



University of Zurich  
Department of Informatics

# Anti Money Laundering under real world conditions

-

## Finding relevant patterns



Master Thesis September 9, 2009

**Linard Moll**  
from Schwyz SZ, Switzerland

Student-ID: 00-916-932  
aml@linardmoll.ch

Advisor: **Jonas Luell**

Prof. Abraham Bernstein, PhD  
Department of Informatics  
University of Zurich  
<http://www.ifi.uzh.ch/ddis>



---

# Acknowledgements

I would like to thank Professor Abraham Bernstein for offering me the opportunity to write my Master Thesis at the DDIS group of the University of Zurich. Many thanks to Jonas Luell for the interesting topic and all the hours of discussion about new algorithms and improvements of the system, since he was the only one allowed to verify the impact of the system on confidential real world data. Additionally, I would like to thank my parents for all the support that enabled this thesis after several years of studying, and my friends for joining the coffee breaks and all the technical and non-technical discussions during the last months.



---

# Abstract

This Master Thesis deals with the search for new patterns to enhance the discovery of fraudulent activities within the jurisdiction of a financial institution. Therefore transactional data from a database is analyzed, scored and processed for the later usage by an internal anti-money laundering specialist. The findings are again stored in a database and processed by *TV*- the Transaction Visualizer, an existing and already commercially used tool. As a result of this thesis, the software module *TMatch* and the graphical user interface *TMatchViz* were developed. The interaction of these two tools was tested and evaluated using synthetically created datasets. Furthermore, the approximations made and their impact on the specification of the algorithms will be addressed in his report.



---

# Zusammenfassung

Diese Masterarbeit befasst sich mit der Suche nach neuen Algorithmen, um betrügerische Aktivitäten innerhalb des Einflussbereiches einer Bank besser zu erkennen. Dazu werden Transaktionsdaten ausgewertet und für die weitere Untersuchung durch einen bankinternen Geldwäscherei-Spezialisten bewertet und aufbereitet. Die gefundenen Resultate werden in einer Datenbank hinterlegt und durch das bestehende und bereits kommerziell verwendete Werkzeug *Transaction Visualizer*, weiterverarbeitet. Zu diesem Zweck wurde die Softwarekomponente *TMatch* entwickelt, welche die erstellten Algorithmen beinhaltet und die Bedienoberfläche *TMatchViz*, um die Konfiguration und die Bewertung der Resultate vereinfacht. Das Zusammenspiel von *TMatch* und *TMatchViz* wurde anhand von synthetischen Daten geprüft und beurteilt. Im Weiteren wird auf gemachte Annahmen und deren Auswirkungen auf die Spezifikation der Algorithmen näher eingegangen.





---

# Table of Contents

<b>Table of Contents</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Goal of this thesis . . . . .	2
1.2 Structure . . . . .	2
<b>2 Motivation</b>	<b>3</b>
2.1 Fraud, fraudulent behavior and money laundering . . . . .	3
2.1.1 Fraud and fraudulent behavior . . . . .	4
2.1.2 Money laundering . . . . .	4
2.1.3 Criticism . . . . .	6
2.1.4 Conclusion . . . . .	7
2.2 Money Laundering detection and related work . . . . .	7
2.2.1 Introduction . . . . .	8
2.2.2 Basic money laundering detection pattern . . . . .	9
2.2.3 Related work . . . . .	10
2.2.4 Available anti-money laundering solutions . . . . .	15
2.3 Suggested solution . . . . .	16
2.3.1 Requirements of the solution . . . . .	16
<b>3 Approach Description</b>	<b>19</b>
3.1 Overview . . . . .	19
3.2 Used Software . . . . .	21
3.3 Assumptions . . . . .	21
3.4 Implementation of TMatch . . . . .	22
3.4.1 Architectural view . . . . .	22
3.4.2 Configuration . . . . .	22
3.4.3 Runner . . . . .	23
3.4.4 Expander . . . . .	23
3.4.5 Component Calculation . . . . .	24

3.4.6 Scorer . . . . .	29
3.5 Implementation of TMatchViz . . . . .	31
3.5.1 Architectural View . . . . .	31
3.5.2 Configurability . . . . .	32
3.5.3 Preselection of the results . . . . .	32
3.5.4 Distribution . . . . .	34
<b>4 Evaluation</b>	<b>37</b>
4.1 Test environment . . . . .	37
4.2 Test configuration . . . . .	37
4.3 Test results . . . . .	38
4.4 Conclusion . . . . .	40
<b>5 Discussion</b>	<b>43</b>
5.1 Limitations . . . . .	43
<b>6 Conclusion and future work</b>	<b>45</b>
6.1 Future Work . . . . .	45
<b>A Appendix</b>	<b>47</b>
A.1 Content CD-ROM . . . . .	47
A.2 TMatchViz graphical user interface . . . . .	48
A.3 TMatch extendability . . . . .	48
A.4 TMatch library creation . . . . .	52
A.5 TMatchViz release creation . . . . .	52
A.6 TMatchViz installation . . . . .	54
A.7 Properties . . . . .	55
A.8 Logging . . . . .	55
A.9 Evaluation results . . . . .	56
A.10 Database related . . . . .	61
<b>List of Figures</b>	<b>63</b>
<b>List of Tables</b>	<b>65</b>
<b>List of Listings</b>	<b>67</b>
<b>Bibliography</b>	<b>69</b>

# 1

## Introduction

Hardly a week goes by without a newspaper article on fraudulent financial actions or even money laundering occurrence. Fraudulent money transactions are often reported in the case of corruption and blackmail. Since the incident of 9/11, money-laundering stands close to the term "terror financing". According to the 9/11 commission [Roth et al., 2004], the airplane hijackers used U.S. and foreign financial institutions to manage funds. The U.S. accounts were filled up by wire transfers and deposits of cash. The foreign accounts were accessed in the U.S. through ATM and credit card transactions. The commission remarks, that neither the airplane hijackers nor the financial facilitators were experts in use of the financial system and created a lasting imprint with their financial transactions. But because the existing money-laundering controls were focussing on drug trafficking and large-scale financial fraud, the hijackers' transactions could not have been detected by the institutions. As a first reaction, the U.S. and later on the E.U. created new laws to focus on the prevention of fraudulent actions on the financial markets [S/CT, 2001].

The Swiss government created an Anti-Money Laundering Act (AMLA) in October 1997 [MROS, 2009]. This forms the official basis of the Money Laundering Reporting Office Switzerland (MROS), a department of the Swiss Office of Police<sup>1</sup>. The MROS receives reports on suspicious transactions so called Suspicious Activity Reports (SARs) from all types of financial intermediaries, such as banks, payment service providers, fiduciaries, asset and investment advisors, attorneys, insurances, casinos, credit card companies, etc. According to [MROS, 2009], the MROS received 851 SARs in 2008 involving assets of around CHF 1.9 billion - compared to 303 SARs in 1999 [MROS, 1999] involving assets of more than CHF 1.5 billion. According to MROS, 69% (572) of all SARs were reported from the banking sector in 2008.

The U.S. Bureau of International Narcotics and Law Enforcement Affairs (INL) published a list [INL, 2009] of countries containing Switzerland as a "Jurisdiction of Primary Concern" saying that the swiss financial institutions engage in transactions involving significant amounts of (laundered) proceeds from all serious crimes like narcotics trafficking and narcotics-related money

---

<sup>1</sup>From time to time, the intergovernmental organization Financial Action Task Force (FATF) assesses the Swiss financial center and the measures for combating money laundering and terrorism financing. In 2005, the FATF certifies the MROS as "largely compliant" with FATF standards [MROS, 2005], in the following years new FATF suggestions were adopted or are in process to.

laundering. In other words, the Swiss finance markets are affected by heavy money laundering attempts.

## 1.1 Goal of this thesis

The main task of this thesis is to find new patterns which support already existing anti-money laundering (AML) efforts. For this purpose we collaborate with a Swiss financial institution. Because of strict confidential agreements, the anonymous virtual financial institution called *AlphaFin* [Luell and Bernstein, 2009] was created. With this background, the main goal was to enhance the discovery of fraudulent activities within the jurisdiction of *AlphaFin*. Therefore, data from *AlphaFin*'s transaction database was analyzed, scored and stored for the later usage by an internal anti-money laundering specialist. This led us to the development of the software module *TMatch* and the graphical user interface *TMatchViz* to fulfill the needs at *AlphaFin*. *TMatch* implements the newly created patterns for transaction processing and scoring. In addition it provides all the needed configuration and database management parts. *TMatchViz* uses *TMatch* as an external library and adds the necessary user-interaction functionality to our software solution. The co-operation and efficiency of these two modules was tested and evaluated by using synthetically created transaction datasets.

## 1.2 Structure

For the upcoming part of this thesis, we follow the structure of a "design science thesis" according to [Bernstein, 2005], including the following chapters.

Chapter 2 mentions the motivation behind this thesis together with recent work in the field of anti-money laundering, the problem definition and a description of a possible solution.

In chapter 3, the implementation of *TMatch* and *TMatchViz*, the made assumptions and the specification are described.

Chapter 4 addresses the evaluation of the *TMatch* and *TMatchViz* software modules by regarding issues like processing power, memory consumption and time durations.

In chapter 5, the results are matched to the initial goals and the limitations of *TMatch* and *TMatchViz* are discussed.

Finally, chapter 6 draws the conclusion of this Master Thesis; suggestions for future work and a personal résumé are given.

# 2

## Motivation

In the end of March 2009, the Office of the Attorney General of Switzerland (OAG) freezes 700 million Swiss francs [Häfliger, 2009]. The OAG suspects managers of the Czech Coal mining company committing crimes of embezzlements. The money was spread across 100 Swiss bank accounts.

This example shows, how an act of crime seems to work today. Money is transferred from one country to another in unforeseeable amounts and deposited into separate places. According to this newspaper article the average account holdings is around 7 million Swiss francs. Whether the transactions were made in full or split up into smaller parts was not stated, but the accused actions should have taken place between 1997 and 2002. The money transfers went through several letterbox companies all over the world to get a shiny face.

This points to the main problems anti-money laundering precaution measures must overcome if fraud is conducted in a continuous manner, over a long time period and from different locations worldwide. At the same time, such money laundering attempts reflects the motivation behind this Master Thesis to enhance detection at *AlphaFin*.

Effective AML measures consists of legal, organizational and technical aspects. In order to reach the goal of this thesis, it is considerably important to know each aspect. The upcoming sections will explain the terms and legal aspects of fraud and money laundering, as needed for this thesis, technical countermeasures found in literature and as commercially available products. This will lead us to the main part of this Master Thesis; the description of the *TMatch* and *TMatchViz* systems.

### 2.1 Fraud, fraudulent behavior and money laundering

As [Wells, 2008] accurately states: "Understanding how fraud is committed is paramount to preventing and detecting it.". This presupposes a clean and relevant understanding of *Fraud* in consideration of this thesis.

### 2.1.1 Fraud and fraudulent behavior

According to [Encyclopædia Britannica, 2009], fraud stands in its broadest sense for “the deliberate misrepresentation of fact for the purpose of depriving someone of a valuable possession” and has its origin in occupational frauds and abuses, various acts of corruption, asset misappropriation and fraudulent statements [Wells, 2008]. Involved actors are in all kind of positions. Financial statement fraud, for instance, is potentially committed by the senior management, mid- and lower-level employers and organized criminals, according to [Wells, 2008].

Relevant for this thesis is the aspect of economic related fraud, most likely based on specific money transactions. Those are transfers from and to financial institutions and in the case of a bank, the deposition or withdraw of money from a certain bank account. Hence, fraudulent behavior describes the circumstances that causes the fraud, and again in this context, resulting in individual money transactions. In turn, this characteristic of fraud denotes the term of *Money laundering*.

### 2.1.2 Money laundering

Money laundering is a subset of fraud and has a legal and a technical interpretation. In this subsection the legal parts will be highlighted. The technical and more important part for this thesis is covered in the Section 2.2.

According to [Altenkirch, 2006], there are several theoretical models of money laundering. The most popular one is the three-step model [Altenkirch, 2006] created by the U.S. customs authorities, including the following steps:

1. *Placement*. Cash reserves obtained from underlying offenses must be transformed to legitimate money. Ways to do so are to place the money on deposit, pay vastly inflated amounts for goods or the purchase of real estate for later renting or leasing.
2. *Layering*. After successfully placed money into the financial system, money launderers tries to cover the paper trail by carrying out multiple transactions using multiple accounts, different owners and different financial institutions in different countries.
3. *Integration*. The money launderer aim is to report the money as regular income and regain access. Interests from rent or deposits are now used for new undertakings having a clean record from legitimate activities.

In the legal industries *Money laundering* describes the legal controls that require regulated entities like financial institutions to prevent or report money laundering actions. In most countries, financial and non-financial institutions are required to identify and report suspicious money transactions to the financial intelligence unit of that particular country.

In the U.S. the legal base is set by the Bank Secrecy Act of 1970<sup>1</sup> and the USA PATRIOT Act<sup>2</sup>.

<sup>1</sup>See [http://www.fincen.gov/statutes\\_regs/bsa](http://www.fincen.gov/statutes_regs/bsa).

<sup>2</sup>See [http://www.fincen.gov/statutes\\_regs/patriot/index.html](http://www.fincen.gov/statutes_regs/patriot/index.html).

They define what transactions must be reported to the government. As an example, cash transactions in excess of USD 10,000<sup>3</sup> during the same business day conducted by or on behalf of the same person, must be reported.

In Switzerland, money laundering was recognized as an offence in the Swiss Criminal Code since 1990. The Swiss Anti-Money Laundering Act of 1998<sup>4</sup> (SAMLA) enacted due diligence obligations to all financial intermediaries like banks, insurance companies, independent asset managers, money transmitter or money changer. The financial intermediaries have to inform the government, if there is a probable cause. This so-called *Know Your Customer* (KYC) principle stands for the obligation of all the intermediaries to verify the identity of the customer; establishing the identity of the beneficial owner; the repetition of these two actions, if some doubt arises during daily business; the duty to clarify the nature and purpose of a business relationship or of an unusual transaction; the duty to keep records of carried out transactions<sup>5</sup>. The SAMLA further regulates the responsibility of the financial intermediaries to instruct and test their employees adequately; the duty to immediately file a report<sup>6</sup> with the reporting office<sup>7</sup> and the obligation of freezing assets. Since 2009, the Federal Act on the Swiss Financial Market Supervisory Authority<sup>8</sup> entered full force with the consequence that the Swiss Financial Market Supervisory Authority (FINMA)<sup>9</sup> is in charge of the governmental supervision in Switzerland.

The Financial Action Task Force (FATF)<sup>10</sup> is an inter-governmental organization. The FATF developed policies to combat money laundering and terrorist financing and was founded in 1989 by the G7 forum<sup>11</sup>. The FATF currently comprises 32 member jurisdictions and 2 regional organizations, representing most major financial centers in all parts of the globe [FATF, 2009].

The Wolfsberg Group<sup>12</sup> is an association of eleven global banks, developing and publishing financial services industry standards and policies, focused on the KYC principle, anti-money laundering and counter terrorist financing. Members of this group are Barclays, Citigroup, Credit Suisse and HSBC to name just some of them. Some of the Wolfsberg Standards<sup>13</sup> are the "Wolfsberg AML Guidance on Credit/Charge Card Issuing and Merchant Acquiring Activities" (2009), the "Wolfsberg AML Principles on Private Banking" (2002) and the "Wolfsberg AML Principles for Correspondent Banking" (2002). The latter principle includes a more risk-based view on due diligence<sup>14</sup> and postulates an "Enhanced Due Diligence" approach containing a richer information

<sup>3</sup> According to the Bank Secrecy Act all banks had to report transactions with an total amount over USD 10,000 to the authorities.

<sup>4</sup> See [http://www.admin.ch/ch/d/sr/c955\\_0.html](http://www.admin.ch/ch/d/sr/c955_0.html).

<sup>5</sup> In Switzerland for a minimum of ten years.

<sup>6</sup> This means the Suspicious Activity Report (SAR).

<sup>7</sup> This stands for the Money Laundering Reporting Office Switzerland (MROS).

<sup>8</sup> See [http://www.admin.ch/ch/d/sr/c956\\_1.html](http://www.admin.ch/ch/d/sr/c956_1.html).

<sup>9</sup> See <http://www.finma.ch>.

<sup>10</sup> The FATF is housed at the headquarters of the OECD in Paris; also known by its French name Groupe d'action financière sur le blanchiment de capitaux (GAFI). See <http://www.fatf-gafi.org>.

<sup>11</sup> Now the G8 forum - the Group of Eight, consisting of the group's member countries Canada, the Russian Federation, France, Germany, Japan, Italy, the United Kingdom, and the United States, together with the European Union. See <http://www.g8italia2009.it>.

<sup>12</sup> See <http://www.wolfsberg-principles.com>.

<sup>13</sup> Available from <http://www.wolfsberg-principles.com/standards.html>.

<sup>14</sup> Correspondent Banking Clients presenting greater risk should be subjected to a higher level of due diligence. Accord-

exchange transfer in bank-to-bank transactions and an examination of the anti-money laundering measures taken by the counterparty.

There are several compliance frameworks and methodologies helping to detect and avoid money laundering like Basel II<sup>15</sup>, Fair and Accurate Credit Transactions Act (FDICIA)<sup>16</sup>, Gramm-Leach-Bliley Act (GLBA)<sup>17</sup> and Sarbanes-Oxley Act (SOX)<sup>18</sup> to name some of them. Depending on the grad of internationality, companies worldwide have to comply with some of them by implementing the key concepts, providing all the required financial statements to stakeholder and governmental agencies and being successfully audited on a regular basis.

### 2.1.3 Criticism

At same time the money-laundering regulations are getting more comprehensive and versatile, they also meet with criticism. Geiger and Wuensch [Geiger and Wuensch, 2007] published a study conducted in Switzerland, Singapore and Germany on the impact of AML measures on banks and the financial services industry. According to their survey, they found that banks consider the compliance with AML rules essential and important but also highly burdensome, causing significant costs and efforts. In addition they observed a "broad consensus amongst practitioners and scientists that the impact of money laundering prevention on the predicate offences is small" [Geiger and Wuensch, 2007]. Possible reasons are found in the anticipation of the AML restrictions and use of still existing alternatives by the criminals. The fraudster can keep the money in an anonymous form like cash and coins and use informal value transfer systems like the *Harwala* or *Hundi* system<sup>19</sup> without paying taxes, avoiding mandatory reporting on large transactions to the governmental agencies nor leaving a treasonable paper trail. Furthermore, the criminals may use non-designated institutions and private individuals to repurify the criminal money or contrariwise they could avoid the need to launder money by keeping the criminal money in the criminal world<sup>20</sup>. Besides the economic damage resulting by increased transaction costs and discrimination of actors and countries with lower reputation, consequences are carried out by the society through loss of civil liberties, most likely privacy.

ing to the KYC principle, the risk should be calculated based on named indicators on time the relationship is initiated and on a continuous basis.

<sup>15</sup>The Basel Committee on Banking Supervision created the Basel II accord containing a three-pillar concept with recommendations on regulations and banking law issues like the amount of capital a bank has to put in reserve to overcome financial and operational risks. Money laundering is faced by a possible reputation or legal risk a bank gets maybe exposed to. See <http://www.bis.org/publ/bcbsca.htm>.

<sup>16</sup>Covering Identity Theft Prevention and Credit History Restoration (Fraud, money laundering), Improvements in Use of and Consumer Access to Credit Information, Enhancing the Accuracy of Consumer Report Information and so forth. See [http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=108\\_cong\\_public\\_laws&docid=f:publ159.108](http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=108_cong_public_laws&docid=f:publ159.108).

<sup>17</sup>Amongst others, GLBA defines pretexting protection, hence fraudulent access to personal non-public financial information by phone, mail or phishing. See <http://banking.senate.gov/conf/>.

<sup>18</sup>SOX defines several requirements for trustworthy financial reporting like Auditor Independence, Corporate, Criminal and Corporate Fraud Accountability. See <http://www.sarbanes-oxley.com/section.php>.

<sup>19</sup>From the United States Department of the Treasury Financial Crimes Enforcement Network (FinCEN) Advisories Number 33 (March 2003) on Informal Value Transfer Systems.

<sup>20</sup>As a side note, the authors points to the fact that if the whole world gets criminal, money laundering even becomes impossible.



Based on the sensed small impact on money laundering, increasing regulations against money laundering and hardly quantifiable costs and benefits of these counteractive measures, the authors ascertain that compared with the monetary and non-monetary costs of AML prevention for the society and the economy, the benefits are small. Geiger and Wuensch propose an in-depth review of the current approach instead of continue broadening and deepening the current AML frameworks.

### 2.1.4 Conclusion

As can be seen, there are several governmental, inter-governmental and non-governmental institutions that try to prevent money laundering by introducing direct obligations, guidelines and standards. And since almost all types of banking activities are made using a computer instead in front of a cashier's desk, the main principle to know your customer is getting hard to realize. In early days, the clerk was in direct contact with the customer and it was much easier to get the customers intentions. Nowadays, computer programs analyze and interpret transactions according to predefined and calculated usage attributes, actual law, international blacklists and so forth. In order to achieve customer satisfaction, they have to prevent false-positives as good as possible to avoid for instance, that a cash transaction originating from a recently graduated Swiss student traveling through the U.S. for the first time, therefore renting a car, paid with a credit-card, is blocked.

These technical aspects are described in the following sections together with the needs of *AlphaFin* and the implemented solution.

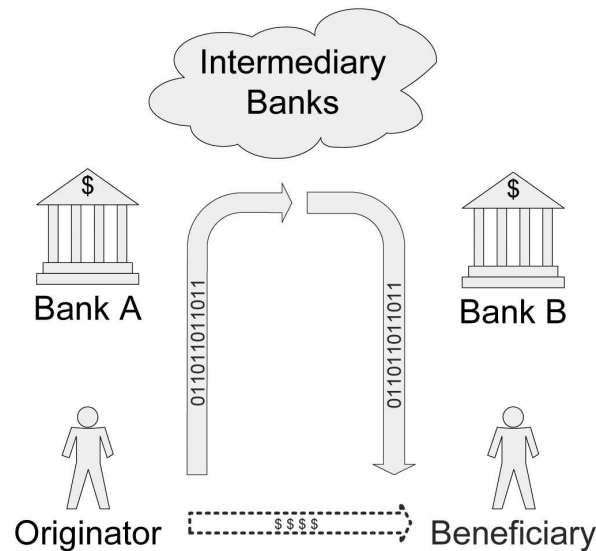
## 2.2 Money Laundering detection and related work

The last section gave an impression about the legal actions taken against financial fraud. Since global electronic (payment) transactions are widely used, the volume is increasing and resulting in the need of having them monitored constantly, autonomously and close to real-time. On the one hand, strict defined legal boundaries are monitored and automatically trigger an alarm. Such examples are transaction amount limits that are crossed or entries from an international blacklist that are hit. On the other hand, the behavior of an individual actor is labeled, stored and monitored over time. Fraudulent behavior is then detected as a significant deviation from the normal condition. But because the criminal mind is visionary and adaptive, criminals usually find ways to bypass existing measures [Bolton and Hand, 2002].

In this section the current approaches and some available commercial products are presented.

## 2.2.1 Introduction

Wire transactions are data telegrams sent from one actor to another. As shows in Figure 2.1, the originator instructs Bank A to send money via a chain of intermediaries to the beneficiaries Bank B. Both, payer and payee could be a person or business. In an international perspective, the Society for Worldwide Interbank Financial Telecommunication (SWIFT) provides a network for interchanging financial messages between all types of financial and non-financial actors. The messages are built according to the SWIFT Standards identified by an message type<sup>21</sup>. Depending on this type, the telegram contains items such as the identity of originator, beneficiary and the participating banks, transaction date, the transaction amount and some free text fields. Sometimes fields are left blank, filled with internal data or at the worst contains errors. Automatic error detection software based on semantic and syntactic constraints is needed to prevent incomplete messages [Bolton and Hand, 2002].



**Figure 2.1:** The flow of money and informations in a wire transaction according to [Chang et al., 2008].

In contrast to credit card and telecommunication fraud, where the fraud normally comes to light at an early stage, time is a significant factor in anti-money laundering efforts. Years can go by until the laundering process is detected, all accounts and involved parties are identified. Therefore, improvements in the range of customer record management systems are needed.

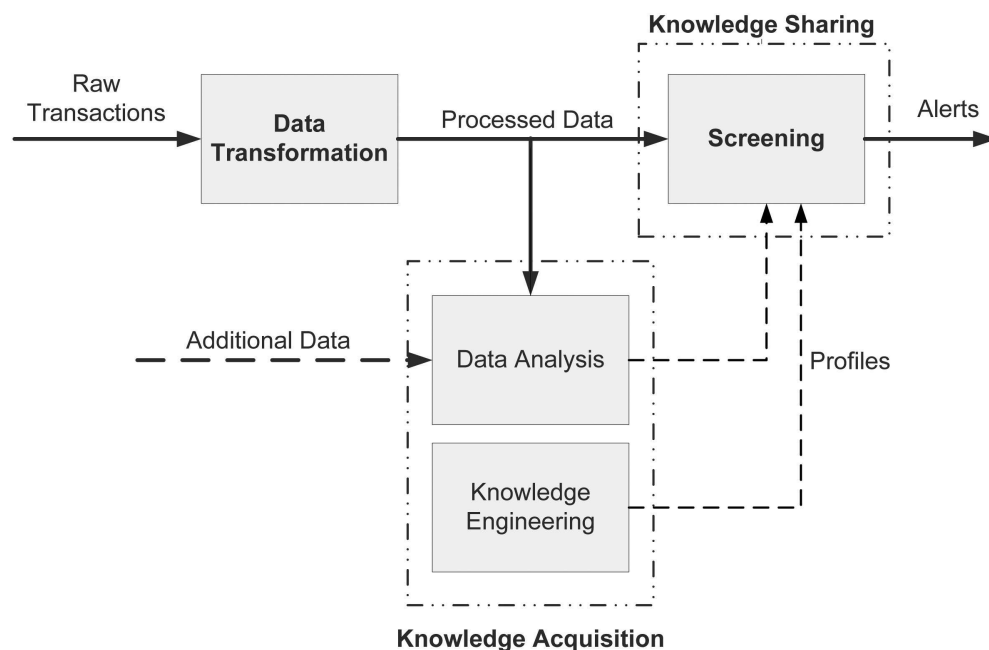
In early days, fraud detection was performed manually triggered by some fixed size transaction amounts. In order to match the tactics of the authorities, the money launderers divided larger sums into multiple smaller amounts and deposit them in different banks [Bolton and Hand, 2002]. This practice, termed smurfing or structuring, shows that anti-money laundering systems must

<sup>21</sup>For example, MT1xx stands for an message concerning customer payments and cheques; MT2xx for a financial institution transfer message. See <http://www.swift.com/solutions/standards/index.page>.

continuously learn how to react on new proceedings. Switching between wire and cash movements, using single transfers of low amount, use of complex bogus companies or false invoicing is used to obfuscate fraudulent actions. The situation gets even more complicated by the fact that banks often won't share all the details of their findings because of strong competition, confidentiality and privacy reasons.

### 2.2.2 Basic money laundering detection pattern

A proposal for wire transfer monitoring was published by the U.S. Congress, Office of Technology Assessment (OTA) in 1995. They list four categories of technologies needed for detecting money laundering according to Figure 2.2.



**Figure 2.2:** Core technologies proposal according to [OTA, 1995].

1. *Data transformation* is used to normalize input data in terms of removing variations in format and spelling; the determination of the real originator and beneficiary even if multiple accounts numbers are used and the creation of new data records by aggregating input.
2. *Knowledge acquisition* is used to construct new profiles used for the screening step later on. Therefore, knowledge-based systems are used to make inferences on fraud detection. They are constructed by knowledge engineering (e.g. interviewing AML experts and creating rules) and knowledge discovery techniques such as machine learning, statistical model building, cluster analysis and case-based reasoning. In addition, visualization techniques

are used allowing human analysts to examine data by recognizing patterns based on their own knowledge. Additional data from several input sources like financial reports and governmental registries is aggregated here too.

3. *Wire transfer screening* is the central part of the pattern including knowledge-based systems (according to knowledge acquisition) and link analysis. It uses the processed input data and the created profiles to create alerts. In most cases this alerts are processed by an AML expert and a SAR will be filled, if the suspicion is substantiated. Link analysis techniques explore associations among a large number of objects with different types. This functionality is needed to uncover connections between individual accounts, people and organizations and support the completion of existing profiles. Visualization is again needed for AML experts to reduce the amount of data to some meaningful dimensions.
4. *Knowledge sharing* allows a quick and reliable sharing of profiles in a useful form, between knowledge-based systems installed at multiple locations. Such locations are at wire transfer systems, like Fedwire<sup>22</sup> or SWIFT; at money center banks and maybe at all banks. Because money laundering approaches can change rapidly, profiles in knowledge-based systems would have to change as well.

As we will see, this pattern is largely used in most of the commercial products in Section 2.2.4 as well as for our *TMatch* and *TMatchViz* system.

### 2.2.3 Related work

Together with the enhancements in modern technologies, the use of information technology to discover and protect against all kind of fraud, increased as well. [Phua et al., 2005] categorized, compared and summarized 51 technical and review articles in automated fraud detection between 1994 and 2005. The proposed categories contribute towards our understanding of the problem and are matter of the next subsection. [Phua et al., 2005] criticizes that in most real-world scenarios, the data mining technique is chosen based on the emerging practical issues of operation requirements and of managerial commitment instead of technical aspects. In addition, the authors perceive that are the time of writing, researchers tend to create complex algorithms like neural network, which are outperformed by less complex and faster algorithms such as Bayesian and logistic regression in the long term.

Another survey based on data mining technologies was written by [Yue et al., 2007]. They found 15 publications focusing on financial statement fraud between 1995 and 2007. This survey supplements the work of [Phua et al., 2005] by adding some details on possible data sources and data mining tasks and proposing a generic data mining based financial fraud detection framework. According to the found publications on detecting financial statement fraud, [Yue et al.,

---

<sup>22</sup>A real time gross settlement funds transfer system run by the U.S. Federal Reserve Banks. See <http://www.federalreserve.gov/paymentsystems>.

2007] state that most of the detection algorithms methods were using regression models or artificial neural networks with accuracy rates of more than 84 per cent. Bayesian based mechanisms were used rarely.

## Four Categories

In order to organize all the different fraud detection mechanisms [Phua et al., 2005] propose the following classification into four categories.

1. *Supervised approaches on labeled data.* Supervised approaches use an internal prediction model trained by evaluating labeled test data sets. The internal model is based on Bayesian [Ezawa and Norton, 1995; Maes et al., 1993] or (supervised) neural networks [Maes et al., 1993], decision trees, case based reasoning, statistical modeling, genetic programming or is connected to expert and rule based systems. The type of algorithm is defined based on the desired characteristics such as learning time, calculation speed, robustness or precision and on restrictions like data format and the value range. After learning, the model should be able to separate fraudulent and non-fraudulent data sets reliably.
2. *Hybrid approaches with labeled data.* These approaches are similar to the first category, but use more than one algorithm to decide if data is fraudulent or not [Maes et al., 1993]; a consequence of the different characteristics and requirements. According to [Phua et al., 2005], most of the hybrid approaches contain only supervised learning algorithms but there exist some unsupervised hybrids too, learning only unlabeled data. The mixed hybrids undercut the results of the pure supervised ones.
3. *Semi-supervised approached with only legal (non fraud) data.* In this category, the approaches use only non-fraudulent data for training the internal prediction model. Fraud is detected if an input data set differs sufficiently later on. According to Phua et al. [Phua et al., 2005], these approaches were used in detecting internal fraud and telecommunication fraud. In the latter case the approach was to calculate the daily behavior profile and comparing it with overall profile. If the daily profile exceeds the overall profile by a predefined threshold, a fraudulent situation was found.
4. *Unsupervised approaches.* This fourth category contains unsupervised approaches based on processing only unlabeled data sets. They usually detect changes in behavior of the observed entity compared to a predicted profile [Kingdon, 2004; Fawcett et al., 1997]. They are vulnerable to a high false-positive rate, if configured to react highly sensible. The used techniques are link analysis, (graph-based) pattern matching [Ullmann, 1976; Gallagher, 2006; Foggia et al., 2001; Cheng et al., 2008], graph mining, peer group analysis [Weston et al., 2008; Bolton et al., 2001], unsupervised neural networks like hidden Markov models. Again, the proper technique is based on the desired characteristics like learning time, calculation speed, robustness or precision.

## Conclusion

As already mentioned, the chosen approach for the final product should depend on the practical usage of the gained results and the intended level of integration [Phua et al., 2005]. If the AML system has the purpose to process unstructured data from several text-based data sources the output will most likely be text-based too, which often requires sophisticated rule based system or workflows, including human expert for further processing. In contrary, in the case of structured data, the AML system is able to make a reliable autonomous decision by, for example, setting a flag, raising an alert or even denying the transaction in real time. That is very probably the reason why all found commercial products in Section 2.2.4 use a combination of different techniques like rule based systems for sanction or PEP screening, data mining techniques for transaction monitoring and profiling of the customers activities and link analysis to highlight hidden linkages between transactions, customers and accounts.

In order to tie with the critics of [Phua et al., 2005], we faced a similar problem with the algorithms in TMatch and the interpretation of the results. If the user has to decide how to proceed with an array of numerical results or which modules he should use for a job, all of them have to be straightforward to understand even without the whole knowledge about the internal computation.

At *AlphaFin*, labeled fraudulent data was not available and all findings were reported in unstructured text without reference to the actual data [Luell and Bernstein, 2009]. Supervised approaches are only used in terms of keyword detection and rule matching. The fourth category of approaches containing graph mining and visual graph analysis which, as you will see in Section 3.4, were used for the *TMatch* and *TMatchViz* systems too. The next subsections will present recent work in these two fields.

## Graph-based data mining

For AML purposes, a transaction between an originator  $A$  and beneficiary  $B$  of a given amount  $W$  can be seen as a directed weighted graph structure containing two nodes  $A$  and  $B$  and one edge from  $A$  to  $B$  with weight  $W$ . If this transaction is repeated over time, this simple graph is changed to a directed weighted multigraph containing several edges from  $A$  to  $B$ .

The basic *graph pattern matching* problem is to find a target graph in another graph structure. According to [Gallagher, 2006; Black and Lovrencic, 2004], this is formally defined as:

1. A data graph  $G$  is composed of a set of vertices  $V$  and edges  $E$ . Each edge  $e \in E$  is a vertex pair  $(v_i, v_j)$  where  $v_i, v_j \in V$ .
2. The target (pattern) graph is defined as graph  $P = (V_p, E_p)$ .
3. For pattern matching, a subgraph  $G'$  of  $G$  is matching if  $G' = (V', E')$  and  $V' \subseteq V$  and  $E' \subseteq E$ . The final result set  $S$  is defined as  $S \subseteq G$ .

A common problem in working with graph structures is to find such subgraphs in a set of graphs. According to [Black and Lovrencic, 2004], a subgraph is defined as graph whose vertices and edges are subsets of another graph structure, or formally: The graph  $G' = (V', E')$  is a subgraph of a data graph  $G = (V, E)$ , iff

1.  $V' \subseteq V$  and
2.  $E' \subseteq E \wedge ((v_i, v_j) \in E' \rightarrow v_i, v_j \in V')$ .

Subgraph isomorphism is known to be an NP complete problem [Gallagher, 2006; Foggia et al., 2001], having a complexity of  $O(n^k)$ . Time requirements of brute force matching algorithms increase polynomially with the size  $n$  of the input graph [Foggia et al., 2001], which causes long computational times and high memory requirements. At *AlphaFin* computational resources like disk-space and memory can be a critical matter of expense [Luell and Bernstein, 2009]. Depending on the task, pattern matching does not have to provide exact matching. Inexact matching algorithms enable the use of wildcards or cardinality operators and are faster to solve, since the target pattern itself is imprecise compared to the exact search pattern.

In *graph mining*, the goal is to find a set of most common or interesting patterns in a graph [Gallagher, 2006]. One method is the mining of frequent subgraphs [Yan and Han, 2002; Han and Kamber, 2000]. Therefore, the definition of graph pattern matching is enhanced by a new function.  $frequency(G')$  is the number (or percentage) of existing subgraphs  $G'$  in the data graph  $G$ . A *frequent* subgraph  $S'$  is a graph with  $frequency(S') \geq min\_freq$ , a threshold value. The generation of frequent subgraphs candidates  $G'$  is done by starting with a small graph and adding additional vertices and edges; called "Apriori levelwise approach". *gSpan* [Yan and Han, 2002] is an approach towards better candidate generation.

Eberle and Holder [Eberle and Holder, 2007] developed new algorithms for analyzing graph substructures. They detect structural anomalies, caused by modifications, insertions and deletions of graph components. The algorithms are based on the fact that fraudsters will try to hide their actions by behaving as close as possible to legitimate activities. The algorithms work for data sets containing all the needed information to detect different types of anomalies. As shown by the authors, this works for cargo shipment and network intrusion detection because of their recurrent patterns. As an example, in cargo shipment an extra traversed port or a missing statement in the shipment documents implies a certain probability of smuggling.

However, for AML purposes the data sets and customer profiles are potentially incomplete and consist of the tacit knowledge of the customer consultant according to the KYC principle. Depending on the customers business, the financial transaction won't even be recurrent. Nevertheless, having the transactions in a graph structure allows the use of graph theory such as subgraphs, network routing and flow calculations [Blessing, 1998]. As you will see in chapter 3, *TMatch* will use graph-based data mining for data processing.

## Visual Explorations

After the Visual Analytics Science Technology Symposium in autumn 2007, [Ribarsky and Dill, 2008] refer to a strong interest in collaboration between visualization scientists and researchers from areas like mathematical and statistical methods, social analyses or financial analytics happened. Fraud discovery is strongly influenced too. Ziegler et al. [Ziegler et al., 2007] describe an approach to detect atypical behavior in financial markets based on two dimensional color maps. Therefore, they monitor the performance of some chosen assets, calculate the significant characteristics and compare them over time to the performance of the whole market. The color map covers a range of colors from red, standing for negative growth and loss, over beige, meaning unchanged growth, to green for positive growth and profits. The two used dimensions are time of purchase and time of sale. After the color map is calculated for a specific asset, it reveals whether the asset behaves differently that most others in the past, or whether another asset would have been a better investment and other correlations. These uncovered facts are easily recognized for a human eye compared to the large sized data sets standing behind the image.

In money laundering, the layering step in Section 2.1.2 is done for instance by buying assets over price and selling at a loss. The loss resulting from the deal is small compared that the residual money has now a cleaner history [Altenkirch, 2006]. It is quite possible that an AML specialist obtains a new visualizations-based instrument to detect layering attempts.

Another approach [Huang et al., 2009] uses visual surveillance of market performance and a behavior-driven visual analysis of trading networks to detect financial fraud. Therefore, 3D treemaps and social networking visualization are used as visualization techniques.

WireVis is a tool for the analysis of financial wire transactions for fraud protection, built and described by [Chang et al., 2008] in collaboration with the Bank of America. It uses a multi-view approach including a keyword heatmap and a keyword network graph to visualize the relationships among accounts, time and keywords within wire transactions. All the textual elements contained in a transaction data record are seen as keywords. Whether a keyword is indicating high risk or not is defined by several keyword lists, based on intelligence reports, previous analyses and input from third parties. The analyst has the ability to aggregate and organize groups of transactions and to drill-down into single records and compare them. If for example a keyword shows abnormal temporal patterns the information provided in the different views let the analyst locate all involved parties and start further investigations.

However, for wire transactions it seems to be adequate to use a keyword-based temporal approach to detect money laundering attempts. But since a transaction data store contains more valuable information like the transaction amount and accounts involved over time, there is an imminent risk of overlooking important evidence.



## 2.2.4 Available anti-money laundering solutions

As expected, most of the commercial AML software solutions are hardly documented for the public and their internal mode of operation is a well-guarded secret, opened only to the customers.

An exception is found in the publication of *Searchspace*<sup>23</sup> approach to fight money laundering. J. Kingdon [Kingdon, 2004] presents the proceedings towards building an artificial intelligence based AML system. Searchspace used a multidimensional adaptive probabilistic matrix to set the customers behavioral pattern and track it over time. The used dimensions were customers, accounts, products, geography and time. In order to improve stability and thereby the detection results, they added a threshold-based monitoring module. A peer group analysis module added the ability to distinguish normal and suspect transactions based on certain criteria such as account type and branch of trade. The author states a 1-in-14 false positive rate of that system. Information on the false-negative rate, estimated missed positives and deployment in other banks are left out.

In august 2006, Celent<sup>24</sup>, an American research and advisory firm, published an evaluation on AML solution vendors. They created a framework containing four categories such as advanced technology, breadth of functionality, customer base and depth of client services applied to watch list filtering and transaction monitoring products. Katkov [Katkov, 2006a] gives an overview on the evaluated transaction monitoring solutions vendors. Vendors selling a feature rich and advanced technical product are Norkom Technologies<sup>25</sup>, Mantas<sup>26</sup>, NetEconomy<sup>27</sup>, SAS Institute<sup>28</sup> and Fortent. Fiserv put its part of the Celent evaluation results online. According to [Katkov, 2006b], their ERASE financial crime suite contains transaction monitoring using mechanisms like rule-based analysis, creation of account profiles, peer group analysis for risk scoring. Users have several predefined configurations to choose from as well as rich visualization tools to display transaction activity over time and compare it to average behavior, influencing scoring properties and profiles (Figure 2.3) or network visualization including account relationships and transaction flows between them. In addition they provide watch list filtering for real-time screening of wire transactions and account transactions, supporting global and governmental high-risk lists. An automated alert investigation module supports the automatic investigation of manual alerts or SARs; a case management module manages the investigation workflow and finally a reporting toolkit generates files for regulatory reports including audit trails of activity and case history.

## Conclusion

It is surely not surprising that the complete evaluation paper is only available for Celent customers. The information on the vendor web pages implies that it crucial to have a risk-based approach in detecting AML attempts. Most of the current commercial solutions use rule-based

<sup>23</sup>In June 2006, Searchspace and Semagix merge and formed Fortent, a company working in the risk and compliance area. See <http://www.fortent.com>.

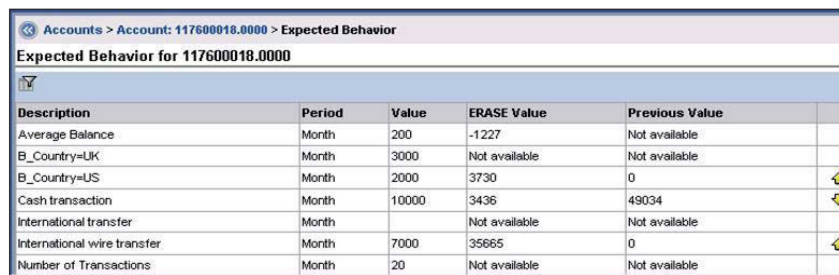
<sup>24</sup>See <http://www.celent.com>.

<sup>25</sup>See <http://www.norkom.com>.

<sup>26</sup>Is now a part of Oracle. See <http://www.oracle.com/us/industries/financial-services>.

<sup>27</sup>NetEconomy is now a business unit of Fiserv, Inc. See <http://www.aml.fiserv.com>.

<sup>28</sup>See <http://www.sas.com>.



Accounts > Account: 117600018.0000 > Expected Behavior				
Expected Behavior for 117600018.0000				
Description	Period	Value	ERASE Value	Previous Value
Average Balance	Month	200	-1227	Not available
B_Country=UK	Month	3000	Not available	Not available
B_Country=US	Month	2000	3730	0
Cash transaction	Month	10000	3436	49034
International transfer	Month		Not available	Not available
International wire transfer	Month	7000	35665	0
Number of Transactions	Month	20	Not available	Not available

Figure 2.3: User interface to modify an behavior profile in Fiserv' Dynamic Risk Scoring Module.

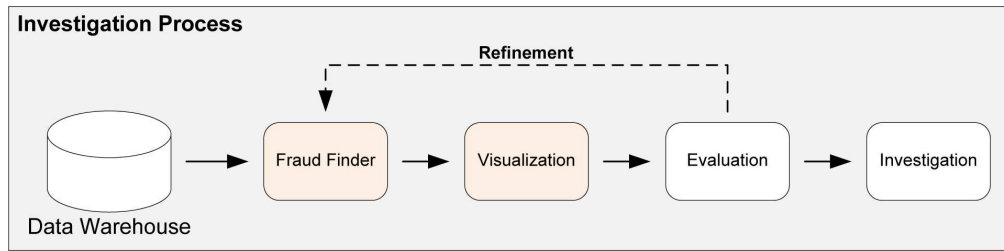
techniques, generate behavior profiles over time using link and group analysis to enrich and compare them and include visual analysis tools. In terms of [Phua et al., 2005] the mechanisms are often from the second and forth category.

## 2.3 Suggested solution

After this introduction into the legal, organizational and technical background needed to fight money laundering, an AML solution for *AlphaFin* should contain a risk-based approach to lower the false-positives rate, the load on the monitoring system, and finally the load of the AML specialists. Furthermore it should contain an intuitive graphical workbench where analysts are able to track fraudulent actions in place and time and set up intuitive filters to focus on relevant data. Legal regulations, international and governmental risk-keyword lists should be considered automatically. If an action hits a certain risk level, an alarm should be sent to the user containing all needed details such as reason of the alarm, all the action details and possible links to existing user profiles. But instead of waiting until a legal or governmental boundary is hit, the solution should be able to detect uncommon financial transactions patterns, rate them and present them to the analysts in suitable format. Figure 2.4 shows the proposed investigation process for *AlphaFin*. The evaluation and, if enough evidence is found, the investigation phase is executed by AML specialists. Nowadays, human resources are expensive. The concept to replace human analysts by information technology contains the risk of losing knowledge in detecting fraudulent behavior. Simply spoken, IT systems work in terms of probabilities and triggers, humans in terms of facts and feelings. In order to combine both worlds should be the aim of this solution.

### 2.3.1 Requirements of the solution

As previously mentioned, *AlphaFin* has a data mart containing financial transactions from several years. Each day a huge amount of new data is stored. Data access and handling is severely limited due to strict confidentiality regulations as well as the computational resources are very limited [Luell and Bernstein, 2009]. The solution has to be built on plain SQL queries, working



**Figure 2.4:** The proposed investigation process.

on a secured IBM DB2<sup>29</sup> database server. Stored procedures should be avoided; the creation of temporary tables is allowed. The final solution should scale to analyze hundreds-of-thousands up to millions of transactions in feasible time. The output has to be stored into another table for further processing by *TV*, the transaction visualizer, or other analysis tools.

As proposed, the final *AlphaFin* solution should use a risk-based approach for an optimal resource allocation. Within our approach, this is done by additional prefiltering of the input database by attributes, like the geographical location or the business category of the account holder. As a direct consequence, the main data source table should be configurable having a regular table, a SQL view or a materialized query table type.

The found transaction patterns should be scored in several ways to assist an AML expert in further processing. The scoring mechanisms and results have to be intuitive and easy to understand. In order to develop the final specifications, one-way prototypes of the algorithm parts have to be build and tested together with an AML expert.

The solutions should contain a modern graphical user interface with the possibilities to configure the mining and scoring algorithms, to start and observe the search process and to evaluate the results. Error messages should be displayed in a suitable way.

Due to the fact, that neither real world data nor fraudulent data examples are available [Luell and Bernstein, 2009] and each testing prototype has to be authorized first, a feasible project management and engineering model is required.

The next section describes the implementation of *TMatch* and *TMatchViz*, the made assumptions and design choices. In Chapter 4 we will evaluate the proposed solution.

<sup>29</sup>See <http://www.ibm.com/db2>.



# 3

## Approach Description

Since *AlphaFin* already has an AML section using tools to detect fraudulent actions, the main task of this thesis is to add a new approach for finding fraudulent data. This part describes the implementation of *TMatch* and *TMatchViz*.

### 3.1 Overview

As one can see in Figure 3.1, our solution contains three steps. In a first step, the *AlphaFin* data warehouse (base view) is queried to create an alert graph view on the interested data. The attributes and constraints to generate this view are based on possible previous alerts or given by compliance-related reasons. The creation of this shortened dataset is large step towards higher performance and supposed lower false-positives rate. Because of the data transformation, the alert graph view contains only a defined set of columns, needed for the next step. This is another improvement of speed for data retrieval. Responsible for this step is the *Base Pattern Matcher* which does the transformation and some analysis, depending on its configuration. At *AlphaFin*, a tool called *ChainFinder* is used [Luell and Bernstein, 2009] and create the datasets, *TMatch* works with. This data transformation is seen in Figure 3.2.

The proper data analysis is done in step two by the *TMatch* module. Therefore the transaction data, generated in step one, is reused as a big graph structure and evaluated by *TMatch*. The details for this module are written in Section 3.4. After this step, the database will contain the found graph components and the associated scorings.

In the last step, the module *TMatchViz* is used to display the findings from *TMatch* in an appropriate way. For convenience, *TMatchViz* contains all visible aspects of this solution including the parameterization of *TMatch* and its execution. After finishing the job and displaying the results, *TMatchViz* offers a first possibility to evaluate the findings by an operator. The operator is able to compare, visualize or remove entries. In order to complete the workflow, the *Transaction Visualizer (TV)* is used for further processing of the results.

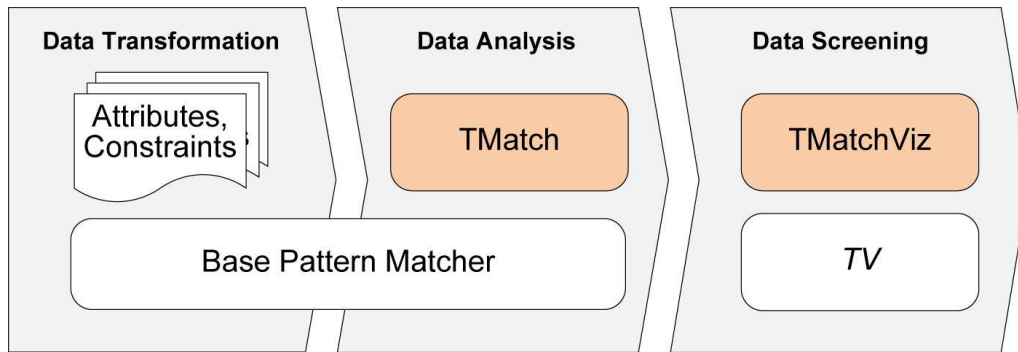


Figure 3.1: Architectural overview.

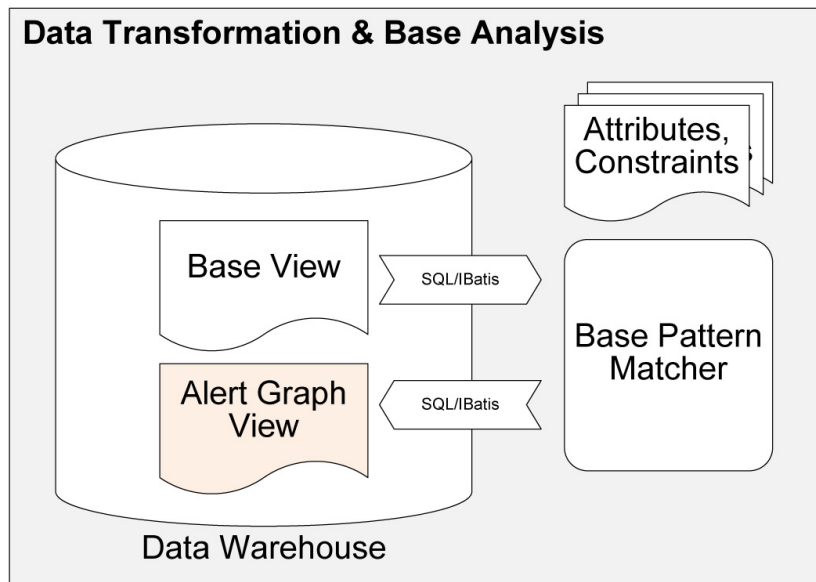


Figure 3.2: Overview data transformation.

## 3.2 Used Software

*TMatch* was implemented using *Java 6 JDK*<sup>1</sup> and the newest version of *Eclipse*<sup>2</sup>. The graph data structures and some of the graph algorithms were provided by *JGraphT*<sup>3</sup>.

*TMatchViz* is based on NetBeans swing application framework<sup>4</sup> implementation *appframework*<sup>5</sup>. For visualization the *Prefuse*<sup>6</sup> library was used. Because of the excellent application framework support, *NetBeans IDE*<sup>7</sup> was used instead of Eclipse.

*iBATIS*<sup>8</sup> was used for DB2 database access. *iBATIS* is a persistence framework, which automates the mappings between SQL and Java objects.

For monitoring CPU and memory consumption, hot spots and leaks, the trial version of the newest *YourKit Java Profiler*<sup>9</sup> was used.

## 3.3 Assumptions

Some assumptions have been made, for developing the two modules.

- Setup. This solution is correctly installed before being used by an AML professional. The database connection is not changeable by the operator. All the needed configurations are done in *TMatchViz*.
- Computational resources. Because of the restriction, not being allowed to save temporary data on the operator's workstation, some of the algorithms were implemented using the database as storage for temporal results. In addition, because of the high memory needs of some algorithms, the database was used instead of local RAM<sup>10</sup>. The possibly resulting higher delays are accepted by *AlphaFin*. Further fine tuning should weaken this effects.
- Dataset prefiltering. As seen in 3.1, *TMatch* is optimized for accessing prefiltered alert tables instead of the whole data mart table size. This is another limitation in order of the computational restrictions.
- User-friendly. The solution has to act user-oriented and stable. In science, solutions are often very sophisticated, prototypically and the results hard to understand. That is why the used algorithms were built returning intuitive results, or changed to do so.

---

<sup>1</sup>See <http://java.sun.com/javase>.

<sup>2</sup>See <http://www.eclipse.org>.

<sup>3</sup>See <http://jgrapht.sourceforge.net>.

<sup>4</sup>See <http://jcp.org/en/jsr/detail?id=296>.

<sup>5</sup>See <https://appframework.dev.java.net>.

<sup>6</sup>See <http://prefuse.org>.

<sup>7</sup>See <http://www.netbeans.org>.

<sup>8</sup>See <http://ibatis.apache.org>.

<sup>9</sup>See <http://www.yourkit.com>.

<sup>10</sup>Abbreviation for Random-Access Memory. See Sections 3.4.4, 3.4.5 and 3.4.6 for further details.

- Specifications. The specifications of the algorithms were developed over time in agreement with an AML expert. They changed according to the actual needs and new suggestions untill the end of this thesis.

Additional (programmatically) assumptions are mentioned in the following sections.

## 3.4 Implementation of TMatch

The following subsections contain the implementation details of *TMatch*. In order to simplify the reuse of this component, a jar library is generated, providing a public API and java-doc<sup>11</sup> pages. This library is used for the implementation of *TMatchViz* in Section 3.5.

Informations on how to add new algorithms or scorer models to *TMatch* are found in Section A.3. The creation of the *TMatch* library is described in Section A.4.

### 3.4.1 Architectural view

Figure 3.3 shows an architectural view of *TMatch*. A global configuration registry administers all the configuration entities needed. The runner object coordinates the execution sequence of the graph expander module, the component calculation and the scoring part. The graph expander works with the data from the alert graph view. This view is created by the predecessor component; the base pattern matcher. As mentioned before, the database access is made using iBATIS and sometimes for convenient or technical reasons plain SQL syntax. The graph expander creates a JGraphT graph structure containing all the transactions of interest, the component calculation module extract all weak components out of the structure and the scorers evaluate the different weak components. The graph components and scorings are saved to the database.

All output is logged by a central logging facade. Further information on how to activate or deactivate logging is given in Section A.8.

### 3.4.2 Configuration

A global configuration is needed, since there are many different parameters that have to be set for each run. The registry is implemented by a Java Properties class. This offers the advantage, that different files for system and user properties are supported. At runtime the files are merged together providing a single configuration access. The keys of the entries are defined at the respective classes, together with public getter and setter methods if needed. This global registry is invoked by the *TMatchViz* system too, using the according API methods. The default configuration values are shown in Table A.2.

---

<sup>11</sup>See <http://java.sun.com/j2se/javadoc>.



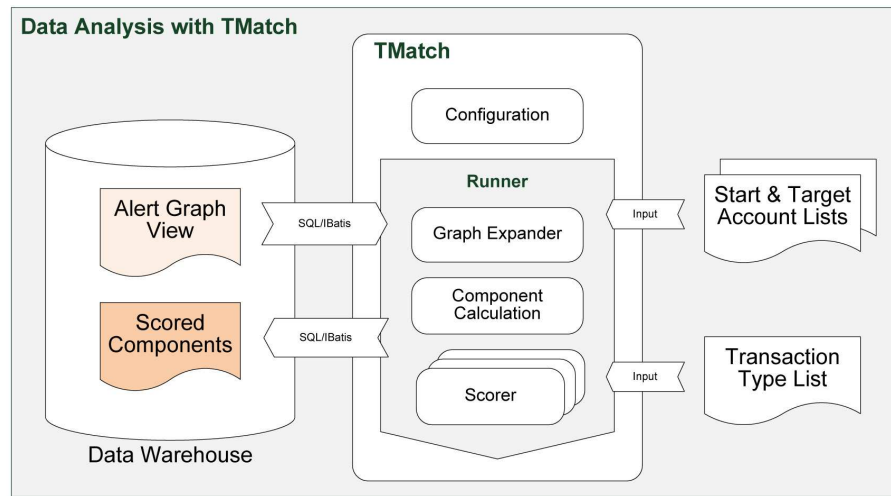


Figure 3.3: Overview TMatch.

### 3.4.3 Runner

The runner object implements the workflow of *TMatch*. It coordinates and configures the different components and the database storage, according to the pipe and filters pattern. As mentioned, the memory consumption had to be kept low. For this reason the invoked components are configured to use temporarily results stored in the database instead of using local RAM, like Java Collection objects<sup>12</sup>. In a first step, this object verifies the database access and all the needed SQL functionality. This is important, due to some of the unique DB2 SQL statements used in the queries.

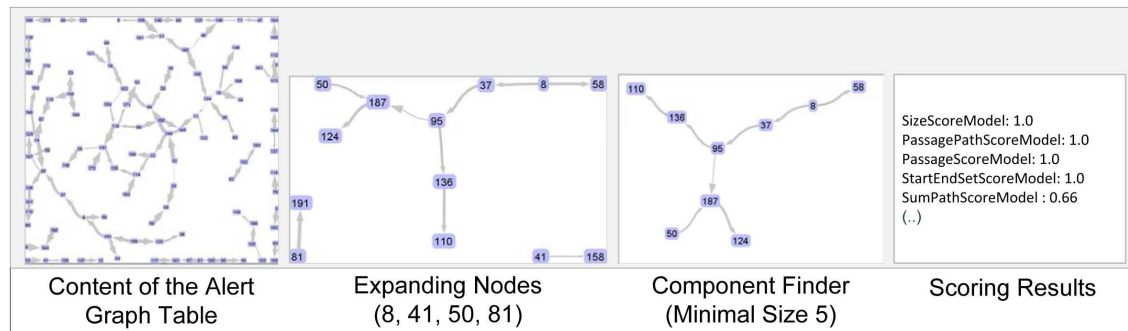
### 3.4.4 Expander

The expander module evaluates a set of interesting nodes to analyze possible connections between them and others, yet unknown nodes. Therefore it loads the account data from the configured start and target account lists, merges them and accesses the alert graph view like a directed multigraph. The entities from the account lists are now the starting points for the search. The search is finished for a specific start-node, if there are no connections left, a loop is detected, another interesting node is hit or the search depth reaches a configured maximum threshold. The decision if a new node should be taken into account is implemented by the class method `ch.uzh.tmatch.algorithm.NodeSetExpander.isInteresting()`. This method handle all the value range and flag verification according to the configured parameters. In Listing 3.1 the pseudo code of the expander class is shown.

A multi-threaded expander was built, to exploit the (marshalling and unmarshalling) time,

<sup>12</sup>Originally, the results from the first components were the input of the second. But since some of the results are kept in the database, the pipe and filter pattern was not applicable in all cases.

iBATIS needs to build and transmit queries or receive results. For the coordination of the expanding process, some Java Collection objects were used. The implementation is thread-safe, includes no prepared statements and works with results, saved in temporal tables; therefore with lower RAM consumption for the local computer. The runtime behavior of the different threads is shown in Figure 4.5.



**Figure 3.4:** TMatch workflow: transaction in the alert graph table; expander, component finder and scorer output.

Figure 3.4 shows the data of the alert graph table and the expanding results. Visible are three different components in the result set, yet combined in the same graph instance. The expander converts a multigraph into a single graph for calculation and node filtering, according to the configuration. The runner component defines how the conversion for edges with same node-pairs and direction works: adding up the edge weight values, or counting the number of edges. The result of this expanding step is an instance of the JGraphT *DirectedWeightedMultigraph* class, assigned to the next filter.

According to the specifications, active nodes are of interest; highly interdependent structures are rather uninteresting. A node is seen as active, if there are several connections or different amounts assigned over time.

Table 3.1 shows the available filter settings and their meanings. The default values are found in Section A.7. These settings have an immediate impact on the size of the resulting node set based on the examination of each node. The component calculation part uses the similar settings, but they are applied on the aggregated component member nodes instead of the underlying alert graph nodes.

### 3.4.5 Component Calculation

According to Figure 3.4, the component calculation module, or component finder (CF), use the results from the previous expander step and calculates all the maximally connected components. Afterwards, CF filters the components according to the configuration, calculates some additional metrics and stores the remaining subgraphs and metrics in the database. Depending on the definition in Section 2.2.3, this step is where data mining is performed. The result is a list of subgraphs,

Property Key	Summary
NSE.DEPTH	This value defines the maximum search depth of the expander algorithm.
NSE.NUM.INCOMING.EDGES	This value defines the minimal overall number of incoming transactions, a node needs to have. If the amount is lower, the node is filtered out.
NSE.NUM.OUTGOING.EDGES	According to NSE.NUM.INCOMING.EDGES, this value defines the minimal overall number of outgoing transactions, a node needs to have.
NSE.TOTALAMOUNT.IN	This value defines the minimal overall incoming transaction amount, a node needs to have. If the amount is lower, the node is filtered out.
NSE.TOTALAMOUNT.OUT	According to NSE.TOTALAMOUNT.IN, this value defines the minimal overall outgoing transaction amount, a node needs to have.
NSE.THRESHOLD.PASSAGE.FLAG	This value defines the threshold, used for passage flag detection. A value of 0.9 means that the overall outgoing amount must be greater or equal than 90 percent of the overall incoming transaction amount.
NSE.REMOVE.LONELY.NODES	If this value is set, the expander algorithm will remove lonely nodes from the final graph structure.
NSE.DB.REUSE.EXPANDED.DATA	If this value is set, the expander algorithm will reuse previous results. This is very time-saving, if a calculation is repeated, having the component finder or scorer configuration changed.
GL.CREATE.DB.INDEX	According to this value, the expander algorithm will create or refresh database table indexes. This is very time-saving but not usable if the alert graph is a database view.

NSE stands for the namespace *NodeSetExpander*

GL stands for the namespace *Global*

**Table 3.1:** The Expander configuration keys

---

```

1 targetNodeSet := {Userinput: merged Start- and TargetNodeSet files}
2 seenNodeList := {Empty list}
3 FOR each originator in targetNodeSet DO
4 BEGIN
5     { Initialize stacks }
6     actualNodeStack := new Stack()
7     nextLevelStack := new Stack()
8
9     { Get all beneficiary nodes for this originator }
10    benList := queryDb(originator)
11    actualNodeStack.add(benList)
12
13    { Analyze until a preconfigured search depth is reached }
14    FOR i:=0 to getMaxDepth() DO
15    BEGIN
16        { Process all node of this level }
17        WHILE actualNodeStack not empty DO
18        BEGIN
19            actualTransaction := actualNodeStack.pop()
20            actualOriginator := actualTransaction.getOriginator()
21            actualBeneficiary := actualTransaction.getBeneficiary()
22
23            { Check for loops }
24            IF actualOriginator == actualBeneficiary DO
25                //Nothing.
26
27            { Check for hit }
28            ELSE IF targetNodeSet.contains(actualBeneficiary) DO
29                // Save connection in seenNodeList and database
30
31            { Check for already known nodes }
32            ELSE IF seenNodeList.contains(actualBeneficiary) DO
33                // Insert the connection or update the connection weight
34
35            { Check if this node is interesting }
36            ELSE IF isInteresting(actualBeneficiary) DO
37                // Save connection in seenNodeList and database
38                // and add it to the nextLevelStack
39        END
40        { Enter the next level: switch the actualNodeStack and -
41          nextLevelStack }
42    END
43 END

```

---

Listing 3.1: TMatch expander pseudo code.

ordered by the node size.

According to [Lovrencic, 2004] and the definitions in Section 2.2.3, a subgraph  $G' = (V', E')$  is a maximally connected component of the graph  $G = (V, E)$ , if:

1.  $G'$  is connected, and
2. for all vertices  $u$  with  $u \in V$  and  $u \notin V'$  there is no vertex  $v \in V'$  for which  $(u, v) \in E$

After the search process is finished, the metrics and flags in Listing 3.2 are calculated for all resulting subgraphs.

Metric	Summary
Graph Density	The graph density is calculated with different connection orders, to filter out "noise". A result of 1 means fully connected.
Degree Centrality	The maximal and average degree centrality is calculated for each subgraph. Therefore, the degree of each node in the subgraph is added up for the average and compared to the highest value seen for the maximum-calculation. A value of 1 for a node means that the node is connected with all other nodes in the subgraph. For the subgraph, a maximum centrality of 1 and a low average centrality are indicators of a "star structure".
Cycle-Node-Ratio	The cycle-node-ratio is calculated by creating a subgraph of all cycles. The ratio is the node size of the found cycle-subgraph, divided by the size of the analyzed subgraph from the CF results.
Total-Amount-In	This value contains the sum of all incoming transaction amounts.
Total-Amount-Out	This value contains the sum of all outgoing transaction amounts.
Total-Amount-In Flag	If the Total-Amount-In value is higher or equals than the configured <code>CF.THRESHOLD_TOTAL_AMOUNT_IN_FLAG</code> , this flag will be set to 1.
Total-Amount-Out Flag	If the Total-Amount-Out value is higher or equals than the configured <code>CF.THRESHOLD_TOTAL_AMOUNT_OUT_FLAG</code> , this flag will be set to 1.
Passage Flag	If the difference of Total-Amount-In and Total-Amount-Out is higher or equals than the configured <code>CF.THRESHOLD_PASSAGE_FLAG</code> , this flag will be set to 1.

**Table 3.2:** The calculated component metrics and flags.

A design decision was to abandon a separate search for strongly connected components, since they are of low interest and the search for weakly connected subgraphs will include them too. The JGraphT class `org.jgrapht.alg.ConnectivityInspector` was used for the connected component detection.

Table 3.3 shows the available filter settings and their meanings. The default values are found in Section A.7.

Property Key	Summary
CF.THRESHOLD.TOTAL_AMOUNT_IN_FLAG	This value defines the threshold, used for the total-amount-in flag detection. It defines the minimal overall incoming transaction amount, a node must reach to get an active flag set.
CF.THRESHOLD.TOTAL_AMOUNT_OUT_FLAG	According to the AMOUNT_IN_FLAG, this value defines the threshold for the total-amount-out flag detection.
CF.THRESHOLD.PASSAGE_FLAG	This value defines the threshold, used for passage flag detection. A value of 0.9 means that the overall outgoing amount must be greater or equal than 90 percent of the overall incoming amount.
CF.MINIMAL_COMPONENT_SIZE	This value defines the minimal component size. Components with lower sizes will be discarded.
CF.CALC.WEAK.COMPONENTS	Deprecated, since only weak connected components are used.

CF stands for the namespace *ComponentFinder*

**Table 3.3:** The Component Finder configuration keys

Property Key	Summary
GL.SCORER.MODEL.LIST	This value is a comma separated list, including all the scoring models that will be invoked for scoring.

GL stands for the namespace *Global*

**Table 3.4:** The Scorer configuration key.

### 3.4.6 Scorer

After the separation into graph components, the seven scorer models are loaded for each sub-graph, calculate the score and save it to the database. This is shown in the last step of Figure 3.4. The scorers are enabled or disabled according to the configuration.

In accordance with the specification not only the grand total but each score is sent to the database, having the scorer identification string as key. The range of the result value is configurable in the *ch.uzh.tmatch.score.AbstractScoreModel* class by the *minimalResult* and *maximalResult* variables. The decision was, to go without a configuration option for this values, because of the normalization and thereby better understanding of the results.

Table 3.4 shows the available configuration setting. The default value is found in Section A.7.

#### **ch.uzh.tmatch.score.SizeScoreModel<V, E>**

The *SizeScoreModel* calculates a score for a component based on all found components. The distribution of the resulting score lies between 0.0 and 1.0 whereas 1.0 means that there is just one component containing all nodes. A score of 0.5 is reached, if there are two components found, having the same number of nodes. The calculation is implemented as follow:

1. Sum up the nodes from all found components
2. Calculate the fraction for a particular component, by dividing the components size by the total node count

#### **ch.uzh.tmatch.score.PassageScoreModel<V, E>**

The *PassageScoreModel* calculates the amount of nodes with an active passage flag, against the size of the component. The distribution of the resulting score lies between 0.0 and 1.0, whereas 0.0 means that no active passage flags were found. The calculation is implemented as follows:

1. Sum up the nodes with an active passage flag
2. Calculate the fraction by dividing the components size by the total flag count

### **ch.uzh.tmatch.score.PassagePathScoreModel<V, E>**

The *PassagePathScoreModel* searches for connected passage-flag nodes (passage-path) in each component and calculates an overall rating. The distribution of the resulting score is between 0.0 and the highest found passage-path length. The calculation is implemented as follows:

1. Search and count connected nodes with an active passage flag
2. Calculate the min/max/average passage-path length for each component

According to the specification, only average results are used. In a component containing 4 passage-paths, a value of 5.0 means for example, that there are two paths of length 4 and two of length 6. This scorer uses a recursive approach and is very memory consuming. For highly comprehensive networks, this scorer should be turned off.

### **ch.uzh.tmatch.score.StartEndSetScoreModel<V, E>**

The *StartEndSetScoreModel* calculates a score for a component, based on the start and target account sets. The distribution of the resulting score lies between 0.0 and 1.0, whereas 0.0 means, that zero nodes matches a start/end set account. The calculation is implemented as follows:

1. Calculate a fraction by dividing the result-value range by the number of nodes in the start and end sets.
2. Walk through a component and search for nodes from the start and end account sets.
3. Add the fraction to the resulting score for each hit.

### **ch.uzh.tmatch.score.TransactionTypeCountScoreModel<V, E>**

The *TransactionTypeCountScoreModel* counts the occurrence of predefined transaction type values within all transactions of the component (e.g. "Cash-in" transactions). The transaction types are read from the transaction-type input file. In order to count all transactions, not the aggregated node-to-node edges are used. The scorer queries all the underlying transactions from the database and count the hits. The distribution of the resulting score is between 0.0 and 1.0. If the transaction type is set to "Cash-in" a result of 0.1 means that 10 percent of all the transaction in the component have that type set. The calculation is implemented as follows:

1. Count the number of transactions matching the transaction-type, according to an predefined input set
2. Count the total number of underlying transactions of this component
3. Calculate and return the result of number of type-matches divided by number-of-transactions



### ch.uzh.tmatch.score.TransactionTypeAmountScoreModel<V, E>

The *TransactionTypeAmountScoreModel* works like the *TransactionTypeCountScoreModel*, but instead of counting the occurrences, it adds the transaction amount of all found transactions. The resulting score is calculated by dividing the transaction-amount of transactions with a particular type, by the total amount of all transactions in the component. The distribution of the resulting score is between 0.0 and 1.0. If the transaction type is set to "Cash-in" a value of 0.1 means that 10 percent of all the transaction-amounts in the component originate from a transaction with that type set.

### ch.uzh.tmatch.score.MaximalFlowScoreModel<V, E>

The *MaximalFlowScoreModel* calculates the Maximum-Flow between start- and target set nodes for each component and calculates an overall rating. The distribution of the resulting score is between 0.0 and the highest calculated flow value. The calculation is implemented as follows:

1. Add an artificial start node to the component, connecting all nodes from the start set
2. Add an artificial end node to the component, connecting all nodes from the end set
3. Calculate and return the Edmonds-Karp Maximum Flow<sup>13</sup>

## 3.5 Implementation of TMatchViz

The following subsections contain the implementation details of *TMatchViz*, like the invocation of *TMatch*, the different views, and the final distribution package.

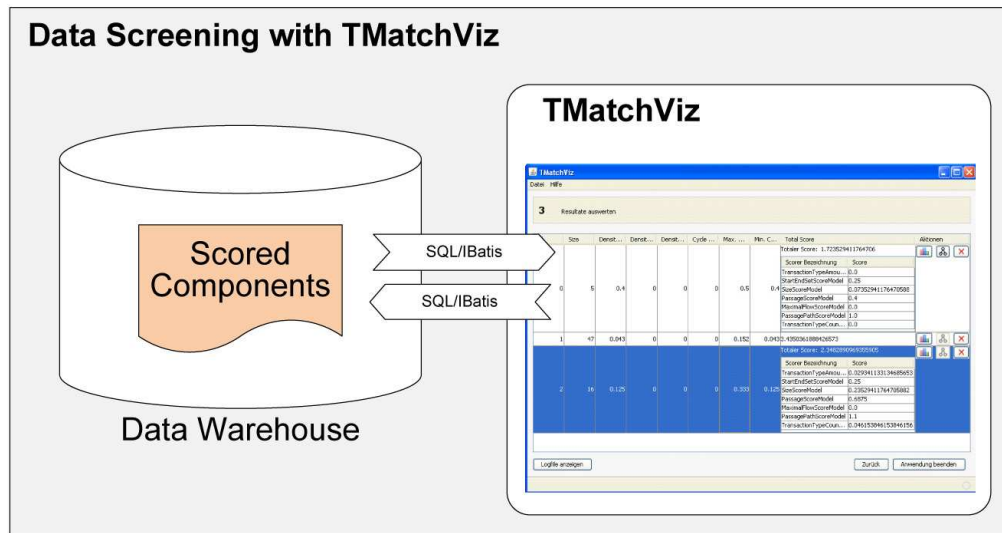
### 3.5.1 Architectural View

As mentioned in Section 2.3.1, *TMatchViz* should offer intuitive possibilities to screen the results from *TMatch*. The decision was to use an assisted approach, guiding the operator from the configuration to the results. At the same time, the visual impression should be a clear and tidy one. The views are built according to the MVC design pattern. The result screen receives the data directly from the database. For performance reasons, a query caching mechanisms was added. In order to force a logical and technical separation of the two projects, *TMatchViz* accesses *TMatch* as an included jar library, over its public API.

According to Figure 3.5, *TMatchViz* uses the scored-component tables from the data warehouse. IBATIS is again used for querying the database. After the AML expert finishes screening the results, these datasets will be used by *TV*, the Transaction Visualizer, as previously shown in Figure 3.1.

<sup>13</sup>See code example [http://en.wikibooks.org/wiki/Algorithm\\_Implementation/Graphs/Maximum\\_flow/Edmonds-Karp](http://en.wikibooks.org/wiki/Algorithm_Implementation/Graphs/Maximum_flow/Edmonds-Karp). Accessed on 25th August 2009.

To follow the specifications, all the *TMatchViz* views are created using German language. *TMatchViz* must support the operation system Microsoft Windows XP.



**Figure 3.5: Overview TMatchViz.**

### 3.5.2 Configurability

In *TMatchViz* search profiles are used for a convenient configuration of *TMatch*. The profiles are planned by business users like AML specialists, since they have the required knowledge. The implementation of the final profiles is done by an assigned developer. Instead of using the predefined profiles, an operator has the possibility to change all required parameters by using the built-in "Expert Mode". Again, the configuration is saved as the default user profile, activated after starting up the application and choosing "Use no profile" in the configuration window.

Figure 3.6 illustrates the configuration screen. The additional options of the “Expert Mode” are shown in Figure A.4.

### 3.5.3 Preselection of the results

In Figure 3.7, the stored results from the database are shown. The operator is able to unhide the intermediate scorer results, simply by clicking on the left-most action-icon. In order to get a first impression of the found component, the operator activates the built-in graph visualizer, by clicking on the middle action-icon. The selected component will appear, as shown in Figure 3.8.

The graph structure is movable, scalable and unfolds itself. If a node is selected by the mouse pointer, all direct connected nodes and edges will be highlighted. If the analyst concludes that a

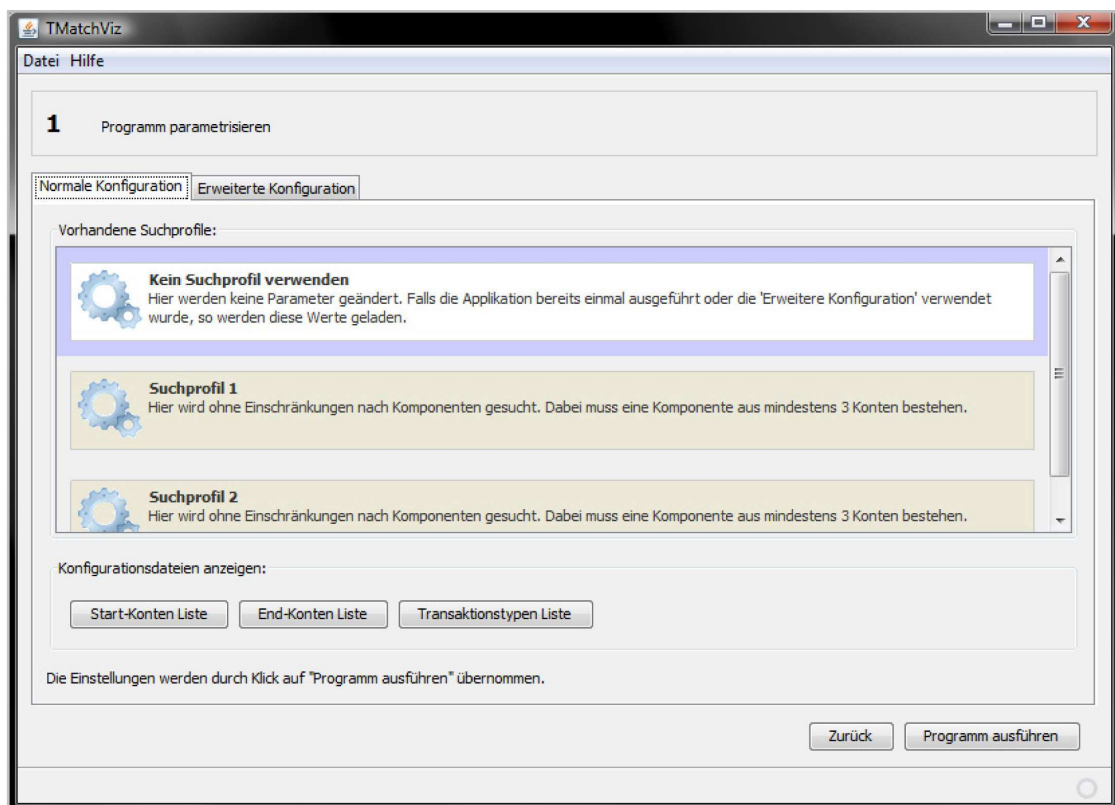
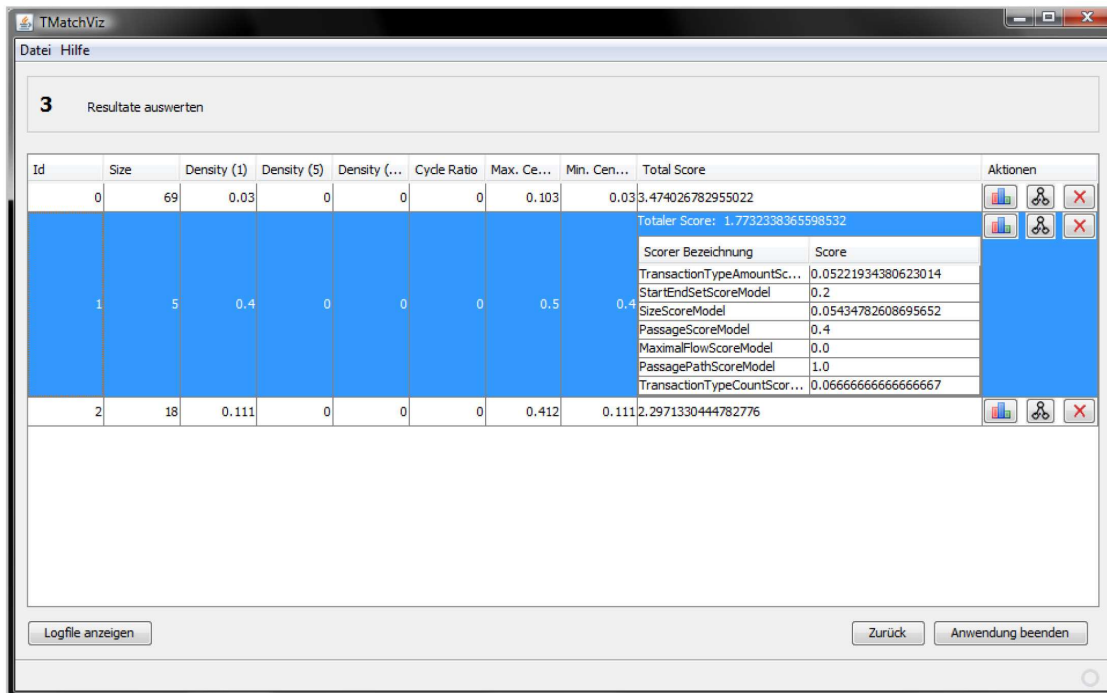


Figure 3.6: The TMatchViz configuration screen.

particular component is of no importance, he can delete it from the database by clicking on the right-most icon.



**Figure 3.7:** The TMatchViz result screen, showing found components and the intermediate scorer results.

### 3.5.4 Distribution

According to the specification, the software system must be easy to install. At *AlphaFin*, the workstations are managed by the central IT support. In order to guarantee that the software will overcome changes in the installed software base, a Java runtime 1.6 (JRE) is bundled together with *TMatchViz*. The batch file "Starter.bat" is used to setup the required execution environment variables and classpath entries. Additional memory settings<sup>14</sup> are set there too.

The *TMatchViz* resources, class files, libraries and the Java JRE files are packaged into an executable archive file. The open source archive manager software 7-Zip<sup>15</sup> was used here.

Figure A.8 shows the *TMatchViz* main directory content after unpacking; Figure A.9 the content of the library folder. The installation is described in Section A.6; the creation of a new *TMatchViz* release in Section A.5.

<sup>14</sup>Memory settings like *Xmx* for the maximum java heap size or *Xss*, the stack size for each thread, to prevent stack overflow errors while analyzing big components.

<sup>15</sup>See <http://www.7-zip.org>.

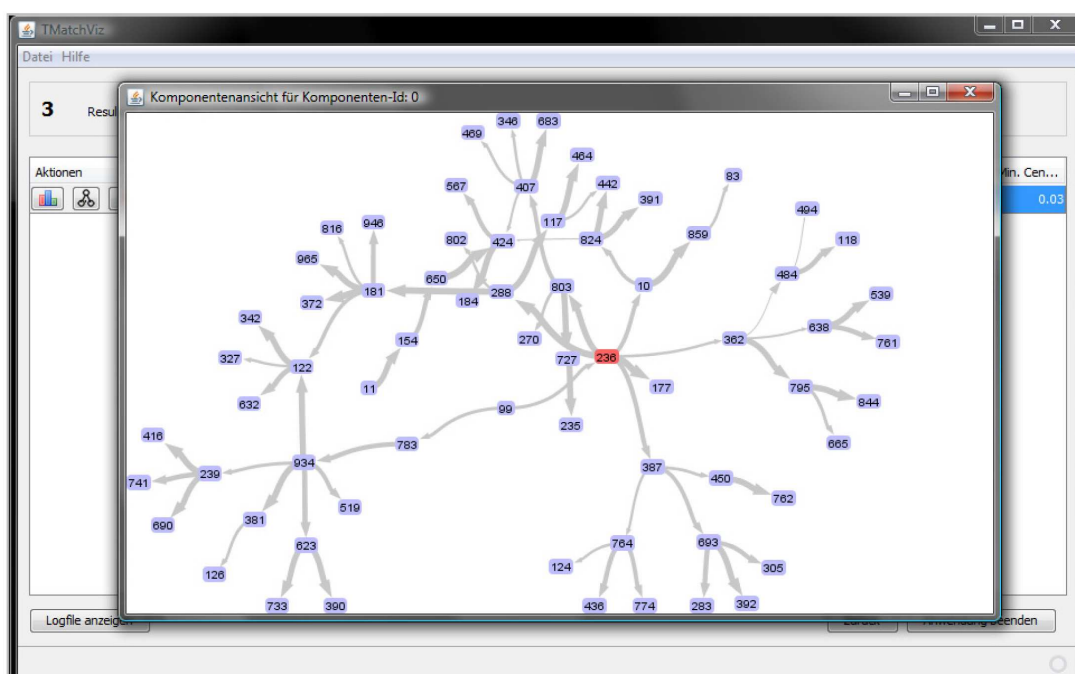


Figure 3.8: The TMatchViz result screen with activated component preview.



# 4

## Evaluation

The main goal of this Master Thesis was not to improve an existing algorithm but to create a concrete solution, based on the requirements of an AML specialist. The solutions should be used intuitively and complement the system landscape. As mentioned before, real world data was not available for testing or benchmarking purposes. The evaluation of the solution is based on the processing of synthetic datasets and should show how fast the calculation is done for several datasets.

In the next section, the experimental setup is described. In Section 4.2, the dataset generation as well as the software configuration is presented. In 4.3, the collected results are analyzed and discussed.

### 4.1 Test environment

All the experiments were conducted on an Intel Pentium 4 machine with 3.0 GHz and 1 GB RAM on a 32 bit Windows XP operating system. Hyper-threading was activated. The computer was connected directly to the university network through a 100 Mbit connection. The DB2 database server was directly connected to the university network as well. Further details about the database server configuration like connection speed, number of CPUs, installed memory, etc. are not available.

### 4.2 Test configuration

The *TMatch* library was evaluated using the *TMatchViz* interface and synthetic datasets. The datasets were created using the *sql.test.DataCreator* class. Depending on the *sql.test.TestdataProvider* class, random transaction entries were created. As seen in Listing 4.1, an entry contains a transaction id "BTX.ID", an originator and beneficiary account, a transaction date "BTX.DT", an amount and a transaction type.

---

```

BTX.ID:    1
ORIG:     584599
BEN:      245425
BTX.DT:   24.04.2009 15:11:38
AMOUNT:   95308
TYPE:     S

```

---

Listing 4.1: A sample test transaction entry from a 1M node dataset.

---

```

StartNodes.txt: 1,2,3,4,5,6,7,8,9,10
EndNodes.txt:  11,12,13,14,15,16,17,18,19,20
TransactionTypes.txt: X, Y, Z

```

---

Listing 4.2: The start node, end node and transaction type file contents.

The start node, end node and transaction type sets are shown in 4.2. These values were left unaltered for all experiments.

In a first part, the *TMatch* library was configured according to the default configuration with a search depth of 5. The default configuration properties are shown in Table A.2. The *ch.uzh.tmatch.algorithm.ThreadedDbNodeSetExpander* class was used as expander class. The measured items were maximal memory consumption and execution time of the Runner, Expander, Component Calculation and Scorer components. Since the Runner class contains the whole workflow, its result was used to calculate an existing overhead that occurs based on the database access test and creation or update of the database table indexes.

In a second part, the three different expander classes *ch.uzh.tmatch.algorithm.RamNodeSetExpander*, *ch.uzh.tmatch.algorithm.DbNodeSetExpander* and *ch.uzh.tmatch.algorithm.ThreadedDbNodeSetExpander* were compared based on maximal memory consumption and execution time.

For the measurement of the maximal memory consumption, the YourKit Java Profiler was used.

## 4.3 Test results

To simplify the evaluation charts, the number of nodes and edges was shifted by a factor of ten to obtain a logarithmic scale.

Figure 4.1 shows the results of the first evaluation part. For different numbers of nodes and edges, the node-edge ratio<sup>1</sup> was calculated and used as x-axis in all the charts. The logarithm of the ratio is plotted against the logarithm of the elapsed time. The execution time is increasing (dis-)proportionately for all curves in Figure 4.1, with a ratio between 0.0001 and 0.1. For a ratio bigger than 0.1, the time is decreasing disproportionately until a ratio of 1.0 is reached. The computation time is between 12 to 15 sec for a ratio bigger than 1.0.

---

<sup>1</sup>This is the number of nodes divided by the number of edges.



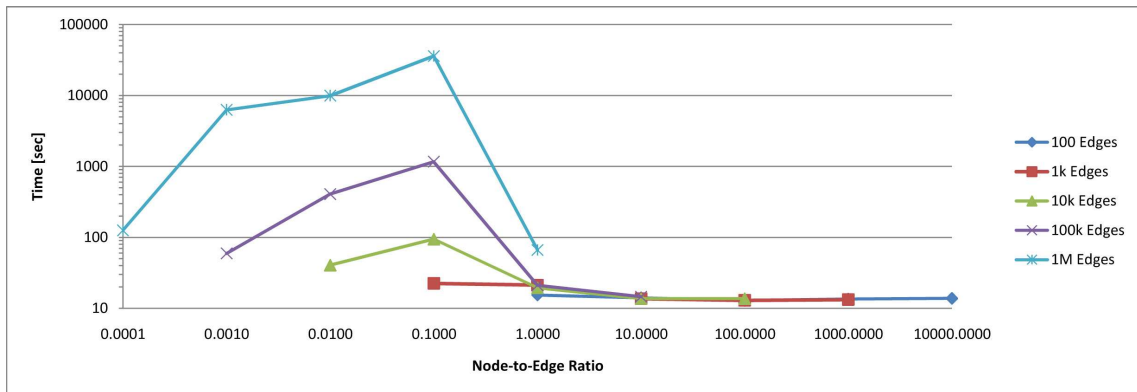


Figure 4.1: The TMatch node-to-edge ratio to time consumption graph.

For the second part the different expander classes were compared, again for different numbers of nodes and edges.

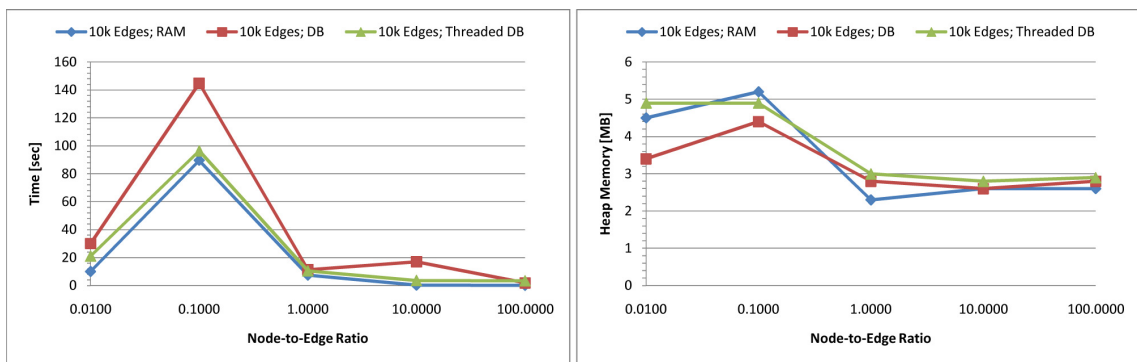


Figure 4.2: The expander node-to-edge ratio to time consumption graph with 10k edges.

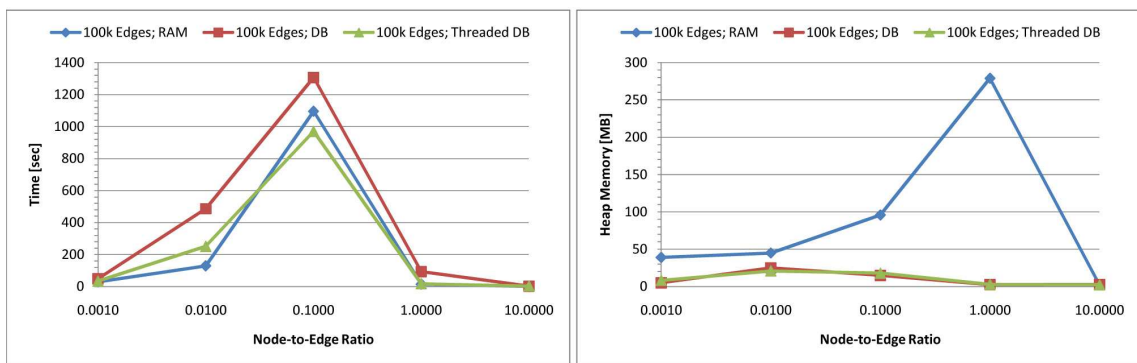


Figure 4.3: The expander node-to-edge ratio to time consumption graph with 100k edges.

The complete results of the first evaluation part are shown in Table A.4. The results of the

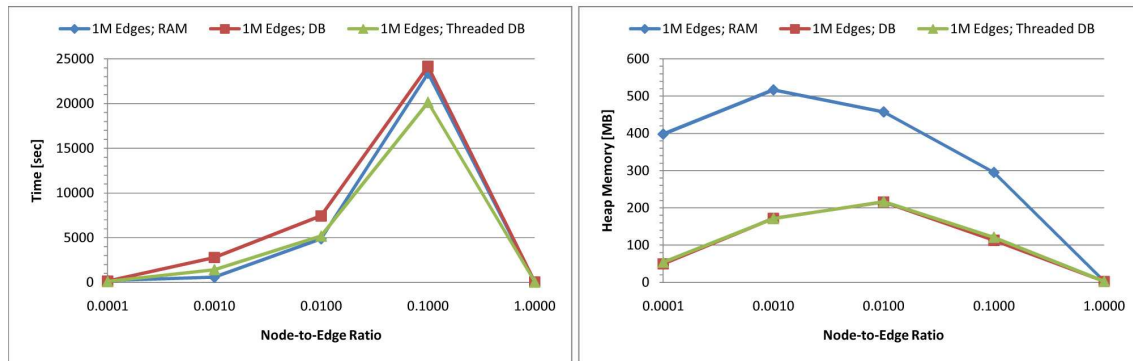


Figure 4.4: The expander node-to-edge ratio to time consumption graph with 1M edges.

second part are shown in Table A.5. Both evaluation results are available on the enclosed CD-ROM.

In Figure 4.5, the behavior of the threaded DB-based expander class is shown over time. Each thread tries to expand a particular node and has a blocked state, while waiting to send or receive data from the database abstraction layer. In addition, the main thread is shown containing the runner class as well as the "Thread-4", running a simple database performance counter (*ch.uzh-tm.match.utils.DbPerfCounter*). The impact on the time behavior of these expander threads caused by a defect network switch is shown in Figure A.10.

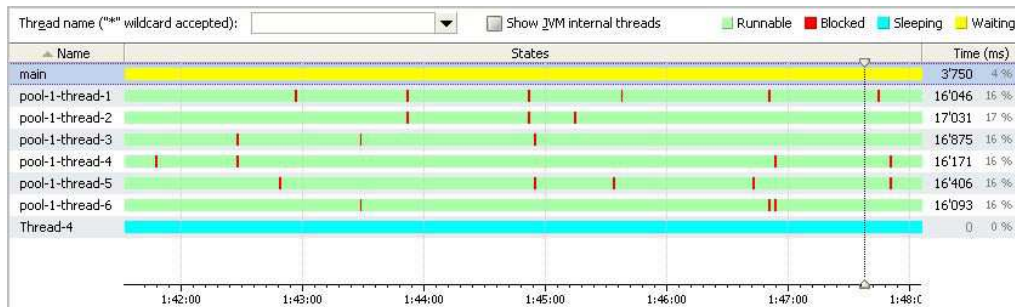


Figure 4.5: Thread view of the threaded DB-based expander.

## 4.4 Conclusion

Since there exist no real-world test datasets to evaluate and compare with, the use of synthetic transaction datasets was the only method. As mentioned before, the *TMatch* library uses a pipe-and-filter based approach. If for example, the nodeset expander does not find any new nodes, the subsequent steps are finished very quickly without significant resource and time consumption. There is a certain (low) probability that the random generated datasets will not contain any of

the searched account numbers. The same problem exists for the component finding module. If there is only one small component found, the following calculation is done very quickly without much resource consumption. In contrary, if the component finder extracts one large component composed of all existing nodes, the time and resource consumption will be potentially high. If the test dataset is regenerated with the same number of nodes and edges, there will be new components of different sizes with different resource consumption and execution time. That is why these evaluation results only provide a first impression of the time and memory requirements, according to certain numbers of nodes and edges.

In addition, the calculation time is strongly dependent on the network throughput. Table A.6 shows, that with a 10 Mbit LAN connection, the expanding step (as well as some scorer) need between 40 to 50 percent more time to finish the task as with an active 100 Mbit LAN connection.

The database server is seen as a black box system, because there is no available information about technical specifications. Nevertheless, great speed enhancements are expected by adjusting the database server setup and the table structures or by adapting the *TMatch* algorithms to the server settings. As an example, the introduction of database table indexes had a heavy impact on the processing speed. In Figure A.11, a query used by the expander class is shown and rated by a DB2 Explain Plan. The Explain Plan shows information about the way the database will process a particular SQL statement. This information is useful for analyzing and improving the performance of SQL queries. The estimated query costs were 8195. After adding table indexes, the Explain Plan changed the calculated query costs to 195, according to A.12. This points to the importance of database table indexes for *TMatch*<sup>2</sup>. Since SQL views works with data from the underlying tables, no indexes can be applied to a view itself. That's why SQL views should be avoided and replaced by either a materialized query table, which stores the query results as data in a table, or a regular SQL table.

If the calculation speed is crucial, a node-to-edge ratio between 0.01 and 0.1 should be avoided by changing the start and target set nodes or applying additional filter conditions on the alert graph. This is shown in Figure 4.1. If the ratio cannot be changed, the threaded nodeset expander shows the lowest processing time for bigger graphs, according to Figures 4.3 and 4.4 compared to the other implementations. At the same time, the threaded DB-based expander shows often the lowest memory consumption. In Figure 4.4 with a ratio of 0.01 and 1M edges, the processing time is almost equal to the memory-based expander implementation but the memory consumption is much lower. The same is true for a ratio of 1.0 with 100k edges, seen in Figure 4.3. For a ratio of 0.1 and 1M edges, processing time as well as used memory is significant lower for the threaded DB-based expander than for the other implementations. This applies for a ratio of 0.1 with 100k edges too. The memory-based expander outperforms the DB-based implementations if the ratio is smaller than 0.01. This is seen in 4.2 where the number of nodes is really low. In the case of 1M edges, its memory consumption is almost eight times as high as for the DB-based implementations.

The load of the database is elevated during processing, since most of the algorithms are im-

<sup>2</sup>The new configuration property *Global.CREATE\_DB\_INDEX* was introduced to control the creation of table indexes.

plemented using the database without storing temporary results locally. If the chosen database structure does not allow the usage of database indexes, the load is even heavier. Depending on the administrative restrictions, this behavior should be avoided. A possible approach to lower down the database load is to change the expander class to the non threaded DB-based expander or if possible, to the local memory-based expander class. This is done in the *ch.uzh.tmatch.main.Runner* class by changing the *AlgorithmLoader.createThreadedDbNodeSetExpander()* method call to *AlgorithmLoader.createDbNodeSetExpander()* or to *AlgorithmLoader.createRamNodeSetExpander()*. The scorer model class *ch.uzh.tmatch.score.PassagePathScoreModel* tries to find connected nodes with active passage flags. Therefore, this model queries all the underlying transactions between the nodes of each found component. An other approach to lower down the database load is to deactivate the passage path scorer in the *TMatchViz* expert configuration view.

In the next Chapter the results of this Master Thesis are matched to the initial goals and the limitations of *TMatch* and *TMatchViz* are mentioned.

# 5

## Discussion

Section 2.3 suggested how a possible AML solution should look like. The main goal of this Master Thesis was not to create a new product like the available commercial products in Section 2.2.4, but to evaluate and implement new solution patterns to enhance the existing AML measures at *AlphaFin*. This was done according to a graph-based data mining approach, where the accounts and transactions are seen as nodes and edges in a multi-connected weighted graph. As written in 2.2.3, graph pattern matching tries to find a target graph in a bigger graph structure. The component finder in *TMatch* calculates all the connected components out of the expanded graph data. Since there is no target graph, this procedure points to a graph mining approach with the goal to find a set of most common or interesting patterns in a graph.

As a result, *TMatch* and *TMatchViz* were developed according to the requirements of Section 2.3.1 and with direct feedback from an *AlphaFin* AML specialist. This lead us to the proposed investigation process as shown in Figure 2.4.

The extendability of *TMatch* should enable further development based on new specifications and real-world results. *TMatchViz* facilitates these intentions due to the MVC-based implementation and the loose coupling between the two software modules. New configuration properties can be defined in *TMatch* and are automatically shown in *TMatchViz*. The implementation of new algorithms or scorer models is done in *TMatch*, thus is transparent for *TMatchViz*. Because of the bundled Java runtime, further development is not influenced by any Java version constraints.

The evaluation in Chapter 4 showed that *TMatch* is capable to process certain amounts of synthetically created datasets within a reasonable time frame. The spotted limitations caused by the used algorithms and the test datasets are mentioned in the following section.

### 5.1 Limitations

As mentioned before<sup>1</sup>, the purpose of this Master Theses was not to create a generic AML product but to enhance an existing solution based on the direct input of an AML specialist. Due the lack

---

<sup>1</sup>According to the last stylistic issue in [Bernstein, 2005] on page 13.

of labeled real-world datasets, *TMatch* had to be evaluated using synthetically datasets. As stated in Section 4.4, the test datasets are only able to give a rough impression of the real-world. Even if generated with the same parameters, the test transactions always change, evolving new graph components that differs in processing time and needed amount of computational resources. That is why these results are hard to compare to real-world scenarios. The evaluation showed that a node-to-edge ratio of 0.01 to 0.1 should be avoided. This is most likely caused by the fact, that the random generated transactions often form a mesh-like structure, if the search depth is higher than five. With real-world data, such a "global" mesh will probably never occur. Expected are tentacles-like structure, based on the fact that the line-of-sight is ending at the borders of the financial institution, where cash machines are available to the customers or the international wire transfers starts. Nevertheless, premature results shows that in a real world dataset consisting of around 30k accounts, several 100k transactions and a target node set of several hundred nodes finds a double-digit number of components. This is surprising and will be matter of further investigations.

It was reported by *AlphaFin'* AML expert, that some database queries generate heavy load on the database. Since testing was done using a nonproductive database server, the database load had a lower priority in development than local memory consumption. Possible solutions were described in Section 4.4, like database index creation, the replacement of the *ThreadedDbNodeSetExpander* or the deactivation of the *PassagePathScoreModel* class in the configuration.

Concerning *TMatchViz*, the component preview framework has a performance issue if the underlying graph structure is big. That's why the component preview is deactivated if the component contains more than 200 nodes. This behavior suits the specification, since the *TV* system is used for the real graphical analysis.

# 6

## Conclusion and future work

This Master Thesis was about discovering new possibilities in the fight against money laundering. *TMatch* uses a modern graph mining approach together with filtering and scoring capabilities to enable a risk-based approach in money laundering detection. Limited data access and data handling, due to strict confidentiality regulations as well as limited computational resources had to be taken care of. The lack of labeled real-world transaction datasets and the fact that there were no exact specification required a feasible project management and autonomous working behavior.

The *TMatch* library and the *TMatchViz* user interface represents a good combination for commercial utilization and future development.

According to the evaluation, the presented approach is capable of processing thousands of accounts connected by hundreds-of-thousands of transactions within a suitable time. The processing of real-world data is currently matter of work, the final evaluation of the results will be part of the future work as well as for the modification of the *TMatch* library according to the real-world results.

The discussion in Section 5 pointed to possible future work in order to overcome the found limitations. Together with additional algorithms, these subjects are matter of the next section.

### 6.1 Future Work

High priority future work is to refine the specifications of *TMatch* and *TMatchViz* according to new result. Therefore real-world data should be processed by *TMatch*, analyzed and rated by an AML specialist. This enables the detection of operational issues which will be handled by the developers. In addition, the search profiles used in *TMatchViz* should be completed according to the results and the AML processes at *AlphaFin*.

Additional future work is to simplify the reuse of the *TMatch* results by the Transaction Visualizer. Therefore, an additional XML dataset exporter should be bundled together with *TMatch* or *TMatchViz*.

To lower down the processing time, *TMatch* should use another way to expand the target nodes and to extract the connected components from the alert graph datasets. One possibility is to implement the component finder to work on the database system instead of working on the operators workstation. This would prevent, that all the expanded graph structures has to be sent from the database to the local memory and back. In [Faloutsos et al., 2004] the relationship between two nodes is calculated in a fast manner by finding a small subgraph out of a large base graph. This can be used in *TMatch* as a new scoring model or to replace the expander and component finder parts, if there is a way to support not only two but many "target" nodes.

To analyze the component or node behavior over time a peergroup analysis based approach should be implemented, as proposed by [Weston et al., 2008]. The peergroup membership could be built on the calculated component scores like transaction amounts or number of transaction.

A low priority future task is to simplify the source building and packaging step. The *TMatch* and *TMatchViz* sources should be handles by one ANT building task. This would reduce the needed steps to create the final distribution package including the program code and the Java runtime.



# A

## Appendix

### A.1 Content CD-ROM

In Figure A.1, the data content of the enclosed master thesis CD-ROM is shown.

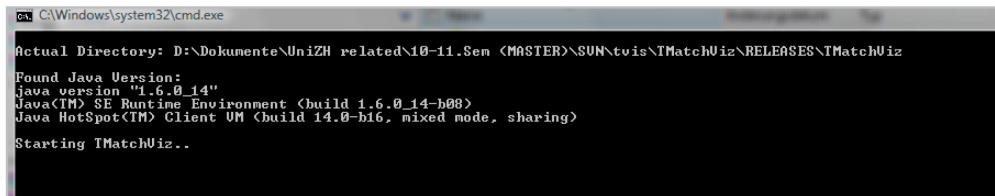
- Folder "Statistics" contains all the evaluation results.
- Folder "TMatch" as well as the archive file "TMatch.7z" contains the TMatch Eclipse project folder.
- Folder "TMatchViz" as well as the archive file "TMatchViz.7z" contains the TMatchViz NetBeans project folder.
- PDF file "Masterarbeit.pdf" contains the written part of this Master Thesis.
- PDF files "Zusfsg.pdf" and "Abstract.pdf" contains the abstract page in German and English.

Name	Typ	Größe
Statistics	Dateiordner	
TMatch	Dateiordner	
TMatchViz	Dateiordner	
Abstract.pdf	Adobe Acrobat Document	87 KB
Masterarbeit.pdf	Adobe Acrobat Document	4'241 KB
TMatch.7z	7Z-Datei	71'809 KB
TMatchViz.7z	7Z-Datei	69'109 KB
Zusfsg.pdf	Adobe Acrobat Document	76 KB

**Figure A.1:** The data content of the master thesis CD-ROM.

## A.2 TMatchViz graphical user interface

Additional *TMatchViz* screens are shown in Figure A.2, A.3, A.4, A.5 and A.6.



```
C:\Windows\system32\cmd.exe

Actual Directory: D:\Dokumente\UnizH related\10-11.Sem <MASTER>\SUN\tvis\TMatchViz\RELEASES\TMatchViz

Found Java Version:
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode, sharing)

Starting TMatchViz..
```

Figure A.2: The TMatchViz Starter.bat output.

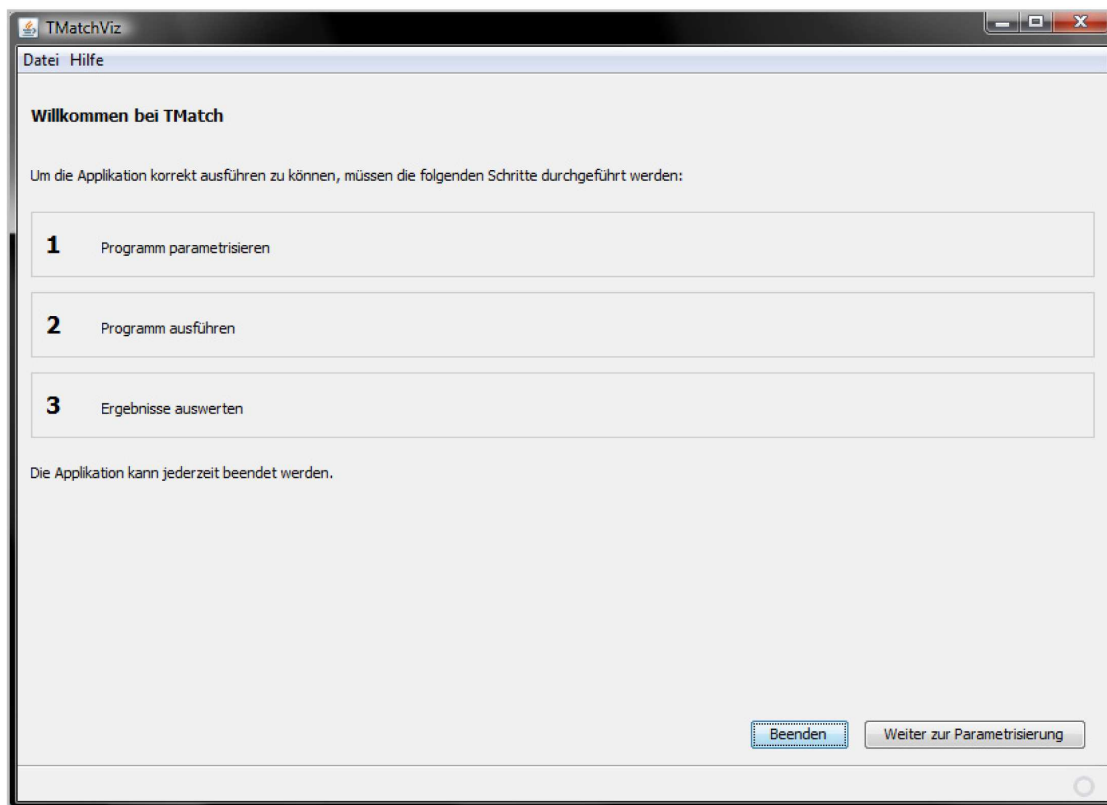
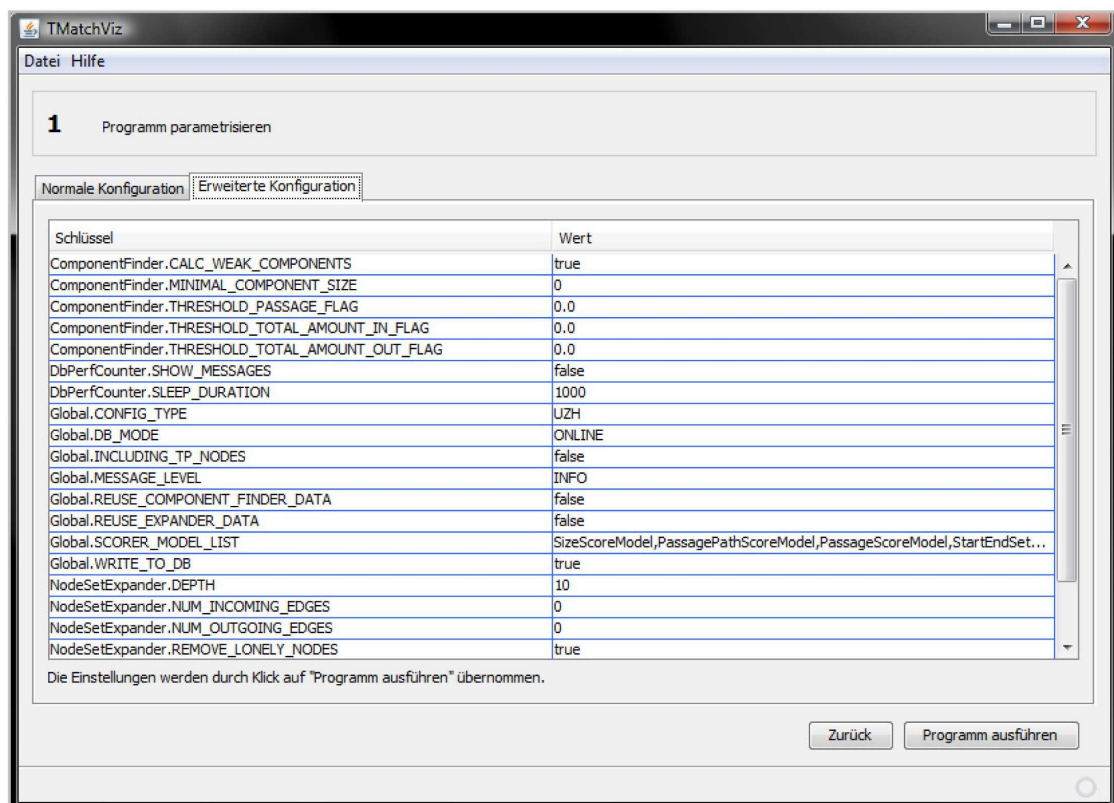


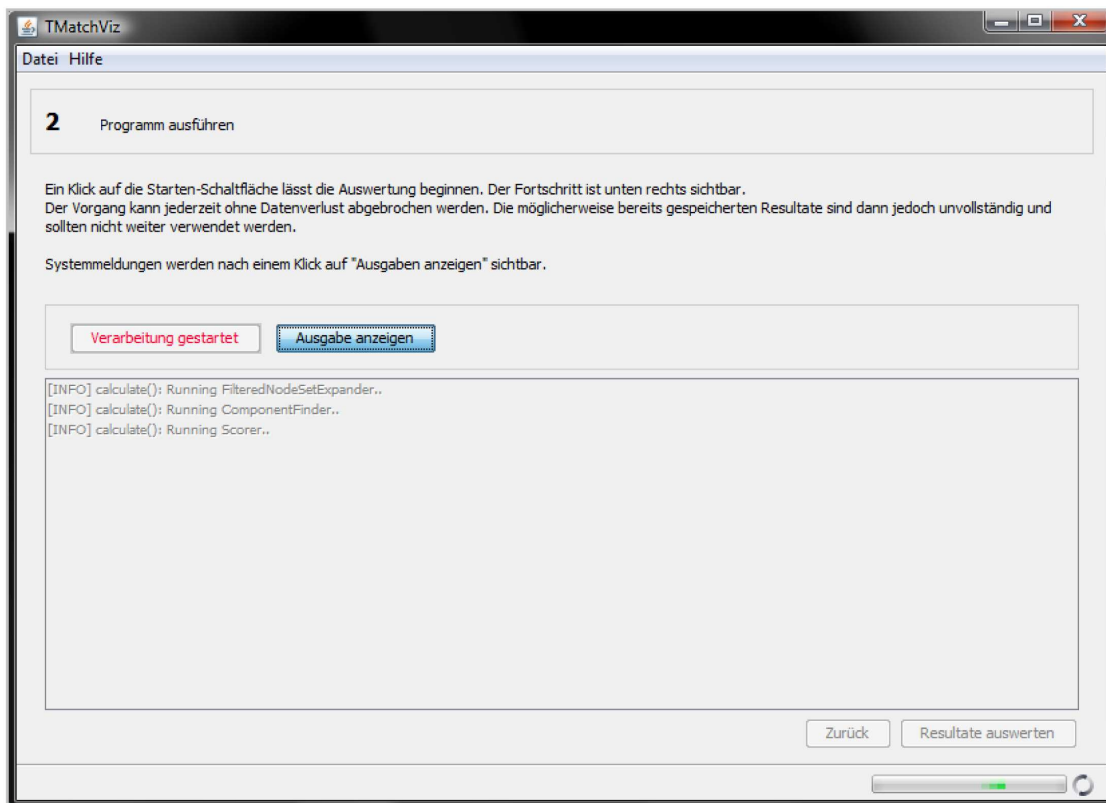
Figure A.3: The TMatchViz welcome screen.

## A.3 TMatch extendability

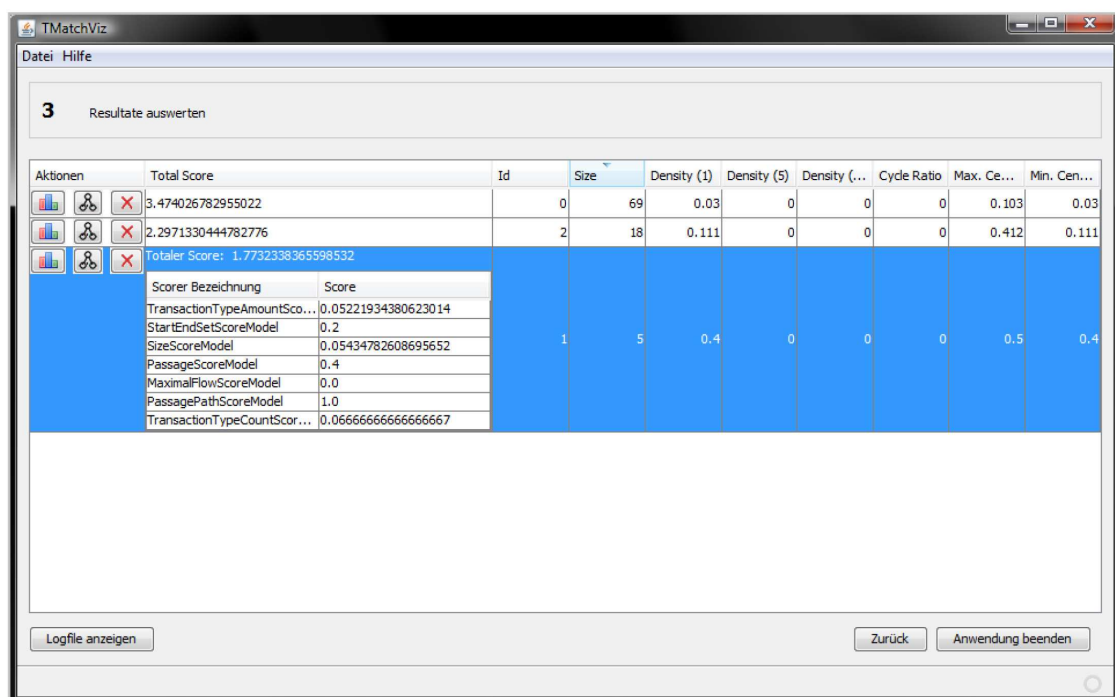
This section describes how to extend *TMatch* by adding new algorithms or scorer models.



**Figure A.4:** The TMatchViz expert configuration screen. Changes are automatically saved in the user profile for later reuse.



**Figure A.5:** The TMatchViz running dialog with start-button and additional process informations.



**Figure A.6:** Same TMatchViz result screen as Figure 3.7, after rearranging the table columns and sorted by the component size.

Basic entities are:

- The class *ch.uzh.tmatch.main.Runner* for the main workflow.
- The class *ch.uzh.tmatch.algorithm.utils.AlgorithmLoader* used as an algorithm factory class.
- The abstract class *ch.uzh.tmatch.algorithm.utils.AbstractAlgorithm* used for all algorithms.
- The abstract class *ch.uzh.tmatch.score.AbstractScoreModel* used for all scorer models.

To implement a new algorithm, the class *ch.uzh.tmatch.algorithm.ShortestPath* can be used as a template. Each algorithm must define its own *algorithmId* and a *free()* method for memory house-keeping. The algorithm is loaded by the algorithm factory class and is added with a new creator method. Finally, the creator method must be invoked to create a new instance of the new algorithm, most likely by the Runner class or in any other part of *TMatch*.

A new scorer model is added by extending the abstract scorer model class. Again, each model has to define its own *algorithmId* and a method according to the signature *double calculate(int componentId)*. The model constructor method can be used to send additional variables to the model. Finally, the Runner class must be extended to invoke this new model. Optionally, the global configuration setting *Global.SCORER\_MODEL\_LIST* may be used to activate or deactivate the scorer model. Therefore, the model id must be added to the default configuration */ch/uzh/tmatch/configuration/TMatch.default.properties* or appended during runtime to the *Global.SCORER\_MODEL\_LIST* configuration parameter in the *TMatchViz* expert configuration view.

## A.4 TMatch library creation

For the creation of the *TMatch* JAR library, an ant task used. Listing A.1 on page 53 shows the used source code.

## A.5 TMatchViz release creation

The following steps are necessary to create a new *TMatchViz* release.

- Build the *TMatchViz* project in NetBeans IDE. The JAR output folder will be `<PROJECT-DIR>\TMatchViz\dist\`.
- Assure to have 7-zip installed and accessible through `C:\Program Files\7-Zip\7z.exe`, or change the *CreateRunnable.bat* in the next step accordingly.
- In folder `<PROJECT-DIR>\TMatchViz\starter\` edit the batch file *CreateRunnable.bat* according to your needs. The files in this folder are seen in Figure A.7. The archive *java-bin.7z* contains an already packed Java JRE 1.6. As already mentioned, *Starter.bat* is the start batch file.

---

```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <project default="create_run.jar" name="Create Runnable Jar for Project TMatch">
3   <target name="create_run.jar">
4     <jar destfile="D:/SVN/tvis/TMatch/dist/TMatch.jar" filesetmanifest="mergewithoutmain">
5       <manifest>
6         <attribute name="Built-By" value="\${user.name}"/>
7         <attribute name="Main-Class" value="ch.uzh.tmatch.main.Main"/>
8         <attribute name="Class-Path" value="."/>
9       </manifest>
10      <fileset dir="D:/SVN/tvis/TMatch/bin"/>
11      <fileset dir="D:/SVN/tvis/TMatch/src"/>
12      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/db2jcc.jar"/>
13      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/db2jcc4.jar"/>
14      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/ibatis-2.3.4.726._
15        jar"/>
16      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/jgrapht-jdk1.5.jar._
17        "/>
18      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/jgraph.jar"/>
19      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/jgraphlayout.jar"/>
20      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/prefuse.jar"/>
21      <zipfileset excludes="META-INF/*.SF" src="D:/UniZH/eclipse/plugins/org.junit4.4.3.1/_
22        junit.jar"/>
23      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/y.jar"/>
24      <zipfileset excludes="META-INF/*.SF" src="D:/SVN/tvis/TMatch/lib/log4j-1.2.15.jar"/>
25      <fileset excludes="*.xls, *.SJProfile0.ser, release.txt, user.properties, TODO.txt, _
26        log4j.log, .settings/**,src/**,bin/**,dist/**,lib-src/**,THESIS/**,PerfTest/**,_
27        starter/**" dir="D:/SVN/tvis/TMatch"/>
28    </jar>
29  </target>
30 </project>

```

---

Listing A.1: *TMatch* library creation ant script. The used file paths must be adjusted before using it.

Name	Änderungsdatum	Typ	Größe
.svn	31.07.2009 14:19	Dateiordner	
CreateRunnable.bat	16.07.2009 01:55	Windows-Stapelv...	2 KB
exclude	16.07.2009 00:10	Datei	1 KB
java-bin.7z	16.07.2009 00:10	7Z-Datei	18'016 KB
Starter.bat	16.07.2009 00:10	Windows-Stapelv...	1 KB

Figure A.7: The TMatchViz release builder directory.

Name	Typ	Größe
java-bin	Dateiordner	
lib	Dateiordner	
EndNodes.txt	TXT-Datei	1 KB
log4j.properties	PROPERTIES-Datei	2 KB
README.txt	TXT-Datei	5 KB
SQLMapConfig.properties.ALPHAFIN.template	TEMPLATE-Datei	1 KB
SQLMapConfig.properties.UZH.template	TEMPLATE-Datei	1 KB
Starter.bat	Windows-Stapelverarbei...	1 KB
StartNodes.txt	TXT-Datei	1 KB
TMatchViz.jar	Executable Jar File	246 KB
TransactionTypes.txt	TXT-Datei	1 KB

Figure A.8: The TMatchViz directory content after installation.

- Run *CreateRunnable.bat*. The executable *TMatchViz.exe* will be created. This is now the newest release of *TMatchViz*.

## A.6 TMatchViz installation

To install *TMatchViz*, the file created in Section A.5 needs to be executed. It will extract the entire system to a new subfolder called *TMatchViz*. The folder will contain around 113 MB data. Figure A.8 shows the content of the *TMatchViz* directory. In order to get started, the following steps are required.

- Configure iBATIS by renaming the appropriate template file `SQLMapConfig.properties.*.template` to `SQLMapConfig.properties` and moving it to the library folder.
- The files *StartNodes.txt*, *EndNodes.txt* and *TransactionTypes.txt* should be edited now or during the configuration step, while running *TMatchViz*.



Name	Änderungsdatum	Typ	Größe
appframework-1.0.3.jar	16.07.2009 12:44	Executable Jar File	258 KB
exclude	25.06.2009 15:58	Datei	1 KB
swing-worker-1.1.jar	16.07.2009 12:44	Executable Jar File	11 KB
TMatch.jar	16.07.2009 12:44	Executable Jar File	22'529 KB

**Figure A.9:** The TMatchViz library directory after installation.

*TMatchViz* is now ready to be started by executing *Starter.bat*.

## A.7 Properties

There are several locations, where properties files are searched for:

- <TMatchViz-Directory>/\*
- TMatchViz.jar:tmatchviz/\*
- TMatchViz.jar:tmatchviz/resources/profiles/\*

There are several properties files stored:

- The TMatch default properties file is located in  
TMatch.jar:ch/uzh/tmatch/configuration/TMatch.default.properties.
- The TMatchViz profile properties files are located in  
TMatchViz.jar:/tmatchviz/resources/profiles/Profile.\*.properties. To change the existing profiles delete or modify the specific file.
- The "user.properties" file is located in  
<TMatchViz-Directory>/user.properties. To reset the user configuration values, the "user.properties" file must be deleted.

A *TMatchViz* profile properties file includes some additional entries, as seen in A.1. New profiles are stored in the *TMatchViz* main directory or, if a new release is created, in the profile folder of the JAR file.

Table A.2 shows all the *TMatch* default values, Table A.3 the valid input ranges.

## A.8 Logging

The *ch.uzh.tmatch.utils.Logger* class is a simple logger facade. It provides several methods and differs between several logging priority levels. In addition, Log4j is also connected to this Logger

Property Key	Summary
Search.Profile.Image	This string value is the filename to an image file in the <code>TMatchViz.jar:/tmatchviz/resources/profiles</code> directory.
Search.Profile.Name	This string value is the displayed profile name.
Search.Profile.Hint	This string value is the profile description text.

**Table A.1:** The Scorer configuration key.

class. Since iBatis uses Log4j too, the `log4j.properties` file is used to activate the different logging channels. As default, the `ch.uzh.tmatch` is set to ALL. The logging output is direct to the active console window as well as to a log file called `log4j.log`. The `log4j.properties` configuration is valid for *TMatch* and *TMatchViz*. To activate additional iBatis logging output, the prepared properties must be uncommented in the `log4j.properties` file.

The following logging setup can be made.

- Configuration property: `Global.MESSAGE_LEVEL`  
The accepted values and theirs precedence are: `ERROR > INFO > DEBUG > TRACE`
- `Log4j.properties` file.  
The *TMatch* logging output is controlled by `Global.MESSAGE_LEVEL`, thus should not be changed.  
The iBatis logging channels are available by `log4j.logger.com.ibatis.*` and `log4j.logger.java.sql.*`. Possible values are OFF, WARN, DEBUG.

## A.9 Evaluation results

To simplify the evaluation charts, the number of nodes and edges was shifted by the factor of ten to obtain a logarithmic scale. Table 4.1 shows all the results from the *TMatch* library evaluation with the different component timings. Table 4.2, 4.3 and 4.4 shows the characteristics of the different expander implementations.

In Figure A.10 the blocked expander threads, caused by a slow network connection, are shown.



**Figure A.10:** Thread view of the threaded DB-based expander influenced by a 10Mbit LAN and permanent packet collisions.

Property Key	Default Value
GL.MESSAGE_LEVEL	INFO
GL.CONFIG_TYPE	UZH
GL.DB_MODE	ONLINE
GL.WRITE_TO_DB	TRUE
GL.INCLUDING_TP_NODES	FALSE
GL.REUSE_EXPANDER_DATA	FALSE
GL.REUSE_COMPONENT_FINDER_DATA	FALSE
GL.SCORER_MODEL_LIST	SizeScoreModel, PassagePathScoreModel, PassageScoreModel, StartEndSetScoreModel, TransactionTypeCountScoreModel, TransactionTypeAmountScoreModel, MaximalFlowScoreModel
GL.CREATE_DB_INDEX	TRUE
NSE.DEPTH	10
NSE.NUM_INCOMING_EDGES	0
NSE.NUM_OUTGOING_EDGES	0
NSE.TOTALAMOUNT_IN	0.0
NSE.TOTALAMOUNT_OUT	0.0
NSE.THRESHOLD_PASSAGE_FLAG	0.0
NSE.REMOVE_LONELY_NODES	TRUE
NSE.DB.REUSE_EXPANDED_DATA	FALSE
CF.THRESHOLD_TOTAL_AMOUNT_IN_FLAG	0.0
CF.THRESHOLD_TOTAL_AMOUNT_OUT_FLAG	0.0
CF.THRESHOLD_PASSAGE_FLAG	0.0
CF.MINIMAL_COMPONENT_SIZE	0
CF.CALC_WEAK_COMPONENTS	TRUE
DPC.SLEEP_DURATION	1000
DPC.SHOW_MESSAGES	FALSE

GL stands for the namespace *Global*; NSE for *NodeSetExpander*;  
CF for *ComponentFinder*; DPC for *DbPerfCounter*

**Table A.2:** TMatch default configuration values.

Property Key	Input Range
GL.MESSAGE_LEVEL	INFO/DEBUG/TRACE
GL.CONFIG_TYPE	UZH/ <i>AlphaFin</i>
GL.DB_MODE	ONLINE
GL.WRITE_TO_DB	Boolean value
GL.INCLUDING_TP_NODES	Boolean value
GL.REUSE_EXPANDER_DATA	Boolean value
GL.REUSE_COMPONENT_FINDER_DATA	Boolean value
GL.SCORER_MODEL_LIST	Comma-separated string
GL.CREATE_DB_INDEX	Boolean value
NSE.DEPTH	Integer value
NSE.NUM_INCOMING_EDGES	Integer value
NSE.NUM_OUTGOING_EDGES	Integer value
NSE.TOTALAMOUNT_IN	Double value
NSE.TOTALAMOUNT_OUT	Double value
NSE.THRESHOLD_PASSAGE_FLAG	Double value: 0.0 to 1.0
NSE.REMOVE_LONELY_NODES	Boolean value
NSE.DB_REUSE_EXPANDED_DATA	Boolean value
CF.THRESHOLD_TOTAL_AMOUNT_IN_FLAG	Double value
CF.THRESHOLD_TOTAL_AMOUNT_OUT_FLAG	Double value
CF.THRESHOLD_PASSAGE_FLAG	Double value: 0.0 to 1.0
CF.MINIMAL_COMPONENT_SIZE	Integer value
CF.CALC_WEAK_COMPONENTS	Boolean value
DPC.SLEEP_DURATION	Integer value (msec.)
DPC.SHOW_MESSAGES	Boolean value

GL stands for the namespace *Global*; NSE for *NodeSetExpander*;  
 CF for *ComponentFinder*; DPC for *DbPerfCounter*

**Table A.3:** TMatch configuration values range.

#Nodes	#Edges	Ratio	Mem	Runner	Expander	CF	Scorer	Overhead
100	100	1.0000	2.7	15.4	4.4	0.2	0.7	10.1
100	1000	0.1000	3.8	22.5	11.2	0.4	0.9	9.9
100	10000	0.0100	7	40.6	25.2	1.4	3.9	10.1
100	100000	0.0010	8.2	59.4	39.4	2.6	6.7	10.7
100	1000000	0.0001	56	125.7	94.5	2.6	10.4	18.2
1000	100	10.000	2.2	14.0	3.1	0.2	0.8	9.9
1000	1000	1.0000	2.9	21.2	9.5	0.3	1.5	10.0
1000	10000	0.1000	13	94.4	77.2	1.5	5.3	10.3
1000	100000	0.0100	56	407.2	251.6	35.7	108.8	11.1
1000	1000000	0.0010	343	6266.5	1667.3	2064.0	2515.3	19.9
10000	100	100.00	2.1	12.9	2.4	0.1	0.4	10.0
10000	1000	10.000	2.9	13.6	2.7	0.2	0.7	10.0
10000	10000	1.0000	3.1	19.4	7.7	0.3	1.3	10.2
10000	100000	0.1000	64	1165.6	939.4	13.3	201.7	11.2
10000	1000000	0.0100	605	9920.0	5967.4	412.5	3521.7	18.3
100000	100	1000.0	3.2	13.5	2.7	0.2	0.6	10.0
100000	1000	100.00	2.7	13.0	2.4	0.1	0.4	10.0
100000	10000	10.000	2.3	13.7	2.7	0.2	0.6	10.2
100000	100000	1.0000	3.1	21.1	8.1	0.3	1.4	11.3
100000	1000000	0.1000	389	46629.0	39793.2	64.4	6754.3	17.1
1000000	100	10000	3	13.8	2.5	0.1	0.4	10.8
1000000	1000	1000.0	2.7	13.3	2.4	0.1	0.4	10.3
1000000	10000	100.00	2.6	13.7	2.4	0.1	0.4	10.7
1000000	100000	10.000	3	14.6	2.6	0.2	0.6	11.2
1000000	1000000	1.0000	2.6	66.3	40.5	0.4	1.4	23.9

*#Nodes* or *#Edges* stands for the number of nodes or edges.

*Ratio* stands for the node-to-edge ratio.

*Mem* stands for the maximal used heap memory in megabytes [MB].

*Runner* stands for the time consumed by the Runner component in seconds [sec].

*Expander* stands for the time consumed by the Expander component in seconds [sec].

*CF* stands for the time consumed by the Component Calculation component in seconds [sec].

*Scorer* stands for the time consumed by the Scorer component in seconds [sec].

**Table A.4:** Evaluation results of the TMatch library.

#Nodes	#Edges	Ratio	Max. Heap Memory [MB]			Time Consumption [sec]		
			RAM	DB	Threaded DB	RAM	DB	Threaded DB
100	100	1.0000	2.9	2.9	2.6	5.10	2.6	7.7
100	1000	0.1000	2.7	7.6	2.6	16.93	2.6	16.2
100	10000	0.0100	4.5	10.0	3.4	29.98	4.9	21.1
100	100000	0.0010	39	29.3	5.1	48.91	8.4	34.5
100	1000000	0.0001	398	208.1	50	139.58	54	107.2
1000	100	10.000	2.1	0.6	2.6	2.90	2.4	4.3
1000	1000	1.0000	2.6	9.0	2.5	14.40	3	12.5
1000	10000	0.1000	5.2	89.6	4.4	144.72	4.9	96.4
1000	100000	0.0100	45	129.9	25	486.35	21	251.3
1000	1000000	0.0010	517	576.7	172	2779.58	171	1428.4
10000	1000	10.000	2.7	0.4	2.4	2.10	2.6	3.6
10000	10000	1.0000	2.3	7.4	2.8	11.34	3	10.4
10000	100000	0.1000	45	1095.5	15	1307.71	18	970.5
10000	1000000	0.0100	458	4876.2	216	7437.83	217	5191.7
100000	10000	10.000	2.6	0.3	2.6	17.06	2.8	3.7
100000	100000	1.0000	279	15.2	2.7	92.86	3.1	17.7
100000	1000000	0.1000	295	23381.6	113	24176.90	121	20156.2
1000000	1000	1000.0	2.1	0.2	2.6	1.84	2.3	3.4
1000000	10000	100.00	2.6	0.2	2.8	1.87	2.9	3.4
1000000	100000	10.000	2.3	0.3	2.8	2.19	2.4	3.7
1000000	1000000	1.0000	2.7	38.7	2.7	42.56	3.1	41.3

#Nodes or #Edges stands for the number of nodes or edges and Ratio for the node-to-edge ratio. RAM, DB and Threaded DB addresses the nodeset expander implementation (see Section 3.4.4).

**Table A.5:** Evaluation results of the three different nodeset expanders.

LAN speed	#Nodes	#Edges	Expander
10 Mbit	100	100000	61.5
10 Mbit	100	1000000	167.9
10 Mbit	1000	100000	454.1
100 Mbit	100	100000	39.4
100 Mbit	100	1000000	94.5
100 Mbit	1000	100000	251.6

*#Nodes* or *#Edges* is the number of nodes or edges.

*Expander* stands for the expander time consumption [sec].

**Table A.6:** Evaluation results with 10 or 100 Mbit LAN speed.

## A.10 Database related

Figure A.11 shows the DB2 Explain Plan rating without having access to database indexes. After creating new indexes, the Explain Plan rating changed according to Figure A.12.



**Figure A.11:** DB2 Explain Plan query costs calculation without table indexes.

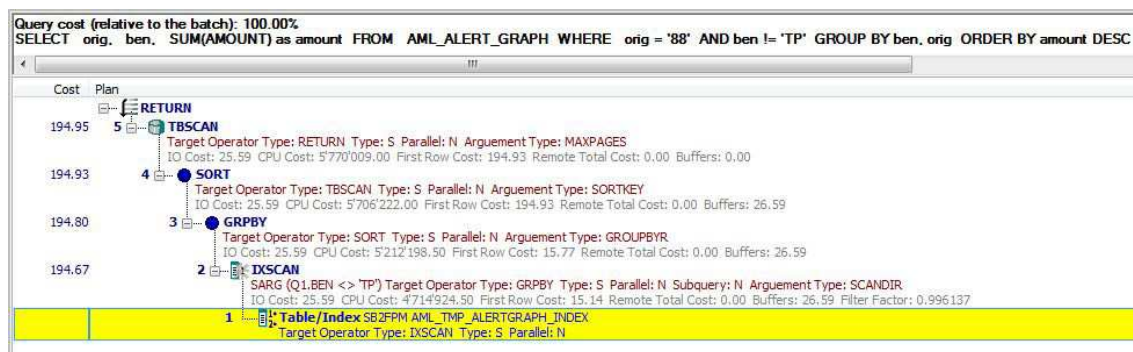


Figure A.12: DB2 Explain Plan query costs calculation with table indexes.



---

# List of Figures

2.1	The flow of money and informations in a wire transaction according to [Chang et al., 2008]. . . . .	8
2.2	Core technologies proposal according to [OTA, 1995]. . . . .	9
2.3	User interface to modify an behavior profile in Fiserv' Dynamic Risk Scoring Module. . . . .	16
2.4	The proposed investigation process. . . . .	17
3.1	Architectural overview. . . . .	20
3.2	Overview data transformation. . . . .	20
3.3	Overview TMatch. . . . .	23
3.4	TMatch workflow: transaction in the alert graph table; expander, component finder and scorer output. . . . .	24
3.5	Overview TMatchViz. . . . .	32
3.6	The TMatchViz configuration screen. . . . .	33
3.7	The TMatchViz result screen, showing found components and the intermediate scorer results. . . . .	34
3.8	The TMatchViz result screen with activated component preview. . . . .	35
4.1	The TMatch node-to-edge ratio to time consumption graph. . . . .	39
4.2	The expander node-to-edge ratio to time consumption graph with 10k edges. . . . .	39
4.3	The expander node-to-edge ratio to time consumption graph with 100k edges. . . . .	39
4.4	The expander node-to-edge ratio to time consumption graph with 1M edges. . . . .	40
4.5	Thread view of the threaded DB-based expander. . . . .	40
A.1	The data content of the master thesis CD-ROM. . . . .	47
A.2	The TMatchViz Starter.bat output. . . . .	48
A.3	The TMatchViz welcome screen. . . . .	48
A.4	The TMatchViz expert configuration screen. Changes are automatically saved in the user profile for later reuse. . . . .	49
A.5	The TMatchViz running dialog with start-button and additional process informations. . . . .	50

A.6	Same TMatchViz result screen as Figure 3.7, after rearranging the table columns and sorted by the component size. . . . .	51
A.7	The TMatchViz release builder directory. . . . .	54
A.8	The TMatchViz directory content after installation. . . . .	54
A.9	The TMatchViz library directory after installation. . . . .	55
A.10	Thread view of the threaded DB-based expander influenced by a 10Mbit LAN and permanent packet collisions. . . . .	56
A.11	DB2 Explain Plan query costs calculation without table indexes. . . . .	61
A.12	DB2 Explain Plan query costs calculation with table indexes. . . . .	62

---

# List of Tables

3.1	The Expander configuration keys . . . . .	25
3.2	The calculated component metrics and flags. . . . .	27
3.3	The Component Finder configuration keys . . . . .	28
3.4	The Scorer configuration key. . . . .	29
A.1	The Scorer configuration key. . . . .	56
A.2	TMatch default configuration values. . . . .	57
A.3	TMatch configuration values range. . . . .	58
A.4	Evaluation results of the TMatch library. . . . .	59
A.5	Evaluation results of the three different nodeset expanders. . . . .	60
A.6	Evaluation results with 10 or 100 Mbit LAN speed. . . . .	61



---

# List of Listings

3.1	<i>TMatch</i> expander pseudo code. . . . .	26
4.1	A sample test transaction entry from a 1M node dataset. . . . .	38
4.2	The start node, end node and transaction type file contents. . . . .	38
A.1	<i>TMatch</i> library creation ant script. The used file paths must be adjusted before using it. . . . .	53



---

# Bibliography

- [Altenkirch, 2006] Altenkirch, L. (2006). *Techniken der Geldwäsche und ihre Bekämpfung*. Bankakademie-Verlag GmbH, Frankfurt am Main.
- [Bernstein, 2005] Bernstein, A. (2005). So what is a (Diploma) Thesis? A few thoughts for first-timers. Technical report, Dynamic and Distributed Information Systems Group, University of Zurich, Switzerland.
- [Black and Lovrencic, 2004] Black, P. E. and Lovrencic, A. (2004). Subgraph. Technical report, U.S. National Institute of Standards and Technology. Available from <http://www.itl.nist.gov/div897/sqg/dads/HTML/subgraph.html>. Accessed on 14th August 2009.
- [Blessing, 1998] Blessing, J. (1998). Applying flow analysis methods to the problem of network design. In *System Theory, 1998. Proceedings of the Thirtieth Southeastern Symposium on*, pages 424–428.
- [Bolton and Hand, 2002] Bolton, R. and Hand, D. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, 17(3):235–255.
- [Bolton et al., 2001] Bolton, R. J., Hand, D. J., and H, D. J. (2001). Unsupervised profiling methods for fraud detection. In *Proc. Credit Scoring and Credit Control VII*, pages 5–7.
- [Chang et al., 2008] Chang, R., Lee, A., Ghoniem, M., Kosara, R., Ribarsky, W., Yang, J., Suma, E., Ziemkiewicz, C., Kern, D., and Sudjianto, A. (2008). Scalable and interactive visual analysis of financial wire transactions for fraud detection. *Information Visualization*, 7(1):63–76.
- [Cheng et al., 2008] Cheng, J., Yu, J. X., Ding, B., Yu, P. S., and Wang, H. (2008). Fast graph pattern matching. *Data Engineering, International Conference on*, 0:913–922.
- [Eberle and Holder, 2007] Eberle, W. and Holder, L. (2007). Discovering Structural Anomalies in Graph-Based Data. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 393–398.
- [Encyclopædia Britannica, 2009] Encyclopædia Britannica (2009). Fraud. From Encyclopædia Britannica Online: <http://www.britannica.com/EBchecked/topic/217591/fraud>. Accessed on 26th July 2009.

- [Ezawa and Norton, 1995] Ezawa, K. J. and Norton, S. W. (1995). Constructing bayesian networks to predict uncollectible telecommunications accounts. *IEEE Intelligent Systems*, 11(5):45–51.
- [Faloutsos et al., 2004] Faloutsos, C., Mccurley, K. S., and Tomkins, A. (2004). Fast discovery of connection subgraphs. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 118–127, New York, NY, USA. ACM Press.
- [FATF, 2009] FATF (2009). Financial action task force annual report. Technical report, Financial Action Task Force. Available from <http://www.fatf-gafi.org>. Accessed on 26th July 2009.
- [Fawcett et al., 1997] Fawcett, T., Foster, and Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1:291–316.
- [Foggia et al., 2001] Foggia, P., Sansone, C., and Vento, M. (2001). A performance comparison of five algorithms for graph isomorphism. In *In Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, pages 188–199.
- [Gallagher, 2006] Gallagher, B. (2006). Matching structure and semantics: A survey on graph-based pattern matching. In *In AAAI FS '06: Papers from the 2006 AAAI Fall Symposium on Capturing and Using Patterns for Evidence Detection*, pages 45–53.
- [Geiger and Wuensch, 2007] Geiger, H. and Wuensch, O. (2007). The fight against money laundering. *Journal of Money Laundering Control*, 10(1).
- [Han and Kamber, 2000] Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1st edition.
- [Häfliger, 2009] Häfliger, M. (2009). Schweizer Justiz jagt Kohle-Barone. *NZZ am Sonntag*, 29th March 2009.
- [Huang et al., 2009] Huang, M. L., Liang, J., and Nguyen, Q. V. (2009). A Visualization Approach for Frauds Detection in Financial Market. In *Information Visualisation, 2009 13th International Conference*, pages 197–202.
- [INL, 2009] INL (2009). Major Money Laundering Countries. Technical report, U.S. Bureau of International Narcotics and Law Enforcement Affairs. Available from <http://www.state.gov/p/inl/rls/nrcrpt/2009/vol2/116550.htm>. Accessed on 26th July 2009.
- [Katkov, 2006a] Katkov, N. (2006a). Evaluating the Vendors of Anti-Money Laundering Solutions 2006. Technical report, Celent. Available from [http://www.celent.com/67\\_69.htm](http://www.celent.com/67_69.htm). Accessed on 14th August 2009.
- [Katkov, 2006b] Katkov, N. (2006b). Evaluating the Vendors of Anti-Money Laundering Solutions 2006, NetEconomy Profile. Technical report, Celent. Available from <http://www.aml.fiserv.com/leadingAMLvendor.aspx>. Accessed on 14th August 2009.



- [Kingdon, 2004] Kingdon, J. (2004). Ai fights money laundering. *Intelligent Systems, IEEE*, 19(3):87–89.
- [Lovrencic, 2004] Lovrencic, A. (2004). Maximally connected component. Technical report, U.S. National Institute of Standards and Technology. Available from <http://www.itl.nist.gov/div897/sqg/dads/HTML/maximallyConnectedComponent.html>. Accessed on 14th August 2009.
- [Luell and Bernstein, 2009] Luell, J. and Bernstein, A. (2009). Detecting Internal Fraud under Real World Conditions. Unpublished Paper. Version of 26 July 2009.
- [Maes et al., 1993] Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (1993). Credit card fraud detection using bayesian and neural networks. In *In: Maciunas RJ, editor. Interactive image-guided neurosurgery. American Association Neurological Surgeons*, pages 261–270.
- [MROS, 1999] MROS (1999). 2th Annual Report. Technical report, Money Laundering Reporting Office Switzerland. Available from <http://www.ejpd.admin.ch>. Accessed on 26th July 2009.
- [MROS, 2005] MROS (2005). 8th Annual Report. Technical report, Money Laundering Reporting Office Switzerland. Available from <http://www.ejpd.admin.ch>. Accessed on 26th July 2009.
- [MROS, 2009] MROS (2009). 11th annual report. Technical report, Money Laundering Reporting Office Switzerland. Available from <http://www.ejpd.admin.ch>. Accessed on 26th July 2009.
- [OTA, 1995] OTA (1995). Information Technologies for the Control of Money Laundering. Technical report, U.S. Congress, Office of Technology Assessment (OTA). OTA-ITC-630 (Washington, DC: U.S. Governemnt Printing Office).
- [Phua et al., 2005] Phua, C., Lee, V., Smith, K., and Gayler, R. (2005). A comprehensive survey of data mining-based fraud detection research. *Artificial Intelligence Review*.
- [Ribarsky and Dill, 2008] Ribarsky, B. and Dill, J. (2008). Visual analytics science and technology. *Information Visualization*, 7(1):1–2.
- [Roth et al., 2004] Roth, J., Greenburg, D., and Wille, S. (2004). 9/11 Commission Report. Technical report, The National Commission on Terrorist Attacks. Available from <http://www.911commission.gov>. Accessed on 26th July 2009.
- [S/CT, 2001] S/CT (2001). Executive Order 13224. Technical report, Office of the Coordinator for Counterterrorism. Available from <http://www.state.gov/s/ct/rls/other/des/122570.htm>. Accessed on 26th July 2009.
- [Ullmann, 1976] Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42.
- [Wells, 2008] Wells, J. (2008). *Principles of Fraud Examination*. Wiley, New York.

- [Weston et al., 2008] Weston, D., Hand, D., Adams, N., Whitrow, C., and Juszczak, P. (2008). Plastic card fraud detection using peer group analysis. *Advances in Data Analysis and Classification*, 2(1):45–62.
- [Yan and Han, 2002] Yan, X. and Han, J. (2002). gspan: Graph-based substructure pattern mining. *Data Mining, IEEE International Conference on*, 0:721.
- [Yue et al., 2007] Yue, D., Wu, X., Wang, Y., Li, Y., and Chu, C.-H. (2007). A Review of Data Mining-Based Financial Fraud Detection Research. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 5519–5522.
- [Ziegler et al., 2007] Ziegler, H., Nietzschmann, T., and Keim, D. (2007). Visual Exploration and Discovery of Atypical Behavior in Financial Time Series Data using Two-Dimensional Colormaps. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 308–315.