ISTANBUL/TURKEY

COMPUTER SCIENCE STUDENT WORKSHOP

Proceedings of the 1st Computer Science Student Workshop

CSW-2010
1st Computer Science Student Workshop

Proceedings of the 1st Computer Science Student Workshop

Koc University Istinye Campus, Istanbul, Turkey, February 21, 2010.

Edited by

Cengiz Orencik *
Mehmet Ali Yatbaz **
Tolga Eren *
Tayfun Elmas **
Tekin Mericli ***

* Sabanci University, Orhanli, 34956 Istanbul, Turkey
** Koc University, Sariyer, 34450 Istanbul, Turkey
*** Bogazici University, Bebek, 34342 Istanbul, Turkey

Sabancı Üniversitesi
Orhanlı - Tuzla, 34956 İstanbul
Telefon: (0216) 483 9000
Faks: (0216) 483 9005
Web adresi: www.sabanciuniv.edu

Preface

This volume contains the proceedings of the 1st Computer Science Student Workshop (CSW). The workshop took place on February 21st, 2010 at the Istinye Campus of Koç University, Istanbul.

CSW aims to bring the Computer Science and Engineering graduate students in Istanbul together in a semiformal workshop atmosphere. This workshop exposes the graduate students to the concepts of academic writing, peer review, research presentation, critical thinking as well as academic way of thinking in general. The students also establish connections in this semiformal environment via meeting each other, sharing ideas, and getting feedback on their work. The ultimate goal of this workshop series is to form a network of young researchers who will support each other and establish a core group of senior graduate student leaders, who will serve as mentors and role models for the coming generation. Therefore, the workshop is organized by graduate students for graduate students.

There were four oral presentation sessions in total, and three poster sessions in between. The oral presentation sessions were categorized into three main groups; namely "Artificial Intelligence", "Network and Security", and "Bioinformatics, Data Mining, and Computer Architecture". There were 50 submissions in total and 16 of the submissions were accepted for oral presentation while 15 of them were accepted to be presented as posters.

Several contributors of the CSW, either as authors or Program Committee members, were awarded in the "Best Tool Paper", the "Best Research Paper", the "Best Work-in-Progress Paper", the "Best Poster", the "Best Presentation", and the "Best Reviewer" categories.

This successful workshop would not be possible without the initiation and support of our professors Esra Erdem and Metin Sezgin, and the hard work of all members of the Organizing Committee and the Program Committee. We would also like to sincerely thank to Koç University and Sabancı University for being the sponsors of the workshop.

Workshop chairs

Tekin Meriçli Tayfun Elmas Tolga Eren

Organization

Organizing Committee

Workshop Chairs

Tayfun Elmas, Koç University

Tolga Eren, Sabancı University

Tekin Meriçli, Boğaziçi University

Local Chairs

Bulut Altıntaş, Yeditepe University

Fatma Ergin, Marmara University

Kenan Kule, İstanbul Technical University

Erkan Uslu, Yıldız Technical University

Publications Chairs

Cengiz Örencik, Sabancı University

Mehmet Ali Yatbaz, Koç University

Logistics Chairs

Baybora Bektaş Baran, Koç University

Duygu Karaoğlan, Sabancı University

Publicity Chair

Duygu Çakmak, Sabancı University

Program Committee

Nazlı Nakeeb Alan, Yeditepe University

Bulut Altıntaş, Yeditepe University

Reyhan Aydoğan, Boğaziçi University

Muhammet Balcılar, Yıldız Technical University

Baybora Bektaş Baran, Koç University

Murat Birben, Yeditepe University

Duygu Çakmak, Sabancı University

Emrah Çem, Koç University

Sevgi Çilengir, İstanbul University
Tayfun Elmas, Koç University
Billur Engin, Koç University
Halit Erdoğan, Sabancı University
Tolga Eren, Sabancı University
Fatma Ergin, Marmara University
Utku Erol, İstanbul University
Özgür Kafalı, Boğaziçi University
Emre Kaplan, Sabancı University
Duygu Karaoğlan, Sabancı University
Kenan Kule, İstanbul Technical University
Şükrü Kuran, Boğaziçi University
Tekin Meriçli, Boğaziçi University
Cengiz Örencik, Sabancı University
Bariş Şenliol, İstanbul Technical University
Oya Şimşek, Sabancı University
Berker Taşoluk, İstanbul University
Sinan Tümen, Koç University
Gönül Uludağ, İstanbul Technical University
Erkan Uslu, Yıldız Technical University
Abuzer Yakaryılmaz, Boğaziçi University
Mehmet Ali Yatbaz, Koç University

Table of Contents

Fixation Prediction Using Local Image Features

Sinan Tumen STUMEN@KU.EDU.TR
T. Metin Sezgin MTSEZGIN@KU.EDU.TR
Dept.of Computer Engineering, Koc University

1. Introduction

Human visual system processes only a tiny region of the scene despite of a large field of view. As the eccentricity of a scene point from the fovea increases, the resolution decreases quadraticly. To recognize a scene precisely, high resolution is acquired using active scan of the environment. This scanning process consists of saccades and fixations. Saccades are rapid movements of the eye in which no information is gathered as opposed to fixations which are long enough to let photoreceptors to respond to visual stimuli. A typical eye scan pattern executed by an observer viewing a scene is illustrated in Figure 1.

Gaze is directed to the most informative regions of the scene to collect as much as information in a short period of time (Land, 2006). Finding informative regions of a scene has many uses in many applications and contexts including computer vision (e.g., robot vision, compression, salience estimation) and human-computer interaction (e.g., interface usability assessment). The objective of this study is to estimate the intensity of fixations in a particular patch.

To explore the factors which determine the locations of the fixations we analyzed the DOVES dataset (Van Der Linde et al., 2009) using machine learning techniques. In this dataset, eye movements are recorded from 29 human observers as they viewed 101 images of size 1024×768. Each image is displayed for 5 seconds, eye trajectory is sampled at frequency of 200 Hz which. The participants are asked whether the given patch belongs to the current image, after they examined it for five seconds.

2. Related Work

Study of computational modeling of eye movements have been resulted in two major models: top-down and bottom-up (Itti, 2000). The While visual attention is directed by high level features such as the goal of the task and context of the scene in top-down approach, it is triggered by the visual local statistics in bottom-up approach. Bottom-up approach is reported to be succeeded in the predicting the region of interests in the absence of a top-down guidance (Itti, 2000). Based on the feature integration theory

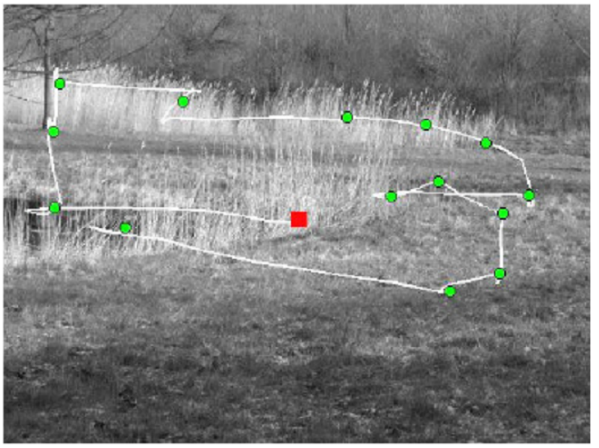


Figure 1. Eye Scan Path

(Treisman & Gelade, 1980), local features are used to create saliency maps of the images in bottom-up approaches. Then the peaks of saliency map are used to predict the sequence of fixations (Itti & Koch, 2001; Rajashekar et al., 2008). In this study, instead of trying to find the most probable sequence of fixations, we try to estimate number of fixations in a particular part of an image.

3. Model

DOVES dataset consists of images without contextual information which makes it appropriate for bottom-up approach. The local features which are reported to be statistically different from non-fixated patches are used to create feature maps of the images (Rajashekar et al., 2006). These features are luminance, RMS contrast, band pass of luminance and band-pass of patch contrast. Then we used feature maps to estimate the intensity of fixations in each region of the images.

3.1 Luminance Map

The mean luminance for an image patch was computed using a circular raised cosine weighting function w, as follows (Rajashekar et al., 2008);

$$\bar{I} = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} I_i w_i$$

where M is the number of pixels in the patch, I_i is the grayscale value of the pixel at location i and the raised cosine function w is expressed as:

$$w(i) = 0.5 \left[\cos\left(\frac{\pi r_i}{R}\right) + 1 \right]$$

3.2 RMS Contrast Map

The mean luminance for an image patch was computed using a circular raised cosine weighting function w, as follows (Rajashekar et al., 2008);

$$C = \sqrt{\frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} w_i \frac{(I_i - \bar{I})^2}{(\bar{I})^2}}$$

where M is the number of pixels in the patch, I_i is the grayscale value of a pixel location i and I is the mean luminance of the patch.

3.3 Bandpass of Patch Luminance Map

Regions that differ from their surroundings can be detected by the outputs of the bandpass Gabor kernels. Since attention often seems to be drawn to regions that differ from their surroundings, output of the band pass Gabor filter is used as input to saliency map (Rajashekar et al., 2008).

3.4 Bandpass of Patch Contrast Map

Bandpass outputs of local image contrast are used to capture higher order image structure that is ignored by the luminance filter (Rajashekar et al., 2008).

3.5 Features

Four 1024x768 saliency maps are generated using the formulas above. Then saliency maps are divided into square patches of size 64x64, which results in 16x12 regions. Statistics in these patches are collected as features of the region. The statistical parameters are mean, standard deviation, maximum and minimum of intensity in each saliency maps that corresponds to 16 features. Since people tend to fixate to the regions close to the center of the image, we added Euclidean distance to the center as another features.

3.6 Objective Function

Each image is divided into 64x64 square regions and the number of fixations of each participant is summed in each patch. The entries of the resulting matrix give the total number of fixations to a particular region in an image. Then regression problem formulated as estimating the number of fixations made to a particular region using features extracted from the saliency maps of the same patch.

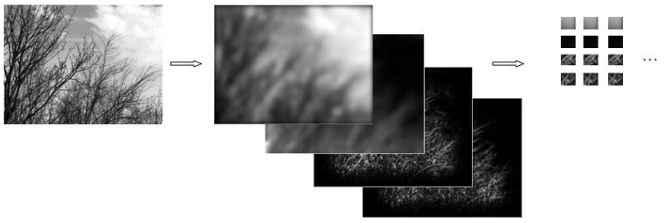


Figure 2. Feature Extraction

4. Estimation Techniques

Generated dataset with 19392 objects corresponding to square patches extracted from images are split into training (60%), validation (20%) and test sets (20%). After regression models are trained on the training set, parameters of the models are optimized on validation set. The mean of the squared residuals and the correlation coefficients of target and estimated values are presented in Table 1. The learning curve is stabilized as data size increases, suggesting that addition of more data will not decrease the error. If we take baseline as mean of all fixations in each patch, the mean square error would be 3.1 and correlation coefficient would be 0.35. The result suggests that performance of the trained models are just between the human visual system and completely arbitrary fixation selection system.

Regression Method	MSE	Corr.Coefficient
Baseline	3.1	0.35
Linear	1.52	0.74
SVM	1.35	0.76
KNN	1.64	0.71
Ridge	1.51	0.74
Lasso	1.86	0.69

5. Conclusion

Although human visual system makes 4-6 fixations per second, these fixations are not deployed to the arbitrary locations. Instead during the deployment of the current fixation, the next fixation is selected and eye moves to the next location according to a attraction rule which is formed by contextual information and local features of the scene. In the absence of a context in the scene, the attraction is dominated by the local features. With the method presented in the study, the most informative regions in an context-free image can be revealed using only the local features. It can

also be used in evaluation of the local features for generation of saliency maps and comparison with the ground truth which is real number of fixations in each region.

6. Future Work

There are many other candidate features suggested for the generation of feature maps. The use of these features may improve the accuracy of the estimation. Another area of improvement is making use of sequential methods such as Kalman filter and Markov chain in modeling the eye movements. We plan to use suggested features and temporal models in our next studies.

References

Itti, L. (2000). *Models of bottom-up and top-down visual attention*. Doctoral dissertation.

Itti, L., & Koch, C. (2001). Computational modelling of visual attention. *Nature Reviews Neuroscience*, *2*, 194–204.

Land, M. (2006). Eye movements and the control of actions in everyday life. *Progress in Retinal and Eye Research*, *25*, 296–324.

Rajashekar, U., van der Linde, I., Bovik, A., & Cormack, L. (2006). Statistical analysis and selection of visual fixations. *Journal of Vision*, *6*, 496.

Rajashekar, U., van der Linde, I., Bovik, A., & Cormack, L. (2008). GAFFE: A gaze-attentive fixation finding engine. *IEEE Transactions on Image Processing*, *17*, 564.

Treisman, A., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive psychology*, *12*, 97–136.

Van Der Linde, I., Rajashekar, U., Bovik, A., & Cormack, L. (2009). DOVES: A database of visual eye movements. *Spatial Vision*, *22*, 161–177.

Advances in malware development: Using emulators weaknesses and cryptography

Can Yıldızlı CANYILDIZLI@SABANCIUNIV.EDU
Sabancı University, Orhanlı - Tuzla
Istanbul / Turkey
P.O. Box 34956

1. Introduction

Malwares become more sophisticated in terms of their obfuscation mechanisms. Hence, it gets harder to detect them with current analysis methods. Today, malwares are capable to detect analysis tools and environments to evade detection. Anti-viruses trying to develop more accurate detection mechanisms to protect users from malware threats. However, anti-virus heuristics are based on emulation of code which is a weak spot for malware authors. If an emulation process can be detected, a malware can stop it's execution or execute some junky code depending on the implementation. In this paper we present ways to hide a malware from anti-viruses by generating a decryption key using the difference between emulation and real execution. We also present our tool "Cryptoware", which has the capability to make any existing malware undetectable to anti-viruses. Our method can not be bypassed by proposed anti-anti-Vmware methods (Sun et al., 2008) since decryption algorithm needs a key which should be generated while executing the code in an environment. The outline of our paper is as follows.

In Section 2, we explain general analysis techniques of anti-viruses and their weaknesses. We also give an example of using emulation and real execution difference to derive a key with an obfuscation method. In Section 3, we show how malicious cryptography usage can help the attackers to hide the functions they are using. In Section 4, we present our tool "Cryptoware" which makes any malware completely undetectable to anti-viruses by using the techniques that we described. We also show results of detection rates using techniques described in this paper. Finally in Section 5, we conclude and propose a new way to emulate obfuscated malwares as our future work.

2. Analysis Techniques and Problems

Anti-viruses make use of static and dynamic analysis techniques to effectively detect and clean malware. In static analysis, code of the malware is never executed(Daoud et al., 2008). This method uses pattern matching which allows detecting malicious code faster. However, the technique is ineffective since most of the malwares are in obfuscated form making use of polymorphism, metamorphism and various other techniques. Also, understanding binary code becomes generally an impossible task with static analysis if the malware is self-modifying. Dynamic analysis, on the other hand, tries to detect malware by emulating its instructions in a protected environment and observes its behavior(Aycock, 2006). Dynamic analysis generally provides better results than static analysis because it makes the identification of obfuscated code faster and easier by emulating the code. However, there exist some differences between emulation and real execution. These differences are heavily used by malware authors to evade detection. Once a difference is found between two execution environments, it becomes easy to encrypt malicious code in a way that the emulator cannot decrypt it. For example, suppose a virus V is encrypted and we have no way of detecting it with static analysis in a reasonable time. V contains two parts. First part contains decryption routine for the encrypted part of the virus. We will call this part S (S represents stub) and the remaining part which contains encrypted malicious code with B (B represents the body of the virus) (Fig. 1).

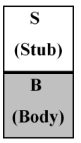


Figure 1. General representation of an obfuscated malware

A typical execution will start from S, S will decrypt B with the key already hard-coded in itself. Since S doesn't contain any malicious code but only decryption routine, V will be considered as safe in static analysis. However, emulating this code will reveal malicious part B since decryption routine decrypts B with the correct key which is calculated by S. In order to hide malicious part from dynamic anal-

ysis and emulators we will use a method to generate decryption keys in which some environment information is involved. Suppose that we used symmetric encryption and we encrypted the malicious code with the key *k* with a symmetric encryption algorithm. Real execution will be able to decrypt the body of the virus correctly since it can calculate *k*. However, from emulator's perspective, calculated value of *k* will be totally different. As a result of this, anti-viruses and emulators will observe meaningless instructions whenever they try to emulate the malware. To achieve this, we need some difference between emulation and real execution so that we can generate different keys for decryption when we execute the malware.

2.1 Timing (execution speed) differences

Emulators are not capable to emulate instructions as fast as CPU does. We will explain how to mislead an emulator to calculate different key with a simple assembly instruction called **rdtsc** (Read time stamp counter) which counts the number of ticks since computer reset[1]. The idea is to make a comparison between the running time of some dummy instructions within the virus and detecting if the code is trying to be emulated by comparing timing values. Suppose that a virus *V* has a body *B* which is encrypted with a key *k*. We make two consecutive **rdtsc** calls and calculate the difference between results of these functions. It turns out that in a real execution, timing value will be always smaller than 100h. However, when this code is emulated, timing value will be significantly higher. This value can be a random number based on the implementation of the emulator. Timing value difference cannot simply be tested with a conditional branch in the malware. Dynamic analysis methods are smart enough to save the places where a conditional branch occurs. They are also capable to revert back to the branch point and check if the other execution flow looks suspicious or not. Because of that, instead of comparing if the value is smaller than 100h, we generate decryption key by using this information directly. Since we know the timing difference is below 100h, we can easily make a bitwise *AND* operation to produce a fixed number from that difference. This fixed number can be used to calculate the decryption key with appropriate addition or multiplication operations. Emulation, on the other hand, produces wrong number from the result of the same *AND* operation. As a result of this, decryption key will be different than the real execution. Using wrong key for decryption will generate corrupted code in emulation, whereas real execution will decrypt the malicious code correctly.

[1]Details for this instruction can be found here: http://www.intel.com/design/intarch/manuals/243191.htm

2.2 Exception handling

Whenever an error occurs in a normal flow of a program, it is expected to generate an exception to inform user about this error. In a Windows operating system, this task is maintained by SEH (Structured Exception Handling). SEH will change the flow of the execution whenever an exception occurs. It is expected that the handler will fix the exception error and returns to normal flow of execution. At worst, SEH will inform the user about the error if the exception is caused by user interaction and stops the program. Emulation of SEH in an anti-virus is different than real execution. Most anti-viruses stop emulating the code whenever they detect a fatal error that causes an exception and terminate the application. Some of them just use their own exception handlers to deal with common types of exceptions. Malwares can make use of this difference by overwriting SEH entries with their malicious routines so that if an exception occurs at some point, control of the program will be directed to the malicious part by the SEH.

2.3 Environment variables

Most emulators return fixed results when some native API is used to retrieve some data about the working environment of a program, like some contents of a file, date, time or details about underlying processor. Those information can be used to produce the decryption key like in example given in section 2.1. Descriptor table addresses of emulators may also help to detect whether the code is being emulated(Quist & Smith, 2004). Addresses of Interrupt Descriptor Table (IDTR), Local Descriptor Table (LDTR) and Global Descriptor Table (GDTR) can be accessed with basic instructions. Those addresses are fixed on real execution of the program but vary in the emulated environments since emulators should provide their own set of fixed table addresses. Detecting virtual machine's presence by using descriptor table addresses is a widely used technique by malware authors. There have been many examples for implementing those detection routines into a single program to detect emulators like Scoopy Doo[2], Red Pill[3], No Pill. Our program Cryptoware also benefits from these common techniques while generating decryption keys.

2.4 Probabilistic execution and time-bombs

Probabilistic execution, when applied correctly, can obfuscate a malware and produce junky code when being emulated. The probability of correct execution can be decreased to evade detection but it will also effect the spreading speed of the malware. Time-bombs use a similar technique like probabilistic execution, but instead of depending

[2]http://www.trapkit.de/research/vmm/scoopydoo/index.html
[3]http://www.invisiblethings.org/papers/redpill.html

on a probability they are implemented to execute whenever an event triggers. Most common usage is setting the malware to execute on a specific date or time. Setting up a cryptocounter to implement a timebomb can enable a malware to execute its malicious instructions without revealing the exact time that triggers the execution(Young & Yung, 2004). We will now propose a way to call APIs without revealing any information to the observer. This API call technique can also be used to check events that trigger timebombs without revealing the event itself.

3. Malicious Cryptography Usage

Malwares generally make use of native APIs which are detected by heuristics on static analysis since import table of the executables contains a list of APIs which will be called during execution. Most APIs are called by loading an external library into memory and calling the function with the name and correct parameters. APIs give information about the behavior of the program and a malware author should obfuscate most of those API calls to evade detection. Instead of supplying the address or the name of the API, a malware author can calculate the hash value for an API which is going to be called on the runtime. After loading the external library into memory, a malware can start from the first API from the loaded library and calculates hash values of APIs until a match occurs with the hash stored in itself.

This allows malware to hide any information about the APIs that will be called upon execution. To make detection harder, a malware can encrypt the hash function and decrypt it whenever neccessary.It can also corrupt hash values after a matching occurs.

4. Cryptoware

Using real execution-emulation differences and cryptography we developed a tool that makes a malware completely undetectable to anti-viruses. The tool "Cryptoware" first encrypts the malware and adds a stub part at the beginning of the file like described in this paper. The stub part contains time execution difference technique and checks for environment variables as described in Section 2. In our experiment we used our tool to process well known malwares such as AgoBot, RxBot, SDBot and SubSeven. Scan results after processing those malwares show that anti-viruses couldn't be able to detect the malicious code. Processsed malwares do not generate any threats for the anti-viruses in both dynamic and static analysis. We test the processed malwares with public virtual environments as well as sandboxes available on the internet. All products we have tested are failed to emulate and detect our sample malwares.[4]

[4]http://www.cryptovirology.org/fig2.png

5. Conclusion and Future Work

In this paper, we address some weaknesses of emulators and how attackers can make use of them. Instead of just detecting if the code is being emulated, we propose a new way to generate decryption keys by using some techniques which will generate different results in a real execution and emulated environment. We show that by using those techniques together, attackers can hide any kind of malware from antiviruses. We also present a tool called 'Cryptoware' which uses techniques that we described in this paper to automatically convert an existing malware, making it undetectable to anti-viruses.

Our future work will be based on developing a new heuristic engine based on partial execution of the program in a real system. In a partial emulated environment we believe that most of the code will still being emulated. However, for other instructions which helps attackers to detect emulation, we will try to execute it without user interaction. This will make emulators more efficient and capable to detect most of the anti-virtual machine methods, thus revealing the malicious code.

References

Aycock, J. (2006). Computer viruses and malware. *Advances in Information Security, Vol. 22*.

Daoud, E. A., Jebril, I. H., & Zaqaibeh, B. (2008). Computer virus strategies and detection methods. *Int. J. Open Problems Compt. Math., Vol. 1, No. 2*.

Quist, D., & Smith, V. (2004). Detecting the presence of virtual machines using the local data table. *Offensive Computing*.

Sun, L., Ebringer, T., & Boztas, S. (2008). An automatic anti-anti-vmware technique applicable for multistage packed malware. *Proceedings of the 3rd International Conference on Malicious and Unwanted Software* (pp. 17–23).

Young, A., & Yung, M. (2004). Chapter 5: Cryptocounters. *Malicious Cryptography: Exposing Cryptovirology*.

Multi-modal Analysis of Dance Performances for Music-Driven Choreography Synthesis

Ferda Ofli FOFLI@KU.EDU.TR
Engin Erzin EERZIN@KU.EDU.TR
Yucel Yemez YYEMEZ@KU.EDU.TR
A. Murat Tekalp MTEKALP@KU.EDU.TR
Electrical and Computer Engineering Department
Koc University

1. Introduction

Choreography is the art of tailoring the sequences of body movements to music in order to embody or express ideas, emotions or even tell a story in the form of a dance performance. Hence, the rhythm and the intensity of body movements in a dance performance are expected to be in synchrony with those of the music. Nevertheless, different arrangements can accompany the same musical piece, and yet, create different choreographies. This exemplifies the many-to-many nature of the relations between music and dance. To account for this many-to-many nature, we define the "exchangeable figures" as the group of dance figures that are accompanied by similar musical melodies and hence can be replaced without causing an artifact in the dance performance (choreography). Our main motivation in this study is to build a multi-modal framework that uses the "exchangeable figures" notion for modeling, analysis, and synthesis of alternative dance choreographies that are coherent and compelling to audience.

2. Related Work

Most of the studies in the context of multi-modal music and dance analysis towards dance motion synthesis focuses solely on the synchronization aspect of the problem between an existing animation and a piece of music. Kim et al. use transition graphs to synthesize new motion sequences from motion capture data using the results of motion rhythm analysis (Kim et al., 2003). Shiratori et al. propose a technique to synthesize dance motion that is perceptually matched to music by using a mapping based on the rhythmic similarities between music and motion segments for synchronizing the animation with the song (Shiratori et al., 2006). Sauer and Yang design a music-driven character animation tool which extracts a set of features such as the beat and dynamics (louds and softs) of the music to build an animation from a dictionary of pre-built dance movements specified by the user through a script file (Sauer & Yang, 2009). In an earlier work, we have described an automatic music-driven dance animation scheme based on supervised modeling of music and dance figures in a simplified scenario, where a dance performance is assumed to have only a single dance figure which is to be synchronized with the musical beat (Ofli et al., 2008).

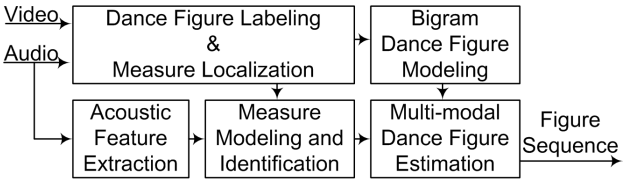


Figure 1. Block diagram of the overall multi-modal dance performance analysis-synthesis framework.

3. System Overview

The overall framework, as depicted in Figure 1, comprises of several blocks that can be grouped into five main tasks: dance figure labeling and measure localization; acoustic feature extraction; measure modeling and identification; bigram dance figure modeling; multi-modal dance figure estimation. The following sections explain these five main tasks in more detail.

3.1 Dance Figure Labeling and Measure Localization

A musical piece is a collection of measures and a measure is a time segment that is defined as the number of beats in a given duration. Since dance figures are performed in synchrony with musical rhythm, the boundaries of dance figures match those of the musical measures. Based on this knowledge, we manually mark the dance figure/measure boundaries and assign labels to different dance figures performed in each music frame, i.e., measure.

3.2 Acoustic Feature Extraction

Chroma features characterize the melodic or harmonic content of music since they represent musical audio by projecting the entire spectrum onto 12 bins corresponding to the 12 distinct semitones of the musical octave. We extract chroma features by following similar approach to the well-known mel-frequency cepstral coefficients (MFCC) calculation. The difference is in how we choose the triangular overlapping windows while calculating the chroma coefficients from the magnitude spectrum of DFT of the audio signal. We basically center the triangular weight windows at the locations of semitone frequencies at different octaves. Then, we take log-average of the harmonics of the calculated semitone coefficients, that gives us 12-bin chroma features.

3.3 Measure Modeling and Identification

We employ hidden Markov models (HMMs) to identify and model the audio measure patterns corresponding to the dance figures. Each HMM is trained over the collection of measures co-occurring with the same dance figure. In other words, each HMM computes $P(F|a)$, i.e., probability of dance figure F given a, acoustic chroma features. Hence, we train as many HMMs as the number of different dance figures that exist in the dance performance.

For measure identification part of this task, we use the trained HMMs to assign figure ids to the sequence of measures extracted from the input music. Instead of identifying each measure with the label of the model that gives the best acoustic score, i.e., the highest likelihood probability of the model match, we create a list of model labels with the highest-N acoustic scores. That is, we generate N alternative transcriptions for each music frame, i.e., musical measure. We then form a lattice, call it $\mathbf{M}$, where the vertical dimension represents the dance figures and the horizontal dimension represents the frames of music (i.e., measures). The entries of $\mathbf{M}$ are the acoustic scores (i.e., likelihood probabilities) of the corresponding models at the corresponding music frames.

3.4 Bigram Dance Figure Modeling

We create a bigram probability matrix, call it $\mathbf{A}$, for the input dance figure sequence to capture the dependency relation of the current dance figure with the previous one. Specifically, each entry in $\mathbf{A}$, namely a_{ij}, is the probability of performing the figure F_j after the figure F_i. This bigram dance figure model provides us with some rules that specify the structure of a dance choreography. For instance, we can enforce a dance figure to always follow a particular figure if it is also the case in the training video with the help of the bigram model.

3.5 Multi-modal Dance Figure Estimation

Using the bigram matrix $\mathbf{A}$ together with the lattice $\mathbf{M}$, we estimate an output dance figure sequence by finding a path along $\mathbf{M}$ in two different ways. In the first one, we follow the *single best path* along $\mathbf{M}$, i.e., the label sequence that has the maximum total likelihood. In the other one, we follow a path in which we pick the *likely* figure, i.e., the figure that is randomly selected according to a predefined distribution, at each music frame.

We employ a Viterbi algorithm to traverse through the columns of $\mathbf{M}$ using $\mathbf{A}$. Recall that an entry in $\mathbf{M}$, namely m_{ij}, represents the likelihood of figure F_i being performed at music frame j. Let N denote the number of rows in $\mathbf{M}$ (which is also the number of different dance figures); T denote the number of columns in $\mathbf{M}$ (which is also the total number of measure frames); and $\phi_j(t)$ represent the partial likelihood score of performing dance figure F_j at frame t along a single path that accounts for the highest partial likelihood from frame 1 to frame t. This partial likelihood can be computed efficiently using the following recursion:

$$\phi_j(t) = \max_i\{\phi_i(t-1)a_{ij}\}m_{jt}. \tag{1}$$

At time t, each partial likelihood score $\phi_j(t-1)$ is known for all dance figures F_j, hence Equation 1 can be used to compute $\phi_j(t)$ thereby extending the partial paths by one music frame. We also define a structure $\psi_j(t)$ to keep track of the argument which maximizes Equation 1, for each j and t, in order to retrieve the dance figure sequence. The overall algorithm for finding the single best dance figure sequence can be summarized as follows:

1. Initialization:

$$\begin{array}{ll} \phi_j(1) = m_{j1}, & 1 \le j \le N \\ \psi_j(1) = 0, & 1 \le j \le N \end{array} \tag{2}$$

2. Recursion:

$$\begin{array}{ll} \phi_j(t) = \max_i\{\phi_i(t-1)a_{ij}\}m_{jt}, & 2 \le t \le T \\ & 1 \le j \le N \\ \psi_j(t) = \mathrm{argmax}_i\{\phi_i(t-1)a_{ij}\}, & 2 \le t \le T \\ & 1 \le j \le N \end{array} \tag{3}$$

3. Termination:

$$\begin{array}{l} \Phi = \max_i\{\phi_i(T)\} \\ \Psi(T) = \mathrm{argmax}_i\{\phi_i(T)\} \end{array} \tag{4}$$

4. Path (dance figure sequence) backtracking:

$$\Psi(t) = \psi_{\Psi(t+1)}(t+1), \quad t = T-1, T-2, \ldots, 1 \tag{5}$$

Even though this procedure is designed for the first synthesis scenario, i.e., picking the *single best path* along $\mathbf{M}$, we can easily modify it for the second synthesis scenario, i.e., picking a *likely path* along $\mathbf{M}$. Instead of picking the maximum in Equation 1, we can randomly pick one of the 'likely' dance figures according to a prespecified distribution P. It is also necessary to update the recurrence relation for $\psi_j(t)$ accordingly.

4. Experiments and Results

In this study, we investigate the Turkish folk dance, *kasik*[1]. Our audiovisual database is 36 minutes long and consists of 20 dance performances with 20 different musical pieces. There are 31 different dance figures (i.e., $N = 31$) and a total of 1265 musical measure segments (i.e., $T = 1265$).

We define the following five assessment levels to evaluate each figure label F_s in the synthesized figure sequence compared to the respective figure label F_a assigned by the expert:

- $L0$: F_s is marked as $L0$ if F_s matches F_a.
- $L1$: F_s is marked as $L1$ if F_s does not match F_a, but it is in one of the expert-specified exchangeable figure groups together with F_a.
- $L2$: F_s is marked as L_2 if F_s does not match F_a, and it is not in one of the expert-specified exchangeable groups together with F_a, either. However, F_s and F_a are performed with the same musical piece.
- $L3$: F_s is marked as L_3 if F_s and F_a should not be performed with the same musical piece, and yet, they are exchanged due to a recognition error possibly because the musical pieces with which they are actually performed have similar rhythmic audio patterns.
- $L4$: F_s is marked as L_4 if it is not marked as one of $L0$ through $L3$.

We also associate a penalty score ranging from 0 to 4 with the levels $L0$ through $L4$, respectively. Then, we calculate an overall penalty score for measuring the 'goodness' of the resulting dance choreography. For both synthesis scenarios, Figure 2 compares the number of figures that fall into each assessment level both for the recognition and the synthesis label sequences. The penalty score for the output figure sequence of the first scenario is 911 whereas it is 2033 for the output figure sequence of the second scenario.

Looking at Figure 2 from another point of view, we see that among all the assessment levels, $L0$, $L1$ and $L2$ are indicators of the diversity of alternative dance figure choreographies rather than being an indicator of error. $L3$ and $L4$, however, can be perceived as indicators of error in the dance choreography synthesis process. In this context, we see that around 94% of the synthesized figures fall into one of the first three assessment levels in the first synthesis scenario. This percentage drops to about 74% for the dance figure sequence of the second synthesis scenario, which is still a high percentage of the entire dance sequence.

[1] *Kasik* means *spoon* in English. The dance is named so because the dancers clap spoons while dancing.

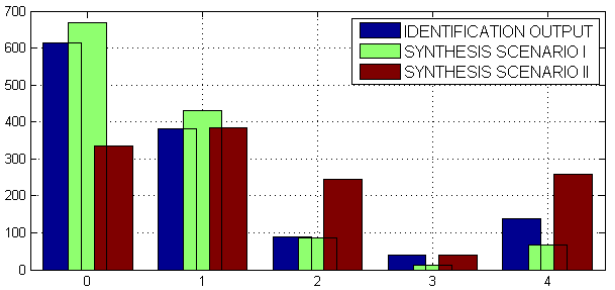


Figure 2. The number of figures that fall into each assessment level for the recognition and the synthesis label sequences in the proposed two synthesis scenarios.

5. Conclusions

In this paper, we propose a mapping from music measures to dance figures based on correlations between dance figures and music measures as well as correlations between successive dance figures, in terms of figure-to-figure transition probabilities. We, then, use this mapping to synthesize a music-driven sequence of dance figure labels. The output sequence of dance figure labels can be considered as a dance choreography that is in synchrony with the driving audio signal. The experimental results show that the proposed framework is successful at creating *acceptable* alternative dance choreographies.

References

Kim, T.-h., Park, S. I., & Shin, S. Y. (2003). Rhythmic-motion synthesis based on motion-beat analysis. *ACM Trans. Graph.*, *22*, 392–401.

Ofli, F., Demir, Y., Erzin, E., Yemez, Y., Tekalp, A. M., Balci, K., Kiziloglu, I., Akarun, L., Canton-Ferrer, C., J., T., Bozkurt, E., & Erdem, A. (2008). An audio-driven dancing avatar. *Journal on Multimodal User Interfaces*, *2*, 93–103.

Sauer, D., & Yang, Y.-H. (2009). Music-driven character animation. *ACM Trans. Multimedia Comput. Commun. Appl.*, *5*, 1–16.

Shiratori, T., Nakazawa, A., & Ikeuchi, K. (2006). Dancing-to-music character animation. *COMPUTER GRAPHICS FORUM*, *25*, 449–458.

Spatial Filtering for Encoding Multi-view Video in Spatially Reduced 3D Displays

Göktuğ Gürler cgurler@ku.edu.tr
Electrical Engineering Departmant, Koc University

1.Introduction

3D video is becoming popular. Key technologies that are required for 3D systems such as stereo acquisition and display have already been developed along with the streaming applications. Today it is possible to use advanced telepresence applications in which the participant can see each other in 3D at HD resolution. Therefore, it would become an important branch of video industry thus efficient transmission of MVV it is a hot research area.

One of the major challenges in streaming multi-view video is the increase in bandwidth requirements due to transmission of additional view(s). Up to now, researches addressing this problem have mainly focused around three solutions: i) Using multi-view video codec such as MVC extension of H.264/AVC for exploiting the redundancy among views. This work has been initialized in 2006 and the final draft proposed in 2009. (Vetro et al. 2008) ii) Using depth map in addition to the video signal to generate the artificial view. (Fehn. 2008) iii) Exploiting the features of human visual system (HSM) that allows degrading visual quality in one of the view i.e., removing high frequency components, without introducing noticeable artifacts. This is commonly implemented by performing asymmetric coding among views. (Ozbek et al. 2008; Fehn et al. 2007)

We address the same problem for spatially reduced stereoscopic displays in which the effective resolution of the 3D content is reduced to accommodate both views in a single video frame. We propose a novel pre-encoding procedure in which we identify and filter out the pixels that will be removed from the scene due to spatial resolution reduction. Since these pixels do not have a contribution in the rendering process, this operation does not degrade the perceived 3D quality. We have also computed the average gain in bitrate by testing the proposed filtering over multiple stereo contents. Moreover, we propose backward compatible modifications in real-time streaming protocol (RTSP) (Schulzrinne et al. 1998) to notify the type of display system to server for taking advantage of the proposed method.

Rest of this paper is organized as follows: In section two we provide brief information about spatially reduced 3D display systems. Following that, we define the interzigging process which is required for achieving stereoscopy in spatially reduced display systems. In section 4 we explain the proposed filtering operation in detail. In section 5 we define modifications for RTSP protocol. In section 6 we provide the achieved gain in compression of MVV using the proposed method for spatially reduced displays. And finally in section 7 we draw our conclusions.

2. Overview of Stereoscopic Display Systems

The stereoscopy is based on projecting the correct view to the corresponding eye and avoiding exposure to wrong view through a filtering mechanism. When successfully implemented, viewers experience a sense of depth and perceive objects differently based on their location in the scene thus it becomes possible to distingue objects that are closer to the screen from the ones that are far away.

2.1 Spatially Reduced Stereoscopic Display Systems

It is possible to experience 3D without wearing special glasses in spatially reduced display systems while in most of the full resolution 3D systems require special glasses to filter or block the non-corresponding view for each eye. There are two alternative technologies for achieving unaided stereoscopy and both of them requires merging sub-pixel values (red, green and blue) of views in a specific order which is known as interzigging or interdigitizing pattern.

In lenticular sheet technology 2D lens array is placed on a LCD screen. When an interzigged image is displaced the lenses direct lights of each view in a certain direction. In parallax barrier technology lights that compose one of the views are blocked for certain direction and create a similar result. In both methods the spatial resolution of the content is halved the viewer has to be located at the correct position which is called the sweet spot.

3. Interzigging Pattern

In spatially reduced displays, the interzigged image contains samples from left and right views at sub-pixel level. However half of the values in each view are discarded in interzigged process. Figure 1 depicts the required sub-pixel samples from each view for the first 3x3 region and can be interpreted as follows; In order to generate the pixel of the interziged image at x=1 (left most), y=1 (top most) position which is highlighted with bold border, red and blue samples are taken from the left view while the green sample is taken from the right view. Similarly, for the pixel located at x=2, y=1, which is highlighted with dashed bold border, red and blue samples are taken from the right view and the green sample is taken from left view. In this figure 'L' refers to left image and 'R' refers to right image and the subscripts define the coordinate of the sub-pixel values on the original images.

x = **1**			x = 2			x = **3**		
Red	Grn	Blue	Red	Grn	Blue	Red	Grn	Blue
$L_{1,1}$	$R_{1,1}$	$L_{1,1}$	$L_{2,1}$	$R_{2,1}$	$L_{2,1}$	$L_{3,1}$	$R_{3,1}$	$L_{3,1}$
$L_{1,2}$	$R_{1,2}$	$L_{1,2}$	$L_{2,2}$	$R_{2,2}$	$L_{2,2}$	$L_{3,2}$	$R_{3,2}$	$L_{3,2}$
$L_{1,3}$	$R_{1,3}$	$L_{1,3}$	$L_{2,3}$	$R_{2,3}$	$L_{2,3}$	$L_{3,3}$	$R_{3,3}$	$L_{3,3}$

Figure 1: Interzigging Pattern for Stereoscopic Display

4. Proposed Filters

Significant reduction in bitrate can be achieved by transmitting only the pixels that are used for generating the interzigged image. Following this argument one trivial solution may seem as encoding the interzigged image and then transmitting the resultant sequence as a monoscopic (single view) video. However, the 3D experience is lost when the interzigged image is encoded using a block-based encoder such as H.264/AVC. The problem with this approach is that, the interzigged image contains very high frequency components which are suppressed during encoding due to block based quantization. As a remedy we propose to split the sub-pixels in a way that we can achieve two separate images with less high frequency components. In the following subsections we define two possible remapping schemes for this purpose.

4.1 Filter 1: Even/Odd Seperation

A frame that is composed of only even or odd pixels of the interzigged image has less high frequency components. This is due the fact that in interzigged image the odd pixels (x-axis) are dominated by left view and even frames are dominated by right view because those views determine two out of three sub-pixel values. Therefore, a filter that separates interzigged images can be used prior to encoding for obtaining two sequences with less high frequency components. Figure 3 the sub-pixel maps for sequence and provides sample frames after such operation.

4.2 Filter 2: Left / Right Seperation

There is a ghostly artifact in the output video sequences due to the fact that the red and blue sub-pixel is determined by one view while the green sub-pixel is determined by the other view. In Filter 2 we swap the green sub-pixels are yield two sequences in which one of the views are completely determined by left view and vice versa. This fix removes the ghostly artifact and corrects the sub-pixel color mismatch. This also increases the compression efficiency of the output sequences.

5. Current 3D Streaming Protocols and Proposed Modifications

If the client's interzigging pattern is signaled to the server then it is possible to transmit the media using a sequence with lower bandwidth requirement. However, current RTSP standard does not define a step for transmitting this information. RTSP is a client driven messaging protocol and it is designed in a flexible way to allow extensions. In (Kurutepe et al. 2007) authors have proposed modifications that allow additional data exchange for 3D video streaming purposes.

We extend these modification by using modified DESCRIBE message in which the client signals the interzigging pattern. In the standard, a client sends DESCRIBE message which is usually replied in session description protocol (SDP) (Handley et al. 2008) in order to learn some key features about the content such as type of the codec used in compression and spatial resolution constraints. Based on the reply from the client may proceed to initialize the session or may end it gracefully. We propose adding a new field to DESCRIBE request that includes the interzigging pattern at the client side. Using this field a server may reply differently and differentiate a spatially reduced display. This modification is compliant with the 'Extending RTSP' section of [6] because a standard server may simply ignore the extra field. There are two common fundamental interzigging patterns are available. The pattern in Figure 1 is a horizontal pattern in the sense that reference view for the sub-pixels remains same in the horizontal axis. Figure 2 presents the modified DESCRIBE message format, and the bold text is the proposed modification.

```
DESCRIBE
rtsp://server.example.com/fizzle/foo
RTSP/1.0
CSeq: #
Accept: application/sdp
Interzig: Horizontal
```

Figure 2: Modified DESCRIBE message from client

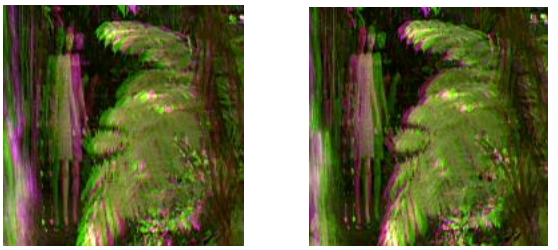
x = 1			x = 2			x = 3		
Red	Grn	Blue	Red	Grn	Blue	Red	Grn	Blue
$L_{1,1}$	$R_{1,1}$	$L_{1,1}$	$L_{3,1}$	$R_{3,1}$	$L_{3,1}$	$L_{5,1}$	$R_{5,1}$	$L_{5,1}$
$L_{1,2}$	$R_{1,2}$	$L_{1,2}$	$L_{3,2}$	$R_{3,2}$	$L_{3,2}$	$L_{5,2}$	$R_{5,2}$	$L_{5,2}$
$L_{1,3}$	$R_{1,3}$	$L_{1,3}$	$L_{3,3}$	$R_{3,3}$	$L_{3,3}$	$L_{5,3}$	$R_{5,3}$	$L_{5,3}$

a) Video Sequence 1

x = 1			x = 2			x = 3		
Red	Grn	Blue	Red	Grn	Blue	Red	Grn	Blue
$R_{2,1}$	$L_{2,1}$	$R_{2,1}$	$R_{4,1}$	$L_{4,1}$	$R_{4,1}$	$R_{6,1}$	$L_{6,1}$	$R_{6,1}$
$R_{2,2}$	$L_{2,2}$	$R_{2,2}$	$R_{4,2}$	$L_{4,2}$	$R_{4,2}$	$R_{6,2}$	$L_{6,2}$	$R_{6,2}$
$R_{2,3}$	$L_{2,3}$	$R_{2,3}$	$R_{4,3}$	$L_{4,3}$	$R_{4,3}$	$R_{6,3}$	$L_{6,3}$	$R_{6,3}$

b) Video Sequence 2

Figure 3: Pixel distribution for Filter 1



a) Video Sequence 1 b) Video Sequence 2

Figure 4: Samples images for Filter 1

x = 1			x = 2			x = 3		
Red	Grn	Blue	Red	Grn	Blue	Red	Grn	Blue
$L_{1,1}$	$L_{2,1}$	$L_{1,1}$	$L_{3,1}$	$L_{4,1}$	$L_{3,1}$	$L_{5,1}$	$L_{5,1}$	$L_{5,1}$
$L_{1,2}$	$L_{2,2}$	$L_{1,2}$	$L_{3,2}$	$L_{4,2}$	$L_{3,2}$	$L_{5,2}$	$L_{5,2}$	$L_{5,2}$
$L_{1,3}$	$L_{2,3}$	$L_{1,3}$	$L_{3,3}$	$L_{4,3}$	$L_{3,3}$	$L_{5,3}$	$L_{5,3}$	$L_{5,3}$

a) Video Sequence 1

x = 1			x = 2			x = 3		
Red	Grn	Blue	Red	Grn	Blue	Red	Grn	Blue
$R_{2,1}$	$R_{3,1}$	$R_{2,1}$	$R_{4,1}$	$R_{5,1}$	$R_{4,1}$	$R_{6,1}$	$R_{7,1}$	$R_{6,1}$
$R_{2,2}$	$R_{3,2}$	$R_{2,2}$	$R_{4,2}$	$R_{5,2}$	$R_{4,2}$	$R_{6,2}$	$R_{7,2}$	$R_{6,2}$
$R_{2,3}$	$R_{3,3}$	$R_{2,3}$	$R_{4,3}$	$R_{5,3}$	$R_{4,3}$	$R_{6,3}$	$R_{7,3}$	$R_{6,3}$

b) Video Sequence 2

Figure 5: Pixel distribution for Filter 2

a) Video Sequence 1 b) Video Sequence 2

Figure 6: Samples images for Filter 2

5. Conclusions

The decrease in bandwidth requirement when the proposed filters are utilized is summarized in Table 1 for various stereo contents. On the average Filter 1 and Filter 2 decreases the bandwidth requirement by %30 and %35 percent respectively. The peak-signal-to-noise-ratio (PSNR) is a qualitative metric based on squared-mean-error and shows the quality of a compressed sequence. The results reveal that using the proposed filter causes 0.2dB loss in quality on the average. However, up to ~1.0dB change in PSNR is not noticeable. Therefore, it is clear that the proposed methods can be safely used for the transmission of 3D content for spatially reduced display systems.

References

Vetro, A., Pandit, P., Kimata H., Smolic A. &. Wang Y. (2008). Joint draft 8.0 on multiview video coding *Joint Video Team, Doc. JVT-AB204*,.

Fehn, C. (2004). Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3 D-TV. *Proceedings of SPIE*,vol. 5291, pp. 93–104,.

Ozbek, N., & Tekalp, A. M. (2008). Unequal inter-view rate allocation using scalable stereo video coding and an objective stereo video quality measure. in *Proc. Of Int. Conf. on Multimedia and Expo,* Germany

Fehn C., Cho P. K., Kwon H., Hur N., Kim J., (2007). Asymmetric coding of stereoscopic video for transmission over T-DMB in *Proc. of 3DTV-CON*, Kos, Greece

Schulzrinne, H., Rao A., & Lanphier R., *Real time streaming protocol (rtsp) (1998).* [Online]. Available: http://www.ietf.org/rfc/rfc2326.txt

Kurutepe, E., Aksay A., Bilen C., Gurler CG., Sikora T., Akar G.B., Tekalp A.M., (2007). A Standards-Based, Flexible, End-to-End Multi-View Video Streaming Architecture. Packet Video Workshop 2007, Lausanne, Switzerland

Handley, M., & Jacobson, V. SDP (1998). Session description protocol.
[Online] Available: http://www.ietf.org/rfc/rfc2327.txt

Haplotype Inference with Polyallelic and Polyploid Genotypes

Ozan Erdem OZANERDEM@SABANCIUNIV.EDU
Sabanci University, Orhanli-Tuzla, Istanbul, Turkey

1. Introduction

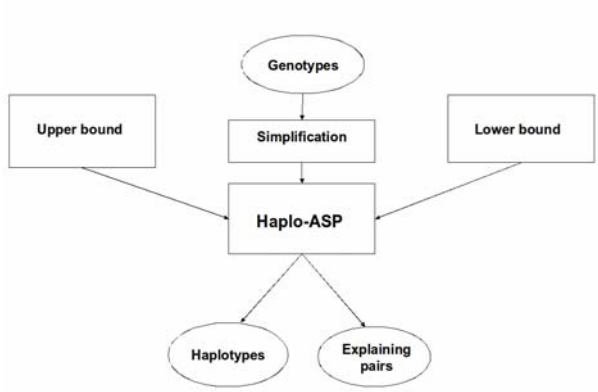
Each genotype has several copies, which are called haplotypes, and they combine to form the genotype. The genetic information contained in haplotypes can be used for early diagnosis of diseases, detection of early transplant rejection, and creation of evolutionary trees. However, although it is easier to access the genotype data, due to technological limitations, determining haplotypes experimentally is a costly and time consuming procedure. With these biological motivations, researchers have been studying haplotype inference–determining the haplotypes that form a given set of genotypes– by means of some computational methods.

One haplotype inference problem that has been extensively studied is Haplotype Inference by Pure Parsimony (**HIPP**) (Gusfield, 2003). This problem asks for a minimal set of haplotypes that form a given set of genotypes; the decision version of **HIPP** is NP-hard (Gusfield, 2003; Lancia et al., 2004). **HIPP** has been studied with various approaches, such as HYBRIDIP (based on integer linear programming) (Brown & Harrower, 2006), HAPAR (based on a branch and bound algorithm) (Wang & Xu, 2003), SHIPS (based on a SAT-based algorithm) (Lynce & Marques-Silva, 2006), RPOLY (based on pseudo-boolean optimization methods) (Graça et al., 2007), and HAPLO-ASP (based on Answer Set Programming) (Erdem & Türe, 2008).

However, these systems consider only biallelic and diploid genotypes; where each site of a genotype can be one of the two allele types, and each genotype is composed of exactly two haplotypes. In this paper, we also considered polyallelic and polyploid genotypes; where each site of a genotype can be one of the four allele types, and each genotype is composed of up to four haplotypes. We call the problem of finding a minimal set of haplotypes that form a given set of polyallelic and polyploid genotypes as Haplotype Inference with Polyallelic and Polyploid Genotypes (**HIPPG**). **HIPPG** has been previously studied by (Neigenfind et al., 2008), which is based on a SAT-based algorithm. In this paper, we introduce a novel approach to solving **HIPPG**, using Answer Set Programming; and extend the HAPLO-ASP system with it.

ASP is a declarative programming paradigm that provides a highly expressive language for knowledge representation, and efficient solvers for automated reasoning. The idea of ASP is to represent a computational problem as a "program" whose model (called "answer sets") correspond to the solutions of that problem, and to compute the answer sets for the program using an "answer set solver", like CLASP (Gebser et al., 2009b), after "grounding" the program, e.g., by the "grounder" GRINGO (Gebser et al., 2009a). (See (Baral, 2003) for more information about ASP.) Our system architecture is shown in Figure 1.

Figure 1. HAPLO-ASP system architecture



As for computations, we have experimented with cultivated potato genotypes (*Solanum Tuberosum*) to solve **HIPPG** using HAPLO-ASP, and compared our results with (Neigenfind et al., 2008). We were able to obtain the same solutions with the system of (Neigenfind et al., 2008), SATLOTYPER.

2. Haplotype Inference with Polyploid and Polyallelic Genotypes

As each site in the genotypes correspond to a single nucleotide polymorphism (SNP), the possible values for an allele is a nucleotide from the set $\{A, C, G, T\}$. We view each genotype as a vector of tuples where each value of the tuples is from the set $\{0, 1, 2, 3, ?\}$. Each num-

ber in the set $\{0, 1, 2, 3, ?\}$ correspond to an allele from $\{A, C, G, T\}$, and ? corresponds to an unknown allele. Similarly, we view each haplotype as a vector of alleles where each allele is from the set $\{0, 1, 2, 3\}$. For instance, $(0, 3, 1), (2, ?, ?), (?, 1, 2)$ is a genotype and 021 is a haplotype with three sites. Additionally, we denote j'th site of the i'th genotype as g_{ij} and the j'th site of the i'th haplotype as h_{ij}.

We say that two alleles i and j are compatible if they are identical or if one of them is ?. A set $H = \{h_1, h_2...h_r\}$ of haplotypes is compatible with a genotype g_i at site j if every allele in g_{ij} is compatible with a different haplotype from H at site j. A genotype g_i is explained by a set H of haplotypes if H is compatible with g_i at each site. For instance, the haplotype set $\{011, 023, 211\}$ is compatible with the genotype $(0, 2, 0), (?, 2, 1), (3, ?, ?)$.

We consider the decision version of **HIPPG**:

HIPPG-DEC Given a set G of n genotypes each with m polyallelic sites, an integer p which denotes ploidity, and a positive integer k, decide whether each genotype in G can be explained by a subset of cardinality p of a haplotype set H containing at most k unique haplotypes.

For sufficiently small k, a solution to **HIPPG-DEC** is a solution to **HIPPG** as well.

To solve **HIPPG-DEC** we assume the following:

A1 H is a set that contains $p * n$ haplotypes, $h_1, \ldots, h_{p*n}$, and

A2 f maps every genotype g_i in G to p haplotypes, $h_{p*i}, h_{p*i-1} \ldots h_{(p-1)*i+1}$, in H.

Then, for this problem, H is a solution if the following hold:

C1 Every genotype g in G is mapped by f to a set of haplotypes which explains g.

C2 There are at most k unique haplotypes in H.

2.1 Representing HIPPG-DEC in ASP

Many answer set solvers, like CLASP, have the same input language of GRINGO; we present our formulation of **HIPPG-DEC** in the input language of GRINGO.

We describe the value v of the allele a of j'th site of a genotype i by atoms of the form $g(i, j, a, v)$. Similarly, we describe the value v of the j'th site of a haplotype i by atoms of the form $h(i, j, v)$. For instance, Genotype 4 with sites $(0, 2, 0), (3, 2, 1)$ and Haplotype 5 with sites 1023 are described by the atoms:

```
g(4,1,1,0). g(4,1,2,2). g(4,1,3,0).
g(4,2,1,3). g(4,2,2,2). g(4,2,3,1).
h(5,1,1). h(5,2,0). h(5,3,2). h(5,4,3).
```

Suppose that we are given n genotypes, each with m sites. We represent that genotypes are labeled `1..n`, that sites are labeled `1..m`, and that haplotypes are labeled `1..p * n` (due to Assumption A1) where p is the ploidy value of the genotypes, by the following domain predicates:

```
geno(1..n).
site(1..m).
haplo(1..p*n).
```

First of all, we generate $p * n$ haplotypes with j sites each with the following set of rules:

```
1{h(H,J,A) : allele(A)}1 :- haplo(H), site(J).
```

We also keep the counts of each allele in each site. The atoms of the form $count(G, J, A, I)$ are interpreted as: "Site J of genotype G has a total of I alleles of type A":

```
count(G,J,A,I) :- I{g(G,J,P,A): ploidity(P)}I,
   geno(G), site(J), allele(A), ploidity(I).
```

To satisfy Constraint C1, we add the following constraints to our program, which tell that if a site of a genotype G contains I alleles of type A, then at least I of the haplotypes that are mapped to it has value A at that site:

```
:- {h(H,J,A) : haplo(H) :
   (G-1)*p+1 <=H<=G*p}I-1, count(G,J,A,I),
   geno(G), site(J), allele(A), ploidity(I).
```

Moreover, we keep track of the unique haplotypes to satisfy Constraint C2 later on:

```
diffhapp(H1,H2) :- h(H1,J,A1), h(H2,J,A2),
   haplo(H1;H2), H1<H2, site(J),
   allele(A1;A2), A1!=A2.
unique(1).
unique(H) :- H-1 {diffhapp(H1,H):haplo(H1)},
   haplo(H), H>1.
```

Then, to satisfy Constraint C2, we add the following constraints:

```
:- k+1 {unique(H):haplo(H)}.
```

We also add some constraints to enforce a lexicographic order among the haplotypes in order to perform symmetry breaking.

2.2 Solving HIPPG using an Answer Set Solver

An instance of **HIPPG** can be solved with the ASP program above, by trying various values for k (the number of unique haplotypes explaining the given genotypes). We compute an approximate lower bound l and an approximate upper bound u for k, and find the optimal k value by doing a binary search between l and u. The system architecture can be seen from Figure 1.

3. Experimental Results

We have extended the HAPLO-ASP system, with a PERL script which includes upper bound computations and system calls to answer set solvers, which can be observed from Figure 1. The performance of HAPLO-ASP was tested using SNP data from a cultivated potato species, *Solanum Tuberosum*, which was obtained from (Neigenfind et al., 2008).

The *Solanum Tuberosum* SNP data is tetraallelic and tetraploid, meaning that each genotype can contain sites with four different alleles and each genotype is mapped to four haplotypes. This data contains 19 genotypes, each with 12 sites. In our experiments, we have found the solution to **HIPPG** with 12 haplotypes. When we instruct an answer set solver to compute all the answer sets for 12 haplotypes, we can compute a total of 114 different solutions to **HIPPG**. Although HAPLO-ASP is slower, it is able to compute all the solutions SATLOTYPER finds for this dataset.

4. Conclusion

We presented an ASP based approach to **HIPPG**, which is capable of solving a broader range of haplotype inference problems than many of the existing haplotype inference systems.

While solving **HIPPG** using ASP, we have intoduced new formulations of these problems, and we were able to compute the results that were previously obtained for the *Solanum Tuberosum* data we used.

Our future work includes enhancing our formulations to decrease our computation times. Moreover, increasing the accuracy rate of the inferred haplotypes with respect to the haplotypes which are found in biological experiments is a crucial part of our ongoing work.

5. Acknowledgements

My deepest thanks to my advisor, Esra Erdem. I also thank Jost Neigenfind and Gabor Gyetvai for answering my questions.

References

Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Brown, D., & Harrower, I. (2006). Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Transactions on Bioinformatics and Computational Biology*, *3*, 348–359.

Erdem, E., & Türe, F. (2008). Efficient haplotype inference with answer set programming. *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence* (pp. 436–441). Chicago, Illinois: AAAI Press.

Gebser, M., Kaminski, R., Ostrowski, M., Schaub, T., & Thiele, S. (2009a). On the input language of asp grounder gringo. *LPNMR '09: Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (pp. 502–508). Berlin, Heidelberg: Springer-Verlag.

Gebser, M., Kaufmann, B., & Schaub, T. (2009b). The conflict-driven answer set solver clasp: Progress report. *LPNMR '09: Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (pp. 509–514). Berlin, Heidelberg: Springer-Verlag.

Graça, A., Marques-Silva, J. P., Lynce, I., & Oliveira, A. (2007). Efficient haplotype inference with pseudo-boolean optimization. *Proc. of Algebraic Biology*.

Gusfield, D. (2003). Haplotype inference by pure parsimony. *Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching (CPM03)* (pp. 144–155).

Lancia, G., Pinotti, M. C., & Rizzi, R. (2004). Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms. *INFORMS Journal on Computing*, *16*, 348–359.

Lynce, I., & Marques-Silva, J. (2006). Efficient haplotype inference with boolean satisfiability. *AAAI*.

Neigenfind, J., Gyetvai, G., Basekow, R., Diehl, S., Achenbach, U., Gebhardt, C., Selbig, J., & Kersten, B. (2008). Haplotype inference from unphased snp data in heterozygous polyploids based on sat. *BMC Genomics*, *9*, 356.

Wang, L., & Xu, Y. (2003). Haplotype inference by maximum parsimony. *Bioinformatics*, *19*, 17731780.

Air Drums: A Computer Vision Based Drum Simulator

Kaan C. Fidan† KAANCFIDAN@SABANCIUNIV.EDU
İhsan Kehribar† KEHRIBAR@SU.SABANCIUNIV.EDU
M. Tuğçe Şahin† MERVETUGCE@SU.SABANCIUNIV.EDU
Serhan Coşar† SERHANCOSAR@SU.SABANCIUNIV.EDU
Devrim Ünay‡ DEVRIM.UNAY@BAHCESEHIR.EDU.TR

†Computer Vision and Pattern Analysis Laboratory, Sabanci University, Orhanli, Tuzla, İstanbul, Turkey

‡Electrical and Electronics Engineering, Bahcesehir University, İstanbul, Turkey

1. Introduction

The aim of this paper is to present a novel system which tracks the motion of a drummer and generates the corresponding drum sounds. Only a camera, some colored markers and an everyday PC are used in the development of the system. The input video sequence from the camera is processed in real-time by using local and adaptive color segmentation and Kalman filter based tracking. The Kalman filter is used to predict the ”hits” so that we can overcome the processing delays and provide a more-realistic drumming experience. We use a local and adaptive search to detect the effective points of the drum sticks, which ensures robustness to background clutter and reduces the computational burden. We developed a working demo and evaluated its performance by comparing with the output signal of an electronic drum pad. We observed that the timing errors have an average of -8.4 ms and a standard deviation of 5.4 ms, where the two extreme values were -22.9 and 3.2 ms in a real drumming experiment consisting of 121 hits.

2. Related Work

The aim of the project is to create an ”edutainment” opportunity in an easily achievable system. The resulting human-machine interface from this work can be used as a way to improve training sessions of the drummers as well as entertainment purposes. There has been some research conducted about tracking drumsticks in the search for new media for educating the next generation in specialized skills. The described system requires previously recorded videos of qualified drummers performing some basic training sets, then the videos are processed and the captured motions are parameterized for future comparison to the students' results (Tansuriyavong et al., 2006). The advantage of our system is its real-time tracking of the drumsticks in order to create a more realistic drumming experience.

Another field of research is focused on audio-visual processing and musical transcription of the drumming performances. These works exploit visual information of the drumsticks and the drums together with the audio of the performance (Gillet & Richard, 2005; McGuinness et al., 2007). To the contrary, our system simulates an imaginary drumset and generates the audio from the visual data.

3. Materials and Methods

3.1 System Overview

Initially our system was planned to work with an everyday webcam, however the nature of the drumstick motions requires processing at high frame rates. The discretization becomes too steep in low frame rates to approximate velocities and accelerations of the tips. Therefore in this work we employ an IDS uEye 1640 camera, which can provide up to 100 frames-per-second (fps) at 320x256 resolution in decent lighting conditions.

For drumstick tracking and hit detection, our system, which is fully implemented in C#, employs the computer vision algorithms presented in the OpenCV library (Bradski, 2000) through the use of the EmguCV wrapper (http://www.emgu.com/wiki). Following the hit detection step, we generate the corresponding MIDI signals that can be picked up through a virtual MIDI cable by any audio processing tool.

The algorithm workflow can be seen in Figure 1. The system only needs clicks of the user on the tips of the drumsticks to initiate the segmentation process.

3.2 Drumstick Segmentation

The segmentation is carried out in HSV space. The hue channel is thresholded within a small tolerance (all the other pixels which do not belong in the range *picked hue*

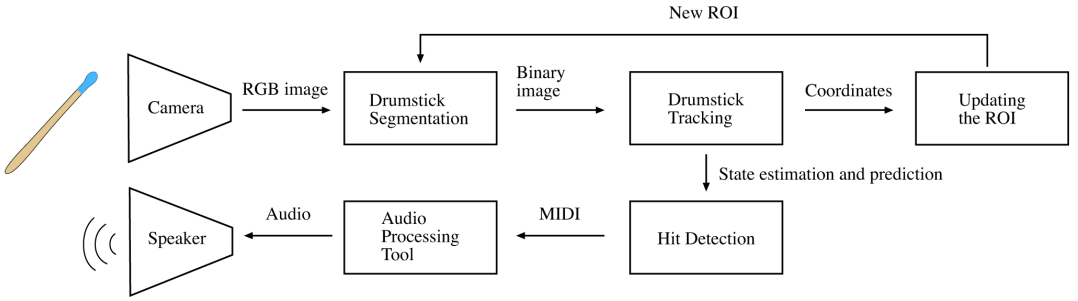


Figure 1. Algorithm Block Diagram

$\pm$*tolerance* are suppressed, chosen pixels are marked with 255). The tolerance for the saturation and the value is tuned by the user and processed in the same manner. The saturation and the value tolerance covers the shadows and bright spots on the colored marker. The intersection between the hue, saturation and the value thresholded images contains only the colored marker. The *gravity center* of the binary image is calculated.

3.3 Drumstick Tracking

We used *Kalman filter* for tracking. The main motives are:

✦Reducing the effects of instantaneous measurement ripples caused by lighting variations on the drumstick.

✦Predicting the ”hits” before they happen for the sound to be generated at the moment of the actual hit.

In our case, Kalman filter works with state vectors which contain the position, the velocity and the acceleration on X and Y coordinates. The measurement is given as the position of the gravity center. Kalman filter uses this information to approximate the velocity and the acceleration, correct the current measurement using noise models and predict the next state (Kalman, 1960).

3.4 Updating the Region Of Interest

The ROI is the window where the segmentation is done in each cycle. It is crucial for reducing the computational weight and increasing the number of processed frames, hence increasing the number of data points. Once the gravity center is defined, a window is formed around it. If the predicted position of the tip gets out of the window, the window gets larger in proportion with the velocity for not losing the tip. When the tip is stable, the window turns to its initial size. If the tip is lost (i.e falls outside the window), the ROI is canceled and whole image is processed until it is found again.

The Fig.2 shows a captured image from the working software. The drumstick is modeled with a boardmarker and it is accelerating downwards at the moment. In the binary image, red point marks the current gravity center and the blue point, the predicted position. The white frame is the ROI, which is expanding to include the next state prediction.

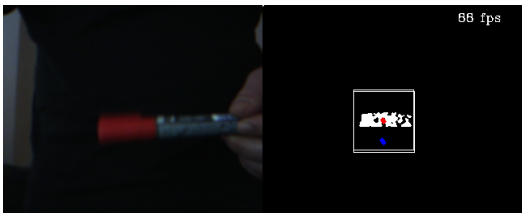


Figure 2. Segmentation and tracking of the drumstick tip.

3.5 Hit Detection

The real-time goal brings the difficulty of predicting the hit without the proper information. Hence, the hit detection algorithm includes a series of assumptions:

✦ *Vertical acceleration should be negative and larger than a small threshold.* The downward acceleration tells that the drumstick is gaining speed and getting close to a “hit”. The downward acceleration peaks for a very small amount of time before the hit due to the nature of the specific motion.

✦ *The predicted position should be in a drum zone.* The sound is generated corresponding to different ”drum zones” and thus, where the hit will land is important.

✦ *There cannot be 2 consecutive hits with the same drumstick in 80 ms.* The downward acceleration may appear in several consecutive frames corresponding to the same hitting motion and cause multiple detections. It is assumed that even if the user is able to perform at such speed, the camera would not be able to gather enough data points to correctly detect the hit.

✦ *Volume of the hit is proportional to the current velocity.* In the real world, the volume is proportional to vibration

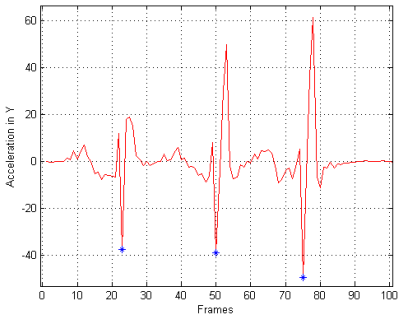


Figure 3. Vertical acceleration of the drumstick tip with the detected "hit" marked by a star.

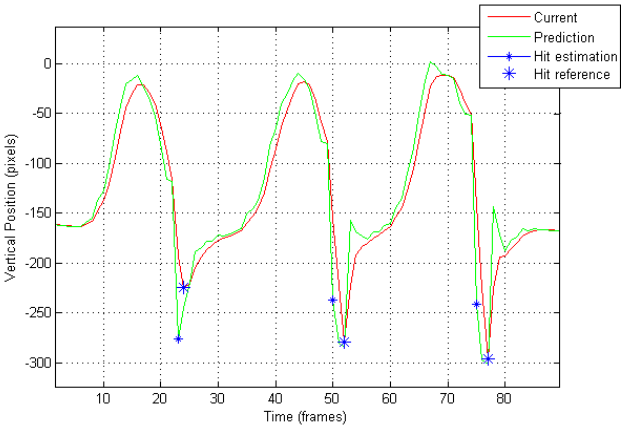


Figure 4. Vertical Position vs. Frames

amplitude of the drumhead caused by the reaction force. The reaction force can be estimated from the acceleration which stops the drumstick with Newton's $F = m.a$. We assume that the current velocity is a measure of how high the reverse acceleration will be when the drumstick stops.

4. Results

Our system currently faces difficulties at robustly tracking the tips of actual drumsticks due to their very fast movements which cause motion blurs at current acquisition settings. We use shorter sticks (i.e. board markers) that have smaller radius of circular motion, hence easier to spot the tips in rigid form. Figures 3 and 4 depict a recording of 3 hits and shows the hit detection algorithm's results.

We have created an experimental setup where we processed the visual data from a drum practice routine consisting of 121 hits on electronic drum pads and compared the resulting MIDI signals. We observed that the timing errors had an average of -8.4 ms and distributed with a standard deviation of 5.4 ms (Fig. 5).

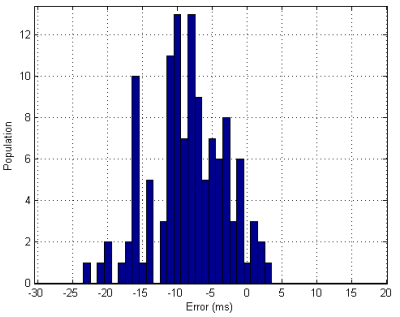


Figure 5. Error distribution of the proposed method evaluated on a recording of a drum practice routine.

5. Conclusion

The results show that we are able to compensate the latencies coming up from the sound generation, and provide the desired experience. The system works well in decent acquisition conditions (i.e uniform and direct lighting, high frame rates) and can be used to play and record drum sequences as if they were played on a MIDI keyboard controller.

As future work, we will try to track 2 other colored markers on actual drumsticks and use them to estimate the position of the lost tip. We will also create a user-friendly calibration routine for mapping the units to the metric system. The calibration will allow the system to be more robust to variations in distance between the user and the camera. Up-to-date information and the demo videos can be found at http://www.airdrums.info.

References

Bradski, G. (2000). The opencv library. *Dr. Dobb's Journal of Software Tools*.

Gillet, O., & Richard, G. (2005). Automatic transcription of drum sequences using audiovisual features. *IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, *82*, 35–45.

McGuinness, K., Gillet, O., O'Connor, N. E., & Richard, G. (2007). Visual analysis for drum sequence transcription. *EURASIP*, 312–316.

Tansuriyavong, S., Nagai, H., Nakahira, K. T., & Fukumura, Y. (2006). Development of multimedia contents for specialized skill education. *Current Developments in Technology-Assisted Education* (pp. 371–375).

Event Ordering for Turkish Natural Language Texts

Şadi Evren ŞEKER
Banu DİRİR
Yıldız Technical University
34349 Besiktas - Istanbul
Turkey
902163023042

academic@sadievrenseker.com
banu@ce.yildiz.edu.tr

1. Time Tagging

Time tagging which can be interpreted as extraction of temporal semantic from natural language texts is mainly used on almost all of the natural language research areas like question answering, text summarization or visualization of the texts which are explained in the works of (Kadri Hacioglu 2005) and (Inderjeet Mani 2000).

A natural language text contains semantic value from many different categories. For example the location or personal information or a know-how can be carried through natural language sentences. Besides of all these information, all of the natural language sentences should contain a temporal logic since all the verbs are strongly connected to time. For example a sentence without a location information is possible but all sentences should contain a time being.

Time tagging is a technique for marking the temporal information of events, time expressions or the relations of events on the time line. Although the linearity of the time is an open discussion, the time tagging techniques are built over only linear temporal logics. For example the Allen's interval logic (Grigore Ro, 2006) or Reichenbach's temporal logic (Reichenbach 1947) are two samples for the representation of time by linear.

After the correct tagging of a natural language text, the implementation of some automat codes is possible to process the event ordering or question answering. The tagging can be done by two possible ways. For mature NLP languages which mostly solved the morphological and syntactic parsing problems, it is possible to implement an autonomous software to tag the temporal information on the NLP. On the other hand for the languages still under massively development of NLP which do not have a satisfactory success on morphological and syntactic levels, the only way of tagging the temporal information is manual.

Time tagging is still important for these languages to show a target for representation of semantic after the syntactic studies and prepare some tools after an achievement on these levels. For example in Turkish there is still no satisfactory syntactic and morphological tools to extract the semantic and also temporal logic in Turkish is different than Indo-European languages in some ways.

In this paper, after a research on temporal logics which are built over Indo-European language family, we have figured out the differences in Turkish and we have also developed a representation tool.

Furthermore, we have implemented a software for question answering and visualization of text automatically. Tests of this tool mainly run over child stories so any child can increase the understanding from text by generating a visual extraction of chronology of events on the text or s/he can question the text in a temporal way of understanding.

2. Reichenbach Temporal Logic

Reichenbach temporal logic is built on simple three temporal anchors.:

- Speech time (symbolized by S)
- Reference time (symbolized by R)
- Event time (symbolized by E)

Most of his study was focused on the natural languages. So he has formulated the order of these times.

For example a sentence like "I read the book" can be formalized as R=E<S on the other hand a sentence like "I have read the book" can be formalized as E<R=S. Please note that on the former model, event is before the speech time and the speech is referring to the event time, so the event and reference times are equal and smaller than speech time on the model. For the latter example, again the event is before the speech time, but the speech is referring to the current time so the speech time and reference times are equal and greater than the event time.

By a simple probability calculation we can end up with 13 possible order of above temporal anchors. Obviously all of these probabilities are not meaningful in a natural language.

Reichenbach has named these possibilities by using Anterior, Simple and Posterior aspects and Past, Present and Future tenses. So in English or in any natural language there can only be 9 possible meaningful time in the opinion of Reichenbach.

Below table covers these possibilities and samples for each of the case:

Table 1. All possible 13 permutation of Reichenbach temporal logic and their English tense/aspect and a sample for each case.

Permu-tation	Reichenbach Tense Name	English Tense	Sample
E<R<S	Anterior past	Past perfect	I had slept
E=R<S	Simple past	Simple past	I slept
R<E<S			
R<S=E	Posterior past		I would sleep
R<S<E			
E<S= R	Anterior present	Present perfect	I have slept
S= R= E	Simple present	Simple present	I sleep
S= R<E	Posterior present	Simple future	I will sleep
S<E<R			
S=E<R	Anterior future	Future perfect	I will have slept
E<S<R			
S<R=E	Simple future	Simple future	I will sleep
S<R<E	Posterior future		I shall be going to sleep

Please note that in the above table, blank lines represents the meaningless cases of the permutation for English.

3. Allen's Interval Logic

Allen's interval logic (AIL) or temporal logic (ATL) deals with orders

of the events. The representation of event orders like "event A is before event B" or "event A is at the same time with event B" are the operators of this logic.
The basic variables in AIL are the intervals and Allen has built his logic over binary operators working on those intervals. In AIL, there are 13 basic binary operator connecting intervals by constraints. These intervals can be considered as running threads or any operations on time line.
For example in an example sentence like "John ate an apple at the table after he has entered the room" we have events "eat" and "enter" also there are hidden event which John goes to the table and takes the apple in order to eat an apple from the table after he has entered the room.

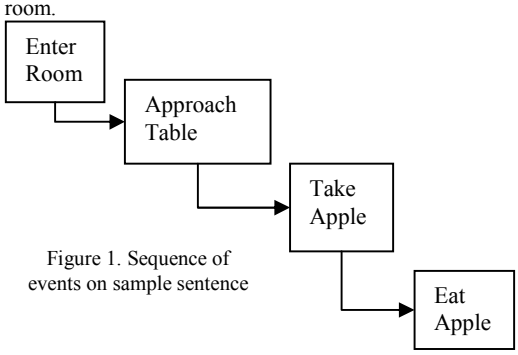


Figure 1. Sequence of events on sample sentence

If the states of the events are considered, we know John was outside of the room, before he has entered the room, also he was away from the table before he approaches table and he has no apples before taking apple.

So from above sample sentence it is possible to conclude John had an apple when he is inside the room or John has had no apple while he was away from the table or while he was outside of the room.
Above sample can be modeled by using AIL. Let's say entering room (ER) requires to be outside of the room (OR) and after entering room the state is inside the room (IR) and similarly approaching table (AT) changes state of being away from table (SAT) to state of being close to table (SCT), taking apple (TA) is a transformation of state from not having apple (NHA) to having apple (HA). All these states are pre-requirements in the case of eating apple (EA).
Above sentence can be modeled in Allen Temporal Logic as below formulation:
Meets(OR,ER) ∧ Meets(ER,IR) ∧ During (ER,SAT) ∧ During (AT,IR) ∧ Meets(SAT,AT) ∧ Meets(AT,SCT) ∧ During(AT,NHA) ∧ During(TA,IR) ∧ During (TA,SCT) ∧ Meets(NHA,TA) ∧ Meets(TA,HA) ∧ During(EA,HA) ∧ During (EA,CT) ∧ During(EA,IR) ∧ Meets(TA,EA)
Above formulation demonstrates all the temporal states and events on the sample sentence. On the other hand a reader can interpret the above sentence and can add more states which can still be modeled by AIL. For example if John does the above order of events when he is hungry than this state can be added to the model of AIL. For this case the model would be:
Occurs (hungry, NHA) ∧ Holds (hungry, TA) ∧ Meets (hungry, EA)
So from AIL model, John eats an apple when he gets hungry while he does not have an apple and he takes an apple while he is hungry and the status of hungry is finished by eating the apple.
Below list holds the possible operators of Allen Interval Logic:

- Before (x,y) or After (y,x)
- Overlaps (x,y) or Overlapped (y,x)
- Meets(x,y) or MetBy(y,x)
- Contains(x,y) or During (x,y)
- Starts(x,y) or StartedBy(y,x)
- Ends(x,y) or EndedBy(y,x)
- Equals(x,y)

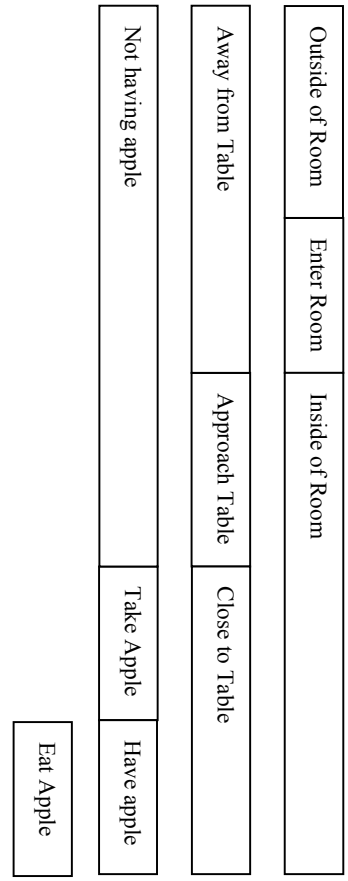


Figure 2. Event and states diagram

4. IMPLEMENTATION AND TESTING

This section covers the application of TimeML with above modification on Turkish natural language texts. We have created a corpus of 15 children stories with 1043 sentences and 1618 events. The corpus is created because this study is the first time try on event ordering for Turkish natural language. Unfortunately there is no previous work and corpus that our study can be compared. We have created an original corpus for further challenges on the area and measuring the success of our study.

The distribution of events on tenses in corpus is listed below:

Table 2. Percentages of Turkish Reichenbach temporal model in our corpus.

Reichenbach Tense Name	Turkish Tense	Number	Percentage
Anterior past	Geçmişin hikayesi	165	10%
Simple past	Geçmiş	133	8%

	Gelecek hikayesi	238	15%
Posterior past			0%
	Gelecek Rivayeti	37	2%
Anterior present	Şimdiki Hikayesi	76	5%
Simple present	Şimdiki	390	24%
Posterior present	Gelecek	47	3%
		478	30%
Anterior future		1	0%
		3	0%
Simple future	Gelecek Zaman	48	3%
Posterior future		2	0%

Please note that the above corpus mainly contains stories so it is an expected result to get higher percentages on the past tenses and present tenses than the future tense. Also the future tenses with "ol-" (Being) is almost zero because of their rarely usage in Turkish. In fact most of their usage are from translated stories in Turkish.

After adding the above relation type alternative to TLink BNF, we have re implemented the ttk-1.0 and Tango v1.5 which are two major software implemented for TimeML applications. Also we have created a corpus of child stories for test purposes in Turkish and tested the success of older and newer versions of TimeML, where in newer version the "CYCLES" relation type is implemented.

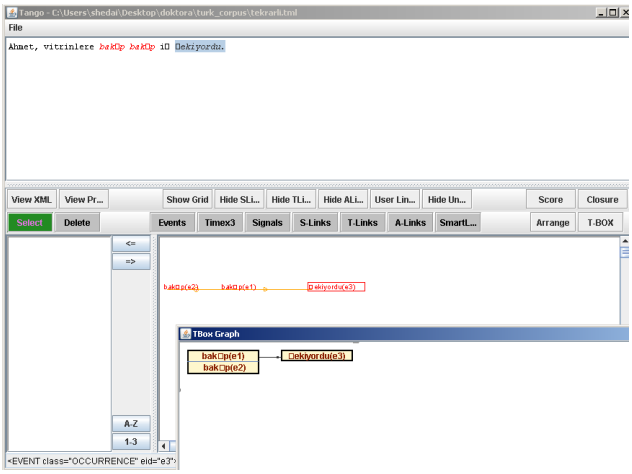


Fig3. Sample screenshot of ATL implementation for Turkish

Success of TimeML in Turkish Corpus without CYCLES relation: 53%

Success of TimeML in Turkish Corpus with CYCLES relation: 55%

There is a 2% of increase after the implementation of the new relation type to TimeML.

By this study, we have enhanced TimeML a step beyond to cover Turkish temporal logic. After the above modifications TimeML can be used in both Turkish and English and can model more events successfully. Also this enhancement depends on the Reichenbach temporal logic and already discovered by him as a permutational manner.

Also the study is applied on a corpus and the numerical results have been demonstrated above. The success of TimeML before modification would be 55% percent and after the modifications we have suggested the success covers all possible event tenses which is 91%.

CONCLUSION

During this study, we have stated the Allen's temporal logic and its applications on natural language processing. The comparison between ATL and temporal logic behind Turkish also criticized and an additional relation type to ATL is suggested to cover Turkic languages. Also this suggestion in logical level is carried on to the implementation layer and an application is modified using this theoretical logic. The tests carried on a corpus composed by simple Turkish child stories have showed the importance of this addition and increased the success of representation of Turkish natural language sentences in both ATL and TimeML.

REFERENCES

Kadri Hacioglu (2005). Automatic Time Expression Labeling for English and Chinese Text, Kadri Hacioglu, Ying Chen and Benjamin Douglas Springer LNCS, Computational Linguistics and Intelligent Text Processing, ISSN 0302-9743 , pages 548-559

Inderjeet Mani (2000) Robust temporal processing of news, Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, Inderjeet Mani , George Wilson Pages: 69 - 76

Grigore Ro (2006) Allen Linear (Interval) Temporal Logic – Translation to LTL and Monitor Synthesis– Grigore Ro¸su1 _ and Saddek Bensalem, CAV'06, LNCS 4144, pp 263-277

Reichenbach (1947). H., Elements of Symbolic Logic, New York: Macmillan (1947)

Ozkirimli A (2001). Türk Dili: Dil ve Anlatım, İstanbul Bilgi University Press.

TimeML, TERQAS, (2002), TimeML has been developed in the context of three workshops starting from 2002.

"TimeML Annotation Guidelines Version 1.2.1" Roser Saur´ı, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky (January 31, 2006)

Natalia Kotsyba (2006). Using Petri nets for temporal information visualization Études Cognitives/Studia Kognitywne, CEEOL

Andr´e Bittar (2009). Annotation of Events and Temporal Expressions in French Texts Proceedings of the Third Linguistic Annotation Workshop, ACL-IJCNLP 2009, pages 48–51, Suntec, Singapore, 6-7 August 2009. c 2009 ACL and AFNLP

Christian Kissig and Laura Rimell (2005). Closing TLink-Relations (Reasoning with Intervals) June 23, 2005

Classifying Exceptions in Agent-Based Protocols: A Thin Line Between Violation and Opportunity

Özgür Kafalı OZGURKAFALI@GMAIL.COM
Department of Computer Engineering, Boğaziçi University, TR-34342, Bebek, İstanbul, Turkey

1. Introduction

Open multiagent systems are characterized by the fact that agents can enter and leave on will. There is usually no central authority that regulates the actions of the agents. Hence, an agent's behavior can neither be controlled nor predicted before the system's run time execution (Fisher & Wooldridge, 1994). When one or more agents behave unexpectedly in such a system, *exceptions* may occur, leading to the improper workings of the entire system. While diagnosing an exception is important, it is not feasible to deal with each exception separately. Thus, when an exception occurs, the agent facing the exception should determine its significance and decide whether to diagnose it further.

Accordingly, we provide categories of exceptions and propose to classify exceptions based on their immediate causes. Several work has been noted before in effort to classify exceptions (Klein & Dellarocas, 2000; Sadiq & Orlowska, 2000). Some of these work focus on exceptions that can be anticipated before. That is, the exception itself and its preconditions are known prior to its occurrence. Those exceptions can therefore be added to the system's design before execution. However, most exceptions of that type are domain-specific. Thus, it is hard to make a general judgment based on that classification.

In this work, we deal with exceptions that are not anticipated at design-time, but rather encountered during runtime. So, the classification we propose can be applied to any domain (e.g., e-commerce) as long as a formal model for action descriptions and agent interactions are provided. Here, we use commitments to model agent interactions and predicate logic to represent outcomes of agent actions. We give a case study from a delivery process to exemplify its applicability. The next section gives necessary background on the concepts that are key to this work. Then, we present our classification algorithm, and the case study. Finally, we conclude with possible future directions.

2. Background

Next, we describe our theoretical framework.

2.1 Commitments

Commitments are formed between two agents and roughly correspond to obligations (Singh, 1999). The debtor of a commitment is the agent that is committed to bring about a condition. The creditor benefits from the commitment. A base-level commitment *c(x, y, p(z))* represents an obligation from debtor *x* to creditor *y* to bring about proposition *p(z)*. A conditional commitment *cc(x, y, q(z), p(z))* represents a contract between debtor *x* and creditor *y* with condition *q(z)* and proposition *p(z)*. When *q(z)* is satisfied, *x* will become committed to *y* for satisfying *p(z)*. We allow commitments to have three states; (1) *active* commitments are created and are in charge, (2) *fulfilled* commitments are no longer in charge due to their propositions being satisfied, (3) *violated* commitments are also not in charge but their propositions have not been satisfied.

2.2 Ontologies

An ontology is a data model used for representing a domain, and it defines a set of concepts and the relations between those concepts (Guarino, 1998). As is customary in multiagent systems, we use ontologies to provide domain knowledge to agents. The agents can then use this to reason on exceptions, and deal with them.

2.3 Protocols

A process is represented by a set of protocols (Papazoglou, 2003). Here, we are concerned with distributed settings where agents are only aware of part of their environment. Thus, it is important to specify protocols from the viewpoint of the roles that the agents are enacting. This enables us to specify the exact information that is available to each role.

Definition 1 A protocol template $\mathbb{P}_r = \langle \mathcal{M}, \mathcal{P}, \mathcal{R} \rangle$ is a description of what role r is aware of in order to act in its environment. $\mathcal{M}$ is a set of messages that role r can send. It simply corresponds to the actions that role r can perform (with particular effects on commitments) while executing the protocol $\mathbb{P}_r$. $\mathcal{P}$ is a set of predicates that role r is aware

of and $\mathcal{R}$ is a set of roles that role r is aware of. ∎

Definition 2 A protocol state $\mathbb{S}$ is a set containing predicates that hold at a particular time point and the status of commitments at that time point. ∎

Definition 3 A goal state $\mathbb{G}_i$ is a desired state for agent i that it wishes to reach during execution of the protocol. ∎

2.4 Exceptions

Exceptions have been classified before in the literature according to several criteria (Klein & Dellarocas, 2000; Sadiq & Orlowska, 2000). In the light of those, we classify exceptions into three distinct categories, each covering separate cases that may arise during protocol execution.

- *Violation:* A violation occurs when a predicate or commitment that is included in the agent's goal state is violated, e.g., a bookstore violating its commitment with a customer for delivering its book.
- *Bad-Fulfillment:* There may be situations that although the goal is satisfied, an extra event obstructs its complete fulfillment, e.g., a CD comes damaged.
- *Bonus:* A bonus is an unexpected situation which is not necessarily a bad thing for the agent, e.g., the book comes with a CD. Although the agent may benefit from the situation, it is still considered an exception, and the agent may benefit from inspecting it further. Usually, protocols are extended with such opportunities. That is, the protocol execution that covers the bonus is added to the protocol itself.

The second and third classifications both represent a monotonic increase in the agent's goal state (i.e, extra predicates). Thus, they point to unexpected situations even though the goal seems to be satisfied. The second represents an unwanted case and the agent's goal is not fully satisfied since the extra predicate obstructs the goal itself. The third represents a beneficial case which the agent may benefit from. In that case, the extra predicate has no relation with the goal whatsoever.

3. Proposed Method

During protocol execution, agents change state due to performed actions or occurred events. When entered a new state, an agent compares the state with its goal state, checking for exceptions.

3.1 Classification

Algorithm 1 describes how an agent classifies an exception based on its current state and its goal state. The first two

Algorithm 1 Classification

Require: $\mathbb{S}_c$ {current state}
Require: $\mathbb{G}_i$ {goal state}
{violation}
1: **for all** $c(x, y, p(z)) : violated \in \mathbb{S}_c$ **do**
2: **if** $(c(x, y, p(z)) : fulfilled \in \mathbb{G}_i)$ or $(p(z) \in \mathbb{G}_i)$ **then**
3: **return** $c(x, y, p(z)) : violated$
4: **end if**
5: **end for**
6: **for all** $p(z) \in \mathbb{S}_c$ **do**
7: **if** $\neg p(z) \in \mathbb{G}_i)$ **then**
8: **return** $p(z)$
9: **end if**
10: **end for**
{bad-fulfillment}
11: **for all** $p(z) \in \mathbb{S}_c$ **do**
12: **if** $(p(z) \notin \mathbb{G}_i)$ and $(z \in \mathbb{G}_i)$ **then**
13: $p(z)$
14: **end if**
15: **end for**
{bonus}
16: **for all** $p(z) \in \mathbb{S}_c$ **do**
17: **if** $(p(z) \notin \mathbb{G}_i)$ and $(z \notin \mathbb{G}_i)$ **then**
18: $p(z)$
19: **end if**
20: **end for**

cases cover violation based exceptions (lines 1-10). The third case covers situations where the goal of the agent is obstructed by an extra event (lines 11-15). The object of the extra predicate provides the relation with the goal. The last case covers bonus situations where the extra event has no relation with the goal (lines 16-20). Further classification can be done within these categories using a domain ontology (e.g., determine the level of violation, how badly the goal is fulfilled, or how significant the bonus is).

3.2 Case Study

Figure 1 describes the delivery process inspired from the *MIT Process Handbook* (Klein & Dellarocas, 2000). It contains three business parties; customer, bookstore, and deliverer. In a normal execution, the customer purchases an item from the bookstore. The bookstore pays for the delivery of the item. The deliverer delivers the item.

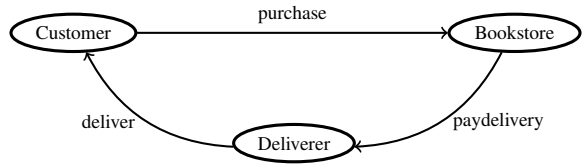


Figure 1. Delivery Process

We use following commitments to represent the contracts between the agents (i.e., parties); cc(bookstore, customer, purchase(z), deliver(z)) tells that if the customer purchases

an item, the bookstore delivers that item. cc(deliverer, bookstore, paydelivery(z), deliver(z)) tells that if the bookstore pays for the delivery of an item, the deliverer delivers that item. The goal state of the customer is $\mathbb{G}_{customer}$ = $\{purchase(z), deliver(z)\}$. Once the customer purchases an item, it wishes to get that item delivered. Let us now consider three cases from the delivery process.

- *Violation:* Assume the customer's state to be $\mathbb{G}_{customer}$ = $\{purchase(book1)\}$. The customer has purchased a book, but the book does not arrive. This is a violation of the bookstore's commitment to the customer, and it is classified as a violation.

- *Bad-Fulfillment:* Assume the customer's state to be $\mathbb{G}_{customer}$ = $\{purchase(cd1), deliver(cd1), damaged(cd1)\}$. The customer has received the CD, but it is damaged. This situation obstructs the customer's goal, and it is classified as a bad-fulfillment as the extra predicate is related with the goal.

- *Bonus:* Assume the customer's state to be $\mathbb{G}_{customer}$ = $\{purchase(book1), deliver(book1), deliver(book2)\}$. So, the customer has received the book, in addition another book is also delivered. This corresponds to a bonus situation which is classified as the extra predicate's object is not included in the customer's goal.

Consider the following change in the bonus situation above; $\mathbb{G}_{customer}$ = $\{purchase(book1), deliver(book2)\}$. So, the customer has purchased *book1*, but *book2* comes while there is still time for *book1* to be delivered. This is again classified as a bonus. However, *book2* may be wrongly delivered instead of *book1* (i.e., violation). This case actually corresponds to a *deviation* from normal protocol execution, and should be considered separately as a new category. In addition, the significance of an exception may vary within a category. Consider the customer's state $\mathbb{G}_{customer}$ = $\{purchase(cd1), deliver(cd1), \neg invoice(cd1)\}$, meaning that an invoice has not been sent with the delivery. This is also considered a bad-fulfillment. However, it may not be as significant as the CD being damaged. The current classification does not support such level of detail. Domain knowledge is required to provide priorities within categories.

4. Discussion

We have previously considered identifying exceptions that occur in business protocols (Kafalı & Yolum, 2009). As a single agent task, identifying an exception's actual cause is hard, if not impossible (due to limited knowledge). When considered as a collaborative task, diagnosing exceptions is even harder and requires considerable amount of computation. Thus, we have provided a classification based on exception categories. This classification is a forward step in determining the significance of each exception, and selecting which ones to diagnose further. Further classification can be done based on domain knowledge. This will provide extra accuracy while determining the diagnosis effort of an agent when faced with several exceptions with strict time considerations. In addition, other types of processes exist in e-commerce scenarios such as *fit* or *sharing* processes rather than *flow* processes as we have seen in our case study (Klein & Dellarocas, 2000). Although not for violation based exceptions, we believe that classification can be harder in those process types.

References

Fisher, M., & Wooldridge, M. (1994). On the formal specification and verification of multi-agent systems. *International Journal of Cooperative Information Systems*.

Guarino, N. (1998). *Formal ontology in information systems*. IOS Press.

Kafalı, Ö., & Yolum, P. (2009). Detecting exceptions in commitment protocols: Discovering hidden states. *LADS Workshop, MALLOW'09*.

Klein, M., & Dellarocas, C. (2000). A systematic repository of knowledge about handling exceptions in business processes. *ASES Working Report. MIT*.

Papazoglou, M. P. (2003). Web services and business transactions. *World Wide Web*, *6*, 49–91.

Sadiq, S. W., & Orlowska, M. E. (2000). On capturing exceptions in workflow process models. *Proceedings of the 4th International Conference on Business Information Systems*.

Singh, M. P. (1999). An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, *7*, 97–113.

Effect of Consistent Exploration in Dynamic Environments: Does Trust Work in Competitions?

Özgür Kafalı OZGURKAFALI@GMAIL.COM
Department of Computer Engineering, Boğaziçi University, TR-34342, Bebek, İstanbul, Turkey

1. Introduction

Agents often need to model their surroundings when working together, either cooperatively or competitively. This is because, each agent has different expertise in different fields of work. And they need others to delegate some of their assigned work when they lack the required expertise. However, each agent is not equally trustworthy due to the autonomy of open environments (Yolum & Singh, 2005). Thus, an accurate model of the environment is required to decide which agents to interact with and delegate tasks to.

Building an accurate model of trust in an open dynamic environment is hard. When competitive agents are involved, it is even harder since each agent is self-interested and may disguise its actual behavior. We aim to identify the effect of consistent exploration on the accuracy of models in such settings. For this purpose, we use two different types of agents, one that builds trust based on traditional methods (Huynh et al., 2004; Sabater & Sierra, 2001), and one that relies on action-based modeling and aggressive exploration. We use the ART Testbed environment to make simulations since it mimics the described settings best (Fullam et al., 2005). In some sense, being trustworthy is related to providing good services in the testbed. Thus, the concepts trustworthy and useful can be used interchangeably in the ART context. The next section briefly introduces ART Testbed, and two types of agents that we use in our experiments. Then, we describe our metrics, the simulation setup, and depict our results. Finally, we conclude with further discussion on related work.

2. Background

2.1 ART Testbed

The ART Testbed simulates an art appraisal domain where clients wish to get their paintings evaluated by the service provider agents (Fullam et al., 2005). Each participating agent is a service provider (i.e., an appraiser) in the game that sells its opinion when requested. Clients as well as other service providers may be willing to purchase opinions about a painting. Each painting belongs to a specific era of art, and every agent has an expertise value for each era, which may change within a game. The expertise of the agent for an era determines how well it can judge the value of paintings for the era (i.e., higher expertise values lead to better evaluation of paintings).

The game runs as a series of timesteps with agents trying to evaluate the paintings assigned by the clients. When a timestep ends, the appraisal errors are computed for the agents by checking their evaluations. When an agent is assigned a painting for an era that it has low expertise, it may ask other agents for their opinions about the value of the painting. However, each transaction in the game has a cost associated with it. The aim of the game is to end up with the highest bank balance (i.e., an overall value considering the agent's income and expenses).

2.2 Agent-Based Trust Model

We use the agent Frost in our simulations as the basis of an agent-based modeling approach (Kafalı & Yolum, 2006). Frost has been ranked third in the first ART competition in 2006, and thus is a good benchmark for our comparisons. Similar to most other traditional approaches (Huynh et al., 2004; Sabater & Sierra, 2001), Frost models all other agents within the environment and chooses the agents to interact with based on these models. In order to model its surroundings, Frost keeps an estimation of every other agent's expertise value for each era in the game. The modeled expertise value represents how well the agent generates opinions about paintings (i.e., how trustworthy it is in that era). Frost does not explore its environment all the time. That is, when it identifies useful opinion providers in the environment, it exploits those agents until their services are no more useful (i.e., they start to provide inaccurate opinions) rather than trying to find other useful agents.

2.3 Action-Based Trust Model

Typically, agents utilizing reinforcement learning model their environments through trial and error interactions (Kaelbling et al., 1996). The agent has an opportunity to select from a variety of actions which either leads to a re-

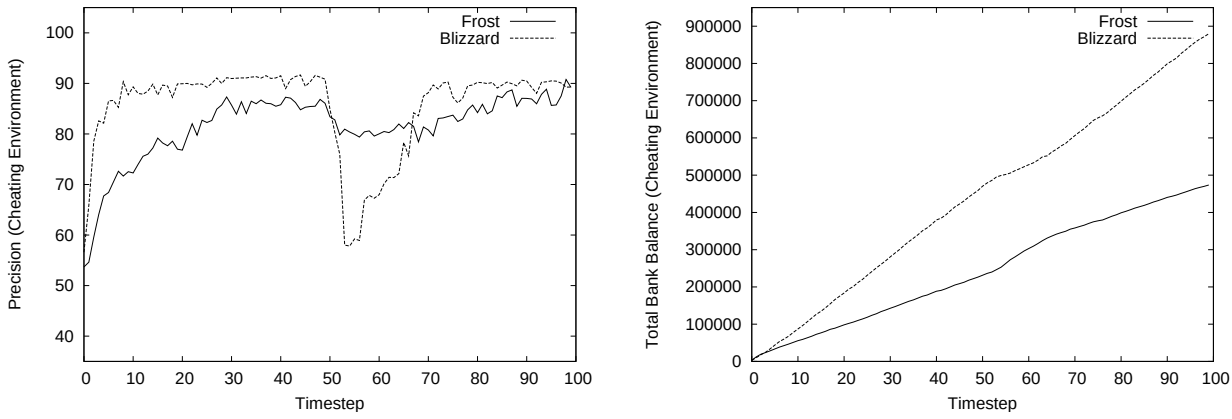


Figure 1. Effect of Consistent Exploration in a Cheating Environment

ward or a punishment as a consequence (i.e., reinforcement value). The agent's primary goal is to maximize the total reinforcement value during the execution.

Blizzard is our concrete agent implementation that adopts action-based modeling. Blizzard also participated in the ART competition in 2007, and it has been ranked third. Here, we use that agent as the basis for our action-based modeling approach (Kafalı & Yolum, 2008). It uses an extension of reinforcement learning, called Q-learning, to model its actions (Wiewiora et al., 2003). In order to act in an environment, Blizzard needs only to be aware of its set of actions that are available for that environment. Unlike traditional trust models, it does not require further heuristics on the operational properties of the environment (e.g., distribution of expertise values to the agents) since it will not aim on predicting how other agents are performing. The main type of action that Blizzard records for the ART domain is the *Opinion Request* action. Unlike Frost, Blizzard consistently explores its environment to check the status of its current service providers (i.e., to point out changes earlier), and to identify new useful providers.

3. Experiments

We evaluate how exploration affects an agent's performance in dynamic and competitive environments. We use the following two metrics, one for measuring the accuracy of trust models, and one for the cost of those models.

- *Precision:* Informally, this metric measures how well an agent finds the useful opinion providers in the game. The metric is designed to measure the percentage of contacted providers that are actually experts in their eras. Equation 1 measures the precision for a particular era j.

$$Precision(j) = \frac{\sum_{i=0}^{numAgents} p_{ij} * e_{ij}}{h_j} \quad (1)$$

  Here, p_{ij} is the percentage of queries directed to agent i in the jth era. e_{ij} is the actual expertise of agent i in era j. To normalize the metric, we divide it by the expertise of the most expert agent (h_j) in that era. Then we average this value over all eras to get a measure in the scale of 1 to 100.

- *Total Bank Balance:* Bank balance is used to evaluate the overall success of the agent. It shows the total of the agent's income minus the agent's expenses over the game rounds. Recall that the agent's income mainly stems from correct opinions it generates and its expenses stem from buying opinions from other agents. Here, we employ bank balance to incur the cost of exploration into the experiments.

The simulation environment consists of three Frost and three Blizzard agents, eight honest, eight cheating, and eight dummy agents. Honest agents always respond truthfully to opinion requests, whereas cheating agents consistently cheat in their responses. The behavior of dummy agents is rather unpredictable, they alternate between honest and cheating behavior throughout the simulation. In addition, the expertise of agents are dynamic in the sense that half of the honest agents' expertise values change after the first half of the simulation. We call this a *Cheating Environment*, since competition is a motivation for cheating. We run five replicas of the simulation to eliminate the effect of randomness, and output those results. Figure 1 plots the two metrics, comparing Blizzard with Frost.

Precision: Due its aggressive exploration strategy, Blizzards maintains a high precision value in a few timesteps.

Frost, however, is not very successful in increasing its precision quickly. It builds up trust progressively and rather slowly. When expertise changes occur, Frost's precision is not affected at all. Since not all of the honest agents have their expertise values changed, it is most probable that Frost has not explored enough to be aware of those agents. Blizzard, on the other hand, is drastically affected by the changes in the environment but regains its highest precision value by exploring the environment again when it senses the expertise changes.

Total Bank Balance: Blizzard has a huge advantage over Frost when the incomes of the two agents are considered. That is, although Blizzard invests more on exploration, it is rewarded well from the very beginning of the simulation. The only drop in Blizzard's bank balance is observed when it explores the environment again, just after the expertise changes occur. During that period, its bank balance does not increase rapidly as it does before. However, it regains its acceleration after the expertise changes stop and its precision reaches its maximum again.

The simulations reveal the fact that trust solely is not enough to maintain a concrete model of an environment without consistent exploration. This is due to the open and dynamic behavior of multiagent systems, where an agent's next action cannot be predicted beforehand.

4. Discussion

Exploring an environment via direct interactions is a proven way of accurately building trust (Huynh et al., 2004; Sabater & Sierra, 2001). However, in dynamic environments with large number of agents, it is unrealistic to assume that each agent can access every other directly. Thus, reputation information can also be used in such settings to justify an agent's belief about others. Costa *et al.* experimented on how reputation can be utilized by the agents competing in ART Testbed (da Costa et al., 2008). While we do not believe that reputation information is directly useful when building trust from scratch, it may be helpful to support an agent's prior belief about a subject agent.

We have other evidence supporting the success of Blizzard's aggressive exploration strategy. Teacy *et al.* also made a similar analysis on the finalists of two past ART competitions (Teacy et al., 2008). There, they have shown that Blizzard explores the environment sufficient enough to find the service providers which are capable of providing the best services available and reaches the highest bank balance. Note that the cost involved in realizing the exploration is also included in the results.

References

da Costa, A. D., de Lucena, C. J. P., da Silva, V. T., Azevedo, S. C., & Soares, F. A. (2008). Computing reputation in the art context: Agent design to handle negotiation challenges. *Eleventh International Workshop on Trust in Agent Societies, AAMAS* (pp. 121–131).

Fullam, K., Klos, T. B., Muller, G., Sabater, J., Topol, Z., Barber, K. S., Rosenschein, J. S., & Vercouter, L. (2005). The agent reputation and trust (ART) testbed architecture (pp. 50–62.).

Huynh, T. D., Jennings, N. R., & Shadbolt, N. (2004). Fire: An integrated trust and reputation model for open multiagent systems. *Proceedings of 16th European Conference on Artificial Intelligence* (pp. 18–22).

Kaelbling, L. P., Littman, M. L., & Moore, A. P. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kafalı, Ö., & Yolum, P. (2006). Trust strategies for ART Testbed. *Ninth International Workshop on Trust in Agent Societies, AAMAS* (pp. 43–49).

Kafalı, Ö., & Yolum, P. (2008). Action-based environment modeling for maintaining trust. *Eleventh International Workshop on Trust in Agent Societies, AAMAS* (pp. 23–32).

Sabater, J., & Sierra, C. (2001). Regret: Reputation in gregarious societies. *AGENTS '01* (pp. 194–195). Montreal, Quebec, Canada: ACM Press.

Teacy, L., Chalkiadakis, G., Rogers, A., & Jennings, N. (2008). Sequential decision making with untrustworthy service providers. *AAMAS* (pp. 755–762).

Wiewiora, E., Cottrell, G. W., & Elkan, C. (2003). Principled methods for advising reinforcement learning agents. *ICML* (pp. 792–799).

Yolum, P., & Singh, M. P. (2005). Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on Systems, Man, and Cybernetics. Part A*, *35*, 396–407.

Use of Cluster Analysis in Twitter

Nadin Kokciyan NADIN.KOKCIYAN@BOUN.EDU.TR
Department of Computer Sciences, Bogazici University

1. Introduction

Social networking became very popular online that we call *Online Social Networking* with the advent of social sites such as Facebook, MySpace, Twitter, Orkut etc. There is a huge number of people participating in social networking sites. People sharing common interests are creating groups to be together.

In this paper, our focus is on one social site: Twitter. Twitter is composed of users sending short messages (*tweets*) to each other. We are interested in relations between users. We are collecting a dataset starting from a seed user and examine interrelations between users within this dataset.

The goal of this paper is to discover close-knit clusters of users according to relations between users in Twitter through the use of two clustering algorithms. After detecting these clusters, we are interpreting the results to find a meaning on these communities.

2. Study Methodology

2.1 Twitter

Twitter is a large network of users and user groups having common interests. There is the notion of "public" and "private" profile for each user. If a user profile is declared as "public", it can be accessed by anyone. Otherwise, it is only accessed by friends of this user. Some vocabulary related to Twitter is as the following:

- *tweet*: it is the short message written and shared by a user. Each tweet contains a maximum of 140 characters
- *following(friend)*: a user can follow other users' tweets.
- *follower*: other users can follow a user.
- *mention*: a user can mention names of other users in his/her tweets. Every mention begins with a character '@', and this character is followed by a username.

2.2 Twitter API[1]

Twitter offers an API for developers to extract information from Twitter. The API supports many methods to send and receive Twitter data. Inherently, there are some limitations. Every user is allowed to use the API in a certain rate limit.

2.3 Scenarios

Once we have the account to use Twitter API, it is possible to use many methods offered by the API to extract all data(friends, tweets, mentions etc.) of a user who has a public profile. Here are some scenarios to collect data:

1. Choosing a seed user and adding all the friends of this user having the same location, to the dataset.
2. Choosing a location, finding random users from this location and adding these users to the dataset
3. Choosing a seed user, adding all the friends of this user to the dataset.

In all scenarios, the idea is to create NxN adjacency matrices (N is the number of users in the dataset). We focus on *friendship* and *mention* relations which reflect user's behavior in a social network such Twitter.

2.4 Datasets

The location information on Twitter is an optional field. One can declare it as "everywhere, anywhere, utopia, pandora" etc. or can leave it as an empty field.

Because of this ambiguity on location information, we decide to continue with the third scenario. We choose some seed users from Twitter. For each seed user, all friends of this user are added to the dataset. We collect all data(tweets, mentions, friends) for each user within the dataset.

This data extraction phase is repeated for each seed user. Once we have the data, it is time to extract information from it.

[1] http://apiwiki.twitter.com/

2.5 Information Extraction

What we call information is the *friendship* and *mention* relations of a user with others within the dataset.. This information is represented as binary values, if a relation exists(1) or not(0). For each seed user, these values are kept in NxN matrices.

So for each seed user, we have two NxN matrices: first one is for *friendship* relation and the second one is for *mention* relation. These matrices are variant from one user to another because of the different user behaviors. (Krishnamurthy et al., 2008) introduced a few categories of Twitter users by analyzing the follower-following count of users.

3. Clustering

There is a huge number of users participating in social networks. In these large networks, we are expecting to discover communities or clusters. For this purpose, we prefer to use two different clustering algorithms: k-Means and Fuzzy k-Means.

3.1 k-Means

k-Means clustering aims to partition n observations into *k* clusters in which each observation is assigned to the cluster with the nearest mean. In a given sample, we have *k* reference vectors and the best reference vectors minimize the total reconstruction error (Alpaydin, 2004).

k-Means is an iterative procedure. We start with some means randomly initialized but in this work, we randomly selected *k* initial means from the dataset. Then, at each iteration we have an *Assignment Step* and an *Update Step*.

In the *Assignment Step*, we assign each observation to the nearest cluster. In the *Update Step*, we calculate the new means to be the center of the observations in the cluster. These two steps are repeated until means stabilize i.e. when the observations' assignments no longer change (Wikipedia, 2010).

3.2 Fuzzy k-Means

In fuzzy clustering, each point has a degree of belonging to clusters, so each point can belong to many clusters. For each point x we have a coefficient giving the degree of being in the *k*th cluster $U_k(x)$. Usually, the sum of these coefficients for any given observation is defined to be 1 as it is like a probabilistic approach.

In this algorithm, the center of a cluster is the mean of all points, weighted by their degree of belonging to the cluster. The algorithm repeats until it converges (Wikipedia, 2010).

We don't start with random means as in k-Means algorithm but with random coefficients assigned to each observation. In these algorithms, results are changing according to the start point randomly generated. In Fuzzy k-Means, there is a fuzzified coefficient m, when m is close to 1, this algorithm has similar results with k-Means.

4. Visualization of Networks

Pajek[2] is a software used for network analysis and visualization. While visualizing networks with Kamada-Kawai[3] perspective on *Pajek*, we can observe clusters within the overall network structure.

Pajek is preferred to visualize the social network structure of Twitter. As *friendship* and *mention* relations are not reciprocal in Twitter, we used arcs(directed relations) in our network model.

5. Implementation

The data extraction part is implemented in Java through the use of *Twitter4j*, a Twitter API for Java and the clustering part is implemented in MATLAB. The *Hamming Distance*[4] is used as a distance measure between two observations. So if two users have many friends in common or many mentions directed to common users, the Hamming distance between these users is very low which means that these users are mostly part of a same cluster.

6. Results and Discussions

In this section, we show some resulting networks of "uskudarli" who is one of our seed users in Twitter. We can observe several clusters with the Kamada-Kawai perspective of *Pajek*. This perspective is useful to approximately observe the results before running the clustering algorithms.

6.1 Friendship Relation

Figure 1 shows an example of clusters in the social network of *uskudarli* based on *friendship* relation. While observing users assigned to this cluster, we realize that this is mostly the cluster of Bogazici University's students using Twitter.

6.2 Mention Relation

Figure 2 shows an example of clusters in the social network of *uskudarli* based on *mention* relation. While observing the cluster in the middle, we realize that these people are researchers from Netherlands working on computer sciences.

[2] http://vlado.fmf.uni-lj.si/pub/networks/pajek/

[3] an algorithm for drawing graphs via grouping

[4] Hamming Distance between two strings of equal length is the number of positions at which the corresponding symbols are different

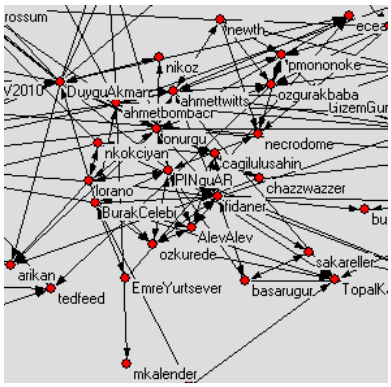


Figure 1. One of the *uskudarli*'s clusters with *friendship* relation

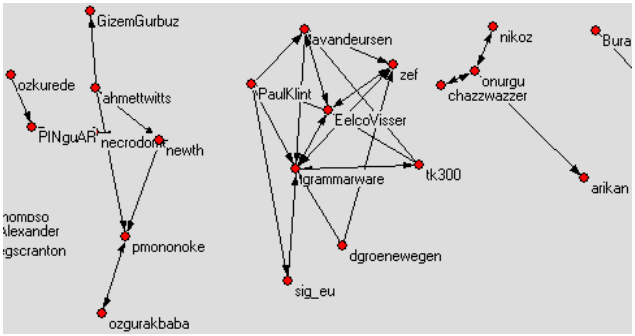


Figure 2. Part of the *uskudarli*'s Mentions Network

The cluster in the middle is the group of researchers according to the clustering algorithms' results. But when examining with *Pajek*, this group has much more members, this is because Kamada-Kawai perspective of *Pajek* doesn't use a clustering algorithm.

7. Summary and Future Work

Twitter is one of the popular social networking sites and we observe the user intentions in this site through the use of clustering algorithms. We focus on two types of relation: *friendship* and *mention*. We also visualize resulting networks via *Pajek*.

As described above, we use a similarity utility and we only group people having similar relations(friendship, mentions) according to this utility. With our approach, as an interpretation, we can say which user is similar to whom because of the similarity utility.

Clustering is a problematic issue for social networks, because a vertex(a node in the graph, in our case a "user") can belong to many clusters. So we have overlapping clusters rather than sparse clusters. To prevent this problem, a new formulation of the graph clustering problem where each node belong to exactly one cluster is done (Mishra et al., 2007). k-Means and Fuzzy k-Means algorithms are fast algorithms. Both depend on the initial choice of weights, both minimizes intra-cluster variance and both focus on local minimum of variance. When using k-Means algorithm, each user is assigned exactly to one cluster. But when using Fuzzy k-Means, each user has a degree of belonging to clusters.

It can be interesting to analyse resulting networks through different similarity measures. A user's profile is dynamic so it is possible to observe a user behavior over time. With a more detailed social network analysis, we can deduce some interesting points from clustering results.

References

Alpaydin, E. (2004). *Introduction to machine learning (adaptive computation and machine learning)*. The MIT Press.

Krishnamurthy, B., Gill, P., & Arlitt, M. (2008). A few chirps about twitter. *WOSP '08: Proceedings of the first workshop on Online social networks* (pp. 19–24). New York, NY, USA: ACM.

Mishra, N., Schreiber, R., Stanton, I., & Tarjan, R. E. (2007). Clustering social networks. *Computer Science*, *4863/2007*, 56–67.

Wikipedia (2010). Cluster analysis. [Online; accessed 13-Feb-2010].

BLUE-CHIP: Energy-Efficient Simultaneous Multi Threaded Processors

Mine Mesta MMESTA@CSE.YEDITEPE.EDU.TR
Gürhan Küçük GKUCUK@CSE.YEDITEPE.EDU.TR
Dept. of Computer Engineering, Yeditepe University

1. Introduction

Processors get faster and more complex, becoming more and more energy-hungry each passing day. The heat that comes out from the processors causes many mechanical and electrical problems. Additionally, low energy dissipation in processors has additional positive outcomes, such as reduction in cost of cooling, increased systems ergonomics, improved processor lifetime and performance, and positive effects over electricity bills. Thus, any solution intended for energy savings is valuable for today's processors.

Simultaneous Multi-Threaded (SMT) processors are very popular due to their improved throughput on running multiple threads and their simplistic design requiring minor modifications to the existing superscalar datapaths. Today, we find examples of SMT processors on server machines such as Intel Xeon as well as inexpensive laptop computers such as Intel Atom. Energy dissipation became a major issue to be solved, since it directly affects the lifetime, reliability, performance and the die size of these processors.

In SMT processors, there are many datapath resources which are either shared or replicated. The shared resources are designed to be larger in size to accommodate the entries from multiple threads, whereas replicated resources layout additional challenges and increase complexity within the processor. As a result, complexity reduction is beneficial for both improving the performance and reducing the energy dissipation for these processors.

2. Related Work

Two different methods are popular to solve the heat related problems emerged from the power and energy dissipation of processors. First method, Dynamic Voltage/Frequency Scaling (DVFS) reduces energy dissipation by lowering the voltage/frequency values (Marculescu, 2000; Weissel et al., 2002; Kondo et al., 2004).

Second method is shortly named as the *architectural solution*. This method aims the energy savings by decreasing the capacitance and the constant, activity factor values (Ponomarev et al., 2001; Kucuk et al., 2004; Buyuktosunoglu et al., 2000; Ponomarev et al., 2006; Kucuk et al., 2001; Kucuk et al., 2003; Raasch et al., 2003; Sharkey et al., 2006). In this method activity factor is directly related with capacitors' switching count, capacitance is directly related with capacitors' sizes and wire length.

This research has been funded by TÜBİTAK under grant no: 107E196.

These two alternative methods are orthogonal, and therefore, can be utilized at the same time to achieve greater energy savings. The study in this paper is in the architectural solution category, and specifically targets SMT processors.

3. System Overview

In this section, first, hardware components, configurations, and simulators used are described. Then, the proposed dynamic resizing algorithm is explained, in detail.

3.1 System Configuration

The hardware components of the platform used for the SMT simulation are as follows: 5 DELL VOSTRO 400MT, Core2Quad Q6600 2.40GHz, 8MB L2 cache, 4GB 667 MHz RAM and 3 TB total storage space.

The software configurations of the platform used for simulations are as follows: Fedora9 x86_64 OS (Fedora Project, 2008), M-sim v2.0 (SMT simulator, 2008) and SPEC2K (SPEC benchmarks, 2000). We arranged 12 application mixtures as a result of our literature survey. In Table 1, the simulation parameters are given, in detail.

Table 1. The processor configuration used throughout the tests

Parameter	Configuration
Machine width	8-wide fetch, issue and commit
Window size	64 entry IQ, 96-entry ROB, 48-entry LSQ, 192-entry PRF
L1 I-Cache	512KB, 32- cache block size, 2-way set associative, LRU
L1 D- Cache	512KB, 32-cache block size, 4-way set associative, LRU
L2 Cache	1024KB, 128- cache block size, 8-way set associative, LRU
TLB	(I)16-entry, 4096- page size, 4-way set associative, LRU (D)32-entry, 4096- page size, 4-way set associative, LRU
Functional Units	8-INT ALU, 3-INT MULT, 8-FP ALU, 3-FP MULT
Thresholds	Upsize: 32K, Downsize: 32K Cycle, Partition Size: 8 Entries

3.2 Dynamic Resizing Algorithms

The dynamic resizing algorithm is distributed over each datapath resource rather than being a centralized mechanism. For each of the resources different finite state machines are utilized.

In this study, we propose adaptive resizing of issue queue, physical register files, reorder buffer, load/store queue in SMT processors. The proposed method physically partitions a given datapath resource into multiple pieces (Figure 1), and turns off unused (or underutilized) partitions of a resource to reduce the energy dissipation on the chip. Main idea is to achieve maximum energy savings without impacting the performance of the processor. It should be noticed that, in Figure 1, the system allows any partition to be turned off. Applying this property to a FIFO queue structure might cause logical problems. Consider partition 1 in Figure 1 is turned off, and queue structure starts from partition 0 and continues to partition 2. In that case, the reactivation of partition 1 for reuse, will damage the integrity of a queue structure. To prevent such problems, we only allow the activation and deactivation of the last partition for all structures we studied. This restrictive approach might prevent higher energy savings, but it will be much simpler to implement in hardware.

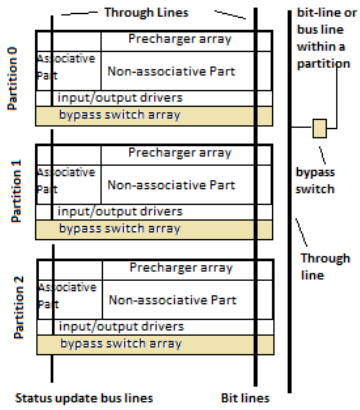


Figure 1. A partitioned resource

The finite state machine for ROB and LSQ structures is shown in Figure 2. Starting from the stable phase according to the decision made, the algorithm tries to finish the resource upsizing or downsizing process. Unfortunately, these decisions cannot be applied immediately for the circular FIFO queues due to possible integrity problems. As a result, another resource upsizing or downsizing decision can be made before the system handles the current active decision. In those cases, we give priority to Resource Upsize Decision due to our performance concerns.

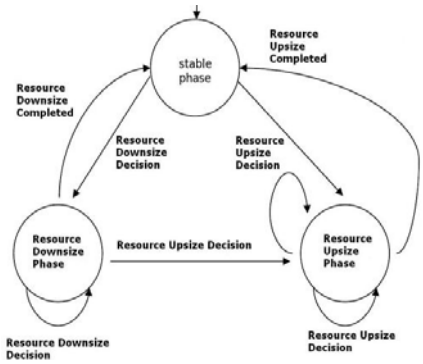


Figure 2. Resizing algorithms for queue structures

We aimed to activate or deactivate one partition at a time for a stable system, depending on our experiences from the previous studies. An alternative method can be implemented for resizing a resource with more than one partition. These aggressive techniques are to be studied in a future work, and are not the focus of this paper.

Resource Downsize Decision is taken at the end of predefined *Sampling Periods*. For this study, the sampling period is chosen to be 32K cycles. While taking the decision, average resource occupancy values within the preceding cycles are evaluated according to the active partitions at that moment. Consider the first three partitions are active and the last one is deactivated in Figure 1. After a sampling period, if it is investigated that the average resource occupancy fits in only the first two partitions, then the current last partition (partition 2) is decided to be deactivated. Here to be able to keep the average occupancy values, we need a 32 or 64 bit counter and a shift register.

For Resource Upsize Decision, experiences show that waiting until the end of a specific period may degrade the system performance. In that case, resource upsize decision can be made at any time. A new counter is added to each resource for counting the number of stalls due to limited resource sizes. This counter is checked at every cycle, and if it exceeds a threshold (Upsize Threshold), resource upsize decision is made. After the decision, the stall counter is immediately reset.

IQ and PRF resource structures are shared among all the threads in the SMT processors. The differences of these structures from ROB and LSQ are they contain out of order instructions, and they are implemented as buffer structures. Thus, resizing of these structures has less complexity compared to the queue structures. Here, upsizing can be performed right after the upsize decision without any need of a specific Resource Upsize Phase. Figure 3 depicts the finite state machine of buffer structures.

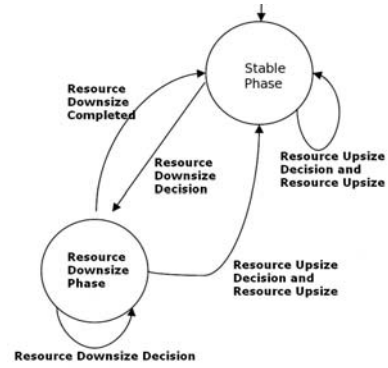


Figure 3. Resizing algorithms for buffer structures

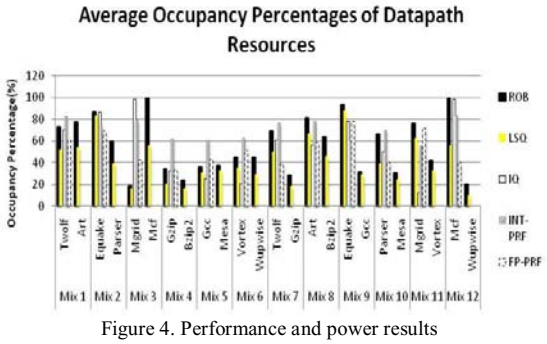


Figure 4. Performance and power results

In Figure 4, average occupancy percentages of datapath resources are shown. As it can be noticed, most of the applications do not use these resources in full capacity. In Figure 5, results of instructions per cycle (IPC) drop percentage (performance penalty) for each benchmark in a mixture, total instructions per cycle (TPC) drop percentage for each mixture and average total power per instruction reduction percentage for each mixture is presented.

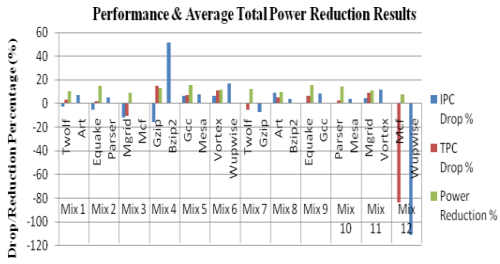


Figure 5. Datapath resource occupancies

In our tests, we studied various resource upsize and downsize threshold values and partition sizes. The Blue Chip architecture achieves better performance and power results compared to the baseline configuration in some of these configurations. For example, in Figure 6, the D64_U16_P8 (i.e. Downsize Threshold: 64K cycles, Upsize Threshold: 16K cycles and Partition Size: 8 entries) configuration gives the best results.

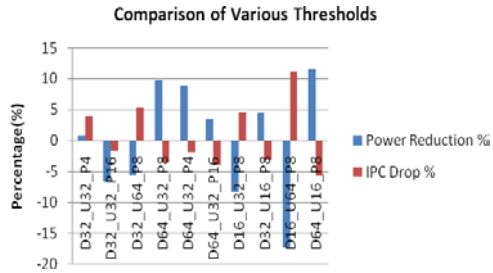


Figure 6. Comparison of various threshold values

4. Conclusion & Future Work

In this study, we resize both shared and replicated datapath resources in SMT processors. We turn off 45% of the ROB, 59% of the LSQ, 43% of the IQ, 30% of the integer PRF and, finally, 48% of the floating PRF. When compared with the baseline configuration, the Blue-Chip reduces total processor power by more than 20% while improving the processor performance by 5.7%, on the average across all simulated SPEC benchmarks. As a future work we now aim to resize L2 cache to achieve better power savings.

References

Marculescu, D. (2000). On the Use of Microarchitecture-Driven Dynamic Voltage Scaling. *Proceedings of the Workshop on Complexity-Effective Design*.

Weissel, A., & Bellosa, F. (2002). Process Cruise Control: Event-Driven Clock Scaling for Dynamic Power Management. *Proceedings of the International Conference on Compilers, Architecture.*

Kondo, M., & Nakamura, H. (2004). Dynamic Processor Throttling for Power Efficient Computations. *Proceedings of the Workshop on Power-Aware Computer Systems*.

Ponomarev, D., Kucuk, G., & Ghose, K. (2001). Reducing Power Requirements of Instruction Scheduling through Dynamic Allocation of Multiple Datapath Resources. *Proceedings of the 34th International Symposium on Microarchitecture*.

Kucuk, G. (2004). Energy-Efficient, Complexity-Effective Superscalar Processor Design. *PhD. dissertation* http://cse.yeditepe.edu.tr/~gkucuk/kucuk_PhD.pdf. State University of New York at Binghamton, Department of Computer Science, Binghamton, New York, USA.

Buyuktosunoglu, A., Schuster, S., Brooks, D., Bose, P., Cook, P., & Albonesi, D. (2000). An Adaptive Issue Queue for Reduced Power at High Performance. *Proceedings of the Workshop Power-Aware Computer Systems*.

Ponomarev, D., Kucuk, G., & Ghose, K. (2001). Dynamic Allocation of Datapath Resources for Low Power. *Proceedings of the Workshop on Complexity-Effective Design*, *held in conjunction with ISCA-28*.

Ponomarev, D., Kucuk, G., & Ghose, K. (2006). Dynamic Resizing of Superscalar Datapath Components for Energy Efficiency. *IEEE Transactions on Computers vol.55, No.2* (pp.192-213).

Kucuk, G., Ghose, K., Ponomarev, D., & Kogge, P. (2001). Energy Efficient Instruction Dispatch Buffer Design for Superscalar Processors. *Proceedings of the International Symposium on Low-Power Electronics and Design*.

Kucuk, G., Ergin, O., Ponomarev, D., & Ghose, K. (2003). Distributed Reorder Buffer Schemes for Low Power. *Proceedings of the International Conference on Computer Design*.

Kucuk, G. Ponomarev, D., Ergin, O., & Ghose, K. (2004). Complexity-Effective Reorder Buffer Designs for Superscalar Processors. *IEEE Transactions on Computers, vol.53, No.6* (pp.653-665).

Raasch, S., & Reinhardt, S. (2003). The Impact of Resource Partitioning on SMT Processors. *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*.

Sharkey, J., Balkan, D., & Ponomarev D. (2006). Adaptive Reorder Buffers for SMT Processors. *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*.

The Fedora Project, *http://fedoraproject.org/*

M-Sim: The Multi-threaded Simulator,www.cs.binghamton.edu/~ msim

Standard Performance Evaluation Corporation, *www.spec.org*

Performance Analysis of NIHs for Survivable Virtual Topology Mapping[1]

Fatma Corut Ergin FATMA.ERGIN@MARMARA.EDU.TR
Marmara University, Computer Engineering Department

Elif Kaldırım ELIFKALDIRIM@GMAIL.COM
Ayşegül Yayımlı GENCATA@ITU.EDU.TR
Şima Uyar ETANER@ITU.EDU.TR
Istanbul Technical University, Computer Engineering Department

1. Introduction

Optical networking (Mukherjee, 1997) is the most effective technology to meet the high bandwidth network demand. The high capacity of fiber used in optical networks, can be divided into hundreds of different transmission channels, using the WDM technology. Each of these channels work on different wavelengths and can be associated with a different optical connection, called lightpath. All the lightpaths set up on the network form the virtual topology (VT). Given the physical and the virtual topologies, VT mapping problem is to find a proper route for each lightpath.

Any damage to a physical link (fiber) on the network causes all the lightpaths routed through this link to be broken. Survivable VT mapping problem is to design the virtual layer such that the virtual topology remains connected in the event of a single link failure.

Since the VT mapping problem is known to be NP-complete (Modiano & Narula-Tam, 2002), heuristic approaches should be used. In this study, as a solution to the problem, we propose evolutionary algorithms (EA) due to their successful applications on NP-complete problems and ant colony optimization (ACO) due to their successful performance on constrained combinatorial optimization problems. As a result of the experiments, we show that both ACO and EA can solve the problem in less than a minute.

The rest of the paper is organized as follows. In section 2 the problem is defined and related literature is given. Next, the details of how NIHs are applied to our problem are given in section 3. Finally, in section 4, the experimental results are given and discussed thoroughly.

[1] An extended version of this paper can be found in Proceedings of IEEE Globecom'09, Hawaii, USA, Nov.30 - Dec.4, 2009

2. Problem Definition and Related Work

In this work, given the physical and the virtual topologies, our aim is to find a survivable mapping of the VT. There are two main constraints of the problem. The survivability constraint states that all the lightpaths of a cut-set in VT cannot be routed using the same physical link. Capacity constraint ensures that the number of wavelengths on a physical link does not exceed its capacity. Our objective is to minimize the total number of physical links used in the whole physical topology.

The survivable VT mapping problem was solved using tabu search (Crochat & Le Boudec, 1998), (Nucci et al., 2001), local search (Ducatelle & Gambardella, 2005) and SMART (Kurant & Thiran, 2007) algorithms. ILP (Integer Linear Programming) formulation to the problem is given in (Modiano & Narula-Tam, 2002).

3. Application of the NIHs to the Problem

Designing a solution encoding is crucial in EA and ACO performance. For the solution encoding, first, the k-shortest paths for each lightpath are determined. Then, a solution candidate is represented as an integer string of length *l*, where *l* is the number of lightpaths in the VT. Each location on the solution string gives the index of the shortest path for the corresponding lightpath, which can take on values between $[1..k]$, where *k* is the predefined number of shortest paths for the lightpaths.

In EA, constraint violations are considered as penalties in the fitness evaluation stage. In ACO, constraints are taken into consideration during solution construction, therefore constraint violation is not possible.

3.1 Evolutionary Algorithm Design

A steady state EA with duplicate elimination is used. After a random initial population generation, binary tournament

selection and uniform crossover are applied as EA operators. If mutation occurs on a gene, its current value is replaced by the index of the least similar shortest path for the corresponding lightpath, similarity being defined as the number of common physical links. The offspring replaces the worst individual in the population.

Violations of the constraints for the problem are included as penalties in the fitness function. The penalty for an unsurvivable solution is determined as the sum of the total number of lightpaths that become disconnected in the event of each physical link failure (Crochat & Le Boudec, 1998). A capacity constraint violation adds a penalty value which is proportional to the total number of physical links which exceed the predetermined wavelength capacity. These two penalties are multiplied with a penalty factor and added to the fitness of the solution.

3.2 Ant Colony Algorithms Design

In this study, we chose elitist ant system (EAS) as the ACO algorithm. The ants are sorted in decreasing order according to the quality of the solutions they constructed.

Solutions are constructed by applying the following simple constructive procedure to each ant: (1) choose a start lightpath and one of its shortest paths, (2) use lightpath pheromone to select the next lightpath (3) use shortest path pheromone together with heuristic values to probabilistically determine the path between the nodes of the corresponding lightpath, until all lightpaths have been visited. If the ant cannot select a shortest path that makes the solution feasible, i.e., all alternative shortest paths violate the survivability and the capacity constraints, this ant is removed from the current iteration.

3.3 An Example

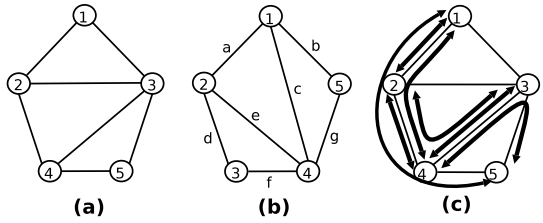


Figure 1. a. Physical Topology, b. Virtual Topology c. Mapping for Example Individual [1 1 2 3 1 1 2]

Consider the physical and virtual topologies given in Figure 1. In Table 1, the second row shows the lightpaths as source-destination node pairs. Three shortest paths for the corresponding lightpath are given in the following rows.

Assume we have a solution encoded as [1 1 2 3 1 1 2]. This encoding means that the lightpaths a and c use the 1^{st} shortest path ((1-2) and (1-2-4), respectively) and lightpath b uses 2^{nd} shortest path (1-2-4-5), etc. If we sum up the number of physical links used in this solution, we have 12 as the resource usage.

For this sample solution, a failure on the physical links connecting nodes 1-2, 2-4, or 3-4, would result in the virtual topology disconnection. If 1-2 link is broken, 3 lightpaths (a,b, and c), if 2-4 link is broken, 4 lightpaths (b,c,d, and e), and if 3-4 link is broken, 2 lightpaths (d and f) would not find an alternative path to communicate. As a result, in EA, a penalty of $9 * p$ is added to the fitness, where p is the penalty factor. On the other hand, since ACO checks the constraints during solution construction, such a solution would not be generated.

Table 1. Three different shortest paths for the lightpaths of the example virtual topology given in Figure 1.

	lightpath						
	1-2 (a)	**1-4 (c)**	**1-5 (b)**	**2-3 (d)**	**2-4 (e)**	**3-4 (f)**	**4-5 (g)**
sp_1	1-2	1-2-4	1-3-5	2-3	2-4	3-4	4-5
sp_2	1-3-2	1-3-4	1-2-4-5	2-1-3	2-3-4	3-2-4	4-3-5
sp_3	1-3-4-2	1-3-2-4	1-2-3-5	2-4-3	2-1-3-4	3-5-4	4-2-3-5

In EA, if a mutation occurs on the second gene of this sample individual, the new individual becomes [1 2 2 3 1 1 2].

4. Experiments

In this section, we present the results obtained from the experiments to evaluate the efficiency of our NIHs. For performance comparisons, we used two metrics, namely success rate, and resource usage. Success rate is the percentage of program runs in which a solution that does not violate the constraints is found. Resource usage is the total number of physical links used throughout the network.

For the experiments, we used two different physical topologies: the 14-node NSF network and a 24-node network (see (Mukherjee, 1997) chapter 11 pp.557). For each physical topology we created 100 random VTs with average connectivity degrees of 3, 4, and 5. We assumed 10 wavelengths per physical link.

4.1 Experimental Results

We performed our initial experiments on the NSF network. Both EA and ACO were able to find feasible solutions of equal quality for all VTs, with 100% success rate. In this paper we report the results for the larger network. For the experimentation, we selected the EA and ACO with the most promising parameter sets according to a set of tests. For each algorithm and VT connectivity degree, we examined three different numbers of alternative shortest paths: 5, 10, and 15.

The results of the experiments are given in Tables 2- 5. Table 2 shows the success rates of both heuristics averaged over 2000 runs (20 runs per VT instance) while Table 3 shows the average of resource usages calculated using only the results of the successful runs. The comparison of our NIHs with basic and relaxed ILP are given in Tables 4 and 5.

A quick observation of Table 2 shows that success rates for both algorithms are very high. Generally ACO achieves higher success rates. From Table 2, we can see that success rates increase with the increase in the number of alternative shortest paths.

Table 2. Success rates

	5 shortest paths		10 shortest paths		15 shortest paths	
	EA	ACO	EA	ACO	EA	ACO
3	97	98	98	100	97	100
4	99	98	100	100	100	100
5	100	100	100	100	99	100

The probability of random candidate solutions being survivable increases with the connectivity degree of the VT. Therefore, for higher connectivity degrees of VTs, both algorithms have higher success rates.

Table 3 shows that the resource usage increases slightly with the increase in the number of alternative shortest paths, especially for EA. This is an expected result, since, the probability of getting stuck at local optima is higher in larger search spaces and both algorithms are allowed to run up to a predefined maximum time. If the run times are increased, the resource usage results for different number of shortest paths will converge.

Table 3. Resource usages

	5 shortest paths		10 shortest paths		15 shortest paths	
	EA	ACO	EA	ACO	EA	ACO
3	111	110	112	110	113	110
4	144	143	145	143	146	144
5	182	181	183	181	186	181

To assess the quality of our solutions, we also implemented basic ILP and ILP Relaxation-1 given in (Modiano & Narula-Tam, 2002), for all connectivity degree VTs. We implemented these ILP formulations using the CPLEX software package. However, since the problem is a large one, we could only solve the problem using basic ILP for only 58% of 3 connected VTs.

Table 4. Number of feasible solutions

	EA	ACO	ILP-Relaxation	Basic ILP
3	100	100	52	58
4	100	100	88	0
5	100	100	100	0

From Table 4, we can see that our NIHs can find a feasible solution for all test sets, although, it is not the case for basic and relaxed ILP. Moreover, if we compare the resource usages of our feasible solutions to the optimum ones found using basic or relaxed ILP as in Table 5, we can see that at least 97% of our solutions are the optimum.

Table 5. Number of solutions having optimum resource usage

	3	4	5
EA	97	97	100
ACO	98	98	100

5. Conclusion

High success rates show that both heuristics are promising for the survivable VT mapping problem. Since the time needed to find a feasible solution is less than a minute, these heuristics can easily be applied to real world applications.

References

Crochat, O., & Le Boudec, J. Y. (1998). Design protection for wdm optical networks. *Journal on Selected Areas in Communications*, 1158–1166.

Ducatelle, F., & Gambardella, L. M. (2005). Survivable routing in ip-over-wdm networks: An efficient and scalable local search algorithm. *Optical Switching and Networking, 2(2)*, 86–99.

Kurant, M., & Thiran, P. (2007). Survivable routing of mesh topologies in ip-over-wdm networks by recursive graph contraction. *IEEE Journal on Selected Areas in Communications, 25(5)*, 922–933.

Modiano, E., & Narula-Tam, A. (2002). Survivable lightpath routing: A new approach to the design of wdm-based networks. *IEEE Journal on Selected Areas in Communications, 20(4)*, 800–809.

Mukherjee, B. (1997). *Optical communication networks*. New York: McGraw-Hill.

Nucci, A., Sanso, B., Crainic, T., Leonardi, E., & Marsan, M. A. (2001). Design of fault-tolerant virtual topologies in wavelength-routed optical ip networks. *Proceedings of IEEE Globecom*.

Preprocessing with Linear Transformations that Maximize the Nearest Neighbor Classification Accuracy

Mehmet Ali Yatbaz MYATBAZ@KU.EDU.TR
Deniz Yuret DYURET@KU.EDU.TR
Computer Engineering, Koç University, Istanbul, Turkey

1. Introduction

In this paper we explore the use of linear transformations as a preprocessing step in classification problems. Our algorithm, STRETCH, seeks a linear transformation of the input features that maximizes the nearest neighbor classification accuracy on the training set. The goal is to bring instances in like-classes closer together, or equivalently to push the instances from different classes further apart. The resulting transformation, when successful, widens the boundaries between classes, and reduces the relative effect of redundant, irrelevant, interacting, or noisy features. We demonstrate that preprocessing with such transformations significantly benefit nearest neighbor (NN) algorithm.

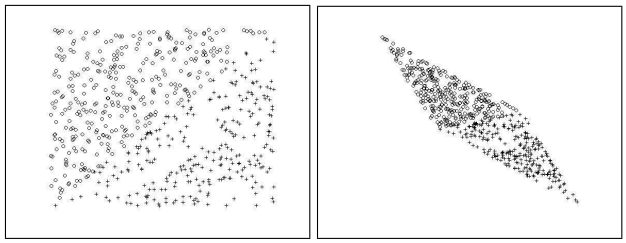


Figure 1. An example of STRETCH at work. The figure on the left is the original data in two dimensions with two classes indicated by circles and pluses. The one on the right shows the same points after stretching.

Figure 1 illustrates the effect of STRETCH on an artificial example. Two classes of points are separated by a diagonal boundary. Some of the points at the boundary are closer to points in the other class instead of their own class. Leave-one-out cross validation of the one nearest neighbor algorithm (1-NN) with the original data set indicates 11 misclassified points. STRETCH transforms the input space by emphasizing the direction orthogonal to the boundary and by reducing the effect of the direction parallel to the boundary. Stretching as illustrated by the right pane in Figure 1 eventually reduces the number of misclassified points to zero.

STRETCH constructs the transform incrementally. At each iteration a random instance is picked and its closest neighbor that is not in the same class is located. We then calculate a linear transform that stretches the input space to push these two points apart. If this transform improves the overall 1-NN performance it is accepted and applied to the data set. The whole process is repeated until no further improvement can be made. The composition of these transforms is the output of the algorithm.

Transformations of the input space have been used to reduce the adverse effects of irrelevant and redundant attributes in instance based learning(Dasarathy, 1991). The focus has been on feature weighting schemes, where each feature is scaled by a constant when computing the distance between two points (see (Wettschereck et al., 1997) for a review). Various approaches have been used to optimize the weights: RELIEF uses incremental update rules similar to the STRETCH algorithm(Kira & Rendell, 1992; Kononenko, 1994). VSM uses conjugate gradient methods (Lowe, 1995; Wettschereck, 1995a) and CCF uses a probabilistic model (Creecy et al., 1992). VDM uses different weight vectors for different parts of the instance space (Stanfill & Waltz, 1986; Aha & Goldstone, 1992). Algorithms that employ more general transformations than basic feature weighting include QM2m (Mohri & Tanaka, 1994) which uses PCA to transform the input features before assigning weights, and IB3-CI (Aha, 1991) which employs domain specific knowledge to construct good feature combinations.

Basic feature weighting corresponds to the specific subset of linear transformations that have diagonal matrices. In contrast, STRETCH can generate arbitrary linear transformations, thus includes feature weighting as a special case. The transformations generated by STRETCH are not restricted to principle components and do not require domain specific knowledge.

2. Algorithm

This section describes the STRETCH algorithm. We will start with the mathematics of constructing matrices that stretch the distance between two points relative to orthogonal directions. This is followed by a description of the main loop that constructs the target linear transform.

Stretch matrices: Consider two instances, x_1 and x_2 as points in $\Re^n$. Let $v = x_1 - x_2$ be the difference vector. The following procedure will construct the matrix of the linear transform that will increase the distance between x_1 and x_2 by a factor α relative to the directions orthogonal to v:

1. Construct an orthogonal basis for $\Re^n$ with the first basis vector parallel to v. Here is one way to do this:
   (a) Let M be an $n \times n$ square matrix.
   (b) Set the first column of M to be the vector v.
   (c) Set the other columns to be different unit vectors making sure the resulting matrix does not have an all zero row.
   (d) Compute the QR decomposition of M.
   (e) The resulting Q will be an orthogonal matrix with its first column parallel to v.
2. Construct a diagonal matrix D with the first diagonal entry α and the other diagonal entries 1.
3. Return the matrix QDQ^T. The first eigenvector of QDQ^T is parallel to v and the first eigenvalue is α. The other eigenvectors are orthogonal to v with corresponding eigenvalues equal to 1.

The main loop: The goal is to construct a linear transform that moves the instances around such that the maximum number of instances belong to the same class as their closest neighbor. Our algorithm accomplishes this incrementally, trying to fix one instance at a time. Let us call the number of instances that have the same class as their closest neighbor the LOO1NN (leave-one-out 1-NN) score. Given a set of instances X in $\Re^n$ and a learning rate α, we initialize the target transformation matrix A to be the identity matrix and repeat the following steps until convergence:

1. Construct misclassified instance set of 1NN and pick a random misclassified instance $x \in X$.
2. Let z be the closest neighbor of x that belongs to a different class. Construct a stretch matrix Z that increases the distance between x and z by α.
3. Calculate the regularization term of E_{AZ} for the updated stretch matrix AZ.
4. If LOO1NN(AZX) + $E_{AZ} \geq$ LOO1NN(AX) + E_A then $A = AZ$.

Convergence: The candidate moves are accepted as long as they do not decrease the accuracy in the training set and distorts the data. However, this default behavior generally results in the convergence on a local maximum. In order to get out of local maxima, we occasionally allow jumps (bad moves) to be made. Specifically, after trying all boundary (misclassified) points and finding no good moves, a random boundary point is picked and its associated stretch matrix is incorporated into the transform. These moves usually allow the algorithm to get out of local maxima.

Normalization: One potential problem during these iterations is floating point overflows. To prevent this the cumulative transform matrix A is normalized by dividing it with the volume of $Volume_A$ after each modification. The $Volume_A$ is defined in terms of the singular vectors of A. Since the singular vectors are orthonormal to each other, we can think of these vectors as the vertices of a prism.

$$A = QVQ^T \tag{1}$$

where Q and V are an orthonormal and a diagonal matrix, respectively. The singular column vectors of Q have only the direction information and their lengths equal to 1. The length of each singular vector is kept in the corresponding diagonal entry of V. The volume of A is given by $Volume_A = \prod_{i=1}^{n} v_i$, where v_i is the i^{th} diagonal entry of V.

Regularization: Initially each singular value is equal to 1 and any linear transformation updates these singular values. However , if the algorithm always stretches along the direction that v_1 corresponds then v_1 becomes larger compared to the rest of the singular values as the number of iteration increases. To punish distortion of the data we define the regularization cost E_{AZ} of a stretch matrix Z as $E_{AZ} = \sum_{i=1}^{n} (\log v_i)^2$. As a result, Z is accepted only if the LOO1NN(AZX) is less than the LOO1NN(AX) while it does not distort the data significantly compared to the other possible linear transformations.

3. Experiments

In this section we empirically analyze the impact of STRETCH on 1NN algorithm using datasets from the UCI repository(Asuncion & Newman, 2007). Each dataset is randomly divided into 70% training and 30% testing portions. Half of the training set is used as a validation set to prevent overfitting and optimization. The best transform on the validation set is determined and the learning algorithm is trained on the transformed data. Later the same transform is applied to the test set and the learned model is evaluated. It is important not to taint the preprocessing by using the test set because STRETCH makes use of the class information when constructing the best transform.

Table 1. Percentage of instances accurately classified by their nearest neighbor on the training and the test data before and after the application of the STRETCH with 400 iterations and 100 jumps. Changes in bold format are statistically significant.

Dataset	Before	1NN Train	1NN Test
Banded	0.82±0.01	**0.97±0.03**	**0.95±0.03**
Sinusoidal	0.86±0.01	**0.95±0.02**	**0.93±0.02**
Gauss-band	0.63±0.02	**0.84±0.03**	**0.75±0.04**
Parity	0.66±0.02	**1±0.00**	**98±0.02**
LED-7 Display	0.51±0.05	**0.53±0.05**	0.51±0.05
LED-7+17B	0.38±0.03	**0.73±0.04**	**0.43±0.02**
LED-7+17C	0.60±0.04	**0.81±0.03**	**0.58±0.03**
Waveform-21	0.78±0.02	**0.93±0.01**	0.77±0.03
Waveform-40	0.67±0.03	**0.96±0.02**	**73±0.04**
Cleveland	0.76±0.03	**0.84±0.02**	0.77±0.05
Hungarian	0.77±0.02	**0.83±0.01**	**0.79±0.03**
Voting	0.92±0.01	**0.96±0.01**	**0.94±0.02**

Performance: Table 1 illustrates the impact of the STRETCH transform on the LOO1NN score of the training set and the test set. In each case the training set error is reduced by the resulting linear transform. However as we see in the 1-NN Test column of Table 1, this improvement is not always reflected on the test set.

4. Contributions

We have introduced a preprocessing algorithm for classification problems based on linear transformations called STRETCH. The algorithm tries to find the transformation that maximizes the number of instances which have the same class as their closest neighbor. Earlier feature weighting algorithms can be represented as linear transformations with diagonal matrices. In contrast STRETCH is capable of constructing any transformation.Experiments with a number of standard data sets indicate that most of the time STRETCH can find a transform that increases the nearest neighbor score of the training set to near 100%. The test set results for 1-NN with and without the STRETCH transform indicate improvements in most of the datasets.

References

Aha, D., & Goldstone, R. (1992). Concept learning and flexible weighting. *Proceedings of the Fourteenth Annul Conference of the Cognitive Science Society* (pp. 534–539). Bloomington, IN: Lawrence Erlbaum.

Aha, D. W. (1991). Incremental constructive induction: An instance-based approach. *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 117–121). Evanston, LI: Morgan Kaufmann.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository.

Creecy, R., Masand, B. M., Smith, S. J., & Waltz, D. L. (1992). Trading mips and memory for knowledge engineering. *Communications of the ACM*, 48–64.

Dasarathy, B. V. (Ed.). (1991). *Nearest neighbor (nn) norms: Nn pattern classification techniques*. Los Alamitos,CA: IEEE Computer Society Press.

Kira, K., & Rendell, L. (1992). A practical approach to feature selection. *Proceedings of the Ninth International Conference on Machine Learning.* (pp. 249–256). Aberdeen, Scotland: Morgan Kaufmann.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of relief. *Proceedings of the 1994 European Conference on Machine Learning.* (pp. 171–182). Catania,Italy: Springer Verlag.

Lowe, D. (1995). Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 72–85.

Mohri, T., & Tanaka, H. (1994). An optimal weighting criterion of case indexing for both numeric and symbolic attributes. *Case-Based Reasoning: Papers from the 1994 Workshop*. Menlo Park,CA: AAAI Press.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the Association for Computing Machinery*, 1213–1228.

Wettschereck, D. (1995a). *A description of the mutual information approach and the variable similarity metric.* (Technical Report). German National Research Center for Computer Science,Artificial Intelligence Research Division, Sankt Augustin,Germany.

Wettschereck, D., Aha, D. W., & Mohri, T. (1997). A review and comparative evaluation of feature weighting methods for lazy learning algorithms. *Artificial Intelligence Review*, 273–314.

QED: A Proof System for the Static Verification of Concurrent Software

Tayfun Elmas TELMAS@KU.EDU.TR
Ömer Subaşı OSUBASI@KU.EDU.TR
Koç University, Rumeli Feneri Yolu, Sariyer, İstanbul, Turkey

1. Introduction

Because of emerging technologies such as multicore processors and grid computing, concurrency is becoming an important issue of today's software systems and is likely to become even more so in the future. A wide-range of systems including web servers, databases, and operating systems contain highly concurrent data structures, e.g., a binary tree, and services, e.g., a persistence manager, in order to respond efficiently to a large number of simultaneously-accessing clients. Such software makes use of sophisticated synchronization techniques, including fine-grained locking and non-blocking operations, and creates extra threads for internal operations, for example, to re-balance a binary tree. These techniques require intensive care to use, and bugs due to incorrect use of them can have serious consequences, such as data corruption, operating system crash, or even more catastrophic results, e.g., failure of an aircraft flight control system. Therefore, the functional correctness of software is as fundamental as its performance.

This paper focuses on statically verifying the (partial) functional correctness of concurrent software. The correctness is specified using two criteria: *validity of assertion*s and *linearizability*. Assertions state local conditions expected to hold when the execution of the program reaches a certain location in the code. Linearizability formalizes the requirement that a concurrently-accessed data structure implementation conform to a sequential specification containing atomic versions of the data structure's operations (Herlihy & Wing, 1990). That is, every operation of the implementation takes effect instantly between call and return points where the effect is determined by a corresponding operation of the specification. We provide a formal proof system and a supporting software tool using which one can show that 1) no execution of a program leads to an assertion violation and 2) every concurrent execution of a data structure is equivalent to an execution of its atomic specification.

2. Background and Related Work

Because the full precise static verification is undecidable, the verification of a program is decomposed into decidable pieces, requiring code annotations from the user. These code annotations express facts about the computations performed by the program and include, for sequential programs, loop invariants, procedure pre- and post-conditions. The effectiveness of the proof is highly dependent on the (manual) selection of the annotations. Annotations for concurrent programs require significantly more intellectual effort to state than those for sequential programs. The fundamental reason behind this is the interaction between threads over the shared memory: while writing the annotations for a program under fine-grained concurrency, one has to consider possible interleavings of a large number of conflicting operations.

In the invariant-based approaches, e.g. Owicki-Gries (Owicki & Gries, 1976), each potential interleaving point in a program must be annotated with an invariant that is valid under interference from other concurrently-executing actions. In other words, each invariant must remain true even after another thread overwriting shared variables is scheduled at the corresponding interleaving point. Rely-guarantee methods (Jones, 1981) make this approach more modular by obviating the need to consider each pair of concurrent statements separately. Both these methods require the programmer to reason about interleavings of fine-grained actions; consequently, the annotations are complex. Concurrent separation logic (O'Hearn et al., 2004) has the ability to maintain separation between shared and local memory, enabling sequential reasoning for multithreaded programs; however, this is not particularly useful for programs with high level of interference.

Fine-grained concurrency also complicates establishing an abstraction map and identifying the commit points of operations while proving linearizability (Herlihy & Wing, 1990; Vafeiadis et al., 2006). Managing these requirements when the operations of a data structure are written in terms of many small actions that make visible changes to the state requires considerable expertise.

Atomicity is a well-known specification for concurrently executed code blocks. A code block is said to be *atomic* every interleaved execution of the block is equivalent to an indivisible execution of the block. Several verification approaches were developed to verify atomicity (Flanagan & Qadeer, 2003; Freund & Qadeer, 2004; Wang & Stoller, 2005), using reduction as a key ingredient. Their use of reduction is limited to simple synchronization disciplines and can only reason about commutativity of accesses that are not simultaneously enabled. In order to enable wider application of reduction, they require using auxiliary variables, which are additional program variables that are used to express extra facts about the program, and access predicates, which are annotations indicating the synchronization disciplines in a program. Abstractions have been used as a mechanism to prove atomicity in the work on purity (Freund & Qadeer, 2005; Wang & Stoller, 2005).

On the other hand, our approach uses atomicity as a reasoning tool to enable more tractable verification of other specifications such as linearizability. Moreover, we use more flexible notions of reduction and abstraction. Our method is orthogonal and complementary to existing methods that do not make direct use of reduction and abstraction, by enhancing their applicability, and subsumes others that do.

3. QED Proof System

The primary contribution of this work is a static proof system, called QED, for proving assertions (Elmas et al., 2009a) and linearizability (Elmas et al., 2010). QED provides a novel proof strategy in which *atomicity is used as a proof tool*: a program with fine-grained concurrency is transformed iteratively to make it consist of larger atomic actions, and the correctness conditions are checked after the program reaches an atomicity level appropriate for local analyses, which can analyze a individual code blocks without needing the entire program. This gradually reduces the influence of thread interleavings on the complexity of the reasoning, and therefore permits significantly more tractable proofs than those provided by existing methods. For example, we prove assertions by performing *sequential (local)* checks within atomic blocks of the final program. The transformations preserve or expand the set of behaviors of the program, so that assertions proved at the end of a sequence of transformations are valid in the original program. In addition, one can simplify a program with larger atomic blocks using QED and can continue the proof with another method, e.g., separation logic (O'Hearn et al., 2004).

A distinguishing and essential aspect of our method is the iterative and alternating application of two kinds of transformations, *abstraction* and *reduction*, which allows us to reach the desired level of atomicity even when there is apparent interference between threads. While reduction and abstraction have been studied in isolation in the literature, they are *symbiotic* in QED. Reduction (Lipton, 1975) replaces a compound statement consisting of several atomic actions with a single atomic action if certain non-interference conditions hold and allows a subsequent abstraction step to summarize the entire calculation in that statement locally. Reduction uses the commutativity of actions. If an action commutes with all other actions, then it can be merged with one of the action. Abstraction replaces an atomic action with a more relaxed atomic action allowing more behaviors, permitting a later application of reduction to reason that it does not interfere with other atomic actions and merge it with other actions.

In addition to checking validity of assertions, our proof system simplifies verifying linearizability by using abstraction and reduction. We prove an implementation is linearizable with respect to a sequential specification by transforming the implementation to the specification in two phases. In reduction phase, we use reduction and abstraction defined above to increase atomic code blocks. In refinement phase, we transform code closer to the specification. Refinement phase help us to remove conflicts that are not real conflicts in terms of the specification such as conflicts regarding some implementation variables. In this manner, we construct *abstraction map* incrementally. *Abstraction map* is usually used in order to establish a linearizability proof which relates the states of the implementation to the states of the specification. Our method does not require identification of commit points in the implementation. Commit points are the points when a concurrent operation effect become visible. The place of these points depends on concurrent executions. The commit point of a procedure may be inside or outside of the procedure.

4. Implementation and Experience

We implemented our verification method in a software tool, also called QED. QED is open source and can be downloaded from **http://qed.codeplex.com**.

Our tool accepts as input a multithreaded program in QEDPL language and a proof script. QEDPL is an extension of the Boogie programming language (DeLine & Leino., 2005) of Microsoft Research with concurrency constructs, e.g., thread creation. We are also developing a translator from Java to QEDPL. The proof script contains a sequence of proof commands. A proof command is used for one of two purposes: to transform the input program using abstraction, reduction or a combination of the two, or to provide a concise specification of the behavior of the current version of the program, including locking protocols and data invariants. The tool automatically generates the verification conditions justifying each step of the proof

and verifies them using Z3 (de Moura & Bjørner, 2008), a state-of-the-art SMT solver developed by Microsoft Research. After executing each step in the proof script, QED allows the user to examine the resulting program, intercept the proof, and give new commands.

We also provide a set of proof script templates. These templates document and mechanize proof idioms, a sequence of low-level proof rules applied for a common scenario, for example, indicating that a variable is always lock-protected. We presented the proof idioms for common synchronization mechanisms, such as mutex and reader/writer locks in (Elmas et al., 2009b).

We have evaluated QED by verifying a number of multithreaded programs with varying degree of synchronization complexity. These examples include programs using fine-grained locking and non-blocking data structures. We have found that the iterative approach embodied in QED provides a simple and convenient way of communicating to the verifier the programmer's understanding of and correctness arguments on the computation and synchronization in the program. Thus, the proofs in our method are invariably cleaner than the proofs based on existing approaches.

The proof script of a program serves as a *reproducible* (by our tool) documentation of its correctness. Therefore, QED provides a proof repository for a collection of concurrent software including the following purity benchmarks from (Freund & Qadeer, 2005), fine grained multiset and lock-coupling linked list implementations, and non-blocking stack, queue, deque, and readers/writer lock implementations (Herlihy & Shavit, 2008).

Contributions:
– A novel proof system for verifying assertions in and the linearizability of multithreaded programs.
– A tool that implements our proof system using a set of concise and machine-checked proof commands.
– Evaluation of our technique and tool on a variety of small to medium-sized multithreaded programs.

References

de Moura, L., & Bjørner, N. (2008). *Z3: An efficient smt solver*, vol. 4963/2008 of *Lecture Notes in Computer Science*, 337–340. Springer Berlin.

DeLine, R., & Leino., K. R. M. (2005). Boogiepl: A typed procedural language for checking object-oriented programs. *Technical Report MSR-TR-2005-70, Microsoft Research*.

Elmas, T., Qadeer, S., Sezgin, A., Subasi, O., & Tasiran, S. (2010). Simplifying the proof of linearizability with reduction and abstraction. *(To appear) TACAS 2010: Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*.

Elmas, T., Qadeer, S., & Tasiran, S. (2009a). A calculus of atomic actions. *POPL '09: ACM Symposium on Principles of Programming Languages*. New York, NY, USA: ACM.

Elmas, T., Sezgin, A., Tasiran, S., & Qadeer, S. (2009b). An annotation assistant for interactive debugging of programs with common synchronization idioms. *Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging*.

Flanagan, C., & Qadeer, S. (2003). Types for atomicity. *TLDI '03: Proceedings of the 2003 ACM SIGPLAN international workshop on Types in languages design and implementation* (pp. 1–12). New York, NY, USA: ACM.

Freund, S., & Qadeer, S. (2004). Checking concise specifications for multithreaded software. *Journal of Object Technology*, *3*, 81–101. Special issue: ECOOP 2003 workshop on FTfJP.

Freund, S. N., & Qadeer, S. (2005). Exploiting purity for atomicity. *IEEE Trans. Softw. Eng.*, *31*, 275–291. Member-Cormac Flanagan.

Herlihy, M., & Shavit, N. (2008). *The art of multiprocessor programming*. Morgan Kaufmann.

Herlihy, M. P., & Wing, J. M. (1990). Linearizability: a correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, *12*, 463–492.

Jones, C. B. (1981). *Development methods for computer programs including a notion of interference*. Doctoral dissertation, Oxford University. Printed as: Programming Research Group, Technical Monograph 25.

Lipton, R. J. (1975). Reduction: a method of proving properties of parallel programs. *Commun. ACM*, *18*, 717–721.

O'Hearn, P. W., Yang, H., & Reynolds, J. C. (2004). Separation and information hiding. *POPL '04: Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (pp. 268–280). New York, NY, USA: ACM.

Owicki, S., & Gries, D. (1976). Verifying properties of parallel programs: an axiomatic approach. *Commun. ACM*, *19*, 279–285.

Vafeiadis, V., Herlihy, M., Hoare, T., & Shapiro, M. (2006). Proving correctness of highly-concurrent linearisable objects. *PPoPP '06 ACM Symposium on Principles and practice of parallel programming* (pp. 129–136). New York, NY, USA: ACM.

Wang, L., & Stoller, S. D. (2005). Static analysis for programs with non-blocking synchronization. *ACM SIGPLAN 2005 Symposium on Principles and Practice of Parallel Programming (PPoPP)*. ACM Press.

Quantifying Solutions in Answer Set Programming

Halit Erdogan HALIT@SABANCIUNIV.EDU
Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, TURKEY

1. Introduction

Answer Set Programming (ASP) (Lifschitz, 2008) is a form of declarative programming oriented towards difficult (NP-hard) search problems. A problem is represented as a logic program whose answer sets correspond to the solutions. The answer sets for the given formalism can be computed by special systems called answer set solvers. Due to the expressive representation language and continuous improvements of efficiency of solvers, ASP can be useful for a wide-range of knowledge-intensive applications from different fields such as: developing a decision support system for a Space Shuttle (Nogueira et al., 2001); phylogeny reconstruction (Brooks et al., 2007); multi-agent planning (Son et al., 2009).

Finding *a* solution to the problem that we are interested in is generally the main objective. But many problems have numerous solutions, instead of a single solution. At that case it might be desirable to compute a subset of preferred solutions, instead of computing all the solutions. Consider, for instance, a product configuration problem: Suppose you want to buy a car. There are various constraints on what type of car you want. A product advisor system might consider your constraints and what is available for sale and offer you a set of cars. But you might not have enough time to consider all the cars that are offered by the system. At that case, it might be desirable to quantify the solutions (cars) offered by the system. For example, you might like to see the similar cars similar to a car that you selected. Or you might like see the cars ranked according to a preference function (from cars to real numbers) that you determined. Motivated by such an application we propose novel methods to quantify solutions in answer set programming. We developed a system called CLASP-NK which is a modification of the existing ASP solver CLASP (Gebser et al., 2007). CLASP-NK takes an ASP description of a problem and a preference function on solutions (in C++); and it outputs quantified solutions to the problem based on the preference function. More details and results can be found in (Eiter et al., 2009); since this paper is a summary of (Eiter et al., 2009).

2. Answer Set Programming

The idea of answer set programming (ASP) (Lifschitz, 2008) is to represent a computational problem as a logic program whose answer sets correspond to the solutions of the problem and to find the answer sets for that program by using an answer set solver.

Two kind of rules play the major role in ASP: those that "generate" many answer sets corresponding to possible solutions, and those that "test" the possible solutions and eliminate the ones that does not correspond to a solution.

For example, recall that a *clique* in a graph is a set of pairwise adjacent vertices. Suppose that we are interested in the *cliques* whose size is at least 10 then we can represent the problem in ASP as follows (Lifschitz, 2008):

```
10 {select(X) : vertex(X)}.
:- select(X), select(Y), vertex(X),
   vertex(Y), X!=Y, not edge(X,Y),
   not edge(Y,X).
```

The first rule correspond to the "generate" part. It generates the possible solutions that contains at least 10 vertices (*select* atoms correspond to the selected vertices). The second rule is a constraint that checks whether the selected vertices correspond to a clique. To use this program, we combine it with a description of the graph, such as:

```
vertex(1..99). % 1,...,99 are vertices
edge(3,7).     % 3 is adjacent to 7
. . .
```

When we run this ASP program in an ASP solver, if there exists a clique of at least size 10 then the solver outputs answer sets. Each answer set corresponds to a clique and the *select* atoms in the answer set defines the vertices that constitutes that clique.

3. Similar Solutions in ASP

Consider the *clique* example of the previous section and suppose that the maximum clique size is three. Assume that for each three consecutive vertices we have a clique: $\{\{1,2,3\},\{4,5,6\},\{7,8,9\},...\}$. In addition we have the

following two cliques: $\{\{1,2,4\},\{1,2,5\}\}$. Now suppose that we would like to find the three most similar cliques in the graph (a set of three cliques where the number of differentiating atoms (Hamming Distance) are minimum). At that case three most similar cliques of the graph will be: $\{\{1,2,3\},\{1,2,4\},\{1,2,5\}\}$ since they only differ with one vertex.

In this example we quantify the solutions based on their similarity to other solutions. And we accomplish that by defining a distance measure for a set of solutions. We used the simple Hamming Distance but more complex problems might require more complex distance measures. Motivated by such an example we defined the following decision problem:

> n k-SIMILAR SOLUTIONS
> Given an ASP program $\mathcal{P}$ that formulates a computational problem P, a distance measure Δ that maps a set of solutions for P to a nonnegative integer, and two nonnegative integers n and k, decide whether a set S of n solutions for P exists such that $\Delta(S) \leq k$.

In the next section we introduce methods to solve the problem with ASP.

4. Computing Similar Solutions in ASP

Offline Method We can compute a set of n k-similar solutions to a given problem, by computing all solutions in advance and then using some clustering methods to find the similar solutions. The idea is to make clusters of n solutions, measure the distance of the set of solutions in each cluster, and pick the cluster whose distance is less than k.

Online Method 1 The idea of this method is to solve the problem by describing it in ASP. This method reformulates the given program $\mathcal{P}$ to compute n-distinct solutions, formulates the distance function Δ as an ASP program $\mathcal{D}$, and formulates constraints on the distance function as an ASP program $\mathcal{C}$, so that all n k-similar solutions can be extracted from an answer set for the union of these ASP programs, $\mathcal{P} \cup \mathcal{D} \cup \mathcal{C}$.

Online Method 2 This is an approximate method to solve the problem. In this method we do not modify the given ASP program $\mathcal{P}$, but formulate the distance $\Delta(S)$ of a given set S of solutions as an ASP program $\mathcal{D}$, and constraint on the distance function as an ASP program $\mathcal{C}$, so that a k-close solution can be extracted from an answer set for $\mathcal{P} \cup \mathcal{D} \cup \mathcal{C}$. By iteratively computing a k-close solution, we can compute a set of n k-similar solutions.

Online Method 3 This is also an approximate method to solve the problem. This method does not modify the given program, and does not formulate the distance function as an ASP program, but it modifies the ASP solver CLASP (Gebser et al., 2007) to compute all n k-similar solutions at once. As a result, we developed a system called CLASP-NK which is capable of computing n k-similar solutions. CLASP-NK takes the ASP program $\mathcal{P}$ and the C++ definition of Δ as input and outputs n k-similar solutions.

CLASP performs a type of branch and bound algorithm. In each step it decides an atom to be added to the answer set (branch). And according to that atom it propagates other atoms that shall be included in the answer set. Then it checks whether there is a conflict by considering the rules of the program. If there exists such a conflict CLASP learns the conflict (to not to repeat it) and performs a backtracking (bound).

CLASP-NK contains a slight modification on the bounding procedure of CLASP. At each step —after CLASP decides the atoms to include to the answer set— CLASP-NK performs an extra check based on the input distance measure. If the currently selected atoms (at the level we are in) violates the distance measure (in the context of similar solutions it means that it is impossible to compute a solution which is similar to the previously computed solutions if we continue branching) then we set those atoms as conflict and perform a backtracking. Therefore, CLASP-NK is forced to compute a similar solution to the previously computed solutions

5. Computing Similar Phylogenies

Phylogenetics studies the evolutionary relationships between taxonomic units (e.g., species) based on their shared traits. These relations are represented as a tree whose leaves represent the taxa, internal vertices represent their ancestors, and edges represent the relationships between them. Such a tree is called "phylogenetic tree" or "phylogeny". Phylogeny reconstruction is an NP-hard problem and there exists ASP programs that can infer phylogenies (Brooks et al., 2007). But in many cases the phylogeny reconstruction programs output numerous phylogenies that describe the historical evolution of the same species. At those cases experts go over these phylogenies manually to find the most plausible ones. Instead of computing all the phylogenies, the experts want to compute a set of similar phylogenies to perform better analysis. Therefore we defined n k-similar phylogenies problem analogous to the n k-similar solutions problem. We used the ASP program of (Brooks et al., 2007) to compute a phylogeny (solution), and we used a distance measure from the literature to compute the distance between phylogenies. And we run some experiments to test the methods described in the previous section. The table below shows the performance of each method from the point of view of computation time and

memory.

Problem	Offline Method	Online Method 1	Online Method 2	Online Method 3
2 most similar	12.39 sec. 32MB $k = 12$	26.23 sec. 430MB $k = 12$	19.00 sec. 410MB $k = 12$	1.46 sec. 12MB $k = 12$
3 most similar	11.59 sec. 32MB $k = 15$	60.20 sec. 730MB $k = 15$	43.56 sec. 626MB $k = 15$	1.56 sec. 15MB $k = 16$
6 most similar	11.66sec. 32MB $k = 25$	327.28 sec. 1.8GB $k = 25$	178.96 sec. 1.2GB $k = 29$	1.96 sec. 15MB $k = 25$

Let us first compare the online methods. In terms of both computation time and memory size, Online Method 3 (CLASP-NK) performs the best, and Online Method 2 performs better than Online Method 1. These results conforms with our expectations: Online Method 1 requires an ASP representation of computing n k-similar phylogenies, and such a program may be too large for an answer set solver to compute an answer set for. Online Method 2 relaxes this requirement a little bit so that the answer set solver can compute the solutions more efficiently: it requires an ASP representation of phylogeny reconstruction, and an ASP representation of the distance measure, and then computes similar solutions one at a time. However, since the answer set solver needs to compute the distances between every two solutions, the computation time and the size of memory do not decrease much, compared to those for Online Method 1. Online Method 3 deals with the time consuming computation of distances between solutions, not at the representation level but at the search level; so it does not require an ASP representation of the distance function but requires a modification of the solver.

The offline method is more efficient, in terms of both computation time and memory, than Online Methods 1 and 2 since it does not compute phylogenies. On the other hand, the offline method is less efficient, in terms of both computation time and memory, than Online Method 3, since it requires both representation and computation of distances between solutions.

6. Conclusion

We introduce one offline and three online methods to compute n k-similar solutions in ASP. We developed a system CLASP-NK (Online Method 3) which is capable of computing n k-similar solutions on the fly. We showed the effectiveness and applicability of our methods on phylogenetics domain.

This paper is a proof-of-principle that CLASP-NK is useful for quantifying solutions in ASP. In this paper, we consider quantifying the solutions based on their similarity to other solutions. CLASP-NK's capabilities are not limited to only computing similar solutions in ASP. We can define any preference function so that CLASP-NK can compute the quantified solutions based on that preference function.

Acknowledgments

I am grateful to my supervisor, Esra Erdem, for excellent advice and encouragement. The central ideas in this work are the product of a close collaboration with Esra Erdem, Thomas Eiter and Michael Fink. I also wish to thank Martin Gebser and Benjamin Kaufmann for their help with CLASP.

References

Brooks, D., Erdem, E., Erdogan, S., Minett, J., & Ringe, D. (2007). Inferring phylogenetic trees using answer set programming. *Autom. Reason.*, *39*, 471–511.

Eiter, T., Erdem, E., Erdogan, H., & Fink, M. (2009). Finding similar or diverse solutions in answer set programming. In (Hill & Warren, 2009), 342–356.

Gebser, M., Kaufmann, B., Neumann, A., & Schaub, T. (2007). clasp: A conflict-driven answer set solver. *Proc. of LPNMR* (pp. 260–265). Springer-Verlag.

Hill, P. M., & Warren, D. S. (Eds.). (2009). *Logic programming, 25th international conference, iclp 2009, pasadena, ca, usa, july 14-17, 2009. proceedings*, vol. 5649 of *Lecture Notes in Computer Science*. Springer.

Lifschitz, V. (2008). What is answer set programming? *Proc. of. AAAI* (pp. 1594–1597). AAAI Press.

Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., & Barry, M. (2001). An a-prolog decision support system for the space shuttle. *Proc. of PADL* (pp. 169–183). London, UK: Springer-Verlag.

Son, T., Pontelli, E., & Sakama, C. (2009). Logic programming for multiagent planning with negotiation. In (Hill & Warren, 2009), 99–114.

L_1 Regularization for Learning Word Alignments in Sparse Feature Matrices

Ergun Biçici EBICICI@KU.EDU.TR
Deniz Yuret DYURET@KU.EDU.TR
Department of Electrical and Computer Engineering
Koç University, Istanbul, Turkey

1. Introduction

In statistical machine translation, parallel corpora, which contain translations of the same documents in source and target languages, are used to estimate a likely target translation for a given source sentence based on the observed translations. Sparse feature representations can be used in various domains. When the number of instances, m is significantly smaller than the number of features, n, $m \ll n$, then we have an under determined system of equations.

We examine the effectiveness of regression to find the mappings between sparsely observed feature sets. Regularization of the cost function plays an important role to increase the performance; therefore we experiment with L_1 regularization. We analyze and devise instance selection methods for a given source sentence to increase the performance of the word alignment. The performance is estimated by comparing with the phrase table obtained by GIZA++ (Och & Ney, 2003), which is a state of the art word alignment tool commonly used in phrase-based machine translation systems. GIZA++ combines the result of various statistical word alignment models and performs symmetrization of the generated directed alignments.

2. Regression Based Alignment Learning

Let the feature matrices $\mathbf{M}_X \in \mathbb{R}^{N_X \times m}$ and $\mathbf{M}_Y \in \mathbb{R}^{N_Y \times m}$ be obtained from m training instances such that each column of $\mathbf{M}_X$ ($\mathbf{M}_Y$) is obtained by a feature mapper $\Phi_X : X^* \rightarrow \mathbb{R}^{N_X}$ ($\Phi_Y : Y^* \rightarrow \mathbb{R}^{N_Y}$). The ridge regression solution using L_2 regularization is given in Equation 1:

$$\mathbf{H}_{L_2} = \underset{\mathbf{H} \in \mathbb{R}^{N_Y \times N_X}}{\arg\min} \| \mathbf{M}_Y - \mathbf{H}\mathbf{M}_X \|_F^2 + \lambda \| \mathbf{H} \|_F^2 \quad (1)$$

$$= \mathbf{M}_Y \mathbf{M}_X^T (\mathbf{M}_X \mathbf{M}_X^T + \lambda \mathbf{I})^{-1} \quad (2)$$

$$\mathbf{H}_{L_1} = \underset{\mathbf{H} \in \mathbb{R}^{N_Y \times N_X}}{\arg\min} \| \mathbf{M}_Y - \mathbf{H}\mathbf{M}_X \|_F^2 + \lambda \| \mathbf{H} \|_1 . \quad (3)$$

$\mathbf{H}_{L_2}$ does not give us a sparse solution as most of the coefficients remain non-zero. L_1 norm behaves both as a feature selection technique and a method for reducing coefficient values. Equation 3 presents the *lasso* (least absolute shrinkage and selection operator) (Tibshirani, 1996) solution where the regularization term is now the L_1 matrix norm defined as $\| \mathbf{H} \|_1 = \sum_{i,j} |H_{i,j}|$. $\mathbf{H}_{L_2}$ can be found by taking the derivative but since L_1 regularization cost is not differentiable, $\mathbf{H}_{L_1}$ can be found by optimization or approximation techniques.

We perform experiments with forward stagewise regression (Hastie et al., 2006) (FSR) and quadratic optimization (QP) techniques to find $\mathbf{H}_{L_1}$. The incremental forward stagewise regression algorithm increases the weight of the predictor variable that is most correlated with the residual by a small amount, ϵ, multiplied with the sign of the correlation at each step. As $\epsilon \rightarrow 0$, the profile of the coefficients resemble the *lasso* (Hastie et al., 2001). We can pose *lasso* as a QP problem as follows (Mørup & Clemmensen, 2007). We assume that the rows of $\mathbf{M}_Y$ are independent and solve for each row i, $\mathbf{M}_{y_i} \in \mathbb{R}^{1 \times m}$, using non-negative variables $\mathbf{h}_i^+, \mathbf{h}_i^- \in \mathbb{R}^{N_X \times 1}$ such that $\mathbf{h}_i = \mathbf{h}_i^+ - \mathbf{h}_i^-$:

$$\mathbf{h}_i = \| \mathbf{M}_{y_i} - \mathbf{h}_i \mathbf{M}_X \|_F^2 + \lambda \sum_{k=1}^{N_X} |h_{i,k}| \quad (4)$$

$$\mathbf{h}_i = \underset{\tilde{\mathbf{h}_i}}{\arg\min} \frac{1}{2} \tilde{\mathbf{h}_i} \widetilde{\mathbf{M}_X} \widetilde{\mathbf{M}_X}^T \tilde{\mathbf{h}_i}^T - \tilde{\mathbf{h}_i} (\widetilde{\mathbf{M}_X} \mathbf{M}_{y_i}^T - \lambda \mathbf{1}) \quad (5)$$

$$\text{s.t. } \tilde{\mathbf{h}_i} > 0, \ \widetilde{\mathbf{M}_X} = \begin{bmatrix} \mathbf{M}_X \\ -\mathbf{M}_X \end{bmatrix}, \ \tilde{\mathbf{h}_i} = \begin{bmatrix} \mathbf{h}_i^+ & \mathbf{h}_i^- \end{bmatrix}$$

Orthogonality of the coefficient matrix can be desirable since the L_2 regularization parameter penalizes in proportion to $\mathbf{H}^T\mathbf{H}$ and setting $\mathbf{H}^T\mathbf{H} = \mathbf{H}\mathbf{H}^T = \mathbf{I}$ corresponds to assuming that features are selected independently (i.e. correlation of source and target features is identity). Therefore, we also experiment with symmetric coefficient matrix $\mathbf{H}_S = \sqrt{\mathbf{H} \times \overleftarrow{\mathbf{H}}^T}$, where $\times$ stands for the element-wise multiplication operator and $\overleftarrow{\mathbf{H}}$ is the coefficient matrix obtained when solving the inverse problem (i.e. estimating $\mathbf{M}_X$ by using $\overleftarrow{\mathbf{H}}\mathbf{M}_Y$).

3. Experiments

Training set contains about 80K English-German parallel news articles available from WMT2009 (Koehn & Haddow, 2009). We conducted experiments on 10 sentences with 10 tokens (*short*) and another 10 sentences with 20 tokens (*long*). The feature mappers are 3-spectrum counting word kernels, which consider all N-grams up to order 3 weighted by the number of tokens in the feature. Proper selection of training instances plays an important role to learn feature mappings within limited time and at expected accuracy levels. Instance selection is performed with the *tf-idf* (term frequency, inverse document frequency) weighting using the cosine similarity. We experiment with different instance selection methods: (i) per source sentence, (ii) per source sentence feature, (iii) instances' longest common matches per source sentence feature. Selection (ii) selects instances per feature (*ipf*) either proportional to the *length* of the feature, f, ($ipf = n \times \text{length}(f)$) or *dynamically* proportional to $n/\log(1 + \text{idfScore}(f)/9.0)$. Dynamic instance selection select more instances from rare features whose *idf* scores are higher. Selection (iii) uses only the longest matching parts to try to remove features coming from irrelevant tokens. We discard features that are observed less than three times from the training set.

Evaluation: We evaluate the performance of the coefficient matrix, $\mathbf{H}$, by measuring the precision, recall, and fmeasure when compared with the entries in the phrase table, *PT*, obtained by GIZA++ using the full training set. Let T contain the training indices of the target features in the *PT* that match the source sentence features, S, found in $\mathbf{H}$ whose values are greater than zero, then we define:

$$\textit{precision} = \frac{\sum_{i \in S} \sum_{j \in T} \mathbf{H}_{j,i}\ \textit{PT}_{i,j}}{\sum_{j \in T} \sum_{*>0} \mathbf{H}_{j,*}} \quad (6)$$

$$\textit{recall} = \frac{\sum_{i \in S} \sum_{j \in T} \mathbf{H}_{j,i}\ \textit{PT}_{i,j}}{\sum_{i \in S} \sum_{j \in T} \textit{PT}_{i,j}} \quad (7)$$

$$\textit{fmeasure} = \frac{2 \times \textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}} \quad (8)$$

where $\mathbf{H}_{j,i}$ stands for the coefficient for target feature j, t_j, and source feature i, s_i, $\sum_{*>0} \mathbf{H}_{j,*}$ sums over all the entries in row j that are greater than 0, and $\textit{PT}_{i,j}$ is the multiplication of the lexical translation probabilities $p(s_i|t_j)$ and $p(t_j|s_i)$ found in *PT*. We also use *top3%*, which measures the percentage of observing the top 3 scored target features in the phrase table translations, *sqLoss*, which measures the squared loss of the estimation with respect to the target sentence, and *cov.*, which measures the average coverage of the training set in representing the target sentence. Table 1 presents our evaluation of the performances of different techniques when training instances are selected dynamically with $n = 4$. The effectiveness of selection (iii) can be seen in the increase in the precision, recall, and fmeasure metrics and decrease in computation time in Table 2.

Conclusion: Our findings are listed below:

- L_1 regularization helps improve the performance. L_2 solution performs worse. QP in general perform better than FSR but takes very long time.
- Symmetrization helps in improving precision, recall, and fmeasure score. It reduces *sqLoss* in FSR and sometimes in QP solutions.
- *Coverage* and *top3%* increase as we select more instances, but this decreases precision and *sqLoss* due to adding more noise.
- QP quickly becomes infeasible due to increased computation time when N_X and N_Y increase. Selection (iii) helps us increase precision, recall, and fmeasure without increasing the *sqLoss* too much.

References

Hastie, T., Taylor, J., Tibshirani, R., & Walther, G. (2006). Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, *1*.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference and prediction*. Springer-Verlag.

Koehn, P., & Haddow, B. (2009). Edinburgh's submission to all tracks of the WMT 2009 shared task with reordering and speed improvements to Moses. *Proceedings of the Fourth Workshop on Statistical Machine Translation* (pp. 160–164). Athens, Greece: Association for Computational Linguistics.

Mørup, M., & Clemmensen, L. H. (2007). Multiplicative updates for the LASSO. *MLSP 2007*.

Och, F. J., & Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, *29*, 19–51.

Tibshirani, R. J. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, *58*, 267–288.

Table 1. Numbers represent averages. Time is in seconds. S suffix is for symmetrized techniques.
Top: Performances of different techniques when training instances are selected dynamically with $n = 4$.
Bottom: Selection (i) results using *long* set of sentences. 50 and 100 instances per sentence are selected.

n=4, dynamic		prec.	recall	fmeas.	top3%	sqLoss	time
short	**L2**	0.007	0.038	0.011	0.206	39.521	0.204
	L2S	0.006	0.039	0.010	0.223	69.514	0.674
	QP	0.061	0.062	0.061	0.377	26.208	335.121
	QPS	0.162	0.072	0.098	0.377	30.281	352.124
	FSR	0.038	0.070	0.049	0.335	62.973	24.490
	FSRS	0.193	0.076	0.106	0.318	32.456	23.674
long	**L2**	0.0	0.034	0.009	0.297	69.240	0.960
	L2S	0.0	0.037	0.008	0.276	129.042	1.932
	QP	0.066	0.081	0.072	0.419	51.146	1105.915
	QPS	0.189	0.095	0.125	0.419	51.172	1058.366
	FSR	0.056	0.094	0.069	0.362	99.644	73.107
	FSRS	0.239	0.102	0.141	0.353	53.125	79.008

n, selection (i)		prec.	recall	fmeas.	top3%	sqLoss	time
50	**L2**	0.010	0.033	0.015	0.172	43.242	0.067
	L2S	0.009	0.036	0.014	0.186	50.194	0.137
	QP	0.091	0.056	0.068	0.350	37.885	31.119
	QPS	0.255	0.054	0.087	0.335	31.685	30.879
	FSR	0.051	0.085	0.063	0.285	79.713	3.906
	FSRS	0.321	0.085	0.131	0.275	33.421	3.190
100	**L2**	0.007	0.035	0.011	0.251	55.511	0.363
	L2S	0.006	0.038	0.010	0.254	74.590	0.815
	QP	0.089	0.071	0.079	0.426	43.309	416.296
	QPS	0.257	0.082	0.123	0.417	39.404	423.335
	FSR	0.053	0.085	0.065	0.377	84.525	31.043
	FSRS	0.294	0.091	0.137	0.358	43.266	27.953

Table 2. Numbers represent averages taken over the *long* set of sentences. Time is in seconds.
Top: QP performance when training instances are selected *dynamically* and with proportion to *length*.
Bottom: QP performance when training instances are selected *dynamically* with n and only matching parts are used as training sentences.

QP	n	ipf	cov.	prec.	recall	fmeas.	top3%	sqLoss	time
dynamic	**1**	1.616	0.324	0.083	0.074	0.077	0.330	42.534	113.205
	2	1.663	0.328	0.081	0.073	0.076	0.342	44.195	143.112
	3	2.111	0.360	0.076	0.080	0.077	0.359	49.779	508.252
	4	2.704	0.378	0.066	0.081	0.072	0.419	51.146	1105.915
length	**1**	1.616	0.324	0.083	0.074	0.077	0.330	42.534	114.167
	2	1.954	0.347	0.074	0.075	0.073	0.359	48.205	411.712
	3	2.439	0.365	0.066	0.079	0.071	0.394	49.872	1132.119
	4	3.113	0.385	0.057	0.079	0.066	0.435	51.777	2508.383

n	m	N_X	N_Y	ipf	cov.	prec.	recall	fmeas.	top3%	sqLoss	time
2	81.000	385.700	427.500	1.737	0.243	0.095	0.072	0.081	0.222	41.411	29.661
3	103.500	428.900	474.000	2.214	0.250	0.106	0.084	0.093	0.250	43.289	25.440
4	133.600	433.700	479.900	2.849	0.254	0.113	0.091	0.100	0.263	43.243	52.357
5	162.000	441.600	490.200	3.450	0.256	0.120	0.091	0.102	0.279	43.399	52.019
6	190.300	441.600	494.100	4.048	0.262	0.122	0.096	0.105	0.283	44.083	92.475
7	216.500	442.000	495.800	4.605	0.264	0.129	0.101	0.112	0.286	44.110	89.570
8	242.400	442.300	497.600	5.148	0.270	0.131	0.101	0.112	0.287	44.310	90.859
9	266.800	442.700	498.700	5.662	0.270	0.134	0.100	0.113	0.296	44.249	155.055
10	290.800	443.000	500.100	6.165	0.273	0.136	0.099	0.113	0.298	44.343	175.650

Collaborative Haptic Negotiation and Role Exchange in Multimodal Virtual Environments

S. Ozgur Oguz SOGUZ@KU.EDU.TR
Ayse Kucukyilmaz AKUCUKYILMAZ@KU.EDU.TR
Tevfik Metin Sezgin MTSEZGIN@KU.EDU.TR
Cagatay Basdogan CBASDOGAN@KU.EDU.TR
College of Engineering, Koc University

1. Introduction

This work is a preliminary study to investigate the benefits of haptic guidance with collaborative role exchange mechanisms over simple guidance methods in dyadic interaction. Our system employs a novel negotiation mechanism that realizes role exchange using a three-state finite state machine. This scheme creates a favorable communication while providing acceptable task performance. Our initial findings suggest that this scheme introduces a tradeoff between the accuracy in task performance and the effort of the user.

We concentrated our studies in tasks where human and computer connect through the haptic channel. Haptics can improve task performance by providing the user with appropriate feedback (Morris et al., 2007; Feygin et al., 2002) and create a sense of acting together (Basdogan et al., 2000; Sallnäs et al., 2000). Few groups studied role definitions for collaboration, most of which implement a leader-follower scheme, where the human partner takes on the leading role (Khatib, 1999; Kosuge et al., 1993). These improve performance yet fail to create a sense of collaboration. Reed and Peshkin (Reed & Peshkin, 2008) examine specialization of partners in dyadic interaction. However applying the observed specialization scheme to the computer fails, probably due to a lack of careful examination of the nature of specialization. Stefanov et al. (Stefanov et al., 2009) propose executor and conductor roles for haptic interaction. This work presents important observations regarding communication for role exchange, yet is not applied to human-computer interaction. Evrard and Kheddar (Evrard & Kheddar, 2009) similarly define leader and follower roles in a symmetric dyadic task. Despite the simplicity of their experimental design, they introduce a novel method for modeling interaction behavior. However this method involves neither a user-centric nor a dynamic negotiation mechanism to handle interaction.

2. Haptic Board Game

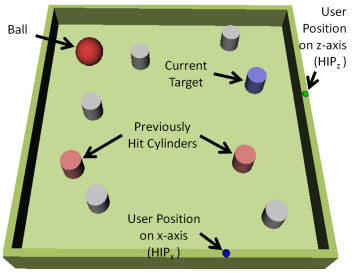


Figure 1. A screenshot of the Haptic Board Game.

The goal of the haptic board game is to hit targets on a flat board with a ball in a specific order with the help of a PHANToM device. Visually, the board is tilted about the x and z axes as a result of the movement of the ball. Figure 1 shows a snapshot of the game where the current and previous targets are clearly visible.

To create a collaborative system, we came up with a novel haptic negotiation scheme as in Figure 2. In this model, the system is controlled by three virtual massless particles: haptic, controller, and negotiated interface points (respectively HIP, CIP, and NIP). HIP, CIP and the ball are interconnected at NIP, through which the ball is directly moved.

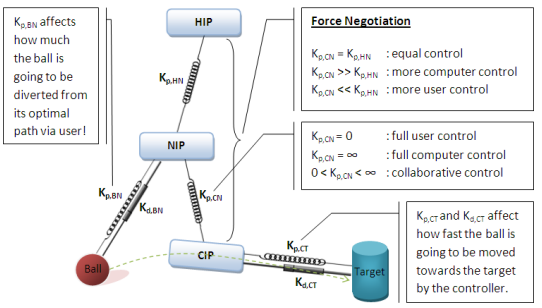


Figure 2. The physical model for the haptic board game. K_p and K_d values in the figure represent the spring and damper coefficients. In no guidance condition CIP and NIP coincide at all times to prevent computer intervention.

In order to investigate the effect of different collaboration mechanisms, we implemented three conditions to be tested with the Haptic Board Game:

No Guidance: The user only feels the resistive forces due to the rotation of the board, but the computer does not provide any haptic guidance. In this condition, CIP and NIP coincide to prevent any computer intervention.

Both Axes Guidance: Both the user and the computer affect the system equally at all times. Using PD (Proportional-Derivative) control algorithm, the controller adjusts the orientation of the board such that the ball automatically moves towards the target

Role Exchange: The system allows dynamic haptic negotiation between the user and the computer by inferring user's intentions. Based on the user's force profile, personalized threshold values are calculated and depending on these values, the computer decides on how the parties share control. The degree of provided guidance is adjusted dynamically with a role exchange policy, implemented with a three-state finite state machine (see Figure 3).

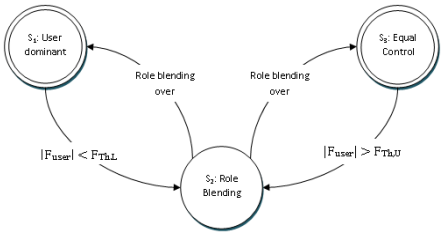


Figure 3. The state diagram defining the role exchange policy. F_{user}: the force applied by the user. F_{ThL}, F_{ThU}: the lower and upper threshold values for initiating the state transitions. S_1: user is the dominant actor, S_3: both computer and user have equal degree of control, S_2: role blending is in process to change the controller's role gradually. Initially the system is in state S_1, where user is the dominant actor of the game while the controller only gently assists him.

3. Experiment

10 subjects (5 female, and 5 male) participated in our study. In order to eliminate learning effects on successive trials, the order of experimental conditions was mixed, with at least three days between two successive experiments. In the beginning of the experiments, we used certain training applications, to familiarize the subjects with the haptic device. We did not inform the subjects about the nature or the existence of different conditions we were testing. All experiments were conducted under the same physical settings. A single experiment took about half an hour, and in each experiment the users played with either no guidance, both axes guidance, or role exchange conditions. In the no guidance and both axes guidance conditions, each subject played the game 15 times for a single experiment. Each game consisted of hitting eight targets in a specific order, with the ball. At the end of each game another game is automatically instantiated without interrupting the system's simulation. To avoid fatigue, subjects were given a short break after the 5^{th} and the 10^{th} games . For the role exchange condition, the users played an extra game under no guidance condition at the beginning of each block of 5 games in order for the system to determine the force thresholds necessary for the role exchange process.

4. Results

4.1 Subjective Evaluation Results

After each experiment, the users were given a questionnaire, designed by the technique used by Slater et al. (Slater et al., 2000). The questions are asked to measure the self-perception of users' performance and the collaborative aspects of the system, as well as the degree to which the users felt they or the computer were in control.

Subjects believed that they performed better on both axes guidance and role exchange compared to the no guidance condition. The differences were statistically significant for both axes guidance and role exchange when compared to the no guidance condition ($p < 0.005$ and $p < 0.02$, respectively). No significant difference is observed between both axes guidance and role exchange conditions.

For the level of perceived collaboration, subjects had a higher sense of collaboration for the role exchange and both axes guidance conditions compared to the no guidance condition ($p < 0.01$). Again, no significant difference is observed between both axes guidance and role exchange conditions.

Subjects' and computer's levels of control were similar in all three conditions. However, in role exchange condition, even though the subjects perceived reduced control over the game, they had a stronger sense of computer participation.

4.2 Quantitative Measurements

We quantified user performance and the utility of providing haptic guidance by task completion time, the deviation of the ball from the ideal path, integral of time and absolute magnitude of error (ITAE) (Dorf & Bishop, 2000), and work done by the user due to the spring located between NIP and HIP. Since the forces accumulated in the system are sent indirectly to the user through this spring only, we assume that this force is the force felt by the user.

According to paired t-test results ($p = 0.05$), all three conditions display significant difference from each other regarding task performance (see Figure 4). We observed that best performance is achieved in both axes guidance condition. In no guidance condition, the performance is the worst, while role exchange falls in between the two.

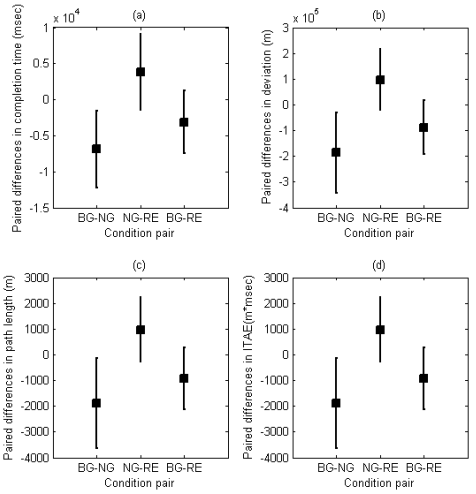


Figure 4. Means and standard deviations of paired differences of (a) completion times, (b) path deviations, (c) path lengths, and (d) ITAEs per condition (NG: no guidance, BG: guidance without negotiation, RE: guidance with negotiation and role exchange)

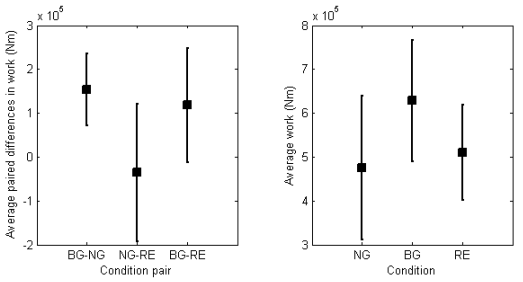


Figure 5. Means and standard deviations of energy on the spring between NIP and HIP per condition and paired differences of energy (NG: no guidance, BG: guidance without negotiation, RE: guidance with negotiation and role exchange)

Figure 5 illustrates the average work done by the user. No significant difference is observed between no guidance and role exchange conditions, whereas both are statistically different from both axes guidance condition.

As a result, both axes guidance has higher energy requirements, while the completion time and spatial error of no guidance is inferior. It is the role exchange mechanism that allows us to trade off accuracy for energy without causing user dissatisfaction.

5. Conclusion

This paper presents the novel haptic negotiation and role exchange model we developed for a collaborative task in a highly dynamic environment. This model realizes dynamic negotiation between a human user and a computer and implements role exchange using a three state finite state machine that allowed us to achieve seamless interaction. With the proposed role exchange mechanism, the users are presented with an option to choose and optimize between precision and energy, implying a tradeoff between the accuracy in task performance and the effort of the user.

References

Basdogan, C., Ho, C.-H., Srinivasan, M. A., & Slater, M. (2000). An experimental study on the role of touch in shared virtual environments. *ACM Trans. Comput.-Hum. Interact., 7*, 443–460.

Dorf, R. C., & Bishop, R. H. (2000). *Modern control systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Evrard, P., & Kheddar, A. (2009). Homotopy switching model for dyad haptic interaction in physical collaborative tasks. *WHC '09: Proceedings of the World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (pp. 45–50).

Feygin, D., Keehner, M., & Tendick, F. (2002). Haptic guidance: Experimental evaluation of a haptic training method for a perceptual motor skill. *HAPTICS '02: Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (p. 40).

Khatib, O. (1999). Mobile manipulation: The robotic assistant. *Robotics and Autonomous Systems, 26*, 175–183.

Kosuge, K., Yoshida, H., & Fukuda, T. (1993). Dynamic control for robot-human collaboration. *Robot and Human Communication, 1993. Proceedings., 2nd IEEE International Workshop on* (pp. 398–401).

Morris, D., Tan, H., Barbagli, F., Chang, T., & Salisbury, K. (2007). Haptic feedback enhances force skill learning. *WHC '07: Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (pp. 21–26).

Reed, K. B., & Peshkin, M. A. (2008). Physical collaboration of human-human and human-robot teams. *IEEE Trans. Haptics, 1*, 108–120.

Sallnäs, E.-L., Rassmus-Gröhn, K., & Sjöström, C. (2000). Supporting presence in collaborative environments by haptic force feedback. *ACM Trans. Comput.-Hum. Interact., 7*, 461–476.

Slater, M., Sadagic, A., Usoh, M., & Schroeder, R. (2000). Small-group behavior in a virtual and real environment: A comparative study. *Presence: Teleoper. Virtual Environ., 9*, 37–51.

Stefanov, N., Peer, A., & Buss, M. (2009). Role determination in human-human interaction. *WHC '09: Proceedings of the World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (pp. 51–56).

Rigid Motion Correction in IVUS Sequences

Gozde Gul Isguder ISGUDER@SU.SABANCIUNIV.EDU
Gozde Unal GOZDEUNAL@SABANCIUNIV.EDU
Sabanci University, Tuzla/ Istanbul

1. Introduction

Intravascular Ultrasound(IVUS) is an imaging technology which provides high resolution images of internal vascular structures. IVUS is a special tool that is widely used in plaque detection and measurement of vessel wall stiffness. These properties of IVUS provide assistance for diagnosis of cardiac diseases. Unfortunately during the pullback due to the catheter motion inside the arteries in-vivo and heart expansion, longitudinal geometry of the vessel wall seems pulsatile and jagged which makes the analysis and diagnosis of the cardiac diseases harder. In this paper, we present two motion correction methods based on spectral analysis and intensity based registration applied across IVUS cross-section images for 3D reconstruction and visualization of reconstructed boundaries. Furthermore we compare the results of these two methods on in-vivo sequences. Currently there are no accepted evaluation techniques for our problem, but from our 3D visualization it can clearly be seen that the motion is strongly reduced.

2. Related Work

There have not been many approaches for the motion problem. One of the main approaches was addressing only the cardiac cycle movement(heart motion). In-vivo studies a special unit, named ECG unit, can be used to capture only the frames synchronized with the cardiac pulse, thus reducing the motion caused by the heart motion. Since it longers the image acquisition procedure, automatic image-gating methods (O'Malley et al., 2008) that can be used after image acquisition, were developed. Although those methods reduce the motion, they cause a big loss of information by ignoring the frames between two cardiac cycles.

There have been also non-rigid methods and vessel geometric appearance based approaches mainly focusing on morphology of the vessel. Although the motivation behind the work is nice, those methods were computationally inefficient.

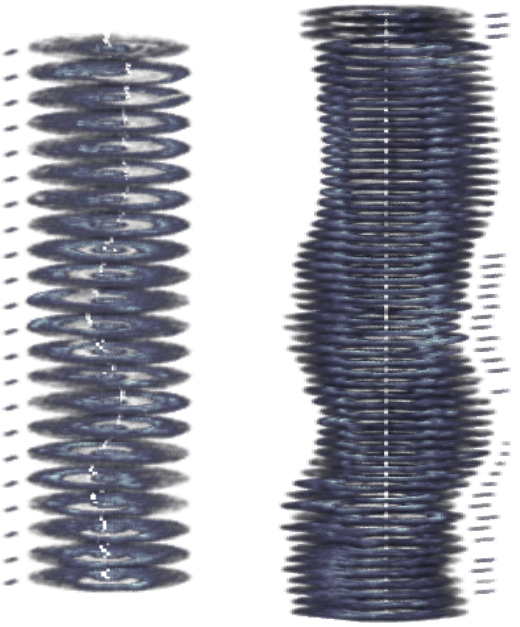


Figure 1. Left:Original stack of IVUS Frames; Right: Registered stack of IVUS Frames

3. Method

During a pullback, as the catheter moves inside the arteries in-vivo, it inevitably translates and rotates and causes a misalignment between the two recorded consecutive cross-section images. Moreover, the catheter may go back and forth as the heart dilates and compresses. The result is a pulsatile and jagged longitudinal geometry of the vessel wall (see Fig 1). For an accurate 3D vessel geometry and longitudinal analysis, these catheter effects should be corrected.

3.1 Preprocessing

In this section two different methods that address rigid motion correction are presented. First method is the spectral based method. For this method, there's a need for previ-

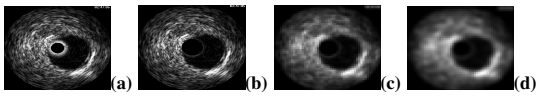


Figure 2. Preprocessing stage for spectral and intensity based registration (a)Original IVUS Frame; (b)After removing the catheter artifact; (c) After downsampling the IVUS Frame; (d) After smoothing the downsampled IVUS Frame

ously segmented lumen borders to find the lumen centers by calculating the center of mass. We use our own segmentation tool that was published in (Unal et al., 2006) for lumen segmentation purposes. Due to the algorithm's efficiency and robustness to noise, no other preprocessing stage is needed.

Second method is an intensity based method that uses mutual information between two consecutive frames. This method is tied strongly to the intensity values and hence not robust to noise. Due to the fact that the optimizer may take a long time if no appropriate initial values are provided, first we downsample the images to reduce the execution time. In order to reduce the noise in the downsampled images, we then smooth the images with a Gaussian filter with a 3x3 kernel with σ =1.

In all IVUS frames, there are brighter areas in a small radius circle, that was caused by the catheter. This artifact is called catheter artifact and must be removed for the sake of effectiveness of both of the algorithms. In order to remove the artifact a horizontal scanning is made through the images and the last line that has a brightness above the mean was chosen to be the artifact line, and the rows above this line are assigned to zero. The preprocessing stage can be seen in Fig. 2.

3.2 Spectral Based Method

In this paper, we don't address the heart motion artifact which causes a back-forth motion and causes us to see some cross-section frames multiple times; but we only address two types of cross-sectional catheter motion: translation and rotation. As the catheter moves inside the vessel, due to the varying vessel curvature, both types of motion occur. We estimate these transformation parameters seperately. First, we account for the translation by aligning the IVUS cross-section images w.r.t. their lumen area's center of mass that was extracted by our segmentation tool.

Secondly, to address the rotation of the catheter during pullback, we estimated rotation between subsequent frames via a spectral correlation analysis method inspired by (Hernandez et al., 2006), where the translation on a rectangular image, which corresponds to a rotation on the display image, is calculated as follows: Let I be the rectangular image and I_t is the translated version of I with a 2D translation

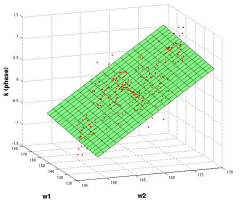


Figure 3. fitted plane to phase ρ

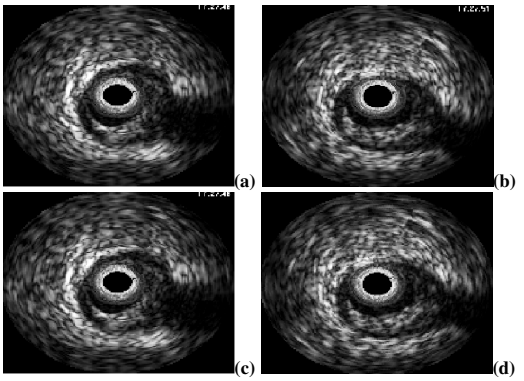


Figure 4. Rotation estimation: Middle column: source image; Right top: reference image; Right bottom: rotated source image with the estimated rotation=5.21 degrees;

$t = (t_1, t_2)$. The ratio between the Fourier transforms of the image I and I_t: $k(w_1, w_2) = F(I_t)/F(I).e^{-j<w,t>}$, can be used to extract the phase: $\rho(w) = \langle w, t\rangle = w_1 * t_1 + w_2 * t_2$, where $w = (w_1, w_2)$ denotes the 2D frequency vector. The idea then is to estimate the amount of shift using the phase defined over the 2D frequency space by fitting a plane to the ρ function that can be seen in Fig. 3: $Aw_1 + Bw_2 + C\rho = D$. Here a translation between the two images in either the horizontal or the vertical direction can be detected over the plane aligned with one of the frequency axis w_1 or w_2. Practically, a least-squares estimator is used via a singular value decomposition. In the IVUS rectangular images, the calculated slope B of the plane represents the estimated rotation value in radians (whereas D=0). An example is shown for rotation estimation between two subsequent IVUS frames in Fig. 4 .

3.3 Intensity Based Method

This is a very well known method that uses image intensity information to find a transformation T, between the images. By means of its simplicity and flexibility, this method was previously adapted to many different vision problems and used with a wide variety of image modalities.

Let's define X and Y as two 2D images that we want to register. Start by initializing a T (holds for transformation) on one of the images.

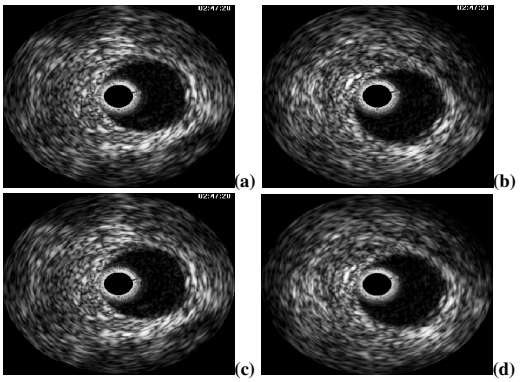


Figure 5. Rotation estimation: Left column: source image; Right top: image to be registered; Right bottom: transformed image with the estimated rotation=12.14 degrees, dx=0.80 , dy=0.75 in pixels ;

Step 1) Transform Y with T to get $T(Y)$.
Step 2) Compare X and $T(Y)$ using the full image content.
Step 3) Refine T and go to step 1 until the convergence criteria is reached.

The most important part of the problem is defining a similarity measure for comparison. Current similarity measures include classical statistic methods like SSD,NCC and information theoretic measures that treats the data as a random variable and uses probability of intensity values instead of the values itself. Mutual information is an example of information theoretic approach that uses joint histograms. For our purposes we used Mutual Information as the similarity measure.

The second problem is choosing an optimization method to minimize/maximize the similarity measure thus refine T. Current optimization methods include downhill simplex, best neighbor and newton like methods. Best neighbor and newton like methods are more likely to stuck at local minima, thus we choose downhill simplex for our optimization problem.

The last important issue is choosing an interpolation technique. Interpolation is crucial to minimize distortions/errors when applying transform. For that reason bilinear interpolation technique, which uses also the neighbor pixel information was used.

A registered example can be seen in Fig. 5 estimated parameters are translation in x direction=0.80796 ,translation in y direction=0.75619 in pixels and the rotation angle=12.1409 degrees.

4. Results and Discussions

All IVUS images were performed by Volcano Eagle Eye IVUS machine using motorized pullback with a speed of 0.5 mm per sec. 100 microgram nitro were given intracoronary before image acquisition. The two methods were applied on one pullback and the results are shown in Fig. 1.

In the result shown in Fig. 1, the white lines present the lumen center(can be referred as vessel center). For the sake of clearness , we've shown the original pullback with less images, while we show the registered pullback more densely. As can be seen from the results, in the registered pullback the vessels' centers are aligned and in contrast to the original pullback, the lumen ,which have a brighter pattern, is clearly aligned. In conclusion we can claim that we extracted the real 3D geometry of the vessel.

Although the spectral based method requires a segmentation process beforehand for the lumen center estimation, it is more efficient and robust to noise. Intensity based methods don't require any primary knowledge about the lumen contours but computationally less efficient and redundant to intensities thus easily effected by the noise.In our experiments, we observed that spectral registration method gives far better results than intensity based methods.

5. Future Work

In our future work, we will work on a method to extract real 3D vessel geometry, that is not strongly tied to the segmentation results or the intensity values, and more efficient. Furthermore we will test our methods on more pullbacks, and try to define a novel evaluation method for this area.

References

Unal, G., Bucher, S., Carlier, S.,Slabaugh, G.,Fang, T., Tanaka, K.(2006). Shape-driven Segmentation of the Arterial Wall in Intravascular Ultrasound Image *Proceedings of the MICCAI: The First International Workshop on Computer Vision for Intravascular and Intracardiac Imaging (CVII)* (pp. 51–58).

Hernandez, A.,Radeva, P., Tovar, A., Gil, D.(2006). Vessel structures alignment by spectral analysis of IVUS sequences. *Proceedings of Computer Vision for Intravascular and Intracardiac Imaging (CVII)* Copenhagen,Denmark.

O'Malley, S.M. Granada, J.F. Carlier, S. Naghavi, M.Kakadiaris, I.A.(2008). Image-Based Gating of Intravascular Ultrasound Pullback Sequences. *IEEE Transactions on Information Technology in Biomedicine* Vol.12, No.3, May 2008,(pp. 299–306).

Genome Rearrangement: A Planning Approach

Tansel Uras TANSELURAS@SABANCIUNIV.EDU
Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey

1. Introduction

In biology, evolutionary trees (or phylogenies) can be reconstructed from the comparison of genomes of species (Sankoff & Blanchette, 1998). One metric of evolutionary distance for this purpose is the number of rearrangement events such as transpositions and inversions to convert one genome to the other where a smaller number of such events implies a closer lineage. A rearrangement event is a genome-wide mutation that changes the order and/or orientations of genes (and sometimes their existence) in a genome. Finding the minimum number of these rearrangement events between genomes is called the genome rearrangement problem and it is conjectured to be NP-hard (Bylander, 1994).

We consider the genome rearrangement problem as a planning problem as in (Erdem & Tillier, 2005): one of the genomes is represented as the initial state and the other one as the goal state; the planner is prompted to find a sequence of at most k actions (rearrangement events) that leads the initial state to the goal state. As in (Erdem & Tillier, 2005), we describe the genome rearrangement problem in ADL (Pednault, 1989), and use TLPLAN (Bacchus & Kabanza, 2000) to compute solutions. Our formulation of the genome rearrangement problem differs from that of (Erdem & Tillier, 2005) in the following ways. First of all, it extends the descriptions of genomes to be able to handle duplicate genes—genes that occur multiple times in a single genome. Accordingly, it not only extends the descriptions of transpositions, inversions, inverted transpositions (transversions) but also introduces new operators for insertions and deletions. The temporal control information is described as preconditions of events. As observed in (Rintanen, 2000; Gabaldon, 2003), such a modification improves the computational efficiency. Also, the goal-check is done in a more computationally-efficient way by means of a biologically motivated measure, called the *breakpoint distance*, which does not require us to check the whole gene orders of the genomes.

The main contribution of our work are as follows:

- We have introduced a computational method that can solve the genome rearrangement problem with duplicates and unequal gene content. Although the genomes of many species (in particular, the chloroplast genomes) contain duplicate genes, no existing genome rearrangement software (e.g., GRIMM (Tesler, 2002), GRAPPA (Moret et al., 2001), DERANGE 2 (Blanchette et al., 1996)) can handle them.

- One method to handle duplications without loss of information is to treat them as different genes and solve the problem for each possible relabeling of these genes (Cui et al., 2006); however, there are exponentially many possible relabelings in the number of occurrences of the duplicate genes. We have introduced a new method to handle duplicates without such enumeration of relabelings, by identifying the duplicates upfront and introducing a 0-cost auxiliary action.

- We have illustrated the applicability and the effectiveness of our planning-based approach to genome rearrangement on three sets of real data: mitochondrial genomes of *Metazoa* (animals with a nervous system, and muscles) (Blanchette et al., 1999), chloroplast genomes of *Campanulaceae* (flowering plants) (Cosner et al., 2000), and chloroplast genomes of various land plants and green algae (Cui et al., 2006). Our results conform with the most recent and widely accepted results.

2. Genome Rearrangement Problem

The *genome* of a single-chromosome organism can be represented by circular configurations of numbers $1, \ldots, n$, with a sign $+$ or $-$ assigned to each of them. For instance, Figure 1(a) shows a genome for $n = 5$. Numbers $\pm 1, \ldots, \pm n$ will be called *labels*. Intuitively, a label corresponds to a gene, and its sign corresponds to the orientation of the gene. By $(l_1, \ldots, l_n)$ we denote the genome formed by the labels $l_1, \ldots, l_n$ ordered clockwise. For instance, each of the expressions $(1, 2, -5, -4, -3), (2, -5, -4, -3, 1), \ldots$ denotes the genome in Figure 1(a).

About genomes g, g' we say that g' is a *transposition* of g (or *can be obtained from g by a transposition*) if, for some

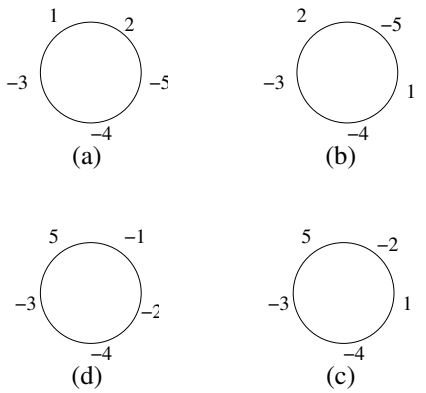


Figure 1. (a) A genome; (b) a transposition of (a); (c) an inversion of (b); (d) a transversion of (c).

labels $l_1, \ldots, l_n$ and numbers k, m $(0 < k, m \leq n)$,

$$g = (l_1, \ldots, l_n)$$
$$g' = (l_k, \ldots, l_m, l_1, \ldots, l_{k-1}, l_{m+1}, \ldots, l_n).$$

For instance, the genome in Figure 1(b) is a transposition of the genome in Figure 1(a).

Similarly, about genomes g, g', we say that g' can be obtained from g by a *deletion* (or g can be obtained from g' by an *insertion*) if, for some labels $l_1, \ldots, l_n$ and a number m $(0 < m \leq n)$,

$$g = (l_1, \ldots, l_n)$$
$$g' = (l_1, \ldots, l_{m-1}, l_{m+1}, \ldots, l_n).$$

Other events, inversions and transversions, can be defined as in (Erdem & Tillier, 2005).

We say that there is a *breakpoint* between two genomes if one of the genomes includes the pair l, l' and the other genome includes neither the pair l, l' nor the pair $-l', -l$. For instance, there are 3 breakpoints between $(1, 2, 3, 4, 5)$ and $(1, 2, -5, -4, 3)$. The number of breakpoints between two genomes is called their *breakpoint distance*.

The *genome rearrangement problem* can be defined as follows: given two genomes g and g', and a positive integer k, decide whether g' can be obtained from g by at most k successive events. We view the genome rearrangement problem as a planning problem:

> given two genomes g and g', and a nonnegative integer k, find a sequence of at most k events that reduces the number of breakpoints between g and g' to 0.

Note that this planning problem is different from the one described in (Erdem & Tillier, 2005) in that both genomes are specified in the initial world, and that the goal is specified in terms of the number of breakpoints.

3. Representing the Planning Problem

We represent a genome by specifying the clockwise order of its labels, and by identifying which genes are duplicates. Both genomes are described in the initial state; for that, we introduce two fluents to describe their gene orders. To describe the goal, we introduce a functional fluent to denote the number of breakpoints: initially, it is counted; after that, at each step, it is decreased by the application of a rearrangement event. We introduce five actions to describe transpositions, inversions, transversions, insertions and deletions, and represent them as ADL-style operators in the language of TLPLAN. The definitions of these operators extend the definitions in (Erdem & Tillier, 2005) with respect to the planning problem above to handle duplicates.

4. Experimental Results

We have experimented with three sets of data using TLPLAN: *Metazoan* mitochondrial genomes (Blanchette et al., 1999), *Campanulaceae* chloroplast genomes (Cosner et al., 2000), and chloroplast genomes of various land plants and green algae (Cui et al., 2006). Only in the last data set, genomes are of unequal content with duplicate genes.

To analyze the accuracy of our approach, for each data set, first we have computed a small number of events for each pair of genomes and constructed a distance matrix, and then we have constructed a phylogeny using the distance matrix program NEIGHBOR (Felsenstein, 2009).

In all experiments, TLPLAN is run with the depth-best-first search strategy. The cost of each action is 1 (except the 0-cost action of swapping duplicates); so the goal is to find a plan with a small cost (rather than a shortest plan). The priorities of insertions, deletions, and swaps are much higher than the other events.

Mitochondrial genomes of *Metazoa*: Each one of these 11 genomes consists of 36 genes. The priorities of transpositions, inversions, transversions are specified as 2, 1, 1 respectively. All 45 plans (each with 1–26 events) are computed in less than 3 minutes. The phylogeny constructed by NEIGHBOR groups chordates and echinoderms together, arthropods, molluscs and annelids together; nematodes are a sister to these two groupings. These results conform with the results of (Nielsen, 2001) based on morphological data. Groupings of chordates and echinoderms, and molluscs and annelids also conform with the most widely accepted view of Metazoan Systematics and Tree of Life, based on the analysis of molecular data (18S rRNA sequences).

Chloroplast genomes of *Campanulaceae*: We consider 13 genomes, each with 105 genes. The priorities of transpositions, inversions, transversions are specified as 2, 3,

4 respectively (since inversions often occur in chloroplast genomes). All 66 plans (each with 1–12 events) are computed in less than 1 minute. According to the phylogeny constructed by NEIGHBOR, the groupings are identical to the ones in the consensus tree presented in Figure 4 of (Cosner et al., 2000). The major division between the grouping of *Codonopsis*, *Cyananthus* and *Platycodon*, and the others conform with the most recent results (Cosner et al., 2004) based on the sequence analysis; also this division corresponds to the distribution of pollen morphology characteristics, unlike the previous results.

Chloroplast genomes of land plants and green algae: These 7 genomes share 85 genes; each genome is of length 87–97. The priorities of transpositions, inversions, transversions are specified as 2, 3, 4 respectively. All 21 plans (each with 6–47 events) are computed in less than an hour. (The computation of a phylogeny for these species takes almost 25 days in (Cui et al., 2006).) The phylogeny constructed by NEIGHBOR groups *Nicotiana* and *Marchantia* with *Chaetosphaeridium*, thus grouping the land plants and charophyte algae; it also groups *Chlorella* and *Chlamydomonas* with *Nephroselmis*, thus grouping the chlorophyte algae; *Mesostigma* is an outlier. These results conform with the biological evidence based on the analysis of 50 concatenated proteins (Cui et al., 2006).

5. Conclusion

We have introduced a new computational method, based on AI planning, to solve genome rearrangement problems with duplicate genes, involving transpositions, inversions, inverted transpositions, insertions, and deletions. No existing genome rearrangement software can handle such problems. We have shown the applicability and the effectiveness of our planning-based method on real data sets; we have observed that the results are similar to those widely accepted.

Acknowledgments

Thanks to my instructor Esra Erdem for her guidance and contributions.

References

Bacchus, F., & Kabanza, F. (2000). Using temporal logic to express search control knowledge for planning. *Artificial Intelligence*, *116*, 123–191.

Blanchette, M., Kunisawa, T., & Sankoff, D. (1996). Parametric genome rearrangement. *Gene-Combis*, *172*, 11–17.

Blanchette, M., Kunisawa, T., & Sankoff, D. (1999). Gene order breakpoint evidence in animal mitochondrial phylogeny. *Journal of Molecular Evolution*, *49*, 193–203.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, *69*, 165–204.

Cosner, M., Jansen, R., Moret, B., Raubeson, L., Wang, L., Warnow, T., & Wyman, S. (2000). An empirical comparison of phylogenetic methods on chloroplast gene order data in *Campanulaceae*. In D. Sankoff and J. Nadeau (Eds.), *Comparative genomics*, 99–122. Kluwer.

Cosner, M., Raubeson, L., & Jansen, R. (2004). Chloroplast DNA rearrangements in *Campanulaceae*: phylogenetic utility of highly rearranged genomes. *BMC Evolutionary Biology*, *4*.

Cui, L., Leebens-Mack, J., Wang, L., Tang, J., Rymarquis, L., Stern, D., & dePamphilis, C. (2006). Adaptive evolution of chloroplast genome structure inferred using a parametric bootstrap approach. *BMC Evolutionary Biology*, *6*, 13.

Erdem, E., & Tillier, E. (2005). Genome rearrangement and planning. *Proc. of AAAI* (pp. 1139–1144).

Felsenstein, J. (2009). PHYLIP (phylogeny inference package) version 3.6. Distributed by the author.

Gabaldon, A. (2003). Compiling control knowledge into preconditions for planning in the situation calculus. *Proc. of IJCAI* (pp. 1061–1066).

Moret, B., Wyman, S., Bader, D., Warnow, T., & Yan, M. (2001). A new implementation and detailed study of breakpoint analysis. *Proc. of PSB'01* (pp. 583–594).

Nielsen, C. (2001). *Animal evolution: Interrelationships of the living phyla*. Oxford University Press.

Pednault, E. (1989). ADL: Exploring the middle ground between STRIPS and the situation calculus. *Proc. of KR'89* (pp. 324–332).

Rintanen, J. (2000). Incorporation of temporal logic control into plan operators. *Proc. of ECAI* (pp. 526–530).

Sankoff, D., & Blanchette, M. (1998). Multiple genome rearrangement and breakpoint phylogeny. *Journal of Computational Biology*, *5*, 555–570.

Tesler, G. (2002). GRIMM: genome rearrangements web server. *Bioinformatics*, *18*, 492–493.

Prime Number Generation: Writing a parallel program on a multi core machine that implements Miller-Rabin Primality Testing

Emre Kaplan EMREKAPLAN@SABANCIUNIV.EDU
Barış Altop ALTOP@SABANCIUNIV.EDU
Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey

1. Introduction

Multi-core machines bring the parallel computing to agenda in today's computing. Parallel computing algorithms were designed to solve problems using as much processing units as possible and running independent computations concurrently. Various applications need big prime numbers because of the underlying cryptographic operations. For instance, in order to use RSA infrastructure we will need big prime numbers like 512, 1024 even 2048 bits long to operate. Hence we need mechanisms to produce and verify such big prime numbers. In this project, we implemented and tested the algorithms (see Section 2 for details) listed below:

1. Miller-Rabin Probabilistic Primality Test (mra,)
2. Solovay-Strassen Probabilistic Primality Test (sol,)
3. Miller-Rabin Deterministic Primality Test (mrd,)
4. Fermat Probabilistic Primality Test (fer,)

2. Algorithms

2.1 Brief Discussion on Algorithms

All those algorithms take the big number say *n* as the input and tries to conclude if *n* is a prime or composite number. Probabilistic algorithms require additional input parameter, k as the number of repetitions of the primality test. Probabilistic algorithms conclude "Composite" with 100% assurance while can conclude "Prime" with a probability depends on the number of repetitions. That is, probability to conclude "Prime" for a composite number n is bounded by ϵ which is the error probability of the algorithm. In Miller Rabin and Solovay-Strassen algorithms as k gets larger, the probability of concluding "Prime" when *n* is composite (i.e ϵ) decreases. In Miller Rabin Algorithm, $\epsilon < 4^{-k}$ while in Solovay-Strassen Algorithm $\epsilon < 2^{-k}$. Hence the success of probabilistic algorithms depends on the k value.

In the case of probabilistic algorithms, test is repeated for k times. Since each test is independent from each other, we can run them on different computing units, like multiple processors or multi-core processors, of a machine in parallel. Algorithms mentioned can be easily converted into parallel ones by exploring Parallel Libraries such as gmplib (gml,) or Microsoft's Task Parallel Library (msp,). These libraries create multiple threads as the number of processors in the machine (or number of cores in a multi-core CPU) and share the iteration load among those worker threads allowing concurrent iterations running at the same time. Note that, with single core, machine can do one iteration per unit time which needs $k * unit\,time$ where parallelized version with 2 cores needs only $(k/2) * unit\,time$. In general, we expect to see the time it needs to complete a job in parallel can be estimated by the formula: $(k/\#\,of\,cores) * unit\,time \approx Time\,required\,for\,single\,core/\#\,of\,cores$

Algorithm 1 Prime Number Generation

```
while (notPrime)
{
p := GeneratePrimeCandidate(bitLength)
notPrime = PrimalityTestingAlgorithm(p); // primality testing algorithm returns true if the number p is prime, false otherwise
}
```

2.2 Prime Generation and Testing

We implemented a basic prime-candidate number generation algorithm which builds a random big number of the given bit length. It checks and returns the candidate if the number is odd and is not divisible by any of the numbers in $I = [3, 10]$. For every prime-candidate, chosen primality testing algorithm runs and decide if the number is a "Prime" or "Composite". This process continues until a prime number is found. Pseudo code for the prime generation algorithm and prime candidate generation algorithm is given in Algorithm 1 and Algorithm 2 respectively.

Due to space limitations, details of primality testing algorithms are skipped in this paper but can be found in (mra, ;

Algorithm 2 Generate Prime Candidate Algorithm

GeneratePrimeCandidate(bitLength) // bitLength: set the bit length of the random big number
{
while (number p is not a candidate)
{
p := random odd big number of length bitLength
if(p not satisfies divisibility rules for basic numbers in $I = [3, 10]$) // so it is *not* divisible by these numbers
return p;
}
}

Algorithm	Complexity
Miller Rabin	$O(k * log^3 n)$
Miller Rabin Deterministic	$O((logn)^4)$
Solovay Strassen	$O(k * log^3 n)$
Fermat	$O(k \times log2n \times loglogn \times logloglogn)$

Table 1. Complexity Table

sol, ; mrd, ; fer,).

2.3 Complexity Analysis

Probabilistic algorithms runs for k times to conclude "Prime" but can can stop iterating whenever any of the iterations reports "Composite". Hence their worst case complexity depends on the k value. Table 1 shows the worst case complexity of the algorithms implemented.

3. Results

3.1 Implementation & Test Environment

We implemented the following algorithms in Microsoft C# language on .NET 4.0 platform. Visual Studio 2010 Beta 2 is used in implementation. For the BigInteger operations, GNU MP Bignum Library (gml,) is used. This library has been coded in C/C++ and even some parts in assembly and claims to be the fastest library on the planet. For the .NET wrapper (emi,) is used. Note that .NET wrapper is also based on (gml,).

Our test environment is a Core 2 Quad 2,40GHz desktop machine. The algorithms were implemented in C# so that the test machine is installed with Windows 7 and .NET 4.0 beta which has native support for parallel library.

3.2 Test Results

For each algorithm and number of cores pair, we generate 50 prime numbers for each bit length of 512, 1024 and 2048 using probabilistic verification algorithms. We measure the time it takes to generate and verify a prime under different

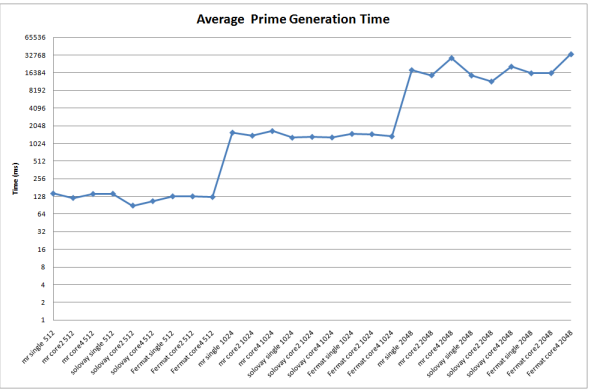


Figure 1. Average prime generation times vs algorithms and the number of cores

number of cores and algorithms. The average prime generation times (including verification times) is given in Figure 1.

In Figure 1, it is seen that time to generate primes is dominant over the prime verification which we parallelized. Figure 1 also concludes that parallelizing verification algorithms is not the best way to speed up the prime number generation.

In prime generation, we generate a prime candidate and try to verify it with the primality testing algorithms. Each iteration consists of two phases: generate a prime candidate and verification. The iterations continues until a prime found.

Since the focus of this paper is to present the effect of parallelism over the prime verification algorithms, we also calculate the time it takes to verify the prime number for each algorithm. Results are the average time of 50 tests. Net effect of parallelism over the prime verification algorithms, the average verification times is shown in Figure 2.

Parallelism of the algorithms are focused on distributing verification tests (i.e the number of repetitions of the algorithms) to all the processors in the machine. Probabilistic algorithms needs to do k tests to conclude. When it is parallelized, the tests will be distributed among all the processing units available. One can expect to see each core in a n-core machine executes k/n verification tests in contrast to the serial execution of a single core machine. This doubles the speed in a dual-core machine while it speeds up to 4 times in quad core machines compared to single core machines. In our tests, each probabilistic test is repeated $k = 40$ times before concluding to prime. $k = 40$ is the number which provides 80 bits of security.

In Figure 2, the effect of parallelism is observed. Speed up is$\approx n$ times in n-core machine. Because of the general overheads, such as partitioning/merging data, scheduling tasks to the run time, synchronization, delegate invocations,

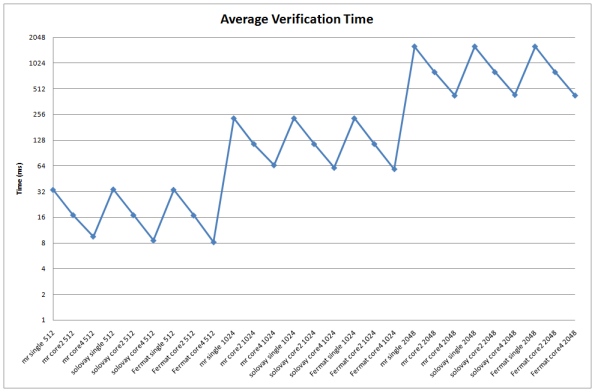


Figure 2. Average verification times vs algorithms and the number of cores

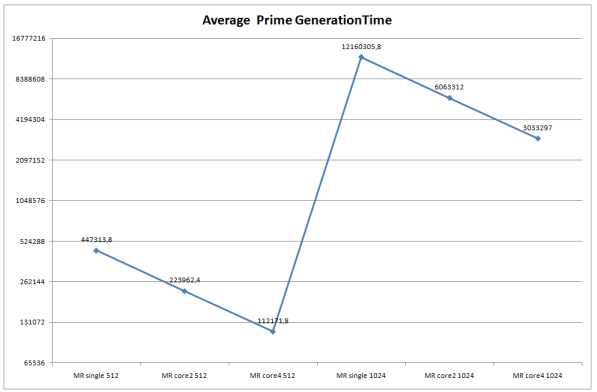


Figure 3. Average prime generation times in MR Deterministic vs bit length and the number of cores

etc. n cores is *not* exactly n times faster.

Deterministic verification algorithms on the other hand needs much longer time to conclude a prime. Since the Miller Rabin Deterministic Algorithm takes too much time, we lowered the number of primes to be generated to 5, and discarded verifying 2048 bits long numbers. Verification time is so high in deterministic algorithms and dominates the prime generation time so we only show the average prime generation times per test in Figure 3.

When we use deterministic algorithms, net effect of parallelism can be easily observed. The speed up is as expected. Unfortunately, these numbers show that deterministic algorithms has no practical usage. When we compare the time it takes to verify a prime using a deterministic algorithm and a probabilistic version, we see a great speed up. For instance, on a single core it takes ≈ 128 ms for MR probabilistic to generate and verify a 512 bits prime. It takes ≈ 447314 ms to generate a 512 bits prime and verify it using MR deterministic algorithm on a single core, showing that probabilistic verification is≈ 3500 times faster than the deterministic algorithm.

4. Conclusion

In this paper, we implemented and presented various primality testing algorithms. Our aim is to do a comparison of these algorithms with respect to their average processing times under different number of processors. We see that number of repetitions (i.e k in the algorithms) is the key factor in determining the performance of the probabilistic algorithms. On the other hand, deterministic algorithm is very inefficient. The most reliable and fast enough algorithm is Miller Rabin Probabilistic Test. We choose Miller Rabin Algorithm because it has higher accuracy. In Miller Rabin Algorithm, $\epsilon < 4^{-k}$ while in Solovay-Strassen Algorithm it is $\epsilon < 2^{-k}$ as discussed in (cry,). For the digital signatures, error probability should be $\epsilon < 2^{-80}$ so repeating Miller Rabin test for 40 times is enough for today's security demands. (cry,).

The performance of the Miller Rabin Probabilistic Algorithm is as expected and suitable for use in practice although it is deterministic version, is very slow with no practical usage.

To sum up, Miller Rabin Probabilistic Algorithm is the best choice for the usage. The key factor in both performance and the accuracy of the algorithm is the k parameter for the probabilistic algorithms and can be chosen as 40 for Miller Rabin test.

References

Cryptography in c and c++ by michael welschenbach.

Gmplib .net wrapper http://www.emilstefanov.net/projects/gnumpdotnet.

Gnu multiple precision arithmetic library http://www.gmplib.org.

Microsoft task parallel library http://msdn.microsoft.com/en-us/concurrency.

Miller rabin deterministic primality test miller, gary l. (1976), "riemann's hypothesis and tests for primality", journal of computer and system sciences 13 (3) 300-317.

Miller rabin primality test rabin, michael o. (1980), "probabilistic algorithm for testing primality", journal of number theory 12 (1): 128-138.

Solovay, robert m.; strassen, volker (1977), "a fast monte-carlo test for primality", siam journal on computing 6 (1): 84-85.

Thomas h. cormen, charles e. leiserson, ronald l. rivest, and clifford stein. introduction to algorithms, second edition. mit press and mcgraw-hill, 2001. isbn 0-262-03293-7. pages 889-890 of section 31.8, primality testing.

Table1 presents some results about p53. The first column stands for a substitution in a given position while the second column describes the effect of that substitution. The last column shows the predictions made by SIFT using the profile matrix created in the prior steps. According to the profile matrix, p53 was found as a highly conserved protein (%57). The substitutions V216A, Q5H, P151S and R175H have been observed in sporadic cancer types. Based on the conservation values for those positions, SIFT predicted those variations as being damaging. Those predictions are also consistent with the information provided from UniProt. P151S, R175H and S241F were found as SNP-related substitutions, two of which being located in a highly conserved position. Their substitution probabilities are 0.0 for the associated SNPs thus, SIFT predicted those substitutions as damaging to the protein, i.e. rendering it nonfunctional. One of those SNPs was found to be located in a position which is not verified as conserved. Hence, SIFT was not able to make a prediction for that position. Creating the multiple sequence alignment profile takes about 3-5 minutes in normal PC for the dataset with 7 sequences. It will have relatively longer running time, if we consider datasets with about 1000 sequences. Therefore, this implementation must be improved in terms of time complexity. On the other hand, the implementation has a memory complexity of O (N^2), which cannot be improved, as the dynamic programming requires the NxN matrices to hold the scoring tables.

In the original CLUSTALW implementation, the calculation procedure used to obtain distance values from the score values is slightly different from ours. The authors of CLUSTALW normalize the number of matching positions by the number of positions compared during the alignment. Meanwhile, in our implementation, it is decided that it is better if the alignment scores were normalized with the highest scores obtained from the pairwise alignment. This leads to the differences in the distance and weight values. Similarly, for the creation of the distance matrix, UPGMA method is implemented, while in the original CLUSTALW application, Neighbor-Joining (NJ) method is used for the generation of the guide tree,. The sample calculations given in the CLUSTALW paper are carried out using the NJ method. This also may lead to differences in the calculated distance values.

5. Conclusion

SIFT Algorithm for nsSNP/protein relation and CLUSTALW algorithm as a subroutine for SIFT, were implemented in Python. For the initial part of the SIFT algorithm BLAST from the Biopython was adopted and MSA step of SIFT was accomplished via implementation of CLUSTALW. The predictions and calculations of SIFT algorithm was again written in Python Programming Language.

The results obtained, were constructed based on the profile matrix without any structural insights. The main concern for the algorithm was the conservation profiles of each position. Highly conserved regions usually have both functional and structural importance. Important sites in biological units, especially in proteins, are low-tolerated to substitutions and preserve a residue-conserved behavior. Therefore, using an evolutionary profile is fairly simple but gives accurate results based on conservation. SIFT tries to make use of this behavior for predicting the effects of amino acid substitutions. In our dataset, we used UniProt database for validation of the results. For p53, the findings of SIFT algorithm are well supported by UniProt.

References

Altschul, S. F., T. L. Madden, et al. (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." Nucleic Acids Research **25**(17): 3389-3402.

Dayhoff, M. O., Schwartz, R. M., Orcutt, B. C. (1978). "A model of evolutionary change in proteins." Atlas of Protein Sequence and Structure **5** (3): 345–352.

Henikoff, S. and L. Comai (2003). "Single-nucleotide mutations for plant functional genomics." Annual Review of Plant Biology **54**: 375-401.

Henikoff, S. and J. G. Henikoff (1992). "Amino-Acid Substitution Matrices from Protein Blocks." Proceedings of the National Academy of Sciences of the United States of America **89**(22): 10915-10919.

Krawczak, M., E. V. Ball, et al. (2000). "Human gene mutation database - A biomedical information and research resource." Human Mutation **15**(1): 45-51.

Ng, P. C. and S. Henikoff (2001). "Predicting deleterious amino acid substitutions." Genome Research **11**(5): 863-874.

Ng, P. C. and S. Henikoff (2002). "Accounting for human polymorphisms predicted to affect protein function." Genome Research **12**(3): 436-446.

Ramensky, V., P. Bork, et al. (2002). "Human non-synonymous SNPs: server and survey." Nucleic Acids Research **30**(17): 3894-3900.

Saitou, N. and M. Nei (1987). "The Neighbor-Joining Method - a New Method for Reconstructing Phylogenetic Trees." Molecular Biology and Evolution **4**(4): 406-425.

Thompson, J. D., D. G. Higgins, et al. (1994). "Clustal-W - Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice." Nucleic Acids Research **22**(22): 4673-4680.

Yue, P., E. Melamud, et al. (2006). "SNPs3D: Candidate gene and SNP selection for association studies." Bmc Bioinformatics **7**: -.

Predicting the effects of non-synonymous SNP variants on protein function using SIFT

Ceren Tüzmen ctuzmen@ku.edu.tr
Beytullah Özgür bozgur@ku.edu.tr
Bora Karasulu bkarasulu@ku.edu.tr

Department of Computational Science and Engineering
Koc University, Sariyer, 34450
Istanbul, TURKEY

1. Introduction

Polymorphisms are defined as variations between the genomes of two randomly selected individuals. Among all kinds, the simplest and the most frequent forms are the Single Nucleotide Polymorphisms (SNPs). Non-synonymous variants of SNPs constitute more than 50% of the mutations which are involved in human inherited diseases(Krawczak, Ball et al., 2000). A non-synonymous single nucleotide polymorphism (nsSNP) when found in a coding gene may result an amino acid substitution, thus may render the resultant protein product nonfunctional. This fact establishes the significance of non-synonymous SNPs in human health and its potent effects on individuals.

Various algorithms are developed to be used in SNP discovery and in understanding its impact on the function of the corresponding protein. Some methods(Ramensky, Bork et al., 2002) concentrate only on human proteins, therefore cannot be applied on data belong to other organisms, while some other(Yue, Melamud et al., 2006) use loads of data to analyze SNP/protein function relationships. 'Sorting Tolerant From Intolerant' (SIFT) has been primarily applied to human genes, however it is applicable to any organism such as bacteria, plants and other animals(Ng and Henikoff, 2001; Henikoff and Comai, 2003). Instead of considering the position of the substitution, i.e. whether the replacement is likely to destroy the hydrophobic core of a protein, electrostatic interactions, interactions with ligands or other important features of a protein, SIFT simply focuses on sequence homology and predicts the effects of all possible substitutions (insertions and deletions -indels- are not included). It assumes important positions in the protein sequence are conserved throughout the evolution, thus cannot tolerate mutations. Hence, nsSNP in those positions can be affecting the phenotype and the protein function.

SIFT is a multi-step algorithm which considers sequence homology for classification of amino acid substitutions(Ng and Henikoff, 2001; Ng and Henikoff, 2002). It was developed by a group at the Fred Hutchinson Cancer Center, and our group implemented the algorithm using Pyhton. Within SIFT a multiple sequence alignment tool is used for aligning the homologous sequences. In our implementation, CLUSTALW(Thompson, Higgins et al., 1994), the most popular MSA tool, is used which sacrifices speed and accuracy, yet uses less memory.

2. Algorithms

2.1 SIFT

Given a protein sequence, SIFT generates a dataset by aligning homologous protein sequences provided by PSI-BLAST(Altschul, Madden et al., 1997). CLUSTALW is used as an MSA tool for aligning homologs of the given protein. In the second step, SIFT probes each position in the alignment and calculates all probabilities for each 20 amino acid at that position.

These probabilities are normalized, then, by the probability of the most frequent amino acid in that position and this information are recorded in a scaled probability matrix. If the nsSNP at a given position has a scaled probability value (or namely a SIFT score) smaller than a pre-defined threshold, then SIFT predicts that nsSNP as deleterious to the host organism. Generally, a highly conserved position is intolerant to most substitutions whereas for positions with low degree of conservations, the case is the other way around. They mostly are able to tolerate substitutions. (Figure1)

SIFT also supplies a measure of confidence within the prediction by calculating a conservation value at each position in the alignment. With this information in hand, SIFT is able to distinguish between neutral and deleterious substitutions by preventing false positive errors which are the false predictions that are found to be deleterious substitutions although they are neutral substitutions.

2.2 CLUSTALW

CLUSTALW is a multiple sequence alignment tool, which is based on progressive-pairwise alignment (using dynamic programming). The homologous protein sequences from PSI-BLAST constitute the input data. It aligns each sequence pairwise using full dynamic programming and calculates the distance matrix in which the information regarding how closely sequences are related with each other are stored. Upon generation of the distance matrix, a phylogenetic guide tree is formed using Neighbor Joining (NJ) method(Saitou and Nei, 1987) or Unweighted Pair Group Method with Arithmetic mean (UPGMA) method. In this tree, closely related sequences are found under the same node or consecutive nodes.

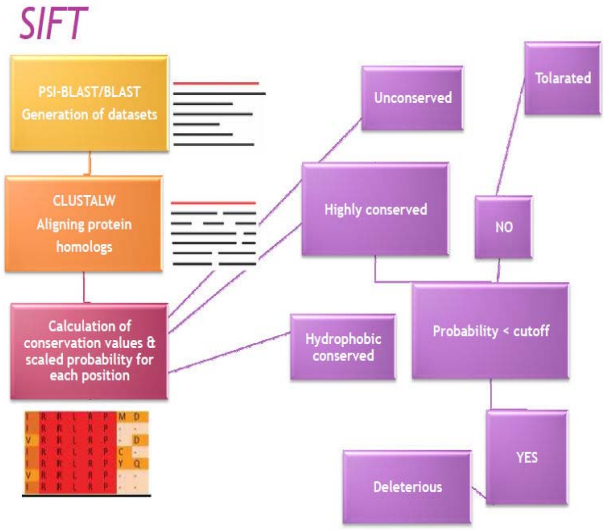


Figure 1-A flowchart for SIFT

Substitution	According to UniProt	According to SIFT
V216A	In sporadic cancers; somatic mutation	Position is Highly Conserved Probably damaging... Probability : 0.0
Q5H	In a sporadic cancer; somatic mutation	Position is Highly Conserved Probably damaging... Probability: 0.012
L43S	In a sporadic cancer; somatic mutation	Position is Not Conserved Not known Probability 0.0
P151S	In LFS; germline mutation and in sporadic cancers; somatic mutation. dbSNP:rs28934874	Position is Highly Conserved Probably damaging... Probability : 0.0
R175H	In LFS; germline mutation and in sporadic cancers; somatic mutation. dbSNP:rs28934578	Position is Highly Conserved Probably damaging... Probability : 0.0
S241F	In LFS; germline mutation and in sporadic cancers; somatic mutation. dbSNP:rs28934573	Position is Not Conserved Not known Probability 0.0

Table 1- Substitutions, their effects and predictions made by SIFT

Within CLUSTALW, progressive alignment is based on solid scoring rules. For scoring, a weight matrix, i.e. a residue-specific score matrix such as BLOSUM(Henikoff and Henikoff, 1992)or PAM(Dayhoff, 1978) is used. Weight matrices are based only on observed alignments, yet no estimations are made for instance by comparing closely related proteins. In CLUSTALW, specific gap opening and gap extension penalty rules are applied. For example, gap opening penalty is reduced for the new gaps in the regions of the sequences, which are observed to have gap-openings in the previous alignments. Another penalty rule is that the gap opening penalty (GOP) is reduced in the hydrophilic regions and GOP is increased at the regions near to the ones already having a gap (for decreasing close gap openings). Actually, the gap opening and gap extension penalties are changed throughout the progressive alignment process based on the weight matrix, similarity of the sequences, lengths of the sequences, and existence of the gaps during early alignments.

3. Implementation

SIFT Algorithm for nsSNP/protein relation and CLUSTALW algorithm as a subroutine for SIFT, are implemented in Python Programming Language.
In our CLUSTALW implementation, for a given sample data set, preliminary pairwise alignments are calculated by subtracting percent identities from 1.0, where percent identities are calculated as number of matching positions in the pairs divided by the length of the shorter sequence. Progressive alignment is carried out using full dynamic programming which uses a scoring function (e.g. BLOSUM62), user input Gap opening (GOP) and extension penalties (GEP). Using the scores of the pairwise alignment, a distance matrix is generated which shows how close the sequence pairs are related.
Distances are calculated by normalizing the alignment score of the given pair by the maximum alignment score of all sequence pairs in the dataset and then, subtracting from 1.0. Successively, a phylogenetic guide tree is created using the UPGMA method and the weight of each sequence is calculated. The sequences are pairwise aligned using full dynamic programming. After the guide was created, it is used for the progressive alignment. In the progressive alignment, sequences are aligned in the order that was defined in the guide tree and the alignment profiles are created using variant scoring functions based on the per cent identities of the sequences calculated during pairwise alignment. Afterwards, following the order of the guide tree, the alignment profiles are aligned using pairwise alignments of the sequences found in the profiles. The gap positions which were newly created in the alignment with the highest score are applied to the other sequences in the profile. The gaps from the previous alignments are kept constant during the new alignments.
Our SIFT implementation uses BLAST from Biopyhton NCBI class for creating a dataset and CLUSTALW manually for multiple alignment of those divergent homologs. It, then, constructs a profile matrix, which contains the probability of 20 amino acids for each position of the alignment. The matrix construction part is the most time-consuming part of the algorithm since it works with complexity of O(N*L) where N is the number of aligned sequences and L is the length of the alignment including the gaps. The remaining parts of the algorithm basically perform assignments for predictions.

4. Results and Discussion

Homologs of p53 were used as a test-set, a protein which is involved in many genetic diseases, such as cancer. 200 homologous sequences were used in order to construct the profile matrix. Median conservation value was calculated as 2.44. A score below a confident score of 3.25 was set by the SIFT algorithm, which presents the divergence of the constructed profile. Using biological feature of each residue, candidate residues were defined and possible effects of the substitutions for those positions were determined. A cutoff probability score of 0.05 was adopted from original SIFT implementation. If the calculated probability for mutation is under this threshold, this mutation is predicted as being deleterious. Our SIFT algorithm was modified such that it also retrieves information from the UniProt about a given protein sequence. After the required information is processed from the UniProt file, the sequence variations, the effects of a substitution and, if there is an SNP for that position, the reference id for that dbSNP are all retrieved.

Table1 presents some results about p53. The first column stands for a substitution in a given position while the second column describes the effect of that substitution. The last column shows the predictions made by SIFT using the profile matrix created in the prior steps. According to the profile matrix, p53 was found as a highly conserved protein (%57). The substitutions V216A, Q5H, P151S and R175H have been observed in sporadic cancer types. Based on the conservation values for those positions, SIFT predicted those variations as being damaging. Those predictions are also consistent with the information provided from UniProt. P151S, R175H and S241F were found as SNP-related substitutions, two of which being located in a highly conserved position. Their substitution probabilities are 0.0 for the associated SNPs thus, SIFT predicted those substitutions as damaging to the protein, i.e. rendering it nonfunctional. One of those SNPs was found to be located in a position which is not verified as conserved. Hence, SIFT was not able to make a prediction for that position. Creating the multiple sequence alignment profile takes about 3-5 minutes in normal PC for the dataset with 7 sequences. It will have relatively longer running time, if we consider datasets with about 1000 sequences. Therefore, this implementation must be improved in terms of time complexity. On the other hand, the implementation has a memory complexity of O (N^2), which cannot be improved, as the dynamic programming requires the NxN matrices to hold the scoring tables.

In the original CLUSTALW implementation, the calculation procedure used to obtain distance values from the score values is slightly different from ours. The authors of CLUSTALW normalize the number of matching positions by the number of positions compared during the alignment. Meanwhile, in our implementation, it is decided that it is better if the alignment scores were normalized with the highest scores obtained from the pairwise alignment. This leads to the differences in the distance and weight values. Similarly, for the creation of the distance matrix, UPGMA method is implemented, while in the original CLUSTALW application, Neighbor-Joining (NJ) method is used for the generation of the guide tree,. The sample calculations given in the CLUSTALW paper are carried out using the NJ method. This also may lead to differences in the calculated distance values.

5. Conclusion

SIFT Algorithm for nsSNP/protein relation and CLUSTALW algorithm as a subroutine for SIFT, were implemented in Python. For the initial part of the SIFT algorithm BLAST from the Biopython was adopted and MSA step of SIFT was accomplished via implementation of CLUSTALW. The predictions and calculations of SIFT algorithm was again written in Python Programming Language.

The results obtained, were constructed based on the profile matrix without any structural insights. The main concern for the algorithm was the conservation profiles of each position. Highly conserved regions usually have both functional and structural importance. Important sites in biological units, especially in proteins, are low-tolerated to substitutions and preserve a residue-conserved behavior. Therefore, using an evolutionary profile is fairly simple but gives accurate results based on conservation. SIFT tries to make use of this behavior for predicting the effects of amino acid substitutions. In our dataset, we used UniProt database for validation of the results. For p53, the findings of SIFT algorithm are well supported by UniProt.

References

Altschul, S. F., T. L. Madden, et al. (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." Nucleic Acids Research **25**(17): 3389-3402.

Dayhoff, M. O., Schwartz, R. M., Orcutt, B. C. (1978). "A model of evolutionary change in proteins." Atlas of Protein Sequence and Structure **5** (3): 345–352.

Henikoff, S. and L. Comai (2003). "Single-nucleotide mutations for plant functional genomics." Annual Review of Plant Biology **54**: 375-401.

Henikoff, S. and J. G. Henikoff (1992). "Amino-Acid Substitution Matrices from Protein Blocks." Proceedings of the National Academy of Sciences of the United States of America **89**(22): 10915-10919.

Krawczak, M., E. V. Ball, et al. (2000). "Human gene mutation database - A biomedical information and research resource." Human Mutation **15**(1): 45-51.

Ng, P. C. and S. Henikoff (2001). "Predicting deleterious amino acid substitutions." Genome Research **11**(5): 863-874.

Ng, P. C. and S. Henikoff (2002). "Accounting for human polymorphisms predicted to affect protein function." Genome Research **12**(3): 436-446.

Ramensky, V., P. Bork, et al. (2002). "Human non-synonymous SNPs: server and survey." Nucleic Acids Research **30**(17): 3894-3900.

Saitou, N. and M. Nei (1987). "The Neighbor-Joining Method - a New Method for Reconstructing Phylogenetic Trees." Molecular Biology and Evolution **4**(4): 406-425.

Thompson, J. D., D. G. Higgins, et al. (1994). "Clustal-W - Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice." Nucleic Acids Research **22**(22): 4673-4680.

Yue, P., E. Melamud, et al. (2006). "SNPs3D: Candidate gene and SNP selection for association studies." Bmc Bioinformatics **7**: -.