ADVCOMP 2022

The Sixteenth International Conference on Advanced Engineering Computing and Applications in Sciences

ISBN: 978-1-61208-990-4

November 13 - 17, 2022

Valencia, Spain

ADVCOMP 2022 Editors

Laura Garcia, Universitat Politecnica de Valencia, Spain

ADVCOMP 2022

Forward

The Sixteenth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2022), held between November 13 and November 17, 2022, continued a series of evens meant to bring together researchers from the academia and practitioners from the industry in order to address fundamentals of advanced scientific computing and specific mechanisms and algorithms for particular sciences.

With the advent of high performance computing environments, virtualization, distributed and parallel computing, as well as the increasing memory, storage and computational power, processing particularly complex scientific applications and voluminous data is more affordable. With the current computing software, hardware and distributed platforms effective use of advanced computing techniques is more achievable.

The conference provided a forum where researchers were able to present recent research results and new research problems and directions related to them. The conference sought contributions presenting novel research in all aspects of new scientific methods for computing and hybrid methods for computing optimization, as well as advanced algorithms and computational procedures, software and hardware solutions dealing with specific domains of science.

We take here the opportunity to warmly thank all the members of the ADVCOMP 2022 technical program committee, as well as all the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to ADVCOMP 2022. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We also gratefully thank the members of the ADVCOMP 2022 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that ADVCOMP 2022 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the field of engineering computing and applications in sciences. We hope that Valencia provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

ADVCOMP 2022 Chairs

ADVCOMP 2022 Steering Committee

Dean Vucinic, Vrije Universiteit Brussel (VUB), Belgium | FERIT, Croatia
Juha Röning, University of Oulu, Finland
Hans-Joachim Bungartz, TUM, Germany
Andreas Rausch, Technische Universität Clausthal, Germany
Marcin Hojny, AGH University of Science and Technology, Poland
Alice E. Koniges, University of Hawai'i at Mānoa, USA
Alfred Geiger, T-Systems Information Services GmbH, Germany

ADVCOMP 2022 Publicity Chair

Mar Parra, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

ADVCOMP 2022

COMMITTEE

ADVCOMP Steering Committee

Dean Vucinic, Vrije Universiteit Brussel (VUB), Belgium | FERIT, Croatia,
Alfred Geiger, T-Systems Information Services GmbH, Germany
Hans-Joachim Bungartz, TUM, Germany
Andreas , Technische Universität Clausthal, Germany
Juha Röning, University of Oulu, Finland
Marcin Hojny, AGH University of Science and Technology, Poland
Alice E. Koniges, University of Hawai'i at Mānoa, USA

ADVCOMP 2022 Publicity Chair

Mar Parra, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

ADVCOMP 2022 Technical Program Committee

Waleed H. Abdulla, University of Auckland, New Zealand
José Abellán, Catholic University of Murcia, Spain
Mohamed Riduan Abid, Alakhawayn University, Morocco
Rashmi Agrawal, Manav Rachna International Institute of Research and Studies, India
Francisco Airton Silva, Federal University of Piauí, Brazil
M. Azeem Akbar, Nanjing University of Aeronautics and Astronautics, China
Haifa Alharthi, Saudi Electronic University, Saudi Arabia
Sónia Maria Almeida da Luz, Polytechnic Institute of Leiria - School of Technology and Management, Portugal
Madyan Alsenwi, Kyung Hee University, Global Campus, South Korea
Mohamed E. Aly, California State Polytechnic University, Pomona, USA
Daniel Andresen, Kansas State University, USA
Anindya Das Antar, University of Michigan, USA
Alberto Antonietti, Politecnico di Milano / University of Pavia, Italy
Mansur Arief, Carnegie Mellon University, Pittsburgh, USA
Ehsan Atoofian, Lakehead University, Canada
Vadim Azhmyakov, Universidad Central, Bogota, Republic of Colombia
Carlos Becker Westphall, University of Santa Catarina, Brazil
Raoudha Ben Djemaa, ISITCOM | University of Sousse, Tunisia
Peter Bentley, University College London, UK
Sergiy Bogomolov, Newcastle University, UK
Alessandro Borri, CNR-IASI Biomathematics Laboratory, Rome, Italy
David Bouck-Standen, Kingsbridge Research Center, UK
Sofiane Bououden, University Abbes Laghrour Khenchela, Algeria
Hans-Joachim Bungartz, TUM, Germany
Xiao-Chuan Cai, University of Colorado Boulder, USA

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Data and Feature Engineering Challenges in Machine Learning

Kendall E. Nygard
Department of Computer Science
North Dakota State University
Fargo, ND, USA
Email: kendall.nygard@ndsu.edu

Mostofa Ahsan, Aakanksha Rastogi, Rashmi Satyal
Department of Computer Science
North Dakota State University
Fargo, ND, USA
Email: {mostofa.ahsan, aakanksha.rastogi, rashmi.satyal}@ndsu.edu

Abstract— **The process of developing of a machine learning application presents multiple challenges relating to data and their features. Based on experiences with several applied studies carried out using machine learning methodologies, we report on addressing challenges in the collection, quantity, distribution, quality, sampling, of and relevancy of data. We also address feature engineering and selection issues, including approaches to identifying, combining, and eliminating attributes and features that are not needed or of low significance. We include insight into overfitting and underfitting training data. Example applications include classification, anti-autonomy and trust modeling and analytics for self-driving cars and intrusion detection systems aimed at detecting malicious activity.**

Keywords- Machine Learning, Data Management, Feature Engineering, Self-Driving Cars, Intrusion Detection.

I. INTRODUCTION

Machine Learning (ML) is a rapidly emerging area of artificial intelligence. Many types of applications have been successfully developed and new successes are regularly reported. The famous Turing award winner Jim Gray referred to data science as a "fourth paradigm," taking a rightful place among empirical, theoretical, and computational sciences [1].

Often viewed as interdisciplinary, data science involves mathematics, statistics and computer science as well as other related areas. In many applications, the availability of large and relevant datasets, and the methods of data science provide the lifeblood of machine learning problem-solving approaches. Analyses and decision support in nearly every area of human endeavor today are related to machine learning.

The example machine learning studies that we describe are in the areas of self-driving cars and intrusion detection [2], [33], [36], [37].

In the case of the self-driving car study, there was availability of multi-attribute data about specific collisions. The data contained a host of features and attributes concerning the vehicle itself, the damage incurred, roadway conditions, etc. The objective of the study was to build a classification model that could translate the detailed data into collision predictions and to drive an anti-autonomy trust model. There were important and difficult choices made related to scale and balance within the available dataset, and in feature engineering. A linear sequential supervised learning machine learning model was employed.

The intrusion detection study used supervised learning techniques to build a model for identifying outside threats initiated by malicious actors who wish to breach or compromise a system. Among other datasets, the study examined the famous dataset that originated in the KDD (Knowledge Discovery and Data Mining) competition and was later modified to form the now publicly available NSL (Network Security Laboratory) KDD dataset [6], [7].

The rest of the paper is structured as follows. In Section II, we describe supervised machine learning with illustrations of the flow of a machine learning model and data splits for cross validation. In Section III, we present a self-driving cars example illustrating an implementation of a linear sequential supervised learning artificial neural network model utilizing multiple pre-processed complex attributes. In Section IV, we present the intrusion detection example, explaining how a machine learning model can be tuned to predict and identify attacks. In Section V, we describe data management and feature engineering issues that are ubiquitous in machine learning practice. This section also includes several categorical encoding techniques for preprocessing data for a machine learning algorithm. Finally, we conclude our work in Section VI.

II. SUPERVISED MACHINE LEARNING

We restrict our discussion to supervised learning models. Fundamentally, such models are well-suited to address applications for which there are available datasets whose data instances have a known classification label or target. The initial task is to computationally train the machine learning model to accept the data instances as input and to produce the correct target as output. Once trained, the model is available to accept new data and predict their target classification. The model is successful if it has high values of performance measures such as percentage of accuracy in correctly classifying the new data instances, called the ability to generalize. There are multiple issues surrounding the characteristics of the available data, the classes into which they fall, their attributes and features, and the learning models charged with producing the predictions. Concerning baseball, Coach Yogi Berra famously said, "It's tough to make

 ISBN: 978-1-61208-990-4

predictions, especially about the future." This aphorism is equally true in machine learning [45].

Figure 1 illustrates the general flow of a machine learning technique. Several tasks are included. The overall task of the DEVELOP phase shown at the top is to produce a Final Model that is fully specified, trained and feeds into the PREDICT phase shown below the dotted line, where it is available for generalization use on new data. Starting from the top, the data is shown as partitioned into splits for Training, Validation and Test. The full data is divided into the Training Split and the Test Split. A good way to perform training is to withhold a portion of the data while training is done. It is viewed as a mistake to train and test a machine learning model on the same data. So doing would result in the model memorizing all of the data/target pairs, resulting in the model perfectly knowing all of the answers, leaving no ability to generalize. The result is known as *overfitting*. For validation purposes, the Training Split is typically divided into pieces called folds. Called k-fold cross validation, Figure 2 illustrates basic logic for splitting the data. In this example, k=5 so there are five equal parts. This corresponds to the Validation Split and Model Tuning blocks in Figure 1. In Figure 2, shown in bold italics on the diagonal, there is a designated fold in each row that is specified for testing, with the other four used for training. The key idea is to find the best set of meta-level parameters for a model being developed. All of the major machine learning models have parameters. For example, an ML that utilizes an Artificial Neural Network (ANN) in some way, will be parameterized with settings like Learning Rate (governs weight adjustments), topology (number of hidden layers, nodes within layers, and interconnectedness), and activation functions. Other ML methods, like a Support Vector Machine (SVM) or Logistic Regression also have parameters. When a model is trained on the folds, a performance metric, such as classification accuracy, can be calculated on the testing fold. After all of the fold splits are evaluated in this way, an average is calculated, which yields a score for the parameter settings. Various optimization methods can be employed to explore the parameter space in a quest to identify the best settings. Viewed more generally, the Model Tuning block can also be viewed as exploring various types of models in a quest to not only optimize the use of one type of model, but to also choose among competing models.

In multiple places of the ML process, there is a need to evaluate the quality of the predictions using a metric. The empirical accuracy of a method is simply the percentage of the predictions made that are correct. Other metrics are available. More details are provided later in this paper.

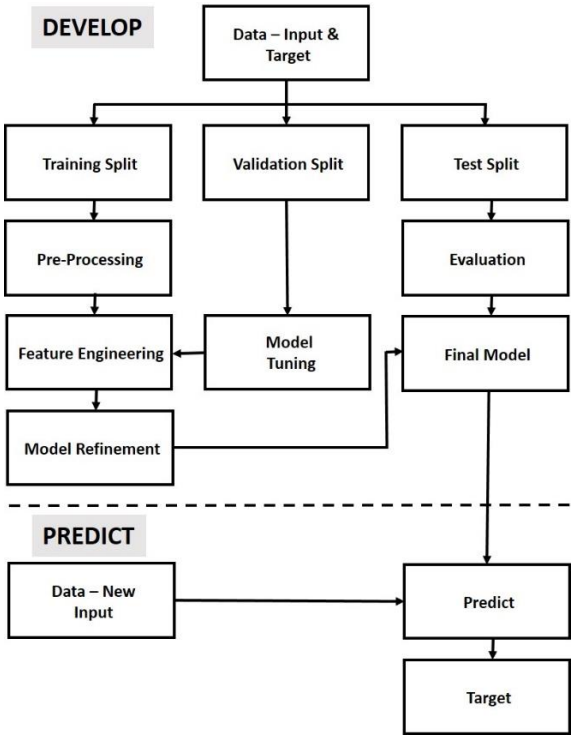


Figure 1. Flow of a Machine Learning Model [2]

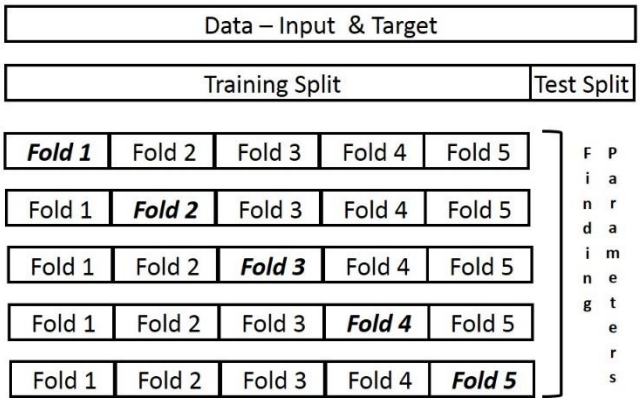


Figure 2. Data Splits for Cross Validation

Raw data is rarely available in a form that is suitable for direct use by an ML model. Pre-processing of data is typically necessary to deal with null or missing values, outliers, transforming or reconciling numeric and categorical values, rescaling, standardizing, etc. We expand on the data-centric issues for the example applications reported in this article.

Feature Engineering also appears in Figure 1. Features are those characteristics that are present in the data that are potentially useful in predicting a target outcome. It is often effective in ML to modify or combine features in some way to produce a new feature that can improve the prediction accuracy of the method. Called Feature Engineering, the operations that can be carried include things like mathematically transforming a single feature or applying a

 ISBN: 978-1-61208-990-4

functional calculation on multiple features. Feature Selection refers to reducing the number of features employed by the model while retaining acceptable results. Reducing the features needed can ease the data collection task and reduce the computational load of running the model. Feature Selection typically follows Feature Engineering. We provide details related to the examples discussed in this paper. Unsupervised Learning is different than supervised learning in many ways. Some of the most known algorithms are, k-means clustering, hierarchical clustering, principal component analysis, and apriori algorithm [44]. The need for unsupervised models is increasing in the cybersecurity domain since attacks are being modified every day [46].

III. SELF-DRIVING CARS EXAMPLE

For an application to self-driving cars, we used available data to study anti-autonomy traits and factors responsible for collisions and diminished trust [2], [38]. Data availability was a challenge since jurisdictions of different states, federal traffic agencies and motor vehicle departments often do not make their data publicly available. Data used in this study was submitted by the manufacturers of autonomous vehicles to the California Department of Motor Vehicles for collisions that occurred with other cars, pedestrians, bicyclists, and other objects during their test drives on roads and freeways in the state. All data applied to collisions that occurred while the cars were being driven in autonomous driving mode. The collisions occurred between October 2014 and March 2020 [2], [38]. The attributes of this dataset are listed in Table I below. All attribute names are feature type categorical and data type object.

TABLE I. COLLISION DATA ATTRIBUTES [2]

Attribute Type	Attribute Names
Autonomous vehicle details	Manufacturer Name, Business Name, Vehicle Year, Vehicle Make, Vehicle Model, Vehicle was (stopped in traffic/moving)
Accident Details	Date of Accident, Time of Accident
Involved in Autonomous vehicle accident	Involved in Autonomous Vehicle Accident (Pedestrian/Bicyclist/Other), Number of vehicles involved with Autonomous Vehicle
Autonomous vehicle damage	Vehicle Damage, Damaged Area
Details of other vehicle involved in accident	Vehicle 2 Year, Vehicle 2 Make, Vehicle 2 Model, Vehicle 2 was (stopped in traffic/moving)
Involved in Other vehicle accident	Involved in Vehicle 2 Accident Pedestrian, Involved in Vehicle 2 Accident Bicyclist, Involved in Vehicle 2 Accident Other, Number of vehicles involved with Vehicle 2
Injuries	Injured, Injured Driver, Injured Passenger, Injured Bicyclist
Vehicle driving mode	Vehicle Driving Mode
Weather conditions for both vehicles	Clear, Cloudy, Raining, Snowing, Fog/Visibility, Other, Wind
Lighting conditions for both vehicles	Daylight, Dusk-Dawn, Dark Street Lights, Dark-No Street Lights, Dark-Street Lights Not Functioning
Roadway surface for both vehicles	Dry, Wet, Snowy-Icy, Slippery/Muddy/Oily/etc., Holes-Deep-Rut, Loose Material on Roadway, Obstruction on Roadway, Construction/Repair Zone, Reduced Roadway Width, Flooded, Other, No Unusual Conditions
Preceding Movement of Autonomous Vehicle before collision	Stopped, Proceeding Straight, Ran Off Road, Making Right Turn, Making Left Turn, Making U Turn, Backing, Slowing/Stopping, Passing Other Vehicle, Changing Lanes, Parking Maneuver, Entering Traffic, Unsafe Turning, Xing Into Opposing Lane, Parked, Merging, Travelling Wrong Way, Other
Preceding Movement of Other Vehicle before collision	Stopped, Proceeding Straight, Ran Off Road, Making Right Turn, Making Left Turn, Making U Turn, Backing, Slowing/Stopping, Passing Other Vehicle, Changing Lanes, Parking Maneuver, Entering Traffic, Unsafe Turning, Xing Into Opposing Lane, Parked, Merging, Travelling Wrong Way, Other
Type of Collision	Head On, Side Swipe, Rear End, Broadside, Hit Object, Overturned, Vehicle/Pedestrian, Other
Other	CVC Sections Violated Cited, Vision Obscurement, Inattention, Stop and Go Traffic, Entering/Leaving Ramp, Previous Collision, Unfamiliar With Road, Defective WEH Equip Cited, Uninvolved Vehicle, Other, None Apparent, Runaway Vehicle

This data was extracted from PDF files and converted into CSV format. Data cleaning was a significant effort, and included pre-processing steps for augmenting, labeling, and classifying the data [2], [38]. The core purpose of the study was to associate conditions into a level of trust that people had in a self-driving car. Anti-autonomy refers to decisions and actions taken by a self-driving car that are in some way inappropriate in terms of increasing risk, diminishing safety, or lowering trust. The values of attributes in the data that describe conditions and circumstances that are present when a collision occurs provide a handle to model a mapping between data and trust level. After pre-processing the data, a linear sequential supervised learning ANN model called NoTrust was devised, validated, and tested to classify the target data, using the basic approach illustrated in Figure 1.

The model used the libraries provided by Keras with the Tensorflow backend [39] -[41]. Python was used for programming since it integrates with Keras to access the

neural network Application Programming Interface. The deep learning libraries of Keras support fast prototyping, modularity, and smooth computation.

There are multiple challenges concerning data, features, and metrics in applying a ML methodology to the self-driving car application. First, there were only 256 collision reports available, which is arguably a small number to use in a ML method. Also, in the context of alternative target value possibilities, the data is somewhat unbalanced in that the number of samples across the distinct classes differs. Section V describes methods, such as oversampling, to deal with unbalanced data. Second, there are 140 attributes, a large number relative to sample size, as shown in Table I. Thus, the possible permutations and combinations that could be evaluated in the ANN model is explosive. Fortunately, with initial analyses of the data and evaluation runs, it was possible to identify a subset of attributes and features which revealed that they were mandatory to include. The five attributes shown below form the mandatory set.

- Vehicle driving mode = autonomous
- Vehicle damage = moderate and major
- First vehicle involved = Pedestrians/Bicycle/Other
- Second vehicle involved = Bicycle/Other
- Injuries sustained = Pedestrians/Bicyclists/Others

While keeping the model simple and still retaining accuracy, the mandatory feature set performed well in making trust and do not trust predictions for autonomous vehicles. However, when anti-autonomous traits of the self-driving car itself were incorporated into the model it became apparent that more attributes had to be utilized. These include vehicle driving mode, type of collision, weather conditions, roadway surface conditions and injuries sustained by pedestrian/bicyclists/others. In addition to the linear sequential ANN, evaluation of Recurrent Neural Networks (RNN) models with Long Short-Term Memory (LSTM) were available for possible comparison purposes. When the additional attributes are included, along with measures of severity of damages sustained by vehicle, the imbalance of the data increased. More specifically, the larger number of predictors added more noise, redundancies, increased overfitting, and decreased the quality of the predictions. A related study by Meiri and Zahavi [3] used simulated annealing to search the attribute space.

Combinatorial problems often have issues related to model accuracy, performance, and optimizer bias. Also, the model solutions offered by machine learning include approximation errors which is further exacerbated by the fact that the input configurations to the ML model can be significantly different between training and validation [4]. This can be solved by two approaches – active learning and passive learning. Active learning involves updating the model itself to assure a convergence between training and validation curves in turn improving model accuracy and optimization bias. Passive learning involves the training set providing a uniform and sufficient coverage of the search space [4]. In a similar context, Charikar et al. [5] defined and studied combinatorial feature selection problems and presented a theoretical framework which provided algorithms on approximation and hardness results of these combinatorial problems [5].

IV. INTRUSION DETECTION EXAMPLE

In today's world of connected devices, security of the network is of critical importance. Unauthorized access and malicious activities are a great threat to confidentially, integrity, and availability that form the information security triad. The role of an Intrusion Detection System (IDS) is to detect abnormalities caused by an unauthorized reach into the network and send alerts. An IDS is an element of support for a wall of defense between cyber-attacks. Supervised ML techniques in an IDS can provide high detection accuracy, particularly against known types of attacks.

The NSL-KDD is an update and improvement to the KDD'99 dataset that that was developed for the KDD Cup competition in 1999 [6]. These datasets are publicly available and are very widely used for IDS experiments. The data is primarily internet traffic consisting of 43 features per record, of which the last two are *class* (attack or normal) and *score* (severity of traffic input) [7]. The *class* column provides information on whether the record is considered normal or is a member of one of four attack classes - Denial of Service (DoS), Probe, Remote-to-Local (R2L) or User-to-Root (U2R). There are14 attack types under these 4 classes: Apache2, Smurf, Neptune, Back, Teardrop, Pod, Land, Mailbomb, Processtable, UDPstorm, WarezClient, Guess_Password, WarezMaster, Imap, Ftp_write, Named, Multihop, Phf, Spy, Sendmail, SmpGetAttack, AnmpGuess, Worm, Xsnoop, Xlock, Buffer_Overflow, Httptuned, Rootkit, LoadModule, Perl, Xterm, Ps, SQLattack, Satan, Saint, Ipsweep, Portsweep, Nmap, Mscan [42], [43]. A mixture of categorical (nominal), binary and numeric variables are in the feature set. Each record has basic, content-related, time-related, and host-based features [8]. The attributes of this dataset are listed in Table II.

TABLE II. NSL-KDD DATASET ATTRIBUTES [8]

Attribute Type	Attribute Names
Basic	Duration, Protocol_type, Service, Flag, Src_bytes, Dst_bytes, Land, Wrong_fragment, Urgent
Content related	Hot, Num_failed_logins, Logged_in, Num_compromised, Root_shell, Su_attempted, Num_root, Num_file_creations, Num_shells, Num_access_files, Num_outbound_cmds, Is_hot_login, Is_guest_login
Time related	Count, Srv_count, Serror_rate, Srv_serror_rate, Rerror_rate, Srv_rerror_rate, Same_srv_rate, Diff_srv_rate, Srv_diff_host_rate
Host based traffic	Dst_host_count, Dst_host_srv_count, Dst_host_same_srv_rate, Dst_host_diff_srv_rate, Dst_host_same_src_port_rate, Dst_host_srv_diff_host_rate, Dst_host_serror_rate,

	Dst_host_srv_serror_rate, Dst_host_rerror_rate, Dst_host_srv_rerror_rate

The study also used the UNSW-NB15 dataset. This dataset has 49 features categorized into 6 groups: basic, flow, time, content, labelled and additional generated features [9]. There are 9 attack types: fuzzers, analysis, back-doors, DoS, exploits, generic, reconnaissance, shell code and worms [10]. This dataset has a mixture of categorical, binary, and numerical datatypes. The attributes of this dataset are listed in Table III below.

TABLE III. UNSW-NB15 DATASET ATTRIBUTES [15]

Attribute Type	Attribute Names
Basic	state, dur, sbytes, dbytes, sttl, dttl, sloss, dloss, service, sload, dload, spkts, dpkts
Flow	srcip, sport, dstip, dsport, proto
Content	swin, dwin, stcpb, dtcpb, smeansz, dmeansz, trans_depth, res_bdy_len
Time	sjit, djit, stime, ltime, sintpkt, dintpkt, tcprtt, synack, ackdat
Additional generated (general purpose)	is_sm_ips_ports, ct_state_ttl, ct_flw_http_mthd, is_ftp_login, ct_ftp_cmd
Additional generated (connection)	ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm
Labelled	attack_cat, attack_cat

The target attribute either identifies records as ‘normal’ or ‘attack’ or distinguishes the record as a particular attack type. Depending on the desired goal of an intrusion detection system, the machine learning model is tuned to identify a particular attack, which is a challenge in itself. It is thus essential to understand the requirement thoroughly and preprocess input data accordingly.

As an illustration of evaluation metrics, at a high level in the IDS study, for each input vector we have exactly one of the following outcomes:

TP = True Positive = Correct predication that the input vector is an Attack

TN = True Negative = Correct prediction that the input vector is not an Attack

FN = False Negative = Incorrect prediction the input vector is not an Attack

FP = False Positive = Incorrect prediction the input vector is an Attack

The most widely reported metric is Basic Accuracy of the model, which simply reports the proportion of attack reports that are correct.

$$\text{Accuracy} = (TP + TN)/(TP = TN = FP + FN)$$

The Basic Accuracy is notoriously deceptive when the classes are unbalanced, as in the case of intrusion detection studies, where most input vectors are not attacks. False reports are of interest. This gives rise to the need for metrics such as Precision and Recall, which can be calculated from information in the confusion matrix given below.

	Prediction		
Actual		Attack	Not an Attack
	Attack	TP	FN
	Not an Attack	FP	TN

$$\text{Precision} = TP/(TP + FP)$$

$$\text{Recall} = TP/(TP + FN)$$

Precision measures the proportion of the vectors reported by the IDS as attacks that are real attacks. Recall measures the proportion of the vectors that are real attacks and do get reported as such. This means that when Recall is high the IDS does not misclassify many true attacks.

In Intrusion Detection applications, false negatives can be very deadly, which favors high Recall. However, dealing with false positives also has a cost. Unfortunately, experiments that improve Precision typically reduce Recall. The reverse is also often true. For this reason, the harmonic mean of the two, called the F1 score is often calculated.

$$F1 = (2*(\text{Precision} * \text{Recall}))/(\text{Precision} + \text{Recall}).$$

The effect of the F1 score, which falls between 0 and 1, is to punish extreme values.

V. METHODOLOGIES FOR ADDRESSING DATA AND FEATURE ENGINEERING ISSUES

We provide detailed descriptions data management and feature engineering issues that are pervasive in ML practice and were of importance in our applied studies.

A. Data Management

Class imbalance in a dataset means that the relative numbers of instances within the classes vary significantly in number [16]. The magnitude of the discrepancies will also vary. Class imbalance is common in most important data domains, including detection of things like fraudulent activities, anomalies, oil spills, and in medical diagnoses. The imbalance of classes occurs in both binary class and multi-class classification [17]. In binary classes, the smaller and larger cardinality classes are called minority and majority classes respectively [16], [18]. Class imbalance can influence the training process of ML techniques and lead to ignoring the minority class entirely. We discuss some of the approaches to treat class imbalances. Figures 3 – 9 illustrate the results of applying each technique.

Random oversampling of a minority class. In this approach data instances in minority classes are duplicated at random to induce a balance of membership between classes. Due to randomness of the oversampling, the method is naïve in that

it makes no assumptions about the classes and their members [19], [20]. Since exact copies of some data instances are included in training, there is a risk of overfitting. Classifier accuracy may also be influenced, and computational effort may be increased.

Random undersampling of a majority class. This approach discards data instances from majority classes to induce balancing [21]. As in the case of the oversampling method, the discarded data is chosen randomly and naively. The method applies to both binary and multiclass data. The approach can make it difficult to distinguish boundaries between classes, with an inimical impact on performance measures [22].

Synthetic Minority Oversampling Technique (SMOTE). This technique was introduced in 2002 to address the shortcomings of the oversampling and undersampling approaches [23], [24]. The technique generates synthetic data by calculating feature space similarities between minority class data instances. The K-nearest neighbors of each data instance in a minority class are calculated, then randomly selected one by one. The method then calculates linear interpolations among the data and uses them to create synthetic data instances.

Borderline SMOTE. The SMOTE approach encounters issues when minority class data instances occur in the vicinity of majority class data instances, creating undesirable bridges. The Borderline SMOTE variation addresses this drawback by oversampling only minority class instances near the borderline. Data points are called border points if they are incident to both minority and majority classes and called noise points otherwise [25]. Border points are then utilized to balance the data between classes.

K-Means SMOTE. This technique generates minority class samples in safe and crucial borders of input spaces and thus assists performance in classification. The method begins by clustering the dataset using the K-means procedure, then selects the clusters that have higher numbers of minority samples [26]. Additional synthetic samples are then assigned to clusters where minority class samples are sparsely distributed. No noise points are generated.

SVM SMOTE. A variation of Borderline-SMOTE, the method finds misclassification points. The borderline points are approximated and classified with a Support Vector Machine (SVM) classifier [27]. Synthetic data points are created randomly around the lines joining each minority class support vector with its neighbors.

Adaptive Synthetic Sampling – ADASYN. A limitation of Borderline SMOTE is that is utilizes only synthetic points generated from extreme observations and the borderline instances and neglects the rest of the points in minority classes. This issue is addressed by ADASYN by creating the synthetic data using the density of the existing data [28]. The ratio of synthetically generated data is created in inverse relation to the minority class density. In this way, a less dense area creates more synthetic data.

The Churn Modeling Data from Kaggle was applied to the methods [29]. Figure 3 shows the distribution of the data in the original classes, followed by the outcomes of the alternative methods in Figures 4 to 9.

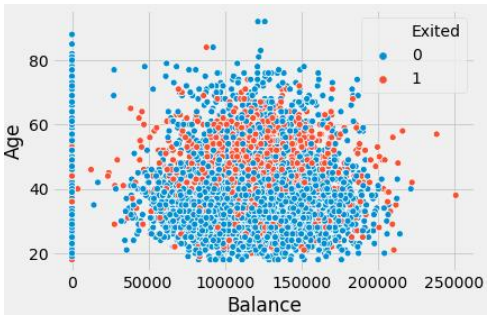


Figure 3. Original Class Imbalance Illustration

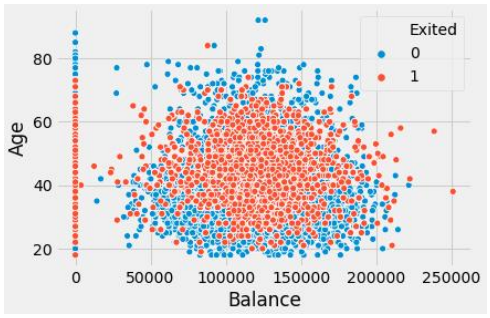


Figure 4. Outcome of Random Oversampling

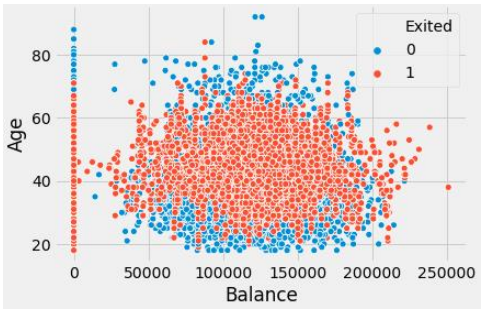


Figure 5. Outcome of SMOTE

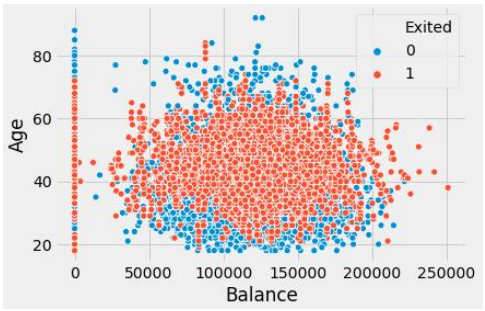


Figure 6. Outcome of Borderline-SMOTE

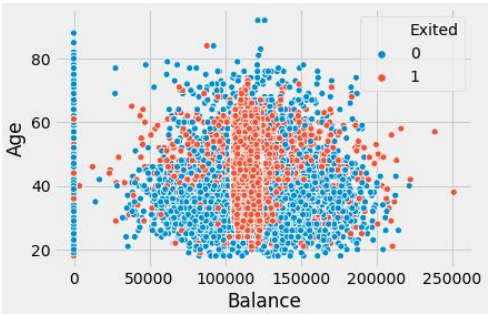


Figure 7. Outcome of K-Means SMOTE.

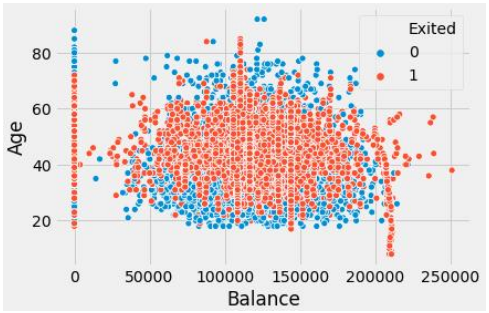


Figure 8. Outcome of SVM SMOTE

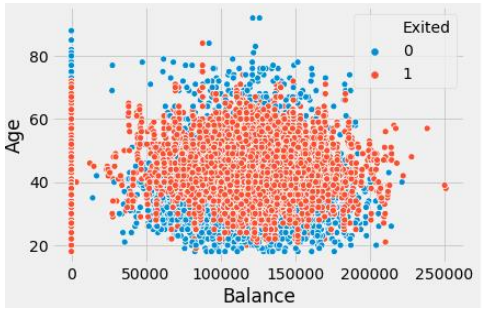


Figure 9. Outcome of ADASYN

B. *Data Management and Feature Engineering*

There are multiple methods for feature engineering on categorical data. The inputs to ML algorithms must be numeric, but many applications have categorical data. In our work with ML methods for self-driving car collisions there are examples of ordinal categorical data, such as rating of severity of damages or a weather condition. The intrusion detection work examples include counts of file access attempts, session duration, or error rates. There are also examples of nominal data, such as a type of vehicle in our self-driving car work, or whether or not a flag is set in the intrusion detection work. Various encoding methods are used to convert the variables into a useful numerical representation [9]. Choosing an appropriate encoding scheme is an essential part of data preprocessing for a ML algorithm. Some of the categorical encoding techniques are described below.

a) One-hot encoding

This method converts an attribute with N possible categories into N distinct features. In the NSL-KDD dataset, the *protocol type* attribute has 3 possible values - Internet Message Control Protocol (ICMP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). One-hot encoding converts this attribute into three feature columns as shown in Figure 10. It follows a 0/1 representation to indicate presence (indicated by 1) or absence (indicated by 0) of a value.

protocol_type
icmp
udp
tcp
tcp
udp

⇒

icmp	udp	tcp
1	0	0
0	1	0
0	0	1
0	0	1
0	1	0

Figure 10. One-hot encoding used in Protocol Type attribute

b) Dummy coding

Like one-hot encoding, dummy coding converts an attribute with N values to a feature set of N-1 values. The converted set of binary variables are called dummy variables. Figure 11 illustrates one-hot ending and dummy coding applied to the same set of categorical records.

Column	Code
A	100
B	010
C	001

One- Hot Coding

Column	Code
A	10
B	01
C	00

Dummy Code

Figure 11. One-hot and dummy encoding used in the same dataset [11]

c) Effect coding

While one-hot encoding and dummy coding use only 0 and 1 to encode categorical variables, effect coding sets values that sum to zero in the new feature set. As a result, negative values may also be generated in the encoded feature

set. Effect coding is a preferred choice when there is an interaction of categorical variables in a dataset as it can provide reasonable estimates of main effect and of the interaction [12].

d) Hash encoding

Hash encoding is appropriately used for categorical variables that have a large number of possible values. The method uses a hash function to map categorical values into numbers. Commonly used hash functions include Message Digest functions MD2, MD4, MD5 and Secure Hash Algorithms SHA0, SHA1, SHA2 and SHA3. The MD5 hash function is used by default [11]. Hash encoding returns a variable map with smaller dimension than other encoding schemes, such as one-hot encoding or dummy coding. Figure 12 below shows the hash-encoding process:

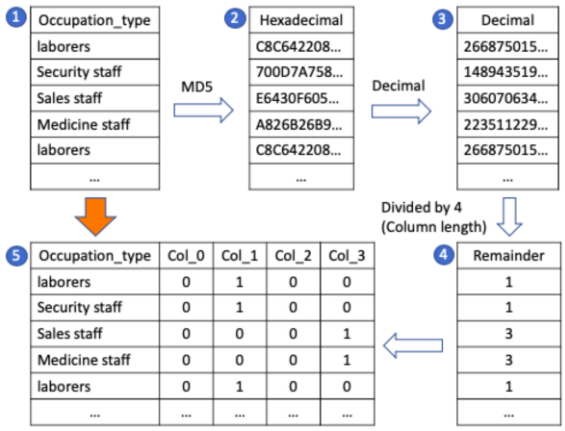


Figure 12. Hash Encoding Process [13]

e) BaseN encoding

BaseN encoding converts categorical variables into a consistently encoded feature set using a selected base, such as base 2 for binary encoding. The base or radix is the available number that can be used in different combinations to represent all values in a numbering system. A BaseN encoder encodes categorical values into arrays of their base-n representation.

f) Target encoding

Target encoding (also known as mean encoding) replaces a variable with a mean of the target value for that variable. Figure 13 provides an illustration. When the values of a categorical variable are already of a high volume, target encoding provides an advantage over other methods as it does not add extra dimensions to the dataset.

	feature	feature_label	feature_mean	target
0	Moscow	1	0.4	0
1	Moscow	1	0.4	1
2	Moscow	1	0.4	1
3	Moscow	1	0.4	0
4	Moscow	1	0.4	0
5	Tver	2	0.8	1
6	Tver	2	0.8	1
7	Tver	2	0.8	1
8	Tver	2	0.8	0

Figure 13. Target Encoding [14]

g) Label or ordinal encoding scheme

Ordinal categorical variables require that the order of the variable be preserved. For example, a road surface when a collision occurs might be categorized into dry, somewhat wet, or very wet so that the 3 values have an order that might provide additional insights. Ordinal encoding scheme aims at preserving this order when mapping values to numeric form. The method simply assigns each label a number (for example dry=1, somewhat wet=2 and very wet=3).

C. Feature Selection

The complexity of ML models increases with the dimensionality of the dataset. Predictive models often fail to achieve high accuracy because of inadequate analyses directed to feature selection. Selecting the most important and significant features reduces the complexity of the model and can also increase the prediction performance [36], [37]. Multiple approaches are available and effective for reducing the feature set. Prominent ones are described below.

Filter Methods. In our work, we choose feature selection methods that apply to situations with a categorical output, such as whether an input vector is an attack or not. The filter methods eliminate features independently of the ML method used. A univariate feature filter evaluates the importance of single features using univariate statistical tests. Each feature is paired with the target to evaluate statistical significance between them. The analysis of variance or ANOVA F-test is widely used. The F-test calculates the ratio between variance values [30]. The resultant measures of the relative importance of individual features provides a tool for determining features that are unnecessary or of little importance.

Wrapper Methods. The wrapper methods directly evaluate combinations of features by running the ML model restricted to the set of candidate features. Taken to an extreme, all combinations would be evaluated, an impossible task in practice. Thus, various search space approaches are employed. Forward search iteratively adds promising feature vectors one by one to build a feature set. Backward search starts with all features and successively eliminates those that

 ISBN: 978-1-61208-990-4

perform poorly. Many approaches based on optimization techniques are available [31]. The self-driving car work basically follows a forward selection approach based on both advance knowledge about the importance of certain features from analytics on the data itself along with test runs of the ML model. Heavy computational load and possibilities of overfitting are potential drawbacks [33].

Embedded Methods. Embedded methods utilize mathematical information that is available during the training of the model to determine the relative importance of features. In some sense embedded methods mitigate the drawbacks of the filter and wrapper methods but retain their strengths. When implemented carefully, they are not prone to overfitting [34]. The XGBoost technique produces an importance score for each attribute that is used to identify those that can be confidently eliminated [35]. In applications like intrusion detection, a large number of attributes presents a huge computational burden. The embedded methods are highly successful in greatly reducing the features needed in intrusion detection ML work.

VI. CONCLUSION AND FUTURE WORK

Machine learning is now a well-established and effective approach in many domains. The studies on collisions involving self-driving cars and on intrusion detection in networks reported in this paper are examples of complex problem-solving domains with special challenges. These applications have dimensions of importance in inter-related areas of cybersecurity, trust, risk, safety, reliability, autonomy, and anti-autonomy. The data-centric and feature engineering challenges are extensive, but addressable. We describe approaches to addressing these challenges. Results reveal several implications for research needs and frontiers for next steps in research. These include the quest for methods that can be deployed in real-time, automate feature engineering, choose, and extract features dynamically, and simultaneously support multiple performance evaluation metrics.

REFERENCES

[1] T. Hey, S. Tansley, and K. Tolle "The Fourth Paradigm: Data-Intensive Scientific Discovery, " Redmond, Washington: Microsoft Research, 2009.

[2] A. Rastogi, "Trust and Anti-Autonomy Modelling of Autonomous Systems," Order No. 28255688, North Dakota State University, Ann Arbor, 2020.

[3] R. Meiri and J. Zahavi. "Using simulated annealing to optimize the feature selection problem in marketing applications," In *European Journal of Operational Research*, vol. 171, no. 3, pp. 842-858, 2006.

[4] M. Lombardi and M. Milano, "Boosting combinatorial problem modeling with machine learning," In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 5472-5478, 2018.

[5] M. Charikar, V. Guruswami, R. Kumar, S. Rajagopalan, and A. Sahai, "Combinatorial feature selection problems," In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 631-640, 2000.

[6] KDD Cup 1999 Data, Kdd.ics.uci.edu [Online]. Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Accessed: 22 August, 2022].

[7] A Deeper Dive into the NSL-KDD Data Set. Medium [Online]. Available from: https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657 [Accessed: 22 August, 2022].

[8] L. Dhanabal and S.P. Shantharajah "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," In *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446-452, June 2015.

[9] S. Choudharya and N. Kesswani "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT," In *Procedia Computer Science*, vol. 167, pp. 1561-1573, 2020.

[10] A. Divekar, M. Parekh, V. Savla, R. Mishra and M. Shirole "Benchmarking datasets for Anomaly-based NetworkIntrusion Detection: KDD CUP 99 alternatives," In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), IEEE*, pp. 1-8, 2018.

[11] 8 Categorical Data Encoding Techniques to Boost your Model in Python!, Analytics Vidhya [Online]. Available from: https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/ [Accessed: 22 August, 2022].

[12] Introduction to SAS, UCLA: Statistical Consulting Group. [Online]. Available from: https://stats.idre.ucla.edu/sas/modules/sas-learning-moduleintroduction-to-the-features-of-sas/ [Accessed: 22 August, 2022]

[13] A Data Scientist's Toolkit to Encode Categorical Variables to Numeric. [Online]. Available from: https://towardsdatascience.com/a-data-scientists-toolkit-to-encode-categorical-variables-to-numeric-d17ad9fae03f [Accessed: 22 August, 2022]

[14] Improve your classification models using Mean /Target Encoding. [Online]. Available from: https://medium.com/datadriveninvestor/improve-your-classification-models-using-mean-target-encoding-a3d573df31e8 [Accessed: 22 August, 2022]

[15] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," In *2015 military communications and information systems conference (MilCIS), IEEE*, pp. 1-6, 2015.

[16] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," InIn *Intelligent data analysis*, vol, 6, no. 5, pp. 429-449, 2002.

[17] R. Gomes, M. Ahsan, and A. Denton, "Random Forest Classifier in SDN Framework for User-Based Indoor Localization," In *2018 IEEE International Conference on Electro/Information Technology (EIT), IEEE*, pp. 0537-0542, 2018.

 ISBN: 978-1-61208-990-4

[18] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: A review," In *Int. J. Adv. Soft Comput. its Appl.*, vol. 5, no. 3, 2013.

[19] A. Ghazikhani, H. S. Yazdi, and R. Monsefi, "Class imbalance handling using wrapper-based random oversampling," In *20th Iranian Conference on Electrical Engineering (ICEE2012), IEEE*, pp. 611-616, 2012.

[20] Z. Zheng, Y. Cai, and Y. Li, "Oversampling method for imbalanced classification," In *Computing and Informatics*, vol. 34, no. 5, pp. 1017-1037, 2015.

[21] A. M. Denton, M. Ahsan, D. Franzen, and J. Nowatzki, "Multi-scalar analysis of geospatial agricultural data for sustainability," In *2016 IEEE International Conference on Big Data (Big Data), IEEE*, pp. 2139-2146, 2016.

[22] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special Issue on Learning from Imbalanced Data Sets," In *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 1-6, 2004.

[23] M. Ahsan, R. Gomes, and A. Denton, "SMOTE Implementation on Phishing Data to Enhance Cybersecurity," In *2018 IEEE International Conference on Electro/Information Technology (EIT), IEEE*, pp. 0531-0536, 2018.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," In *Journal of artificial intelligence research*, vol. 16, pp. 321-357, 2002.

[25] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," In *International conference on intelligent computing*, pp. 878-887. Springer, Berlin, Heidelberg, 2005.

[26] H. Hairani, K. E. Saputro, and S. Fadli, "K-means-SMOTE untuk menangani ketidakseimbangan kelas dalam klasifikasi penyakit diabetes dengan C4.5, SVM, dan naive Bayes," In *J. Teknol. dan Sist. Komput.*, vol. 8, no. 2, pp. 89-93, 2020.

English – H. Hairani, K. E. Saputro, and S. Fadli, K-means-SMOTE to trate class imbalance in the classification of diabetes with C4.5, SVM, and Naïve Bayes, *In J. Teknol. dan Sist. Komput.*, vol. 8, no. 2, pp. 89-93, 2020.

[27] Y. Tang, Y. Q. Zhang, and N. V. Chawla, "SVMs modeling for highly imbalanced classification," In *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 281-288, 2008.

[28] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE*, pp. 1322-1328, 2008.

[29] O. Özdemir, M. Batar, and A. H. Işık , "Churn Analysis with Machine Learning Classification Algorithms in Python," In *The International Conference on Artificial Intelligence and Applied Mathematics in Engineering*, pp. 844-852. Springer, Cham, 2019.

[30] N. O. F. Elssied, O. Ibrahim, and A. H. Osman, "A novel feature selection based on one-way ANOVA F-test for e-mail spam classification," In *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, no. 3, pp. 625-638, 2014.

[31] M. Ahsan, R. Gomes, and A. Denton, "Application of a convolutional neural network using transfer learning for tuberculosis detection," In *2019 IEEE International Conference on Electro Information Technology (EIT), IEEE*, pp. 427-433, 2019.

[32] A. Mustaqeem, S. M. Anwar, M. Majid, and A. R. Khan, "Wrapper method for feature selection to classify cardiac arrhythmia," In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE*, pp. 3656-3659, 2017.

[33] M. Ahsan and K. Nygard, "Convolutional Neural Networks with LSTM for Intrusion Detection," In *The 35th International Conference on Computers and Their Applications (CATA 2018)*, vol. 69, pp. 48-59, pp. 69-79. 2020.

[34] H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," In *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 703-715, 2019.

[35] Y. Wang and X. S. Ni, "A xgboost risk model via feature selection and bayesian hyper-parameter optimization," In *arXiv preprint arXiv:1901.08433*, 2019.

[36] M. Chowdhury, J. Tang and K. Nygard, "An artificial immune system heuristic in a smart grid," In *The 28th International Conference on Computers and Their Applications*, 2013.

[37] M. Chowdhury and K. Nygard, "Machine learning within a con resistant trust model," In *The 33rd International Conference on Computers and Their Applications (CATA 2018)*, pp. 9-14, 2018.

[38] A. Rastogi and K. Nygard, "Threat and alert analytics in autonomous vehicles," In *The 35th International Conference on Computers and Their Applications (CATA 2018)*, vol. 69, pp. 48-59, 2020.

[39] Home - Keras Documentation, Keras.io, 2020. [Online]. Available: https://keras.io/. [Accessed: 16 August, 2022].

[40] "TensorFlow", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/. [Accessed: 16 August, 2022].

[41] "tensorflow/tensorflow", GitHub, 2020. [Online]. Available: https://github.com/tensorflow/tensorflow. [Accessed: 16 August, 2022].

[42] M. Ahsan, , N. Rifat, M. Chowdhury, and R. Gomes, Detecting Cyber Attacks: A Reinforcement Learning Based Intrusion Detection System. *In 2022 IEEE International Conference on Electro Information Technology (eIT)*, IEEE, pp. 461-466, 2022.

[43] M. Ahsan, K. E. Nygard, R. Gomes, M . M . Chowdhury, N. Rifat, and J. F. Connolly, "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review, " *In Journal of Cybersecurity and Privacy*, vol. 2, no. 3, pp. 527-555, 2022.

[44] N. I. Rifat, "Feature Engineering on the Cybersecurity Dataset for Deployment on Software Defined Network, " 2020.

[45] Y. Berra, "A Quote By Yogi Berra", 2022. [Online]. Available: https://www.goodreads.com/quotes/261863-it-s-tough-to-make-predictions-especially-about-the-future. [Accessed: 22-August-2022].

[46] M. Ahsan, N. Rifat, M. Chowdhury, and R. Gomes, "Intrusion Detection for IoT Network Security with Deep Neural Network, " *In 2022 IEEE International Conference on Electro Information Technology (eIT)*, IEEE, pp. 467-472, 2022.

 ISBN: 978-1-61208-990-4

(Un)Like Schumann: Applying Cope's Music Signature Pattern Matching Algorithms to Tchaikovsky's Children's Album

Evgeny Pyshkin
The University of Aizu
Aizu-Wakamatsu, 965-8580
Japan
email: pyshe@u-aizu.ac.jp

Andrei Kuznetsov
JetBrains
1017 ZM Amsterdam
Netherlands
email: andrei.kuznetsov@jetbrains.com

Andrey Matveets
Peter the Great St. Petersburg Polytechnic University
St. Petersburg, 195251
Russia
email: matveets.av@gmail.com

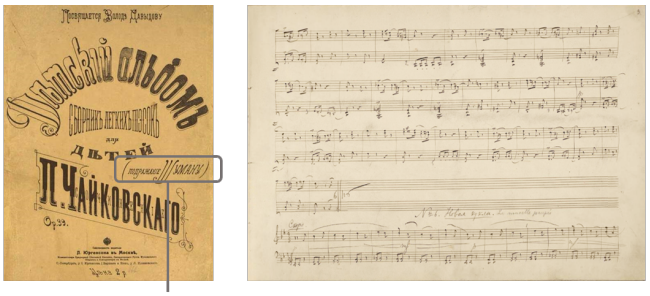
***Abstract*—This paper describes a preliminary study looking into applying pattern matching algorithms working with music signatures to the "Children's Album" by Pyotr Tchaikovsky. Though music signatures introduced by Cope are usually used for author identification and computer music generation, we make an effort to use them for analysis of origins and links of the existing compositions. We take Tchaikovsky's "Children's Album" as an interesting case, where we can try to apply the computational models to resolving the questions, which are usually mostly in scope of musicology studies. Specifically, our experiments demonstrate that one can find only a few Schumann signatures in the pieces from the "Children's Album", in contradiction to the Tchaikovsky's note in the published version claiming the imitation of Schumann's approach. Thus, our experimental results can provide additional important insights for musicologists searching to unravel the possible reasons of significant transformations that occurred on the way from the accurately organized manuscript to the first published edition. The previous studies addressing this issue are mostly in the scope of music theory, with almost no involvement of computational approaches to music analysis. Techniques based on formal mathematical methods and computer technology, though being unable to completely resolve these issues, bring new data to the discourse of musicology and art.**

Keywords–Musicology; music information retrieval; human-centric computing; music similarity; pattern matching; music signature; music modeling.

I. Introduction

In [1], in the scope of a conceptualization of the analysis of Pyotr Tchaikovsky's "Children's Album" (Op. 39) with the use of computational models, an approach based on David Cope's signatures [2][3] is sketched as a promising way to help musicologists in resolving a number of riddles posed by Tchaikovsky in his famous cycle of 24 piano pieces thought to be for children. The analysis of possible sources, metaphors, and renditions of this masterpiece originally published as far as in 1878 by Yurgenson [4] still remains a constant topic of interest for researchers [5][6]. While admitting Schumann's influence on Tchaikovsky (along with other precursors, such as Bach, Mozart, Beethoven, Chopin, or Berlioz), there is still a challenging question on whether we have to completely accept the author's claim that the compositions from the "Children's Album" are a form of imitation of Schumann's pieces for the young [7], even with respect to the subtitle appeared in the first published edition by Yurgenson: "Simple pieces for children *(imitating Schumann)*" (Figure 1*(a)*). Interestingly, such a subtitle is missing in the manuscript [8], though the latter is a fascinating example of the accurately presented and organized hand-written work (Figure 1*(b)*).

Unlike Tchaikovsky, from the outset Schumann had not intended for his collection of compositions for children to be a seamless large work, finally organized as Op. 68. Indeed, he initially composed 10 pieces considered as nice exercises for his own children, and even though he announced its completion, more pieces appeared later, in 1848. By composing the exercises which would be nicer than most things that children normally needed to play during their piano studies, Schumann "not only revolutionized attitudes concerning music education, but also inaugurated an entirely new genre of piano literature – programmatic music written explicitly for children" [9]. The latter facts do not contradict the idea that Tchaikovsky could still have wished to imitate Schumann's approach, though musicological analysis of his Op. 39 usually debates this hypothesis much.



(a) Subtitle "imitating Schumann"

(b) Fragment of No. 5 "March of Wooden Soldiers" immediately followed by No. 6 "A New Doll" on the same sheet

Figure 1. Cover of the Yurgenson's edition and a fragment from Tchaikovsky's manuscript.

The remaining text is organized as follows. Section II provides an introduction to the concept of signatures according to Cope. In Section III, we describe a case study on signature elicitation for Tchaikovsky's "Children's Album". To conclude, we summarize the most important insights on how the formal approach we used can be beneficial for musicology experts. We also sketch necessary extensions that can improve the accuracy and veracity of the applied computational models.

II. Music Signatures by Cope

Signatures form one of the core elements of music representation and automatic generation system developed by Cope [2][3], which uses an implementation of augmented transition network, a finite-state automaton with recursive succession rules between music sub-phrases allowing for logical syntax substitutions [10].

 ISBN: 978-1-61208-990-4

Cope defined a *signature* as a set of contiguous intervals found in more than one work by the same composer [11]. Typically, a signature is composed of two to nine notes (or more, if combined with harmonies). The idea of signature is to represent a composition-independent pattern, which does not sound as an excerpt from a particular work, but rather represents a characteristic description of one of composer's style elements.

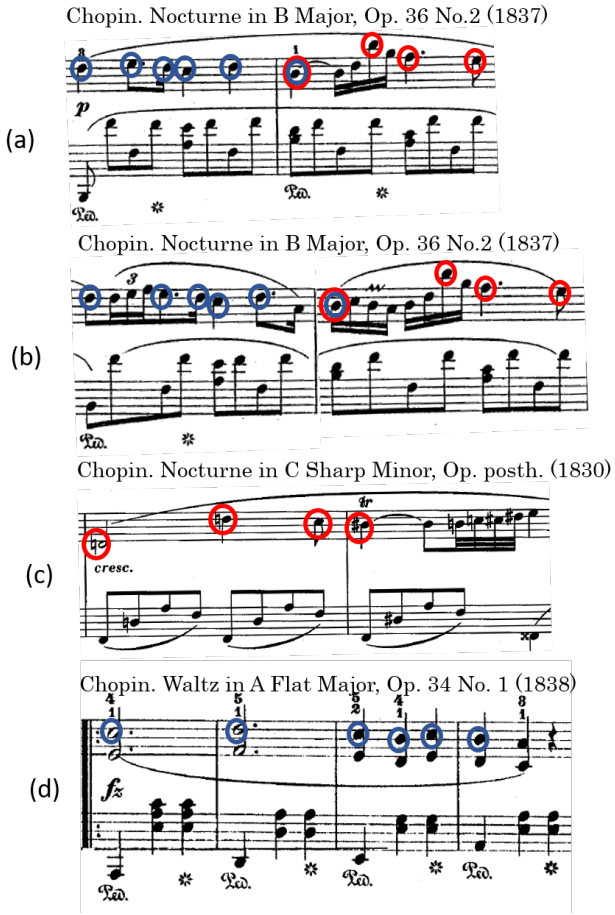


Figure 2. Possible signature candidates from Chopin's compositions.

Signatures portray patterns combining melody, harmony and rhythm information, with possible **transformations** of interval, pitch, rhythm and voice exchange. Even transformed, the patterns can still be recognised by ear. Figure 2 shows two examples of possible pattern candidates coloured red and blue along with their transformations in the works of Chopin.

As we can see from the examples in Figure 2, many significant patterns need to be detected approximately, not exactly. Cope compares the pattern matching fine-tuning process to the sieving the candidates through a mesh. The sieving process controls the *granularity* of allowed pattern transformations enabling inexact pattern matching. Cope introduced a number of parameters – controllers. The controllers help to establish pattern matching thresholds to work with variances and be able to find signatures, which are not identical, but are musically comparable. For example, if we set the maximum number of notes in the target pattern too small, the discovered patterns might be too short to be sufficiently reflective (though, some harmonic cadence signatures can be short enough). If we set the value of the same parameter too high, the resultant output can be too specific to a particular composition, and again, unreflective of a composer's style in general. To illustrate the concept, there is a list of some examples of controllers as defined by Cope [12]:

- `allowance` – defines the possible deviation of the melody in half steps (see example with the red patterns in Figure 3).
- `contour` – defines how much the general contour of two fragments conforms to each other (see the cases *(b),(c)* in Figure 4).
- `inversion` – looks for patterns with inverted sequences of notes (or intervals).
- `interpolation` – allows for intervening notes (see the cases *(a),(b)* in Figure 2).
- `pattern-size` – defines the size of pattern selected for comparison.
- `rhythm` – determines whether the patterns match because of rhythmic match only (like in many genre pieces, such as dances, where the rhythm is a significant component of the style).

Reusing a signature of one composer in the work of another author (often with recombination and variations) could create allusions to appear as a more sophisticated construction rather than straightforward borrowing of melodies or motifs.

In Figure 3, we can see the analysis demonstrating an exact pattern reuse (shown in blue), as well as possible inexact variation (shown in red). In the latter case, typical pattern with a sub-melody often appearing in the second voice of Chopin's piano composition, also used by Adam with a slightly transformed sub-melody (shown using a dashed line).

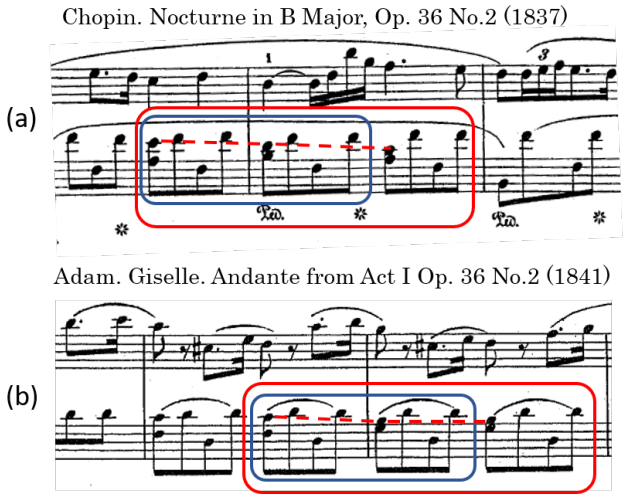


Figure 3. Similar patterns in the works of Chopin and Adam.

Figure 4 illustrates pattern elicitation in Chopin's Waltz in C Sharp Minor *(b)*, which can be inexactly matched to the pattern found in Griboyedov's Waltz in E Minor *(a)*. The latter work contains the pattern *(d)* matching (with slight transformations) the fragment from the part 2 of Mozart's piano concerto No. 23.

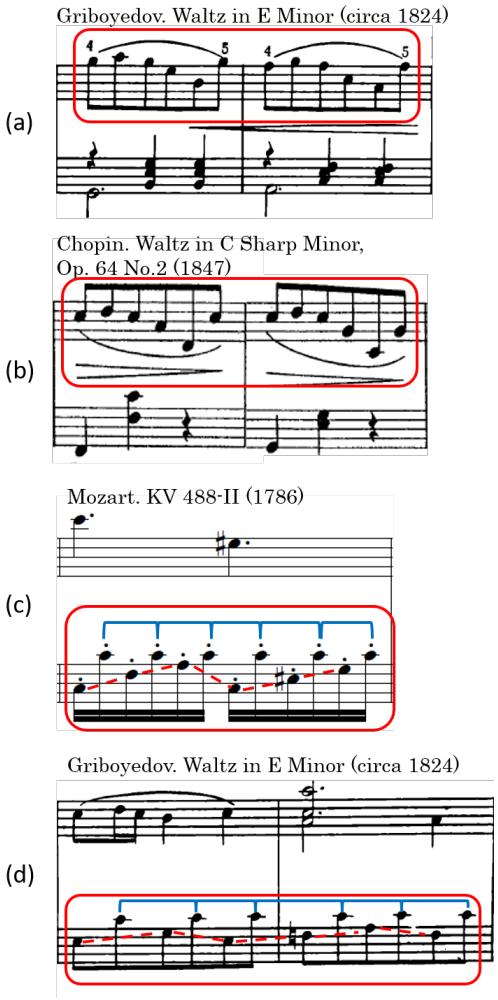


Figure 4. Examples of inexact pattern matching in the works of Mozart, Griboyedov, and Chopin.

Pattern matching methods based on signature elicitation can be applied to authorship identification, along with other known approaches, such as analysis and modeling of music structure using N-grams [13][14], Markov chain models [15][16], deep neural networks [17][18], analysis of high-level music features like counterpoint structures [19], grammatical inference [20], and cortical algorithms [21]. However, signatures are not only helpful for authorship attribution, but also for in-depth analysis of music compositions to discover the characteristics of style, their genesis, and their development. The links between the authors and the periods may also be identified, which will be of great interest for musicologists.

III. "Children's Album" as a Test Bed for Signature Elicitation Algorithm

In this section, we describe a case study on applying the signature elicitation software to obtain new data showing the links between the compositions from "Children's Album" and the works of other composers that influenced the style of Tchaikovsky.

A. Software

The signature discovery process was organized using the software we developed [22] based on Cope's algorithms adapted to the present-day standard music analysis tools available in numerous Python libraries. Notably, in order to parse and analyze input files in different formats (like MIDI and KERN) we are using `Music21` [23] – an open source library for computer-aided musicology created and supported by the the Music and Theater Arts Section, MIT.

The software we developed accepts pairs of *(file,author)*, where the file contains notated music in symbolic formats like MIDI [24] or KERN [25]. Since the source file usually contains polyphonic music, the preparation step is to extract the melody (we use Skyline[26]-like algorithm).

The next stages of the workflow are as follows:

1) In each *single* composition, find the patterns appearing 5 to 12 times. These patterns are considered as signature candidates.
2) Collect the candidates across the different available works of the given composer.
3) From the complete list of collected patterns, eliminate the patterns that do not appear often enough (i. e., in less than 10% of all the analyzed works of the same composer).
4) Compare the found patterns against the patterns of different composers and eliminate the patterns, which are frequently used by other composers, as such patterns that appear to characterize the epoch rather than a particular composer's style.

As a result of the above-described procedure, there is a database of signatures considered to uniquely characterize each single composer (at least, within the context of the given dataset), along with additional metadata, such as the location of signatures in the original works.

We used our software in the experiments with the composer identification task (to be discussed in the separate publication), and found that the signatures can be used to reliably identify composers in many cases. In terms of *success rate*, the achieved results are comparable to some well-known approaches such as Markov chains (giving a rate of 75%), but not so fine-grained compared to the best approaches demonstrating up to 90% success rate.

As a side result of these experiments, we built a database containing the characteristic signatures of Bach, Beethoven, Haydn, Mozart, Vivaldi, and others, extracted from a dataset containing over 100 works of each author.

B. Data

With respect to the goals of this case study, our training set was constructed using 90 compositions by Tchaikovsky, and 61 works by Schumann (including those from the Album for the Youth, Op. 68). Thus, we used an input dataset of 151 works in total, to extract signatures of both composers, and extend the collection of found signatures with the signatures of other composers, which became available due to the process of evaluating our software for signature based composer identification. Adding the signatures of other composers can be helpful in filtering the output so as to discard patterns that may be attributes of period or genre rather than of particular composers.

 ISBN: 978-1-61208-990-4

Our target testing set included all the 24 piano pieces from Tchaikovsky's "Children's Album".

C. Experiment at a Glance

For this experiment, we used a database of more than 26485 signatures in total (including variations). The database was filtered by eliminating 11745 duplicates that can be found in the works of different composers. The resulting log is available at https://github.com/andrei-kuznetsov/signatures/files/9377147/signatures-wo-dupplicates.txt.

TABLE I. SIGNATURES FOUND IN TCHAIKOVSKY'S "CHILDREN'S ALBUM"

Composer	Acronym	Signatures (including variations)	Compositions
Tchaikovsky	TCH	19	13
Haydn	HAY	20	8
Beethoven	BEE	9	8
Mozart	MOZ	8	7
Vivaldi	VIV	13	5
Bach	BAC	1	1
Schumann	SCH	5	3

TABLE II. SIGNATURE DISTRIBUTION

No.	TCH	HAY	BEE	MOZ	VIV	BAC	SCH	Total
1					1			1
2	1		1	1				3
3	1						2	3
4	1		1					2
5								0
6	1	1			1			3
7			1					1
8		1						1
9	1	3	1	2	2			9
10				1				1
11	1			1		1		3
12		1					1	2
13			1					1
14	4			1				5
15			1	1				2
16		1						1
17		3	2					5
18		9	1	1	3			14
19	3							3
20								0
21	2							2
22	1				6		2	9
23	1							1
24	3							3
All	**19**	**20**	**9**	**8**	**13**	**1**	**5**	**75**
All*	**18**	**11**	**8**	**7**	**4**	**1**	**3**	**52**

* excluding No. 18 and No. 22, where disproportionately big number of signatures of one composer is found compared to others

Table I lists the cumulative results of signature elicitation process. For each composer, we show the total number of signatures found and the number of compositions in which the signatures were discovered. Here and after we use the composition numbering according to Tchaikovsky's manuscript [8]. Table II shows the distribution of signatures among all the compositions.

Based on the analysis of *All* row from Table II, we could suggest that the results of automatic signature elicitation process need to be corrected with further elimination of some repeated signatures based on manual expertise of music files with marked up signatures (e.g., using *MuseScore* software). For example, in the compositions No. 18 "Neapolitan Song" and No. 22 "Lark's Song", there is a definitive disproportion between the signatures of one composer against the signatures of others. It can be explained by the fact that in these compositions notably we can find a lot of small repetitive (but not always completely equal) patterns (as shown in Figure 5) that could lead to signature over-count.

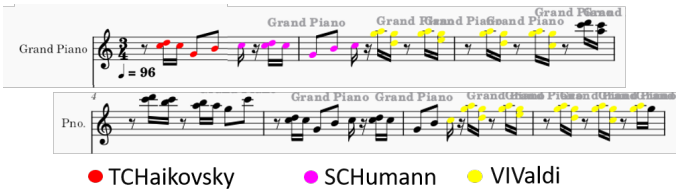


Figure 5. Repeating patterns in No. 22 "Lark's Song".

D. Preliminary Analysis

Schumann signature variations are only found in 3 compositions: No. 3 "Mama" (2 cases), No. 12 "Russian song" (quite surprisingly perhaps), and No. 22 "Lark's Song" (2 cases). We can also see that in all these cases the characteristic signatures of Schumann appear along with other signatures, particularly, in No.22, when, in addition to those of Schumann, we discovered 7 signatures of other composers.

From the experiments with the signature database, we learned that the recognition rate for particular composers is unstable and ranges between around 50% to 80% . Even if we acknowledge that some unique Schumann signatures were missed due to the imperfectness of automatic signature elicitation process, we can still argue that compared to other discovered signatures, there is a very low number of Schumann's cases. Though, in concordance with Cope's definition, an imitation of style does not assume appearance of signatures of the imitated composers, their very rare occurrences provide a rationale for disputing the possibility for deliberate imitation of Schumann's style by Tchaikovsky.

IV. Discussion

The question of attributing Tchaikovsky's masterpiece as an imitation of Schumann is important as a part of the broader challenge: to approach possible explanations on why we find so many disruptive transformations in the first published edition compared to the so accurately prepared manuscript. Figure 6 illustrates the transformations related to the order of compositions. Since if we can provide a good rationale for understating the author's claim on imitation, we can also call into question the meaningfulness (or rather meaninglessness) of those transformations destroying the structure of the album as an indissociable whole, and deforming the micro-cycles and internal links existing in the manuscripts [27][28].

So far, the studies of the above-mentioned questions mostly remained in scope of music and art theory, with almost no involvement of computational approaches to music analysis. Techniques incorporating the formal mathematical methods and computational approaches could not (fortunately) completely resolve these questions *ex cathedra*; however, they could produce a number of important additional insights to the problems usually addressed exclusively from the musicology and human science positions.

As mentioned earlier [1], one could hardly accept an idea that the alteration of numbers, which led to destruction

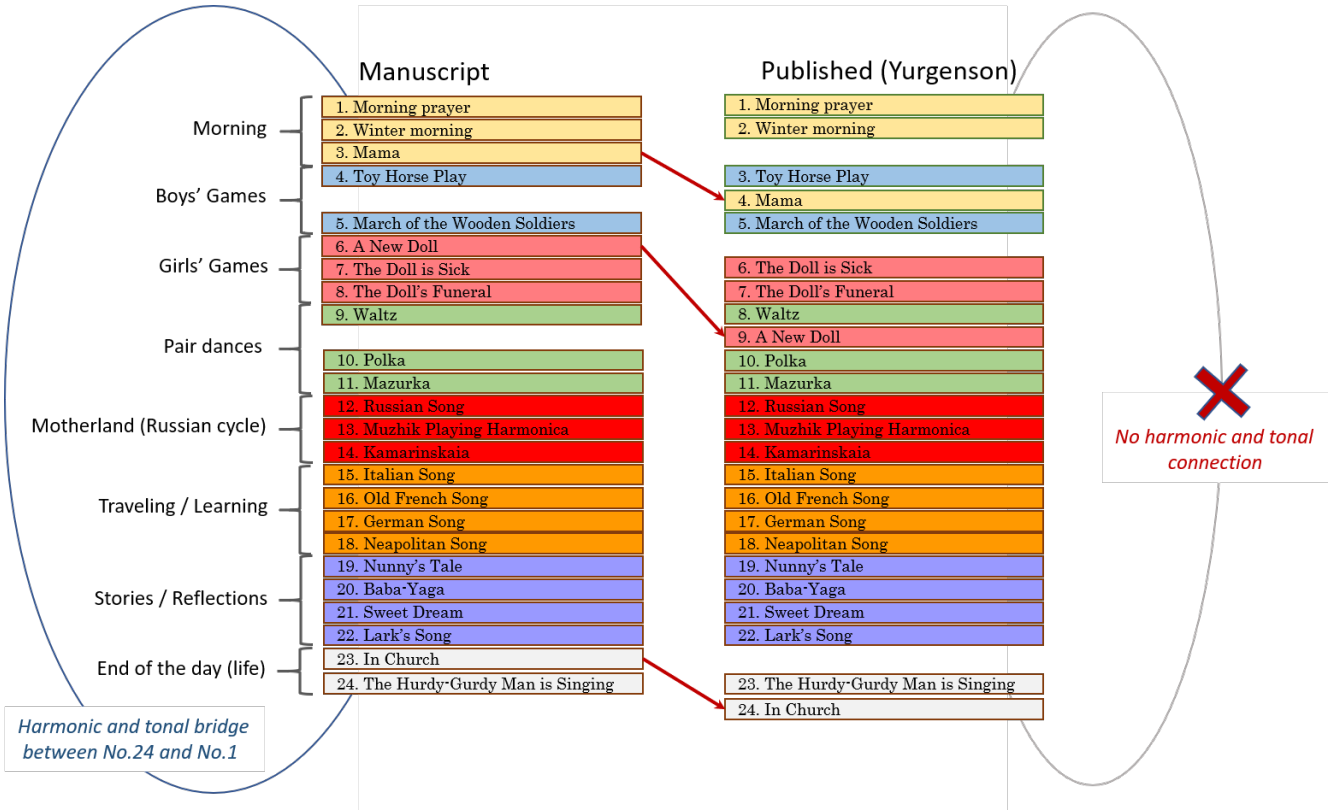


Figure 6. Order of compositions and micro-cycles in "Children's Album".

of junctions between the pieces existing in the manuscript, was a publisher's mistake. Indeed, Tchaikovsky approved and signed that version. Nekhaeva suggested that these serious transformations (partially shown in Figure 6) can be interpreted as a "gesture of the composer, a natural desire to overcome the temporary barrier and directly appeal to future generations of musicians" [6]. This opinion supports a hypothesis claiming that Tchaikovsky probably preferred to simplify the language and to decrease the emotional tension of the original version, and, in so doing, to hide some metaphors, to make them less explicitly exposed. The author's explicit claim on imitating Schumann's approach could be understood being in line with the above mentioned simplifications and transformations. Therefore, by raising the arguable doubts on this declared imitation, we can support the analysis of appropriateness of the discussed transformations as well, and, actually, vice versa.

V. Conclusion

The novel results we obtained from our preliminary experiments are very interesting, though not sufficient, and need to be reexamined after extending the dataset with respect to the following important types of input:

- Compositions with expected high degree of style similarity, which were attributed by their authors as imitations; and
- Characteristic compositions (e.g., by Tchaikovsky), where style similarity was reported by musicology experts.

The studies [29][30] can provide information for selection of relevant referential datasets necessary to improve the accuracy of the signature based music pattern recognition process.

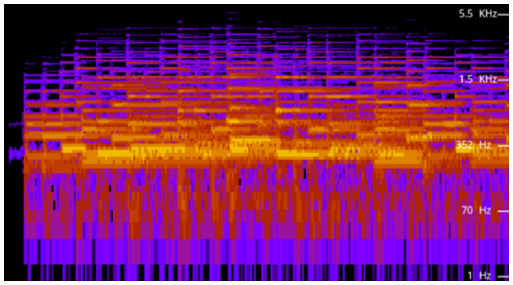


Figure 7. Sample spectrogram of "A New Doll" on a logarithmic scale (Op. 39, orig. No. 6) created using Spectrum Analyzer [31].

It is worth noting that the discussed signature elicitation models do not oppose an idea to use machine learning approaches to music style identification. On the one hand, we already mentioned a number of deep learning methods suggested by different researchers for author identification and style recognition. These models often work with the input represented in the form of spectrograms (similar to one shown in Figure 7).

Though there are many challenges in constructing an explainable machine learning algorithm producing the results and making conclusions that can be understood by musicologists, the possibility to apply such approaches to our problem need to be investigated in more depth. On the other hand, the process of signature discovery itself can be implemented using machine

 ISBN: 978-1-61208-990-4

learning (e.g., CNN as the most common possible solution). Such an implementation surely would be an interesting option for further explorations. However, though not novel, the semi-automated approach based on Cope models, can still be relevant to our study for a number of important reasons:

1) Rather then asking a question of author or style attribution, we are searching for the structural elements that can be understood by music experts and enhance their knowledge on the genesis and development of music style in the work of a particular composer (e.g., Tchaikovsky).
2) With respect to the possibility to develop or use machine learning algorithms (including those applied to the process of signature elicitation), one needs to have ground truth information that, as we believe, can be delivered based on "adjustable machine-oriented models", such as those described by Cope.
3) In contrast to signatures defined using music notations (e.g., music scores) and representations (e.g., MIDI), and, therefore, can be directly analyzed and perceived by music experts, machine learning features, coefficients, and probability distributions are definitely less favorable to humans.

We admit that playing with pattern matching settings and further adjustments of the signature elicitation algorithms might affect the specific signature scores we obtain from these algorithms. However, our preliminary experiments demonstrate that the relative distribution of signatures between different composers does not fluctuate too heavily upon the changes in pattern matching controlling parameters; thus, making our qualitative judgements well-reasoned, though not conclusive.

ACKNOWLEDGEMENT

Many thanks to Natalia Bogach and John Blake for their very helpful suggestions during our discussions of this project.

REFERENCES

[1] E. Pyshkin, "Towards demystifying transformations of Tchaikovsky's children's album with support of computational models: Problem conceptualization," in Proceedings of the 15th International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2021). IARIA, 2021, pp. 6–10.

[2] D. Cope, "Experiments in musical intelligence (EMI): Non-linear linguistic-based composition," Journal of New Music Research, vol. 18, no. 1-2, 1989, pp. 117–139.

[3] ——, Computer models of musical creativity. MIT Press Cambridge, 2005.

[4] P. Tchaikovsky, Children's Album. Op 39. Yurgenson, 1878.

[5] A. Lazanchina, "Fenomen detstva v fortepiannykh tsiklakh R. Shumana i P. Tchaikovskogo (The phenomenon of childhood in piano cycles by R. Schumann and P. Tchaikovsky)," in Transactions of Russian Academy of Science Samara Research Center. RAS, 2015, pp. 1224–1227, (In Russian).

[6] I. Nekhaeva, "The modern measurement of Tchaikovsky (Definition experience of the "modernity" concept on example of the "Children's Album" by PI Tchaikovsky)," in Tomsk State University Journal of Cultural Studies and Art History. Tomsk State University, 2018, vol. 30, pp. 146–147, retrieved: Aug, 2022. [Online]. Available: http://case.asu.ru/files/form_312-31545.pdf#page=146

[7] R. Schumann, 43 Piano Pieces for the Youth. Op 68 (Orig. Title in German: 43 Clavierstücke für die Jugend). Schuberth and Co., 1848.

[8] "Tchaikovsky: Otkrytyi mir. Detskiy albom (Tchaikovsky: Open world. Children's Album," 2015, retrieved: Aug, 2022 (In Russian). [Online]. Available: https://www.culture.ru/catalog/tchaikovsky/ru/item/archiv/detskiy-albom-24-legkih-pesy

[9] L. Deahl, "Robert schumann's" album for the young" and the coming of age of nineteenth-century piano pedagogy," in College Music Symposium, vol. 41. JSTOR, 2001, pp. 25–42.

[10] P. da Silva, David Cope and Experiments in Musical Intelligence. Spectrum Press, 2003.

[11] D. Cope, Computers and musical style. Oxford University Press Oxford, 1991, vol. 6.

[12] ——, Experiments in Musical Intelligence, 2nd Ed. A–R Editions, 2014.

[13] J. Wołkowicz, Z. Kulka, and V. Kešelj, "N-gram-based approach to composer recognition," Archives of Acoustics, vol. 33, no. 1, 2008, pp. 43–55.

[14] M. Hontanilla, C. Pérez-Sancho, and J. M. Inesta, "Modeling musical style with language models for composer recognition," in Iberian conference on pattern recognition and image analysis. Springer, 2013, pp. 740–748.

[15] Y.-W. Liu and E. Selfridge-Field, "Modeling music as markov chains: Composer identification," 2002, retrieved: Aug, 2022. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.8307&rep=rep1&type=pdf

[16] M. A. Kaliakatsos-Papakostas, M. G. Epitropakis, and M. N. Vrahatis, "Weighted markov chain model for musical composer identification," in European conference on the applications of evolutionary computation. Springer, 2011, pp. 334–343.

[17] G. Buzzanca, "A supervised learning approach to musical style recognition," in Music and artificial intelligence. Additional proceedings of the second international conference, ICMAI, vol. 2002, 2002, p. 167.

[18] Z. Hu, K. Fu, and C. Zhang, "Audio classical composer identification by deep neural network," arXiv preprint arXiv:1301.3195, 2013.

[19] L. Mearns, D. Tidhar, and S. Dixon, "Characterisation of composer style using high-level musical features," in Proceedings of 3rd international workshop on Machine learning and music, 2010, pp. 37–40.

[20] J. Geertzen and M. van Zaanen, Composer classification using grammatical inference. Narcis, 2008.

[21] N. Hajj, M. Filo, and M. Awad, "Automated composer recognition for multi-voice piano compositions using rhythmic features, n-grams and modified cortical algorithms," Complex & Intelligent Systems, vol. 4, no. 1, 2018, pp. 55–65.

[22] "Phyton implementation of music signature search algorithm based on david cope's approach," retrieved: Aug, 2022. [Online]. Available: https://github.com/Nekobitlz/signatures

[23] "Music21: A toolkit for computer-aided musicology," retrieved: Jul, 2022. [Online]. Available: https://web.mit.edu/music21/

[24] G. Loy, "Musicians make a standard: the midi phenomenon," Computer Music Journal, vol. 9, no. 4, 1985, pp. 8–26.

[25] C. S. Sapp, "Online database of scores in the humdrum file format." in ISMIR, 2005, pp. 664–665.

[26] A. Uitdenbogerd and J. Zobel, "Melodic matching techniques for large music databases," 01 1999, pp. 57–66.

[27] A. Kandinskiy-Rybnikov and M. Mesropova, "Vremena goda i Detskiy albom Tchaikovskogo: Tsyklichnost i problemy ispolneniya (Tchaikovsky's The Seasons and Children's Album: Cyclicity and performing problems)," in Tchaikovsky: Voprosy istorii, teorii i ispolnitelstva (Tchaikovsky. Questions on history, theory, and performing). Moscow Conservatory, 1990, pp. 120–137, (In Russian).

[28] ——, "O ne opublikovannoy P.I. Tchaikovskim pervoy redaktsii "Detskogo alboma" (On an unpublished first edition of the "Children's album" by P.I. Tchaikovsky)," in Voprosy muzykalnoy pedagogiki (Issues of Musical Pedagogy). Muzyka, 1997, pp. 138–161, (In Russian).

[29] P. Georges, "Western classical music development: a statistical analysis of composers similarity, differentiation and evolution," Scientometrics, vol. 112, no. 1, 2017, pp. 21–53.

[30] P. Georges and N. Nguyen, "Visualizing music similarity: clustering and mapping 500 classical music composers," Scientometrics, vol. 120, no. 3, 2019, pp. 975–1003.

[31] "Spectrum analyzer," retrieved: Aug, 2022. [Online]. Available: https://academo.org/demos/spectrum-analyzer/

 ISBN: 978-1-61208-990-4

On the Proportional-Integral-Derivative Based Trading Algorithm under the Condition of the log-Normal Distribution of Stock Market Data

Vadim Azhmyakov
National Research University Higher School of Economics
Moscow, Russian Federation
email: v.azhmyakov@hse.ru

Ilya Shirokov
Algorithmic Systems Corp
Moscow, Russian Federation
email: iyushirokov@gmail.com

Yuri Dernov
Algorithmic Systems Corp
Moscow, Russian Federation
email: dernovyua82@gmail.com

Luz Adriana Guzman Trujillo
LARIS, Université d'Angers
Angers, France
email: luz.guzmantrujillo@etud.univ-angers.fr

***Abstract*—Our paper deals with a novel trading algorithm based on the conventional feedback control methodology. A profitable trading algorithm design for stock markets constitutes a very challenging problem of the modern financial engineering. We apply a model-free version of the classic Proportional-Integral-Derivative (PID) control to the modern Algorithmic Trading (AT). The proposed control theoretical application of the classic PID methodology is combined with a specific statistical information on the available historical stock market data. We consider a generic condition of the log-normal distribution of the available stock data. The log-normal property mentioned above implies a new efficient calibration rule for the gain coefficients (gains tuning) for the resulting PID type trading algorithm. We finally apply the developed PID based optimal AT strategy to a specific real-world example of the Binance Bitcoin / USD market. This application illustrates the effectiveness of the proposed trading algorithm.**

***Index Terms*—algorithmic trading, financial engineering, model-free PID control, statistical decision making.**

I. Introduction

Consider an idealized stock market model in discrete time. This trading abstraction includes some ideal assumptions, among others, the "no transaction costs" and "one stock portfolio" conditions. Moreover, one also assumes "zero interest" "continuous trading" and some further simplifying hypothesises. We refer to [4][14][16] for the necessary technical details. The discrete time model consideration is mainly motivated by the real stock market dynamics as well as by the decision making mechanism. We next introduce the trading ticks $t = 1,...,$ and the corresponding time-intervals of the trading buckets $[t, t+1)$. Note that the development of the efficient and robust trading algorithms for the financial markets constitutes a sophisticated problem. Recall that one deals with a stochastic dynamic behaviour in that case. The highly frequent non-regular stochastic nature of the modern markets makes it impossible any suitable forecasting of the prices of financial instruments traded on the stock markets.

A systematic, control theory based approach to the AT was initially developed in [5]-[8] [20]. It studies the model-free PID trading strategy and proposes to react to the stock price variations instead of modeling them. An interesting, frequency domain involved extension of the above approach can be found in [14]. Let us also refer to [4] for a novel PID related trading algorithm with a switched structure.

Following [20], we introduce the current gain $\Delta g(t)$ and the current investment level $\Delta I(t)$ for a time instant t. Moreover, by $g(t)$ and $I(t)$ we next denote the cumulative profit and the cumulative investment, respectively. The initial investment level I_1 is assumed to be given. Consider the nonlinear discrete-time PID type feedback with a saturation rule:

$$\begin{aligned} &\delta I(t+1) = K_P(t)\Delta g(t) + K_D(t)\dot{\Delta} g(t) + \\ &K_I(t)\int_{t-T}^{t} h(\tau)\Delta g(\tau)d\tau, \\ &\Delta I(t+1) = \chi(\delta I(t+1)), \quad \text{for} \quad t = 1,..., \\ &I(1) = I_1. \end{aligned} \tag{1}$$

Here $K_P(\cdot)$, $K_D(\cdot)$ and $K_I(\cdot)$ are dynamic gains associated with the proportional, integral, derivative, and second order derivative terms of regulator (1). We put

$$K(\cdot) := \{K_P(\cdot),\ K_D(\cdot),\ K_I(\cdot)\}.$$

Similar to the classic PID control (see e.g., [21]), the integral term in (1) is defined on a given time interval $[t-T, t]$. The time instant T belongs to the given discrete time grid. Moreover, $h(\cdot)$ in (1) is a suitable "memory loss" function. One can consider the generic exponentially weighted "memory

 ISBN: 978-1-61208-990-4

loss" function $h(\cdot)$ with $h(t)=1$. The saturation function $\chi(\cdot)$ in 1 can be defined as follows

$$\chi(\delta I) := \begin{cases} \delta I, & \text{if } \delta I^{\min} \le |\delta I| \le \delta I^{\max}; \\ \pm\delta I^{\max}, & \text{if } |\delta I| > \delta I^{\max}; \\ 0, & \text{if } |\delta I| < \delta I^{\min}. \end{cases} \quad (2)$$

where $\delta I^{\max}$ and $\delta I^{\min}$ are prescribed maximal and minimal current investment levels, respectively.

Using the decision about the current investment $\Delta I(t+1)$ and the corresponding stock price $p(\omega,t+1)$, we next calculate the current profit obtained at the time instant $(t+1)$:

$$\Delta g(t+1) = \frac{(p(\cdot,t+1)-p(\cdot,t))}{p(\cdot,t)}\Delta I(t+1) \quad (3)$$

Note that the investment level $\Delta I(t+1)$ in (1) constitutes a "control input". We next call it "investment decision". Note that it is deployed at a current time instant t under the natural unknownness of the market price $p(\omega,t+1)$. Here, $\omega \in \Omega$ and Ω is a probability state space with a specific probability measure. The stock price $p : \Omega \times \mathbb{Z}_+ \to \mathbb{R}$ is assumed to be a measurable (stochastic) function. Note that the current profit $g(t+1)$ is an a posteriori value such that $p(\cdot,t+1)$ in (3) denotes a concrete realization of a stochastic price $p(\omega,t+1)$. The block diagram of the proposed model-free PID trading strategy is illustrated in Figure 1.

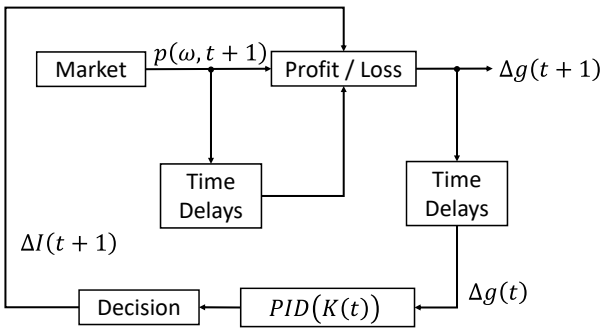


Fig. 1. Model-free PID based trading algorithm

The general formula for $\delta I(t+1)$ in (1) can easily be specified in the discrete-time case:

$$\begin{aligned} &\delta I(t+1) = (K_P(t)+K_I(t)+K_D(t))\Delta g(t)+ \\ &(K_I(t)h(t-1)-K_D(t))\Delta g(t-1)+ \\ &K_I(t)\sum_{\tau=t-T}^{t-2} h(\tau)\Delta g(\tau). \end{aligned} \quad (4)$$

The main problem in the conventional PID control theory as well as in the model-free version under consideration constitutes in searching of adequate gains $K(\cdot)$ tuning rules (see e.g., [19] and references therein). In the conventional application areas of the classic model-based PID controllers these tuning techniques are usually well established [19][21]. In the sophisticated model-free stochastic case, the design of a suitable PID tuning scheme represents a challenging problem. It constitutes in fact a key problem of an intelligent investment decision.

The remainder of this paper is organized as follows: Section 2 contains a general statistical analysis of a generic stock data. In Section 3, we apply the obtained log-normal distribution of the available data to backtesting driven tuning (calibration) of the PID gains. In this section, we also consider an application of the proposed PID based trading algorithm to a specific stock market, namely, to the Binance BTC (bitcoin) / USD futures. Section 4 summarizes our paper.

II. Statistical Analysis of the Stock Data

The conceptually important PID gains tuning problem mentioned in Section I will be considered here from the statistical point of view. For these aims let us examine the log-normal hypothesis for the probability distribution of the following "price/volume" ratio:

$$\theta(\omega,t+1) := \frac{p(\omega,t+1)}{v(t+1)}.$$

The (a posteriori) statistical analysis of a wide spectrum of stock markets has demonstrated that the distribution of the closing prices normalized by investment volume $v(t+1)$, namely the value $\theta(\omega,t)$ fits well a specific log-normal law (see [1] and references therein). Recall that $v(t+1)$ can be calculated as follows:

$$v(t+1) := \frac{\Delta I(t+1)}{p(\cdot,t)}.$$

Note that in the trader praxis the investment volume is usually restricted by a maximal investment volume. In the case of PID based trading algorithm under consideration, the maximal investment volume is a direct consequence of the bounded structure of investment $\Delta I(t+1)$ in (1).

The log-normal probability distributions in the stock market data have comprehensively been studied for the stock price differences and for option prices. Let us refer to the celebrated Black and Scholes model [12]. A full review of this subject can be found in [13]. Additionally, the log-normal properties of the volatility and related market values in the stock prices and indices are also considered in [15]. In this section, we follow [1] and analyze the (stationary) statistical law for $\theta(\omega,t+1)$:

$$\rho(\theta) = \frac{a}{\sqrt{2\pi}\sigma(\theta-s)}\exp -(0.5\sigma^2)(\ln(\theta-s)-\mu)^2, \quad (5)$$

where $\mu \in \mathbb{R}$ is a mean, $\sigma \in \mathbb{R}_+$ is a dispersion and $s \in \mathbb{R}$ denotes a shifting parameter. Note that (5) is a so called "three-parameters" $\{\mu,\sigma,s\}$ log-normal distribution (see e.g., [22]). The necessary parameters of the proposed log-normal distribution $\rho(\theta)$ can be determined using the historical stock market data. This way, we incorporate the generic backtesting into the resulting PID based trading algorithm we develop.

As mentioned above, (5) constitutes an adequate distribution hypothesis for the price/volume ratio $\theta(\omega,t+1)$. The quality

of this statistic hypothesis can be established by the standard Chi-Quadrat-Test for distributions (see e.g., [22]). In this paper, we consider the normalized value χ^2/q for this purpose. Here q is the number of degrees of freedom.

Consider now the daily market prices and investment volumes data

$$\{p(1),...,p(T)\},\ \{v(1),...,v(T)\}.$$

Following [24], we now assume the log-normal distribution for the value

$$\Big(\frac{p(\omega,t+1)}{p(\cdot,t)}\Big),$$

where $t=1,...,T-1$. Note that this assumption or the equivalent assumption on the normal distribution of the logarithmic return

$$\ln\Big(\frac{p(\omega,t+1)}{p(\cdot,t)}\Big)$$

is well motivated. We refer to [1][24] for the necessary statistical consideration. We now consider the following value:

$$\ln\big(\theta(\omega,t+1)/\theta(\cdot,t)\big)=\ln\Big(\frac{p(\omega,t+1)}{p(\cdot,t)}\Big)- \ln\Big(\frac{v(t+1)}{v(t)}\Big). \tag{6}$$

From (6) we next obtain

$$\ln\Big(\frac{v(t+1)}{v(t)}\Big)=\ln\Big(\frac{p(\omega,t+1)}{p(\cdot,t)}\Big)+\ln\theta(\cdot,t)- \ln\theta(\omega,t+1). \tag{7}$$

Expression (7) makes it possible to make a decision related to the investment volume $v(t+1)$. We evidently have

$$v(t+1)=\exp\Big[\ln\Big(\frac{p(\omega,t+1)}{p(\cdot,t)}\Big)+\ln\theta(\cdot,t)- \ln\theta(\omega,t+1)+\ln v(t)\Big]. \tag{8}$$

Considering (8), we observe that $\theta(\cdot,t)$ and $v(t)$ and the corresponding logarithms are known values. Moreover, the price/volume ration $\theta(\omega,t+1)$ can be simulated using the above log-normal distribution (5). As mentioned above, we can also forecast the value $\ln\big(p(\omega,t+1)/p(\cdot,t)\big)$ using a suitable log-normal distribution (see [24]). Note that the necessary parameters of the log-normal distributions for values $\theta(\omega,t+1)$ and $p(\omega,t+1)/p(\cdot,t)$ needed to be determined using the available historical data.

III. Application of the log-Normal Distribution to the PID Gains Calibration

The main problem of interest for many researchers working in the financial engineering is to anticipate the behavior of stock markets giving a certain amount of historical data. For this purpose we study the PID type trading algorithm (1) in combination with the statistical log-normal characteristics discussed in Section II. We now apply formula (8) and obtain $M\in\mathbb{N}$ simulations $v^j(t+1),\ j=1,...,M.$ of the expected investment value at $(t+1)$. Using the given algebraic structure of the PID algorithm (1), we next calculate the necessary (three-dimensional) gain coefficient $K(\cdot)$ as a solution of the following optimization problem

$$\begin{aligned}&\sum_{j=1}^{M}\big(\chi(\delta I(t+1))-v^j(t+1)p(\cdot,t)\big)^2\to\min\\ &\delta I(t+1)=K_P(t)\Delta g(t)+K_D(t)\dot{\Delta}g(t)+\\ &K_I(t)\int_{t-T}^{t}h(\tau)\Delta g(\tau)d\tau,\end{aligned} \tag{9}$$

where the nonlinear function $\chi(\cdot)$ is given by (2). Evidently, (9) constitutes a specific nonlinear regression. This nonlinear optimization problem finally leads to some optimal gains

$$K^{opt}(\cdot):=\{K_P^{opt}(\cdot),\ K_D^{opt}(\cdot),\ K_I^{opt}(\cdot)\}.$$

for the PID type trading algorithm (1). Finally, we use $K^{opt}(\cdot)$ and define the deployed (optimal) investment level $\Delta I^{opt}(t+1)$ by expressions (1)-(2). The really deployed investment volume $v^{opt}(t+1)$ can be calculated as follows:

$$v^{opt}(t+1):=\frac{\Delta I^{opt}(t+1)}{p(\cdot,t)}.$$

Note that the trading decision at the future time instant $(t+1)$ can be expressed by the current optimal investment level $\Delta I^{opt}(t+1)$ calculated above. It also can be determined by the pair $\{\text{sign}[\Delta I^{opt}(t+1)],\ \ |v^{opt}(t+1)|\}$, namely, by the current trade direction $\text{sign}[\Delta I^{opt}(t+1)]$ and by the associated absolute value $|v^{opt}(t+1)|$ of the investment volume. The resulting PID based strategy (1) combined with the optimal gain selection (9) is next called Optimal PID (OPID) trading algorithm.

We now present an application of the developed PID based AT technique to a real-world stock market data. Consider the Binance Bitcoin / USD futures and apply the proposed AT technique. The BTC futures price dynamics is presented in Figure 2.

Fig. 2. Binance BTC / USD one day price index

We have applied the novel OPID trading algorithm to the above example. The corresponding profit dynamics is presented in Figure 3.

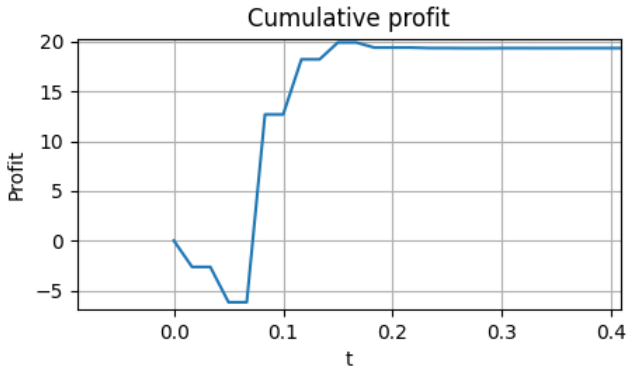


Fig. 3. Binance BTC / USD one day price index

Note that the operation time of the OPID in that example is timely restricted. The stock market under consideration has an obvious high-frequency behaviour. This fact naturally implies some (expected) difficulties of the common trading algorithms. In particular, this concerns the widely used moving average based trading strategies.

As one can see, the developed OPID constitutes a High-Frequency Trading (HFT) strategy. The time ticks in the above trading example under consideration are very dense. The length of the corresponding intervals of trading buckets is equal to 1.728 sec. Let us also note that the typical "hyperregulation and stabilization" dynamics of the proposed PID based strategy is visible (see Figure 3). The presented practical example illustrates the implementability of the developed PID involved AT scheme.

IV. Conclusion and Future Work

In this paper, we developed a combined trading algorithm that involves the PID control methodology and some a priori given statistical characteristics of the stock market data. We studied an idealized market under the assumption of only one asset. However, the proposed analytic approach and the resulting trading methodology can easily be extended to the real-time multi-asset trading. Moreover, it can incorporate some additional efficient data driven optimization techniques.

The given historical stock market data and the statistical properties mentioned above are constructively used for an adequate calibration (tuning) of the PID controller gains. This calibration is based on the backtesting procedure. The resulting optimal trading strategy generates adequate (profitable) decisions of the "buy/sell/hold" market orders at every subsequent time instant. The proposed approach to the AT involves a combination of two mathematically rigorous tools, namely, the classic PID control methodology and applied statistics. It also contains a specific data driven optimization procedure. This combination finally leads to a novel and very promisingly trading strategy. The resulting algorithm and the corresponding real-world scenario based simulation technique conceptually extend the family of the feedback based trading algorithms.

Note that the developed PID based trading approach can be used as an additional tool in the several theoretic frameworks of the modern financial engineering. For example, it can be applied in combination with the well established financial time series analytics. The proposed PID based trading algorithm is also compatible with the generic price prediction techniques (see e.g., [17]).

Finally note that the algorithmic trading strategy proposed in our paper constitutes an initial (conceptual) development. We are mostly concentrated on the formal algorithmic aspects of the proposed technique. The financial solutions for the stock market considered in our contribution need additional comprehensive simulations and prototyping, further backtesting, adequate data driven optimization and applications to the real markets. We also expect a profitable application of the proposed trading methodology specifically in the High-Frequency Trading.

References

[1] I. Antonioua, V. V. Ivanova, V. V. Ivanovb and P. V. Zrelova, On the log-normal distribution of stock market data, Physica A, vol. 331, 2004, pp. 617 – 638.

[2] V. Azhmyakov, A Relaxation Based Approach to Optimal Control of Switched Systems, Elsevier, Oxford, UK, 2019.

[3] V. Azhmyakov, J. Pereira Arango, M. Bonilla, R. Juarez del Torro and St. Pickl, "Robust state estimations in controlled ARMA processes with the non-Gaussian noises: applications to the delayed dynamics", IFAC PapersOnline, vol. 54, 2021, pp. 334 – 339.

[4] V. Azhmyakov, I. Shirokov and L. A. Guzman Trujillo, "Application of a switched PIDD control strategy to the model-free algorithmic trading", IFAC PapersOnline, to appear in 2022.

[5] B. R. Barmish, "On trading of equities: a robust control paradigm", IFAC Proceedings Volumes, vol. 41, 2008, pp. 1621 –– 1626.

[6] B. R. Barmish and J. A. Primbs, "On market-neutral stock trading arbitrage via linear feedback", in: Proceedings of the American Control Conference, Montreal, Canada, 2012, pp. 3693 – 3698.

[7] B. R. Barmish and J. A. Primbs, "On a new paradigm for stock trading via a model-free feedback controller", IEEE Transactions on Automatic Control, vol. 61, 2016, pp. 662 -- 676.

[8] M. H. Baumann, "On stock trading via feedback control when underlying stock returns are discontinuous", IEEE Transactions on Automatic Control, vol. 62, 2017, pp. 2987 –– 2992.

[9] A. Bemporad, T. Gabbriellini, L. Puglia and L. Bellucci, "Scenario-based stochastic model predictive control for dynamic option hedging", in: Proceedings of the IEEE Conference on Decision and Control, Atlanta, USA, 2010, pp. 3216 –– 3221.

[10] D. Bertsekas, Reinforcement Learning and Optimal Control, Athena Scientific, Nashua, USA, 2019.

[11] D. Bertsimas and A. W. Lo, "Optimal control of execution costs", Journal of Financial Markets, vol. 1, 1998, pp. 1 –– 50.

[12] F. Black and M. Scholes, The pricing of options and corporate liabilities, Journal of Political Economy, vol. 81, 1973, pp. 637 — 659.

[13] E. L. Crow and K. Shimizu (Eds.), Lognormal Distributions, Theory and Applications, Marcel Dekker, Inc., New York, 1988.

[14] S. Formentin, F. Previdi, G. Maroni and C. Cantaro, "Stock trading via feedback control: an extremum seeking approach,

in: Proceedings of the Mediterranean Conference on Control and Automation, Zadar, Croatia, 2018, pp. 523 – 528.
[15] C. Hammel and W. B. Paul, Monte Carlo simulations of a trader-based market model, Physica A, vol. 313, 2002, pp. 640 – 650.
[16] T. Hens and M. O. Rieger, Financial Economics, Springer, Berlin, Germany, 2010.
[17] S. Huang, "Online option price forecasting by using unscented Kalman filters and support vector machines", Journal of Expert Systems with Applications, vol. 34 , 2008, pp. 2819 -– 2825.
[18] St. Jansen, Machine Learning for Algorithmic Trading, Packt, Birmingham, UK, 2020.
[19] H. K. Khalil, Nonlinear Control, Pearson, Boston, USA, 2015.
[20] S. Malekpour, J. A. Primbs and B. R. Barmish, "On stock trading using a PI controller in an idealized market: the robust positive expectation property", in: Proceedings of the IEEE Conference on Decision and Control, Florence, Italy, 2013, pp. 1210 – 1216.
[21] A. Poznyak, Advanced Mathematical Tools for Automatic Control Engineers: Deterministic Technique, Elsevier, NY, USA, 2008.
[22] A. Poznyak, Advanced Mathematical Tools for Automatic Control Engineers: Stochastic Tools, Elsevier, NY, USA, 2009.
[23] S. Taylor, Modeling Financial Time Series, Wiley, Chichester, UK, 1986.
[24] N. Vo and R. Slepaczuk, Applying hybrid ARIMA-SGARCH in algorithmic investment strategies on S&P500 index, Entropy, vol. 24, 2022.

 ISBN: 978-1-61208-990-4

Identification of Critical Nodes and Links in a Supply Chain by Robust Optimization

Tim vor der Brück
School of Computer Science and Information Technology
Lucerne University of Applied Sciences and Arts
Lucerne, Switzerland
email: tim.vorderbrueck@hslu.ch

René Meier
School of Computer Science and Information Technology
Lucerne University of Applied Sciences and Arts
Lucerne, Switzerland
email: rene.meier@hslu.ch

***Abstract*—The impact of a supplier or transportation link breakdown in a supply chain can strongly differ depending on which nodes/links are affected. While the breakdown of producers of rarely needed products or backup suppliers might result in no or only minor repercussions, the breakdown of central suppliers or transportation links, also called critical nodes/links, can be fatal and may cause a severe delivery delay or even a complete production failure of certain product lines. Therefore, it is of high importance for a company to identify its critical nodes/links in the supply chain and take precautionary actions such as organizing additional backup suppliers or alternative ways of transportation. In this paper, we describe a novel method to identify critical nodes and links in a supply chain based on robust optimization, which has the advantage that supply chain risks are considered, and also precise risk cost estimates regarding the possible breakdown of each supplier node are provided. Finally, we demonstrate this method on an example supply chain and discuss its distribution of critical nodes and links.**

***Keywords*—supply chain management; critical nodes; critical links; robust optimization; supply chain risks.**

I. Introduction

According to Craighead et al.[1], node criticality is defined as the relative importance of a given node or set of nodes within a supply chain (see Figure 1). A breakdown of a critical node has typically severe implications, such as serious delay or even a complete collapse of the production process for certain product lines, which can result in non-fulfillment of customer demand. Consequently, the affected company suffers lost revenue and faces a potential non-delivery contract penalty. Thus, it is of great importance to identify the critical nodes in the supply chain and mitigate their possible breakdown risks by implementing precautionary measures such as identifying backup suppliers.

The concept of critical nodes can also be transferred to important transportation links. A link in a supply chain denotes a certain transport mode (e.g., airplane, truck, or ship transportation) and a route between two suppliers or between a supplier and a customer. Analog to the definition of critical nodes, a critical link denotes a link that is of high importance for the total supply chain. Critical links should therefore be secured by identifying alternative means of transportation.

The rest of the paper is structured as follows. Related work is given in the upcoming section (Section II). The employed optimization model is given in Section III. In Section IV, we describe how the node criticality is assessed and discuss the obtained results. Finally, we conclude the paper with Section V where we summarize our contribution and give potential future work.

II. Related work

Zhang and Han [3] propose to use network centrality (especially degree and betweenness centrality) as indicators for the criticality of a node in a supply chain.

Gaura et al. [4] assess the criticality of a certain network node by determining the decrease in network efficiency when this node is removed from the network. The network efficiency is measured by the normalized sum of the reciprocal of graph distances between any two nodes in the network. Prior to applying their approach, nodes with low clustering indices are removed from the network, wherefore the authors termed their approach clustering-based.

The approaches described so far assess a node criticality alone by topological network measures. In contrast, Falasca et al. [2] propose to also consider throughput through the network, but fail to suggest a concrete measure. Sebouhi et al. [5] consider a node as critical, if the throughput through this node as determined by solving a linear optimization problem, exceeds a certain predefined threshold. However, this measure does not take into account the use of backup suppliers as we do here, which can de facto reduce node criticality of alternative suppliers.

There are also some existing approaches to identify critical links. Scott et al. [6] introduce the so-called Network Robustness Index (NRI), "for evaluating the critical importance of a given highway segment (i.e., network link) to the overall system as the change in travel-time cost associated with rerouting all traffic in the system should that segment become unusable." Note that the NRI only takes into account costs that are directly transportation-related but disregards repercussions of item non-delivery for downstream production processes as we considered in our proposed method.

III. Employed optimization model

Our approach is based on robust optimization, which itself is based on stochastic optimization, which again is based on a deterministic optimization model.

We describe each of these three models subsequently in the following sections starting with the most basic one.

 ISBN: 978-1-61208-990-4

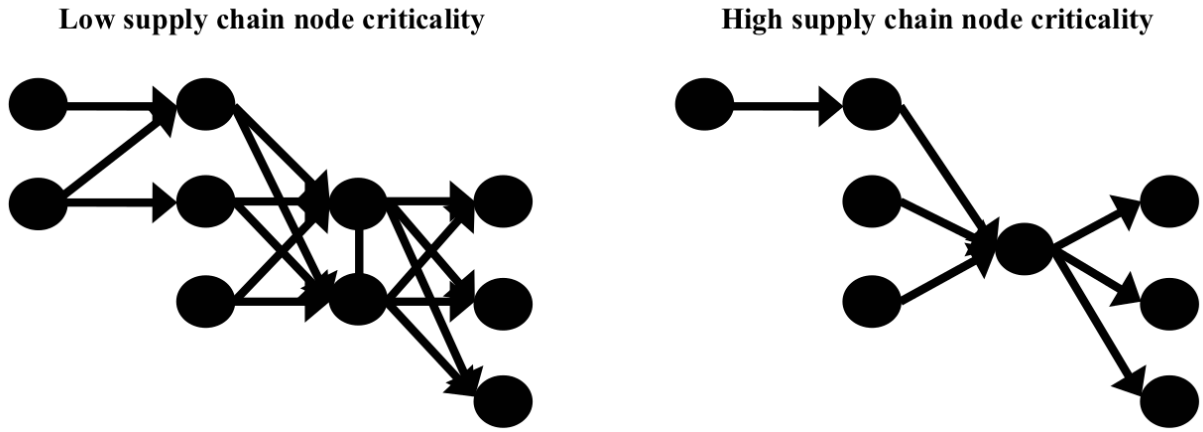


Fig. 1. Supply chain with (right) and without a critical node (left) [2].

A. Deterministic optimization model

The deterministic model disregards any potential risk for the supply chain and determines the minimum costs of the so-called "happy flow", which denotes the best-case situation that no supply chain disruption occurs. Since such a model contains no stochastic part, it can be computed very efficiently. Note that we use due to the computational complexity of the stochastic and robust model, for all of our 3 optimization models a single period of 12 months, over which we aggregate the total customer demand.

The following constants must be specified beforehand:

- d_{jz}: demand at node j for product z
- c_{ij}: cost to move one kg over one km from i to j
- pc_{iz}: cost to produce one item of product z at supplier i
- a_{xz}: number of items of product x to produce one amount of product z
- cap_{iz}: production capacity of product z on node i
- in_{iz}: initial number of items of product z contained in the inventory of supplier i
- ic_{iz}: inventory cost for storing z at location i
- $dist_{ij}$: geographical distance between node i and j
- $weight_z$: weight of product z

The following decision variables are to be determined by the optimizer:

- T_{ijz}: number of items z that are moved from location i to j
- IT_{il}: internal transfer of item l from inventory at location i
- P_{iz}: number of items z produced at supplier i
- WT_{iz}: number of items z removed from the warehouse of supplier i

Model constraints:

- $d_{jz} \leq \sum_i T_{ijz}$: demand of item z at location j is met
- $\sum_z a_{lz} P_{iz} = P_{il} + IT_{il} + \sum_k T_{kil}$: number of items l required to build items z at location i
- $P_{iz} \leq cap_{iz}$: supplier at node i can at most produce cap_{iz} items for product z
- $P_{iz} + WT_{iz} \geq \sum_j T_{ijz}$: produced + removed from the inventory of supplier $i \geq$ number of items transported from supplier i
- $IT_{iz} + WT_{iz} \leq in_{iz}$ for each item z and supplier i: inventory contents cannot become negative

The following objective is used:
Minimize $costs_{total}$ with:

$$costs_{total} := \sum_{ijz} T_{ijz} c_{ijz} dist_{ij} weight_z + \sum_{iz} (P_{iz} pc_{iz} + in_{iz} ic_{iz}) \quad (1)$$

B. Stochastic optimization model

The stochastic model takes supply chain risks into account and computes the expected value of the supply chain costs ($\mathbb{E}(C)$) determined over all generated risk scenarios. In a stochastic optimization setting, the set of risk scenarios describes the potential hazards for the whole supply chain. Hence, the nine scenarios from our case company's supply network are used as input for the stochastic optimization approach, which are given in Table I.

The stochastic optimization model determines the minimal supply chain costs under these risks and estimates the supply network resilience of the entire supply chain. Note that certain inventory costs as well as production surges are currently still disregarded in our model but may be considered for future work. We have expanded our initial deterministic optimization model as follows. First, each decision variable is assigned an additional index denoting the associated risk scenario. For instance: P_{izs} denotes the number of item z produced at location i in risk scenario s. Furthermore, an additional decision variable named $Missed_{jzs}$ has been included to denote the shortfall of a produced item z at location j for risk scenario s with respect to the actual demand. To represent the effect of a

TABLE I
SUPPLY CHAIN DISRUPTION RISK SCENARIOS FOR OUR EXAMPLE SUPPLY CHAIN.

Number	Risk Scenario
1	Product line simplification of supplier 1 - supplier no longer delivers the component due to strategy change
2	Product line simplification of supplier 2 - supplier no longer delivers the component due to strategy change
3	Covid19 pandamic
4	Cyber attack
5	Transport disruption
6	Supplier disruption due to export restrictions
7	Delivery problems of a certain part from supplier 3
8	Delivery problems of a certain part from supplier 4
9	Happy Flow - no disruptions

missed demand, we define a variable (per item) non-delivery penalty term pen_{jz}. The penalty is invoked when the demand for item j and location z cannot be met ($Missed_{jzs} > 0$). The non-delivery penalty comprises lost revenue and a possible contract penalty. As a result, the demand constraint changes as follows: $d_{jz} \leq Missed_{jzs} + \sum_i P_{izs} T_{ijzs}$ for every scenario s and the objective function becomes:

$$\begin{aligned} &\text{Minimize } \mathbb{E}(C) + \sum_{jzs} pen_{jz} Missed_{jzs} \\ &\text{(Missed demand is penalized.)} \\ &\mathbb{E}(C) := \sum_s p_s C_s = \sum_{ijzs} p_s T_{ijzs} c_{ijz} dist_{ij} weight_z \\ &\quad + \sum_{izs} p_s (P_{izs} pc_{iz} + in_{iz} ic_{iz}) \end{aligned} \tag{2}$$

where C_s denotes the total supply chain cost and p_s specifies the probability of occurrence of risk scenario s. We use this cost estimate as objective in the optimization problem.

C. Robust optimization model

The robust model introduces an additional constant σ that specifies the risk affinity of the decision-maker [7] [8]. Large values of σ cause a considerable increase in risk costs accounting for the unsureness about the actual costs. Thus, a risk-averse decision-maker would select a rather high σ, whereas a risk-tolerant decision-maker would select a small value or drop this term altogether. Thus, the objective function changes to:

$$\text{Minimize } \mathbb{E}(C) + \sigma \mathbb{V}(C) + \sum_{jzs} (pen_{jz} Missed_{jzs}) \tag{3}$$

Since the computation of the variance requires quadratic programming, we decided to approximate it by the absolute variance [7] [9]:

$$\mathbb{V}_{abs}(C) := \sum_s p_s |C_s - \mathbb{E}(C)| \tag{4}$$

The absolute variance can be modeled by linear programming as follows. First, we introduce additional non-negative decision variables : $\phi(s)^+$ und $\phi(s)^-$ with the following two constraints

$$\begin{aligned} \phi_s^+ &\geq p_s (C_s - \mathbb{E}(C)) \\ \phi_s^- &\geq p_s (\mathbb{E}(C) - C_s) \end{aligned} \tag{5}$$

The objective function is then given by:

$$\text{Min. } \mathbb{E}(C) + \sum_s \sigma (\phi_s^+ + \phi_s^-) + \sum_{jzs} (pen_{jz} Missed_{jzs}) \tag{6}$$

ϕ_s^+ captures the part of the variance, where the costs exceed their expected value, whereas ϕ_s^+ captures the remaining part, where the costs fall below their expected value. It can be shown that for the absolute variance, both parts must coincide. Thus:

$$\phi_s := \phi_s^+ = \phi_s^- \tag{7}$$

With this, the constraints in (5) simplify to [9]:

$$\phi_s \geq p_s (C_s - \mathbb{E}(C)) \tag{8}$$

and the objective function changes to

$$Minimize \ \mathbb{E}(C) + \sum_s \sigma \cdot 2\phi_s + \sum_{jzs} (pen_{jz} Missed_{jzs}) \tag{9}$$

IV. ASSESSING NODE CRITICALITY

Thus far, we have explained our robust optimization model, which is the basis for our proposed node criticality assessment. In particular, the robust optimization method as described above estimates the supply chain‘s risk costs that are composed of the expected total supply chain costs considering several disruption risk scenarios and their variance. A large variance implies that the supply chain costs can vary strongly depending on the occurred risk scenarios. In this case, there is high uncertainty about the incurring costs and therefore the overall supply chain risk is quite high. In contrast, low variance means that the supply chain costs do not deviate much across the scenarios. In this case, the overall supply chain risk remains small. The risk costs are leveraged in our approach for identifying the critical nodes of the supply chain.

By using risk costs instead of ordinary deterministic costs, we obtain more accurate criticality assessments of the nodes. Consider for example the case, that an important supplier S is backed up by a second supplier, which is threatened by probable bankruptcy. In a deterministic setup, the supplier S would be assigned a low criticality because of the provided backup supplier. However, in case supply chain risks are considered, the criticality of supplier S remains high due to the foreseeable default of the backup supplier.

In our approach, a supplier node is considered critical, if its complete breakdown causes a high increase in risk costs of the supply chain, which can be estimated by our robust optimization approach. In contrast, a node is considered uncritical, if the total risk costs of the supply chain do not change in case the associated supplier breaks down and can no longer produce or deliver any goods. Therefore, we consider the criticality of a node being proportional to the overall risk

 ISBN: 978-1-61208-990-4

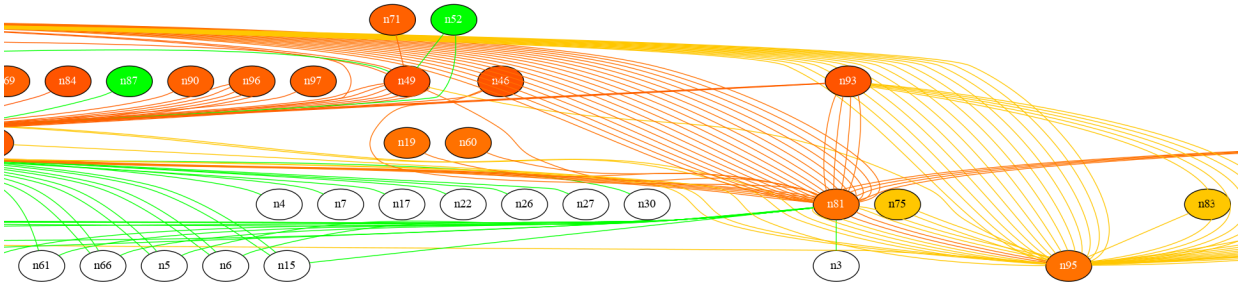


Fig. 2. Part of our example supply chain, where supplier nodes and transportation links are colored according to their criticality.

costs increase of the supply chain when the node in question is removed.

A node in the supply chain network can represent either a supplier or a customer, while the edges represent transportation links either between two suppliers or between a supplier and a customer. We consider in the following an example supply chain with 40 customers, 80 suppliers, 200 components and products, 200 transportation links, and 400 product demands. Due to its size, we only depict a part of the total supply chain in Figure 2, which has similar characteristics in terms of critical links and nodes as the total supply chain.

Each supplier node in this network is colorized according to its criticality. Suppliers are colored green if the risk costs of the supply chain are not increased by its potential breakdown, they are colored yellow if the supply chain risk costs are increased by a certain threshold factor f_1 (we use 30%), and red if the costs were increased by a second larger threshold factor f_2 (we use 60%) or more. Note that the exact values of factors f_1 and f_2 can vary depending on the corporate branch and the degree of competition. For costs increases between 0 and f_1, we interpolate the RGB color values linearly between green (red=0, green=255, blue=0) and yellow (red=255, green=255, blue=0), for costs increased between f_1 and f_2, we interpolate the color values between yellow and red (red=255, green=0, blue=0). Customers are not associated with any production risks and therefore their associated graph nodes are not colored and instead visualized by unfilled circles. The entire process is illustrated in the form of pseudocode in Figure 3.

Like critical nodes, we also visualize critical links in the supply chain. Analog to the node case, they are colored in green if uncritical, in yellow if somewhat critical, and in red if critical. Again, mixtures of the colors red and yellow as well as green and red are possible. In case there are several transportation modes available between two connected nodes, we consider only the most critical mode for the visualization. Note that a link originating from an uncritical supplier node must also be uncritical. However, the opposite does not hold. A link originating from a critical supplier node, can be considered uncritical, if alternative (backup) transportation modes are available.

The most critical node in our example supply chain would increase the risk costs by 50% in case of failure. Furthermore, by far the largest part of the suppliers is considered rather

1: **procedure** GET_RISK_COSTS_COLOR($nodes, costs_{hf}$)
2: Input $nodes$: list of total supply chain nodes
3: Input $costs_{hf}$: "happy flow" costs
4: $red := (255, 0, 0)$
5: $green := (0, 255, 0)$
6: $yellow := (255, 255, 0)$
7: $hm := \{\}$ # associated risk costs of a node
8: $hm\_color := \{\}$ # associated RGB values for a node
9: **for** $n \in nodes$ **do**
10: **if** type(n)==Supplier **then**
11: $costs := obj\_value(opt(nodes \backslash \{n\}))$
12: $hm[n] := costs$
13: **if** $costs < (1 + f_1)costs_{hf}$ **then**
14: $w := (costs - costs_{hf})/(f_1 \cdot costs_{hf})$
15: $hm\_color[n] := w \cdot yellow + (1-w) \cdot green$
16: **else if** $costs < (1 + f_2)costs_{hf}$ **then**
17: $df := f_2 - f_1$
18: $dcosts := costs - (1 + f_1)costs_{hf}$
19: $w := dcosts/(df \cdot costs_{hf})$
20: $hm\_color[n] := w \cdot red + (1 - w) \cdot yellow$
21: **else** $hm\_color[n] := red$
22: **end if**
23: **end if**
24: **end for**
25: return hm, hm_color
26: **end procedure**

Fig. 3. Identification of risk costs and node criticality for all suppliers.

critical by our chosen definition of f_2, which is caused by the fact that backup suppliers are missing in most cases. The remaining suppliers are to the same part either non-critical (visualized in green) or somewhat critical (visualized in yellow). In contrast, the distribution of links is much more balanced. Almost 56% of the links are regarded as critical, the rest is either somewhat critical or uncritical. In particular, transportation links leading to a customer are all considered uncritical due to existing alternative transportation modes, while most of the inter-supplier links are critical. Optimally, the decision-maker should supply backup suppliers / transportation modes for all critical nodes and links so that all critical nodes / links become somewhat critical or uncritical.

 ISBN: 978-1-61208-990-4

V. Conclusion

We described a method for identifying critical nodes in a supply chain based on robust optimization. In contrast to other state-of-the-art methods, our method is very precise, since it not only considers network topology but also network throughput as well as possible supply chain disruption risks. Furthermore, our method provides a concrete risk costs estimate for the breakdown of each supplier and transportation link. For future work, we are planning to identify critical supplier groups, i.e., collections of suppliers being located in geographical or political neighborhoods (and therefore vulnerable to similar supply chain risks) that are causing major disruptions to the supply chain if they are failing simultaneously.

Acknowledgment

Hereby we thank all who supported us in this work, especially Gunter Krell for a lot of fruitful discussions and Uta Jüttner who provided us with fitting risk scenarios.

References

[1] C. W. Craighead, J. Blackhurst, M. J. Runtusanatham, and R. B. Handfield, "The severity of supply chain disruptions: Design characteristics and mitigation capabilities," *Decision Sciences*, vol. 38, no. 1, pp. 131–156, 2007.

[2] M. Falasca and C. W. Zobel, "A decision support framework to assess supply chain resilience," in *Proceedings of the 5th International ISCRAM Conference*, 2008, pp. 596–605.

[3] X. Zhang and J. Han, "Analysis of the importance of nodes in supply chain network," in *Proceedings of the 2011 International Conference on Business Computing and Global informatization*, Shanghai, China, 2011, pp. 387–388.

[4] V. Gaura, O. Yadavb, G. Sonia, and A. Rathorea, "Identification of critical nodes and edges in a network based on clustering," in *Proceedings of the 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)*, Athens, Greece, 2021, pp. 1298–1304.

[5] F. Sabouhi and M. S. Jabalameli, "A stochastic bi-objective multi-product programming model to supply chain network design under disruption risks," *Journal of Industrial and Systems Engineering*, vol. 12, no. 3, pp. 196–209, 2019.

[6] D. M.Scotta, D. C. Nova, L. Aultman-Hall, and F. Guo, "Network robustness index: A new method for identifying critical links and evaluating the performance of transportation networks," *Journal of Transportation Geography*, vol. 14, pp. 215–227, 2006.

[7] R. Babazadeh and J. Razmi, "A robust stochastic programming approach for agile and responsive logistics under operational and disruption risks," *International Journal of Logistics Systems and Management*, vol. 13, no. 4, pp. 458–482, 2012.

[8] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios, *Operations Research*, vol. 43, no. 2, pp. 264–281,

[9] C.-S. Yu and H.-L. Li, "A robust optimization model for stochastic logistic problems," *International Journal of Production Ecnonomics*, vol. 64, pp. 385–397, 2000.

 ISBN: 978-1-61208-990-4

Towards a Semantic Model for Wise Systems A Graph Matching Algorithm

Abdelhafid Dahhani
LISTIC
Université Savoie Mont Blanc
Annecy, France
email: abdelhafid.dahhani@univ-smb.fr
0000-0001-6314-662X

Ilham Alloui
LISTIC
Université Savoie Mont Blanc
Annecy, France
email: ilham.alloui@univ-smb.fr
0000-0002-3713-0592

Sébastien Monnet
LISTIC
Université Savoie Mont Blanc
Annecy, France
email: sebastien.monnet@univ-smb.fr
0000-0002-6036-3060

Flavien Vernier
LISTIC
Université Savoie Mont Blanc
Annecy, France
email: flavien.vernier@univ-smb.fr
0000-0001-7684-6502

***Abstract*—Wise systems refer to distributed communicating software objects, which we named Wise Objects, able to autonomously learn how they behave and how they are used. They are designed to be associated either with software or physical objects (e.g., home automation) to adapt to end users while demanding little attention from them. Construction of such systems requires at least two views: on one hand, a conceptual view relying on knowledge given by developers to either control or specify the expected system behavior. On the other hand, wise systems are provided with mechanisms to generate their own view based on behavior-related knowledge acquired during their learning processes. The problem is that, while a conceptual view is understandable by humans (i.e., developers, end users, etc.), a view generated by a software system contains mainly numerical information with mostly no meaning for humans. In this paper, we address the issue of how to relate both views using two state-based formalisms: Input Output Symbolic Transition Systems for conceptual views and State Transition Graphs for views generated by the wise systems. Our proposal is to extend the generated knowledge with the conceptual knowledge using a matching algorithm founded on graph morphism. This provides the ability to make wise systems' generated knowledge understandable by humans and to enable human evaluation of wise systems' outputs.**

***Keywords*—statecharts; monitoring systems; adaptive system and control; knowledge-based systems; discrete-event systems; graph matching.**

I. INTRODUCTION

Software systems are now everywhere in our daily life. Their usage may vary depending on the end user and may evolve in time. A wise system should be able to adapt itself gracefully according to its usage. It can be seen as a particular Multi-Agent System [1][2] that monitors only its internal changes and does not observe its external environment. To tackle this issue, we have previously proposed Wise Objects (WO) and Wise systems. A WO is a piece of software able to monitor itself: the way it is used and the way it could be used (through introspection). A Wise system is simply a collection of communicating WOs. The main specificity of a WO is that it is able to autonomously learn about itself: it monitors its method invocations and their impact, it can also simulate method invocations to envision possible use and explore/discover new states. A method implements a service or functionality provided by a WO. Then, the collected monitoring data can feed a learning process to be able to determine usual and unusual behavior (for instance). The autonomic behavior is a key property: the end user should not be required to interact with the WO to help it in its learning process. An example is that of a home-automation system that collects someone's behavior in a room and analyses it to be able to act "silently" when necessary. Such systems should require the minimum attention from their end users while being able to adapt to changes in their behavior.

To meet those requirements and as the development of such systems is non-trivial, we developed an object based framework named Wise Object Framework [3] (WOF) to help developers design, deploy and evolve wise systems. Generally, knowledge in Artificial Intelligent (AI) enabled systems can be provided according to two ways: describing a priori the arrangement of activities to be performed by the system, or, letting the system acquire the required knowledge using learning mechanisms.

In the former case, ontologies and/or scenarios are usually used to describe the arrangement of activities to achieve a goal as in [4][5]. In [4], functional behavior as well as inter-operation of system entities are described a priori using state-diagrams. Reference [5] goes a step forward by combining ontologies to design ambiant assisted living systems with specifications based on logic and analyzers to check in logic clauses before system deployment to create relevant scenarios. In those approaches, the end user is at the heart of the scenario creation process, as described in [6][7]. *In the second case*, knowledge is provided by the AI-enabled system in

 ISBN: 978-1-61208-990-4

representations and views not necessarily understandable by humans. This relates to the wide problem of comprehensibility of AI and to the distance between the business domain and technological domain views [8].

In this context, the WO acquires by itself knowledge about its capabilities – services to be provided – and its use to moderate attention from the end-users [9] (Calm technology). Mark Weiser and John Seely Brown in [10] describe calm technology as "that which informs but does not demand our(users) focus or attention". The WO also analyses this knowledge to generate new one. As a result, it produces a State Transition Graph (STG) of the WO behavior. We consider STGs as the most natural way to model system dynamics. More precisely, this graph is built by iteration, i.e., step-wise construction, during a process called introspection. This process is launched during a phase called dream phase in which the WO discovers all its states (configurations) [11]. The downside of an STG generated by a WO is that numeric data provided has no meaning for humans. In the literature [12][13], others graphs like Input Output Symbolic Transition System (IOSTS) are often used to model the behavior of systems to manage them using oracle or controller synthesis. Since this type of graph is conceptually understandable by humans and it has semantics, it can increase the knowledge of WO and brings semantic to STGs.

Our proposal is to extend the generated knowledge with the conceptual knowledge using a matching algorithm based on graph morphism [14][15]. This provides the ability to make wise systems' generated knowledge understandable by humans and to enable human evaluation of wise systems' outputs. Explicitly, the contribution presented in this paper attempts to relate both views, consequently enabling machine-human communication: (a) a conceptual view relying on knowledge given by developers to either describe or control the system behavior, and, (b) behavior-related knowledge acquired during wise system's learning process. In this way, we use two state-based formalisms:

- STGs for representing behavior-related knowledge generated by the wise systems.
- IOSTSs for modelling conceptual views of developers/experts,

For many years, graphs have been used in many fields to represent complex problems in a descriptive way (e.g., maps, relationships between people profiles, public transportation) for various purposes: analysis, operation, knowledge modeling, etc. Although initiated in the 18th century with Euler's work on the now famous problem of Königsberg bridges [16], Graph theory remains a powerful tool for software-intensive system development. Among the most well-known operations on graphs is the comparison of two or more graph representations that requires many theoretical and complex concepts [14], like graph matching and graph morphism. These are at the basis of our proposal in this work. The rest of the paper is organised as follows. Section II presents the basic idea, describes the architectural overview and gives the definition of important terms. Section III presents STG and IOSTS formalisms and illustrates them through examples. Finally, Section IV presents our graph matching algorithm, before Section V, which concludes the paper.

II. Basic idea & architectural view

The first step toward WO concept is to respect the notion of "calm technolog" claimed by Mark Weiser and John Seely Brown in [17], by giving an entity the ability to autonomously adapt itself to its usage. We sketch in this section an overall view of how we designed the WO concept to meet this requirement.

A. Basic idea & definitions

The basic idea underlying the WO concept is to give a software entity (object, component, subsystem) the core mechanisms for learning behavior through introspection and analysis. Our aim is to go further by enabling software to execute "Monitoring", "Analyze", "Plan" and "Execute" loops based on "Knowledge", called MAPE-K [3]. At the core of this concept, we built the WOF [18] with design decisions mainly guided by reusability and genericity requirements: the framework should be maintainable and used in different application domains with different strategies (e.g., analysis approaches).

Seeking clarity, we borrowed some terms used for humans to refer to abilities a WO possesses. Awareness and wisdom both rely on knowledge. Inspired by [19], we give some definitions of those terms commonly used for humans [20] and present those we chose for WOs.

Knowledge: refers to information, inference rules and information deduced from them, for instance: "Turning on a heater will cause temperature change".

Awareness: represents the ability to collect - to provide internal data - on itself by itself. For instance, it is when an entity/object/device collects information and data about its capabilities *(what is intended to do)* and its use *(what it is asked to do)*. Capabilities are the services/functionalities the WO may render. They are implemented by methods that are invoked by the WO itself during the "dream" phase or from outside during the "awake" phase.

Wisdom: is the ability to analyse collected information and stored knowledge related to their capabilities and usage to output useful information. It is worth noticing that a WO is highly aware, while the converse is false.

Semantic: is the meaning given to something so that it can be understood by humans as mentioned in [20]. This definition also applies to objects/devices, as semantic is used to communicate with humans. The value "100" of a variable "data" means nothing to an end user if we do not give her/him the information that it represents a percentage of humidity.

B. WO from an architectural view

From an architectural perspective, according to the target application, a WO may be considered as [3]:

- a stand-alone software entity (object, component, etc.),

 ISBN: 978-1-61208-990-4

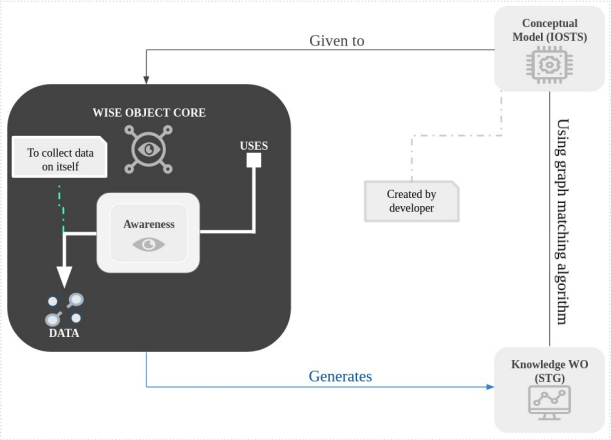


Figure 1. Generic functional architecture of a WO.

- a software avatar designed to be a proxy for physical devices (e.g., a heater, vacuum cleaner, light bulb) [21],
- a software avatar designed to be a proxy for an existing software entity (object, component, etc.).

A WO is characterised by its:

- autonomy: it is able to operate with no human intervention,
- adaptability: it changes its behavior when its environment evolves,
- ability to communicate: with its environment according to a publish-subscribe paradigm.

Figure 1 illustrates a partial view of a WO's functional architecture defined in the WOF. As depicted, the WO uses awareness to collect data on itself, either in simulation mode or usage mode. It analyses those data and generates a behavioral graph represented by an STG (Section III-A). States are constructed by the WO from attribute values of invoked methods and transitions from method invocations. On another hand, when designing an application, developers provide a conceptual model describing/specifying the way they view the behavior of the system's entities associated to WOs. Such models are represented using IOSTS and contain the semantic given by developers to WOs (Section III-B). The IOSTS formalism is mostly known in simplifying system modelling by allowing symbolic representation of parameters and variable values instead of concrete data values enumeration [22].

The STG and IOSTS will be given to the matching algorithm (Section IV) by the WO to automatically add the developer's semantic to the STG. A concrete implemented example will illustrate this matching.

III. Behavioral models, definitions and illustrations

Modeling the behavior of a system is enabled by tools and languages that result in informal, semi-formal (e.g., UML) or formal representations based on already proven theories [23] like graph theory. We have chosen STG and IOSTS graph-based theories to WO's behavior representation, respectively at the conceptual level (i.e., developer's view) and the wise system level (WO's generated view).

A. Definition of an STG

An STG is a directed graph where vertices represent the states of an object and transitions represent the execution of its methods. Let us consider an object defined by its set of attributes A and its set of methods M. According to this information (A and M) on the object, the STG definition is given in Definition 1.

***Definition** 1:* An STG is defined by the triplet $G(V, E, L)$ where V and E are, respectively, the sets of vertices and edges, and L a set of labels.

- V is the set of vertices, with $|V| = n$ where each vertex represents a unique state of the object, and conversely, each state of the object is represented by a unique vertex. Therefore $v_i = v_j \Leftrightarrow i = j$ with $v_i, v_j \in V$ and $i, j \in [0, n[$.
- E is the set of directed edges where $\forall e \in E$, e is defined by the triplet $e = (v_i, v_j, m_k)$, such that $v_i, v_j \in V$ and $m_k \in M$. This triplet is called a transition labeled by m_k. The invocation of method m_k from state v_i switches the object to state v_j.
- L is a set of vertex labels where any label $l_i \in L$ is associated to v_i.
  A label l_i is the set of pairs $(att_j, value_{i,j})$ $\forall att_j \in A$, with $value_{i,j}$ the value of att_j in the state v_i and $Dom(att_j)$ the value domain of att_j, i.e., the set of $value_{i,j}$ for all i. By definition, 2 states v_i and v_j are different $v_i \neq v_j$, iff $\exists att_k \in A$, such that $value_{i,k} \neq value_{j,k}$. Conversely, if $\forall k \in [0, |A|[$ $value_{i,k} = value_{j,k}$, the states v_i and v_j are considered the same, i.e., $v_i = v_j$, thus $i = j$.

The matching algorithm we propose in Section IV takes as input an STG with a specific property we name exhaustiveness. The definition of "exhaustive STG" is given in Definition 2.

***Definition** 2:* An exhaustive STG is an STG such that from each vertex v_i there exist $|M|$ transitions, each labeled by a method m_k in M:

$$\forall v_i \in V, \forall m_k \in M, \exists v_j \in V | (v_i, v_j, m_k) \in E.$$

It is worth noting that v_i and v_j may be different or same states ($v_i \neq v_j$ or $v_i = v_j$).

Consequently, an exhaustive STG is deterministic, i.e., from any state, on any method invocation, the destination state is known. Moreover, the number of transitions $|E|$ in an exhaustive STG depends on the number of vertices $|V|$ and methods $|M|$ such that:

$$|V| \times |M| = |E|.$$

Figure 2 illustrates an exhaustive STG for an object's behavior, defined by the attribute "level" ($A = \{level\}$) and 2 methods "open" and "close" ($M = \{open(), close()\}$). The methods "open" and "close" increase and decrease the level by 50, respectively. In the STG generated by an object

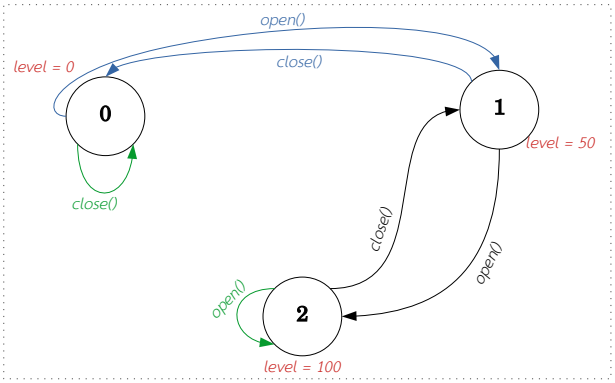


Figure 2. Example of an exhaustive STG.

for the shutter, except the methods that give semantic to the transitions, the states have no semantic. Considering the level is initialized to 0, the corresponding STG has 3 states and its exhaustive form has 6 transitions.

B. Definition of an IOSTS

An IOSTS is a directed graph whose vertices, called localities, represent different states of the system (in our case, the system is a software object) and whose edges are transitions. The localities are connected by transitions triggered by actions. In graph theory, an IOSTS allows us the definition of an infinite state transition system in a finite way, contrary to an STG where states are defined by discrete values. IOSTS are used to verify, test and control systems. Verification and testing are formal techniques for validating and comparing two views of a system while control is used to constrain the system behavior [13].

The definition of IOSTS given in Definition 3 is taken from [13][24] and especially from the use case given in [22].

***Definition** 3:* An IOSTS is a sixfold $\langle D, \Theta, Q, q_0, \Sigma, T \rangle$ such as:

- D is a finite set of typed data consisting of two disjoint sets of: variables X and action parameters P. The value domain of $d \in D$ is determined by $Dom(d)$.
- Θ : an initial condition expressed as a predicate on variables X.
- Q is a non-empty finite set of localities with $q_0 \in Q$ being the initial locality. A locality q is a set of states such that $q \in Dom(X)$, with $Dom(X)$ being the cartesian product of the domains of each $x \in X$. Let us note that a state is defined by a single tuple of values for the whole variables.
- Σ is the alphabet, a finite, non-empty set of actions. It consists of the disjoint union of the set $\Sigma^?$ of input actions, the set $\Sigma^!$ of output actions, and the set $\Sigma^\mathcal{T}$ of internal actions. For each action a in Σ, its signature $sig(a) = \langle p_1, \ldots, p_k \rangle | p_i \in P$ is a tuple of parameters. The signature of internal actions is always an empty tuple.
- T is a finite set of transitions, such that each transition is a tuple $t = \langle q_o, a, G, A, q_d \rangle$ defined by:
  - a locality $q_o \in Q$, called the origin of the transition,
  - an action $a \in \Sigma$, called the action of the transition,
  - a boolean expression G on $X \cup Sig(a)$ related to the variables and the parameters of the action, called the transition guard. Transition guards allow us to distinguish transitions that have the same origin and action but disjoint conditions to their triggering.
  - An assignment of the set of variables, of the form $(x := A^x)_{x \in X}$ such that for each $x \in X$, A^x is an expression on $X \cup Sig(a)$. It defines the evolution of variable values during the transition,
  - a locality q_d, called the transition destination.

According to this definition, each variable has a subdomain in each locality. Thus, let us define the function $dom(q, x)$ that returns the definition domain of the variable $x \in X$ in the locality $q \in Q$; consequently $dom(q, x) \subseteq Dom(x)$.

Figure 3 shows an example of an IOSTS given by a developer to control a roller shutter. This IOSTS expresses that the roller shutter expects an input $up?/down? \in \Sigma^?$ carrying the parameter $step \in]0, 100]$, the relative elevation to respectively increase or decrease the shutter level. Let us note that the shutter elevation is between 0 and 100.

There are 2 localities:

- The locality where the system is closed (i.e., $height = 0$). If the system receives the $up?(step)$ command, the transition will be made from the $Closed$ to $Open$ locality by increasing the value of the $height$ variable by $step$, but if the system receives the $down?(step)$ action, it will not perform any operation (NOP).
- The locality where the system is open (i.e., $height \in]0, 100]$). If the system receives the action $up?(step)$, the transition will be reflexive from $Open$ to itself and will compute the value of the variable $height$ by executing this assignment $height = min(height + step, 100)$, the shutter elevation cannot be increased more than the maximum of elevation. If it receives the $down?(step)$ action and the action closes the shutter less than it is open ($step < height$), $height$ is decreased by $step$, otherwise the transition will be from the locality $Open$ to the locality $Close$ by assigning 0 to the variable $height$.

According to Definition 3, this IOSTS is composed of the sets of variables $X = \{height\}$ with $Dom(height) \in [0, 100]$ and parameters $P = \{step\}$ with $Dom(step) \in]0, 100]$, the set of localities $Q = \{Open, Closed\}$ and the set of actions $\Sigma = \{up?, down?\}$ where the signatures of the actions are $Sig(up?) = Sig(down?) = \langle step \rangle$. This IOSTS models an infinite state system based on 5 guarded transitions in T:

$$T = \langle \begin{array}{l} t_{Close-Open}, \\ t_{Open-Close}, \\ t^1_{Open-Open}, \\ t^2_{Open-Open}, \\ t_{Close-Close} \end{array} \rangle$$

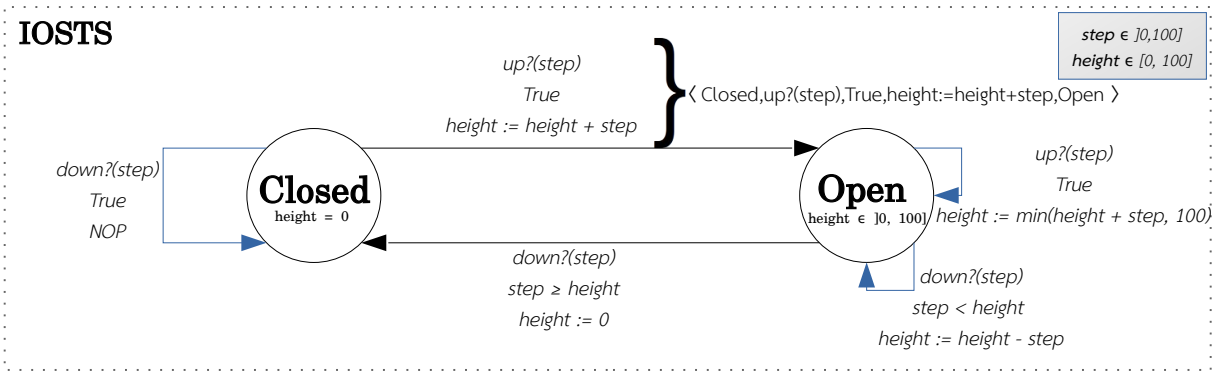


Figure 3. IOSTS representation of a roller shutter.

such as:

$$
\begin{aligned}
t_{Close-Open} = \ & \langle \ Open, up?(step), \\
& True, height := height + step, \\
& Open \rangle \\
t_{Open-Close} = \ & \langle \ Open, down?(step), \\
& step \geq height, height := 0, \\
& Close \rangle \\
t^1_{Open-Open} = \ & \langle \ Open, down?(step), \\
& step < height, height := height \\
& -step, Open \rangle, \\
t^2_{Open-Open} = \ & \langle \ Open, up?(step), \\
& True, min(height + step, 100), \\
& Open \rangle, \\
t_{Close-Close} = \ & \langle \ Close, down?(step), \\
& True, NOP, \\
& Close \rangle.
\end{aligned}
$$

As can be noticed, there exists an infinity of paths and states represented by the variables $height$ since its domain is the interval $[0, 100]$.

IV. Graph matching algorithm

In this section, we introduce the matching algorithm we propose to relate WO's generated STG to developers' semantic expressed in an IOSTS. In the example of Figure 2, the generated STG is composed of states automatically labelled by the object: 0, 1 and 2 according to the value of attribute level: 0, 50, 100. The main challenge is how to match states 0, 1 and 2 to the localities defined by developers in the IOSTS of Figure 3.

A. Matching algorithm

Constraint: The STG and IOSTS must meet certain criteria to properly apply the algorithm.

1) There are two equivalent characteristics: a variable x_e belonging to the set of variables X of the IOSTS and an attribute att_e belongs to the set of STG attributes A. Moreover, to simplify the problem in this paper, let us consider they are unique:

$$\exists! x_e \in X, \exists! att_e \in A | x_e \equiv att_e,$$

$x_e \equiv att_e$ means that both represent the same information, thus:

$$Dom(att_e) \subseteq Dom(x_e).$$

Let us note that $Dom(att_e)$ is a subset of $Dom(x_e)$ due to the fact that x_e is theoretically defined into the IOSTS and att_e is partially discovered by the WO at runtime.

2) The domains of x_e in the different localities in the IOSTS are disjoint:

$$\forall q, q' \in Q, dom(q, x_e) \cap dom(q', x_e) = \emptyset,$$

Algorithm: According to the definitions of STG and IOSTS, and both constraints, a vertex $v_i \in V$ matches a locality $q_j \in Q$ (noted $v_i \implies q_j$) if and only if $value_{i,e} \in dom(q_j, x_e)$, with $value_{i,e}$ the value of att_e in the vertex v_i:

$$\forall v_i \in V, \exists! q_j \in Q$$
$$value_{i,e} \in dom(q_j, x_e) \Leftrightarrow v_i \implies q_j.$$

As the matching algorithm is a graph morphism, this latter needs to respect the structure of the matched graphs [25]. In our context, each vertex matches one locality and a locality is matched by at least one vertex. Moreover, the adjacency relations must be respected by the matching; if 2 vertices are linked by a transition in the STG, their matched localities are the same or linked by an equivalent transition in the IOSTS. The STG $\rightarrow$ IOSTS maching is surjective homomorphism called epimorphism [25].

The pseudo-code in Algorithm 1 is the first version of the matching algorithm we have developed.

B. Matching illustration

In the previous examples: the STG in Figure 2 is automatically generated by a WO and the IOSTS in Figure 3 is provided by a developer. Both represent the same roller shutter behavior. The STG uses discrete values with a level of opening of 50%, while the IOSTS uses continuous intervals, without any constraint on the step that is a real value.

Figure 4 illustrates the result of matching both graphs using our graph matcher implemented with Python. Localities in the IOSTS are $Closed$ and $Open$, each containing variables with

Algorithm 1 Graph matching algorithm

Inputs:
$iosts$: IOSTS,
stg: Exhaustive STG
Outputs:
$match$: Dictionary<state, locality>
Locales:
Set of possible attribute/variable pairs
E: Set Of Tuples< (attribute, variable) >
Possible matches for each possible equivalent pair
M: Dictionary< (attribute, variable), <state, locality>>
Initialize:
Build possible equivalent attribute/variable pairs, such that $dom(a) \subseteq dom(x)$
$E \leftarrow compatibleDomain(stg.A, iosts.X)$
for $(a, x) \in E$ **do**
  for $v_i \in stg.V$ **do**
# Get the locality where the domain of variable x contains the value of a in v_i, according to the second constraint, q_i is unique
    $q_i \leftarrow iosts.getLocality(x, v_i.getValue(a))$
    $M((a, x))(v_i) \leftarrow q_i$
  end for
# As the matching is a surjective application, remove the pair if it does not generate surjective matching
  if $M((a, x)).getKeys() \neq stg.V$
  or $M((a, x)).getValuesAsSet() \neq iosts.Q$ **then**
    E.remove((a,x))
    M.remove((a,x))
  else
# If the application is surjective, check the transitions' consistency
    for $e \in stg.E$ **do**
      $v_1 \leftarrow e.getSource()$
      $v_2 \leftarrow e.getDestination()$
      $q_1 = M((a, x))(v_1)$
      $q_2 = M((a, x))(v_2)$
      if $iosts.getTransition(q_1, q_2)$ **is null then**
        E.remove((a,x))
        M.remove((a,x))
      end if
    end for
  end if
end for
# Checking that just only one matching exists according to constraints defined in Section IV-A
if $M.getKeys().size() == 1$ **then**
  $match \leftarrow M.getValues()[1]$
else
  $exception$("Required conditions not satisfied")
end if
return $match$

disjoint domains, in our example, a single variable named $height$ that takes different values depending on its locality.

According to the constraints of the matching algorithm:

1) there are equivalent characteristics between the STG and the IOSTS, the attribute "level" and the variable "height", respectively,
2) the domains of "height" in the different localities are disjoint from the others: in $Closed$ locality, the variable can only take the value 0 and in $Open$ locality, the variable can take any value in the range $]0, 100]$.

On the STG side, there are three vertices, each one labeled with a set of attribute-value pairs (att, value). In our case, the unique attribute $level$ takes the values $(0, 50, 100)$ respectively for (v_0, v_1, v_2). Therefore, to establish a correspondence between the two graphs, a comparison between the definition domain of the attribute $level$ in each vertex of the STG with the definition domain of the variable $height$ in each locality of the IOSTS must be done.

Those comparisons lead us to a correspondence of state v_1 with locality Closed meaning that the roller shutter is closed, and a correspondence of states v_2 and v_3 with locality Open meaning that the roller shutter is open.

This first version of the matching algorithm has been developed and tested as a first step towards human semantics.

V. Conclusion and Future Work

In this paper, we addressed the problem of relating numerical representations generated by wise systems to developers' semantics. The contribution is a matching algorithm that computes a morphism between two behavioral graphs:

1) an STG generated by a WO along its learning process,
2) an IOSTS representing a developer conceptual view.

That algorithm extends a WO's view with semantics that allow it to communicate with humans. From the developer's perspective, the resulted matching may help developers discover errors and/or inconsistencies between the conceptual view and the system implementation. In its first version, the algorithm has obviously several limitations, the strongest being over the number of equivalent attributes/variables in STG/IOSTS. Another limitation is the constraint on the existence of only one matching between an STG and an IOSTS. Ongoing work is being done to gradually generalize the algorithm and raise those restrictions. We also intend to investigate other matching algorithms towards other semantic formalisms than IOSTS, such as ontology and scenario-based ones [4][5].

Acknowledgment

This research was supported by French National Research Agency (ANR), AI Ph.D funding project.

References

[1] R. A. Flores-Mendez, "Towards a standardization of multi-agent system framework," XRDS, vol. 5, no. 4, pp. 18–24, 1999.
[2] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," IEEE Access, vol. 6, pp. 28573–28593, 2018.

 ISBN: 978-1-61208-990-4

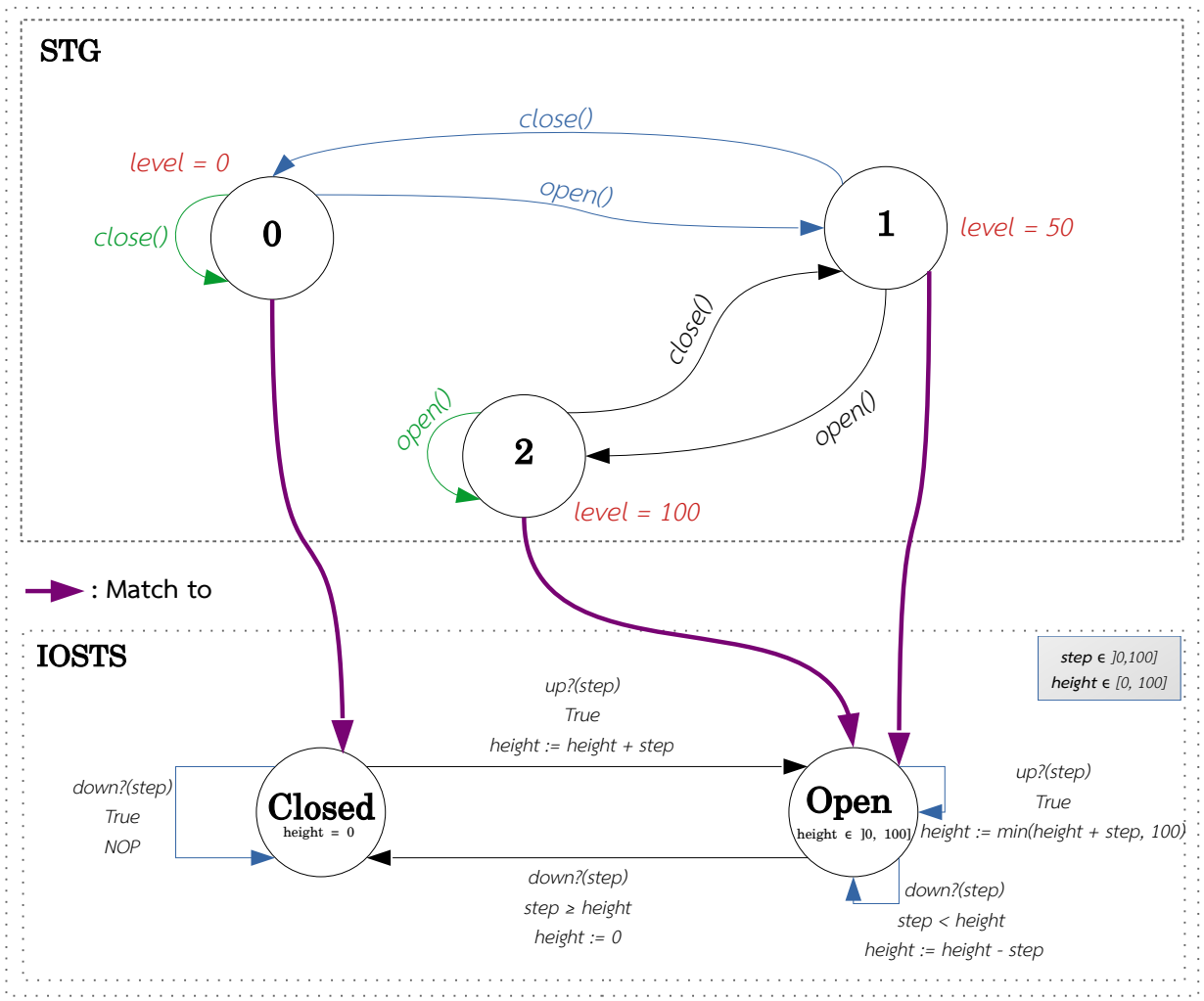


Figure 4. Algorithm result of the graph matching.

[3] I. Alloui and F. Vernier, "WOF: Towards Behavior Analysis and Representation of Emotions in Adaptive Systems," Communications in Computer and Information Science, vol. 868, pp. 244–267, 2018.

[4] D. Bonino and F. Corno, "Dogont - ontology modeling for intelligent domotic environments," in The Semantic Web - ISWC 2008, pp. 790–803, Springer Berlin Heidelberg, 2008.

[5] H. Kenfack Ngankam, H. Pigot, M. Frappier, C. H. Souza Oliveira, and S. Giroux, "Formal specification for ambient assisted living scenarios," UCAmI, pp. 508–519, 2017.

[6] J.-B. Woo and Y.-K. Lim, "User experience in do-it-yourself-style smart homes," in Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 779–790, 2015.

[7] R. Radziszewski, H. Ngankam, H. Pigot, V. Grégoire, D. Lorrain, and S. Giroux, "An ambient assisted living nighttime wandering system for elderly," in Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services, iiWAS '16, pp. 368–374, Association for Computing Machinery, 2016.

[8] R. S. Michalski, "A theory and methodology of inductive learning," in Machine Learning: An Artificial Intelligence Approach, pp. 83–134, Springer Berlin Heidelberg, 1983.

[9] M. Weiser, "The computer for the 21st century," Scientific American, vol. 265, no. 3, pp. 66–75, 1991.

[10] A. Tugui, "Calm technologies in a multimedia world," Ubiquity, vol. 2004, pp. 1–5, 2004.

[11] I. Alloui and F. Vernier, "A Wise Object Framework for Distributed Intelligent Adaptive Systems," in ICSOFT 2017, the 12th International Conference on Software Technologies, pp. 95–104, 2017.

[12] C. Constant, T. Jéron, H. Marchand, and V. Rusu, "Validation of Reactive Systems," in Modeling and Verification of Real-TIME Systems - Formalisms and software Tools, pp. 51–76, Hermès Science, 2008.

[13] C. Constant, T. Jéron, H. Marchand, and V. Rusu, "Integrating Formal Verification and Conformance Testing for Reactive Systems," IEEE Transactions on Software Engineering, vol. 33, no. 8, pp. 558–574, 2007.

[14] M. R. Garey and D. S. Johnson, "Computers and intractability. a guide to the theory of np-completeness.," Journal of Symbolic Logic, vol. 48, no. 2, pp. 498–500, 1983.

[15] V. A. Cicirello, "Survey of graph matching algorithms," technical report, Geometric and Intelligent Computing Laboratory, Drexel University, 1999.

[16] H. Sachs, M. Stiebitz, and R. Wilson, "An historical note: Euler's königsberg letters," Journal of Graph Theory, vol. 12, pp. 133 – 139, 2006.

[17] M. Weiser and J. S. Brown, "Designing calm technology," PowerGrid Journal, vol. 1, pp. 75–85, 1996.

[18] I. Alloui, D. Esale, and F. Vernier, "Wise objects for calm technology," in Proceedings of the 10th International Conference on Software Engineering and Applications - ICSOFT-EA, (ICSOFT 2015), pp. 468–471, INSTICC, SciTePress, 2015.

[19] T. Davenport and L. Prusak, Working Knowledge: How Organizations Manage What They Know, vol. 1. Harvard Business School Press, 1998.

[20] "Cambridge Dictionary Online," 2022.

[21] I. Alloui, E. Benoit, S. Perrin, and F. Vernier, "Wise objects for IoT

 ISBN: 978-1-61208-990-4

(WIoT): Software framework and experimentation," Communications in Computer and Information Science, pp. 349–371, 2019.

[22] P. Moreaux, F. Sartor, and F. Vernier, "An effective approach for home services management," in 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 47–51, 2012.

[23] M. N. Nicolescu and M. J. Matarić, "Extending behavior-based systems capabilities using an abstract behavior representation," in AAAI 2000, pp. 27–34, 2000.

[24] V. Rusu, H. Marchand, and T. Jéron, "Automatic verification and conformance testing for validating safety properties of reactive systems," in Formal Methods 2005 (FM05), vol. 3582 of Lecture Notes in Computer Science, pp. 189–204, Springer-Verlag, 2005.

[25] G. Hahn and C. Tardif, "Graph homomorphisms: structure and symmetry," in Graph Symmetry: Algebraic Methods and Applications, pp. 107–166, Springer Netherlands, 1997.

 ISBN: 978-1-61208-990-4