with 21*21 cells and 2205 people inside. We set one source of fire in the middle of the left edge, and the exit is in the middle of the top edge. All measurements are averaged over 10 different ignition time setups per fire pattern. *Time_to_ignite* parameters are initialized with values taken from the log-normal distribution $LogN(\mu, \sigma)$.

2.4.1 Reduced information updates

Reductions in the number of updates in different states may change rescuing performance, life time, and data cost. Our intention is to reduce the number of update messages, while keeping *NR* and *LT* performance at, or above those information diffusion characteristics. With the reduction in update frequency, sensors will use obsolete information for decision making, thus providing people with suboptimal routes. As we discussed earlier, suboptimal decisions in sensors, which are remote from the fire and the exit (state *S2*), are expected to hurt little the system performance. Meanwhile, reduced update frequency for sensors in the vicinity of the fire (state *S1*) and the exit area (state *S3*) is expected to severely reduce the scores of *NR* and *LT*. **Figure 2.6** shows how performance depends on different update strategies: when update reduction is applied only to state *S2*, or to all states (*S1, S2, and S3*). Setup parameters for this set of tests were taken as the Pareto optimal values for rescuing and life time performance plots in **Figure ??**, while weighting *NR* and *LT* equally important.

For **deterministic fire scenario**, rescuing performance does not change with a decrease in frequency of updates in only *S2*. However, it decreases when the interval between updates increases for states *S1, S2* and *S3*. This means, that even if update reduction makes people go in suboptimal way from the area of state *S2* to state *S3*, there are still enough people near the exit area. Frequent updates in state *S3* help to navigate enough people to the exit, regardless of the supply from *S2*.

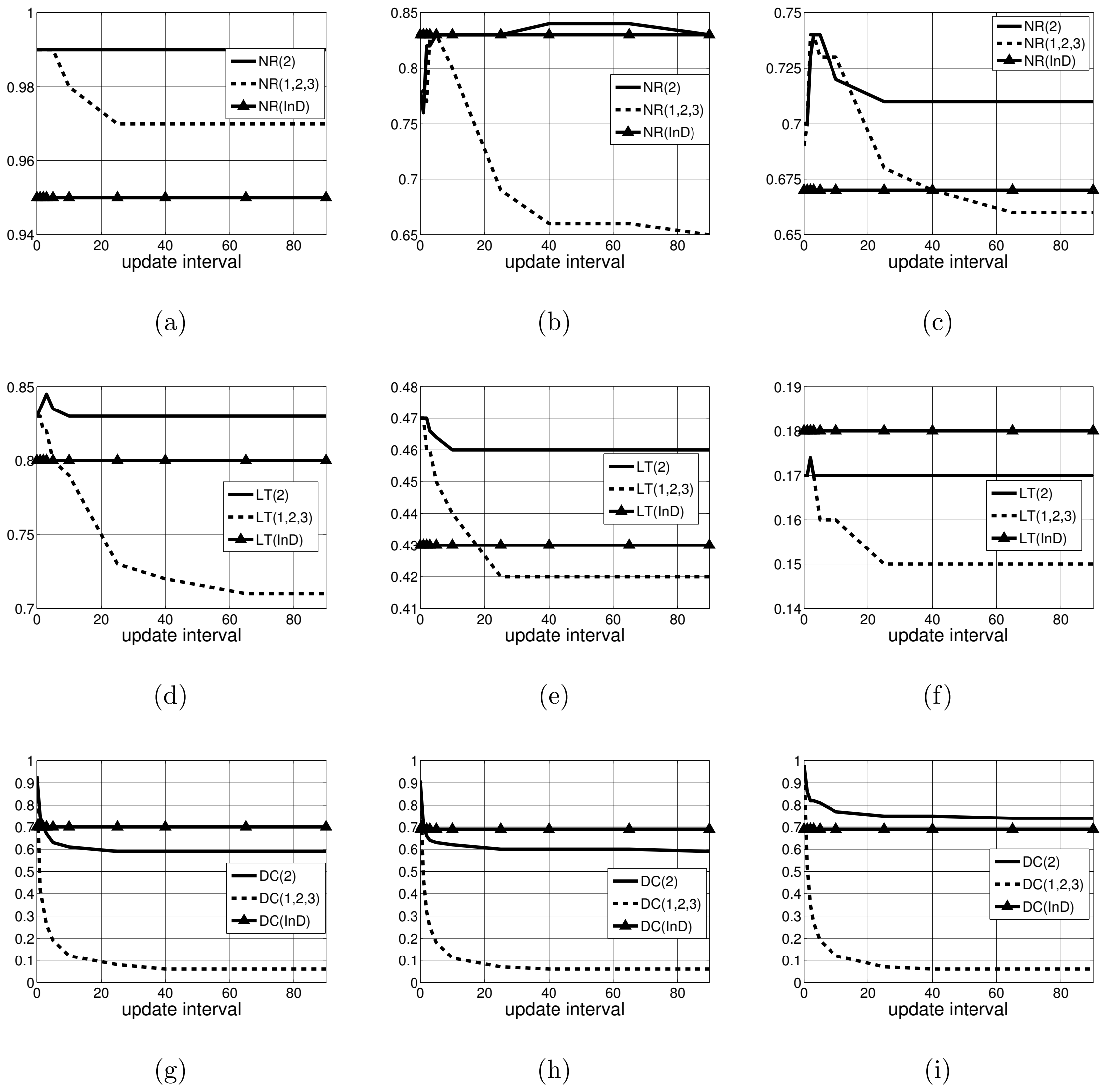


Figure 2.6: The effect of the reduced information updates: in state S2, all states S1, S2, S3, and global knowledge scenario, InD. As a reminder, NR - normalized rescuing performance, LT - life expectancy of the population, DC - network communication cost. Deterministic fire (DF), moderately stochastic (MSF)- , highly stochastic (HSF) (a) NR for DF (b) NR for MSF (c) NR for HSF (d) LT for DF (e) LT for MSF (f) LT for HSF (g) DC for DF (h) DC for MSF (i) DC for HSF

The situation changes when the **fire is moderately stochastic** and **highly stochastic**. With the reduction in the number of updates, rescuing performance even increases for certain update frequencies for both strategies. We may conclude that frequent updates are not always beneficial for rescuing performance. When the update interval keeps increasing, the performance decreases. However, rescuing performance is higher when the update reduction applies only to state *S2*. This confirms our hypothesis about the lack of tolerance of suboptimal decisions in the areas with the highest density of people. The algorithm, in order to provide the best rescuing performance, should react quickly to the changing environment.

Update reduction influences life time performance as well. For a deterministic fire, life time achieves its local maximum at interval = 3 ticks, and gradually decreases for update reduction in *S2*. For update reduction in all states: *S1, S2* and *S3*, it sharply goes down even for slight decrease in frequency of updates. The same tendency is observed for moderately and highly stochastic fires. Life time performance records for update reduction in all three states are significantly below the corresponding records for update reductions for state *S2* only.

There are two major reasons for this type of behavior. First, the data dissemination process is rather simple. It assumes that the time required for a fire to reach a certain place is proportional to the distance between fire and the cell. Meanwhile, that is not always true. If a cell has n neighbors on fire, the resulting time for this cell to become on fire will be less than the average speed of fire. That is why frequent updates help to adjust the difference between what was assumed and what is actually happening in the system. The second reason is the stochastic behavior of fire. If a fire spike happens while there are no updates in the system, the information about hazard time becomes obsolete. Thus, there are sensors which still believe that they should remain in state *S2*; however, they should have changed their state to *S1* and adjust their strategy for decision-making. This lack of information reduces life time. This also confirms the higher importance of information updates in the vicinity to fire.

For reduced information updates in *S2* only, data cost falls below what the information diffusion has for deterministic and moderately stochastic fire. Meanwhile, rescuing and life time performance are maintained at the same or better level. This is not true for a highly stochastic fire. Here, the total number of message exchanges is slightly higher than for information diffusion, and life time has a 0.5 % higher value, which could be treated as roughly equal. Information cost for updates in all three states is significantly below the needs for information diffusion and updates in *S2* only.

The explanation for this result lies in the communication schema. When the update interval is big enough, the only information being sent in the reduced number of updates in states *S1*, *S2*, and *S3*, are the measurements for the average speed calculation in the fire front area. If updates are reduced in state *S2* only, then in states *S1* and *S3* updates are performed at every time tick, which requires sending more data. As shown in 13, for deterministic and moderately stochastic fires, the total area of states *S1* and *S3* remain the same. For highly stochastic fires, the total area of *S1* and *S3* increases, and this results in an increase in the number of communications.

2.4.2 Emergent behavior and multi-factor systems

The pattern of behavior during fire evacuation is formed by multiple factors: local and global. The main global factor is impacted by a producer-consumer paradigm, while the main local one is a micro-crowd formation. Global behavior emerges from the local patterns. In the building, sensors lay out routes to common destinations, such as the exit or the safest place. The form of the route depends on multiple factors; among them, the location of hazards and crowdedness of the environment. Following the local optimality criteria for route formation, the resulting route from a specific part of the building may be narrow or wide. When the route is wide, the interactions between humans are minimal on the way to the destination; thus, they do not obscure the route from each other. On the contrary, when the optimal route is narrow, then people have to compete for it because, in most cases, we can expect that every cell of the route will be totally occupied. This is the **mechanism of micro-crowd formation**.

Frequent updates about current cell delays perform a sort of load balancing by sending people via a wider route, and mitigating the effect of local crowds, as shown in **Figure 2.7**. Thus, frequency of updates changes the speed of human flow in the system.

Frequent updates, re-routing, and micro-crowd formation pose a challenge for state-space model, when we have transitions between very similar states. As a result, when we aggregate

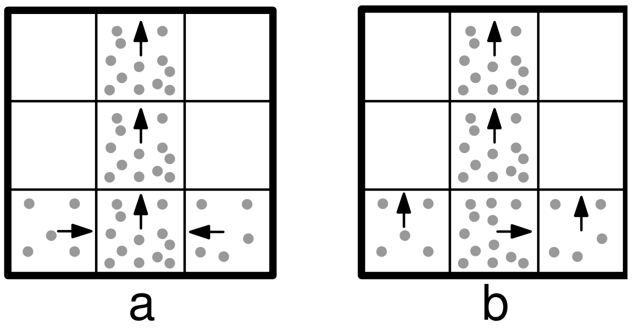


Figure 2.7: Micro crowd formation

state-space in a way, that same type of the behavior corresponds to the same state (our 5-state model), we reduce the complexity of the model.

2.5 RELATED WORK

Researchers have been developing intelligent cost-based strategies for optimizing the data delivery in sensor networks Zheng et al. [2003], Schurgers et al. [2002], Ye et al. [2002], Chen et al. [2002]. In Zheng and Kravets [2003], the authors proposed a cross-layer design for power management that utilized knowledge about the route set-up and packet forwarding. In-network aggregation has also been proposed to save energy by reducing the amount of communications at the expense of extra computation Madden et al. [2002], Younis et al. [2002]. TAG Madden et al. [2002] and Cougar Yao and Gehrke [2002] generate query routing trees similar to relevant work in DTA Zadorozhny et al. [2004]. TiNA Sharaf et al. [2003] is a middleware layer sitting on top of either TAG or Cougar. TiNA employs query semantics (in particular, Quality of Data) and can reduce energy consumption significantly by eliminating redundant transmissions. Prior work in the area of fire evacuation is considerable. The following references are those closely related to the current work.

The FIREGRID project Berry et al. [2005] and Lim et al. [2007] uses real-time data on hazard spread, obtained from a wireless sensor network (WSN), to generate an emergency response scenario. One important weakness of these methods is centralization: data is being processed on a computer directly connected to the WSN. In case of network partitioning, these solutions will not be able to do their job.

The next group of algorithms: Tseng et al. [2006], Pan et al. [2006], Gosalia et al. [2004], Tsunemine et al. [2008], Barnes et al. [2007], and Christakos [2006] establish the core of distributed algorithms for WSNs to assist in disaster evacuation. They lay routes towards evacuation exits, taking into consideration actual fire spread. In addition, Barnes et al. [2007] proposes to predict the dynamics of fire by gathering information from detectors and substituting measured data into the fire spread model. On the contrary, Tsunemine et al. [2008] proposes to monitor the behavior of people to map evacuation routes as safe ones. Mentioned algorithms provide a strategy for evacuation from a building, valid for a single person. Lu et al. [2005] was the first attempt to construct a bridge between computationally expensive optimal methods and naive heuristics by introducing capacity constraints on the routes. It aimed to efficiently generate evacuation scenarios which consider the mutual influence of people. At its heart is the principle of deterministic evacuation: people who are closer to the exit will leave the building earlier; thus, they will introduce delays which other people will face. The main argument against this approach is determinism in behavior: usually people behave non-deterministically, which may be critical at some point and which will require constant route recalculation. The second important point is the absence of fire or any other source of hazard in the model: it does not predict how the optimal schedule should be modified in the presence of a threat.

To the best of our knowledge, none of these algorithms address the issue of having none of exits available for evacuation: every method assumes that people will successfully leave the building. Also proposed are data dissemination schemes such as SPIN Heinzelman et al. [1999] which uses flooding; gradient-based Directed Diffusion Estrin et al. [1999]; clustering-based LEACH Heinzelman et al. [2000], and GAF Xu et al. [2001]. Wave scheduling Trigoni et al. [2004] minimizes packet collisions by carefully scheduling the sensor nodes. It results in energy savings at the expense of increased message latency. Synopsis Diffusion Nath et al. [2004] proposes a multi-path routing scheme, which is more robust than tree-based TAG and avoids message double-counting.

2.6 LESSONS LEARNED

In this chapter we demonstrated how our method efficiently applies to improve the utility of WSNs for fire evacuation. We showed that ROCAS **Section 1.3** approach considerably reduces the amount of delivered sensor data without compromising the stringent application requirements.

However, can we apply this method to other problems? For instance, what if we want to design an algorithm to detect a notable transition in a CAS, when we do not know how exactly this transition will look like? And the only piece of information we know is that this change will separate two distinct system behaviors, and that transitions do not repeat themselves in their exact shape. Let us consider an example time series in **Figure 2.8**, referenced in Beutel et al. [2012]. This plot shows the relative popularity of Web browsers' usage over the course of nine years. We observe, that typically this popularity does not change dramatically, except of two: Firefox and Chrome.

The competition between those two started in 2008, when Chrome was introduced. We observe the dramatic change, when Google Chrome, from being the least popular (and the newest) browser, and gradually suppressed its rival Firefox. As we look back in history, there is not much which we can learn to predict this sort of mutual dynamics. From the available data, we know that this process can happen, like it happen for Firefox in from April 2004 to Jan 2005, **Figure 2.8**, but the dynamics of this process was completely different then for Firefox vs Chrome. Back in 2004, browsers used different versions of DOM model W3C, and the same browser would appear differently in different browsers. Because of this problem, Firefox became a solution, because it started to fully support and follow the W3C standard. In 2009, the change was initiated not with the browser ability to display a web page, but design issues, speed, extra capabilities, and just loyalty to google products. Thus this transition is not repeatable, and cannot be learned from the data. We known changes like this can happen, but we do not know how exactly they will happen.

Another example of this non-repeatability can be observed on the stock market trends, e.g. in **Figure 2.9** for Google and Apple. From the beginning, the business models of those two companies were different, and there stock market price per share had different

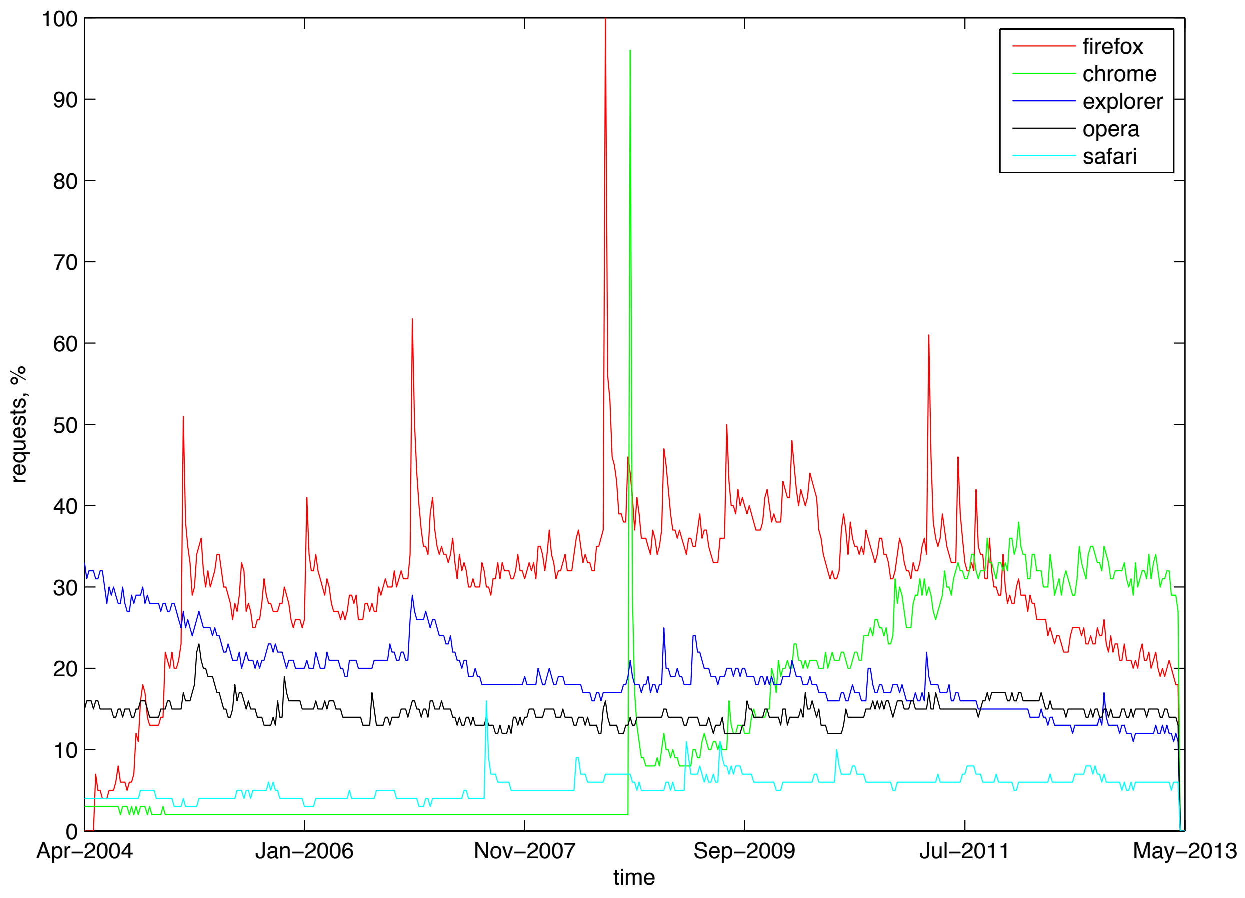


Figure 2.8: Browser usage popularity over time from Google Trends

dynamics. We obviously can see the impact of the global crisis of 2008 on both companies. But beyond that, not much can be said about their mutual evolution. Except, if we zoom in the last six-month period, as shown in **Figure 2.10**, we observe something, which has never happened before. Different analysts use different terms to describe the effect: **Google's rise is a mirror to Apple's decline** BBC or **"Apple and Google stock prices in 2013 look like a zero sum game"** Bloomberg. This happened because two companies became competitors in the same segments, which was not the case before.

To generalize the objective, we would like to design an algorithm to recognize a practically non-repeatable changes in a CAS behavior (aka a black swan Taleb [2010]). Non-repeatability and rareness of the events of our interest makes it difficult to apply statistical learning of these phenomenons. One of the difficulties is that we are looking for extrapolation of the

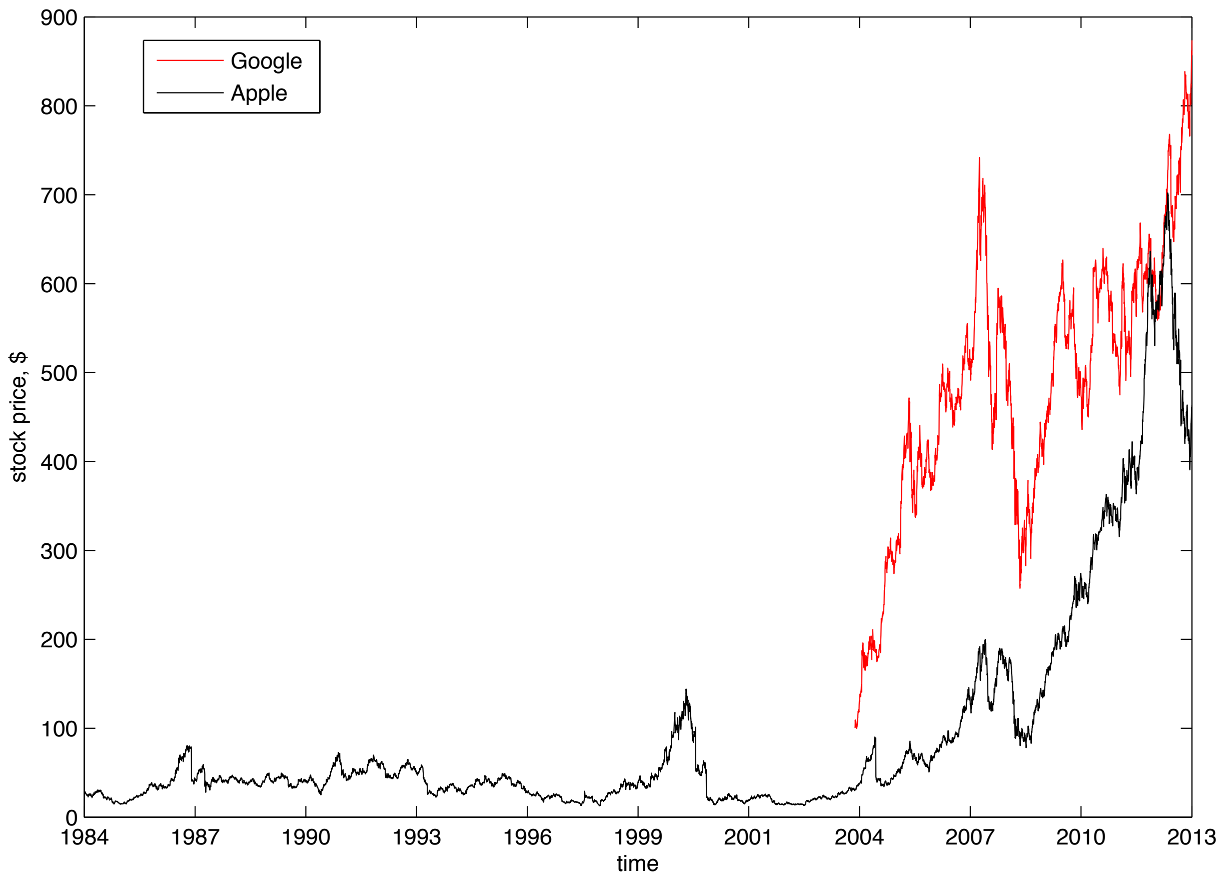


Figure 2.9: Stock market trends: Google vs Apple. From http://finance.yahoo.com

process: we do not know for how long the same type of trend will persist. We know that at some point the trend will change, but when? Moreover, we are not interested to detect an event, which has already happened (epidemics reached its maximum **AND** went down). We want to be able to say that the trend will go down a few steps **BEFORE** it went down. We need to derive a sort of locality property of the change in the dynamics of the process. Is it possible to say, that a notable transition will happen, as we get closer to the transition but before it actually happens? Is there a system coherence before a transition, and can we detect this coherence? Thus, instead of predicting the transition we can detect it on the early stage, before it becomes obvious.

In the next section I derive a signature behavior of a transition in a process. I derive this signature using the principles of "emergent leveling" and hierarchy of patterns, when

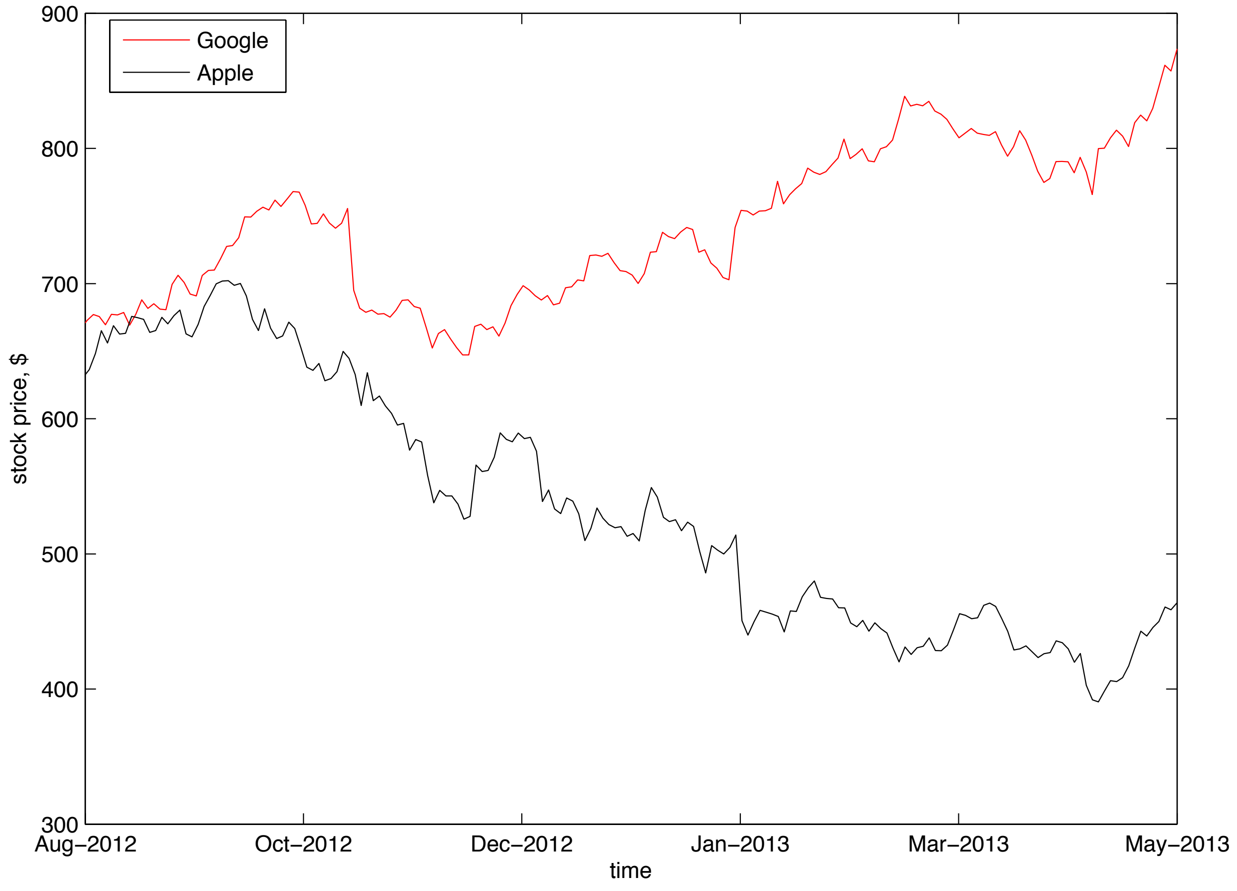


Figure 2.10: Stock market trends: Google vs Apple (rivalry)

we decompose the time series into a set of patterns of different level of hierarchy and look at the properties of the obtained hierarchy. As we do not know how a transition should look in general, it is not clear how pattern hierarchy should look for a transition. Essentially we will find it when we see it. Thus, I employ the principle of emergence into the pattern detection. Since a transition in a process is a coherent behavior, I introduce the coherency measure into the transition detection, and compute the coherency in behavior in the pattern hierarchy. Thus, using the principles of emergent behavior and hierarchy of patterns, i derive a simplified set of rules of how to detect a transition, when we do not know a-priori what kind of transition to expect.

3.0 DETECTING NOTABLE TRANSITIONS IN NUMERICAL DATA STREAMS

A major challenge in large-scale process monitoring is to recognize significant transitions in the process conditions and to distinguish them from random fluctuations that do not produce a notable change in the process dynamics. Such transitions should be recognized at the early stages of their development using a minimal "snapshot" of the observable process log. We developed a novel approach to detect notable transitions based on analysis of coherent behavior of frequency components in the process log (coherency portraits). We have found that notable transitions in the process dynamics are characterized by unique coherency portraits, which are also invariant with respect to random process fluctuations. Our experimental study demonstrates significant efficiency of our approach as compared to traditional change detection techniques.

3.1 INTRODUCTION

Large-scale process monitoring requires continuous assessment of the stability of that process. Examples vary from detection of developing epidemics in disease dynamics to real-time structural health monitoring. The ultimate goal here is to recognize major transitions in the process conditions and to distinguish them from random fluctuations that do not produce a notable change in the process dynamics. It is desirable to recognize such transitions at early stages of their development using a minimal "snapshot" of the observable process log.

If there is a system model of the process, then the task of detecting notable transitions becomes relatively straightforward. However, in many real-life scenarios, obtaining an adequate process model is not realistic due to the complexity of the observable phenomena. In such cases, the process log itself could be used to identify notable process transitions. At their initial stages, such transitions can be obscured by the massive amount of noisy measurements and often become obvious only when it is too late. In many cases, *post-fact-um* transition detection of notable changes in process dynamics is straightforward. However, it is not a trivial task to detect such changes in real-time when only a small part of the process log is available.

Our major contribution in this work is a novel approach to notable transition detection based on analysis of coherent behavior of frequency components in the process log (coherency portraits). We assume that the process log consists of various numeric parameters recorded periodically according to a process monitoring policy. We formalize our task as a problem of transition detection in time series. We have found that notable transitions in the process dynamics are characterized by unique coherency portraits, which are also invariant with respect to the random process fluctuations. In this work, we consider data streams consisting of a mixture of stable zones and transitions. We are especially interested in those data streams for which the process behavior at the initial stage of the transition is similar to a process fluctuation in a stable zone. As an example, consider the sample time series in **Figure 1(a)** and **Figure 1(b)** that were generated using a virus propagation model from Prakash et al. [2010], which we briefly explain below.

Assume that there is a network of randomly-connected computers; some of those nodes are infected with a virus. If two nodes are connected and one of them is infected, the virus will spread to the other computer with a probability of β. Each infected computer will be disinfected with the probability of δ. If we initially infect a few computers and leave the network unattended, at some point in time the number of infected computers will stabilize. For our example, we will distinguish between two states: epidemy, when the majority of nodes become infected, and healthy state, when the number of infected nodes drops exponentially over time.

We generated **Figure 1(a)** and **Figure 1(b)** by counting the number of infected computers over time with "days" followed by "nights" (separated with vertical red lines). We tuned the model parameters in such a way that epidemy develops during the "day" time,

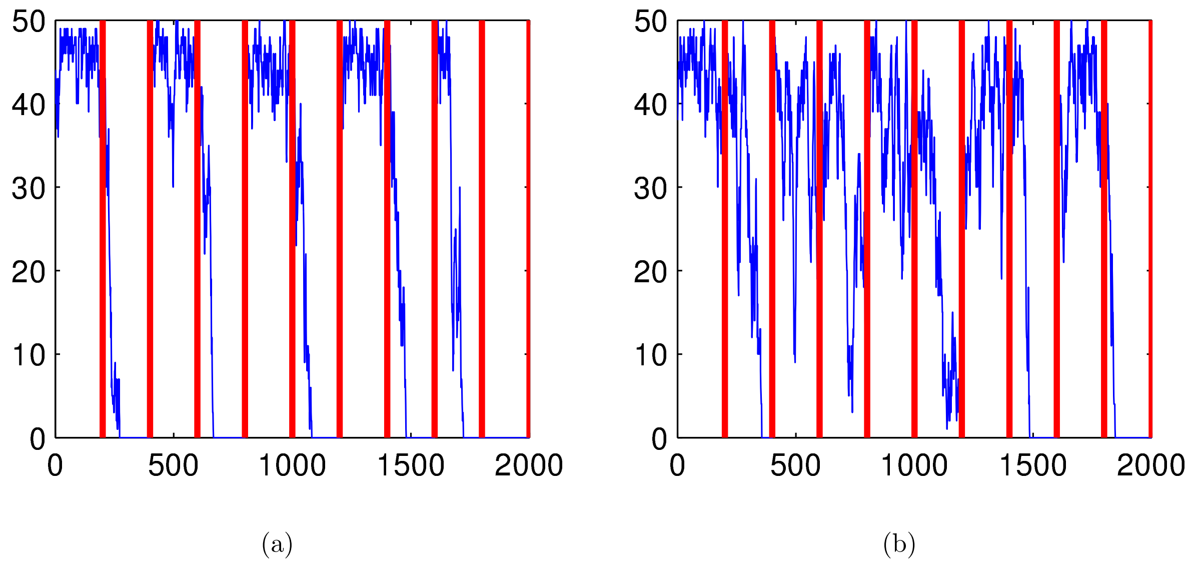


Figure 3.1: Process dynamics with transitions: (a) rapid transition; (b) slow transition

while healing occurs at night. We considered two distinct scenarios reflected in **Figure 1(b)** and in **Figure 1(a)**. In time series shown in **Figure 1(b)** transitions do not develop as rapidly as in **Figure 1(a)**. They can also halt at early stages and do not result in notable changes.

Both data streams in **Figure 1(a)** and **Figure 1(b)** demonstrate volatile behavior with notable transitions between different stable states and outliers. Since in this example we generated the data stream using an *a priori* process model, we can easily distinguish notable transitions from outliers. Meanwhile, recognizing this difference in observable data stream is challenging. As the process switches between epidemy and healing, the data stream does not immediately reflect that change. For example, in **Figure 1(b)** a transition started at time x = 200; however, the waveform does not show clear visual signs of the transition until approximately x = 300. A piece of the time series in $x \in [200..300]$ can be treated as local outliers. Meanwhile, the process dynamics changed significantly at time x = 200.

In this chapter, we propose an approach to detect notable transitions at early stages of their development. We show, that we can detect the onset of the transition earlier than the traditional techniques. At the heart of this approach, is the assumption that the mutual alignment of the stream components are more informative in predicting a transition than the shape of the stream itself. For instance, a transition can happen, when stream components, without changing in their amplitudes, come out of phase synchronization.

CAS description: we apply a bunch of band-pass filters to the data stream and obtain series of harmonic signals, and trying to understand the order from the chaos of mutual alignment of those components.

What we are trying to optimize: our working hypothesis is that the mutual alignment of the components is the key to detect the onset of the transitions. However, we do understand that this alignment can be arbitrary complex. We are looking to derive a model, which will allow us to reduce the dimensionality of the problem (mutual alignment), and tune the parameters in the way, that we can detect the transition at the earliest stage.

We begin the description with how ROCAS techniques from **Section 1.3** can be applied for the task of notable transition detection.

3.2 APPLYING ROCAS SCHEMA TO THE PROBLEM

In this section, I will show how ROCAS schema from **Chapter 1.3** can be applied for this task.

- **Statistics from raw data stream:** we work under the assumption, that mutual component alignment can help detect transitions. Thus, we apply bandpass filtering to the input data stream to single out individual harmonic components. Then we locate local extrema on these components, and draw epsilon-areas around those extrema locations. This input transformation helps with the following analysis.
- **Local observations:** following the assumption, we are looking for the mutual alignment of stream components. After the stream transformation into epsilon-areas, we are looking at the mutual coverage of those areas, or coherency portrait, as we call it later in the chapter. If this epsilon-area is very small, we may not see any overlap between components. If we keep epsilon-area too wide, we will observe arbitrary complex coverage between different components. More importantly, as we increase the epsilon-area,

we may loose the locality information: we may start combining alignment of extrema, which are not related. Running a bit ahead of time, **Figure 3(a)** shows that exterma overlap resembles curves. We want to keep epsilon-interval big enough to have meaningful overlap, but small enough, to make sure obtained curves do not branch into more complex structures. In **Figure 3(a)**, extrema overlap resembles smooth curves around the transition, and noise-like structures outside the transition. We have to make sure that we preserve locality when combine stream components, and do not mix unrelated component coverage together.

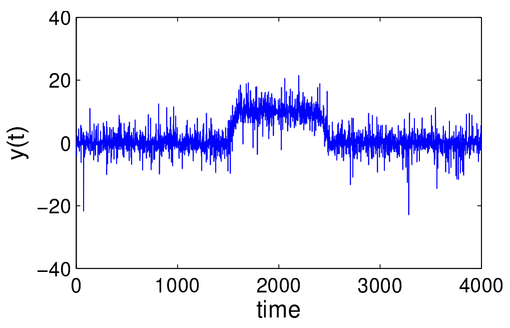
- **System constraints :** we do understand, that the shape of the component alignment can be very complex, and we want to derive some simplified model to deal with this complexity. At the core idea, that if we are to detect a transition, our method surely has to detect the abrupt change. Thus, we present a few abrupt transitions to the method, and record the most distinct features in the coherency portrait around the transition and away from the transition. And then present less abrupt transitions, and record, which features have changed the least, and use those as the signature.
- **Reasonably simple cost model:** As we observed in **Figure 3(a)**, and later proved in **Appendix A**, coherency portrait around transitions resemble smooth curves. To combine "smooth" and "curvy" properties of the portrait, we proposed a score function, which is a weighted sum of two components, which individually describe both attributes. We showed, that such score function is sufficient to perform detection of a transition.

3.3 RELATED WORK

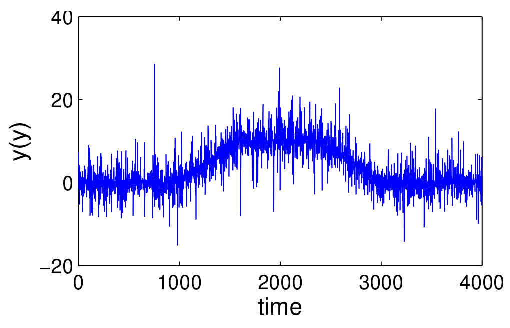
The ideal method for change / transition detection should notice the transition as early as possible (having as few data points sampled from the transition area as possible). At the same time, it should demonstrate high efficiency: if the method indicates that the transition has started it should be confident about it. We will analyze existing methods from the point of view of those requirements, comparing them to our proposed method.

It is a well-known fact that frequency components of a signal posses higher energy at steep signal transitions. In general, Parseval's theorem Weisstein [d] establishes energy equivalence calculated for a data stream $x(t)$ and its Fourier Transform $X(f)$ Oppenheim and Schafer [2009]. Specific dynamics near sharp transitions in a signal is reflected in *ringing artifacts* and *Gibbs phenomenon* as an "overshoot" in the convergence of the partial sums of certain Fourier series in the neighborhood of a a signal discontinuity Weisstein [b]. In principle, we could use a frequency energy diagram to recognize abrupt transitions in a time-series. In order to obtain such diagram, we could use either *Discrete Fourier Transform (DFT)* Oppenheim and Schafer [2009] or the family of *wavelet transforms* Chui [1992]. An implementation of the latter approach, though in a different context, is reported in Prakash et al. [2009]. The authors used *Haar wavelet transform* to develop an automated *Border Gateway Protocol (BGP)* update analysis tool based on scalograms. The scalograms plot the absolute values of the wavelet coefficients in the scale-space domain. The high-energy areas on a scalogram are reported as dark "funnel" (or inverted "tornado") shapes corresponding to spikes in the original time-series (e.g., BGP updates). **Figure 2(c)** shows a scalogram corresponding to a time-series with a steep transition **Figure 2(a)**. The vertical axis is scale, with coarser scales corresponding to higher frequencies at the top. We observe two dark tornadoes on the scalogram that correspond to the transition area. The figure also reveals a cumulative energy plot **Figure 2(e)** corresponding to the scalogram with two distinct energy spikes reflecting the signal transition. In general, this approach does not work well for smoother transitions (e.g. **Figure 2(b)**. The corresponding scalogram in **Figure 2(d)** becomes fuzzier and the cumulative energy plot **Figure 2(f)** cannot be used as a transition pattern anymore.

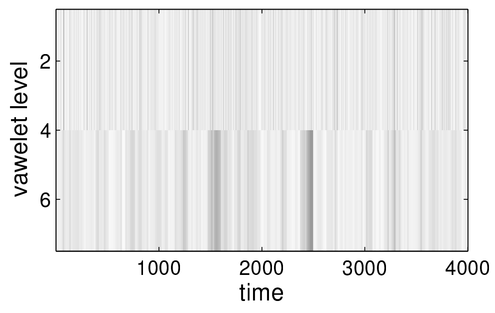
Another group of methods employs machine-learning techniques to ***build a process***



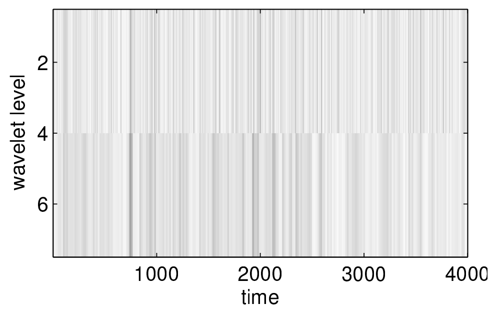
(a)



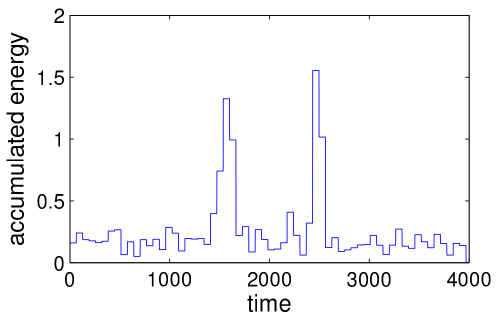
(b)



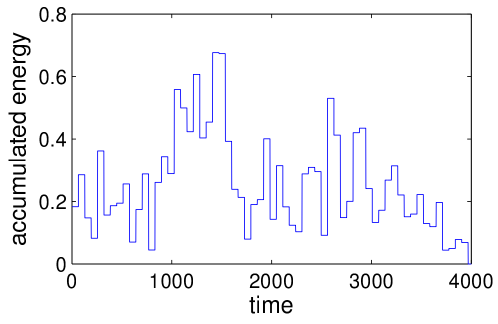
(c)



(d)



(e)



(f)

Figure 3.2: Energy-based vs signature transition detection (a) steep signal (b) smooth signal (c) scalogram for steep signal (d) scalogram for smooth signal (e) steep signal energy plot (f) smooth signal energy plot

model from the observed data streams. We can group them in the following categories:

Global models: Gu and Wang [2009], Chen et al. [2008], Wang et al. [2011], Borges and Levene [2007]. Global models are trained on data samples large enough to be representatives for the whole process. Assuming that new streams are generated from the same process, the obtained model can be used to detect or predict a transition. However, these methods fails when the model is not trained on all possible observations Xiang et al. [2010], which reduce their applicability. For instance, while building a model for a D.O.S. attack on a production computational clusters, like in Chen et al. [2008], we cannot guarantee that all possible scenarios have been observed.

Pattern mining. Pattern mining methods utilize sequential patterns in data streams Bayardo [1998], Han et al. [2000], Chiu et al. [2003], Chandramouli et al. [2010], Agrawal et al. [2008], Floratou et al. [2011] associated with a transition. We can index time series with and without transitions and store them in a database. Detecting a transition in a data stream becomes a task to find similar time series in the database Faloutsos et al. [1997], Fu et al. [2008], Perng et al. [2000], Keogh [2002]. Thus, transition detection applies event stream processing methods Wu et al. [2006] to find a particular sequence of events associated with a transition (transition signature).

Short-term or local models Wang et al. [2003], Yang et al. [2005], Severo and Gama [2010], Krishnamurthy et al. [2003], Bifet and Gavalda [2007]. These methods sample sequential blocks of data from the stream to build process models and use the obtained models to predict the process behavior. Prediction error is used to recognize a transition Brewer et al. [2008] Anderssen and Bloomfield [1972]. When the transition occurs, the old model can no longer predict the new data and therefore the error will be high. While these methods are easy to implement, they have considerable performance issues. It may be difficult to extrapolate an adequate model and large prediction error may reflect the model inaccuracy.

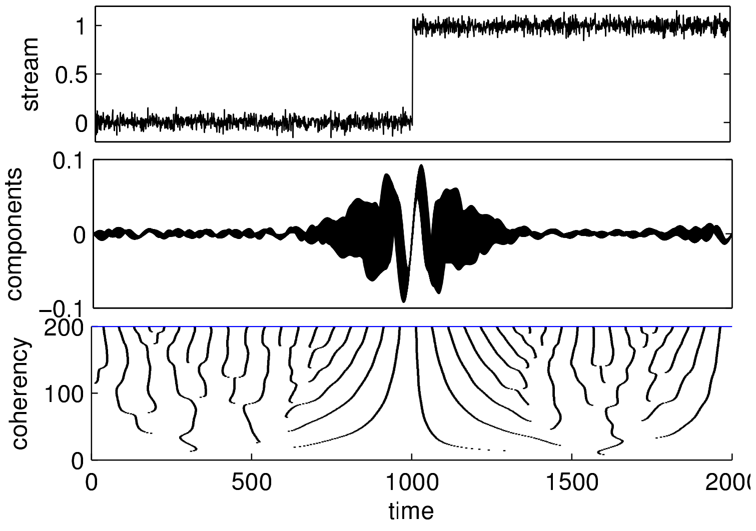
Anomaly and novelty detection. These methods aim to build a steady-state process model to find transition points using anomaly detection procedures Chandola et al. [2009], Neil and Wong [2009], Moore et al. [2002], Noble and Cook [2003], Neill [2009]. A sub-class of anomaly detection methods - novelty detection Hoffmann [2007], Markou and Singh [2003], Dasgupta and Forrest [1996], Spinosa et al. [2009], Fujimaki et al. [2005], Gazen et al. [2005], - assume that a transition or a switching point indicates something new. Thus, the novelties in the data streams can be interpreted as transitions.

Change detection. This group of methods explore change in local properties of the time series. A significant change indicates a transition. This change can be measured using distance between clusters in the feature space Ponte and Croft [1997], Allan et al. [1998] and other types of non-parametric statistical tests, such as density ratio estimation Kawahara and Sugiyama [2012], Kawahara and Sugiyama [2009], Yamanishi et al. [2004], KL-divergence Steinert and Gillblad [2010], Dasu et al. [2009]. Hypothesis testing Muthukrishnan et al. [2007] can be used to verify whether previously observed data is statistically different from the current measurements Kifer et al. [2004] Takeuchi and Yamanishi [2006]. Later we will show that methods based on KL-divergence are essentially equivalent to log-likelihood ratio test. While these methods perform well with multiple numerical streams, they do not perform well for transition detection in data streams with complex waveform. In this case, the onset of the transition remains indistinguishable from the stream lacking a transition until a later point in time.

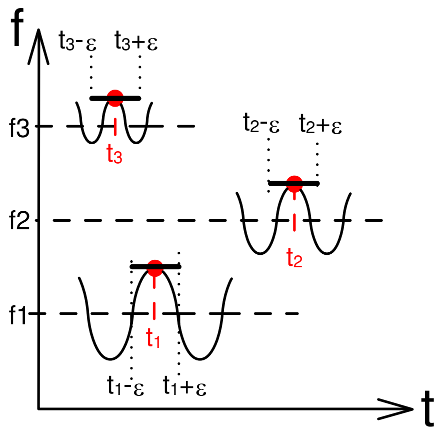
Several approaches, e.g., Dasu et al. [2009], Bifet and Gavalda [2007], Krishnamurthy et al. [2003], Muthukrishnan et al. [2007] explore optimization strategies to speed up computations or reduce memory usage for transition detection. All of those methods are based on techniques considered above, such as KL-divergence, hypothesis testing and short-term modeling.

Two other solutions related to our work are reported in Aggarwal [2003] and Li et al. [2010]. Both of those methods transform original time series into a set of features and apply transition detection techniques in the feature space. Aggarwal [2003] uses velocity density estimation to transform the original stream and to locate areas of **global data shift**, which are essentially notable diffusions in measurements density. Li et al. [2010] decomposes original time series into a set of harmonics and detects areas with different harmonics mix. In contrast to Li et al. [2010], we explore coherent behavior of frequency components (mutual alignment), not just the strength of presence of particular harmonics. Li et al. [2010] was shown to detect transitions on periodic-like time series. Li et al. [2010] was used to cluster burst-like time series (BGP updates); it was not reported its effectivenes to detect transitions in burst-like time series.

We propose a novel approach to transform the raw time series into its feature space and to look for the characteristic pattern (signature) of a notable transition. Our method is based on analysis of coherent behavior of frequency components in the process log (coherency



(a)



(b)

Figure 3.3: Data stream representation: (a) coherency portrait (b) frequency voting

portrait). Notable process transitions can be characterized by unique coherency portraits. Such portraits are invariant with respect to the random process fluctuations. We consider our method in the next section.

3.4 METHOD DESCRIPTION

The central idea of our approach is to recognize a transition in a noisy numerical stream by decomposing the initial stream into adjacent frequency bands and analyzing mutual alignment of the obtained components (we will also refer to them as stream components). Note, that we do not use binary presence / absence of a particular harmonic as a fingerprint for the stream (e.g., as in Li et al. [2010]). Instead, we explore the mutual alignments of stream components at specific time points in order to recognize patterns of transitions. The alignment of the stream components at different frequency bands makes mutual maximum and minimum co-occurrence amplify or re-compensate each other.

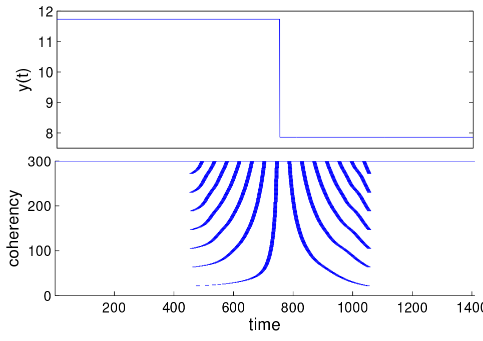
Consider a few adjacent frequency bands from a step function with Gaussian noise, as shown in **Figure 3(a) upper plot**. We use a zero-phase digital filter of order N to cut frequency bands from the original stream. As we see in **Figure 3(a) middle plot**, frequency components are aligned quite arbitrarily everywhere except for the area where the transition occurs. We observe that stream components form a specific alignment pattern at the transition segment, which is different from the surrounding area. To explore the mutual alignment of the stream components we perform the following feature selection procedure. First, we filter the stream with a set of narrow-frequency adjacent band-pass filters. Then, we locate maxima of the resulting stream components and build ϵ-intervals around each maximum, as shown in **Figure 3(b)**. In this example, we select 3 adjacent frequency bands, locate maxima on all of the bands, and mark ϵ-area around each maximum. In the example, for 3 different maxima: t_1, t_2, t_3, we obtained intervals: $[t_1 - \epsilon, t_1 + \epsilon]$, $[t_2 - \epsilon, t_2 + \epsilon]$, $[t_3 - \epsilon, t_3 + \epsilon]$. In this way, we changed the representation of the numerical stream from its wave form into extrema alignment – what we call a **coherency portrait**. If we apply the same procedure to the numerical stream shown in **Figure 3(a) upper plot**, we obtain its coherency portrait in **Figure 3(a) lower plot**. A coherency portrait helps to visualize the fact that stream components are aligned rather sporadically at a distance from the transition, while they form a more coherent structure around the transition point. Our working hypothesis is that there exists a correspondence between frequency alignment and the process dynamics. Detecting a notable transition is equivalent to finding a specific pattern in the coherency portrait: a signature of a notable transition in the process dynamics.

3.4.1 Frequency signature invariant

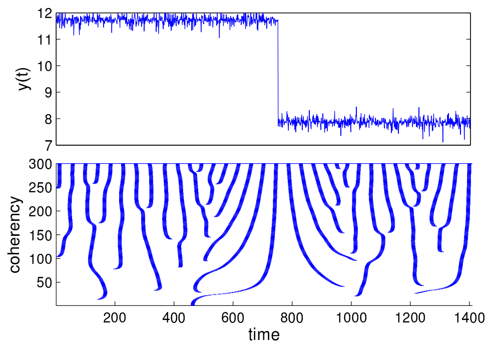
To detect the transition, we have to define an invariant signature pattern in the coherency portrait that occurs every time the transition occurs. Ideally, this signature should remain stable for

1. random noisy fluctuations in the numerical data streams
2. partially available information around the transition time.
3. different shapes of the transition (i.e., steep vs smooth transitions)

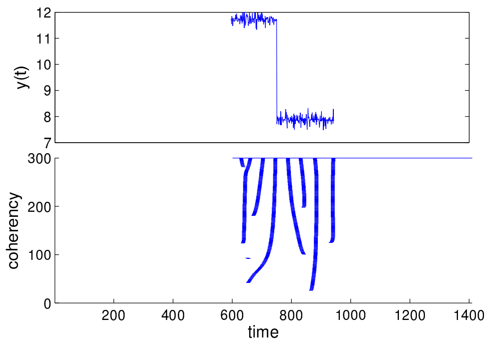
We generate a number of different time series, generate their corresponding coherency portraits and try to visually identify the invariant signature. We start with a simple step



(a)



(b)

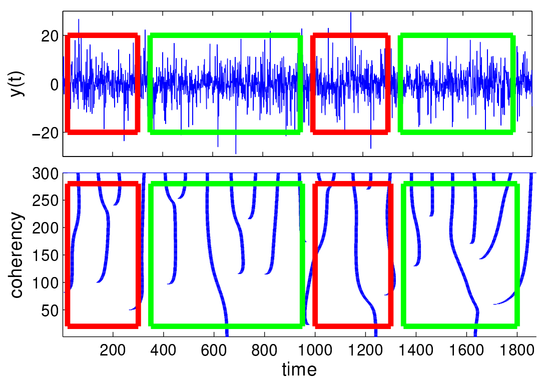


(c)

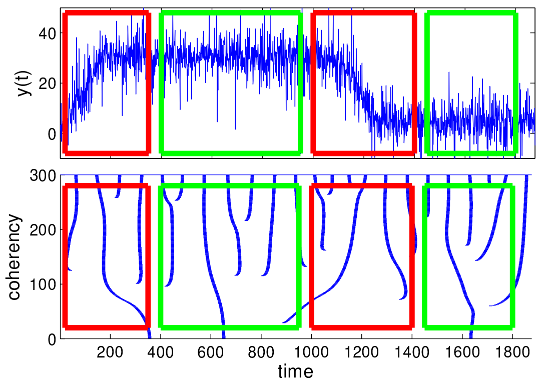
Figure 3.4: Data stream coherency portrait : (a) without noise (b) with noise (c) partial information

function and its coherency portrait as shown in **Figure 4(a)**. For a step function around the transition point, its coherency portrait will consist of a set of rather smooth curves and will resemble a group of inverted letters V. Then, we add a small amount of exponential noise to it, and build a coherency portrait for that data stream as shown in **Figure 4(b)**. Exponential noise makes the task of transition detection more challenging: a rare large fluctuation may be confused with the beginning of a transition. We observe that the inverted V shape remains around the transition area, and the actual curves that form that V-shaped pattern remain rather smooth.

To ensure that an inverted V-shape on a coherency portrait (caused by a transition) is preserved in a partial process log, we separate a segment from the original time series around the transition point in **Figure 4(b)** and build a frequency portrait for this partial log as



(a)



(b)

Figure 3.5: Coherency portrait for: (a) noise only (b) transition and noise

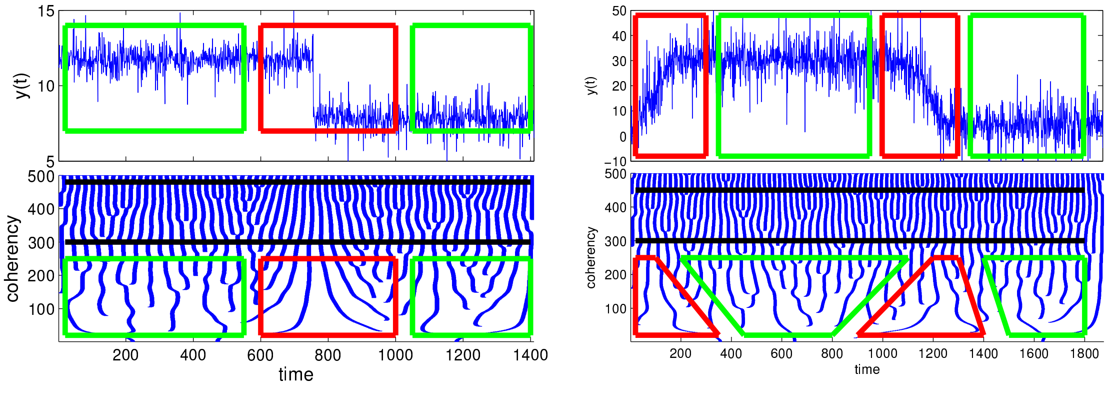
shown in **Figure 4(c)**. We can observe that the pattern around the transition point still preserves the smooth inverted V-shaped curves.

In the next set of experiments, we build frequency patterns for those times series with smoother transitions. We generate two noisy data streams: one without the transition in **Figure 5(a)** and one with the transition in **Figure 5(b)**. When we compare the obtained coherency portraits, we see that coherency portraits for stable areas on these time series are identical (circled with green). However, the coherency portraits are considerably different in the areas around transition points (circled with red).

In order to use coherency portraits to detect transitions, we have to define features which will indicate the transition. We increase the number of frequency bands (in comparison with **Figure 5(b)** and **Figure 4(b)**) to provide more information about the data stream. We increased the number of bands for data streams with both abrupt **Figure 6(a)** and smooth **Figure 6(b)** transitions. For our streams in **Figure 6(a)** and **Figure 6(b)**, we observe that higher frequency components (circled in black) show quite complex noisy-like behavior which is difficult to interpret and analyze. Meanwhile, lower-frequency components appear to be sensitive to the transition and they can be used as a signature for a transition. Considering patterns circled in red in **Figure 6(a)** and **Figure 6(b)** together with **Figure 4(c)** and **Figure 4(b)**, we conclude that a particular V-shape can be observed in all cases: in the presence of noise, at different stages of the transition and when only partial information about the transition is available. Another observation is that for noisy data streams near the transition area, the curves appear to be smoother than those far from the transition. Thus, our proposed method for transition detection is to **look for a smooth inverted V-shape pattern on the frequency diagram in the lower frequency range**. In **Appendix A** we present a more formal justification of the V-shape transition signature invariant.

3.4.2 Pattern discovery

Next we consider an approach to discover those smooth V-shape patterns in a coherency portrait which correspond to the signature of a transition in the actual time series. In the classical approach to pattern recognition, we have to select a set of features from our pattern which will maximize the distance between normal (areas without transition) and abnormal (transition) clusters in the feature space Duda et al. [2000]. If clusters are not linearly separable, a common remedy is to use *features of features* Scholkopf and Smola [2001] to perform non-linear feature space transformation to make clusters linearly separable in the new space. While this is a common method for pattern classification, it may be problematic to apply this approach to our task. First, space transformation requires learning to find the optimal kernel function. Given that we cannot generate all possible shapes of transitions



(a) (b)

Figure 3.6: Coherency portrait for: (a) noisy step function (b) noisy incline transition

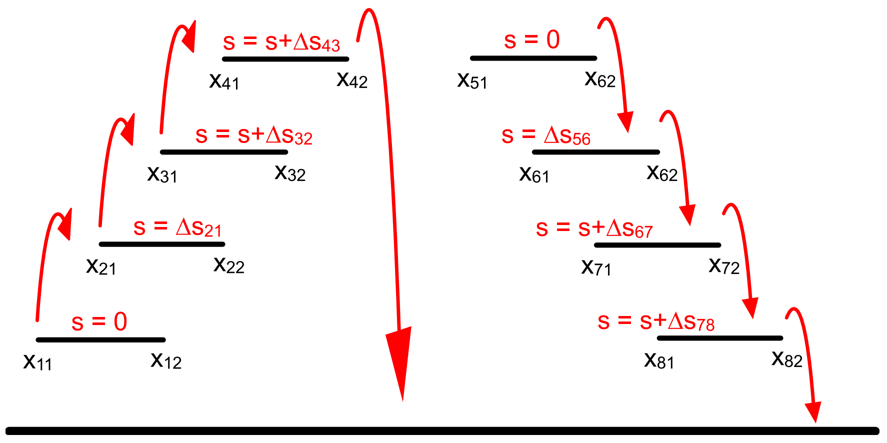


Figure 3.7: Frequency pattern and its score

and their corresponding coherency portraits, we cannot apply it in our case. Second, the kernel trick Scholkopf and Smola [2001] does not allow us to explain the interactions between features of our pattern; nor does it allow us to understand the meaning of the pattern and hopefully provide feature space reduction and pattern simplification. Therefore, we decided to approach the problem of pattern recognition from the perspective of *Complex Adaptive Systems* Miller and Page [2007]. We define the global pattern we want to detect, and derive a set of simple conditions which, if applied, would manifest our desired pattern. We split the pattern detection into three parts: **1)** for each curve, estimate how well it resembles a part of a V-shape; **2)** calculate how smooth a particular curve is; and **3)** make a decision based on those two parameters.

From **Figure 6(a)** and **Figure 6(b)** we visually separate two types of alignment of frequency components: “curved up” and “curved down”. To detect these patterns, we utilize an agent climbing the ”stairs” of the coherency portrait. **Figure 3.7** illustrates our approach. As the agent moves, it reports a score as a measure of how well the frequency pattern exhibits the properties of the desired V-shape. Assume that every ϵ-interval around each maximum is an individual step. Initially, an agent starts from the lowest available step at a current location with a zero score. In our example in **Figure 3.7**, the initial starting point would be x_{11}. If an agent remains at the same level it does not accumulate any score. An agent climbs one step up if there is an overlapping step on the higher adjacent frequency, and thus increases its score. In our example on **Figure 3.7**, that situation occurs when an agent reaches coordinate x_{21}. Then, it will climb up to the second step and increase its score. The same happens when an agent reaches x_{31} and x_{41}. An agent "falls" to the ground level and loses all of its score if the current "step" ends and there is no other higher adjacent steps. In our example on **Figure 3.7**, this will happen when an agent reaches coordinate x_{42}.

For "steps down," the procedure is reversed. An agent starts from the highest available step, say at x_{51}, and remains on the same level or climbs down to the lower adjacent frequency while accumulating a score. In our example, an agent steps down when it reaches x_{61}, x_{71}, x_{81}. It falls to the ground level and loses its score when there is no step on the adjacent lower frequency.

3.4.2.1 Score function Now we introduce a score function used by the agents to explore the alignment patterns. We calculate the score between "steps" on adjacent frequencies, without looking at the global shape of the curve. In **Figure 3.7**, let us denote $\Delta_{i,i+1} = x_{(i+1)1} - x_{i1}$ and $\Delta_{i+1,i} = x_{i1} - x_{(i+1)1}$. The score function should award mutual shift of ϵ-steps on adjacent frequencies to enable agent movements, and to penalize it otherwise. The function is non-linear: a big shift in the right direction can outperform the negative score obtained from a few smaller shifts in the wrong direction, i.e., $\forall x1, x2 \geq 0 : \quad f(x1 + x2) > f(x1) + f(x2)$. The function has to be monotonic: $\forall \Delta_1 < \Delta_2 : f(\Delta_1) < f(\Delta_2)$. Exponential function satisfies these requirements, and we use it to calculate the score. We also use weights assigning more importance to scores obtained at lower frequencies. For the "curved up" pattern, the score function is defined as follows:

$$sU_i = \begin{cases} \text{if } \Delta_{i,i+1} \geq 0 : \frac{maxF - i + 1}{maxF} \left(e^{\frac{\Delta_{i,i+1}}{2\epsilon}} \right) \\ \text{if } \Delta_{i,i+1} < 0 : \frac{maxF - i + 1}{maxF} \left(1 - e^{\frac{\Delta_{i,i+1}}{2\epsilon}} \right) \end{cases} \tag{3.1}$$

Thus, the corresponding score for " curved down" patterns is

$$sD_i = \begin{cases} \text{if } \Delta_{i+1,i} \geq 0 : \frac{maxF - i + 1}{maxF} \left(e^{\frac{\Delta_{i+1,i}}{2\epsilon}} \right) \\ \text{if } \Delta_{i+1,i} < 0 : \frac{maxF - i + 1}{maxF} \left(1 - e^{\frac{\Delta_{i+1,i}}{2\epsilon}} \right), \end{cases} \tag{3.2}$$

where $maxF$ is the number of frequency bands. Additionally, we are looking for smooth curves. Smoothness is a measure of how gracefully the curve changes its shape. We use a second derivative to actually measure smoothness of each curve. Since we work with discrete data, the smoothness is defined as follows:

$$smooth = \frac{1}{n-2} \sum_{i=1}^{n-2} \left(x_{(i+2)1} - 2x_{(i+1)1} + x_{i1} \right)^2 , \tag{3.3}$$

where n is a number of overlapping ϵ-intervals which form a particular curve. The total score is a liner combination of individual scores and smoothness:

$$\begin{cases} scoreU = \sum_{i=1}^{n-1} sU_i + C * smooth \\ scoreD = \sum_{i=1}^{n-1} sD_i + C * smooth, \end{cases} \tag{3.4}$$

where C is a **smoothing coefficient**. The smoothing coefficient balances the score. The bigger the C value, the higher the score obtained for smoother V-shapes.

3.4.2.2 Epsilon-interval selection To obtain coherency portraits, we use ϵ-intervals around each maximum on each frequency band. Clearly, distance between maxima for higher frequency bands is smaller than for lower-frequency ones. Thus, if we keep the ϵ-interval relatively small, on the coherency portrait we will see all maxima at all frequencies. However, if these intervals are very small, we will have limited interval overlap on adjacent frequencies resulting in smaller V-shapes. If we keep increasing the ϵ-intervals, then separate intervals on higher frequencies will overlap and will form longer lines. These longer lines may cover a wider region causing V-shapes from transition areas overlap with the same line on the upper frequencies. This will decrease the ability of the algorithm to generate a proper transition signature. We used heuristics to select the size for the ϵ increasing it until we observe that intervals on higher frequencies start forming lines covering non-transition areas.

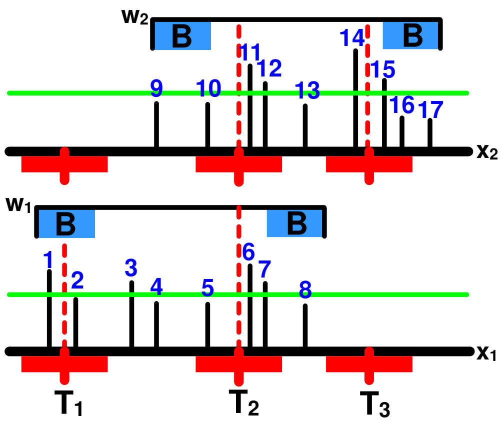


Figure 3.8: Signature metric explained

3.4.2.3 Signature detection method Our signature detection method periodically slices a segment of time series, builds coherency portraits and calculates score for every V-shape. We use a sliding window approach and the central issue with our method is information fusion: for overlapping windows we need to define how to work with redundant and conflicting information. Consider an example shown in **Figure 3.8**. Let us assume the initial position of a sliding window is w_1. In the next time step, the window has been shifted to the position $w2$. Scores obtained from window w_1 are plotted on axis $x1$, and scores obtained from w_2 are plotted on axis $x2$ correspondingly. Scores on x1 and x2 have common and unique parts. We use two approaches to resolve the conflicts due to overlapping windows: **incremental updates and glimpses**. In the incremental updates, we would scan the whole data stream with an overlapping sliding window, and would consider only those scores which were obtained for the non-overlapping portion of the sliding windows. In our example for window w_1, we take scores $1, 2, 3$. For window w_2 we take only score 6. We ignore scores 4,and 5 since they were obtained for time interval covered by the previous window w_1. The idea behind glimpse analysis is not to scan the entire data stream. Instead, we periodically sample from the stream and determine whether or not we are in the transition area. The decision making procedure considers either an incremental window or a full glimpse window.

We will use an example in **Figure 3.8** to help us explain the performance metrics. We have three transition points: at times T_1, T_2 and T_3. A sliding window is moving through the data stream with its step t_{step}. For each window w_i, we denote a starting point w_{i1} and ending point w_{i2}. If a window covers the information about the transition, it should be recognized. However, if the window covers only a fraction of a transition, the data may not be sufficient to recognize it. We specify a buffer B at both ends of a window as a no-detection zone. We expect that a transition point will be detected if it is covered by a window and it is not within a no- zone. In our example, window w_1 should detect transition T_2, while window w_2 should detect transitions T_2 and T_3. In total, we should expect three true detections **(TD)** . Assume that, after sliding a window over a data stream, we obtained a set of scores plotted on the x axis in **Figure 3.8**. When we set a threshold, every score above the threshold is a detection **(D)** of a transition. In our example, we would consider scores 1,3,6,7,11,12,14,15 to be detections. If a detection d_j occurs within an ϵ - area of a transition, i.e.,

$$d_i \in [T_k - \epsilon .. T_k + \epsilon] \tag{3.5}$$

then this is a correct detection **(CD)**. Referring to our example, we have the following correct detections: 6,7,11,12,14,15. A particular transition point T_k may be detected under multiple windows. All correct detections will contribute to the true positive **(TP)** measure. All detections that do not satisfy **eq. 3.5** contribute to false positives **(FP)**. In our example, the only false positive is 3. For incremental update, the additional condition is that the transition is detected in the area which was not covered by the previous window.

To illustrate our approach, in **Figure 9(a)** we show time-series with transitions and corresponding score dynamics **Figure 9(b)**. We observe that despite that transition is rather smooth, the scores clearly manifest the transitions. In section 3.5 we provide a comprehensive experimental study of our method and demonstrate its efficiency comparing it with other popular change detection techniques. Next, we provide a complexity analysis of our method.

3.4.2.4 Complexity analysis Our transition detection procedure consists of the following steps: obtaining a segment of a data stream, applying a filter to extract frequency bands, and calculating coherency portrait and scores. We will estimate computational complexity for each of these steps and the complexity of the algorithm in total. We denote n as the

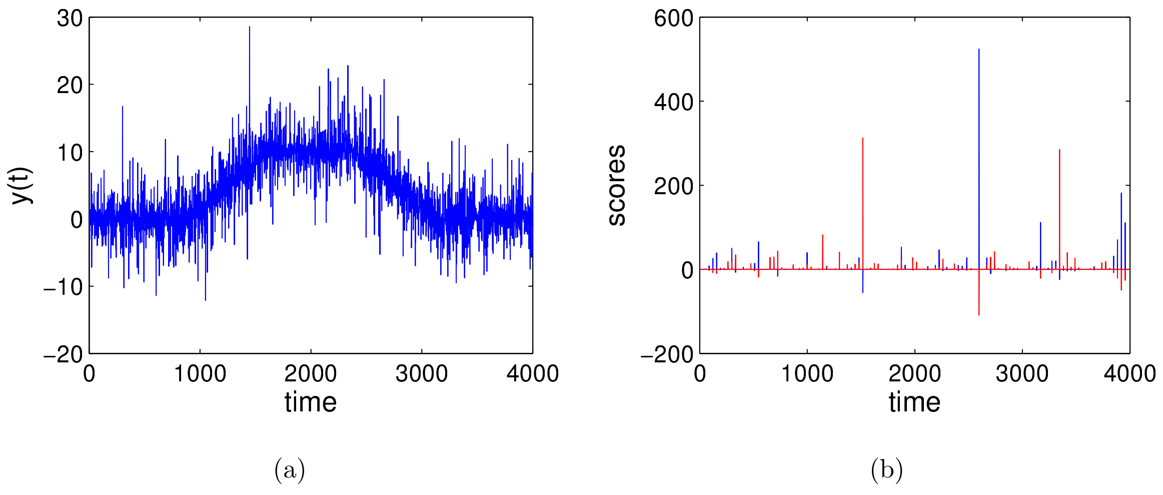


Figure 3.9: Scores for signature transition detection (a) smooth signal (b) scores for coherency portrait

length of the data stream, w - as the length of a sliding window, *step* - shift for a sliding window along the data stream, N - order of a filter, k - number of frequency bands. In our method, we use $w = 3 * N$. We will have to analyze $\frac{n-3*N}{step}$ sliding windows, thus the complexity of shifting a window is $O(n)$. For each sliding window, we apply k filters to obtain k frequency bands. Each filter is of order N. We allocate every maximum on every band and draw an ϵ-area around each one of them. For data stream filtering, we used a zero-phase FIR filter with the following output sequence:

$$y[n] = \sum_{i=1}^{N} b_i x[n-i] \tag{3.6}$$

Coefficients b_i are parameters for a particular filter and should be precomputed only once. The complexity to calculate one single element in the filter output is $N + N - 1 \approx N$. There are $3N$ points in each sliding window, so filtering a single window requires $\approx N^2$ operations. We use the zero-phase filter and the resulting harmonics have 0-degree shift from the original signal. However, processing the signal with a zero-phase filter implies that the effective order of the filter is doubled. Still, the total complexity to filter one window remains CN^2. Filtering k frequency bands would require kCN^2 steps. The number of shifts of a sliding window required to process an incoming data stream is n. Thus the total complexity of filtering operations is CN^2n.

It is more difficult to analyze the exact complexity of building V-shapes inside a sliding window. The maximum number of ϵ-intervals in a sliding window of length $3*N$ is $N_\epsilon = \frac{3*N}{\epsilon+1}$, assuming that the distance between ϵ-intervals is minimal and equals 1. For each ϵ-interval, the procedure to find overlapping ϵ-intervals on an adjacent upper-frequency band would require ϵ operations. To find all overlapping intervals between two frequency bands would require $\epsilon * N_\epsilon = \frac{\epsilon*3*N}{\epsilon+1} \approx N$ operations. To find overlapping intervals on all k frequency bands would require $\approx N * k$ operations. For all sliding windows, it would require $\approx N * k * n$ operations.

Once the overlapping intervals are found, there may be N_ϵ curves at most. Thus, calculating scores may require $\approx N_\epsilon * k$ operations. To calculate scores for all sliding windows would require $\approx \frac{3*N*n}{\epsilon+1}$ operations. To calculate scores for the whole data stream of length n, we would need $n + N^2n + Nkn + \frac{Nn}{\epsilon+1} \approx nN(N+k)$ operations. Thus we achieve linear complexity in terms of the size of the input data stream.

In terms of memory usage, we perform data analysis for each sliding window. For a window length $3 * N$, we store k frequency bands which would require us to store $3 * N * k$ stream elements. To find overlapping intervals, we need to store an additional $3 * N * k$ elements. The total data needs are $\approx N * k$ stream elements.

3.4.2.5 Comparison of complexity In table **table 3.1** we compare complexity of our approach with several related techniques. Here n is the size of the input data stream, and $N, k, m, h, d, \delta, \epsilon$ are the parameters of the corresponding methods. We observe that our signature-based method has higher CPU cost compared to the methods based on statistical change detection (such as Dasu et al. [2009] and Muthukrishnan et al. [2007]). However, it is less expensive computationally than methods utilizing fingerprints in the process dynamics (Li et al. [2010]). Our method requires only a limited amount of memory and it does not depend on the size of the data stream.

Table 3.1: Complexity comparison

Method	CPU	Memory
coherency portraits (current work)	$O(nN(N+k))$	$O(N*k)$
Parsimonious linear fingerprinting Li et al. [2010]	$n(m^2h+h^3)+mh^2$	no data
distributional shifts in data streams Dasu et al. [2009]	$O(nd\log(\frac{1}{\delta}))$	$O(dn\log(\frac{1}{\delta}))$
sequential change detection Muthukrishnan et al. [2007]	$O(n\frac{\log n}{\epsilon}\log\frac{\log n}{\delta})$	$O(\frac{\log n}{\epsilon}\log\frac{\log n}{\delta})$

3.5 EXPERIMENTS

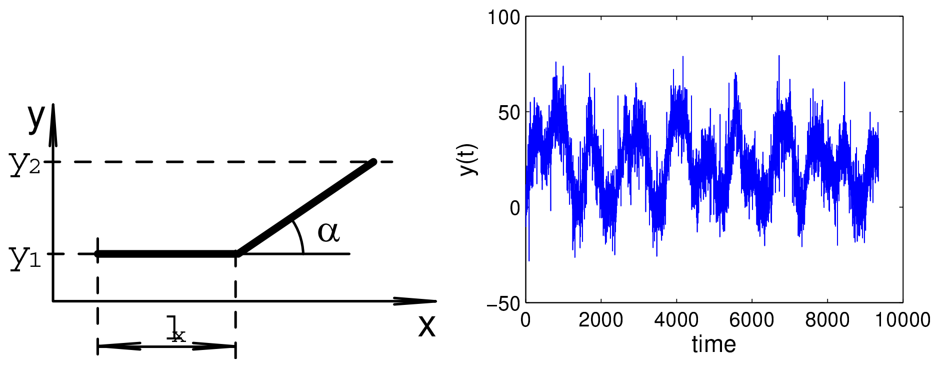
We compare our method with two widely used off-the-shelf methods: auto-regression (AR) Box et al. [2008] and kernel density estimation (KDE) Botev et al. [2010]. The AR model can be defined as

$$X_t = c + \sum_{i=1}^{p} a_i X_{t-i} + \epsilon_t \tag{3.7}$$

where a_i describes the parameters of the model, X_t describes the measurement at time t which we want to predict based on a set of predecessor measurements X_{t-i} and ϵ_t is the normal noise $pdf(\epsilon_t) = N(0,\sigma^2)$. Thus, the AR method learns the parameters of the model and noise, and uses previous data points to predict data stream dynamics. We will use two adjacent sliding windows of the same size for M data points. The first one allows us to learn all of the parameters for the AR model. We then use the obtained model to predict the following M data points in the adjacent window. Transition detection occurs when the error between what AR predicted and the actual measurements go beyond to what noise in the model can account for. The KDE model, or kernel density estimator, is defined as

$$\hat{h}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) \tag{3.8}$$

where kernel $K(\bullet)$ is a symmetric function that integrates to 1. Kernel functions help to approximate the probability of density function in the observed numerical measurements. To



(a) (b)

Figure 3.10: Stream generation: (a) stream base shape (b) example data stream

sum up, KDE transition detection method learns the probability model of the data stream (the probability to observe each particular measurement) by observing a data segment (M data points) and calculating the probability to observe new data (M data points). It assumes that the data stream model does not change. If the probability to observe the new data points is very low, we interpret this as a transition. Later, we explain our hypothesis testing procedure in more detail.

3.5.1 Experimental Setup

The data stream that we use for the comparison consists of a base shape shown on **Figure 10(a)**, repeated N times with added exponential noise. Values l_x and y_2 are i.i.d. uniform from $[l_{min} \dots l_{max}]$ and $[y_{min} \dots y_{max}]$ accordingly. We used $y_{min} = 0$, $y_{max} = 50$, $l_{min} = 25$, $l_{max} = 300$, $\alpha = \frac{\pi}{18}$, $\lambda = 5$. We generate two data streams: one for training/calibration, and another one for testing. For the training stream, we repeated the base shape **Figure 10(a)** $N = 60$ times. For the testing stream, we repeated the base shape $N = 120$ times. Example of the obtained time series is shown on on **Figure 10(b)**.

To compare our signature method to KDE and AR we, use ROC, precision and recall curves Duda et al. [2000]. We will also use the area under the curve (AUC) for each ROC curve. For the KDE and AR methods, training means selecting the optimal window size together with the corresponding parameters for each method. For the signature-based method, training would involve determining the number of frequency components to use, the order of the filter and what the smoothing coefficient C in **eq. 3.4** should be.

We move two adjacent sliding windows of length Δt with a step t_{step}: $t_2 - t_1 = t_{step}$, $t_3 - t_3 = t_{step}$ through a time series. We use data points from both preceding and following window to train KDE and AR models. We use likelihood ratio test to confirm a hypothesis that at time t_i a transition in a time series occurred. The details about likelihood ratio test are in the following subsection. For a given threshold, if likelihood ratio **(LR)** at $t_1 < threshold$ and at $t_2 >= threshold$, then we assume that a transition was detected at time $t_1 + \frac{LR(t_2)-threshold}{LR(t_2)-LR(t_1)} t_2$. We assume that a transition can be correctly detected only if an actual transition has happened within the sliding window. The sliding window has to have enough information about the transition to be sure that it detects it. We use a **buffer zone** B from both ends of sliding windows. If a transition happened at time T_k, and this transition falls into $[(t_i - \Delta t + B) \ldots t_i + \Delta t - B]$, and KDE or AR detected it within the interval $[T_k - \epsilon \ldots T_k + \epsilon]$, we consider this to be true positive **(TP)** detection. Otherwise we consider this detection as a false positive **(FP)**. If a transition is detected at t_1 and $t_1 \in [T_1 - \epsilon \ldots T_1 + \epsilon]$ it will be a true positive. If a sliding window (together with its buffer zones) is not positioned over a transition, then this event is labeled as negative. This gives us the total number of negatives **(N)**. If we do not detect a transition when it is not present, this event is labeled as a true negative **(TN)**. The equations for precision, recall, true positive rate **(TPR)**, false positive rate **(FPR)** are shown in **eq. 3.9**:

$$precision = \frac{TP}{TP+FP} \quad recall = \frac{TP}{K}, TPR = \frac{TP}{K} \quad FPR = \frac{FP}{N} \tag{3.9}$$

3.5.2 Likelihood ratio test

For both AR and KDE methods, we operate on windows located on top of the data stream. We compare data points in two adjacent windows w_{t_0} and $w_{t_0+\Delta t}$, with starting points at t_0 and $t_0 + \Delta t$, respectively. Here Δt is the size of each window. Our goal is to determine whether the new section of the data (in window $w_{t_0+\Delta t}$) is generated by the same process as data in the preceding window w_{t_0} or if those data are substantially different. We use the likelihood ratio test Casella and Berger [2001] to make the determination. We assume data points are i.i.d. Thus, $\mathbf{x}_0$ - is a vector of data points in w_{t_0}, and $\mathbf{x}$ - is a vector of data points in $w_{t_0+\Delta t}$. Using the likelihood ratio test, we obtain:

$$\lambda(\mathbf{x}) = \frac{L(\hat{\theta}_0|\mathbf{x})}{L(\hat{\theta}|\mathbf{x})} \tag{3.10}$$

where $L(\theta|\mathbf{x}) = L(\theta|x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i|\theta)$ is the likelihood function, $f(x|\theta)$ is pdf, $\hat{\theta}_0 = \hat{\theta}_0(\mathbf{x}_0)$ is the estimate of the parameter θ_0 which maximizes the $L(\theta_0|\mathbf{x}_0)$ on window w_{t_0}, and $\hat{\theta} = \hat{\theta}(\mathbf{x})$ - the estimate of the parameter θ which maximizes the $L(\theta|\mathbf{x})$ on window $w_{t_0+\Delta t}$. We test two rival hypothesis: $H_0 : \theta = \theta_0$ and $H_1 : \theta \neq \theta_0$. The decision criterion is:

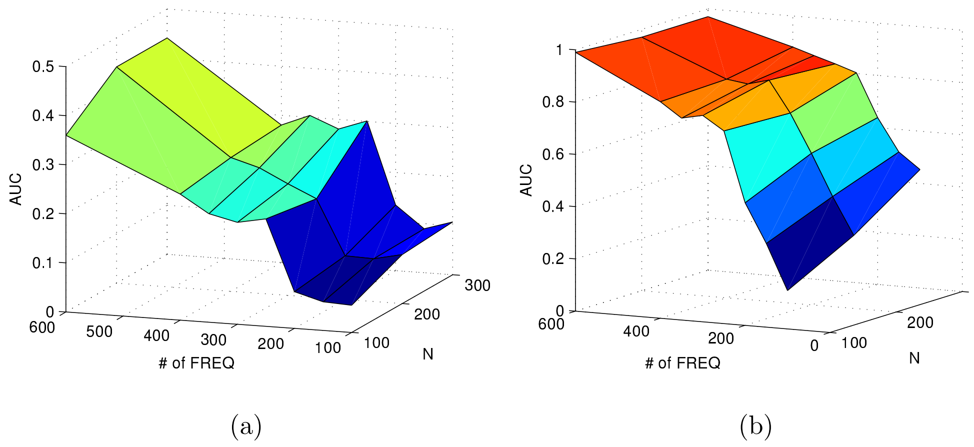
$$\begin{cases} \lambda(\mathbf{x}) \leq c : \text{reject } H_0 \\ \lambda(\mathbf{x}) > c : \text{do not reject } H_0 \end{cases} \tag{3.11}$$

where c - is a chosen threshold. In **Appendix B** we provide the proof that log-likelihood ratio test is essentially KL-divergence for big enough data sample size.

3.5.3 AR, KDE vs signature based comparison (incremental updates)

First, we apply our signature-based method to detect transitions and calculate all the required metrics. We vary the value of threshold over a wide range, thus collecting a set of results for the same data stream. Later, we plot precision and recall curves as a function of the threshold. For every value of the threshold, we calculate the true positive rate and false positive rate. Then we order the result according to the false positive rate and plot a ROC curve.

We perform calibration of AR, KDE, and signature-based methods on the training dataset to obtain the best parameter values. We calibrate these methods for smaller values of ϵ in **eq. 3.5**, tuning these methods for precise transition detection. We will also compare the performance of all methods for larger ϵ values.



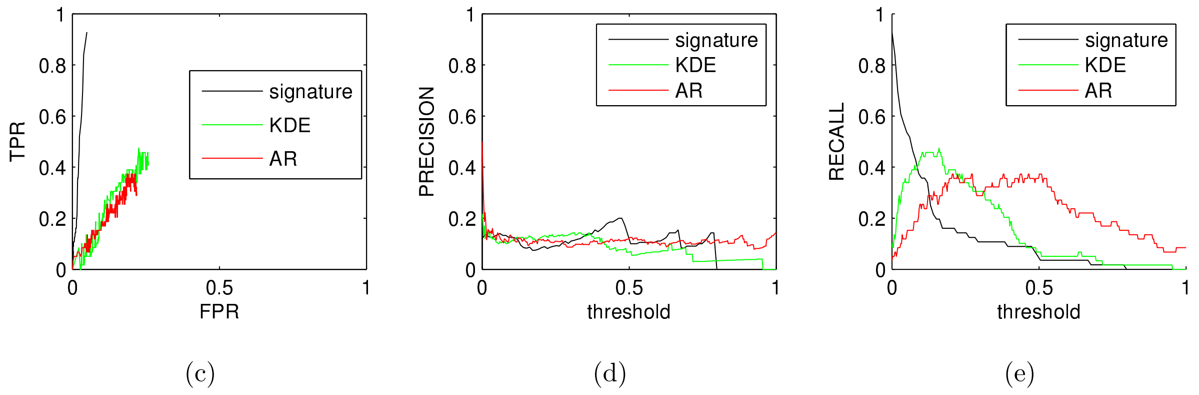


Figure 3.11: KDE, AR and signature methods on training dataset: (a) ROC plot for signature method, $\epsilon = 10$ (b) ROC plot for signature method, $\epsilon = 100$ (c) ROC curves comparison, $\epsilon = 10$ (d) PRECISION curves comparison, $\epsilon = 10$ (e) RECALL curves comparison, $\epsilon = 10$

In our performance comparison, withing the signature-based method, we were shifting the sliding window with a step $t_{step} = 100$ points. For the signature-based method, the calibration procedure helps to determine (1) number of frequency components to include into a coherence portrait, (2) smoothing coefficient value, and (3) size of the data sample (sliding window). As a reminder: we use a digital filter to obtain frequency components from the original signal. This digital filter has a parameter N - order of the filter. We always keep the size of the sliding window equal to $3 * N$. For KDE and AR, the window size was in the interval [10..100]. For the signature-based method, the resulting AUC plot on the training data set **Figure 10(b)** was calculated for the small values of $\epsilon = 10$ and is shown on **Figure 11(a)**. The overall conclusion from this plot is that lower frequency components play much more important roles in the performance of the method. We can see that the increase in performance (AUC value) is more significant when we change from 200 to 250 components than from 400 to 600. Figure **Figure 11(b)** shows an AUC plot that reflects what happens if we increase the value for ϵ to 100. This plot supports the claim that lower frequency components play more important roles in the transition detection. Another conclusion, based on these plots, is that the order of the filter (and correspondingly, size of the sliding window) plays less important roles in signature-based transition detection than the number of frequency components. We will use this property later when we analyze the results from our glimpse method. We omit AUC calibration plots for KDE and AR because of space limitations. The comparison with transition-detection methods on the training dataset is shown on **Figure 11(c)**, **Figure 11(d)** and **Figure 11(e)**. While the precision curves look similar for all three methods, the recall and ROC curves for the signature-based method are almost twice as high when compared to the KDE and AR methods. The performance of KDE and AR are similar. On the precision and recall plots, we normalized values for the threshold to be able to compare performance curves from different methods on one plot.

We used the same parameters, which we obtained for the optimal performance on the training data set, to analyze the test data set. The comparison of methods' performance for small $\epsilon = 10$ is shown on **Figure 12(a)**, **Figure 12(b)** and **Figure 12(c)**. The result is the same as for the training data set: the signature-based method demonstrates superior performance on recall and ROC curves in comparison with KDE and AR methods, while the performances of the KDE and AR are similar. However, if we make a comparison while keeping high $\epsilon = 100$, the situation becomes different. The precision is higher for KDE and AR methods. ROC and recall plots for the signature-based method are still better than for KDE and AR. The conclusion is that AR and KDE methods for big $\epsilon = 100$ can recognize obvious transitions quite accurately. However, they will not recognize less obvious transitions, which the signature-based method detects successfully. Also, AR demonstrates better performance over KDE on all three comparison plots.

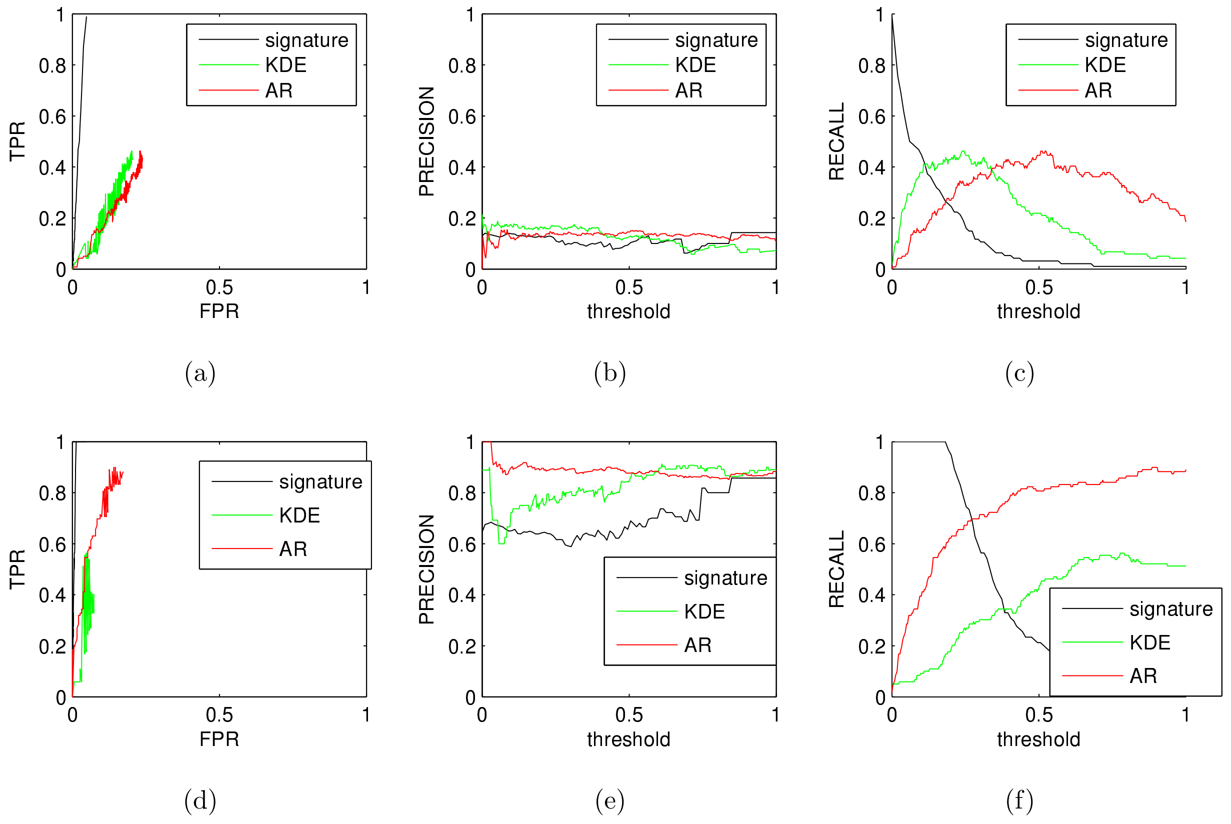
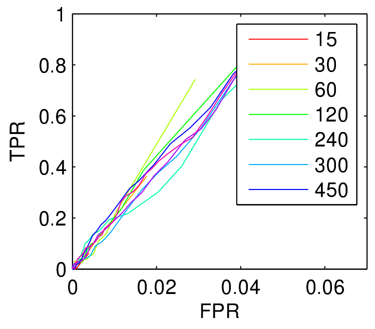


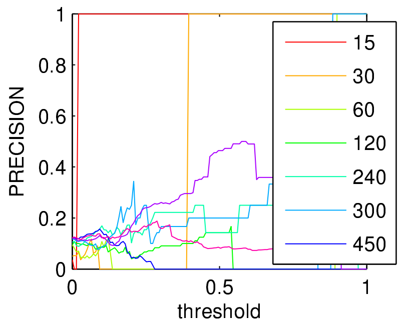
Figure 3.12: KDE, AR and signature methods on test dataset: (a) ROC curves comparison, $\epsilon = 10$ (b) PRECISION curves comparison, $\epsilon = 10$ (c) RECALL curves comparison, $\epsilon = 10$ (d) ROC curves comparison, $\epsilon = 100$ (e) PRECISION curves comparison, $\epsilon = 100$ (f) RECALL curves comparison, $\epsilon = 100$

3.5.4 Glimpse analysis

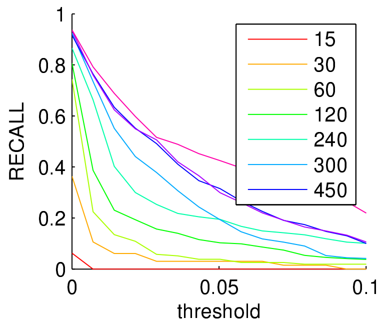
In this subsection, we explore the performance for signature-based method in glimpse analysis, when we analyze separate segments of data stream. Thus our decision about whether or not we detect a transition is based only on the limited information. The results are shown in **Figure 13(a)**, **Figure 13(b)** and **Figure 13(c)**. We zoomed in on the ROC plot **Figure 13(a)** to show that all of the curves, except for very small windows (sizes 15 and 30), demonstrate approximately the same behavior. On the recall plot **Figure 13(c)**, we observe that all the curves (except for small window of sizes 15 and 30) reach similar maximal values. Precision plot **Figure 13(b)** shows that for small windows (sizes 15 and 30) those



(a)



(b)



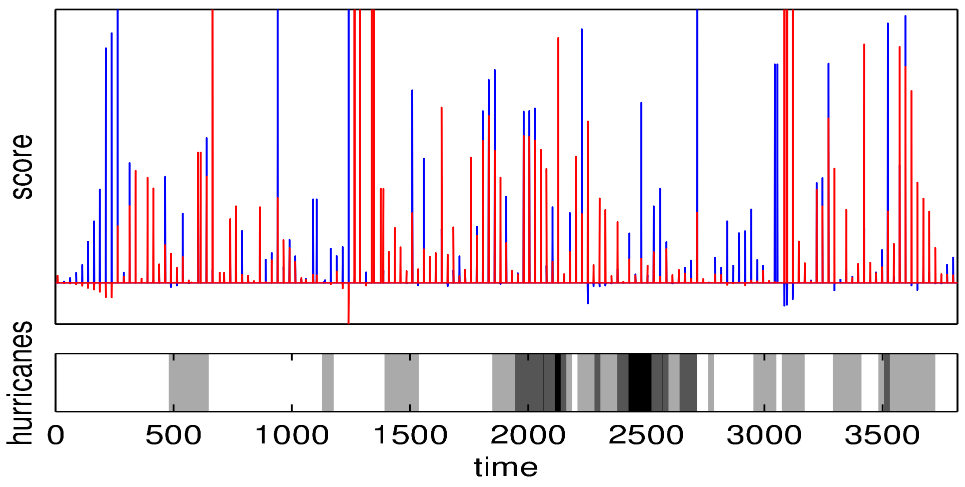
(c)

Figure 3.13: Stability of signature-based method performance as a function of the sample size (sliding window size) (a) ROC curves (b) PRECISION curves (c) RECALL curves

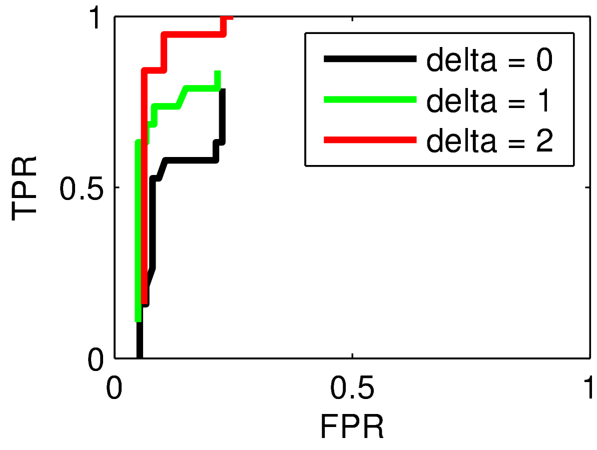
curves can reach values close to 1. This observation, together with the recall performance, allows us to conclude that for relatively small data windows the signature-based method can distinguish only quite obvious transitions. However, to detect less obvious or smooth transitions, signature-based method needs more data. The obtained results proves that we do not need to scan the whole data stream to look for transitions: we can periodically sample from the data stream and make a decision about the presence of a transition based on that discrete segment from a data stream. That confirms that *using our method, we are able to observe invariant properties of transitions even with a small amount of available data.*

3.5.5 Real data: signatures of hurricane season

We applied our method to track hurricanes in proximity of the US Atlantic coast in 2010. The original time series was collected near ICON Reef, Little Cayman, Cayman Islands (sensor id LCIY2). These measurements were obtained from the National Oceanic and Atmospheric Administration web site **ecoforecast.coral.noaa.gov**. We used sea surface temperature time series with an hourly sampling rate, taken from Jun 5th 2010 through Nov 10th 2010. The lower part of **Figure 14(a)** is a diagram that shows the duration of storms in hours within that time interval (0 on the x-axis corresponds to Jun 5th, 12 am). Higher intensity on the diagram indicates overlapping storms.



(a)



(b)

Figure 3.14: Signature-based hurricane tracking (a) **top plot:** scores for ocean surface temperature; **bottom plot:** hurricanes; time unit is hours (starting June 5th, 12 am) (b) ROC plot for hurricane detection; here **delta is specified in days**

The upper part of **Figure 14(a)** shows the coherence portrait scores obtained from the time series. Red and blue lines correspond to the "up" and "down" curves in the coherency portrait. We observe that the score spikes are tracking hurricane dynamics quite accurately. **Figure 14(b)** shows ROC curves reflecting the performance of the signature-based detection. Since hurricanes develop and dissipate gradually, we allowed for a certain time delta interval around the detection point to compensate for the inertia of the ocean surface temperature. We used one day as a time unit and the days without hurricanes were used to compute false positive rate. We observe that as we increase the time delta interval (compensating for the ocean inertia), the hurricane detection performance improves. We found that the detection accuracy depends on the proximity of a sensor on the Cayman Islands to the hurricane/storm as well as on its strength. While the sensor remains stationary, the trajectory of a storm is changing. The further the storm from the sensor, the weaker its impact on the ocean surface temperature around the sensor. The tracking is less accurate in the area with the higher concentration of storms, especially when it is a chain of weaker storms(e.g., a tropical storm not developing to hurricane). Note, that we did not use any *predefined model* for storm dynamics and the results are obtained entirely from the coherency portrait of the observed time series. This illustrates the high potential of our approach in the large-scale monitoring of highly dynamic processes.

3.6 SIGNATURE-BASED DETECTION OF BIFURCATIONS IN SYSTEM DYNAMICS

At the end we will explore applicability of our signature-based approach to complex system dynamics involving phase transition Saitta et al. [2011]. Phase transition reflects a dramatic change in system behavior. Complex dynamics involving phase transition manifests itself in various fields and is studied in diverse topics. Some common examples include: stock markets where buyers, sellers and the market adapt to economic conditions and each other's actions, or harvester-ant colonies, which show the emergence of colonies from relatively simple tasks performed by the ants. Commonly phase transition is associated with a bifurcation event Weisstein [a], - when a dynamic system demonstrates sudden appearance of qualitatively new type of behavior.

In this section we consider bifurcation detection using our signature-based method. We will generate two data streams: one using logistic equation Weisstein [c] and another one containing fold bifurcation Weisstein [a]. For these data streams we know theoretical results

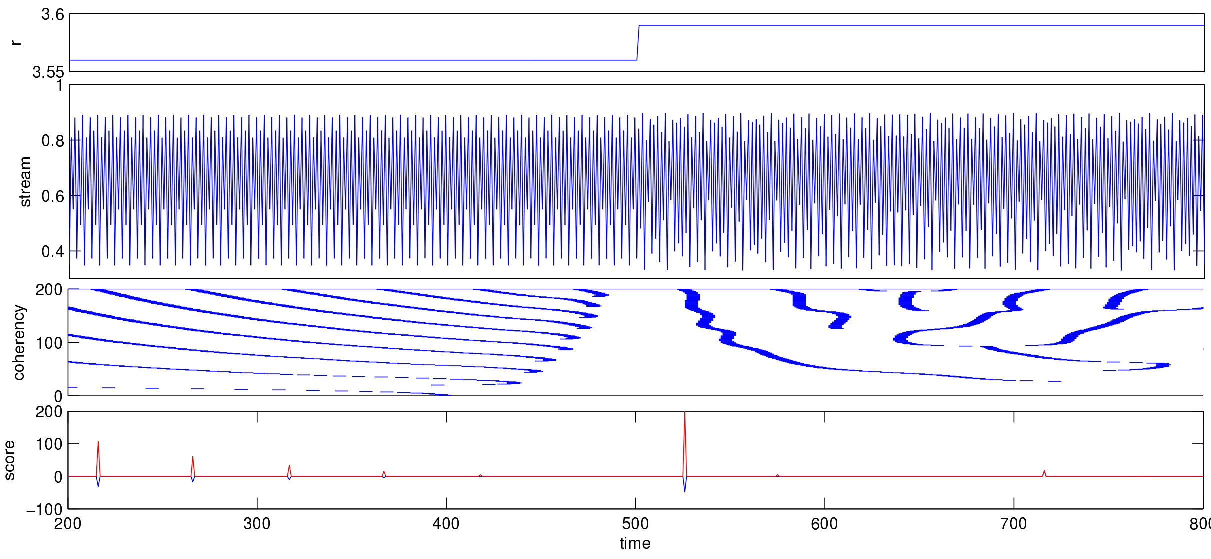


Figure 3.15: Bifurcation detection with signature for logistic equation: parameter r dynamics; data stream; corresponding coherency portrait; scores for the coherency portrait

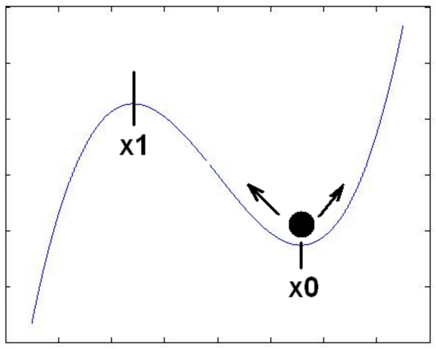
on where a bifurcation happens, and we will build corresponding coherency portraits to show how our scores detect a bifurcation event.

Logistic equation can be defined as $\frac{dx}{dt} = rx(1-x)$, where x is a number between zero and one. The logistic equation can be used to model population growth of animal species, where x represents the ratio of existing population to the maximum possible population. Parameter represents a combined rate for reproduction and starvation. Logistic equation implies a change in population dynamics depending on its parameter r. For $r \approx 3.57$ the system enters the onset of chaos. For our experiment we generated a data stream from logistic equation by setting two values for the parameters $r = \{3.56, 3.59\}$. For $r = 3.56$ we observe periodic oscillations. For $r = 3.59$ we enter into the zone with chaotic behavior. The shift from $r = 3.56$ to $r = 3.59$ happens at $time = 500$. In **Figure 3.15** we show parameter r value, the data stream, coherency portrait and the corresponding score. In **Figure 3.15** we presented the central part of the data stream and the corresponding coherency portrait (and scores) to skip additional coherency alignment which we observe closer to the start and the end of the data stream. This coherency alignment is not associated with any transition but with the fact that the data stream is finite.

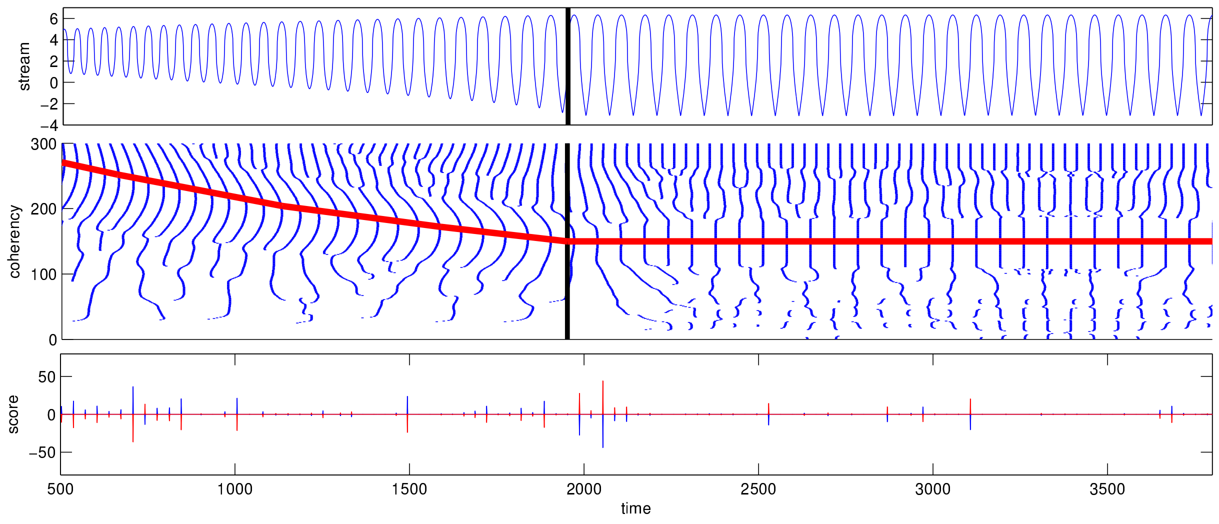
For $time \leq 500$ we observe periodic oscillations. For $time > 500$ chaotic behavior is observed. The coherency of the data stream experiences a substantial change. This change is clearly seen on the coherency portrait **Figure 3.15 middle plot** and on the corresponding signature diagram **Figure 3.15 bottom plot**. Thus signature-based method proves its ability to detect notable phase transitions.

Next we consider fold bifurcation. Fold bifurcation is a type of bifurcation in which two fixed equilibrium points of a dynamical system collide and annihilate each other. The normal expression of a fold bifurcation $\frac{dx}{dt} = \mu - x^2$. Data stream with fold bifurcation was obtained by recording oscillations of a material point in a potential field. The potential field is expressed as equation $y(x) = x^3 + ax$, where $a < 0$. The profile of this potential is shown in **Figure 16(a)**. The material point oscillates around the equilibrium x_0 with no frictions. The profile of the surface is almost symmetrical around x_0. The symmetry of the profiled gets broken as we approach x_1. While the point oscillates around x_0 due to profile symmetry we observe sinusoid-like oscillations. As the material point obtains more energy, the amplitude of the oscillations increases. Once the the point approaches x_1, the oscillations no longer resemble a sinusoid. We stop increasing energy when the point reaches $x_1 - \epsilon$ where ϵ - is a very small number. The obtained data stream together with the corresponding coherency portrait and the scores are shown in **Figure 16(b)**. Again we show only the central part of the data stream and the corresponding coherency portrait (and scores) to eliminate coherency alignment related to the start and the end of the data stream.

Vertical black line on **Figure 16(b), top plot** partitions the data stream. On the left side of the black line (for $time \leq 1950$) the amplitude increases. On the right side (for $time > 1950$) the amplitude remains constant. The coherency portrait indicates a shift around $time = 1950$. The corresponding scores in **Figure 16(b)** show that a coherent behavior before the transition ($time = 1950$) is replaced with another mode of a coherent behavior after the transition point. Once the material point approaches x_1 in **Figure 16(a)**, the total period of oscillations increases and the velocity of the point decreases around x_1. In **Figure 16(b)** we observe that the coherency portrait captured the nature of evolution in oscillating behavior. This red line follows a characteristic pattern on the coherency portrait.



(a)



(b)

Figure 3.16: Fold bifurcation analysis: (a) oscillation surface for fold bifurcation (b) **(top plot):** data stream; **(middle plot):** corresponding coherency portrait; **(bottom plot):** scores

This pattern is moving towards lower frequencies for $time \leq 1950$. This line shows that a specific shape on the coherency portrait evolves towards lower frequencies, which corresponds to increase in period of oscillations and to decrease in the frequency of oscillation. After $time = 1950$ the amplitude became constant and this evolution on the coherency portrait ends. The coherency portrait not only captures the notable phase transition in the stream, but also reflects the data stream evolution

3.7 LESSONS LEARNED

In this chapter, I presented a novel signature-based method for notable transition detection in numeric data streams. I validated that this method performs excellently in recognizing notable transitions from random fluctuations in the process dynamics. The proposed signature based method is also capable of analyzing glimpses - short pieces of data streams, to identify whether or not they reflect a notable transition. I have also demonstrated the efficiency of this approach for the task of storm tracking during hurricane season in the US, as well as for detecting phase transition in complex system behavior.

I demonstrated so far that we can find simplified behavioral patterns in a multi-agent system, when we can observe every interaction between agents (zoomed-in view). I also showed, that using the approach from CAS, we can also find a set of behavioral patterns even for a process, where we cannot observe the interval process directly.

However, what if we have a system where we can observe all the system parameters, and all the interactions. But what if the number of the factors which influence those interactions, is enormous? We can try to apply relaxed optimization or just relaxation, to replace a difficult task (exact optimization) with an easier (relaxed) task, which still represents the general properties of the original task. However, how should we design this relaxed optimization? How do we choose which parameters to use? And should we available parameters only, os should we come up with a set of new (derived) parameters?

For instance, we have a hadoop cluster, which resources are shared between many users. The major objective for every user is to have his task to be computed as fast as possible. And typically that would imply that each user would want to obtain as much resources as possible for his job. Yet, this may not be the best strategy, given that users need to compete for finite resources and the task computation acceleration is sub-linear as a function of resources and has a saturation point. A hadoop cluster has hundreds of tune parameters and system settings. However, a cluster is not exactly a deterministic system. Since users login almost at arbitrary time, and submit tasks which differ in the size of the input, it is virtually impossible to capture every single aspect of their behavior. In this case study I propose to use the approach based on the notion of strong emergence in a CAS. Strong emergence implies that we observe properties in a CAS, which cannot be attributed to the individual properties. Instead of concentrating on low-level system parameters, like CPU load, memory usage, etc I propose to use higher-order properties of the cluster as a whole. In this way we can still detect simple patterns, which will precisely enough describe the processes happening in this CAS. In the next session I show, that this approach not only simplifies the model, but also provides a substantial performance improvement exactly when its needed the most: when cluster reaches its maximum load.

4.0 COMPLEX PATTERNS IN RESOURCE SHARING [1]

Big Data challenges involve various aspects of large-scale data utilization. A common way to deal with big data analytics is to set up a pipeline of a high-performance data warehouse (e.g., Teradata Teradata or Vertica Vertica), an efficient analytics engine (e.g., SAS SAS or SPSS SPSS) and an advanced visualization tool (e.g., MicroStrategy MICROSTRATEGY or Tableau Tableau). However, the cost of such infrastructure may be considerable TeradataPricing.

Meanwhile, not every data analytics task is time-critical or requires the full functionality of a high-performance data analysis infrastructure. Often for non-time-critical applications, it makes sense to use other computational architectures, such as Hadoop/MapReduce. One might name Amazon Amazon, Google Tang et al. [2010], Yahoo Baldeschwieler [2009], Netflix Sabah [2012], Facebook Thusoo et al. [2010], Twitter Lin and Kolcz [2012], which use MapReduce computing paradigm for big data analytics and large-scale machine learning.

A/B testing Kohavi et al. [2009b] is a mantra at eBay to verify the performance of each new feature introduced on the web site. We may need to run hundreds of concurrent A/B tests analyzing billions of records in each test. Since A/B tests are typically scheduled on a weekly basis and are not required to provide results in real-time, they are good candidates for migration from the expensive conventional platform to a more affordable architecture, such as Hadoop HADOOP.

CAS description: A corporate Hadoop cluster has impressive, but still finite computational capabilities. The total amount of the resources in the cluster is fixed once the cluster setup is complete. Each job submitted to a Hadoop cluster needs to be optimized to be able to effectively use those limited resources. One way is to use as many resources as

[1] THIS WORK WAS PARTIALLY ACCOMPLISHED WHILE BEING AT EBAY INC

possible in the expectation to decrease the time for execution of a job. However, hundreds of A/B tests need to be run concurrently, and Hadoop resources need to be shared between those jobs. Even a single A/B test may require as many as 10-20 MapReduce jobs to be executed at once. Each of these jobs may need to process terabytes of data, and thus even a single A/B test can introduce substantial cluster load. Some jobs may be dependent on other jobs. Thus for optimization purposes, we need to consider all jobs which belong to the same A/B test as a whole, not as independent processes. In addition, each job has to co-exist with other jobs on the cluster and not compete for **unnecessary resources**.

What we are trying to optimize:

- minimize the execution time of a typical A/B test on Hadoop;
- optimize resource usage for each job, thus our A/B test can co-exist with other tasks;

The results reported in this paper were obtained in a pilot project to assess the feasibility of migrating A/B testing from Teradata + SAS analytics infrastructure to Hadoop. Preliminary work was conducted at eBay in the Fall 2011. A month-long A/B test experiment execution and cluster resource monitoring was completed in the Fall 2012. All our experiments were executed on Ares Hadoop cluster at eBay, which in spring 2012 had 1008 nodes, 4000+ CPU cores, 24000 vCPUs, 18 PB disk storage Parikh [2012]. The cluster uses **capacity scheduler**. All our analytics jobs were implemented using **Apache Hive**.

Consider an example Hive job, repetitively executed on Hadoop, as shown in **Figure 4.1**. We monitored the amount of resources (here - the number of map slots) used by the job together with total map slot usage in the entire Hadoop cluster. The upper plot shows how many map slots were in use in the entire Hadoop cluster during the experiment. The bottom plot shows how many map slots our sample MapReduce job received during the execution. We observe that when the cluster is becoming busy, MapReduce jobs have difficulty accessing desired amount of resources.

There are three major contributions we provide:

1. empirical evidence that each MapReduce job execution is impacted with the load on the Hadoop cluster, and this load has to be taken into consideration for job optimization purposes

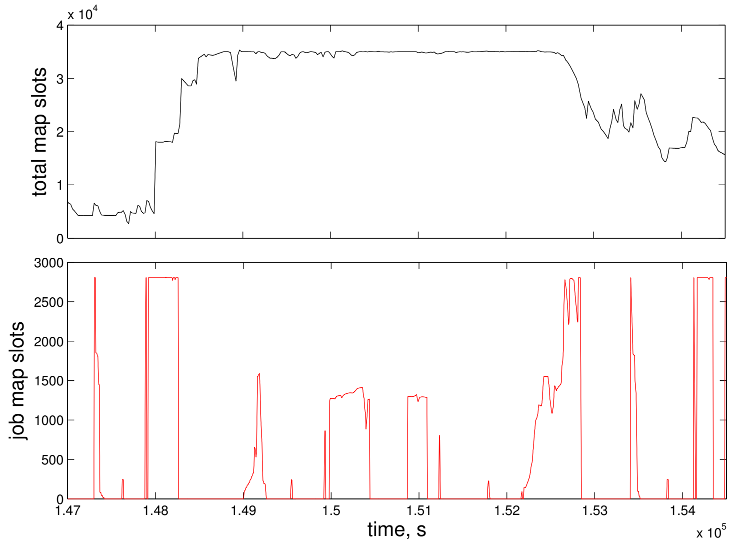


Figure 4.1: A/B test execution monitoring. Top plot: map slot usage in the entire cluster. Bottom plot: map slot usage by the A/B test jobs

2. based on our observations, we propose a probabilistic extension for MapReduce cost model
3. we provide an algorithm for optimization of concurrent Hive/MapReduce jobs using this probabilistic cost model

4.1 APPLYING ROCAS SCHEMA TO THE PROBLEM

When we get access to a very powerful computational resource, we want to use all its capabilities to obtain more accurate results (use more data, or more variety of data), and to get them faster (usually, to use more computational resources). It is obvious, that at some point the users may use all its capacity. Moreover, many jobs are scheduled ad-hoc, which may cause abrupt spikes in capacity usage. This poses a challenge for efficient scheduling of jobs on a cluster, because certain parameters for MapReduce jobs have to be set before the job gets executed, and those parameters cannot be adjusted as the the job is being executed. Moreover, some jobs depend on the completion of other jobs, and this piece of information was not possible to add to the scheduler at the time, when this work was performed.

Thus being said, we need to address those challenges in order to provide more efficient execution pipeline. And here are the steps which we apply to this problem.

- **features from raw data stream:** there are a lot of parameters, which influence job execution. But ultimately it boils down to how many resource slots a job can receive. Which is a subject to dynamic resource allocation. We need to align time series of resource usage with the job execution, and compute a derived value, which corresponds to how much data can be processed on this particular cluster in a unit time.
- **system constraints:** we understand, that cluster has limited capacity. This comes for two reasons : queue capacity limits and whole cluster capacity limits. When a cluster has spare resources, a queue may borrow extra resources from those unused resources (soft limit). When a cluster is in heavy use, a queue provides a portion of the resources, specified at cluster configuration time (hard limit). And the more jobs we submit, the more those resources being re-distributed between jobs, which affects job execution time
- **cost model:** there are many hadoop / mapreduce cost models out there, trying to capture in very detail cluster performance. However, since everything boils down to resource allocation for each job, and the data size for each job, we can use an approximate, however significantly more simple formula, which capture those relations
- **relaxation:** given that there are dependencies between jobs, some jobs need to compute their results as early as possible, while the results from other jobs are needed much later in the process. Thus, we can run these jobs as lower priority ones, in the shadow of the execution of the main jobs. These "shadow" jobs become resource scavengers, because they use only those resources, which are not used by other, higher-priority jobs. Thus we revoke resources from the "shadow" jobs. We care only that they get completed before the time, when we need their results. Yet, we do not care much about how long exactly it will take for those jobs to get completed.

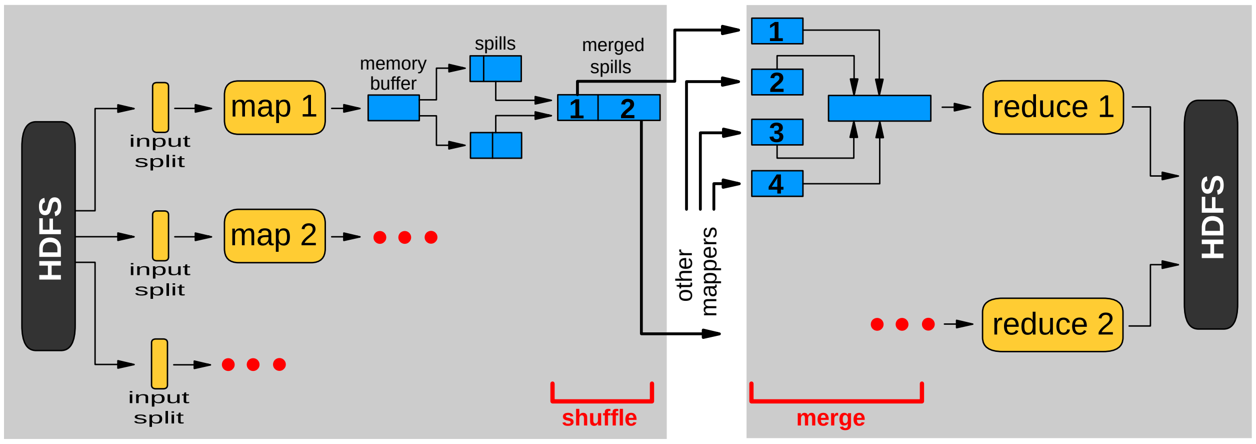


Figure 4.2: MapReduce execution schema

4.2 BACKGROUND

Apache Hadoop is a general-purpose framework for distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from a single server to thousands of machines, each offering local computation and storage. Hadoop provides the tools for distributed data storage (HDFS: Hadoop distributed file system Shvachko et al. [2010]) and data processing (MapReduce). Each task submitted to a Hadoop cluster is executed in the form of a MapReduce job Dean and Ghemawat [2008], as shown in **Figure 4.2**. JobTracker HADOOP is the service within Hadoop that farms out MapReduce tasks to specific nodes in the cluster. The JobTracker submits the work to the chosen TaskTracker nodes. TaskTracker is a node in the cluster that accepts tasks - Map, Reduce and Shuffle operations - from a JobTracker. A TaskTracker is configured with a set of slots (map and reduce), which indicate the number of tasks that it can accept White [2010] at a time. It is up to the scheduler to distribute those resources between MapReduce jobs.

There are several ways to write MapReduce jobs. The most straight-forward one is to write a Java program using MapReduce API. While this method provides the highest data manipulation flexibility, it is the most difficult and error-prone. Other tools, such as **functional languages** (Scala Scala), **data processing workflow** (Cascading Cascading), and **data analysis languages** (Pig Pig, Hive HIVE) help to cover the underlying details of MapReduce programming, which can speed up development and eliminate typical errors. **Apache Hive** was chosen as a tool for these experiments because of its similarity with SQL, and to eliminate the need for low-level programming of MapReduce jobs.

4.2.1 MapReduce data flow

Every MapReduce job takes a set of files on HDFS as its input and, by default, generates another set of files as the output. Hadoop framework divides the input to a MapReduce job into fixed-size pieces called splits. Hadoop creates one map task for each split HowManyMapsAndReduces. Hadoop takes care of scheduling, monitoring and rescheduling MapReduce jobs. It will try to execute map processes as close as possible Shvachko et al. [2010] to the data nodes, which hold the data blocks for the input files. We cannot explicitly control the number of map tasks HowManyMapsAndReduces. The actual number of map tasks will be proportional to the number of HDFS blocks of the input files. It is up to the scheduler to determine how many map and reduce slots are needed for each job.

Our Hadoop cluster was configured to use **capacity scheduler** CapacityScheduler. The whole cluster was divided into a set of queues with their configured map and reduce slots capacities. Hard capacity hard limits specify the minimum number of slots each queue will provide to the jobs. Soft limits specify how much extra capacity this queue can take from the cluster provided so that the cluster resources are under-utilized. When the cluster gets fully loaded, those extra resources will be reclaimed for the appropriate queues. In **Figure 4.1**, we observe the effect of this reclamation: when the total load on the cluster reached its maximum: MapReduce jobs in our queue were not able to obtain as many resources as they had before. A similar effect happens when we submit yet another job to a queue: each job from that queue will receive less resources compared to what they had before.

4.2.2 Concurrent MapReduce optimization

Let us assume that we submit MapReduce jobs to a Hadoop cluster with a queue Q, which has limits: M - max. number of map slots and R - max. number of reduce slots. At first, let

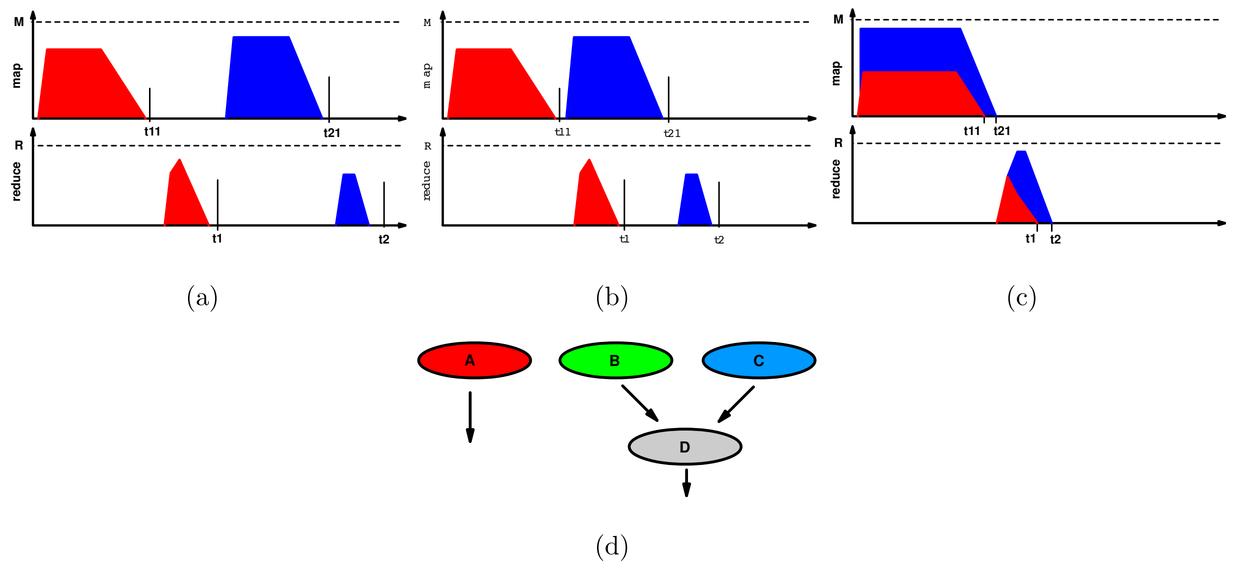


Figure 4.3: Examples of execution schedule for MapReduce jobs: (a) Completely sequential schedule (b) Interleaving schedule (c) Concurrent schedule for independent jobs (d) Concurrent schedule for jobs with dependencies

us have 2 **independent** MapReduce jobs (colored in red and blue). We will look at different scenarios for those jobs to execute. If we submit them sequentially, as shown in **Figure 3(a)**, then the cluster resources will remain idle for much of the time. When the map part for the red job is over, map slots will remain idle while Hadoop is processing the reduce part for the job. From **3(a)**:t_{11} to **3(a)**:t_1 map slots are idle.

The most obvious solution to improve resource utilization is to monitor the progress of running jobs and launch new jobs when the cluster is idle. **Figure 3(b)** demonstrates this principle. At time **3(b)**:t_{11}, when the map part of the red job is complete, the blue job gets launched. Thus, from **3(b)**:t_{11} to **3(b)**:t_1 the Hadoop cluster has its map and reduce slots utilized. This schedule will reduce the total execution time: **3(b)**:t_2 < **3(a)**:t_2, as shown in Zhang et al. [2012a]. In Verma et al. [2011a], they report a map slot utilization diagram when running a simulation for MapReduce job execution. Map task execution may have long tails where most of the tasks have been completed, however the whole job is waiting for the last few map tasks. This situation is a variation of resource starvation problem Tan et al. [2012]. Another possible improvement to the scheduling scenario shown in **Figure 3(c)**, is when we submit all independent jobs simultaneously. This approach will maximize map and reduce slot usage. However, if some of the jobs are interdependent, this approach may lead to a very interesting problem.

Consider an example scenario from **Figure 3(d)**. Here, task D consumes the output from tasks B and C. Task A is completely independent of tasks B,C, and D. The optimization in this scenario is difficult because it depends on all 4 tasks and how many resources provided by Hadoop. In a typical setup for an A/B test, **each of those four tasks is big enough that, on its own, it can occupy all map and reduce slots in the queue**. If we apply the optimization logic from **Figure 3(c)**, then task A will consume a portion of the map and reduce slots, thus for tasks B and C will take longer to finish their work and task D will start (and finish) later. If task A is relatively quick, then we would want to implement optimization from **Figure 3(b)** when task A launches its map jobs after tasks B and C finished their map part and then execute the reducer jobs. If task A is relatively long, then we would like to use a scheduling scenario from **Figure 3(c)** when task A is launched together with B and C, but receives a lower priority and uses only spare map slots and does not slow down B and C. There are many possible scenarios in this 4-task problem, and the solution to those are often non-trivial. We need to consider that:

1. other users submit their MapReduce jobs to the cluster and each job which we submit receives less resources than it is asking for

2. there is no way to explicitly control the amount of resources for each MapReduce job when using Capacity Scheduler

Launching concurrent MapReduce jobs helps to utilize Hadoop resources more effectively. Yet each of those jobs may influence other jobs running on the cluster. The scenario presented in **Figure 3(d)** is a very typical one for analytics jobs. We will return to concurrent MapReduce job optimization with data dependencies later in the thesis. In the next subsection, we will explore existing approaches to deal with MapReduce optimization together with the applicability limits of those solutions for our analytics task.

4.2.3 Current approaches for Hadoop / MapReduce optimization

We can group existing efforts in MapReduce jobs optimization into the following categories: scheduler efficiency, Hadoop and MapReduce (**MR**) system parameters optimization, and MR execution modeling.

4.2.3.1 Hadoop scheduler optimization

The default Hadoop scheduler is FIFO White [2010]. However, this may not be the best scheduler for certain tasks. The existing work in scheduling optimization addresses some of the most typical issues with execution optimization of MR jobs. One sample issue is resource sharing between MR jobs (Capacity scheduler CapacityScheduler, Fair Scheduler Zaharia et al. [2009]), so one job does not suppress the execution of other jobs or does not occupy too many of the resources. Rao and Reddy [2012] provides a summary of some of the existing schedulers for Hadoop. Here we provide a summary of the existing approaches to improve scheduling for Hadoop and how their optimization is applicable to our task of A/B testing.

Resource-aware scheduler Yong et al. [2009] suggests using fine-granularity scheduling through monitoring resources usage (CPU, disk, or network) by each MR job. Delay scheduling Zaharia et al. [2010] is trying to address the issue of data locality for MR job execution. Coupling Scheduler Tan et al. approach is aimed at reducing the burstiness of MR jobs by gradually launching Reduce tasks as the data from the map part of a MR job becomes available. These schedulers treat each MR job as completely independent of one another, which is not true for our case. In addition, these schedulers say nothing about how to re-distribute Hadoop resources between multiple concurrent MR jobs.

Verma et al. [2011a] introduced the "earliest-deadline-first" approach for scheduling MR jobs. The proposed approach (SLO-based scheduler) provides sufficient resources to a process, thus, it can be finished by the specified deadline. The authors report in the thesis that when the cluster runs out of resources, then the scheduler cannot provide any guarantees regardless of process execution time. This happens because SLO-based scheduler is backed by FIFO scheduler Zhang et al. [2012b]. Thus, first, we still need to perform off-line calculations about the optimal timing for each MR task. Verma et al. [2011a] says nothing about how to undertake this optimization, when there are many inter-dependent MR jobs, each of which is big enough to use all of the cluster resources available. Second, this approach is based on FIFO scheduler, therefore it provides the capability to control resources for each process, but this is not available for capacity scheduler which is running on our Hadoop cluster. And third, this approach does not consider the background Hadoop load caused by other processes, which can be launched at any arbitrary time.

Verma et al. [2012] considers optimization of a group of independent concurrent MR jobs, using FIFO scheduler with no background MR jobs running (from other users) on the cluster. We use capacity scheduler instead of FIFO scheduler on our cluster, and we cannot explicitly control resources assignment for each MR job. Typically A/B testing jobs show strong dependency between each other, and this dependence impacts the performance.

Wolf et al. [2010] presented a scheduler which tries to optimize multiple independent MR jobs with given priorities. It functions by providing requested amount of resources for jobs with higher priority, and redistributes the remaining ”slack” resources to jobs with lower priorities. Wolf et al. [2010] provides batch optimization for a set of MR jobs where each job is assigned its priority. This is not true for our case: other users submit their jobs whenever they want and can take a significant portion of the available resources. Thus, the approach has to take on-line load into consideration.

4.2.3.2 Hadoop/MapReduce system parameter optimization There were several approaches to model Hadoop performance with different levels of granularity. For instance, Wang et al. [2009] considers a detailed simulation of the Hadoop cluster functionality. It requires the definition of a large number of system parameters; many of them may be not available to a data analyst. Similarly, Herodotou [2011] is based on quite detailed knowledge of the cluster set up. Herodotou et al. [2011] and Herodotou and Babu [2011] approaches relax the need for up-front knowledge of your cluster system parameters and provide a visual approach to investigate bottlenecks in system performance and to tune the parameters.

While these approaches address very important issues of MR job and Hadoop cluster parameter tuning, they do not address a more subtle issue: concurrent job optimization for a cluster which is shared between many users, executing all sorts of MapReduce jobs. These approaches do not consider how to adjust Hadoop/MR parameters so that many MR jobs can effectively co-exist.

4.2.3.3 MapReduce cost model While MR job profiling Herodotou and Babu [2011] can help to optimize a single job, it is much more practical to have a simplified generative model which would establish the relationship between the MR job execution time, the size of the input dataset and the amount of resources which a MR job receives. Thus, we can calibrate this model once by executing a few sample MR jobs, measure their execution time, and use the obtained model for a quick approximate prediction of a MR job completion time.

Verma et al. [2012], Tian and Chen [2011], Verma et al. [2011b], Yang and Sun [2011], Herodotou [2011], Wang et al. [2009] attempt to derive an analytical cost model to approximate MR execution timing. Each of these approaches specifies an elementary cost model for data processing steps in MR and then combine elementary cost estimates in an overall performance assessment. It is interesting to mention, that different approaches have differences even with this simplified model. For instance, Verma et al. [2011b] and Verma et al. [2012] do not account for sorting-associated costs and the overhead required to launch map and reduce jobs as in Tian and Chen [2011]. These differences are associated with the fact that MR framework can launch in-memory or external sorting White [2010]. And thus, cost models obtained for different sizes of the data set will differ.

The Hadoop/MR framework has a substantial number of tuning parameters. Tuning some of those parameters may have a dramatic influence on the job performance, while tuning others may have a less significant effect. Moreover, some of the tuning may have a non-deterministic impact, if we optimize the usage of a shared resource. For instance, assume that after the job profiling from Herodotou et al. [2011] we decided to increase the sort buffer size for a MR job. Let us do the same for every MR job. If, after this buffer tuning, we launch too many MR jobs at once, some nodes may run out of RAM and will start using the disk space for swapping to provide the requested buffer size. Thus, from time to time, instead of speeding up, we will slow down concurrent job execution. This non-deterministic collective behavior has to be reflected in the cost model.

We based our cost model on the approaches proposed in Tian and Chen [2011] and Yang and Sun [2011], to reflect the deterministic characteristics of MR execution. We extend those models by adding the probabilistic component of MR execution. In **Section 4.3**, we elaborate on the MapReduce cost model and describe how we use it to estimate our Hadoop cluster performance.

4.2.3.4 Applicability limits of the existing solutions towards large-scale analytics tasks While the Hadoop community has made great strides towards Hadoop/MR optimization, those results are not exactly plug-and-play for our A/B testing problem. Our production cluster has capacity scheduler running and we cannot change that. Approaches assuming any other scheduler cannot be directly applied here. We cannot explicitly control the amount of resources for each MR job, because capacity scheduler controls resource sharing dynamically. We can only explicitly set the number of reduce tasks for a MR job. Our results were obtained for Hive using **speculative execution** modelWhite [2010], thus Hadoop can launch more reduce tasks than we specified. Production analytics jobs need to be executed on a real cluster which is shared between many people who submit their jobs at arbitrary times. Thus, any optimization strategy disregarding this external random factor is not directly applicable.

4.3 UPDATED MAP-REDUCE COST MODEL

Assume that we have a MR job, which takes M blocks of data as its input and a user specified R reduce tasks to be executed. We have m slots available to run map tasks and r slots to run reduce tasks. Using the notation from Tian and Chen [2011], to complete the map phase we need $\lceil \frac{M}{m} \rceil$ rounds to process it, since in one cycle Hadoop can execute only m map tasks. Following the same reasoning, it takes $\lceil \frac{R}{r} \rceil$ cycles to complete the reduce part and the total MR execution time becomes **Equation 4.1**:

$$T = \lceil \frac{M}{m} \rceil F_m + \lceil \frac{R}{r} \rceil F_r + \Theta(M, R) \tag{4.1}$$

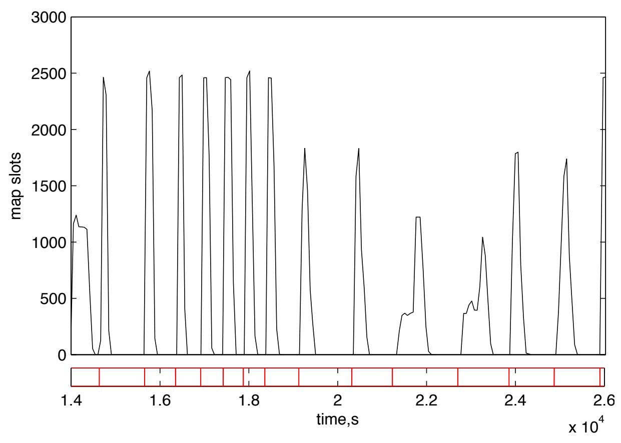


Figure 4.4: Upper plot: Map slots' usage for MR jobs. Lower plot: indicates boundaries for each MR job

where F_m is the time required for one map task to complete, F_r is the time required for one reduce task to complete, and $\Theta(M, R)$ -is the cost associated with the overhead of launching map and reduce tasks. We are interested in processing a significant amount of data, thus $M >> m$. For practical purposes, we can replace

$$\lceil \frac{M}{m} \rceil \approx \frac{M}{m} \tag{4.2}$$

Figure 4.4 shows how many map slots were assigned to the same MR job over time when it was recursively executed. During a MR job execution, resource usage is not constant and depends on how many resources are available. Thus, we should replace parameter m with its effective value, shown in **Equation 4.3**

$$\lceil \frac{M}{m} \rceil \approx \frac{M}{\frac{1}{N} \sum\limits_{m_i \in m_+} m_i} = \frac{M}{I_m}; \quad ||m_+|| = N \tag{4.3}$$

where m_+ is a set of measurements during a single MR job when map slot usage by this MR job was > 0. Similar reasoning goes for $\lceil \frac{R}{r} \rceil$. Consider **Figure 4.5** which shows a part of the experiment on reduce slots' usage. In this experiment, we sequentially executed the same Hive job, asking for different numbers of reduce tasks to be executed: **[50, 100, 200, 300, 500, 700, 900]**. The red line on **Figure 4.5** shows the number of reducers we would expect to be executed for the job. The black line shows the actual number of reducers which

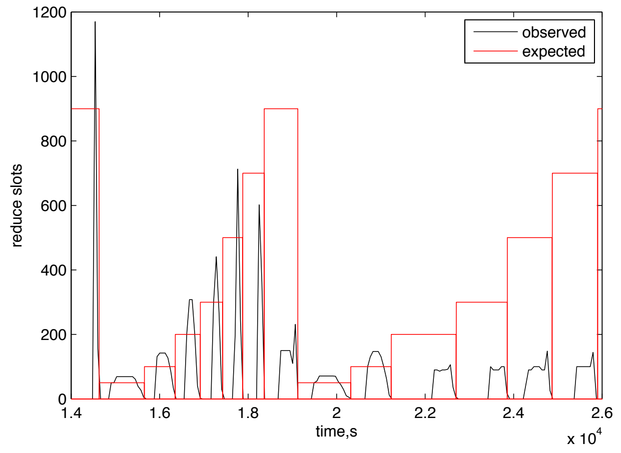


Figure 4.5: Reduce slots' usage as a function of Hadoop cluster load and speculative execution

were executed. We see that the number of reducers is not constant: it changes during the reduce part. One of the reasons is that the cluster may be busy during the job execution and some of the reducers will be reclaimed back to the pool.

We observe examples when we aimed to use 900 reducers but received only about 100. It also happens that Hadoop launches more reduce tasks than we asked for. This happens because Hadoop detects that some of the reducers made less progress than the majority, and launches copies of slow running reducers.

Based on this reasoning, we transform $\lceil \frac{R}{r} \rceil$ into **Equation 4.4**. Here the denominator is the area under the reduce slots usage curve. To eliminate counting of extra reducers (from speculative execution), we limit the maximum number of reduce slots to R. For example, if we requested 300 slots but at some point we got 350 slots, we would count only 300.

$$\lceil \frac{R}{r} \rceil \approx \frac{R}{\frac{1}{N}\sum\limits_{r_i \in r_+} min(r_i, R)} = \frac{R}{I_r}; \quad ||r_+|| = N \tag{4.4}$$

where r_+ is a set of measurements during a single MR job where reduce slot usage by this MR job was > 0. Combining **Equations 4.1,4.2, and 4.4**, we obtain **Equation 4.5**

$$T = \beta_0 + \beta_1 \frac{M}{I_m} + \frac{R}{I_r}\left(\beta_2 \frac{kM}{R} + \beta_3 \frac{kM}{R} log(\frac{kM}{R})\right) + \beta_4 M + \beta_5 R \tag{4.5}$$

In **Equation 4.5** β_0 is only a constant; $\beta_1 M_m$ is the time required to read the MR job input; kM is the size of the output of the map part of the MR job, $k \geq 0$; $\beta_2 \frac{kM}{R}$ is the time needed to copy the data by each reducer; $\beta_3 \frac{kM}{R} log(\frac{kM}{R})$ is the time required to merge-sort data in each reducer; $\beta_4 M + \beta_5 R$ is the cost associated with the overhead of launching map and reduce tasks, as suggested in Tian and Chen [2011]. However, from the dynamics of map and reduce tasks shown in **Figure 4.4** and **Figure 4.5**, it is not clear which number should be put in the calculation. We will assume that this extra cost is relatively small compared to the timing caused by the actual data processing, and will omit this overhead part, like, for instance, in Verma et al. [2011a]. We obtain **Equation 4.6**.

$$T = \beta_0 + \beta_1 \frac{M}{I_m} + \beta_2 \frac{kM}{I_r} + \beta_3 \frac{kM}{I_r} log(\frac{kM}{R}) \qquad (4.6)$$

4.3.1 Probabilistic resource allocation

Job completion time in **Equation 4.6** depends on how many resources a MR will receive. We conducted 150 iterations of the same cycle of MR jobs, each cycle consists of 7 sequential MR jobs, asking for [**50, 100, 200, 300, 500, 700, 900**] reducers, totaling in 1050 jobs execution and 9-day duration.

Resource allocation for MR jobs are shown in: **Figure 4.6** for map slots allocation and **Figure 4.7** for reduce slots allocation. From **Figure 4.6**, we observe that in probability, each MR job received the same number of map slots. This happens because we cannot

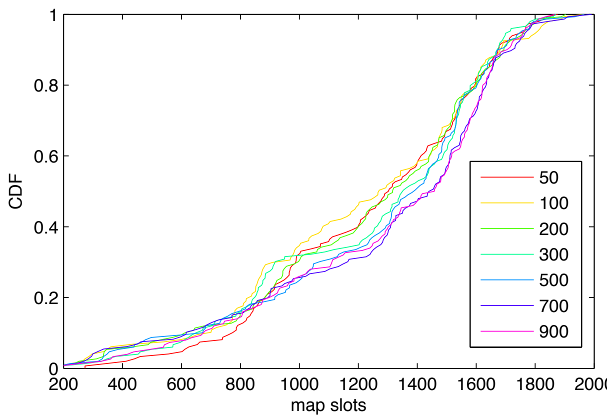


Figure 4.6: CDF plot for map slots' allocation to a MR job as a function of requested reducers

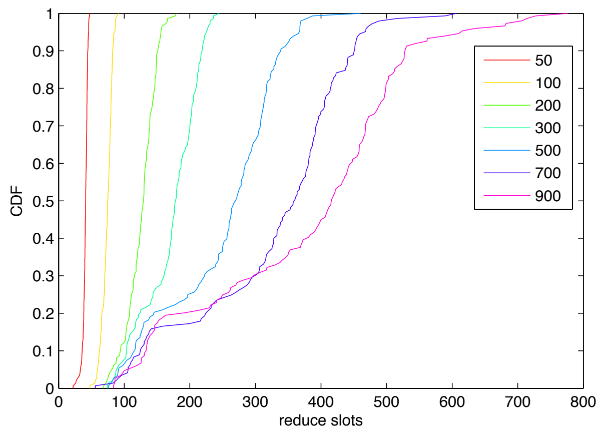


Figure 4.7: CDF plot for reduce slots' allocation to a MR job as a function of requested reducers

explicitly control map slot assignment and must rely on the scheduler to obtain resources. The situation is completely different with the reduce slots: we can explicitly ask for a specific amount of the resources. However, even if we ask for more slots, it does not mean that we will receive them. If we ask for fewer resources, most likely the scheduler will be able to provide those to a MR job. In **Figure 4.7**, when we ask for 50 reduce slots, in 90% of cases we will receive those 50 slots. However, when we ask for 900 reducers, in 90% of cases we will receive less than 550 reducers. From **Equation 4.6**, the volatility of the MR job completion time will increase when we ask for more reduce slots.

Figure 4.8 provides another perspective on the volatility of the MR execution. We have plotted the average completion time of a MR job as a function of the number of requested reduce slots together with 95% interval of the observed measurements. We observe that for jobs requesting 300 reducers or more, the average completion time remains almost constant. However the 95% upper bound of the execution time is increasing. This time increase is connected to the dynamic nature of Hadoop cluster load: some of the nodes which execute our MR job reduce tasks may arbitrarily receive extra load (other users submit their MR jobs). Speculative execution White [2010] was introduced to Hadoop to mitigate this problem, however it does not solve the issue completely. The higher the number of the reducers requested, the higher the chances are that at least one reducer will get stuck on a busy node.

To reflect this effect we propose to alter the MapReduce cost model for **Equation** 4.6 into **Equation** 4.7.

$$T = \beta_0 + \beta_1 \frac{M}{I_m} + \left(\beta_2 \frac{kM}{I_r} + \beta_3 \frac{kM}{I_r} log(\frac{kM}{R})\right) * (1 + D_R(R)) \tag{4.7}$$

where $D_R(R)$ is probabilistic extra delay, associated with the chance that a reducer gets stuck on a busy node. For each R, $D_R(R)$ is a set, containing pairs (delay, p) - shows possible extra delay together with the probability to observe this extra delay. At first, we assume that $D_R(R) = 0$ and obtain the coefficients β for **Equation** 4.7. Then we add [70..95] interval of all measurements and learn that $D_R(R)$.

4.3.2 Functional dependencies for resource allocation

Figure 4.9 shows how map resources were assigned to a local queue as a function of the total cluster map slot usage. We observe that the distribution of the measurements does not change significantly until the total cluster load reaches approximately $3 * 10^4$ map slots. Below that threshold, queue load and total cluster load seem to be uncorrelated, e.g.: $p(Q|HC) = p(Q)$, where $p(Q)$ is the probability to observe certain queue load, and $p(HC)$ is the probability to observe certain total Hadoop cluster load. When the cluster load becomes higher than

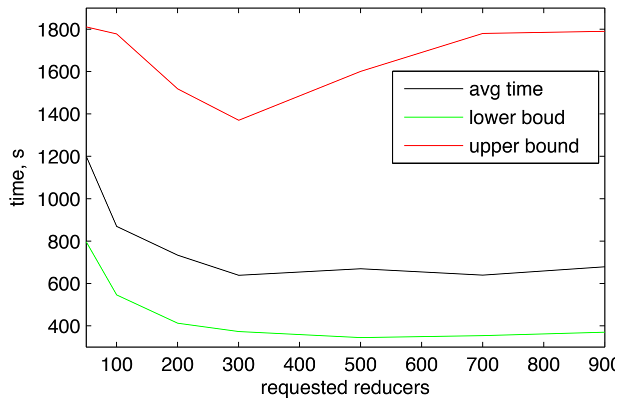


Figure 4.8: MapReduce job completion time as a function of requested reduce slots: (average time, upper and lower bounds for 95% interval)

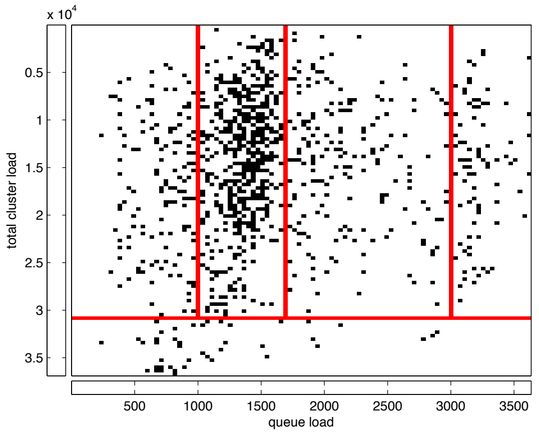


Figure 4.9: Map slot usage in a queue as a function of total cluster load

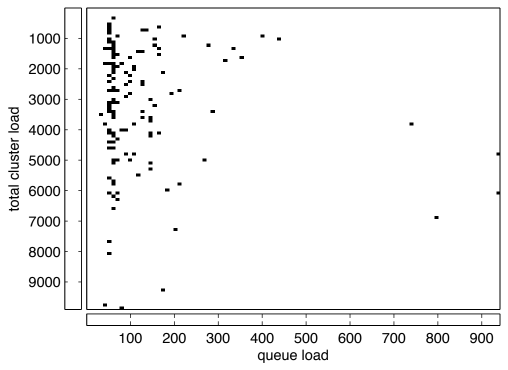
$3 * 10^4$ in **Figure 4.9**, we can clearly see that the total cluster load influences the number of slots which a queue will get: $Q \leftarrow HC$, and $p(Q|HC) \neq p(Q)$.

Obviously, the amount of the resources which a MR job obtains depends on how many resources in a queue are used by other jobs, and whether map slots can be borrowed from the rest of the cluster. If we denote r_J to be the amount of resources for a job J, then $r_j \leftarrow HC, Q$. For given values of hc_i - particular Hadoop load, and q_i - particular queue resource usage, $p(r_J) = p(r_J|q_i, hc_i)$.

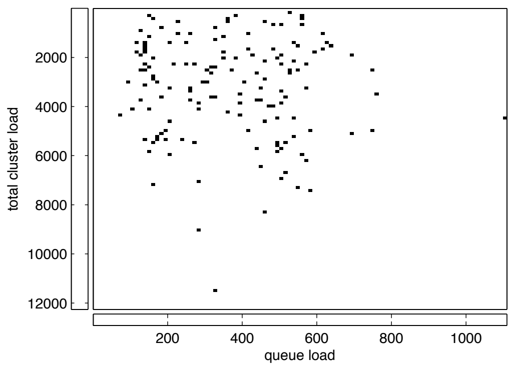
Figure 4.9 is intrinsically 3D, where the 3rd dimension - the number of map slots which a MR job obtained, is collapsed. Hadoop load analysis provides us with **(Hadoop load, queue load, MR resources)** tuples, which allow us to build a probabilistic model to describe how many resources we would receive given what we know about the total cluster load and other jobs in the queue, as shown in **Equation 4.8**.

$$p(r_J) = \sum_{hc_i \in HC} \sum_{q_i \in Q} p(r_J|q_i, hc_i) * p(q_i|hc_i) * p(hc_i) \tag{4.8}$$

Figure 10(a) and **Figure 10(b)** show how reduce resources are being distributed in a queue as a function of the total reduce usage in a cluster. What we conclude is that the nature of the distribution changes depending how many reducers we are seeking. Thus, we



(a)



(b)

Figure 4.10: Reduce slot usage in a queue as a function of total cluster load and number of requested reducers: (a) here 50 reducers were requested (b) here 900 reducers were requested

can derive **Equation 4.9** for probabilistic reduce resource allocation, similar to **Equation 4.8**:

$$p(r_J, R) = \sum_{hc_i \in HC} \sum_{q_i \in Q} p(r_J | q_i, hc_i, R) * p(q_i | hc_i, R) * p(hc_i) \tag{4.9}$$

where parameter R - specifies how many reducers we actually asked for.

4.4 CASE STUDY: MIGRATING A/B TEST FROM TERADATA TO HADOOP

In this thesis, we focus on one particular example of Big Data analytics: large-scale A/B testing. We performed an A/B test for eBay Shopping Cart dataset. An outline of the test is shown in **Table 4.1**. Originally this test was executed using both Teradata and SAS. Teradata would pre-process data and send the result to a SAS server, which would finalize the computations. We start with an overview of Teradata - a high-performance data warehousing infrastructure, and then move to the discussion of the A/B test schema.

Table 4.1: A/B test schema

procedure	**equivalent SQL query**	**engine**
extraction	$A = t_1 \bowtie \cdots \bowtie t_5$	Tera-data
pruning	$A_1 =$ select $A.c_1, \ldots$ where $\ldots$	
aggregation	$A_2 =$ select $stddev(A_1.c_1), \ldots$ group by $\ldots$	
capping	Update A_1 set $A_1.c_1 = = min(A_1.c_1, k * A_2.stdc_1), \ldots$	SAS
aggregation	$A_3 =$ select $stddev(A_1.c_1), \ldots$ group by $\ldots$	
lifs, CIs	computed using SAS	

Table 4.2: Data set size

table	**Original number of tuples**	**Number of tuples after pruning**
t_1	9,125	48
t_2	429,926	215,000
t_3	2,152,362,400	533,812,569
t_4	4,580,836,258	263,214,093
t_5	6,934,101,343	3,250,605,119

4.4.1 Teradata

Teradata Teradata is an example of a shared-nothing Stonebraker [1986] massive parallel processing (MPP) relational database management system (RDBMS) Teradata [2002], Pfeffer. A simplified view of its architecture is shown in **Figure 4.11**. The key architecture components of Teradata are: **PE** - parsing engine; **AMP** - Access Module Processor; **BYNET** - communication between VPROCs.

AMP and PE are VPROCs - virtual processors, self-contained instances of the processor software. They are the basic units of parallelism DeWitt et al. [1992]. VROCs run as multi-threaded processes to enable Teradata's parallel execution of tasks without using specialized physical processors. They are labeled as "Virtual Processors" because they perform the

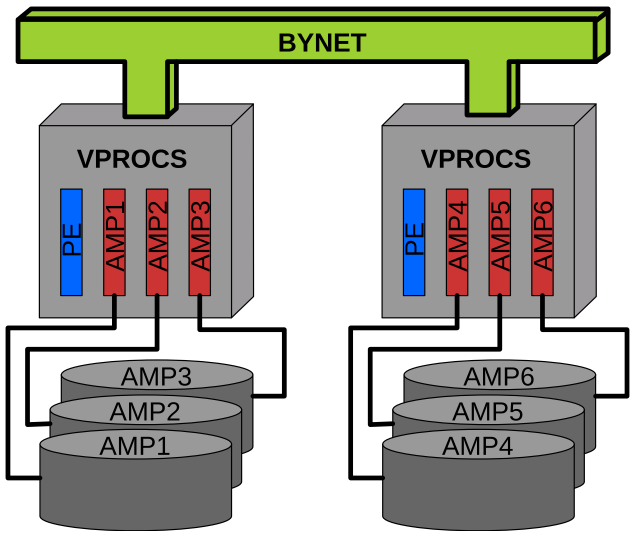


Figure 4.11: Teradata

similar processing functions as physical processors (CPU). In Teradata architecture, each parallel unit (AMP) owns and manages its own portion of the database Pfeffer, DeWitt and Gray [1992]. Tuples are hash-partitioned and evenly distributed between AMPs. And the workload for joining, aggregates calculation and sorting can be achieved in a way that the load redistributes equally between AMPs. As with any RDBMS, Teradata uses indexing Teradata [2002] on tables, which allows speeding up data access.

4.4.2 Test schema

While select, join and aggregate computations are the routine RDBMS operations, the real challenge is the size of data in tables $t_1, \ldots t_5$, which is shown in **Table 4.2** as the original number of tuples. In our sample A/B test, all the heavy data processing was executed on Teradata, and a much smaller data set would be sent to SAS. Thus, it is more important to compare the Teradata part of the A/B test with its Hadoop equivalent.

The Teradata part of the A/B test from **Table 4.1** can be translated 1-to-1 into Hive queries. The SAS part of the A/B test cannot be translated completely into Hive. We use PHP scripting language to finalize the computations which cannot be translated in Hive. However, PHP scripts would perform only a very small portion of final data assembly.

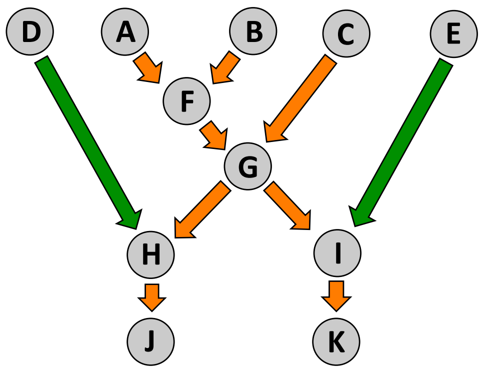


Figure 4.12: Data extraction, pruning, and aggregation schema

The schema in **Table 4.1** possesses strong dependencies between different execution steps. For instance, we cannot proceed with capping unless we computed the aggregates of the dataset. Or, we cannot compute lifts and confidence intervals unless we capped the data.

After data dependency analysis, we transform the Teradata part from **Table 4.1** into the diagram, shown in **Figure 4.12**. Letters A through K denote the data processing operations, as shown in **Table 4.3**. So now we can proceed with the comparison.

Table 4.3: A/B test data loading: extraction, pruning, and aggregation

$A = t_4 \bowtie \sigma(t_1)$ $B = \sigma(t_5)$ $C = \sigma(t_2)$ $D = \sigma_{t_3.c=v_3}(t_3)$ $E = \sigma_{t_3.c\neq v_3}(t_3)$
$F = A \bowtie B$, $G = F \bowtie C$, $H = G \bowtie D$, $I = G \bowtie E$
$J = sum(\dots), count(\dots)\ from\ H$, $K = sum(\dots), count(\dots)\ from\ I$

4.4.3 A/B test without explicit resource control

This part of the work was completed in the Fall, 2011. We run sequential and parallel versions of data loading routines from **Table 4.3** on Hadoop and compare the obtained results with Teradata timing. In these experiments, **we do not control the amount of resources (number of reduce tasks per Hive job). We let Hive to infer this based on the data size for each job**. However, all of the experiments were performed during the weekends, when the queue to which we were submitting MR jobs, was completely free.

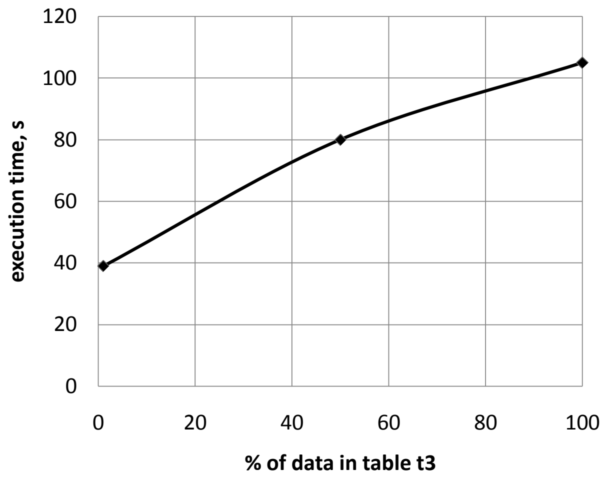


Figure 4.13: Timing for data extraction, pruning, and aggregation on Teradata

Each result was averaged over 10 executions.

In the first experiment, we execute data loading routines completely sequentially. In the second experiment, we schedule those routines concurrently, preserving the data dependencies between them, as shown in **Figure** 4.12. We launch jobs **A, B, C, D, and E** simultaneously, and proceed with other jobs as the data becomes available.

We conducted experiments with different sizes of table t_3 for data loading on both Hadoop and Teradata. Data load timing for Hadoop is shown in **Figure** 4.14, and for Teradata is

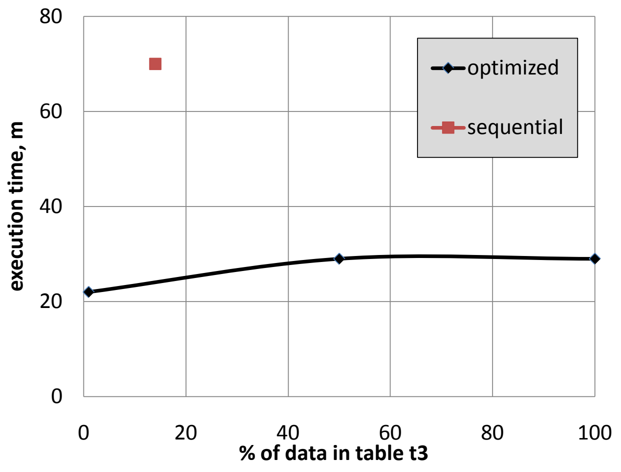


Figure 4.14: Timing for data extraction, pruning, and aggregation on Hadoop

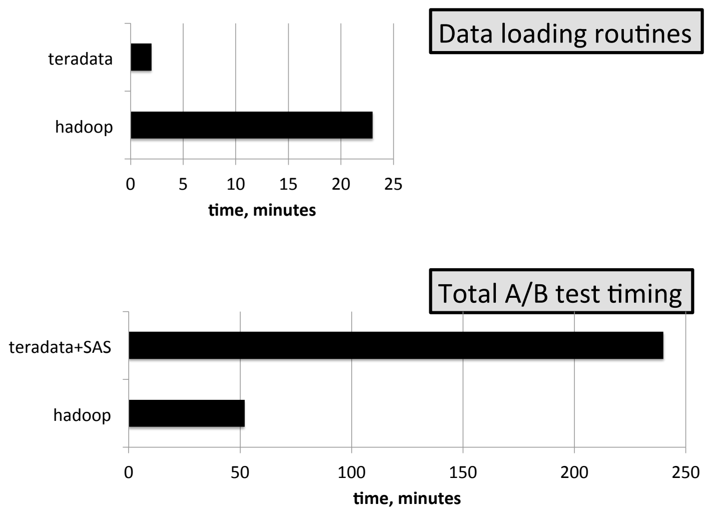
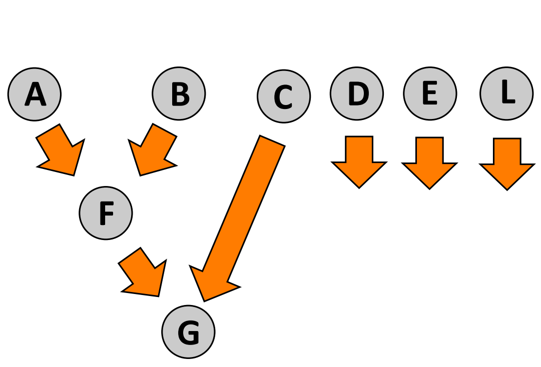


Figure 4.15: Timing for cart A/B test. **top:** time comparison of data loading routines, executed on Hadoop and Teradata; **bottom:** time comparison for execution of the whole A/B test on Hadoop vs Teradata+SAS

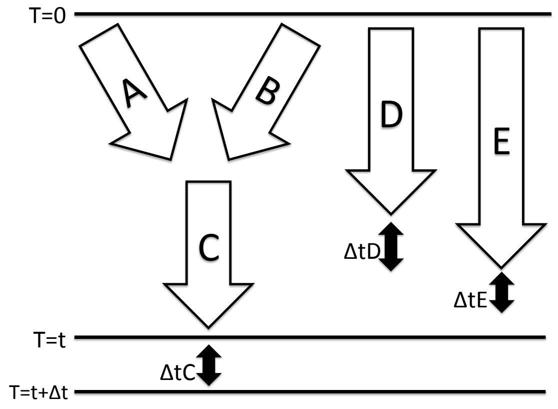
revealed in **Figure 4.13**. At first, we observe that sequential data loading on Hadoop takes approximately 70 minutes (**Figure 4.14**) when table t_3 contains 14 % of data. For concurrent Hadoop loading routines, it takes approximately 20 minutes to execute the routines having 100% of data in table t_3.

For Teradata, it takes only 2 minutes to load the data having 100% of the size of table t_3. However, the slope of the plot for Teradata in **Figure 4.13** is much steeper than for Hadoop in **Figure 4.13**. When the amount of the data in table t_3 increases, execution time for Teradata increases almost linearly. The reason is that data selection from the table happens in the background of running other processes. Processes D and E in **Figure 4.12** select data from the table and their results are required only in later stages of job execution.

In **Figure 4.15**, we compare the total timing to compute an A/B test using Hadoop only with a combination of Teradata + SAS. **While Teradata is about 10 times faster than Hadoop to load the data, Teradata + SAS appears to be about 5 times slower to compute the whole A/B test, than to do it on Hadoop.**



(a)



(b)

Figure 4.16: A/B test analysis: (a) Modified data loading routines for the A/B test (b) Resources sensitivity explained

4.4.4 Applying stochastic optimization for A/B test

This part of the work was completed in the Fall, 2012. In this section, we will present our combined optimization strategy. We will use all of the obtained knowledge about the job dependencies, MR performance model and Hadoop cluster load. For evaluation purposes, we will modify the data loading schema for the A/B test from **Figure 4.12** into the one shown in **Figure 16(a)**. In this scenario, we added an extra job L which processes a substantial amount of data, but the results of which are needed only at the later stages of the A/B test. We would like to investigate how this extra job impacts the performance. To be able to use stochastic optimization, we need the corresponding data sizes for each of these jobs. Those numbers are shown in **Table 4.4**.

4.4.4.1 Resource sensitivity Before we proceed to the algorithm description, we need to introduce the notion of resource sensitivity for a MR job. Assume that our task consists of a few MR jobs with certain data dependencies between them, e.g. **Figure 16(b)**. Resource sensitivity shows how the task execution time changes if we reduce the amount of resources for a particular job J. From the example in **Figure 16(b)**: the task starts at $T = 0$ and finishes at $T = t$. Now lets assume that we decrease the number of map or reduce slots for a particular job J on ΔR, which may lead to the increase of the task execution time from $T = t$ to $T = t + \Delta t$. We use **Equation 4.7** to predict a job completion time. Task resource sensitivity for a particular job J is shown in **Equation 4.10**.

$$RS(J) = \frac{\Delta t}{\Delta R} \tag{4.10}$$

In the example shown in **Figure 16(b)**, $RS(D) = RS(E) = 0$ because the increase in the execution time for jobs D and E does not influence the total timing: job C finishes its execution later than jobs D and E (even after we reduce the amount of resources for jobs D and E). However $RS(C) > 0$, because job C finishes its execution last, and any resource reduction to job C increases the task total execution time.

4.4.4.2 Algorithm description The central idea for our stochastic optimization is to find those MR jobs, in which results are needed much later during the A/B test execution and which require fewer resources or can be assigned a lower execution priority. This idea comes as a result of the discussion about the scalability of the A/B test data loading on Hadoop in **Figure 4.14**. In capacity scheduler we cannot explicitly control map slot assignment to a job. However, when we set a lower priority to a MR job, we force this job to use map and reduce slots only when they are not used by other processes in the same queue from the same user. When there are many users submitting their jobs to a queue, capacity scheduler provides a share of a queue resources to each user. Thus, even the lowest priority Hive job will obtain a portion of the resources.

All these considerations help us to derive **Algorithm 2** for MR job optimization strategies. When we apply **Algorithm 2** to our modified A/B test as shown in **Figure 16(a)**, we note that jobs D, E, and L receive lower priority of execution. Also job L receives 150 reducer slots instead of 800 slots, originally determined by Hive based on the size of the data input.

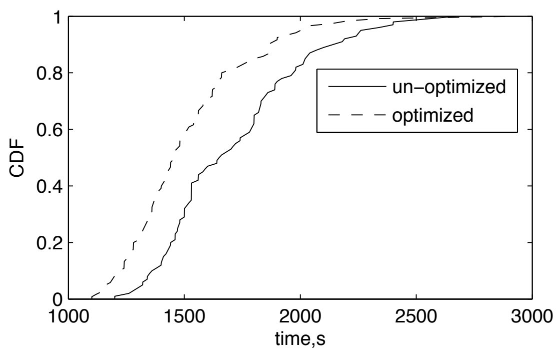
The result of this stochastic optimization is shown in **Figure 17(a)**. The un-optimized A/B test was executed 100 times; the optimized version of A/B test was executed 120 times. The first 50 iterations of both the un-optimized and optimized tests were executed from Saturday night through Sunday. The remaining iterations were executed from Monday

Table 4.4: MapReduce job size

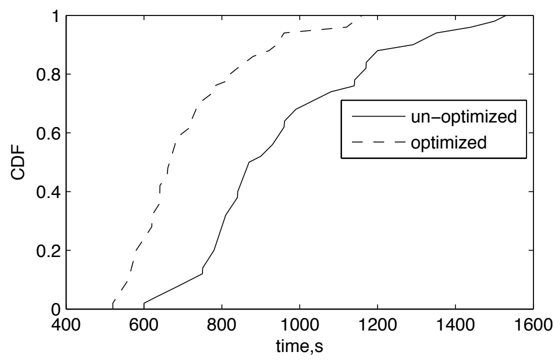
Job name	Input size (TB)	Output size (TB)
A	1.909100	0.002200
B	1.194800	0.201700
C	0.000014	0.000008
D	0.810000	0.004000
E	0.810000	0.006000
F	0.204000	0.002600
G	0.002600	0.002600
L	0.814800	0.002300

through Tuesday. In this way, we tried to measure the effect of stochastic optimization for an A/B test on variation in Hadoop cluster load.

We observe that the optimized version of data loading may take take approximately 16% less time to complete the task execution. Here, we count the optimization effect as the biggest difference between two cumulative distribution functions **CDF** for both optimized and un-optimized results. We divide this difference on the corresponding timing for un-optimized



(a)



(b)

Figure 4.17: CDF plots for: (a) Modified A/B test completion time (b) Job F start time for the modified A/B test schema

```
input  : job details; MR performance model Equation 4.8, 4.9, 4.7
output: reducer allocation per MR job in the task, priorities for each MR job
begin
    1: $(r_i; p_i) \leftarrow$ Equation 4.8, 4.9 - resource probability
    2: instantiate each MR job with resources $r_{ij} \sim size(MR_j)$, $\sum_j r_{ij} = r_i$
    repeat
        repeat
            1: compute reducer resources sensitivity (RRS) $\leftarrow$ Equation 4.10
            2: add $\Delta R$ of reducers to those jobs with highest RRS
            if total timing does not increase then
                1: subtract $\Delta R$ reducers for those jobs with the lowest RRS
                2: use Equation 4.7 to compute task total timing
            end
        until no improvement;
        foreach job $\leftarrow$ test do
            using Equation 4.10, compute map resources sensitivity (MRS), as if the
            job was assigned lower priority
        end
        1: choose the job J with the lowest MRS
        2: recalculate $r_{ij}$, timing as if job J lowered priority
        if total timing did not increase then
            lower priority for job J; recalculate $r_{ij}$, timing
        end
    until no improvement;
    1: repeat for each pair $(Resources_i, p_i)$
    2: report resources assignment
end
```

Algorithm 2: Stochastic optimization for A/B test

schedule. This optimization effect happens because jobs D, E, and L from **Figure 16(a)** do not interfere with the main execution line. And therefore jobs A, B, F, and G receive as many

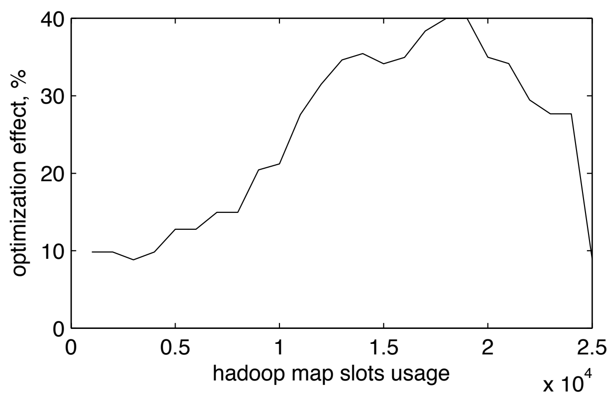


Figure 4.18: Optimization effect for a modified A/B test schema as a function of total Hadoop map slots usage

resources as available or they possibly can utilize and speed up the execution. Consider the result from **Figure 17(b)**, which shows how the start time for job F changes for optimized and un-optimized scenarios. When jobs D, E, and L do not compete for resources with jobs A and B, then A and B finish earlier and job F can start earlier. Meanwhile, jobs D, E, and L obtain their resources when these are not in use by the higher-priority jobs.

Figure 17(a) displays a cumulative optimization effect. We would like to know the effect of this optimization, depending on the load on a Hadoop cluster, for both map and reduce slots usage. **Figure 4.18** shows the effect of optimization strategies as a function of map slot usage on the entire Hadoop cluster. Here Hadoop map slot usage was calculated only during times when the A/B test jobs were using map slots. The experimental results show **the biggest optimization effect for higher levels of cluster load**, which is a very valuable contribution of this optimization. The reported value of $2.5 * 10^4$ was the highest integral map slot usage on the cluster during our experiments.

A different optimization effect is observed regarding reduce slot usage on the cluster. The improvement is at least 15% for higher reduce loads, but does not have a particular pattern. The map part of the A/B test is doing most of the heavy data pre-processing for each MR job, and the reduce part of a MR job receives a much smaller portion of the data. Thus, it is more difficult to spot the exact optimization pattern.

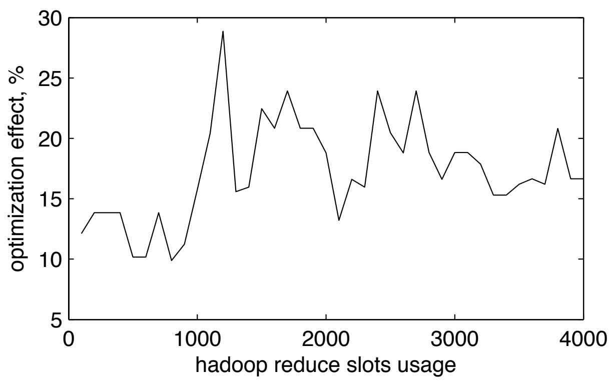


Figure 4.19: Optimization effect for a modified A/B test schema as a function of total Hadoop reduce slots usage

4.5 ACKNOWLEDGMENTS

We thank Kiran Patlolla and Jesse Bridgewater from eBay Inc, Christos Faloutsos from Carnegie Mellon University, and Martijn de Jongh from the University of Pittsburgh for valuable feedback on this work. We thank Tony Thrall, manager of the EDA team at eBay Inc. for permission to publish the results of this study.

4.6 LESSONS LEARNED

In this chapter, we report on optimization strategies which can be applied to a broad class of analytical tasks on Hadoop. We developed those strategies based on our experience of migrating large-scale analytical tasks (e.g. A/B testing) from a traditional data-warehousing infrastructure, like Teradata + SAS to an open-source Hadoop. Our optimization strategies benefit from exploring data dependencies within the analytical jobs, together with the probabilistic model of Hadoop cluster load. The effectiveness of our methods for a group of independent MapReduce tasks requires further investigation.

In our experiments, we implemented MapReduce jobs using Apache Hive. However, the obtained results are portable to any other implementation language.

In order to apply our strategies, a user must have enough privileges to perform Hadoop load monitoring to calculate performance coefficients for **Equation** 4.7 and the probabilistic Hadoop cluster load. As an option, those results can be optioned by the system administrator, and provided upon request. However, Hadoop load time series are crucial to derive the optimization strategies.

Our results are obtained for a Hadoop cluster using capacity scheduler. Optimization results provided in this thesis may not be 1-to-1 applicable to those clusters running different scheduling mechanisms. While the performance **Equation** 4.7 would remain valid, the validity of the **Algorithm** 2 is yet to be confirmed for other schedulers.

However, what if we go even further in the task of CAS optimization? For instance, what if we have a social multi-agent system, e.g. eBay marketplace. And the task is to improve the this system. Where would we start? How would we decide what needs to be optimized ? This task is similar to cluster resource optimization, but there are some differences. At first, there is very big social component and less defined optimization parameters. The social component leads to even higher number of the parameters. Moreover, this also leads to a question of how to define the state-space in this task, and how to define the optimization goals.

5.0 SESSION MODELING TO PREDICT ONLINE BUYER BEHAVIOR [1]

All major internet companies use controlled experiments (randomized experiments, A/B testing) Kohavi et al. [2009b] to learn about how well their new products / features / algorithm updates perform before making them available to all users Kohavi et al. [2009a]. However, despite its benefits, A/B testing is not a perfect tool, and does not solve every problem Kohavi et al. [2007]. For instance, **A/B tests do not directly show the root causes of the problem**. A new feature may result in poor visual formatting or an error in the page functionality in certain web browsers. These errors may drive buyers away from purchasing. A/B test may reveal negative outcome, but it will not identify the root cause of it. Also **A/B tests show only what is taking place at the current moment for the particular selected population sample**. It cannot predict the effect on a market place in the long run. And we also have to limit the duration of an A/B test: there are always more tests waiting to be scheduled.

In this chapter, I explore the feasibility of augmenting experimentation by predicting the behavior of buyers in the eBay marketplace. This prediction would allow us to address one of key business questions: how a certain feature will influence the whole marketplace in a few weeks or months from its launch time [2].

CAS description: an online marketplace, where users buy, sell, check items, and contact sellers to verify item details and shipment policies.

What we are trying to optimize: the accuracy of predicting buyer purchase activity

[1]THIS WORK WAS PARTIALLY ACCOMPLISHED WHILE BEING AT EBAY INC

[2]This work summarizes our exploration towards the possibility of long-term marketplace prediction, and does not reflect current production practices at eBay Inc.

5.1 APPLYING ROCAS SCHEMA TO THE PROBLEM

Following the same rule as the previous sections, I will show how the steps from **Section 1.3** can be applied to help address this problem.

- **statistics from raw data streams**. This bullet point is definitely running ahead of time, before the formal introduction of this technique in **Session 5.5**. However, this is one of the corner stones of the principle, which finally allowed us to make buyer prediction a feasible task. In **Section 5.4** I showed, that without session summarization we hit the problem of data sparsity: people use slightly different sequences of actions to accomplish the same goal. We need to account for this variability. We do this by working with session summaries instead of raw sequence of observed events. One assumption, which we rely on, is that actions in a single session are related to each other, and represent one intent. We will question the correctness of this assumption in **Chapter 6**.
- **system constraints**. It is not a trivial task for the problem of buyer behavior analysis to understand what those constraints might be. In **Section 5.6.2** a regression analysis was performed to determine how different features influence prediction accuracy. While it is very difficult to provide much details on the complex interplay between all of the features, we showed, that certain features work as **"reset states"** for short-term behavior prediction: these features reset the memory of the process about what has happened earlier. Once a buyer executed that type of action, he or she is almost equally likely to execute any other action afterwards, regardless of what he / she was doing before. We call this phenomenon as **"mission completion"**. If a buyer made a purchase, he might be done with his intent (ran out of money if the purchased item is costly, or he is just not a frequent online shopper), and we need to capture this fact in the model. Similarly to time series predictions, where we cannot extrapolate time series beyond physical limits of the system. Here, we cannot extrapolate purchasing activity past the fact of actual purchase. In **Section 5.6.3** we discovered a phenomenon, which we call **"behavior indicator"**, where certain actions demonstrate long-term effect on buyer activity. For instance, if a user is only a **"window shopper"**, we should not expect any purchase. On the contrary, if a buyer is trying to negotiate the price, this indicates that he is interested

in buying. These system constraints were discovered and described at the end of this chapter this chapter, after the model was built. Features, built around these constraints were not put in the model in this chapter. However, I put them in use in **Section 6**, when I discuss a follow-up method to model buyer behavior and new ways to generate behavioral features.

- **relaxation techniques**. At the beginning of this work, the objective was to predict the exact sequence of actions in buyer activity. The preliminary results of this attempt in **Section 5.4** showed, that this is, unfortunately, a mission impossible. We had to incorporate a couple of relaxations in the problem definition, to make it tractable. At first, we decided to predict only financial events, like Bid and BIN. We do not really care about the precise sequence of actions before those financial events: we want to understand whether a particular user is a buyer or not. Another relaxation we introduce to the session model itself. In **Section 5.6.3** we conclude, that certain actions significantly reduce the uncertainty about short-term history, and Bid and BIN activity is among those. We use this fact for session summarization: whenever we see that there was Bid or BIN activity within this session, we leave only BidBin action as a session summary, as those actions dominate everything else in this session.

In the next section I provide the overview of existing approaches, which can be applied to the task of online buyer behavior analysis, and the detailed description of the proposed method.

5.2 RELATED WORK

In this section, we review the research in the area of modeling online user behavior, together with potential applicability limitations for our work. We divide the approaches into different groups:

- **markov models** Chierichetti et al. [2012] and Borges and Levene [2007], Bühlmann and Wyner [1999], Chierichetti et al. [2012], Begleiter et al. [2004], Borges and Levene [2010],Cao et al. [2009b], Cao et al. [2008], Ron et al. [1996], Xiang et al. [2010] and Borges and Levene [2010] share the results on the effectiveness of variable-length markov models to predict online buyer behavior. however, these methods do not address the issue of long-term behavior prediction
- **understand the user's browsing intent**. Hassan et al. [2010],Shen et al. [2011], Cao et al. [2009a], Shen et al. [2011] , Sadikov et al. [2010], Hu et al. [2011], and Lin et al. [2012] discuss mainly how to model online browsing goals. Major applicability limit of these methods is the fact that we have thousands of different intents, and we do not necessarily know how quantitatively to distinguish between those intents.

Our work is most closely related to the usage of higher-order markov methods provided in, for instance Chierichetti et al. [2012], to model the context and the prediction procedure. We also utilize the buyer intent assumption from Hu et al. [2011], in that we assume one intent per buyer session. We investigate the validity of those methods. In **Section 5.3**, we start with the description of how we use higher-order markov models for our prediction task. In **Section 5.5**, we extend our approach by modeling buyer intents.

5.3 CONTEXT-BASED BEHAVIOR PREDICTION

At the heart of our approach is the assumption that we can predict future buyer activity through context-based simulation. We build this model from the behavioral logs collected from all the buyers. The obtained model would predict a buyer's future actions, given what we have learned from all other buyers. We assume that there is a finite set A of possible actions a buyer can take, e.g. **Equation 5.1**

$$A = \{a_1 \dots a_N\} \tag{5.1}$$

We are interested in predicting the future action of a particular buyer given the last few actions we have observed from this person, as shown in **Equation 5.2**:

$$p(a^{t+1}) = p(a^{t+1} | a^t \dots a^{t-k+1}) \tag{5.2}$$

where a^{t+1} is the action, which we would like to predict, and $a^t \dots a^{t-k+1}$ is the history (sequence of actions) of up to k actions, which we have observed from a buyer behavioral log. Each action can be one from the set of actions $A:\ a^{t+1} \dots a^{t-k+1} \in A$, and $\sum_{i=1}^{N} p(a_i^{t+1}) = 1$. To store this probabilistic model, we use a context tree, shown in **Figure 5.1**, similar to Cao et al. [2008].

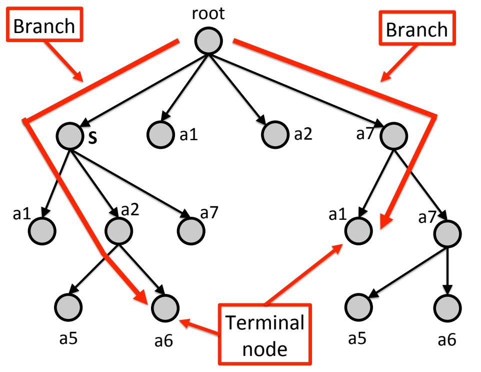


Figure 5.1: An example context tree structure. Here S corresponds to START

Table 5.1: Example buyer actions

Asq - ask seller a question; **Watch** - add to watch list; **Srp** - search request page;
Vi(cat).dt[...] - viewing an item; **Bid** - bidding on an item; **BIN** - buy it now;
Selling - Selling an item; **Offer** - Buyer offers a different price for a fixed price item;
Feedback - leave a feedback on the item;
Sign-in - signing in with login and password

5.4 BUYER BEHAVIOR PREDICTION ASSUMING NO HIDDEN STRUCTURE OF BEHAVIOR

In this section, we assume that we can effectively predict all buyer future activity by looking only at the last few actions of a buyer. Our data set contains logs from all eBay buyers, collected during 24 hours of observations. Each line contains sequences of all the actions in their chronological order, executed by a distinct buyer. We will call it **DATASET 1**. Some of the events we record, are shown in **Table 5.1**.

We use **MAE** (mean absolute error) score, as suggested in Borges and Levene [2010]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|1 - p_{norm}(a_i)| \tag{5.3}$$

here n is the length of a path, p_{norm} is the normalized probability to observe action a_i after observing a certain sequence of actions.

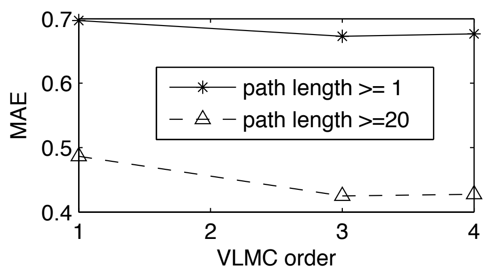


Figure 5.2: Mean absolute error (MAE) for path prediction: for all buyer paths, and for paths, longer than 20 events

From **Figure 5.2**, we observe that MAE is lower for longer paths. We observe that, in general, MAE score does not decrease dramatically as we increase the order of the VLMC: higher-order VLMC out-performs first-order MC, but not significantly. In order to understand, why this is happening, we have a closer look at the data itself in the next subsection.

For this purpose, we will collect two statistics from the learned context tree: the combined distribution of entropy for each terminal node and the node out-degree for each terminal node.

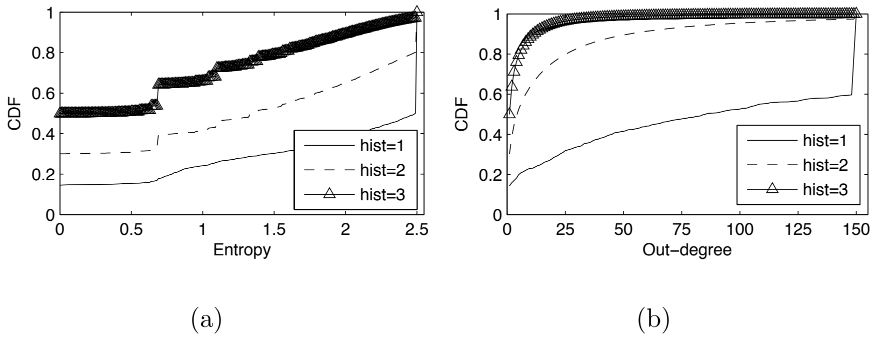


Figure 5.3: Visualizing context tree properties, as they change when we learn higher-order (longer history) MC (a) CDF plot for context tree entropy (b) CDF plot for context tree out-degree

From **Figure 3(b)** we observe that, for VLMC model of order 3, about 50% of the nodes have only one possible action for future transition in the terminal node. From the out-degree distribution in **Figure 3(b)**, we hypothesize that the reason for this is the sparsity of the behavioral patterns: many patterns can be found in the training data set only once.

From **Figure 3(b)**, nodes with out-degree of 2 and 3 have the highest-possible entropy for future transition. Which is equivalent to the fact that nodes with out-degree 2 and 3 have further transitions ofalmost equal probability.

We also measure how the proximity to Bid or BIN event influences the accuracy of prediction for those events. We step back a few events from the location of the target event, and generate a set of predictions. The result of this exercise is shown in **Figure 5.4**.

We observe quite interesting dynamics in the prediction of Bid or BIN events. The accuracy of the prediction is very sensitive to the proximity to the targeted event. Yet, the accuracy of the prediction is almost not sensitive to the distance, if we are far from the target

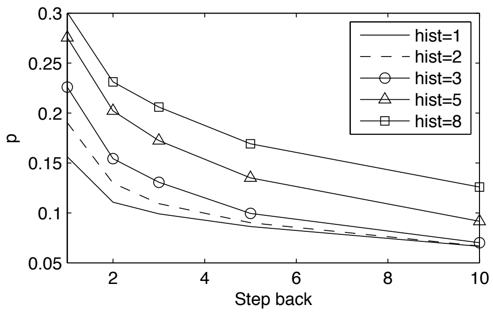


Figure 5.4: Probability to correctly predict Bid/BIN events as a function of the proximity to the target events

event. At this time we hypothesize that there exists a **buyer context around a Bid or BIN event**. It would appear that buyer actions form correlated clusters.

5.4.1 Summary of the results for prediction of "raw" session events

In this session we began our discussion with the assumption that we can predict a buyer's future activity by looking only at his last few actions. We discovered, that:

1. the **training data set is sparse**. As we increase the order of the VLMC, most of the terminal nodes in the context tree start to have an out-degree of 1.

2. for a VLMC of order 3 approximately 20% of terminal nodes have out-degrees of 2 and 3 (possible actions). And the probability to choose one of those actions is almost identical. This happens because there are certain actions within buyer paths which work like **"reset states"**, after which a buyer can take almost any action.

3. we discovered that within the buyer path, there are sequences of highly correlated events, which correspond to buyer goals. We need to introduce the notion of a goal in our behavioral model in order to increase the prediction accuracy.

5.5 BUYER SESSION MODEL

In this section, we explore the idea that a session as a whole may represent a single intent or goal, and that is why certain events are highly correlated, because buyers are on a mission. We assume that within a session, all the actions correspond to a single intent. And a session summary will be able to represent a buyer intent.

For the experiments in this section, we collected more than 7 days of observations of buyer activity in the eBay marketplace. We separate buyer actions into different sessions, when there was more than a 30 minute break between the actions. We will call this log **DATASET 2**.

We model each session as a **"bag of actions"**: we assume that all of the actions are independent from each other, conditional upon the buyer's intent or goal. Thus, a session is represented with the help of tuples **(action, number of occurrences)**, alphabetically ordered by the action name. If a Bid or BIN event happens within a session, we consider this event to be much more important than other events, and replace the whole session with a **BidBin** event An example of a session model is shown in **Equation 5.4**:

$$Asq(>1), Srp(>1), Vi.dt[0-10)(>1); \quad BidBin; \tag{5.4}$$

where ; is the separator between two sessions. The summary of the session provides us with **session features**. From **Equation 5.4**, example features can be identified as $Asq(>1)$, $BidBin$ etc.

For session prediction, we use **DATASET 2**, which we collected during a week of observing buyer activity. When we deal with session models, the distribution of the length of each path is more predictable. We divide actions into two different sessions when we observe at least a 30 minute break in buyer activity. Thus, on a weekly basis, we know the maximum number of sessions a buyer can possibly have. We also know the typical frequency of buyers visiting the web site, thus we can assess the typical number of sessions.

According to typical experimental protocols, we use 80% of the data set for training, and 20% for testing. We observe a monotonic increase in prediction accuracy for a higher order of VLMC in **Figure 5.5**. This result suggests that modeling session as a whole is the right feature generation step. Within the session, a buyer may have a very sophisticated pattern navigating through items, and it is impossible to predict. Moreover, we do not need to predict it. On the other side, a higher-level session summary tells us about buyer engagement, and his goal.

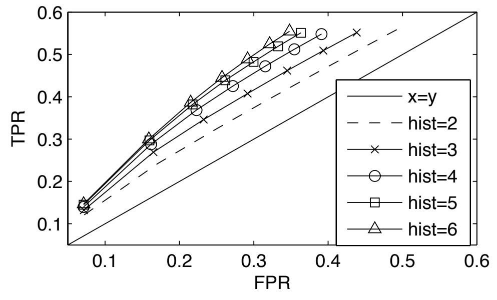


Figure 5.5: A set of ROC curves for paths built from session models. All paths consist of least 20 sessions. Here, history corresponds to the order of VLMC

Buyer prediction is a very data-intensive task. When we learn probabilistic models, we artificially inflate the state-space. The measurements of memory usage are shown in **Figure 5.6**.

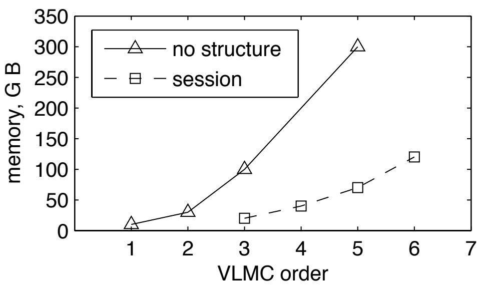


Figure 5.6: Memory usage by two approaches: when we assume no session structure, and when we build a session model

DATASET 1 and DATASET 2 occupy $\sim 9\,GB$ on disk. When we learn the context tree, the size of this tree is much larger than the actual data set. We observe that the context tree, which we learned from the session model (for **DATASET 2**) consumes $\sim 120\,GB$ for the VLMC of order 6. The context tree, learned from the buyer action log without the session model (**DATASET 1**), consumes $\sim 300\,GB$ of RAM. It appears that the **session model provides more accurate results. The context tree for the session model consumes less memory than the non-session model**. Our experiments were conducted on a corporate server with 80-core Intel(R) Xeon(R) E7- 4870 Processor @ 2.40GHz, 1TB of RAM, and running under CentOS Linux 6.0. All of the data structures were able to fit entirely in RAM.

5.6 LOWER-DIMENSION PATTERNS

5.6.1 Session components hypothesis

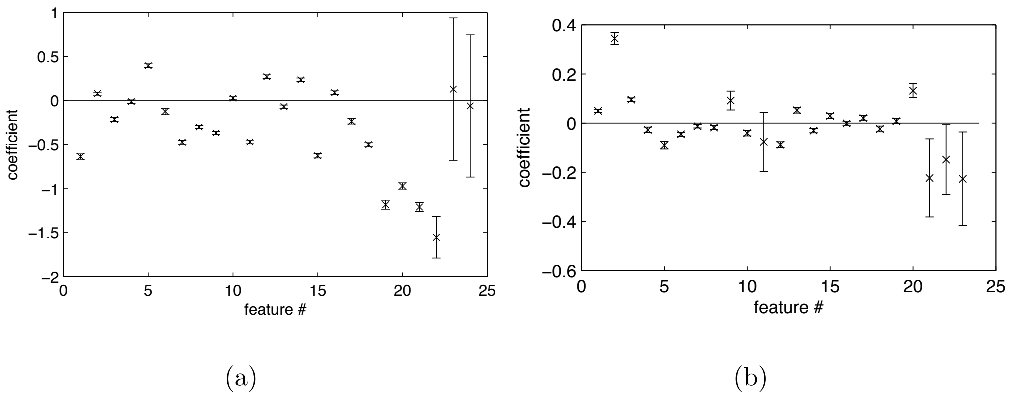
In **Section 5.3**, we started with the assumption that we only need to know the last few actions of a buyer to predict this buyer's future activity. In **Section 5.5** we questioned this simple assumption and proved, that it is more beneficial to build a session model to predict buyer behavior. In this section we question the premise that a longer history (or higher order of VLMC) improves the prediction accuracy of buyer activity. In particular, we determine branches in the context tree for which the increase in history length is beneficial, and for which it is not. We outline three hypothesis regarding behavioral models.

The first hypothesis is that the usage of a higher-order VLMC (the length of the history we consider) provides more accurate predictions for the next buyer action, compared to lower-order VLMC. Essentially, we seek to determine, if the increase in VLMC order decreases the uncertainty (entropy) about the predicted action (session). Our belief is that buyer behavior prediction using a VLMC of order 3 should have a smaller error than the prediction using a VLMC of order 2. Or in general, if we uniformly (for each branch) increase the history length, the prediction accuracy for the next action should increase.

Our second hypothesis is that some of the actions may work as **"reset events"**. All of the buyer activity before a "reset event" is completely disconnected from what this buyer will do after the "reset event".

5.6.2 Experiment design

The common sense suggests that some of the session features (which were introduced in **Section 5.5**) are more useful than others in predicting buyer behavior. Also, some of the



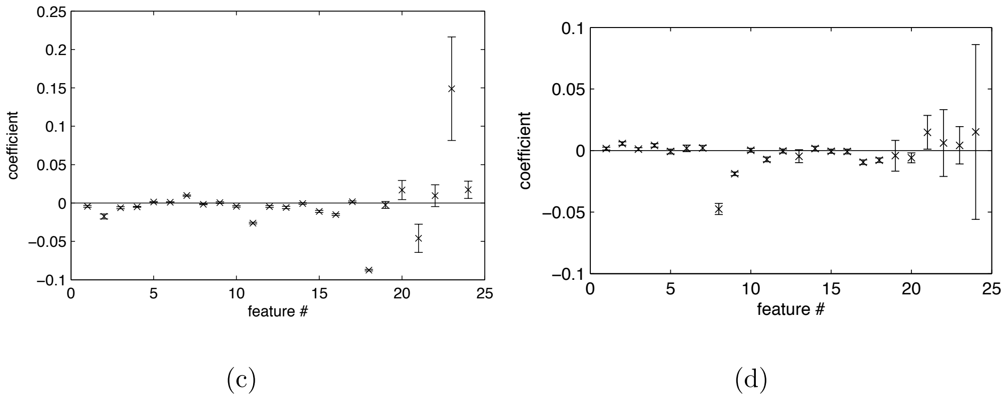


Figure 5.7: Regression coefficients together with their 95% confidence intervals for session features. (a) **features' contribution to uncertainty** (b) **features altering uncertainty** (c) **features' contribution to brier score** (d) **features altering brier score**

session features may introduce noise, and make the prediction task harder. In this section, we will outline the procedure to investigate how different session features influence the prediction performance. In particular, we seek to learn, which features help to reduce the uncertainty in predicting the buyer's next session (short-term prediction), and which features influence the prediction outcome in the next few sessions (long-term prediction). We will use two metrics to provide the answer: entropy Bishop [2006] and brier score Borges and Levene [2010]. Brier scorer is defined in **Equation 5.5** as

$$BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2 \tag{5.5}$$

in which f_t is the probability to forecast feature t, and o_t is the probability to observe feature t in the actual data set.

We assume that all session features are independent, and that we can represent the entropy for a certain terminal node, or the brier score for a path prediction, as a simple sum of the individual effects contributed by each feature. This simplified assumption allows us to use linear regression Bishop [2006] to measure the strength of influence of each feature on the prediction, as seen in **Equation 5.6**:

$$M_j = \beta_0 + \sum_{i=1}^{N} \beta_i * \delta(f_{ij}) \tag{5.6}$$

where M_j is the j-th observation for one of the metrics, $\delta(f_{ij})$ is equal to 1 if the feature i is present in the j-th measurement; otherwise, it is equal to 0, and β_i is the regression coefficient for a particular feature i, which shows the strength of the influence of this feature on the outcome.

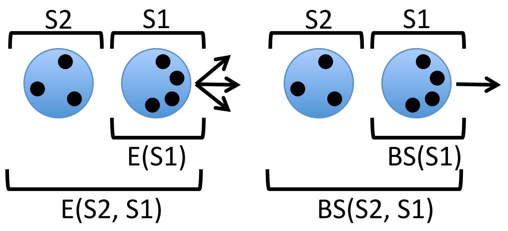


Figure 5.8: Metrics dynamics as we increase the order of a VLMC: (a) entropy alteration (b) brier score alteration

As we mentioned, not every session feature may be of equal importance: some features may show dominant influence within the session, some may not. We also hypothesize, that some features may have influence not only for the single session where they appear, but also on the following sessions. Using regression analysis, we will investigate whichever is true.

5.6.2.1 Regression on entropy For this exercise, we will learn a first-order VLMC. For each node S_j in the first-order VLMC, we compute the entropy E_{S_j}, and then combine this value with the list of session features into **Equation 5.6**. The obtained regression coefficients are shown in **Figure 7(a)**.

5.6.2.2 Entropy alteration In this section, we explore how the increase in order for VLMC changes the uncertainty about predicting the next session. We learn a second-order VLMC, and will re-use the first-order VLMC from **Section 5.6.2.1**. We describe the procedure with the help of **Figure 8(a)**. For a VLMC of order 2, each branch would consist of 2 nodes (sessions). In the example, those are: $(S2, S1)$, where session $S1$ occurred right after session $S2$. For a VLMC of order 1, we know the entropy for session $S1$ is $E(S1)$. Now, for a VLMC of order 2, we know the entropy for a branch $(S2, S1)$ is $E(S2, S1)$. Features from session $S2$, which happens before $S1$, alter the uncertainty about which session to observe after session $S1$. Combining VLMCs of order 1 and 2, we obtain tuples $\mathbf{S2->(E(S2,S1)-E(S1))}$. We plug-in features from session $S2$ and entropy values $(E(S2, S1) - E(S1))$ into **Equation 5.6** and then perform linear regression. The regression coefficients show how features f_i, if they occured before session S_j contribute to the change in the uncertainty about which session to observe after session S_j. The regression coefficients for session features are shown in **Figure 7(b)**.

5.6.2.3 Brier Score regression Brier score in **Equation 5.5** shows how close our prediction is to the ground truth. Here, we investigate how the session features influence the correctness of the prediction of Bid and BIN events many steps in the future, and if there is such influence. The results of the regression are shown in **Figure 7(c)**.

5.6.2.4 Brier Score alteration We would like to know, how the accuracy of long-term buyer path prediction changes, if we increase the order of a VLMC, and how different session features influence this change. The procedure is very similar to uncertainty alteration, shown in **Section 5.6.2.2**. From **Section 5.6.2.3**, for each session $S1$ we calculate the corresponding brier score $BS(S1)$, using VLMC of order 1. Now we learn a VLMC of order 2, compute the brier score, and combine the results, as shown in **Figure 8(b)**, to obtain a set of pairs
$\mathbf{S2->(BS(S2,S1)-BS(S1))}$. Finally, we perform linear regression on the obtained pairs. The results of the regression are shown in **Figure 7(d)**.

5.6.3 Regression results explained

In **Figures 7(a) - 7(d)** I show the regression coefficients and their 95% confidence intervals around those values. These confidence intervals help us to make some conclusions about the effect each of the features brings. For instance, if the confidence interval crosses y = 0 line, it is a strong indicator, that this feature is inconclusive on its own, and the effect of this feature may dependent on presence or absence of other features. However, if a particular feature is an outlier in either way, that provides us with some qualitative conclusion.

Consider **Figure 7(a)**, which shows how certain features influence the uncertainty about the following session. Features 19-22 are outliers, relative to other features. And the biggest outlier is feature 22, which corresponds to Bid and BIN activity. This provides us some insight, that once we observe one of these financial events within a given session, these events pretty much summarize the essence of a session. This provides us with some confidence, that we can perform session summarization, described in **Section 5.5**.

On the other hand, when we consider **Figure 7(b)**, one obvious outlier is feature 2, which corresponds to adding an item to a shopping cart. In terms of semantics of actions, we can think of this action as "the buyer found what he wanted". This feature works as a reset state - if this feature was observed in a certain session, then it dramatically increases our uncertainty about what to expect from the following session.

From **Figure 7(c)** we observe, that feature 18 is an outlier in reducing brier score (increases the most the accuracy of long-term prediction), which corresponds to offering a new price for an item. This action semantically means that a buyer is really interested in this item or a similar item, and we should not be too much surprised to see a purchasing event soon down the line from this user.

And finally in **Figure 7(d)** we observe yet another outlier - feature 8, which says that the user is not very interested in items he is checking, because he is not spending much time carefully checking those items. This lack of interest works almost as an indicator that we should not expect any purchase-related from this user soon.

5.7 CONCLUSION

We started this work with the assumption that we can predict buyer behavior by studying only a buyer's last few actions from the log. In **Section 5.4**, we tested our ability to predict every individual action from a buyer path, and we showed that it is a very difficult task. We cannot predict every event from the buyer path, and **we should not**. Instead, it is much more beneficial to concentrate on predicting most important (financial) events: Bid and BIN. Another important lesson learned concerning the purchasing context: it appears that a few actions right before Bid or BIN actions help to reveal the purchasing intent.

We learned another important lesson from **Section 5.5**. We assumed that each buyer session represents a single buyer intent. Following this assumption, we learned a session model of buyer activity. The prediction accuracy increased by 300%, compared to the accuracy from the individual event model. These results suggest that we do need to consider buyer actions with regard to buyer intents.

In the next chapter, I will present my final approach to address the problem of long-term buyer behavior prediction. I will focus on the issues of how to model intent continuation between sessions, as well as how to incorporate some of important conclusions I derived in this chapter, such as: mission completion, multiple intents per session, and some more complex buyer behavior patterns.

6.0 FEATURE ENGINEERING FOR LARGE-SCALE BUYER BEHAVIOR MODELING [1]

When we are trying to model either human behavior or a notable transition is a multi-agent system, we are facing over and over the same fundamental problem: feature engineering and feature selection. I showed, that careful feature selection helps to understand complex dynamics in multi-agent systems (e.g. fire evacuation or resource sharing for cluster computations), and to detect the fundamental changes in system behavior (notable transitions in data streams). However, the process of designing those features can be very tedious, (and yes, it is).

In this last chapter of my dissertation, I am building on top of my previous experience with eBay marketplace modeling, introduced in **Chapter 5**. I showed that markov models are suitable for long-term buyer prediction. However, this result was not obtained by applying out of the shelf learning method. If fact, the first results questioned the entire feasibility of such endeavor. Only after we have applied meaningful session summarization, markov chain model approach showed promising results.

While performing this summarization, we relied very heavily on the assumption, that each session has only one mission or one intent. Also, in our consideration we completely omitted the aspect of time. Empirically, we observed, that some of the buyers visit the web site very frequently, while others do not. This irregularity poses a tremendous challenge for behavior prediction: we do not have a solid idea what the prediction horizon should be for VLMC model. I showed, that the accuracy of the prediction increases when we consider longer prediction horizons. Thus knowing the exact horizon is paramount for the accuracy purposes, and we should use a better estimate than just a random guess.

[1]THIS WORK WAS PARTIALLY ACCOMPLISHED WHILE BEING AT EBAY INC

Moreover, buyers often use more than one physical device for their online shopping. We observed, that buyers can be put into a few different shopping categories based on their device usage. This is an informal classification, used to qualitatively describe predominant usage pattern.

- **desktop-only:** these are the users, who perform most (or all) their actions using a desktop or laptop computers.
- **mobile-only:** uses, which predominantly use a mobile device, like a smartphone or a tablet
- **multi-screen:** users, who use both : laptop and mobile devices

All these different categories have certain unique aspects in their behavior. Before the modeling process, we performed an extensive visualization study of distinct buyer sessions to get a flavor of how complex indeed their behavior is, and what are the key aspects in their behavior. I continue with the visualization in the next section.

6.1 VISUALIZATION STUDY OF BUYER SESSIONS

This chapter will predominantly contain visualizations of some of the distinct shopping scenarios we have stumbled upon, and which helped us to shape the understanding about some of the key behavior.

6.1.1 The speaker guy

The first shopping case, which we call "the speaker guy", is shown in **Figure 6.1**. This

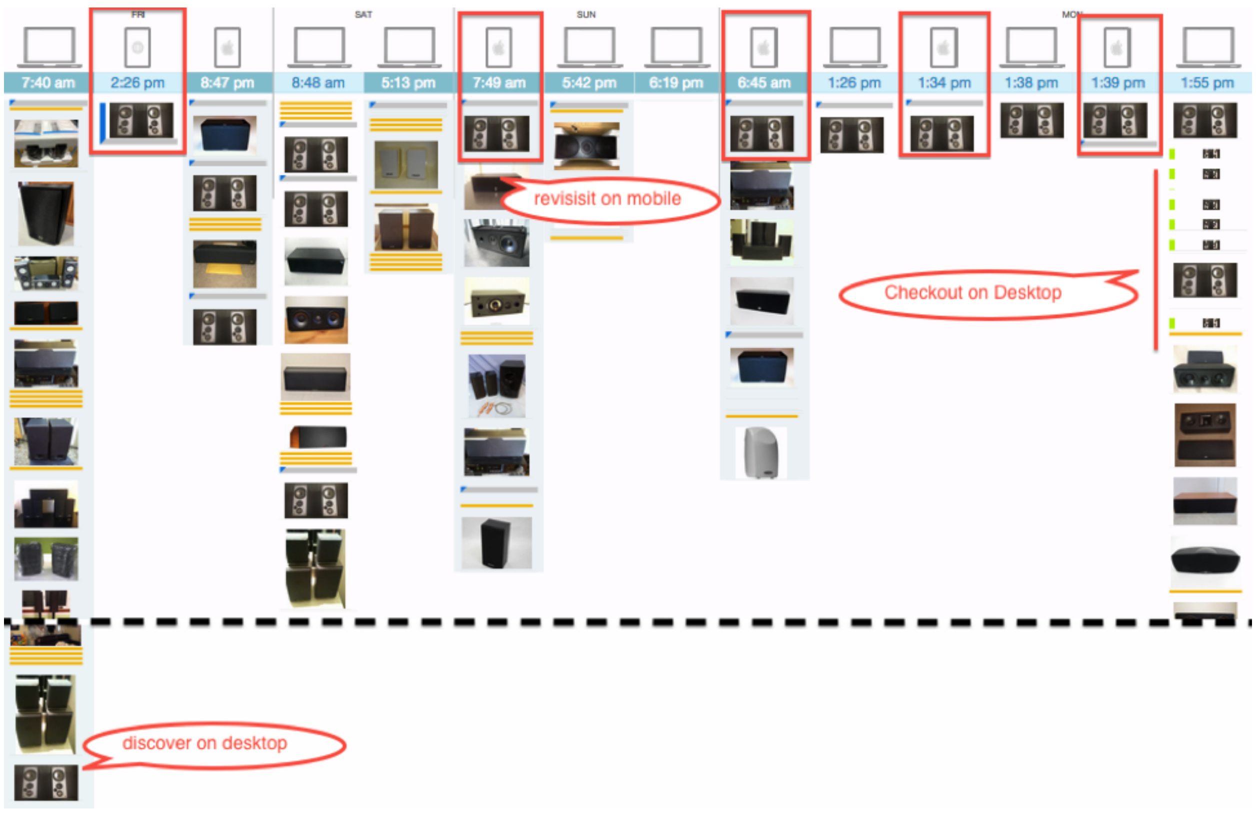


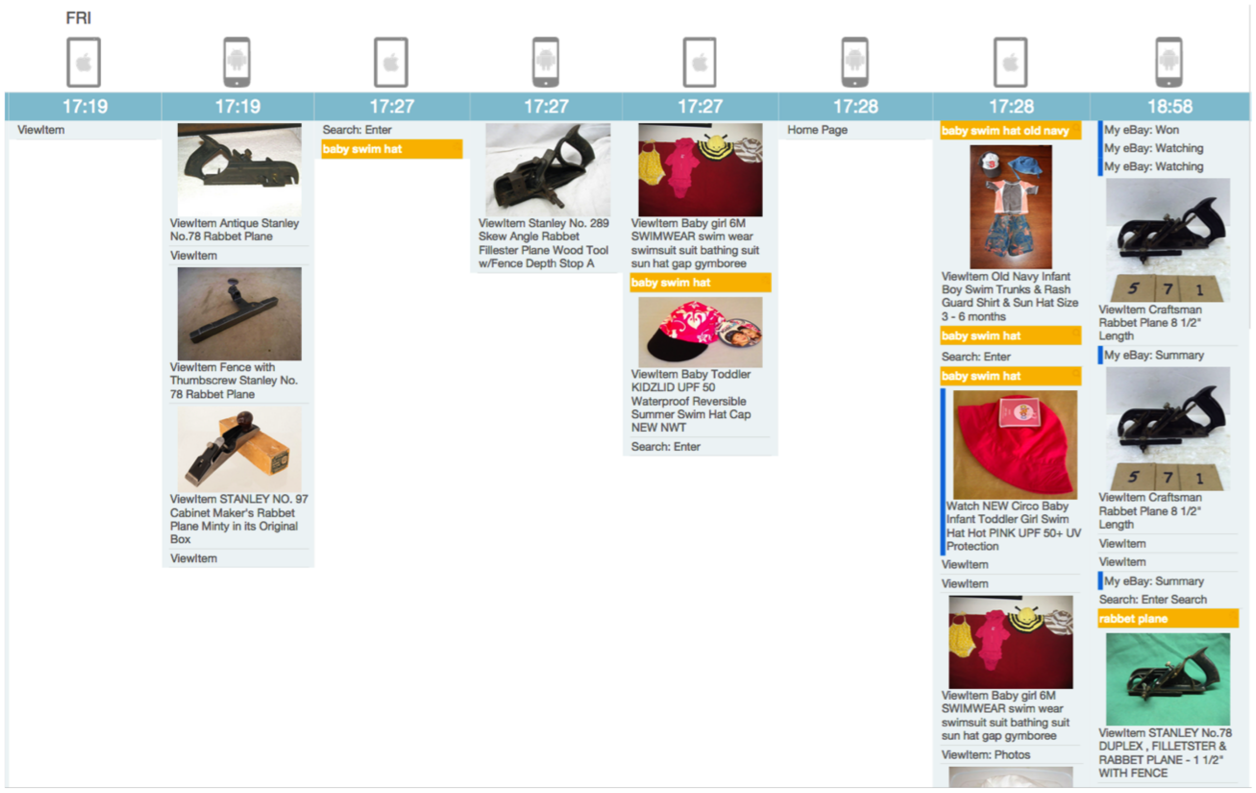
Figure 6.1: "The speaker guy" shopping case

is a typical example of a multi-screen shopping activity. This buyer actively uses both: a laptop and a mobile device to complete the purchase cycle. We can hypothesize, that he is interested to buy speakers. At first, the buyer does not know which exactly speakers he looking for. And then finally he spotted a pair, which he ended up buying. However, he did not purchase them right away. Before that happened, he spent four days deciding whether this is something what he needs, and in good price range. We can derive a bunch of insights about his behavior.

- **revisitation:** speakers are not, say, pair of scissors: they are more expensive, and there are more parameters which you need to check, before you make a purchase, if you do not want to regret about your action. Revisitation happens on both laptop and mobile. What is really interesting is the time. The buyer uses a laptop when he has a bit of time (in the morning, when he is still at home, at work at the end of the day). During the day he uses a mobile device to check on an item, when he gets a few minutes.
- **device usage pattern:** we observe, that on average buyer sessions are longer on a laptop, than on mobile. By length we assume the number of different actions executed (here: the number of items visited). Thus we conclude, that a buyer wants to achieve some meaningful outcome, but does not want to spend much effort. And there might be good reasons for that. A buyer might be waiting on a bus stop, or between meetings, or enjoying a short walk after lunch, and he has a few minutes, so he goes to the mobile app, checks a few items, maybe something more. For instance, if a buyer engaged in an auction, he wants to check, whether he was outbid by someone else.
- **preferred device:** finally the buyer purchases the speakers. However, he prefers to do it on his laptop, not mobile. Despite the fact, that he checked those speakers a minute earlier on his mobile app. This signifies, that some buyers want to use familiar interface, to make sure an important transaction goes through.

This scenario demonstrates an opportunistic behavior: the buyer engages with the site when he has an opportunity, either with a laptop, or mobile. And there might be many sessions during the day, because of the mobile component, which varies throughout the day. We have no means to predict how many times he will come back to the site or the mobile app. Thus the old model, when we use the prediction horizon, is fundamentally flawed, because we assumed that we know how many sessions we should expect.

We also observed, that a buyer revisits the item many times before the final decision. Moreover, the list of items he is comparing is getting shorter, as he is getting closer to the final decision.


FRI

17:19
ViewItem

17:19
ViewItem Antique Stanley No.78 Rabbet Plane
ViewItem
ViewItem Fence with Thumbscrew Stanley No. 78 Rabbet Plane
ViewItem STANLEY NO. 97 Cabinet Maker's Rabbet Plane Minty in its Original Box
ViewItem

17:27
Search: Enter
baby swim hat

17:27
ViewItem Stanley No. 289 Skew Angle Rabbet Fillester Plane Wood Tool w/Fence Depth Stop A

17:27
ViewItem Baby girl 6M SWIMWEAR swim wear swimsuit suit bathing suit sun hat gap gymboree
baby swim hat
ViewItem Baby Toddler KIDZLID UPF 50 Waterproof Reversible Summer Swim Hat Cap NEW NWT
Search: Enter

17:28
Home Page

17:28
baby swim hat old navy
ViewItem Old Navy Infant Boy Swim Trunks & Rash Guard Shirt & Sun Hat Size 3 - 6 months
baby swim hat
Search: Enter
baby swim hat
Watch NEW Circo Baby Infant Toddler Girl Swim Hat Hot PINK UPF 50+ UV Protection
ViewItem
ViewItem
ViewItem Baby girl 6M SWIMWEAR swim wear swimsuit suit bathing suit sun hat gap gymboree
ViewItem: Photos

18:58
My eBay: Won
My eBay: Watching
My eBay: Watching
ViewItem Craftsman Rabbet Plane 8 1/2" Length
My eBay: Summary
ViewItem Craftsman Rabbet Plane 8 1/2" Length
ViewItem
ViewItem
My eBay: Summary
Search: Enter Search
rabbet plane
ViewItem STANLEY No.78 DUPLEX , FILLETSTER & RABBET PLANE - 1 1/2" WITH FENCE

Figure 6.2: Family account sharing

6.1.2 Account sharing

We have discovered, that the same account can be used by multiple people. **Figure 6.2** demonstrates one such case. We observe that the mobile app is open on two devices at the same time, and actions on those devices being executed almost at the same time. More importantly, the types of items being checked on those devices, are completely different. Which supports our belief that these are indeed two different people sharing the account. However, this observation poses another major obstruction on our goal. Particularly, how to model a session ? And what exactly a session is? If we use our previous definition of a session from **Chapter 5** that two sessions are sequences of activities, with a break between those activities for more than 30 minutes, we are facing a problem of modeling sessions with multiple intents in a session, which goes against the assumption, which we had in the previous chapter.

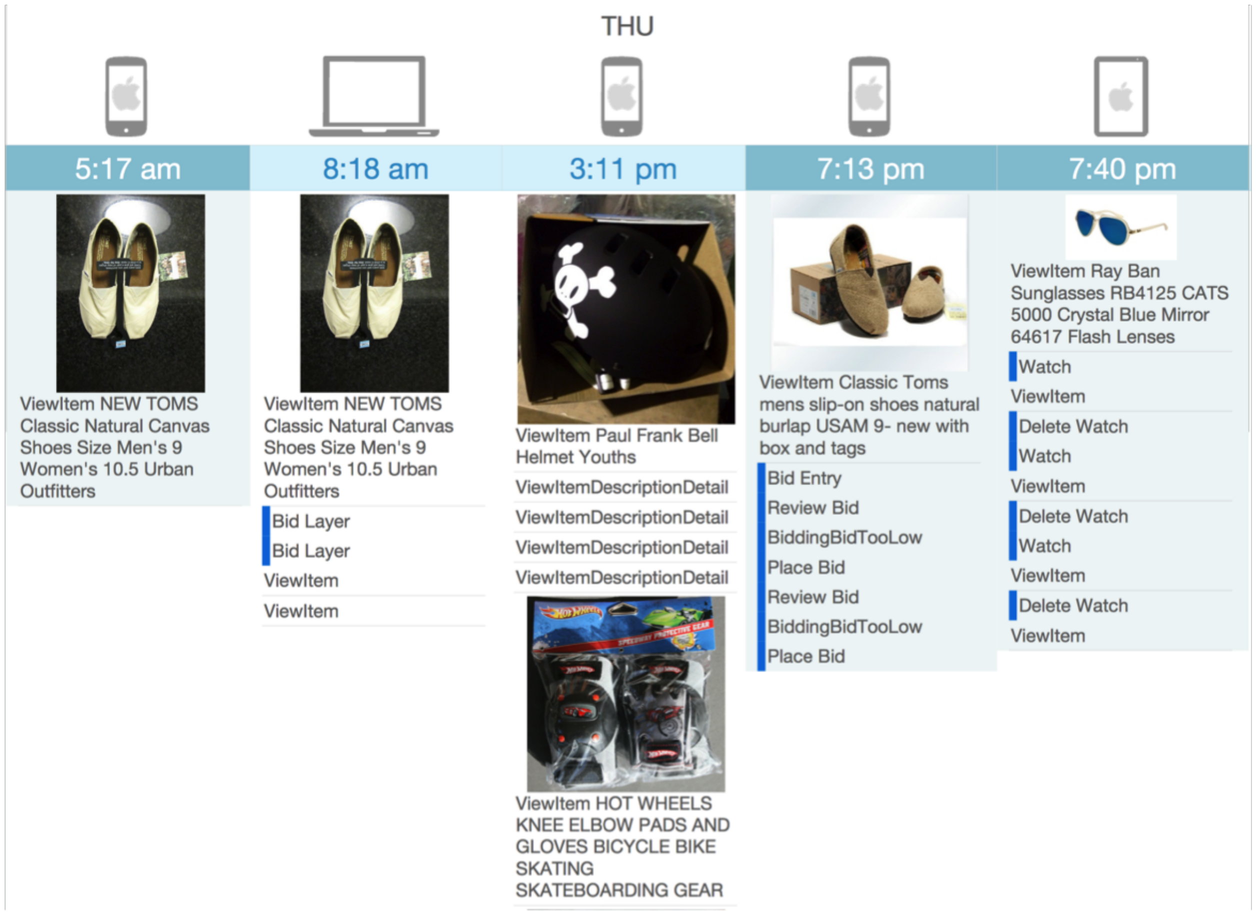

THU
5:17 am
8:18 am
3:11 pm
7:13 pm
7:40 pm
ViewItem NEW TOMS Classic Natural Canvas Shoes Size Men's 9 Women's 10.5 Urban Outfitters
ViewItem NEW TOMS Classic Natural Canvas Shoes Size Men's 9 Women's 10.5 Urban Outfitters
Bid Layer
Bid Layer
ViewItem
ViewItem
ViewItem Paul Frank Bell Helmet Youths
ViewItemDescriptionDetail
ViewItemDescriptionDetail
ViewItemDescriptionDetail
ViewItemDescriptionDetail
ViewItem HOT WHEELS KNEE ELBOW PADS AND GLOVES BICYCLE BIKE SKATING SKATEBOARDING GEAR
ViewItem Classic Toms mens slip-on shoes natural burlap USAM 9- new with box and tags
Bid Entry
Review Bid
BiddingBidTooLow
Place Bid
Review Bid
BiddingBidTooLow
Place Bid
ViewItem Ray Ban Sunglasses RB4125 CATS 5000 Crystal Blue Mirror 64617 Flash Lenses
Watch
ViewItem
Delete Watch
Watch
ViewItem
Delete Watch
Watch
ViewItem
Delete Watch
ViewItem

Figure 6.3: Opportunistic mobile usage

6.1.3 Opportunistic mobile usage

Figure 6.3 shows one more interesting case of mobile usage. The buyer discovers an item of interest on his mobile at 5 am. It happens because of the simplicity of action - he does not engage in heavy search activity. Just a habit to switch on the phone (almost instantaneous), and clicking on an app, and checking the updates. Something similar happens during the day - this buyer gets engaged in bidding on an item, when he was outbid. Again, this is a simple action, and does not require much effort. However, it demonstrates a very important issue - mobile provides more opportunity to engage. When you get outbid, sending a push notification is simple and effective, and provides an easy way to react back.

6.1.4 Relatively "heavy" users and their missions

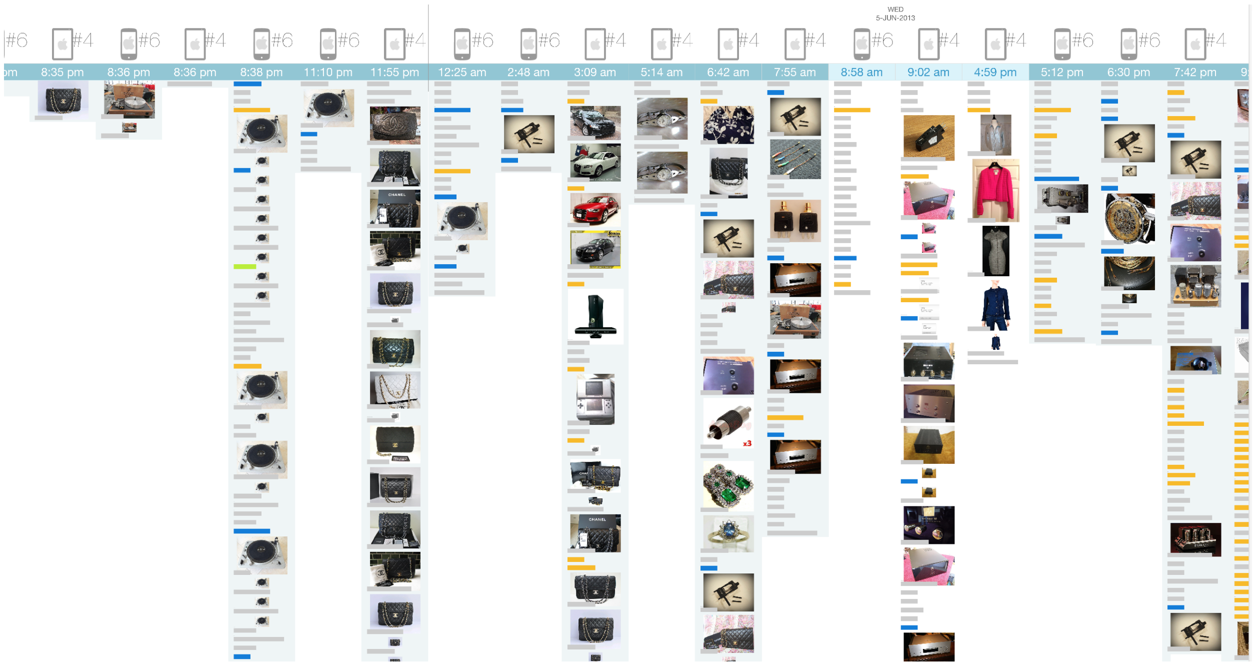


Figure 6.4: Heavy user without a particular shopping goal

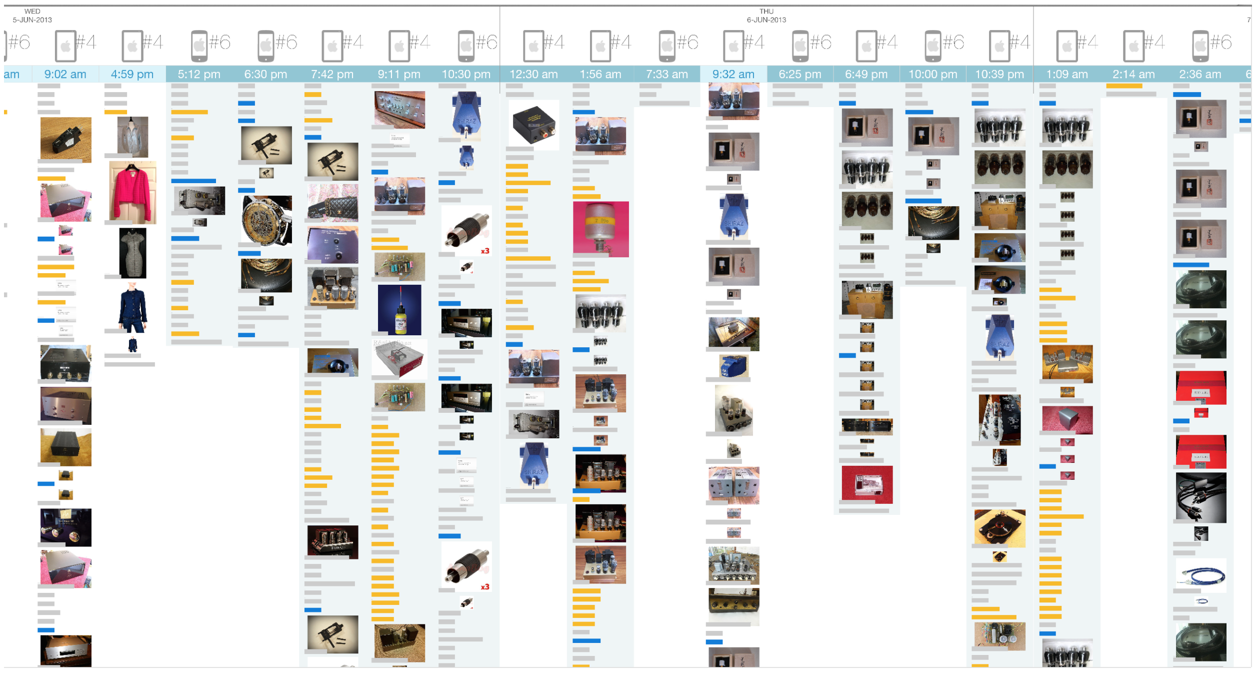


Figure 6.5: Heavy user on a mission

Figures 6.4 and 6.5 visualize shopping activity of the same buyer, but at different times. They reveal a few more interesting aspects of buyer behavior. First of all, this particular buyer uses six different devices to shop on eBay. And he engages with the mobile app very obsessively, some of the intervals between his sessions are only a few minutes apart. Secondly, this buyer does not always have a clue what exactly he is looking for. **Figure 6.4** shows, that he is very diverse in product categories: vinyl records player, cars, clothing, accessories. However, visiting pattern in **Figure 6.5** shows a specific search target, related to finding parts for music equipment. These shopping patterns emphasizes the need to model the diversity, focus and intensity of buyer activity.

6.1.5 Important lessons from multi-screen usage, and combining that knowledge with the previous observations

In this section I showed only a few examples from multi-screen usage pattern. By no means these are all the possible cases. In fact, I have investigated many more, however for brevity I include only a limited number. But even these cases tell a lot about some of the important patterns.

We learned, that the same account may be shared between many people, and at the same time. These can be family accounts or big seller accounts. This observation prevents us from using session summarization technique, described in the previous chapter. We also discovered, that mobile app usually drives opportunistic behavior - people tent to have many short bursts of simple activity on mobile, versus longer and more complicated sessions on laptop. We do not know how many opportunistic sessions each buyer will execute, and thus cannot apply previously used notion of prediction horizon. However, some buyers or sellers may be mobile-only, and have long sessions on their mobile apps. And finally, we learned that buyers tend to revisit items before they make their purchasing decision.

These new observations shift the priorities for marketplace optimization tasks. Item revisitation and mobile app opportunistic usage are very important aspects of online shopping on eBay marketplace. And shopping experience optimization task can be viewed as engagement optimization: if we know that the buyer will revisit items in to finalize his decision, we need to assist him with this mission. If the buyer needs to spent a lot of effort on his mobile device to revisit previously seen items and search for related ones, he may loose the interest in purchase. But we also do not want to show him items he has lost interest (unrelated items, or he has bought what he wanted, and now). If his session on mobile lasts a few minutes, we do not want a buyer to retype his search queries, or show unrelated suggested items.

What we are trying to optimize: to predict whether the buyer will resume his browsing session or not, so we can optimize the user interface.

6.2 APPLYING ROCAS SCHEMA TO THE PROBLEM

From the steps, described in ROCAS model, here we utilize **notable change detection** principle the most. Via visual explorations, and the results of the discussions, presented in **Section 5.7**, we discovered different behavioral scenarios, which are present on the marketplace. We use the most typical scenarios as a test bed to derive the feature patterns, which can capture typical behavior.

6.3 RELATED WORK

To the best of our effort, we were not able to locate a piece of work, directly related to predicting online buyer behavior. However, we found a group of papers, related to modeling online search continuation. For instance, Aiello et al. [2011] touches the problem of topic mining in search queries. This approach is not directly applicable to our task, since we do not know what our "behavioral words" are. Shen et al. [2012] is concerned with personalized click model and talks about collaborative filtering approach. However, this model does not utilize time at all. Another group of research, presented in White and Dumais [2009], Wang et al. [2013], Savenkov et al. [2013], Agichtein et al. [2012], and Kotov et al. [2011] concentrated on deriving all sorts of possible statistics from the browsing behavior as a set of features to predict search query resumption. These results are most closely related to the task we are trying to address, but they do not claim what sort of statistics are the best to use to describe the dynamics of online behavior.

6.4 MODEL DESCRIPTION

Example session visualizations, performed in **Section 6.1** demonstrate that different buyers have different intensity, diversity, and focus of their activity. We have shown in previous chapters, that we can derive good features by hand, which capture complex dynamics happening in the system. However, our ability to derive features by hand is limited. The problem is very complex, as we showed in exploratory visualization analysis in **Section 6.1**. We will try another approach: generate candidate features and then let the learning method choose which features or combinations of features are more important.

We observed in **Section 6.1** that we need to capture not just the stationary statistics, like average or sum / count of the events, but also the dynamics of the process. Our method for feature generation was inspired by Histograms of Oriented Gradients approach Dalal and Triggs [2005], Chandrasekhar et al. [2009] - one of the feature generation techniques used in computer vision.

Feature generation process conceptually is shown in **Figure 6.6**. For different positions of time window we compute different statistics. For adjacent time widows, we compute the gradients of the obtained statistics. And combine the results into one feature vector.

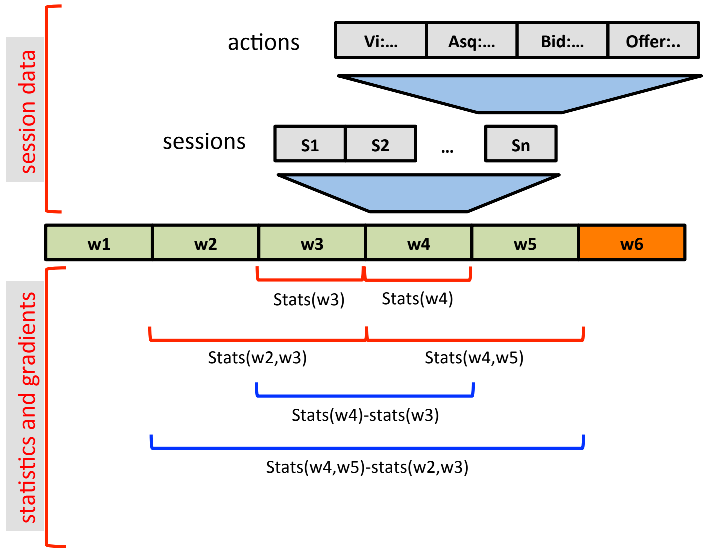


Figure 6.6: Computing features from behavioral data

6.4.1 Dataset description

We have collected a data set from 120000 buyers during their 6 weeks of activity, coming from eBay web site and eBay mobile app. Each week of observations may contain zero or more sessions. Users in this data set had to show reasonable activity in weeks 4 and 5 (at least two sessions in each week). We did not have an exact sequence of actions, which the user executed in a session. Instead, each session was provided a summary of the activities, which happened in this session. For instance, we had an information, which items user viewed (Vi) and for how long, for which items he asked seller a question (Asq), on which items he was bidding (Bid), or bought them right away (BIN), offered new price (Offer), and to which categories those items belong. We use the sessions from weeks 1-5 to build behavioral features, and the data in week 6 to compute, whether the buyer resumed his activity or not. If $S_6 = \{id_{6_1}, id_{6_2}, \ldots, id_{6_n6}\}$ - is a set of item IDs, for which a buyer applied Bid, BIN, Asq, Offer actions in week 6. And $S_{1-5} = \{id_{1_1}, \ldots, id_{1_n1}, \ldots, id_{5_1}, \ldots, id_{5_n5}\}$ is a set of item IDs for which a buyer applied Vi, Bid, BIN, Asq, Offer actions in weeks 1-5. Then we say, that a buyer resumed a meaningful activity in week 6, if $S_{1-5} \cap S_6 \neq \emptyset$. Otherwise he did not.

6.4.2 Feature generation

The most related approaches in terms of feature engineering, are Agichtein et al. [2012], Wang et al. [2013] and Savenkov et al. [2013]. While the prediction accuracy results, reported in these papers are impressive, the feature engineering process is tedious: authors report on designing a number of hand-crafted features, which were of all sorts of statistics from the event stream. These authors do not claim any feature optimality, and to the best of our knowledge, we have not seen a single paper, which provides a reasoning what is the best way to generate features for behavioral data. We adopt the same strategy, described in Agichtein et al. [2012],Wang et al. [2013] and Savenkov et al. [2013] for our feature generation process, with two differences:

- in our task there is no prior information about how time factor impacts features. Thus, we use variable-length and variable position sliding window, as shown in **Figure 6.6** and **Algorithm 3** to generate statistics from the session sequence

- we also compute changes (gradients) in statistics for different adjacent intervals, as shown in **Figure** 6.6 and **Algorithm** 4

```
features = [];
for $start \leftarrow 1$ to 5 do
    for $end \leftarrow start$ to 5 do
        features.append(statistics(start, end));
    end
end
```

Algorithm 3: Computing features via basic statistics

```
features = [];
for $start \leftarrow 1$ to 5 do
    for $end \leftarrow start$ to 5 do
        for $middle \leftarrow start$ to $end$ do
            if $start < end$ then
                features.append(statistics(start, middle)-statistics(middle+1,end));
            end
        end
    end
end
```

Algorithm 4: Computing features via gradients of basic statistics

Table 6.1 summarizes what kind of statistics we compute.

For each feature type FT_i from **Table** 6.1, we generate a series of features for different time widows $[start, end]$ where *start* and *end* combinations are generated using **Algorithm** 3. We also add gradients of the same features, applying **Algorithm** 4. For each feature type FT_i the combined feature sets we call FS_i. Each feature set is then pruned using correlation-based feature selection routine in Weka Hall et al. [2009], and we obtain PFT_i feature sets. Which then put together into a set $PFT = \cup_i PFT_i$. Set PFT contains 167 features. Then we prune PFT set again using correlation-based feature selection routine in Weka, and obtain another set $PPFT$, which contains 53 features.

Table 6.1: Statistics for feature generation, computed for a time window $\left[start, end\right]$, where *start* and *end* - week number

feature type	**explanation**
session entropy	extract all ItemIDs = IDS, output entropy(IDS)
auction/FP ratio	extract all unique items, and compute the ratio
determined	extract all ItemIDs = IDS, obtain corresponding item categories for IDS = CIDS and categories for de-duplicated (IDS) = CDIDS, and finally compute entropy(CIDS)-entropy(CDIDS)
purchase count	compute the number of bought items
purchase effort completion ItemID	extract ItemIDs in all weeks 1-5 = IDS. for each ItemID which was bought: $ItemID \in IDB$, compute $cID_B = \forall ItemID \in IDS \rightarrow 1if(ItemID == ItemID_B)$, compute : $B = \sum_{ItemID \in IDB} cID_B$, **output** : $B/count(IDS)$
same for category	same as for ItemID
item revisits	for a given threshold, consider item was frequent, if number of times it was touched $\geq$ *threshold*. IDF = frequent items. for each item in IDF, compute how many sessions contained this item output the largest number
category revisits	same as for items

6.5 EXPERIMENT RESULTS

Experimental results were obtained for two feature sets, mentioned in **Section 6.4**: $PPFT$ or **Set 1**, containing 53 features, which is a reduced set of features from **Set 2**, and PFT or **Set 2**, containing 167 features. We have applied the following learning techniques: logistic regression (LR), random forest (RF), decision trees (J48) using Weka Hall et al. [2009], L1-regularized logisitic regression (L1-LR), linear SVM (SVM) using liblinear Fan et al. [2008],

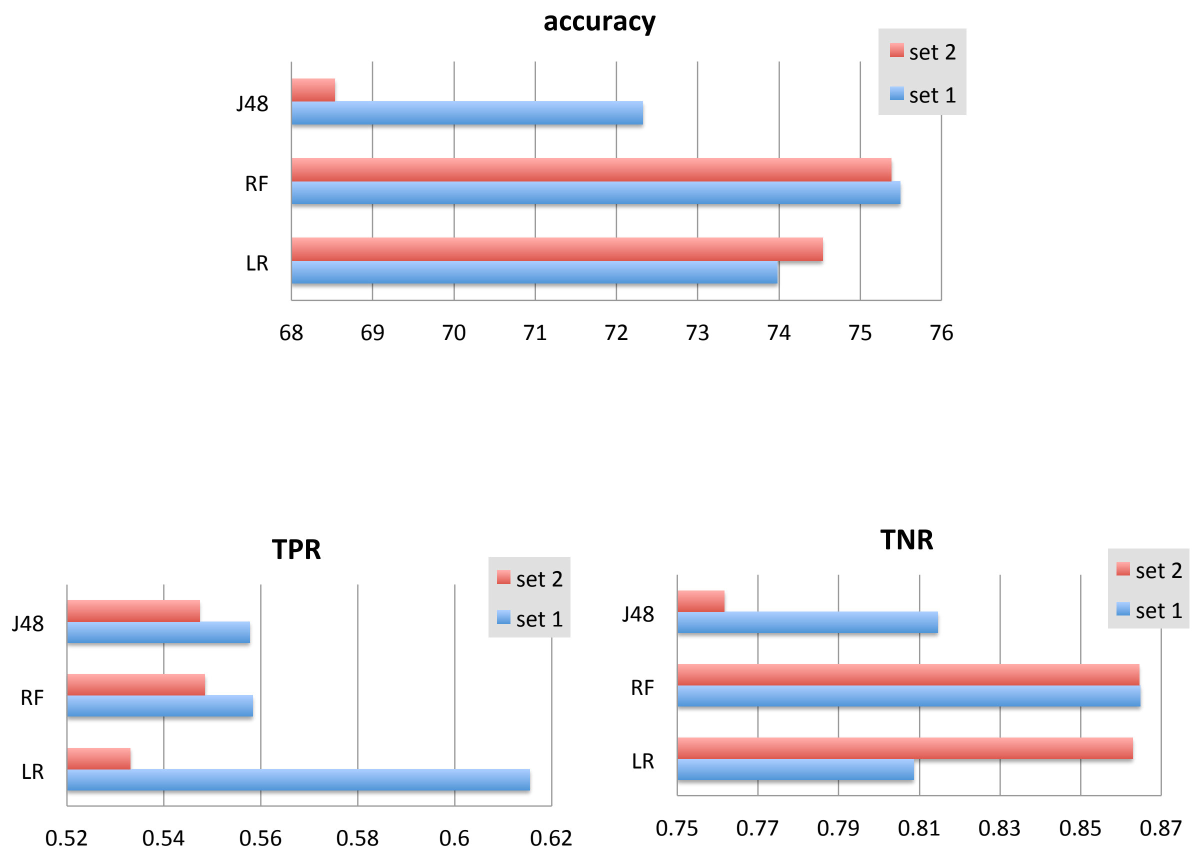
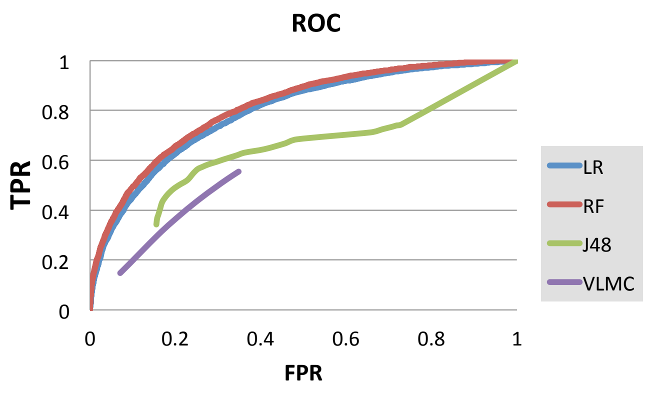


Figure 6.7: Results obtained on feature set 1 and set 2 using LR, RF, and J48

kernel SVM (gaussian kernel) (kSVM) using libsvm Chang and Lin [2011].

At first we investigated how feature selection mechanism impacts the performance. For this purpose, we applied the same learning mechanisms: RF, LR, and J48 to two feature sets: set 1 and set 2, where set 1 obtained from set 2 by applying correlation-based feature selection mechanism. The resulting performance is shown in **Figure 6.7**. These results are not surprising. Random forest demonstrates the most robust result: the performance almost does not change (RF finds the best feature subsets). Decision tree usually demonstrates moderate to high variance, and benefits the most from feature pruning. Logistic regression suffers from feature pruning, which is also not very surprising: attributes may have complex interactions Jakulin and Bratko [2004], and removing even correlated attributes may impact the performance.



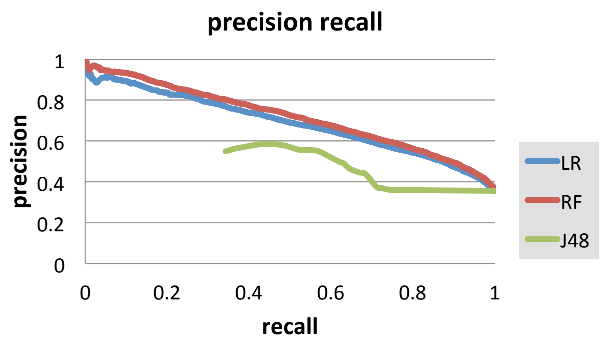


Figure 6.8: Results obtained on feature set 2 using L2-LR, RF, J48

For better analysis, we build ROC curve and precision/recall curves for LR, RF, and J48 learning methods, computed on feature set 2. The results are shown in **Figure 6.8**.

- **Conclusion 1:** RF performs better than LR, however not by much. However, it takes only a small fraction of time to train a LR vs RF.
- **Conclusion 2:** histograms of gradients applied for feature generation, helps to produce better prediction, than the approach based on VLMC, obtained in **Chapter 5**.

In the next comparison we used feature set 2 with all features, to investigate how the accuracy of the prediction depends on the choice of a machine learning method. **Figure 6.9** summarizes the results. J48 decision trees show the worst performance (because of high variance). We observe that L1-regularized LR demonstrates slightly better performance, than L2-regularized LR. This suggests, that the input space is in fact sparse. Kernel SVM approach showed worse performance than even L2-LR. Given the size of the data set, we

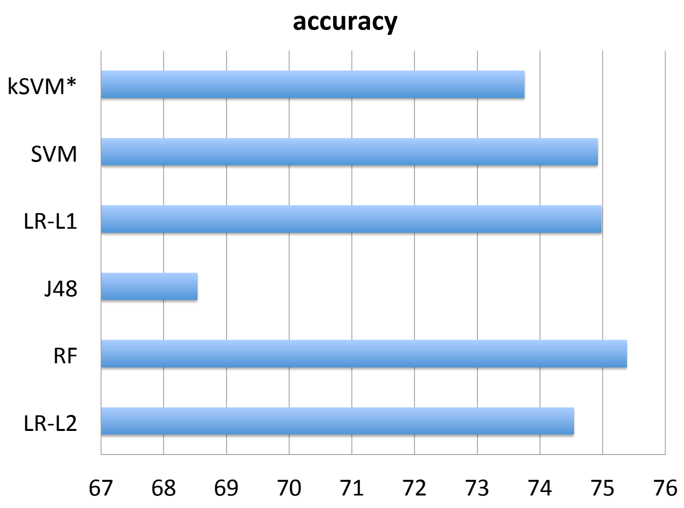


Figure 6.9: Results obtained on feature set 2 using L2-LR, RF, J48, L1-LR, SVM and kSVM (used only 10 % of data due to time issue)

were not able to train it on 100% of data: we handled only about 10%, which can explain why the results are worse.

6.6 CONCLUSION

In this chapter I proposed a new approach to address the issue of buyer behavior prediction on eBay marketplace. The need for a new prediction method was dictated by the reality that becomes virtually impossible to design high-quality features by hand. This method utilizes automatic feature generation using feature patterns, which in turn are based on a principle, very close to histograms of gradients approach. I showed, that the obtained accuracy is better, compared with the results, obtained in **Chapter 5**.

Moreover, I investigated whether the choice of a machine learning method has a significant impact on the accuracy of the prediction. The results suggest, that (except of the obvious outliers: decision trees (high variance) and kernel SVM (very slow, does not scale to our data set)) methods produce very similar results. We did not have a chance to train any deep neural networks, thus cannot make a conclusion regarding deep learning. However, based on applied learning techniques we observe, that feature engineering plays more important role than the choice of a machine learning method. And histograms of gradients (HOG) approach can be applied to the task of feature generation to capture different aspects of the dynamics in buyer behavior.

Yet we do not claim that HOG is the only way to generate features, neither that it is the optimal one. A substantial effort has to be put to address the issue of optimality of feature generation.

7.0 THESIS CONCLUSION

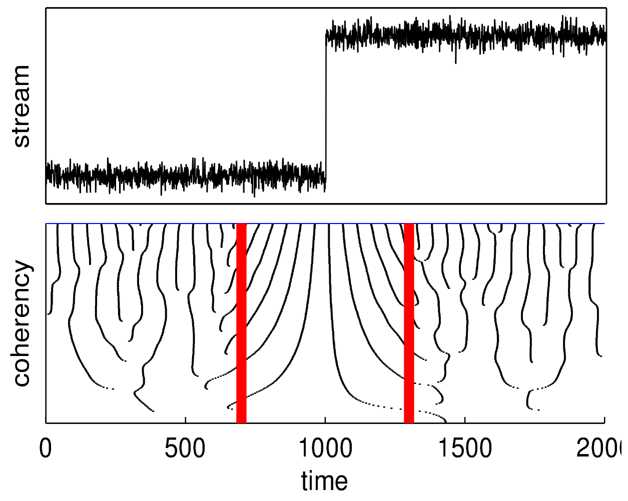
The validity of ROCAS model is tightly connected to the answers to the research questions, presented in **Section 1.4**. Now is the time to answer to those questions.

- **Question 1:** Can we analyze global system properties based on observable local interactions?
- **Answer:** Yes, we can. For instance, in **Chapter 4** I show that we can derive CAS properties based on agent interactions and system constraints in a highly-dynamic environment, like cluster computing. I also show that these properties can be used to optimize resource sharing between different processes and obtain a significant (about 40%) improvement in execution speed.
- **Question 2:** How to capture behavioral patterns that correspond to notable transitions in system dynamics?
- **Answer:** Feature engineering is a very tedious task: it takes a lot of time and effort to design good features. Alternatively, by applying ROCAS model, first we concentrate on the corner cases and require the proposed method to perform on those cases, as discussed in **Chapter 3**. To deal with the complexity of the patterns, we define a relatively simple cost function, which can combine different patterns into a single number (score). And that score can be used to detect notable transitions.
- **Question 3:** How to use CAS behavioral patterns to perform relaxed large-scale optimization?
- **Answer:** By applying ROCAS method, we can design a set of features, which are good enough to capture complex processes in a CAS, as I show in **Chapter 6**.

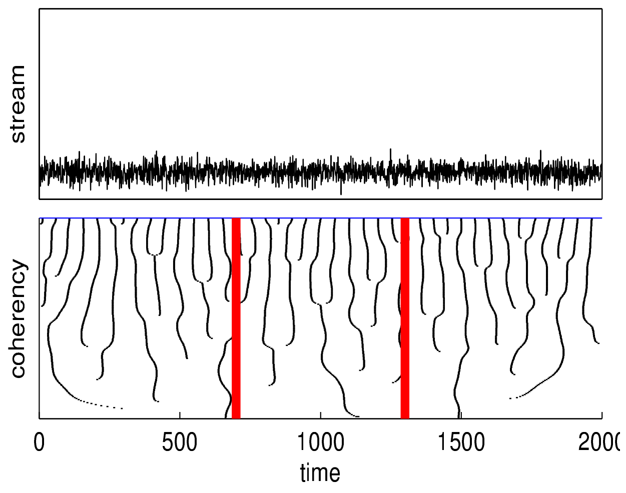
APPENDIX A

COHERENCY PORTRAIT ANALYSIS

Consider two data streams with identical additive normal noise. One stream reflects a transition (**Figure 1(a)**), another one does not (**Figure 1(b)**). Away from the transition point the corresponding coherency portraits for both streams look similar. However, the coherency portrait around the transition (between vertical lines) in **Figure 1(a)** is different from the corresponding portrait in **Figure 1(b)**. For a signal with a transition we observe a set of smooth V-shapes. This pattern can be used as a signature for a transition.



(a)



(b)

Figure A1: Coherency portrait for: (a) stream with a transition; (b) stream without a transition

In this appendix we will show that:

1. coherency portraits for an abrupt transition will produce a set of smooth V-shapes around the transitions;
2. coherency portraits are more volatile further away from a transition point;
3. coherency portraits at lower frequencies are more stable and better candidates for transition detection.

First, we introduce some properties of a bandpass filter.

A.1 ZERO-PHASE BANDPASS FILTER

The essential tool for harmonic alignment is a digital filter. A digital filter (DF) reduces or enhances certain components of a sampled, discrete-time signal. For an input signal **x** DF produces an output **y** by applying *delay*, *multiply* and *add* operations. The filter can be described using a difference equation by mapping *multiply* and *add* operations to mathematical expressions as follows:

$$\begin{aligned} y_n = b_1 * x_n + b_2 * x_{n-1} + \cdots + b_{n_b+1} * x_{n-n_b} \\ - a_2 * y_{n-1} - \cdots - a_{n_a+1} * y_{n-n_a} \end{aligned} \tag{A.1}$$

When there is no feedback loop (i.e., $a_i = 0$) we say that the filter is non-recursive and has a finite impulse response. Our method utilizes a non-recursive band-pass filter (i.e., it passes frequencies within a certain range and rejects frequencies outside that range).

A.2 EXISTENCE OF V-SHAPE

Let us consider a simple example of filtering Heaviside step function:

$$y(x) = \begin{cases} 0: & x < 0 \\ 1: & x \geq 0 \end{cases} \tag{A.2}$$

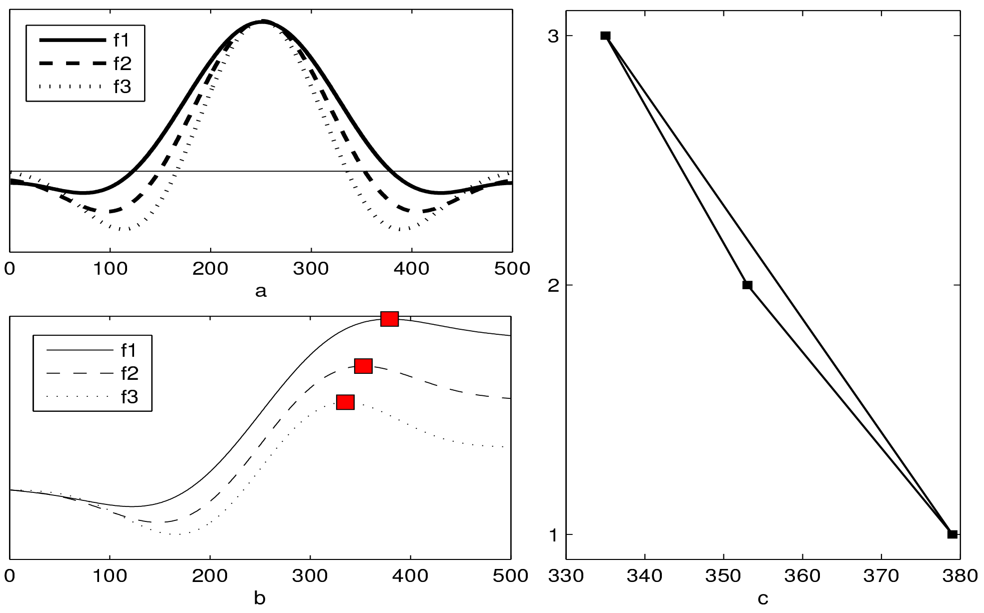


Figure A2: Filtering Heaviside step function: (a)B-vectors for band-pass filters; (b):filtered bands for Heaviside function; (c): coherency portraits

using a bandpass filter **eq. A.1**. Let us assume that the order of a band-pass filter is N (i.e., the number of coefficients $b_i | i \in [1 \ldots N]$). When the filter is positioned far to the left from the transition (x=0), the output of the filter is constant $y_- = 0$, since $x_i = 0, \forall i \in [1 \ldots N]$. If the filter is positioned far to the right from the transition, the output of the filter is also constant $y_+ = \sum_{i=1}^{N} b_i$, since $x_i = 1, \forall i \in [1 \ldots N]$. Thus, perturbations in filtered signal occur around the transition point $x = 0$ and the output of the filtered signal is a partial sum of the filter coefficients $y_0 = \sum_{i=1}^{n} (b_i | x_i = 1), n \leq N$. As we move a band-pass filter through the transition point more signal components are getting equal to one ($x_i = 1$), causing the output y to change until it reaches saturation of y_+. To demonstrate the behavior of the filtered signal, we built a set of bandpass filters adjacent in frequency bands (order $N = 500$, sampling frequency $SF = 1000$ Hz and frequency band $dF = 0.1$ HZ). We have picked 3 lower frequency filters to illustrate the concept. The corresponding B coefficients are shown in **Figure A2 (a)**. Here $f_1 < f_2 < f_3$ represent frequency bands; f_1 corresponds to the lowest frequency. The output of the filtered function for all 3 filters is shown in **Figure A2 (b)**. Red squares on **Figure A2 (b)** correspond to local maxima of functions. We plot those maxima on **Figure A2 (c)**, keeping the same x-coordinate as in **Figure A2 (b)**, but assigning y=1,2,3 to the output of filter f_1, f_2, f_3 correspondingly. The output appears to be a concave function. Band-pass filters for higher frequencies have more extrema. After filtering a step function the obtained extrema will form a set of concave functions. **Thus a transition in the input stream corresponds to a set of concave functions on the coherency portrait**.

A.3 STABILITY OF COHERENCY PORTRAITS AROUND THE TRANSITION POINT

When we try to build a coherency portrait for a noisy signal which contains a transition, it is important to know how noise influences coherency portrait. To answer this question we explore the volatility of the locations of maxima on the filtered bands. We assume that after applying a bandpass filter to a signal we will obtain at least one local maximum for a band. Without the loss of generality, we assume Gaussian noise $N(0, \sigma^2)$. We will consider two data streams. The first one is simply a normal noise **eq. A.3**

$$y(x) = N(0, \sigma^2), x \in (-\infty, \infty) \tag{A.3}$$

$$y(x) = \begin{cases} N(0, \sigma^2) : & x < 0 \\ N(1, \sigma^2) : & x \geq 0 \end{cases} \tag{A.4}$$

The second one is heaviside function with the same normal noise: **eq. A.4**. Both streams are plotted on **Figure A3, (right)**. We will filter them with a band-pass filter with B vector coefficients shown in **Figure A3, (left)**. Vertical lines partition filter coefficients into four groups: F_1, F_2, F_3, F_4. In F_1 and F_2 all $b_i \leq 0$. In F_3 all $b_i \geq 0$, in F_4 all $b_i \leq 0$. Let us position a filter around the transiton **Figure A3, (right)** so that it produces a local maximum at the output. The position of the filter should be as follows: groups F_4,F_3 and F_2 should be to the right from the transition point (jump in Heaviside step function), while group F_1 to the left from the transition. The output of the filter positioned in the mentioned

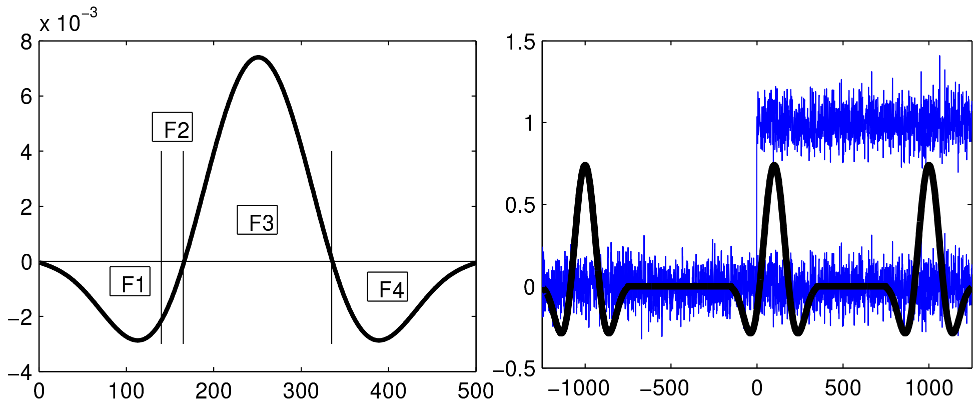


Figure A3: Filtering a noisy signal

way is:

$$y = \sum_{i \in F_1} b_i N(0,\sigma^2) + \sum_{i \in F_2} b_i (1 + N(0,\sigma^2)) + \\ \sum_{i \in F_3} b_i (1 + N(0,\sigma^2)) + \sum_{i \in F_4} b_i (1 + N(0,\sigma^2)) \quad (A.5)$$

Performing equivalent transformations obtain

$$y = \sum_{i \in F_2 \cup F_3 \cup F_4} b_i + \sum_{i \in F_1 \cup F_2 \cup F_3 \cup F_4} b_i N(0,\sigma^2) = \\ -A_{F_2} + A_{F_3} - A_{F_4} + A_N = D + A_N \quad (A.6)$$

where

$$A_{F_2} = \sum_{i \in F_2} \|b_i\|,\ A_{F_3} = \sum_{i \in F_3} \|b_i\|,\ A_{F_4} = \sum_{i \in F_4} \|b_i\|$$
$$D = -A_{F_2} + A_{F_3} - A_{F_4}$$
$$A_N = \sum_{i \in F_1 \cup F_2 \cup F_3 \cup F_4} b_i N(0,\sigma^2) \quad (A.7)$$

A_{F_2}, A_{F_3}, A_{F_4} depend on the characteristics of the filter and do not depend on the input stream. If we apply a band-pass filter to a stream without a transition (e.g. **eq. A.3**), **eq. A.6** becomes

$$y = A_N, \quad (A.8)$$

which is essentially **eq. A.6** with $D = 0$. A_N is the sum of random values and thus a random value itself. We explore how a random fluctuation in A_N influences the output of the filter. Assuming that $A_N = A_{N0} + \Delta$, where Δ is a small fluctuation on top of A_{N0}, from **eq. A.6, A.8**, relative variation becomes

$$V_h = \frac{D + A_{N0} + \Delta}{D + A_{N0}}; \quad V_{pn} = \frac{A_{N0} + \Delta}{A_{N0}} \tag{A.9}$$

where V_h and V_{pn} are relative variations for streams with transition (noisy Heaviside) and without a transition (pure noise). Assuming that $\Delta << A_{N0} << D$, we obtain

$$V_h = 1 + \frac{\Delta}{D}; \quad V_{pn} = 1 + \frac{\Delta}{A_{N0}} \tag{A.10}$$

From **eq. A.10**: since $A_{N0} << D$, relative variation for the output of a bandpass filter is much smaller when we filter a signal with transition than when we filter just pure noise. **Thus around the transition point we expect to observe smoother coherency portrait**

A.4 STABILITY OF COHERENCY PORTRAITS OUTSIDE THE TRANSITION POINT

Let us position our band-pass filter outside of the transition: (e.g. **Figure A3 (right)**). When the filter is positioned far to the left from the transition, then the equation for filter output **eq. A.6** becomes essentially **eq. A.6**. Thus, in terms of relative variation **eq. A.10**, coherency portrait father to the left of the transition shows the same volatility as a coherency portrait built for a stream without a transition. Now let us position our band-pass filter far to the right from th transition. Then **eq. A.6** parameter D becomes $D' = D - A_{F_1}$ where $A_{F_1} = \sum_{i \in F_1} \|b_i\|$. Thus $D' < D$. From **eq. A.10**, relative variation is higher when the output of the filter depends on a smaller value of D. this means that far to the right the coherency portrait is more volatile that around the transition. **Thus, around the transition point we expect coherency portrait to be smooth; far from the transition it does not have this property.**

A.5 STABILITY OF COHERENCY PORTRAITS FOR DIFFERENT FREQUENCIES

From **eq. A.10**, relative variation is inverse proportional to parameter D. Parameter D from **eq. A.7**, $D = -A_{F_2} + A_{F_3} - A_{F_4}$. While $A_{F_2} << A_{F_3}, A_{F_4}$, $D \approx A_{F_3} - A_{F_4}$. For higher frequencies (compare filters f_1, f_3 on **Figure A2**) the corresponding A_{F_3} decreases while A_{F_4} increases, thus D decreases parameters A_{F_3}, A_{F_4} defined in **eq. A.7**, **Figure A3 (left)**). **Thus for higher frequencies coherency portrait demonstrates more volatile behavior than for lower ones**.

APPENDIX B

LIKELIHOOD RATIO TEST AND KL-DIVERGENCE

Likelihood ratio test statistic can be written as:

$$\lambda(\mathbf{x}) = \frac{L(\hat{\theta}_0|\mathbf{x})}{L(\hat{\theta}|\mathbf{x})} \tag{B.1}$$

Suppose $x_1, x_2, \dots x_n \sim i.i.d.q(x)$. From **eq. B.1**, the log-likelihood ratio, normalized by dividing by n is

$$\hat{\lambda}_n = \frac{1}{n}\sum_{i=1}^{n} \log\frac{p_0(x_i)}{p_1(x_i)} \tag{B.2}$$

Then, computing the expectation of **eq. B.2**, we obtain:

$$\begin{aligned}
E[\hat{\lambda}_n] &= \frac{1}{n}\sum_{i=1}^{n} E[\hat{\lambda}_i] = \frac{1}{n} n E[\hat{\lambda}_1] = E[\hat{\lambda}_1] \\
&= \int \log\frac{p_0(x)}{p_1(x)} q(x)dx = \int \log\frac{p_0(x)q(x)}{p_1(x)q(x)} q(x)dx \\
&= \int \left[\log\frac{q(x)}{p_1(x)} - \log\frac{q(x)}{p_0(x)}\right] q(x)dx \\
&= D(q||p_1) - D(q||p_0) \qquad \text{(B.3)}
\end{aligned}$$

where $D(q||p_1)$ and $D(q||p_0)$ are KL-divergence between distributions q and p_1 and q and p_0 respectively. Because $x_1 \dots x_n$ were sampled from the new window, $q = p_1$. Thus, plugging $q = p_1$ in **eq. B.3** obtain

$$E[\hat{\lambda}_n] = 1 - D(p_1||p_0) \tag{B.4}$$

From the law of large numbers, we know that

$$E[\hat{\lambda}_n] \xrightarrow{a.s.} \hat{\lambda}_n \tag{B.5}$$

Combining **eq. B.2**, **eq. B.3** and **eq. B.5**, obtain

$$\hat{\lambda}_n \xrightarrow{a.s.} 1 - D(p_1||p_0) \tag{B.6}$$

that for large sample likelihood ratio test is essentially KL-divergence

BIBLIOGRAPHY

C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 575–586, New York, NY, USA, 2003. ACM. ISBN 1-58113-634-X. doi: 10.1145/872757.872826. URL http://doi.acm.org/10.1145/872757.872826.

E. Agichtein, R. W. White, S. T. Dumais, and P. N. Bennet. Search, interrupted: Understanding and predicting search task continuation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 315–324, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1472-5. doi: 10.1145/2348283.2348328. URL http://doi.acm.org/10.1145/2348283.2348328.

J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 147–160, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6. doi: 10.1145/1376616.1376634. URL http://doi.acm.org/10.1145/1376616.1376634.

L. M. Aiello, D. Donato, U. Ozertem, and F. Menczer. Behavior-driven clustering of queries into topics. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1373–1382, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063775. URL http://doi.acm.org/10.1145/2063576.2063775.

J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, Lansdowne, VA, USA, Feb. 1998. 007.

Amazon. http://aws.amazon.com/elasticmapreduce/.

A. M. Amey. *Real-time ridesharing : exploring the opportunities and challenges of designing a technology-based rideshare trial for the MIT community.* PhD thesis, Massachusetts Institute of Technology, http://hdl.handle.net/1721.1/61563, 2010.

R. S. Anderssen and P. Bloomfield. Numerical differentiation procedures for non-exact data. *Numerische Mathematik*, 22:157–182, 1972. doi: 10.1007/BF01436965. URL http://dx.doi.org/10.1007/BF01436965.

S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Number 0521424267. Cambridge University Press, 1st edition, April 2009.

J. Arthur F. Veinott. *Lectures in Supply-Chain Optimization.* Department of Management Science and Engineering, Stanford University, Stanford, California 94305, 2005.

R. Axelrod. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration.* Number 0691015678. Princeton University Press, August 1997.

R. Axtell. The complexity of exchange. *The Economic Journal*, 115(504):F119–F210, June 2005.

P. Bak. *how nature works: the science of self-organized criticality.* Copernicus, 1999.

E. Baldeschwieler. Hadoop @ yahoo! - internet scale data processing. In *Cloud Computing Expo*, Santa Clara, CA, USA, Nov 2009.

M. Barnes, H. Leather, and D. K. Arvind. Emergency evacuation using wireless sensor networks. In *Proceedings of the 32nd IEEE Conference on Local Computer Networks*, LCN '07, pages 851–857, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3000-1. doi: 10.1109/LCN.2007.63. URL http://dx.doi.org/10.1109/LCN.2007.63.

R. J. Bayardo, Jr. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 85–93, New York, NY, USA, 1998. ACM. ISBN 0-89791-995-5. doi: 10.1145/276304.276313. URL http://doi.acm.org/10.1145/276304.276313.

BBC. http://www.bbc.co.uk/news/business-21759259.

R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order markov models. *J. Artif. Int. Res.*, 22(1), Dec 2004.

J. Berkovici. Uber director defends surge pricing as sxsw riders lament it. http://www.forbes.com/sites/jeffbercovici/2014/03/11/uber-director-defends-surge-pricing-as-sxsw-riders-lament-it/, March 2014.

D. Berry, A. Usmani, J. L. Torero, A. Tate, S. McLaughlin, S. Potter, A. Trew, R. Baxter, M. Bull, and M. Atkinson. Firegrid: Integrated emergency response and fire safety engineering for the future built environment. In *Workshop on Ubiquitous Computing and e-Research*. UK e-Science Programme All Hands Meeting, 2005.

A. Beutel, B. A. Prakash, R. Rosenfeld, and C. Faloutsos. Interacting viruses in networks: can both survive? In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 426–434, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1462-6. doi: 10.1145/2339530.2339601. URL http://doi.acm.org/10.1145/2339530.2339601.

A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*. SIAM, 2007. doi: http://www.siam.org/meetings/proceedings/2007/datamining/papers/042Bifet.pdf.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Number 978-0-387-31073-2. Springer, 2006.

Bloomberg. http://www.bloomberg.com/news/2013-03-06/google-apple-valuation-gap-widest-since-2005-on-ads.html.

J. Borges and M. Levene. Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Trans. on Knowl. and Data Eng.*, 19(4):441–452, 2007.

J. Borges and M. Levene. A comparison of scoring metrics for predicting the next navigation step with markov model-based systems. *International Journal of Information Technology and Decision Making*, 9(4):547–573, 2010.

Z. Botev, J. Grotowski, and D. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38:2916–2957, 2010.

G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2008.

D. Brewer, M. Barenco, R. Callard, M. Hubank, and J. Stark. Fitting ordinary differential equations to short time course data. *Philos Trans A Math Phys Eng Sci.*, 366(1865): 519–544, February 2008.

P. Bühlmann and A. J. Wyner. Variable length markov chains. *Annals of Statistics*, 27(2): 480–513, 1999.

S. Bullock and D. Cliff. Complexity and emergent behaviour in ict systems. Technical report, HP Labs, 2004.

H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 875–883, New York, NY, USA, 2008. ACM.

H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 3–10, Boston, MA, USA, 2009a. ACM.

H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 191–200, New York, NY, USA, 2009b. ACM.

CapacityScheduler. http://developer.yahoo.com/ blogs/hadoop/posts/ 2011/02/capacity-scheduler/.

Cascading. http://www.cascading.org.

G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, 2001.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL http://doi.acm.org/10.1145/1541880.1541882.

B. Chandramouli, J. Goldstein, and D. Maier. High-performance dynamic pattern matching over disordered streams. *Proc. VLDB Endow.*, 3(1-2):220–231, Sept. 2010. ISSN 2150-8097. URL http://dl.acm.org/citation.cfm?id=1920841.1920873.

V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. Chog: Compressed histogram of gradients a low bit-rate feature descriptor. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2504–2511, June 2009. doi: 10.1109/CVPR.2009.5206733.

C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL http://doi.acm.org/10.1145/1961189.1961199.

B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wirel. Netw.*, 8(5):481–494, Sept. 2002. ISSN 1022-0038. doi: 10.1023/A:1016542229220. URL http://dx.doi.org/10.1023/A:1016542229220.

S. Chen, H. Wang, S. Zhou, and P. S. Yu. Stop chasing trends: Discovering high order models in evolving data. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 923–932, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-1-4244-1836-7. doi: 10.1109/ICDE.2008.4497501. URL http://dx.doi.org/10.1109/ICDE.2008.4497501.

A. Cherniak and J. Bridgewater. Session modeling to predict online buyer behavior. In *Proceedings of the 2013 Workshop on Data-driven User Behavioral Modelling and Mining from Social Media*, DUBMOD '13, pages 1–4, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2417-5. doi: 10.1145/2513577.2513583. URL http://doi.acm.org/10.1145/2513577.2513583.

A. Cherniak and V. Zadorozhny. Towards adaptive sensor data management for distributed fire evacuation infrastructure. In *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, pages 151–156, 2010. doi: 10.1109/MDM.2010.53.

A. Cherniak and V. Zadorozhny. Signature-based detection of notable transitions in numeric data streams. *Knowledge and Data Engineering, IEEE Transactions on*, 25(12):2867–2879, Dec 2013. ISSN 1041-4347. doi: 10.1109/TKDE.2012.241.

A. Cherniak, H. Zaidi, and V. Zadorozhny. Optimization strategies for a/b testing on hadoop. *Proc. VLDB Endow.*, 6(11):973–984, Aug. 2013. ISSN 2150-8097. URL http://dl.acm.org/citation.cfm?id=2536222.2536224.

F. Chierichetti, R. Kumar, P. Raghavan, and T. Sarlos. Are web users really markovian? In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 609–618, New York, NY, USA, 2012. ACM.

B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 493–498, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956808. URL http://doi.acm.org/10.1145/956750.956808.

T. Y. Choi, K. J. Dooley, and M. Rungtusanatham. Supply networks and complex adaptive systems: control versus emergence. *Journal of Operations Management*, 19(3):351 – 366, 2001. ISSN 0272-6963. doi: http://dx.doi.org/10.1016/S0272-6963(00)00068-1. URL http://www.sciencedirect.com/science/article/pii/S0272696300000681.

C. Christakos. Sensor networks applied to the problem of building evacuation: An evaluation in simulation. In *Proceedings of the 15th IST Mobile and Wireless Summit*, Mykonos, Greece, June 2006.

C. K. Chui. *An Introduction to Wavelets*, volume 1. Academic Press, 1 edition, 1992.

P. Clayton and P. Davies. *The Re-Emergence of Emergence: The Emergentist Hypothesis from Science to Religion*. Oxford University Press, 2008.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.177. URL http://dx.doi.org/10.1109/CVPR.2005.177.

D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. In *the International Conference on Intelligence Systems*, 1996.

T. Dasu, S. Krishnan, D. Lin, S. Venkatasubramanian, and K. Yi. Change (detection) you can believe in: Finding distributional shifts in data streams. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, IDA '09, pages 21–34, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-03914-0. doi: 10.1007/978-3-642-03915-7_3. URL http://dx.doi.org/10.1007/978-3-642-03915-7_3.

J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327492. URL http://doi.acm.org/10.1145/1327452.1327492.

D. DeWitt and J. Gray. Parallel database systems: the future of high performance database systems. *Commun. ACM*, 35(6):85–98, June 1992. ISSN 0001-0782. doi: 10.1145/129888.129894. URL http://doi.acm.org/10.1145/129888.129894.

D. J. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri. Practical skew handling in parallel joins. In *Proceedings of the 18th International Conference on Very Large Data Bases*, VLDB '92, pages 27–40, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. ISBN 1-55860-151-1. URL http://dl.acm.org/citation.cfm?id=645918.672512.

R. O. Duda, P. E. Hart, and D. Stork. *Pattern Classification.* Wiley-Interscience, 2000.

J. Epstein and R. Axtell. *Growing artificial societies: social science from the bottom up.* Number 978-0-262-55025-3. Brookings Institution Press, 1996.

D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 263–270, New York, NY, USA, 1999. ACM. ISBN 1-58113-142-9. doi: 10.1145/313451.313556. URL http://doi.acm.org/10.1145/313451.313556.

C. Faloutsos, H. Jagadish, A. Mendelzon, and T. Milo. A signature technique for similarity-based queries. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 2–, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-8132-2. URL http://dl.acm.org/citation.cfm?id=829502.830045.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1390681.1442794.

J.-B. P. Faucher, A. M. Everett, and R. Lawson. A complex adaptive organization under the lens of the life model: The case of wikipedia. In *The Fourth Organization Studies Summer Workshop: "Embracing Complexity: Advancing Ecological Understanding in Organization Studies"*, Pissouri, Cyprus, 2008.

A. Floratou, S. Tata, and J. Patel. Efficient and accurate discovery of patterns in sequence data sets. *Knowledge and Data Engineering, IEEE Transactions on*, 23(8):1154–1168, 2011. ISSN 1041-4347. doi: 10.1109/TKDE.2011.69.

A. W.-C. Fu, E. Keogh, L. Y. Lau, C. A. Ratanamahatana, and R. C.-W. Wong. Scaling and time warping in time series querying. *The VLDB Journal*, 17(4):899–921, July 2008. ISSN 1066-8888. doi: 10.1007/s00778-006-0040-z. URL http://dx.doi.org/10.1007/s00778-006-0040-z.

R. Fujimaki, T. Yairi, and K. Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 401–410, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081917. URL http://doi.acm.org/10.1145/1081870.1081917.

C. Gazen, J. Carbonell, and P. Hayes. Novelty detection in data streams: A small step towards anticipating strategic surprise. In *Novel Intelligence from Massive Data (NIMD) PI Meeting*, 2005.

J. Goldstein. Emergence as a construct: History and issues. *Emergence*, 1:49–72, 1999.

M. Gosalia, K. Lin, A. Redfern, S. Romanovsky, N. Shah, D. Steingart, S.-H. Teh, N. Turner, W. Watts, X. Yang, and P. Levis. Smoke: Mote powered fire evacuation, Fall 2004. URL http://cents.cs.berkeley.edu/.

X. Gu and H. Wang. Online anomaly prediction for robust cluster systems. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 1000–1011, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3545-6. doi: 10.1109/ICDE.2009.128. URL http://dx.doi.org/10.1109/ICDE.2009.128.

HADOOP. http://wiki.apache.org/hadoop/.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278. URL http://doi.acm.org/10.1145/1656274.1656278.

J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM. ISBN 1-58113-217-4. doi: 10.1145/342009.335372. URL http://doi.acm.org/10.1145/342009.335372.

X. Hao, L. Duo-lin, and L. Zhi-jie. The building of e-commerce transaction network based on multi-agent and cas theory. In *Wearable Computing Systems (APWCS), 2010 Asia-Pacific Conference on*, pages 295–298, April 2010. doi: 10.1109/APWCS.2010.81.

A. Hassan, R. Jones, and K. L. Klinkner. Beyond dcg: user behavior as a predictor of a successful search. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 221–230, New York, NY, USA, 2010. ACM.

W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 174–185, New York, NY, USA, 1999. ACM. ISBN 1-58113-142-9. doi: 10.1145/313451.313529. URL http://doi.acm.org/10.1145/313451.313529.

W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8 - Volume 8*, HICSS '00, pages 8020–, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0493-0. URL http://dl.acm.org/citation.cfm?id=820264.820485.

H. Herodotou. Hadoop performance models. Technical Report CS-2011-05, Computer Science Department, Duke University, June 2011.

H. Herodotou and S. Babu. Profiling, what-if analysis, and cost-based optimization of mapreduce programs. *PVLDB*, 4(11):1111–1122, 2011.

H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu. Starfish: A self-tuning system for big data analytics. In *CIDR*, pages 261–272, 2011.

HIVE. http://hive.apache.org.

H. Hoffmann. Kernel pca for novelty detection. *Pattern Recogn.*, 40(3):863–874, Mar. 2007. ISSN 0031-3203. doi: 10.1016/j.patcog.2006.07.009. URL http://dx.doi.org/10.1016/j.patcog.2006.07.009.

J. H. Holland. Studying complex adaptive systems. *Journal of Systems Science and Complexity*, 19:1–8, 2006.

HowManyMapsAndReduces. http://wiki.apache.org/hadoop/ HowManyMapsAndReduces.

B. Hu, Y. Zhang, W. Chen, G. Wang, and Q. Yang. Characterizing search intent diversity into click models. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 17–26, New York, NY, USA, 2011. ACM.

M. Hulsmann, H. Kopfer, P. Cordes, and M. Bloos. Collaborative transportation planning in complex adaptive logistics systems: A complexity science-based analysis of decision-making problems of "groupage systems". In J. Zhou, editor, *Complex Sciences*, volume 4 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 1160–1166. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02465-8. doi: 10.1007/978-3-642-02466-5_116. URL http://dx.doi.org/10.1007/978-3-642-02466-5_116.

A. Jakulin and I. Bratko. Testing the significance of attribute interactions. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 52–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015377. URL http://doi.acm.org/10.1145/1015330.1015377.

R. Jhawar and V. Piuri. *Fault Tolerance and Resilience in Cloud Computing Environments*, volume abs/1106.5457. Morgan Kaufmann, 2nd edition, 2013.

R. M. Kanter. On twitter and in the workplace, it's power to the connectors. *Harvard Business Review*, 2009.

Y. Kawahara and M. Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the SIAM International Conference on Data Mining*, SDM, pages 389–400, 2009.

Y. Kawahara and M. Sugiyama. Sequential change-point detection based on direct density-ratio estimation. *Stat. Anal. Data Min.*, 5(2):114–127, Apr. 2012. ISSN 1932-1864. doi: 10.1002/sam.10124. URL http://dx.doi.org/10.1002/sam.10124.

E. Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 406–417. VLDB Endowment, 2002. URL http://dl.acm.org/citation.cfm?id=1287369.1287405.

D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 180–191. VLDB Endowment, 2004. ISBN 0-12-088469-0. URL http://dl.acm.org/citation.cfm?id=1316689.1316707.

R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 959–967, New York, NY, USA, 2007. ACM.

R. Kohavi, T. Crook, R. Longbotham, B. Frasca, R. Henne, J. L. Ferres, and T. Melamed. Online experimentation at microsoft, 2009a. URL http://www.exp-platform.com/Pages/expMicrosoft.aspx.

R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data Min. Knowl. Discov.*, 18(1):140–181, feb 2009b.

A. Kotov, P. N. Bennett, R. W. White, S. T. Dumais, and J. Teevan. Modeling and analysis of cross-session search tasks. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 5–14, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0757-4. doi: 10.1145/2009916.2009922. URL http://doi.acm.org/10.1145/2009916.2009922.

B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, pages 234–247, New York, NY, USA, 2003. ACM. ISBN 1-58113-773-7. doi: 10.1145/948205.948236. URL http://doi.acm.org/10.1145/948205.948236.

R. Kurzweil. *How to Create a Mind: The Secret of Human Thought Revealed*. Viking Adult, 2012.

S. Lamparter, S. Becher, and J.-G. Fischer. An agent-based market platform for smart grids. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Industry Track*, AAMAS '10, pages 1689–1696, Richland, SC, 2010. International

Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9826571-4-0. URL http://dl.acm.org/citation.cfm?id=1838194.1838197.

T. Laroum and B. Tighiouart. A multi-agent system for the modelling of the hiv infection. In J. O'Shea, N. Nguyen, K. Crockett, R. Howlett, and L. Jain, editors, *Agent and Multi-Agent Systems: Technologies and Applications*, volume 6682 of *Lecture Notes in Computer Science*, pages 94–102. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-21999-3. doi: 10.1007/978-3-642-22000-5_11. URL http://dx.doi.org/10.1007/978-3-642-22000-5_11.

L. Li, B. A. Prakash, and C. Faloutsos. Parsimonious linear fingerprinting for time series. *Proc. VLDB Endow.*, 3(1-2):385–396, Sept. 2010. ISSN 2150-8097. URL http://dl.acm.org/citation.cfm?id=1920841.1920893.

Y.-s. Lim, S. Lim, J. Choi, S. Cho, C.-k. Kim, Y.-W. L. H. Hu, H. Zhang, H. Hu, B. Xu, J. Li, and A. Ma. A fire detection and rescue support framework with wireless sensor networks. In *Proceedings of the 2007 International Conference on Convergence Information Technology*, ICCIT '07, pages 135–139, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3038-9. doi: 10.1109/ICCIT.2007.28. URL http://dx.doi.org/10.1109/ICCIT.2007.28.

J. Lin and A. Kolcz. Large-scale machine learning at twitter. In *SIGMOD '12 Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 793–804, New York, NY, USA, 2012. ACM.

T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: actions for entity-centric search. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 589–598, 2012.

Q. Lu, B. George, and S. Shekhar. Capacity constrained routing algorithms for evacuation planning: a summary of results. In *Proceedings of the 9th international conference on Advances in Spatial and Temporal Databases*, SSTD'05, pages 291–307, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-28127-4, 978-3-540-28127-6. doi: 10.1007/11535331_17. URL http://dx.doi.org/10.1007/11535331_17.

C. M. Macal, C. M. Macal, C. M. Macal, M. J. North, and M. J. North. Validation of an agent-based model of deregulated electric power markets. In *Proc. North American Computational Social and Organization Science (NAACSOS) 2005 Conference, South*, 2005.

B. MacLennan. Evolutionary psychology, complex systems, and social theory. *Soundings: An Interdisciplinary Journal*, 90(3/4):169–189, 2007.

S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, Dec. 2002. ISSN 0163-5980. doi: 10.1145/844128.844142. URL http://doi.acm.org/10.1145/844128.844142.

B. Mandelbrot. How long is the coast of britain? statistical self-similarity and fractional dimension. *Science*, 156(3775):636–638, 1965.

M. Markou and S. Singh. Novelty detection: a review - part 1: statistical approaches. *Signal Process.*, 83(12):2481–2497, Dec. 2003. ISSN 0165-1684. doi: 10.1016/j.sigpro.2003.07.018. URL http://dx.doi.org/10.1016/j.sigpro.2003.07.018.

H. Mehlum, K. Moene, and R. Torvik. Predator or prey?: Parasitic enterprises in economic development. *European Economic Review*, 47(2):275–294, April 2003. URL http://ideas.repec.org/a/eee/eecrev/v47y2003i2p275-294.html.

MICROSTRATEGY. http://www.microstrategy.com.

J. H. Miller and S. E. Page. *Complex Adaptive Systems: An Introduction to Computational Models of Social Life*. Princeton Studies in Complexity. Princeton University Press, 2007.

M. J. Miranda and P. L. Fackler. *Applied Computational Economics and Finance*. Number 0262633094. The MIT Press, 2002.

A. Moore, G. Cooper, R. Tsui, and M. Wagner. Summary of biosurveillance-relevant statistical and data mining technologies. February 2002.

S. Muthukrishnan, E. van den Berg, and Y. Wu. Sequential change detection on data streams. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 551–550, 2007. doi: 10.1109/ICDMW.2007.89.

S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 250–262, New York, NY, USA, 2004. ACM. ISBN 1-58113-879-2. doi: 10.1145/1031495.1031525. URL http://doi.acm.org/10.1145/1031495.1031525.

D. B. Neil and W.-K. Wong. Tutorial on event detection. In *KDD*, 2009.

D. B. Neill. Expectation-based scan statistics for monitoring spatial time series data. *International Journal of Forecasting*, 25(3):498–517, 2009. URL http://EconPapers.repec.org/RePEc:eee:intfor:v:25:y:2009:i:3:p:498-517.

Netlogo. http://ccl.northwestern.edu/netlogo/.

C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 631–636, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956831. URL http://doi.acm.org/10.1145/956750.956831.

A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 3rd edition, August 2009.

M. Pan, C. Tsai, and Y. Tseng. Emergency guiding and monitoring applications in indoor 3d environments by wireless sensor networks. *Int. J. Sen. Netw.*, 1(1/2):2–10, Sept. 2006. ISSN 1748-1279. doi: 10.1504/IJSNET.2006.010829. URL http://dx.doi.org/10.1504/IJSNET.2006.010829.

N. Parikh. Mining large-scale temporal dynamics with hadoop. In *Hadoop Summit*, San Jose, CA, Jun 20 2012.

C.-S. Perng, H. Wang, S. Zhang, and D. Parker. Landmarks: a new model for similarity-based pattern querying in time series databases. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pages 33–42, 2000. doi: 10.1109/ICDE.2000.839385.

R. Pfeffer. *Teradata RDBMS.* NCR, Teradata Division.

Pig. http://pig.apache.org.

J. M. Ponte and W. B. Croft. Text segmentation by topic. In *In Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 120–129, 1997.

B. A. Prakash, N. Valler, D. Andersen, M. Faloutsos, and C. Faloutsos. Bgp-lens: patterns and anomalies in internet routing updates. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1315–1324, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557160. URL http://doi.acm.org/10.1145/1557019.1557160.

B. A. Prakash, H. Tong, N. Valler, M. Faloutsos, and C. Faloutsos. Virus propagation on time-varying networks: theory and immunization algorithms. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 99–114, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15938-9, 978-3-642-15938-1. URL http://dl.acm.org/citation.cfm?id=1889788.1889796.

B. T. Rao and L. S. S. Reddy. Survey on improved scheduling in hadoop mapreduce in cloud environments. *CoRR*, abs/1207.0780, 2012.

D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.

M. Sabah. Hadoop and cloud and netflix: Taming the social data. In *Hadoop Summit*, San Jose, CA, June 13-14 2012.

E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 841–850, New York, NY, USA, 2010. ACM.

L. Saitta, A. Giordana, and A. Cornuejols. *Phase Transitions in Machine Learning.* Cambridge University Press, 1st edition, 2011.

SAS. http://www.sas.com.

D. Savenkov, D. Lagun, and Q. Liu. Search engine switching detection based on user personal preferences and behavior patterns. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 33–42, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2034-4. doi: 10.1145/2484028.2484099. URL http://doi.acm.org/10.1145/2484028.2484099.

Scala. http://www.scala-lang.org.

B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

C. Schurgers, V. Tsiatsis, and M. Srivastava. Stem: Topology management for energy efficient sensor networks. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 3, pages 3–1099–3–1108 vol.3, 2002. doi: 10.1109/AERO.2002.1035239.

M. Severo and J. Gama. Ubiquitous knowledge discovery. In M. May and L. Saitta, editors, *Change detection with Kalman filter and CUSUM*, chapter Change detection with Kalman filter and CUSUM, pages 148–162. Springer-Verlag, Berlin, Heidelberg, 2010. ISBN 3-642-16391-2, 978-3-642-16391-3. URL http://dl.acm.org/citation.cfm?id=1986531.1986542.

M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Tina: a scheme for temporal coherency-aware in-network aggregation. In *Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, MobiDe '03, pages 69–76, New York, NY, USA, 2003. ACM. ISBN 1-58113-767-2. doi: 10.1145/940923.940937. URL http://doi.acm.org/10.1145/940923.940937.

S. Shen, B. Hu, W. Chen, and Q. Yang. Personalized click model through collaborative filtering. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 323–332, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5. doi: 10.1145/2124295.2124336. URL http://doi.acm.org/10.1145/2124295.2124336.

Y. Shen, J. Yan, S. Yan, L. Ji, N. Liu, and Z. Chen. Sparse hidden-dynamics conditional random fields for user intent understanding. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 7–16, New York, NY, USA, 2011. ACM.

K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

E. J. Spinosa, A. P. de Leon F. de Carvalho, and J. a. Gama. Novelty detection with application to data streams. *Intell. Data Anal.*, 13(3):405–422, Aug. 2009. ISSN 1088-467X. URL http://dl.acm.org/citation.cfm?id=1551768.1551770.

SPSS. http://www.ibm.com/software/analytics/spss.

R. Steinert and D. Gillblad. Long-term adaptation and distributed detection of local network changes. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5, 2010. doi: 10.1109/GLOCOM.2010.5684137.

M. Stonebraker. The case for shared nothing. *IEEE Database Eng. Bull.*, 9(1):4–9, 1986.

T. Sueyoshi and G. Tadiparthi. Why did the california electricity crisis occur?: A numerical analysis using a multiagent intelligent simulator. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(6):779–790, Nov 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2008.2001691.

Tableau. http://www.tableausoftware.com.

J. Takeuchi and K. Yamanishi. A unifying framework for detecting outliers and change points from time series. *Knowledge and Data Engineering, IEEE Transactions on*, 18(4): 482–492, 2006. ISSN 1041-4347. doi: 10.1109/TKDE.2006.1599387.

N. N. Taleb. *The Black Swan: Second Edition: The Impact of the Highly Improbable*. Random House Trade Paperbacks, 2010.

J. Tan, X. Meng, and L. Zhang. Delay tails in mapreduce scheduling. In *SIGMETRICS '12 Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, pages 5–16.

J. Tan, X. Meng, and L. Zhang. Delay tails in mapreduce scheduling delay tails in mapreduce scheduling. In *SIGMETRICS '12 Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, pages 5–16. ACM, 2012.

D. Tang, A. Agarwal, D. O'Brien, and M. Meyer. Overlapping experiment infrastructure: more, better, faster experimentation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 17–26, New York, NY, USA, 2010. ACM.

Teradata. http://www.teradata.com.

Teradata. *Introduction to Teradata® RDBMS*. B035-1091-122A. NCR Corporation, Dec 2002.

TeradataPricing. http://www.teradata.com/brochures/teradata-purpose-built-platform-pricing-eb5496/. URL `http://www.teradata.com/brochures/Teradata-Purpose-Built-Platform-Pricing-eb5496/`.

L. Tesfatsion. Agent-based computational economics: Growing economies from the bottom up. *Artif. Life*, 8(1):55–82, Mar. 2002. ISSN 1064-5462. doi: 10.1162/106454602753694765. URL `http://dx.doi.org/10.1162/106454602753694765`.

A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Anthony, H. Liu, and R. Murthy. Hive - a petabyte scale data warehouse using hadoop. In *ICDE*, pages 996–1005, 2010.

F. Tian and K. Chen. Towards optimal resource provisioning for running mapreduce programs in public clouds. In *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, CLOUD '11, pages 155–162, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4460-1. doi: 10.1109/CLOUD.2011.14. URL http://dx.doi.org/10.1109/CLOUD.2011.14.

N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. Wavescheduling: energy-efficient data dissemination for sensor networks. In *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, DMSN '04, pages 48–57, New York, NY, USA, 2004. ACM. doi: 10.1145/1052199.1052209. URL http://doi.acm.org/10.1145/1052199.1052209.

Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai. Wireless sensor networks for emergency navigation. *Computer*, 39(7):55–62, 2006. ISSN 0018-9162. doi: 10.1109/MC.2006.248.

T. Tsunemine, E. Kadokawa, Y. Ueda, J. Fukumoto, T. Wada, K. Ohtsuki, and H. Okada. Emergency urgent communications for searching evacuation route in a local disaster. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pages 1196–1200, 2008. doi: 10.1109/ccnc08.2007.267.

T. Varadharajan and C. Rajendran. A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. *European Journal of Operational Research*, 167(3):772–795, December 2005. URL http://ideas.repec.org/a/eee/ejores/v167y2005i3p772-795.html.

A. Verma, L. Cherkasova, and R. H. Campbell. Aria: automatic resource inference and allocation for mapreduce environments. In *ICAC '11 Proceedings of the 8th ACM international conference on Autonomic computing*, pages 235–244, New York, NY, USA, 2011a. ACM.

A. Verma, L. Cherkasova, and R. H. Campbell. Slo-driven right-sizing and resource provisioning of mapreduce jobs. In *Workshop on Large Scale Distributed Systems and Middleware (LADIS) in conjunction with VLDB*, Seattle, Washington, 09/2011 2011b.

A. Verma, L. Cherkasova, and R. Campbell. Two sides of a coin: Optimizing the schedule of mapreduce jobs to minimize their makespan and improve cluster performance. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*, pages 11–18, 2012.

Vertica. http://www.vertica.com.

W3C. http://www.w3.org/tr/dom-level-3-core/.

G. Wang, A. Butt, P. Pandey, and K. Gupta. A simulation approach to evaluating design decisions in mapreduce setups. In *MASCOTS*, pages 1–11, 2009.

H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 226–235, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956778. URL http://doi.acm.org/10.1145/956750.956778.

P. Wang, H. Wang, and W. Wang. Finding semantics in time series. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11, pages 385–396, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0661-4. doi: 10.1145/1989323.1989364. URL http://doi.acm.org/10.1145/1989323.1989364.

Y. Wang, X. Huang, and R. W. White. Characterizing and supporting cross-device search tasks. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 707–716, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1869-3. doi: 10.1145/2433396.2433484. URL http://doi.acm.org/10.1145/2433396.2433484.

L. A. Wehinger. *Agent-based modeling in electricity markets: Introducing a new predictive agent learning approach*. PhD thesis, Eidgeoessische Technische Hochschule Zurich and Carnegie Mellon university, 2010.

E. Weisstein. Fold bifurcation. from mathworld - a wolfram web resource. http://mathworld.wolfram.com/foldbifurcation.html, a.

E. Weisstein. Gibbs phenomenon. from mathworld - a wolfram web resource. http://http://mathworld.wolfram.com/gibbsphenomenon.html., b.

E. Weisstein. Logistic equation. from mathworld - a wolfram web resource. http://mathworld.wolfram.com/logisticequation.html., c.

E. Weisstein. Parseval's theorem. from mathworld - a wolfram web resource. http://mathworld.wolfram.com/parsevalstheorem.html., d.

R. W. White and S. T. Dumais. Characterizing and predicting search engine switching behavior. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 87–96, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3. doi: 10.1145/1645953.1645967. URL http://doi.acm.org/10.1145/1645953.1645967.

T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, 2nd edition, Sep 2010. ISBN 0596521979.

U. Wilensky and M. Resnick. Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology*, 8(1):N/A, 1999.

J. Wolf, D. Rajan, K. Hildrum, R. Khandekar, V. Kumar, S. Parekh, K.-L. Wu, and A. balmin. Flex: A slot allocation scheduling optimizer for mapreduce workloads. In I. Gupta and C. Mascolo, editors, *Middleware 2010*, LNCS 6452, pages 1–20, 2010.

E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 407–418, New York, NY, USA, 2006. ACM. ISBN 1-59593-434-0. doi: 10.1145/1142473.1142520. URL http://doi.acm.org/10.1145/1142473.1142520.

F. Wu, C. Yeung, A. Poon, and J. Yen. A multi-agent approach to the deregulation and restructuring of power industry. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 3, pages 122–131 vol.3, 1998. doi: 10.1109/HICSS.1998.656079.

B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li. Context-aware ranking in web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 451–458, New York, NY, USA, 2010. ACM.

Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 70–84, New York, NY, USA, 2001. ACM. ISBN 1-58113-422-3. doi: 10.1145/381677.381685. URL http://doi.acm.org/10.1145/381677.381685.

K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Min. Knowl. Discov.*, 8(3):275–300, May 2004. ISSN 1384-5810. doi: 10.1023/B:DAMI.0000023676.72185.7c. URL http://dx.doi.org/10.1023/B:DAMI.0000023676.72185.7c.

X. Yang and J. Sun. An analytical performance model of mapreduce. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 306–310, 2011.

Y. Yang, X. Wu, and X. Zhu. Combining proactive and reactive predictions for data streams. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 710–715, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081961. URL http://doi.acm.org/10.1145/1081870.1081961.

Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, Sept. 2002. ISSN 0163-5808. doi: 10.1145/601858.601861. URL http://doi.acm.org/10.1145/601858.601861.

W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1567–1576 vol.3, 2002. doi: 10.1109/INFCOM.2002.1019408.

M. Yong, N. Garegrat, and M. Shiwali. Towards a resource aware scheduler in hadoop. In *ICWS*, 2009.

M. Younis, M. Youssef, and K. Arisha. Energy-aware routing in cluster-based sensor networks. In *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, MASCOTS '02, pages 129–, Washington, DC, USA, 2002. IEEE Computer Society. URL http://dl.acm.org/citation.cfm?id=882460.882620.

V. I. Zadorozhny, P. K. Chrysanthis, and P. Krishnamurthy. A framework for extending the synergy between mac layer and query optimization in sensor networks. In *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, DMSN '04, pages 68–77, New York, NY, USA, 2004. ACM. doi: 10.1145/1052199.1052211. URL http://doi.acm.org/10.1145/1052199.1052211.

M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Job scheduling for multi-user mapreduce clusters. Technical Report UCBEECS200955, EECS Department University of California Berkeley, 2009.

M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *EuroSys '10 Proceedings of the 5th European conference on Computer systems*, pages 265–278, New York, NY, USA, 2010. ACM.

Z. Zhang, L. Cherkasova, A. Verma, and B. T. Loo. Optimizing completion time and resource provisioning of pig programs. In *CCGRID '12 Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 811–816, Washington, DC, USA, 2012a. IEEE Computer Society.

Z. Zhang, L. Cherkasova, A. Verma, and B. T. Loo. Automated profiling and resource management of pig programs for meeting service level objectives. In *Proceedings of the 9th international conference on Autonomic computing*, pages 53–62, New York, NY, USA, Sept. 14-18 2012b. ACM.

R. Zheng and R. Kravets. On-demand power management for ad hoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 481–491 vol.1, 2003. doi: 10.1109/INFCOM.2003.1208699.

R. Zheng, J. C. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '03, pages 35–45, New York, NY, USA, 2003. ACM. ISBN 1-58113-684-6. doi: 10.1145/778415.778420. URL http://doi.acm.org/10.1145/778415.778420.