A publication of ISCA*:
International Society for Computers
and Their Applications

INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

TABLE OF CONTENTS

*** "International Journal of Computers and Their Applications is abstracted and indexed in INSPEC and Scopus."**

Volume 24, No. 2, June 2017 **ISSN 1076-5204**

International Journal of Computers and Their Applications

A publication of the International Society for Computers and Their Applications

ISCA Headquarters···•···P. O. Box 1124, Winona, MN 55987 USA···•···Phone: (507) 458-4517
E-mail: isca@ipass.net • URL: http://www.isca-hq.org

May 2017

Guest Editor's Note

CATA (Computers and their Applications) is the flagship conference for the International Society of Computers and their Applications (ISCA). The intent of the conference has been to blend theory and practice as a means of stimulating researchers from both research dimensions.

The papers for this special issue have been selected to illustrate the spectrum of the 52 papers presented at the CATA 2017 conference. The authors were asked to extend their work to make the papers journal appropriate, and to change the title of their paper to avoid confusion with their conference paper. This CATA special issue contains the following four papers.

In their paper "Evaluation of Fatigue from 90-Min Reading by Paralanguage Recognition and Gazing-Point Analysis", M. Suganuma *et al.* describe an automated system to evaluate the physical condition of patients.

In their paper "Cloud Service Reliability Assessment and Prediction Based on Defect Characterization and Usage Estimation", A. Bokhary *et al.* present a method to assess the reliability of cloud services.

In their paper "Computing Covers from Matchings with Permutations", A. Fernández *et al.* propose a graph algorithm to derive covers from matchings.

Finally, in their paper "A Software Development Environment for a Multi-Chip Convolutional Network Accelerator", T. Ohkubo *et al.* describe a software development environment which significantly improves development efficiency.

Antoine BOSSARD

Evaluation of Fatigue from 90-min Reading by Paralanguage Recognition and Gazing-Point Analysis

Miyuki Suganuma*, Saaya Urakabe, Ryota Kuramochi,
Shinya Mochiduki, Yuko Hoshino, Mitsuho Yamada
Tokai University, Minato-ku Takanawa 108-8619, JAPAN

Abstract

In 2025, 8 million baby-boomers in Japan will be 75 years old. It is called "2025 problem." Due to the aging of Japanese society, many hospitals and nursing homes require more nurses. If it were possible to determine patients'/care receivers' condition from their everyday behavior, it could reduce nurses' burden and improve patients' quality of life (QOL). In previous research, we examined utterance recognition and training based on lip movement. Also, we examined eye-movement in gazing-point when driving. We found that it was possible to detect not only utterance recognition but also participants' feelings and condition from paralanguage. It has been suggested that activation status of the brain can be measured from visual fixation while gazing. In this study, we collected gazing point (using two types of devices), critical fusion frequency (CFF) value, lip-movement data and pulse in an effort to comprehend participants' fatigue and physical condition.

Key Words: Fatigue; physical condition; paralanguage; lip-movement; eye movement; gazing-point.

1 Introduction

Feelings such as pleasure, sadness and anger have been implemented in emotional recognition technology based on voice analysis (AGI Inc. 2015), and applied in humanoid robots (Softbank Robotics Corp. 2015). In such emotional recognition technology, involuntary elements of the brain affected by emotion create an emotional-recognition algorithm that directly affects the fundamental frequency of the voice (Sato 2007). This means we recognize emotions from nonlinguistic keys like voice quality and variety of phonation, instead of from the words used. These nonlinguistic keys are called paralinguistic information. Paralanguage is information attached to language, such as voice quality and manner of utterance. It is nonverbal elements, such as pitch, volume, speech speed, and voice quality, which were defined by Trager (Trager 1958). We have been working on utterance recognition without voice information, and utterance training based on lip movement to improve participants' pronunciation (Saito 2012 and Wakamatsu 2014).

*Graduate School of Information and telecommunication engineering. Email: m.suganuma@hope.tokai-u.jp.

Figure 1 shows one participant's lip movements as he pronounced "Wabinureba", which is from Ogura Hyakunin Isshu (a traditional Japanese card game). We collected his data at two different times per day on four different days. The left figure shows the morning data, while the right figure shows the evening data. We had already confirmed that similar lip movements could be seen when we collected lip-movement data on the same day at different times. However, utterance timing, strength and weakness of rhythm, and amplitude were different on different days at the same time. This transition was considered to reflect not only vigorousness, but also of fatigue, ingestion, brain disorder, aphthous ulcers, etc. Miwa, et al. indicated that the relationship between fatigue and daytime sleepiness is likely complex, and might be related to Parkinson's disease (Miwa 2011). In the process of this research, we observed fatigue effects of articulation. In addition, stroke patients cannot speak clearly, and they exhibit a prodome of down-curved lips. Based on this background, we consider lip movement to be useful for checking physical condition and symptoms in those new to paralanguage.

On the other hand, our eyes convey a great deal of information in daily life; for instance, they can be energetic or glazed over. Nevertheless, such eye changes cannot be evaluated objectively or measured psychophysically. Thus, we have been focusing on involuntary eye movement during fixation, and evaluating the level of alertness and the concentration ratio (Saito 2015). There are three types of visual fixation: drift, tremor, and flicker. The microsaccade component is said to be particularly reflective of mental state (Stasi 2013). For example, visual fixation is biased in the direction of individuals' attention when they are attending to surrounding areas without changing the gaze point (Kaneko 2009). When we continued the eye movement measurement for a long time, however, the gazing point was expanded because of unstable eye movement while gazing. We then hypothesized two reasons for the gazing-point expansion. The first was an increase in the microsaccade component when an individual's attention is increased. The second was that eye movement became unstable due to fatigue and illness. Based on these two hypotheses, we examined changes in physical condition from eye movement during activity.

In order to expand on our two previous studies of lip movement and eye movement, we carried out three types of measurement: measurement of the gazing point, acquisition

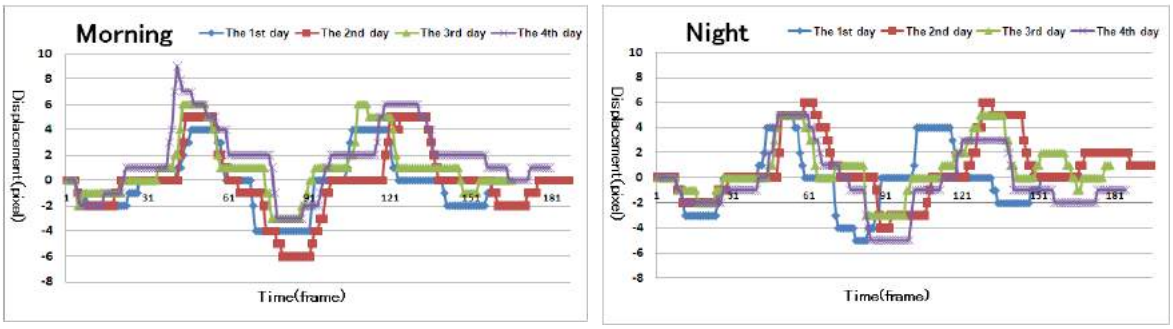


Figure 1: One participant's lip movements pronouncing the word "Wabinureba", which is from Ogura Hyakunin Isshu (a traditional Japanese card game) Left: morning data; Right: evening data

of lip movements, and critical flicker frequency (CFF) value, which is implemented as an evaluation index over time. We proposed two hypotheses for this research. The first was that eye movement of the gazing point and convergence angle would get gradually larger during a 90-min reading task. The second was that lip movement value before reading task would be different from that after reading task. Our aim is to reduce the nurse/caregivers' burden, by helping them to comprehend the patient/care receivers' physical condition, and ultimately, improve their quality of life. Our approach is to attempt to determine changes in fatigue, physical condition, and emotions of patients by analyzing miniature eye and lip movement in face-to-face talk.

In section two, we explain the lip feature point-collecting application developed in our laboratory. The third section describes the methods used in this study, and the fourth presents our results and discussion. Finally, the fifth section contains our conclusions.

2 Lip Feature Point-Collecting Application

Figures 2 and 3 contain screen shots of the display used for the acquisition of lip feature points, and the display for lip-movement training, respectively. When a subject is recognized by the lip feature point-collecting application, the acquired face feature points can then be shown for training. This application was developed in our laboratory using faceAPI (Seeing Machines, Inc., Australia) (Seeing Machines Inc. 2014). To collect lip-movement data, we used the "Get data" function and saved the file name to save data. Data acquisition was initiated by clicking the "Start" button. When a subject closed his or her mouth for 0.5 s, the application recognized that person's mouth, and subjects could start speaking. If the application did not recognize a subject, the "Re-recognition" button was used. The "Stop" button was used to end a recording. The lip-movement training display was shown after the recording was stopped. The red line in the lip-movement training display (Figure 3) shows the teacher's lip movements, which are used as model data; the black line shows the participant's lip movements. The two sets of lip movements

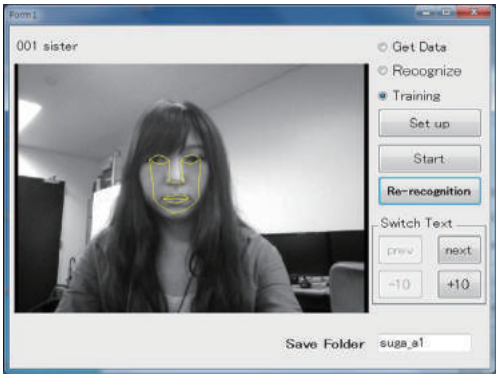


Figure 2: Display for the acquisition of lip feature points

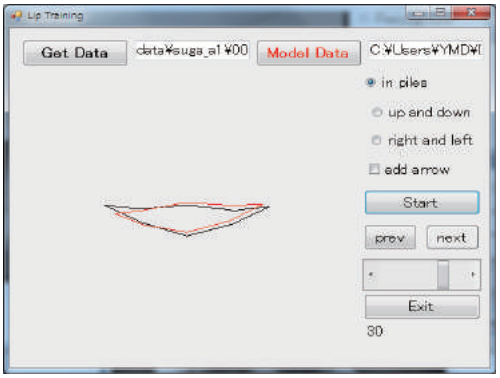


Figure 3: Display for lip movement training

could then be compared by clicking the "Start" button (Figure 3).

3 Methods

The present study was carried out to evaluate participant's fatigue and assess their physical condition based on involuntary eye and lip movements using a lip feature-point collecting application developed in our laboratory. We conducted two types of experiments to confirm the relation between two eye-

movement measuring devices: Tobii EyeX (Tobii AB), and EMR-9 (nac Image Technology Inc.).

3.1 Experiment 1

Figure 4 is a flow chart of the experimental process. First, we measured the participant's CFF value using a flicker-measuring instrument (Takei Scientific Instruments Co., Ltd. 2012). Figure 5 is an experimental image of measuring CFF value. We used the raising and descending method. We then obtained the lip-movement data. Figure 6 is an experimental image of obtaining lip-movement data. Table 1 lists the sentences used in experiment 1. We chose three sentences. "*Watashino namaeha xxdesu* (My name is xx)", "*Kyouha kumotte imasune* (Today is cloudy)", and "*Attara aisouyoku aisatsu shinasai* (You should greet him/her in a friendly way when you meet them)". These are common phrases used in daily life. We choose these 3 sentences because they were easy to analyze in terms of how much speakers opened their mouths. Participants pronounced each phrase 3 times. They then read a novel for an hour using a SONY note PC (VAIO Duo 11). Figure 7 is an experimental image of reading a novel. We selected, "The night of the milky way train", by Kenji Miyazawa for this purpose. We also collected participants' involuntary eye movements using the eye-movement tracker (Tobii AB, Tobii EyeX) while they were reading the novel (Tobii AB. 2014). Finally, we measured the CFF value and lip movement again, in order to compare the participants' fatigue before and after reading the novel.

Four students participated in this experiment (2 males and 2 females); age range was 20-22. The experiment was conducted in our laboratory. To obtain lip-movement data, we used a DELL note PC (Dell XPS). The web camera screen resolution was 640x480.

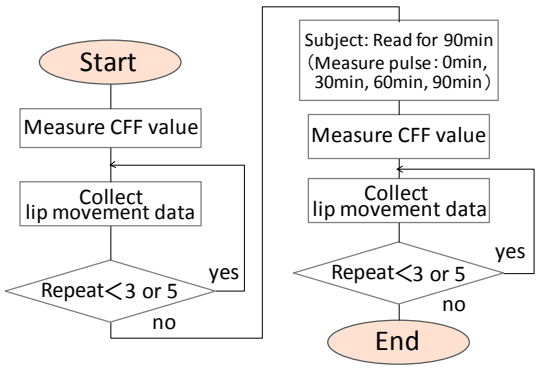


Figure 4: Flow chart of the experimental process

3.2 Experiment 2

The experimental procedure was basically the same as in experiment 1. In experiment 2, we selected five sentences, "*Attara aisouyoku aisatsu shinasai* (You should greet him/her in a friendly way when you meet them)"; "*Ikigai-wo motomete ikou* (Let's seek fulfillment in your life)"; "*Uta-wo utatte usabrashi* (I sing away my troubles)"; "*Eiyo-yo eikou-yo eiennare* (Honor and glory, forever)"; and "*Ookami-no ookina tooboe* (The howling of the wolf)". These sentences were selected from a book called "Easy training for a good voice" (Fukushima 2006). Table 2 presents the five sentences used in experiment 2. Participants pronounced each phrase five times. Participants again read a book for 90 minutes using the same novel and PC. We collected participants' involuntary eye movements using an eye-movement measuring device (nac Image Technology Inc., EMR-9). Figure 8 is an example image of a subject reading a novel in experiment 2. In addition to lip movement, eye movement, and CFF data, we also collected pulse (Custom Corp., NURSE ANGIE Pulse oximeter PLS-01BT). Participants were five students (two males and three females) aged 20–21. The experiments were conducted according to the Ethics Committee regulation about "research for people" of our university.

Figure 5: Experimental image of measuring CFF value

Figure 6: Experimental image of obtaining lip movement data

Table 1: Sentences used in experiment 1

Watashino namaeha xxdesu (My name is xx)
Kyouha kumotte imasune (Today is cloudy)
Attara aisouyoku aisatsu shinasai (You should greet him/her in a friendly way when you meet them)

Figure 7: Experimental image of reading a novel (using Tobii EyeX)

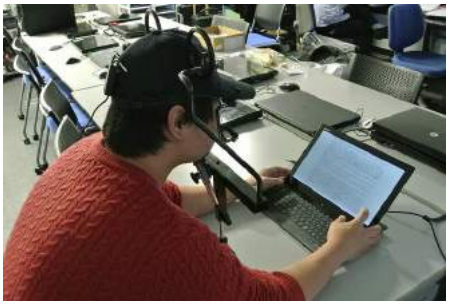


Figure 8: Experimental image of reading a novel (using EMR-9)

Table 2: Sentences used in experiment 2

Attara aisouyoku aisatsu shinasai (You should greet him/her in a friendly way when you meet them)
Ikigai-wo motomete ikou (Let's seek fulfillment in your life)
Uta-wo utatte usabarashi (I sing away my troubles)
Eiyo-o eikou-yo eien-nare (Honor and glory, forever)
Ookami-no ookina tooboe (The howling of the wolf)

4 Results and Discussions

4.1 Analysis of Standard Deviation of Eye Movement During Gazing While Reading

4.1.1 Experiment 1 (Tobii EyeX)

Figure 9 shows participant WTN's eye movement from the beginning to the end of reading. The vertical axis indicates pixels obtained from the eye tracker, while the horizontal axis is time in minutes. In order to confirm the transition of a gazing point's standard deviation (SD), we collected three intervals: 2–3 min, 29–30 min, and 59–60 min. It is clear in the figure that eye movements appeared more frequently at the end of reading compared to the beginning. To calculate the SD of gazing point, we extracted gazing point from the eye movement data while reading. In this paper, we applied fixation separated by the Tobii EyeX as the gazing point. The maximum velocity of smooth-pursuit eye movement, which is eye movement that occurs while gazing at a moving target, is at most 30 deg/s (Westheimer 1954 and Robinson 1961). Therefore, smooth-pursuit eye movement did not occur in the present experiment. Additionally, eye movement velocity of 30 deg/s was used as the threshold for distinguishing saccade and gaze point from eye movement (Matsumiya 2004). The mean SD of the gazing point of all participants is shown in Figure 10. The SD decreased slightly in the middle of the reading (30 min). However, it increased again at the end of reading. We consider that participants were concentrating on reading in the middle of the experiment. However, an indication of fatigue from reading was apparent.

4.1.2 Experiment 2 (EMR-9)

Figure 11 shows participant NKG's average standard deviations in the x- and y-directions during gazing from 2–3, 29–30, 59–60 and 89–90 min after starting measurement. The horizontal axis shows time, while the vertical axis shows angle. The blue bar indicates the eye's movement in the X-direction movement, and the white bar movement in the Y-direction. Gazing point was determined based on eye-movement components with velocity below 5 degrees/s (Yamada 1986). It may be seen in Figure 11 that the average of the standard deviation of the gazing point gradually increased in the X-axis direction. However there was no similar change in the Y-axis direction. We considered that this might have been due to the letters in the novel being written from left to right.

4.2 Changes of Convergence Angle and Standard Deviation

Convergence angle is the angle between the visual axes of the right and left eyes when a subject looks at an object (Nakamizo 1982). When gazing at a point on a 2D plane, the line of sight of both eyes should converge to that one point. However, due to fatigue from reading, we predicted that the angle of convergence would increase by gazing.

Figure 12 presents the changes in participant NKG's convergence angle and standard deviation in each time course. The horizontal axis shows time, and the vertical axis shows angle. Results show that convergence angle decreased from the start of reading to 59–60 min, and then increased again at the end. This suggests that the convergence angle widened due to fatigue.

4.3 CFF Value Before and After

4.3.1 Experiment 1

Table 3 shows the CFF value results before and after reading of 4 participants. We collected this data 4 times, and

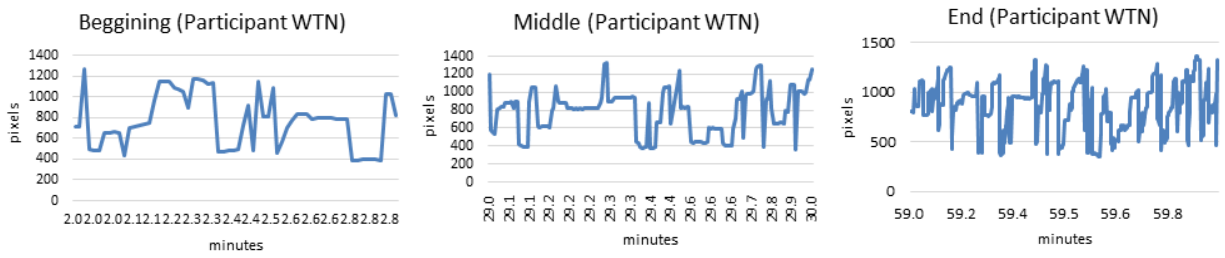


Figure 9: Participant WTNs eye movement from the beginning to the end of reading

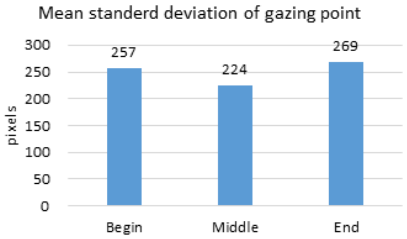


Figure 10: Mean standard deviation of gazing point for all participants

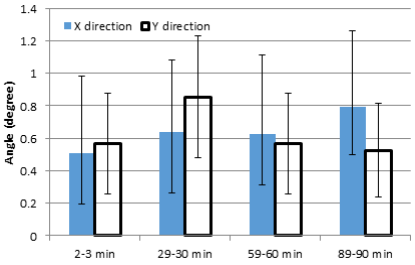


Figure 11: Average standard deviations in the X- and Y-directions of eye movement while gazing (Participant NKG)

then we took an average. We confirmed that all participants' CFF values decreased. Therefore, participants became tired over the 1 hour reading period.

4.3.2 Experiment 2

Table 4 shows results for the CFF value before and after reading in the five participants. We collected this data six times, and results were averaged. We confirmed that CFF values decreased in four of five participants. Therefore, a tendency toward mental fatigue was shown over the 1 hour reading period, as in experiment 1.

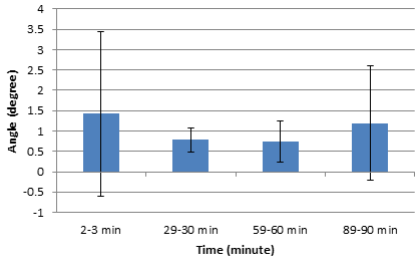


Figure 12: Changes of convergence angle and standard deviation in each time course (Participant NKG)

Table 3: CFF value results before and after reading in experiment 1

	Before	After
WTN	32.6	31.5
KYM	38.8	37.6
KRS	35.9	34.8
KBY	34.9	34.6

Table 4: CFF value results before and after reading in experiment 2

	Before	After
NKG	34.8	30.9
ONE	33.9	33.2
KSM	35.8	33.8
KYM	36.8	36.8
NKI	29.6	28.8

4.4 Lip-Movement Before and After

4.4.1 Experiment 1

Figure 13 shows an example of the area of open-lip space. The hatched area indicates the mouth opening. The open

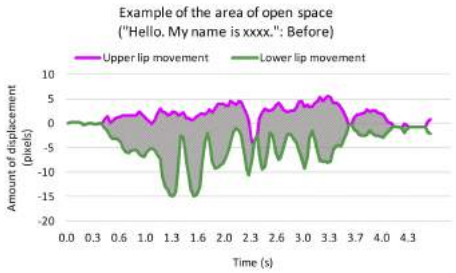


Figure 13: Example of the area of open lip space

Table 5: Area of the opening space of lip movement before and after reading novel for each sentence

	Before	After
Hello. My name is xx.	867.4	761.3
Today is cloudy.	356.3	474.4
You should greet him/her in a friendly way when you meet them.	654.3	807.7

space in the lip movement for each sentence before and after reading the novel is shown in Table 5. Figure 14 is a longitudinal lip-movement line graph of participant KRS uttering, "*Konnichiha. Watashino namaeha xxxx*" (Hello. My name is XXXX). Figure 15 shows the longitudinal lip-movement line graph when participant KRS pronounced, "*Kyouha kumotte imasune*" (Today is cloudy), and Figure 16 is for the same participant saying, "*Attara aisouyoku aisatsu shinasai*" (You should greet him/her in a friendly way when you meet them). As you can see from these 3 line graphs, utterance time length and opening of the mouth were different before and after reading.

In Figure 14, before reading the participant moved his mouth clearly, because some parts show a closed mouth, and utterance time was 3.9 s. On the other hand, in Figure 14 after reading, it appears at first glance that his mouth was opened widely. However, utterance time was shorter (3.0 s). We confirmed fatigue due to reading by the decrease in the lip-opening space.

In Figure 15, utterance time transitions were 2.3 to 1.9 s, not a huge difference. Also, the lip-opening space became larger after reading the novel. It was considered that this was because this sentence was shorter than the others in comparing the difference between before and after reading.

In Figure 16, the utterance time was 2.3 s before reading and 2.2 s after, a little bit decreased. We confirmed that participants were not able to speak clearly, because the number of wave crests also decreased. Moreover, it was suggested that this sentence was most affected by fatigue as the reason for the decrease in lip-opening space.

In Table 5, the sentences "Hello. My name is XXXX," also

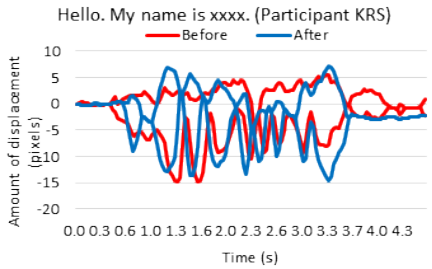


Figure 14: Longitudinal lip-movement line graph when participant KRS uttered Konnichiha. Watashino namaeha xxxx (Hello. My name is XXXX)

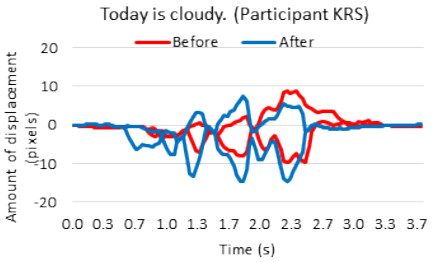


Figure 15: Longitudinal lip-movement line graph when participant KRS uttered Kyouha kumotte imasune (Today is cloudy)

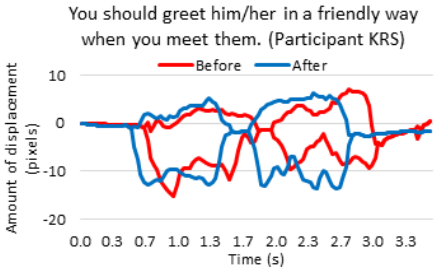


Figure 16: Longitudinal lip-movement line graph when participant KRS uttered Attara aisouyoku aisatsu shinasai (You should greet him/her in a friendly way when you meet them)

indicate decreased lip-opening space. However, this space increased in the sentences "Today is cloudy" and "You should greet him/her in a friendly way when you meet them". It is assumed that participants attempted to pronounce words carefully even after the reading task. From these results, we confirmed that reading a novel for an hour tended to affect not only visual fixation but also participant's utterance speed and lip-opening space.

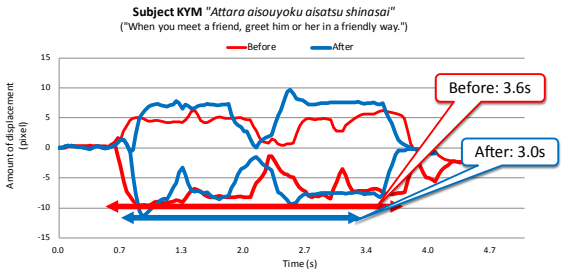


Figure 17: Longitudinal lip-movement line graph when participant KYM uttered Attara aisouyoku aisatsu shinasai in experiment 2

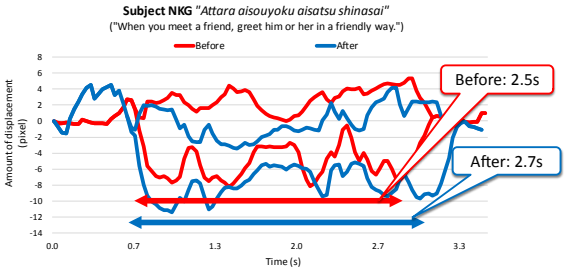


Figure 18: Longitudinal lip-movement line graph when participant NKG uttered Attara aisouyoku aisatsu shinasai in experiment 2

4.4.2 Experiment 2

Figure 17 shows subject KYM's lip movement when she pronounced "*Attara aisouyoku aisatsu shinasai*". The red line indicates lip movement before reading, and the blue line after reading. The utterance time was 3.6 s before reading, and 3.0 s after. Therefore, the time became shorter after reading. Figure 18 shows subject NKG's lip movement pronouncing the same sentence ("*Attara aisouyoku aisatsu shinasai*"). Unlike KYM, NKG's utterance time was 2.5 s before reading, and 2.7 s after; i.e., utterance time increased.

As for open-lip space, it was calculated using the same method as in experiment 1 (Figure 13). Table 6 shows only the increase and decrease of lip-opening space before and after reading. The sentences used in this experiment included many Japanese vowels. For example, "*Attara aisouyoku aisatsu shinasai*" includes many "*a*" sounds. Japanese has five vowels: /a/, /o/, /u/, /i/, /e/. Pronouncing /a/, /o/ and /u/ requires opening the mouth widely in the longitudinal direction; meanwhile, for /i/ and /e/ the mouth must open widely in the horizontal direction. The area of open-lip space increased in subject KYM when sentences had a lot of vowel "*i*" and "*e*" sounds. In contrast, the area of lip-opening space increased in subject NKG during sentences that had a lot of vowel "*a*", "*u*" and "*o*" sounds. As these results were opposite, we could not affirm whether such changes in open-lip space increased from fatigue or if subjects became accustomed to using the lip-feature point collecting application.

Table 6: Area of open lip space in five sentences

	KYM	NKG
Attara aisouyoku aisatsu shinasai	-0.1	+179.4
Ikigai-wo motomete ikou	+111.0	-45.3
Uta-wo utatte usabarashi	-162.3	+29.4
Eiyo-yo eikou-yo eien-nare	+21.7	-200.2
Ookami-no Ookina tooboe	-117.9	+106.1

4.5 Pulse (Only Experiment 2)

Figure 19 shows the pulse rate of five subjects during reading. The horizontal axis shows time, and the vertical axis shows pulse. In this experiment, no tendencies were observed between the five subjects during 1.5 hour reading. This was possibly because the pulse is very individual. KSM's pulse was high at the beginning of reading, but this reaction was likely due to strain, because it was his first time participating as a subject.

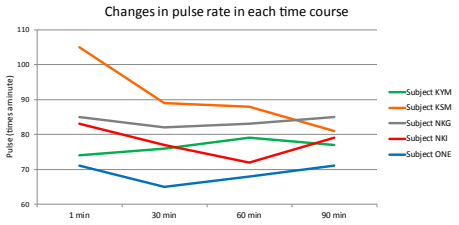


Figure 19: Changes in pulse rate in each time course of five participants

5 Conclusions

We investigated participants' fatigue and physical condition based on involuntary eye movement and lip movements with the goal of ultimately reducing nurses'/caregivers' burdens by allowing them to more easily comprehend patients'/care receivers' physical condition. This will also improve patients' quality of life. As we mentioned in the beginning, we set two hypotheses for this research. The first was that eye movement of gazing point and convergence angle would gradually increase during a 90-min reading task. This hypothesis was disproved due to individual differences. The second was that lip-movement values before the reading task would be different from those after the reading task. This hypothesis was confirmed. There

was an evident difference in lip movement before and after reading.

Our results showed that the SD of the gazing point was decreased in the middle of the reading task (30 min) in experiment 1. However, the SD gradually increased in experiment 2. The convergence angle decreased from the beginning to the middle of reading, suggesting the possibility of mental fatigue due to reading.

As for the CFF value results, we confirmed that almost all participants had mental fatigue over the course of the reading task in both experiments, causing the CFF values to decrease.

The lip-movement data confirmed that reading a novel for 1 or 1.5 h tended to affect not only the gazing point but also the participants' speaking speed and lip-opening space. In experiment 2, two participants had opposite results in terms of time and area of open-lip space. We therefore cannot assert whether these changes were caused by fatigue. We plan to attempt to prove this phenomenon in future research. Nor did we observe a tendency in the relationship between pulse and other parameters. Participants were pronounced Japanese sentences in this experiment because they were Japanese students. However, we need to apply the lip feature point-collecting application to English or any other language utterance data in order to expand this research to worldwide. In our previous research, we had been researched English pronunciation training using the lip feature point-collecting application and the results suggested that the participants' pronunciation was closer to the English teacher's pronunciation (Suganuma 2017). With regard to this research, it is assumed that this application can apply to other language such as English. Our findings indicate the possibility of being able to recognize changes in fatigue and physical condition from miniature eye and lip movements.

In future research, we plan to collect more data to clarify the usefulness of the indexes that we implemented and to conduct statistical analysis. In addition, we plan to include other physiological indicators, and carry out a detailed fatigue questionnaire, and a short quiz about the novel to prevent aimless reading. In terms of eye movement, we would like to confirm there is further change in the gaze point depending on the task content. We will reexamine tasks with fewer burdens on the subject's posture, because some participants reported that the posture during the task was difficult. For the lip movement, we need to compare and examine each phrase. Although we collected only students' data in this experiment, Hara et al. said that the sound factors of the speech of the caregivers showed a significant association with the QOL score related to the physical health condition. Also, they said the value of Pitch Period Perturbation Quotient, Amplitude Perturbation Quotient and Noise-to-Harmonic Ratio by the sound analysis may be one evaluation index for tracking the elderly health survey and its progress (Hara 2015). Therefore, we are going to apply this research to the elderly people by analyzing not only the lip-movement data also the speech data. Furthermore, we will apply this research in various fields such as robots that communicate with patients and care givers, robot receptionists, and remote diagnosis via smart phone.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 16K01566.

References

[1] AGI Inc. Emotion Recognition. http://www.agi-web.co.jp/english/index.html, October 2015.

[2] E. Fukushima, *Easy Training for a Good Voice*, Seibido Printing, Tokyo, 2006, in Japanese.

[3] S. Hara, H. Miura, K. Yamasaki, N. Morisaki and Y. Sumi, "The Association Between Health-Related Quality of Life and Voice as Evaluated by an Acoustic Analysis in Elderly Japanese Nursing Home Residents," *Japanese Journal of Geriatrics*, 52(4): 391-398, 2015, in Japanese.

[4] H. Kaneko, "Fixational Eye Movements", *The Journal of the Institute of Image Information and Television Engineers*, 63(11): 1538–1539, 2009.

[5] K. Matsumiya and K. Uchikawa, "Measurement of Contrast Sensitivity in the Peripheral Visual Field During Saccadic and Pursuit Eye Movements Using Method of Fixed Retinal-Area Stimulation," *Japanese Journal of the optical society of Japan*, 33(2): 122–129, 2004.

[6] H. Miwa and T. Miwa, "Fatigue in Patients with Parkinson's Disease: Impact on Quality of Life," *Internal Medicine*, 50(15): 1553–1558, 2011.

[7] S. Nakamizo, K. Shibuta and M. Noguchi,"Magnitudes of Disparity Vergence Responses at Different Convergence Levels," *Japanese Psychological Research*, 24(4): 181–187, 1982.

[8] D. A. Robinson, "The Mechanics of Human Smooth Pursuit Eye Movements," *F. Physiol.*, 180: 569–591, 1961.

[9] N. Sato and Y. Obuchi, "Emotion Recognition Using Mel-Frequency Cepstral Coefficients," *Information and Media Technologies*, 2(3): 835–848, 2007.

[10] T. Saito, M. Ohiro, T. Onoue, A. Kasahara, T. Shinkawa and M. Yamada, "A Study of an Utterance Recognition Method Using Correlation of Power Spectrum of Characteristic Lip Movements," *Bulletin of Tokai University Department of information and telecommunication engineering*, 5(2): 36–44, 2012, in Japanese.

[11] Y. Saito, G. Iiduka and M. Yamada, "Gazing Point Analysis by 4K Driving Simulator," *IEICE technical report*, 115(133), pp. 5–8, 2015, in Japanese.

[12] Seeing Machines Inc. FaceAPI. https://www.seeingmachines.com/, May 2014.

[13] SoftBank Robotics Corp. http://www.softbank.jp/en/robot/, July 2015.

[14] D. Stasi, L. Leandro, M. B. McCamy, A. Catena, S. L. Macknik, J. J. Canas and S. Martinez-Conde, "Microsaccade and Drift Dynamics Reflect Mental Fatigue," *European Journal of Neuroscience*, 38(3): 2389–2398, 2013.

[15] M. Suganuma, T. Yamamura, Y. Hoshino and M. Yamada, "Proposal of the Way of English Pronunciation Training Evaluation by Lip Movement," *Japan Personal Computer Application Technology Society*, 11(2): 8-20, 2017, in Japanese.

[16] Takei Scientific Instruments Co., Ltd. http://www.takei-si.co.jp/en/index.html, April 2012.

[17] Tobii AB. Tobii EyeX. https://tobiigaming.com/, March 2014.

[18] G. L. Trager, "Paralanguage: A First Approximation," *Studies in Linguistics*, 13: 1–12, 1958.

[19] E. Wakamatsu, Y. Hoshino and M. Yamada, "Proposal for an Utterance Training Method Based on Lip Movements," *Image Media Quality and its Applications 2014*, pp. 44–47, 2014.

[20] G. Westheimer, "Eye Movement Responses to a Horizontally Moving Visual Stimulus," *Arch Ophthamol*, 52: 932–941, 1954.

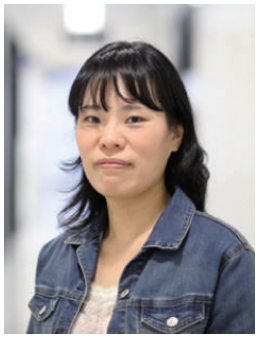
[21] M. Yamada and T. Fukuda, "Quantitative Evaluation of Eye Movements as Judged by Sight-Line Displacements," *SMPTE Journal*, 95(12): 1230–1241, 1986.

Miyuki Suganuma received the B.S. in Information Technology from Tokai University in 2016. Currently underway of paralanguage recognition and gazing point study for fatigue and health condition evaluation at graduate school of Tokai University.

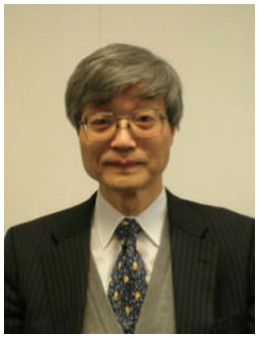
Saaya Urakabe received the B.S.in Information Technology from Tokai University in 2017. Engaged paralanguage recognition and gazing point study for fatigue and health condition evaluation at School of Tokai University.

Ryota Kuramochi received the B.S.in Information Technology from Tokai University in 2017. Engaged paralanguage recognition and gazing point study for fatigue and health condition evaluation at School of Tokai University.

Shinya Mochiduki received the B.S. and M.S. degrees in Information Technology from Tokai University in 2015 and 2017. Currently underway of information processing of visual system and human interface study at graduate school of Tokai University.



Yuko Hoshino received the B.S. and M.S. degrees in Electrical Engineering from Kogakuin University in 1998 and 2000, respectively. Entered Creo, Co in 2000. She is now with Tokai University. Currently underway of software development study.



Mitsuho Yamada received the B.S. and M.S. degrees in Electrical Engineering from Nagoya University in 1978 and 1980, respectively. Entered NHK (Japan Broadcasting Corporation) in 1980. He is now with Tokai University. Doctor of engineering. Currently underway of information processing of visual system and human interface study.

Cloud Service Reliability Assessment and Prediction Based on Defect Characterization and Usage Estimation

Abdullah Bokhary*
Southern Methodist University, Dallas, Texas 75275, USA
and University of Jeddah, Jeddah, SAUDI ARABIA

Jeff Tian†
Southern Methodist University, Dallas, Texas 75275, USA
and Northwestern Polytechnical University, Xi'an, CHINA

Abstract

Cloud computing has become a major resource for fulfilling people's computational and storage needs. Investing in these services requires measuring and assuring its reliability. However, using traditional reliability models can be challenging because of the environmental constraints and limited data availability due to the heterogeneous environment and diverse stakeholders. This paper proposes a framework to measure reliability with alternative available information that most cloud providers offer in three stages: 1) Defects are extracted and weighed from issue report based on their validity. 2) Workload is measured by the number of clients as a new proxy to estimate daily clients usage. 3) Both results are linked together to examine the defect behavior over time. Software reliability growth models (SRGMs) are used to analyze this behavior, to assess current reliability, and to predict future reliability. Google Maps APIs is used as a case study to demonstrate the applicability and effectiveness of our new framework. Finally our framework is validated by extending the models to provide reasonably accurate long term reliability predictions.

Key Words: Cloud reliability, usage measurement, clients count, defect data, Google maps APIs.

1 Introduction

Nowadays, Internet gives birth to cloud computing, which changes the way people use computation resources. According to the National Institution of Standard and Technology (NIST), cloud computing is a model that uses communication technology to allow global customers to share computing resources such as networks, servers, storage, applications, and services [6]. In addition to providing cloud computing services to end users, organizations use it to improve performance and reduce cost by replacing server rooms with cloud computing. Developers also take advantage of cloud computing to embed cloud services in their systems and applications to satisfy their customer's needs.

The high demand for cloud computing increases the need to assure its quality, including reliability, usability, and security as primary concerns [10]. Reliability is an important attribute in building a heterogeneous system with intrinsic high complexity. Software reliability is the probability of not having a failure over a specific period [8]. Therefore, we defined the cloud service reliability as the probability of not having a failure for the services, where a failure is the inability to correctly process a customer request.

Researchers and practitioners have used many methods to measure and predict software reliability during development or operation phases. However, these traditional approaches are difficult to apply in cloud computing because of the environmental constraints and limited data availability. Unlike in traditional software systems, defects can be associated with heterogeneous components distributed over wide areas and different layers of the cloud infrastructure. Workload measurement is harder to obtain due to the different stakeholders involved. Also, service providers may rely on clients to report defects who may be reluctant to share detailed circumstantial information about these defects due to legal and proprietary concerns. These limitations need to be taken into consideration when we address cloud reliability problems from the perspective of clients or developers who use these services.

This paper proposes a framework which overcomes these limitations to assess and predict cloud service reliability. The framework has three stages: 1) extract and process defect data from clients report system, 2) identify and extract proper workload from client usage, 3) using both results to assess and predict reliability using existing reliability models. Google

*Dept. of Computer Science and Engineering and Dept. of Information Technology. Email: abokhary@uj.edu.sa
†Dept. of Computer Science and Engineering and School of Computer Science. Email: tian@lyle.smu.edu

Maps APIs was analyzed using the proposed framework as a case study to demonstrate its applicability and effectiveness.

The paper is organized as the following: Section 2 includes background and related work. Section 3 describes our new method, its three stages, and steps for each stage. Section 4 uses the framework on Google Maps APIs to assess and predict its reliability. Section 5 provides a long term validation for the framework by extending the fitted reliability models to provide reasonably accurate reliability predictions. Finally, Section 6 summarizes the paper and future work planned to overcome some limitations of our study.

2 Related Work

Software quality is usually associated with satisfying user expectations as characterized in user requirements and product specifications [11]. Deviation to such expectations are characterized by system defects, either in the form of failures, which are observable behavioral deviations, or faults, which are the internal problems in the system that may cause failures. From a user's perspective, system quality can be measured by its reliability, or how likely the system is going to satisfy user needs without causing a problem.

Software reliability is defined as the probability of a software system to perform its specified functions correctly during a specified exposure period under the customers' usage environment or similar environments [8]. Software reliability growth models (SRGMs) are time-based models commonly used to assess and predict reliability by analyzing defects data over time [5]. The reliability growth is due to the defect detection and fixing that lowers the number of system faults and leads to improved reliability over time. One of the widely used SRGMs is Goel-Okumoto model (GO) from Non-Homogenous Poisson Process (NHPP) model class [4]. In this model, the mean value function for the number of failures is:

$$\mu(t) = N(1 - e^{-bt}) \tag{1}$$

which predicts the cumulative defects in each given time (t), where b and N can be estimated from observation data.

Another NHPP model is the S-shaped model [16], which considers the learning curve in the beginning and uses the following formula:

$$\mu(t) = N(1 - (1 + bt)e^{-bt}) \tag{2}$$

where t, b, and N are similar to the GO model.

Another widely used NHPP model is Musa-Okumoto Model(MO) [7]. It uses the following formula:

$$\mu(t) = \beta_0 \ln(\beta_1 t + 1) \tag{3}$$

where β_0 and β_1 are constants that can be estimated from observation data.

The period can be measured by time units that reflect actual usage by the customer and users, because software failures are triggered by actual usage that exposes some internal defects. Software reliability modeling used calendar time as the time measurement until Musa introduced execution time to better characterize actual system usage or workload [8]. Alternative usage related time measurements have also been used for reliability modeling, including test runs and transitions from test tracking reports and number of usage instances and web traffic extracted from web logs [1][9][14][15].

Although cloud computing provides the advantage of sharing computing resources, it has some limitations. One such limitation can be a result of encapsulation of cloud service which hides the system specification including low level architecture. Also, the traditional data sources of defect such as testing reports or log files are generally unavailable for clients due to their legal or proprietary concerns.

Past failure data of other similar users were used to predict the web service reliability for the current users [17][18]. An enhancement to this work was done by considering provider and client location, service load, and computational requirements [12]. However, this approach required historical data from other similar users which might not be available for cloud services. Also, the approach does not predict future reliability.

Available methods for measuring software reliability would be challenging to apply in cloud service because of the limited data availability. Calendar time is not an accurate workload representation in reliability models for cloud services due to the fast growth or change of service usage. Also, execution time, or number of service invocations are not available. Therefore, we need an alternative data source that represents defect behavior over appropriately defined usage time to measure cloud service reliability.

3 A New Method

We propose a new framework that uses weighted defects from issue report over a new client usage proxy to characterize defect behavior over time and to assess and predict reliability.

3.1 Overall Approach

The limited information of cloud computing, such as lack of access to source code or execution logs, prevents clients from directly applying the traditional reliability models. Also, difficulty of analyzing the extremely large log file motivates providers to search for alternative metrics to measure and predicate the reliability of their services. In order to measure reliability, an alternative measurement of defect over an accurate representation of workload need to be used.

Most cloud providers offer defect reports or a feature requests system. This system can be a source for defect data, since it has all the details of each defect, including discovery time and how it has been treated. However, these defect data are in calendar time, which is not an accurate usage metrics for cloud service due to large usage variations. Also, invocation count for the service by customer is not available. Therefore, we propose to use the number of clients instead of calendar time or service

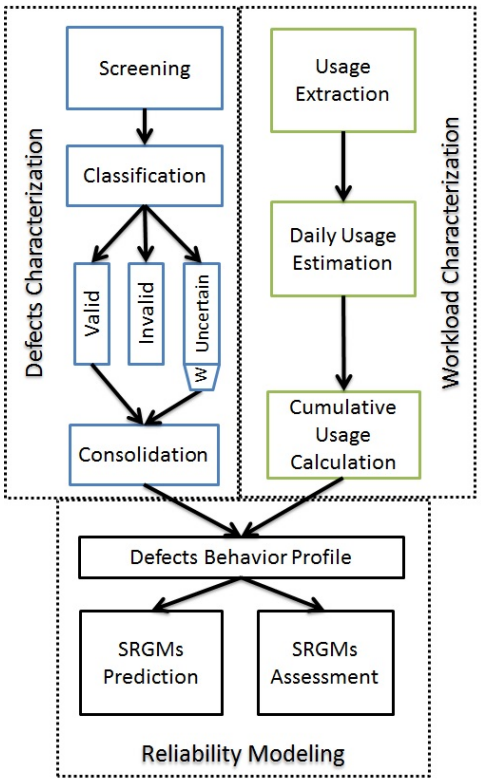


Figure 1: Framework to measure cloud reliability

invocations. In other words, the number of clients that accessed the service when defects were reported can be a proxy for usage. These types of information are offered by some providers or available in external sources such as web analytics sites.

This paper proposes a new framework to measure cloud service reliability in three stages: defects characterization, workload characterization, and reliability modeling, as shown in Figure 1 and briefly described below:

- Defects characterization: Extract and process defects from the issue tracking system which contains all issues that the clients have reported or suggested to improve the service. Then classify each issue according to its validity. The result of this stage is weighted defects depending on their validity.
- Workload characterization: Extract service workload from published number of clients statistics. Then estimate the number of clients for each day to calculate the cumulative usage or workload.
- Reliability modeling: Use the weighted defects over clients usage to plot the defect profile. Then use SRGMs to assess current reliability and predict future reliability.

Details of the framework and its three stages, including individual steps in each stage, are described below.

3.2 Defects Characterization

An issues report system is offered by most cloud service providers for clients or developers who use the services. Since each record in the system is created by a client, it needs to be confirmed by a provider before it is considered as a defect. Therefore, we propose the defects characterization stage in Figure 1 consisting of:

1. Screening: Collect all records from issues report system, and exclude all records unrelated to defect observations or the target environment.
2. Classification: Categorize remaining issues to three types: valid, invalid, and uncertain.
3. Consolidation: Using historical data to weigh uncertain issues.

The screening step is to extract issues from the issue system that the provider offers to record all client issues, including system failures or any request to improve or add new functionality. The issue system covers all types of environments or programing languages that the provider uses to deliver the service. The screening step should exclude unrelated data such as enhancement requests or defects related to languages or environments outside the study scope. The result should be all defect issues in the specific services or the environment for the specific clients of a given study.

The classification step is to classify the output of the previous step to three categories: valid, invalid, and uncertain. This classification is not part of the data rather than a label we provide according to the provider team response to each issue. Valid issues are all records that are agreed as defects by the service provider. Invalid issues are all records that the provider disagreed as defects due to several reasons, e.g. it can be labelled as a duplicate, it works as intended, or it is obsolete because of a new release. The uncertain issues are all pending records that the provider did not categorize yet because it is a new record or awaiting clarification. The result of this step is a classification for all issues under the three classes we suggested.

The consolidation step counts all the valid defects but excludes all the invalid ones. Uncertain issues will be considered according to the history of the clients issues. Actually, every issue starts as a new status which is uncertain, then it ends eventually as a valid or an invalid issue. Therefore, we can use the historical data to estimate the likelihood of its validity. In other words, we use the valid to invalid ratio as a weight for uncertain issues using this formula $w = (\frac{V}{V+I})$, where V and I are the total number of valid and invalid issues respectively. In effect, the uncertain issues are partially counted as defects according to the weight.

3.3 Workload Characterization

Calendar time is not an accurate representative for workload in cloud service due to large usage variations. Also, the number of invocations for the service is not available for the clients. Therefore, we propose the workload characterization stage in

Figure 1 to use an alternative workload measure, the number of clients, which consists of the following steps:

1. Extraction: Find and extract a statistical information about clients count of the service.
2. Estimation: Estimate the daily clients count using a statistical model.
3. Accumulation: Calculate the cumulative clients count by the suggested formula.

The extraction step is to find alternative data sources for workload since the number of service invocations by end users or customers is not available for the clients. As a proxy of cloud service usage, we extract the number of clients from the service provider report or from a public report that include usage statistics about the desired service. In other words, the report should provide the number of clients, such as websites or mobile applications, that embedded the desired service in their system during the investigation period. The number of clients will be used as a proxy for workload in reliability models. The result of this step is the number of clients that accessed the service on some given days.

The estimation step concerns about the number of clients that use the cloud service in each day. If the report does not show clients access for every day, we propose to estimate this number by applying a statistical model or some other estimation methods on the available data. In effect, we use such models as interpolations to estimate the number of clients for every day by filling the gaps in the available data.

The accumulation step is to calculate the cumulative number of clients that embed the service in their software for the investigation period using the following equation:

$$U_i = \sum_{j=d_0}^{d_i} N_j \tag{4}$$

where U_i is the cumulative usage or cumulative clients count up to day d_i and N_j is the number of clients in day d_j that we estimated in the previous step.

3.4 Reliability Modeling

Most SRGMs require time between failures or the actual time instance or period that the software failed. In our framework, the time is represented by the sum of the number of clients that used the service each day until the day of the specific defect discovery. Therefore, the cumulative weighted defects over the cumulative clients count will be used to plot the defects behavior over time for cloud services. Then, we assess the current reliability and predict the future reliability using the suggested models in Section 2. Reliability modeling stage in Figure 1 consists of the following steps:

1. Defect behavior profile: Plot weighted defects over clients count to examine the defect profile.
2. Assessment: Use SRGMs to assess the current reliability of the service.
3. Prediction: Use SRGMs with partial data as training data set to predict future reliability.

The first step employs the results of the previous stages to plot the defect behavior profile. The result of defects characterization stage is the cumulative weighted defects in their arriving sequence. The result of workload characterization stage is the number of cumulative clients that subscribed to the service up to a given day. We link the weighted defects and cumulative clients count using the data. Plotting the defects that we extracted in the defect characterization stage over the cumulative clients count allows us to examine the shape and trend of the defect profile over time.

The assessment step uses the output of the previous step to assess the cloud service reliability using the selected models in Section 2. The defects behaviors profile over cumulative clients count will be quantitatively assessed using reliability models including Goel-Okumoto, S-shaped, and Musa-Okumoto SRGMs. The goodness of fit to the actual data will also be examined.

The prediction step will use the selected models to predict future defect behavior. The models will use 75% of the clients count and associated defect observations as training data to make reliability prediction into the future and to test the models' prediction accuracy.

4 Case Study

We chose Google Maps APIs as a case study since it is one of the mature, well developed, and widely used cloud services. The case study will demonstrate the applicability and effectiveness of our proposal approach.

4.1 Background and Data Availability

One of the earliest cloud services that provide geographic and location information is Google Maps APIs [13]. According to Google, "it is a collection of APIs that enable you to overlay own data on customized Google Maps". Frequently, Google is updating, adding, and terminating types and versions of APIs. However, each version has its own updates and it is backward compatible. Google introduced Google Maps as a website only in February 2005. In June 2005, Google Maps APIs was announced for public use. Nowadays, Google Maps APIs supports different environments using several programing languages. The three main environments are: Android for smart devices, iOS for Apple devices, and Java-Script for web browsers. This case study focuses on Java-Script Version 3 (JS3) since it is the most popular and well developed in contrast to the other APIs.

Google does not offer specific information about usage or number of subscribers on a daily basis. Therefore, we estimate the daily usage by using a web analytics site called BuiltWith.com that provides general information about various services. One of its services is a usage statistical reports for

thousand of web technologies including Google Maps APIs usage statistics[1].

4.2 Defect Characterization

This stage in the framework has three steps as shown in Figure 1. First, we extracted defects information. Google Maps APIs uses a web application system called gmaps-api-issues[2] for reporting and tracking all defect and enhancement requests by customers. We analyzed the collected data to exclude unrelated issues. We focused on the following fields in issue reports:

ID: Each issue has a sequence ID, which was given at the time of filling the report by a client.

Type: The issue type can be a defect or an enhancement. Defect type is an issue that causes a failure to the system or disables a functionality. Enhancement type is a request to add a new functionality. This latter issue type does not affect the reliability of the service, therefore it was excluded.

Status: This field has fifteen different categories. Each issue that was reported by a client will start as a "new" status. Then it will change from one to another status by Google's team until it is closed. Table 1 contains all status types and our explanation based on the team response.

API Type: It includes Java script, Java Script v2, Java Script v3, Android2, IosSDK, and other language that support different environments. This case study covers Java Script v3 only. Therefore, all other API types are excluded.

Time Open: When the report is issued or opened. This field will allow us to link the defect to the usage in reliability modeling.

The second step is classifying records according to defect validity to three categories: Valid, Invalid, and Uncertain. The classification process uses the status as indication for the provider treatment for each issue as shown in Table 1. Table 2 shows each class with its collection of statuses and the number of observations in the data.

The final step is to weigh each defect according to its classification. All valid issues were included, while invalid issues were excluded. We used the percentage of valid defect issue to invalid defect issue as a weight for each uncertain issue. According to Table 2, 524 issues are valid, while 1214 issues are invalid. The weight of uncertain issues is $\frac{524}{524+1214} = 0.30$. Therefore, uncertain issues were weighted by 0.30 corresponding to the ratio of valid issues in the past.

To exam our weighting process for uncertain defects, we followed up on all uncertain issues six months after the initial investigation period. 171 of the 346 uncertain issues have been resolved, where 45 issues became valid and 126 became invalid.

[1]Google Maps API usage statistics. Accessed on 09/27/2015 http://trends.builtwith.com/mapping/Google-Maps-API

[2]Google Maps API bug reports and feature requests. Accessed on 09/27/2015-https://code.google.com/p/gmaps-api-issues

Table 1: Google Maps APIs issue status

Status	Num	Explanation
New	116	just posted, no action taken
Accepted	60	team has accepted as bug
Acknowledged	3	team is aware of this issue
Cannot reproduce	172	team can't reproduce the same bug, so it is closed
Confirmed	27	team understands the bug and it remains open
Duplicate	220	team merged the bug with other ones and closed it
Fixed	433	team fixed the bug and closed it
Fixed not released	1	team fixed the bug and closed it but not released yet
Invalid	495	team sees no problem since there is a work around
Needs more info	160	team needs more details before it is confirmed
Obsolete	173	team sees that it is not an issue anymore because of new updates
Pending further review	70	team leaves it for future review
Post elsewhere	94	team sees it as not related to this type of API or it is a browser issue, so it is left open
Won't fix	15	team can't fix this bug due to unsupported browser or internal and external constraints
Working as intended	45	team sees it not as a bug and it works as it should be, so it was closed

Table 2: Issues classification & defect weight

Class	Status	Total	weight
Valid	accepted, acknowledged, confirmed, fixed, and fixed not released	524	1
Invalid	cannot reproduce, duplicate, invalid, obsolete, post else where, won't fix, and working as intended	1214	0
Uncertain	new, need more inof, and pending further review	346	0.30

The result shows that the ratio of invalid issues (73%) to valid issues (27%) is very close to our estimation (70% vs. 30%).

4.3 Workload Characterization

The next stage in our framework is workload characterization, which contains three steps. We extracted the number of clients that accessed Google Maps API JS3 in the first step. Since

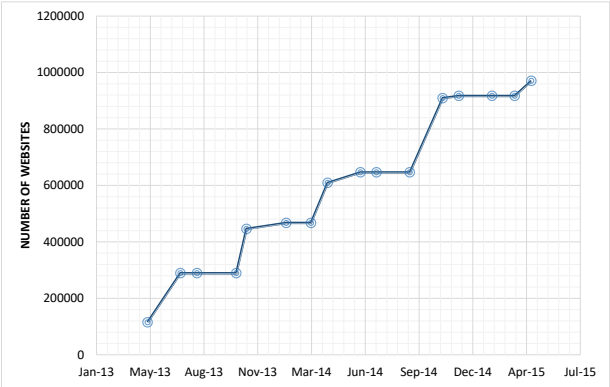


Figure 2: Number of websites using Google Maps APIs

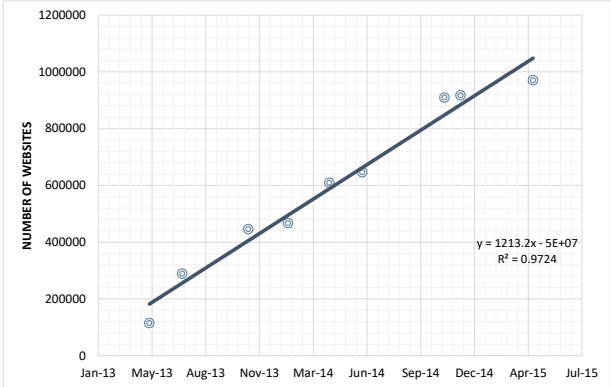


Figure 3: Regression line to estimate the number of websites using Google Maps API

API JS3 is offered for website environment, we need statistics about the number of subscribers to this API or the number of websites that embedded Google Maps in their pages during the investigation period. As stated earlier, BuiltWith.com provides statistical information about technology usage. One of their statistics is a report showing the number of websites using Google Maps APIs, as shown in Figure 2.

The second step is to estimate the clients daily usage for the service. Figure 2 shows the number of websites using Google Maps APIs from May 2013 to April 2015 with some repetitions. The repetitions appeared in several periods such as from July 2013 to October 2013 and from December 2014 to March 2015. These repetitions are likely caused by a lack of updates. Therefore, we only included first occurrence point and excluded each repeated ones. We observed a linear trend in the results, so we applied a linear regression to estimate the daily usage in Figure 3. This regression is fitted with high accuracy, where R^2 =0.97. Then, we used the regression to predict the number of websites in each day from May 2013 to April 2015.

Finally, we calculated the cumulative websites count up to each day according to Equation 4 from Section 3.3. The result is the sum of the number of the websites that had access to Google Maps API up to each day in the investigations period. The result is plotted in Figure 4.

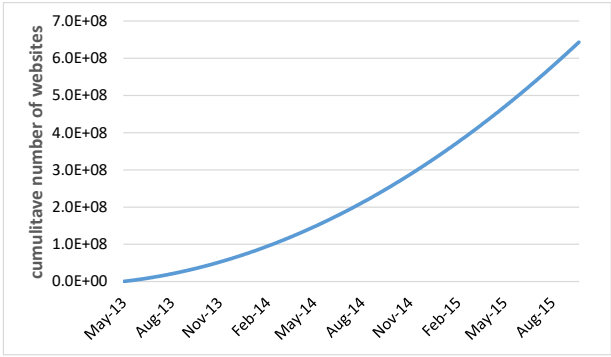


Figure 4: Cumulative websites count

4.4 Reliability Assessment

We begin with an examination of defect behavior by plotting the cumulative weighted defects over calendar time. Each defect will increase the y value according to its weight with respect to the arriving sequence. Figure 5 shows the defects behavior over calendar time which contains 428 records with a total of 255.1 cumulative defects from 1-May-2013 to 21-Sep-2015. There is no clear trend of reliability, which can be explained by the fast growth of usage as characterized by Figure 4. Therefore, alternative workload measurement instead of calendar time should be used in reliability modeling.

Before applying any reliability model, we used the results from both previous stages to plot defects behavior over time, plotting weighted defect over the cumulative websites count in Figure 5. Substituting calendar time with cumulative websites usage yields a stable plot that resembles a typical reliability growth curve. The data can be used now in SRGMs to assess and predict reliability of Google Maps API JS3.

Table 3 shows fitted models, estimated defects, failure rate, and R^2 value respectively. The estimated defects at the end of the investigation period $N(t)|_{t=T}$ are 454.98, 237.02, and 255.89 using GO, S-shaped, and MO models respectively, while the actual defects is 255.10. Also, we used the slope of the models at last data point $\lambda(t)|_{t=T}$ to estimate the failure rate at that point. It is clear from Figure 6 that the service is estimated to have a high reliability using both GO and MO models. Both models have high goodness of fit to the actual defects where R^2=0.996. S-shaped model has R^2 = 0.969, fitting the actual data less closely that MO and GO models. This is expected because

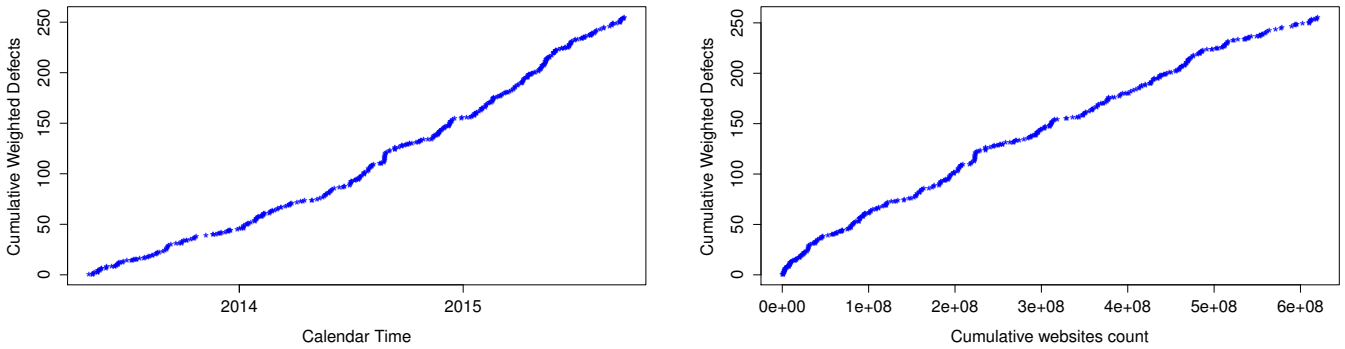


Figure 5: Weighted defects over time and over cumulative websites

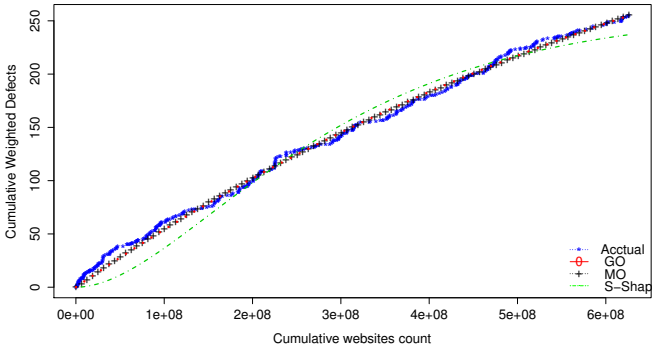


Figure 6: Reliability assessment using selected models

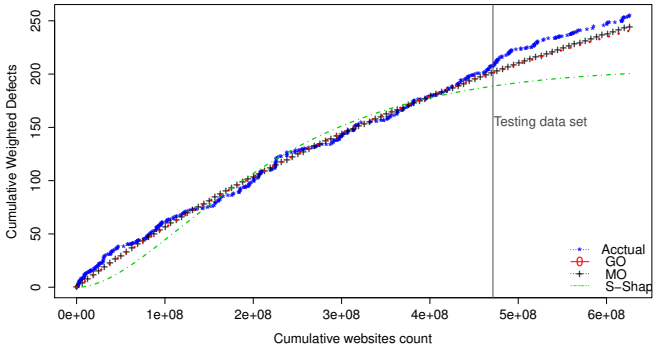


Figure 7: Reliability prediction using selected models fitted to partial data

the service has already passed the learning curve that S-shaped is considering. In other words, the investigation period does not cover the learning period since Google Maps APIs have been in the market for a long time.

4.5 Reliability Prediction

The prediction step uses 75% of websites count and associated defect observations as a training set to make predictions. Figure 7 shows the actual data with the three selected models with a vertical line to separate the training data set from the testing data set. The summary of prediction results are shown in Table 4. It includes the fitted model equations for each selected model, number of predicted defects in last data point, and the failure rate $\lambda(t)|_{t=T}$ of the model from the last data point. Also, we estimate the failure rate of the actual data by fit linear regression in the last ten data points and use the slope λ of this regression as the failure rate for actual data.

Figure 7 and Table 4 show that the models are underestimating the actual defects. Therefore, we investigated further and discovered that Google Maps APIs added sixteen new functions during our investigation period. Each newly added function can invite more failures to the system. Therefore, prediction in such cases would underestimate the actual defects.

5 Long Term Validation

To provide a long term validation of our approach, we continued monitoring the defect and usage trend for Google Maps APIs JS3 since September 2015, the ending date for our initial case study [3] described in the previous section. The models fitted to the entire period of the previous case study were extended to predict defects and reliability for the new period from September 2015 to August 2016. These predictions were compared to actual defect observations over time.

5.1 Defect Characterization

Google migrated issues data to a new system that contains all Google cloud service products issues. However, this new system did not change the concept or the policy of issues tracking system. But it did change the interface and some structures by renaming or adding fields. For instance, issue "Id" was restructured to keep it unique and to avoid repetition with other

Table 3: Assessment results

Models	Fitted models	$N(t)\|_{t=T}$	$\lambda(t)\|_{t=T}$	R^2
GO	$\mu(t) = 493.8(1-e^{-1.16E-09t})$	254.98	2.770E-07	0.996
S-shaped	$\mu(t) = 258.3(1-(1+6.6E-09t)e^{-6.6E-09t})$	237.02	1.130E-07	0.969
MO	$\mu(t) = 249.4\ln(1.7E-09t+1)$	255.89	1.464E-07	0.996
Actual		255.10	5.105E-07	

Table 4: Prediction results based on models fitted to partial data

Models	Fitted models	$N(t)\|_{t=T}$	$\lambda(t)\|_{t=T}$
GO	$\mu(t) = 384.3(1-e^{-1.57E-09t})$	240.79	2.257E-07
S-shaped	$\mu(t) = 206.3(1-(1+8.7E-09t)e^{-8.7E-09t})$	200.51	4.251E-08
MO	$\mu(t) = 189.8\ln(2.3E-09t+1)$	244.96	1.032E-07
Actual		255.10	5.105E-07

service issues. Also, "summary" field was renamed to "title". The most relevant change to the case study is the categories of status which changed according to Table 5. Actually, they moved all status data to a new field called "triaged" and reset the pending status to "new". Then, they used triaged field to extract the old statuses for the resolved ones. Also, they added a new status called "assigned" to distribute the responsibility of issues among Google team members.

We started the defect characterization stage by collecting all issues from the new Google issues tracker system for the new period and used the new field called "component" to filter the result by Google Maps APIs JS3 issues only. Then we classified the issues according to the new validity classes as shown in Table 5, providing a new mapping from individual statuses in the new system to our validity classes.

The new period we added included 378 issues. We included all valid issues (78), and excluded the invalid ones (238). For uncertain issues (62), we used the same weight (0.30) we used in the first period. The total of cumulative weighted defects is 109.

Table 5: Google Maps APIs JS3 issue status before and after migration with our classification

Old Status	New status	Class
New	New	Uncertain
Accepted	Accepted	Valid
Acknowledged	Accepted	Valid
Cannot reproduce	Won't fix (not reproducible)	Invalid
Confirmed	Accepted	Valid
Duplicate	Duplicate	Invalid
Fixed	Fixed	Valid
Fixed not released	Fixed	Valid
Invalid	Won't fix (infeasible)	Invalid
Needs more info	New	Invalid
Obsolete	Won't fix (obsolete)	Invalid
Pending further review	New	Invalid
Post elsewhere	Won't fix (infeasible)	Invalid
Won't fix	Won't fix (infeasible)	Invalid
Working as intended	Won't fix (intended behavior)	Invalid
Non	Assigned	Uncertain

5.2 Workload Characterization

We used the same statistical website (BuiltWith.com) to extract the usage statistics, with the result plotted in Figure 8. One noticeable difference between data in Figure 8 for the new observation period and that in Figure 4 is the lack of clear trend or pattern. Therefore, we used the latest observation data for each day between these points instead of polynomial or other nonlinear interpolations to avoid overfitting [2].

To build the cumulative usage, we used the last cumulative usage we reached in the previous period and added up the new estimated usage for each day after, with the result plotted in Figure 9.

5.3 Reliability Modeling

We extended the defect behavior profile for the previous case study. The defects for the new period were added to these from our initial investigation period, and plotted over calendar time in Figure 10. The defect behavior profile contains 607 records with a total of 364.8 cumulative defects. The same defects were also plotted in Figure 11 against the cumulative website count we obtained above.

Figure 10 shows that defect behavior over calendar time does not demonstrate a trend of reliability growth, while the same defects over cumulative website count has a clear concave shape signifying reliability growth as shown in Figure 11. These observations reconfirm the appropriateness of using the website count as a proxy for cloud service usage in our framework.

Then, we extended the models we fitted to the data from

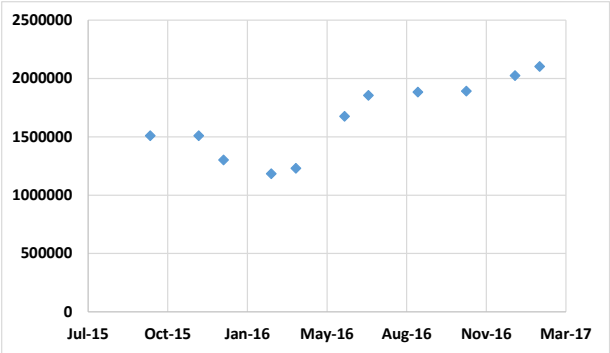


Figure 8: Google Maps APIs usage for the second period

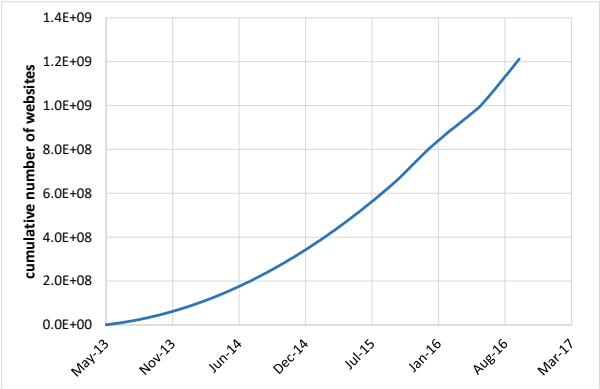


Figure 9: Cumulative website counts for both periods

the entire initial investigation period in Section 4.4 to the new period as shown in Figure 12. The result in Table 6 shows the fitted models, defect estimation, and slope. The estimated defects at the end $N(t)|_{t=T}$ are 366.57, 257.37, and 387.77 using GO, S-shaped, and MO models respectively, while the actual defects is 364.8. Also, we used the slope of the models at the last data point $\lambda(t)|_{t=T}$ to estimate the failure rate at that point and compared it with the liner regression slope of the last actual eight defects. It shows that reliability predictions based on GO model conformed well to the actual data. MO model also performed reasonably well.

In section 4.5, we discussed that the underestimation of the models using partial data from the initial period is due to adding new functions to the service in the last 25% part of that period. We investigated further and discovered that Google Maps APIs did not add any new functions during the new investigation period. Consequently, this led to accurate reliability predictions in the long term, without suffering the same underestimation problem.

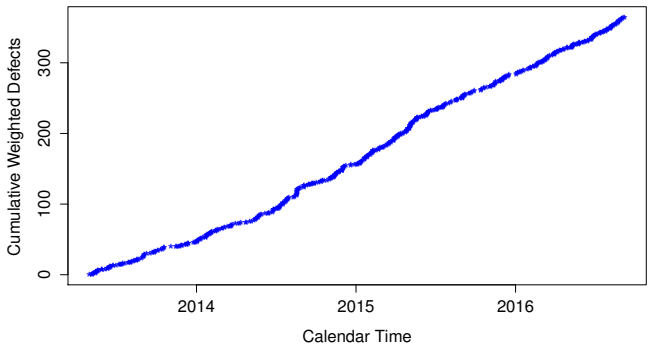


Figure 10: Defect behavior over calendar time

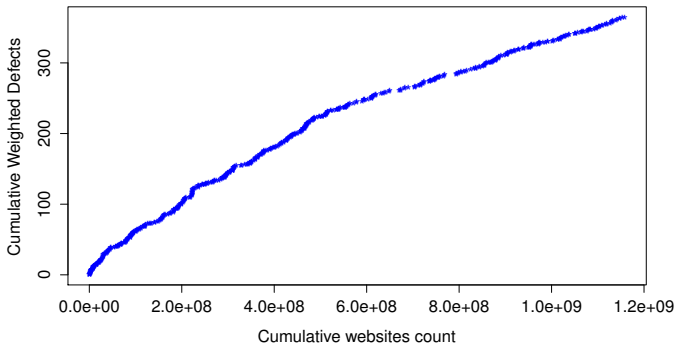


Figure 11: Defect behavior over website count

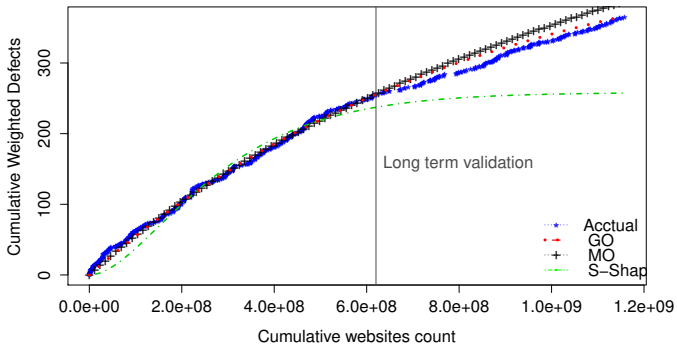


Figure 12: Long term prediction

To summarize, the reliability modeling results provide a long term validation of our approach. In particular, our method of defect characterization can be adapted to work effectively after the migration of issue report system for Google Maps APIs and changes to the detailed data fields. Our proxy for cloud service usage, the cumulative website count, can provide appropriate usage measurement for reliability modeling. Finally, the

Table 6: Long term validation results

Models	Fitted models	$N(t)\|_{t=T}$	$\lambda(t)\|_{t=T}$
GO	$\mu(t) = 493.8(1-e^{-1.16E-09t})$	366.57	1.479E-07
S-shaped	$\mu(t) = 258.3(1-(1+6.6E-09t)e^{-6.6E-09t})$	257.37	5.771E-09
MO	$\mu(t) = 249.4\ln(1.7E-09t+1)$	387.77	5.880E-07
Actual		364.8	2.052E-07

close fit between the model predictions and the actual defect observations provides evidence that our method will provide accurate long term predictions.

6 Conclusions and Perspectives

Reliability assessment and prediction for cloud services is a challenging problem. Clients and developers using these services in their systems or applications have limited access to data sources and appropriate measurements derived from these data sources. In this paper we developed a framework to measure cloud service reliability using alternative data sources. This framework consists of three stages: defects characterization, workload characterization, and reliability modeling. Defects characterization uses issue system reports as a defect data source to obtain weighted defects according to their validity after screening, classifying, and consolidating these reports. Workload characterization uses published statistical reports to obtain clients count as a new proxy to estimate usage time or workload and calculate the cumulative usage. Finally, the reliability modeling stage uses SRGMs to assess and predict cloud service reliability using the outcome of the previous stages.

Google Maps APIs was used as a case study to demonstrate the applicability and effectiveness of our new framework. The result of this case study shows a high accuracy for reliability assessment and prediction, which is an indication that weighted defects and cumulative client counts are appropriated substitutes for the unavailable data and related metrics for this environment.

Furthermore, we monitored the defect behavior over a longer period comparable to the initial period of our case study. The models fitted to the data from our initial case study were extended to provide long term reliability prediction. This prediction is compared to the actual data capturing the new defects and new workload after similar screening and data processing. We observed a close match between the prediction and actual data to provide a long term validation of the applicability and effectiveness of our approach.

To generalize our framework to measure other cloud services, we need to examine the similarities and differences in data availability and application environments before appropriate data sources and related metrics can be obtained and calculated. This framework and its followup improvements will offer a clear vision to the developers about the reliability of the cloud service before it is used or embedded in their applications. It also should offer a new method for providers to predict the future reliability of their services, which will help them take remedial and proactive actions to improve their services and to reduce cost.

Acknowledgment

This research was supported in part by NSF Grant #1126747, NSF Net-Cetric I/URC.

References

[1] M. O. Alannsary and J. Tian, "Measurement and Prediction of SaaS Reliability in the Cloud," *IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 123-130, August 2016.

[2] M. Baron, *Probability and Statistics for Computer Scientists, Second Edition.* Chapman & Hall/CRC, 2nd edition, 2013.

[3] A. Bokhary and J. Tian, "Measuring Cloud Service Reliability by Weighted Defects over the Number of Clients as a Proxy for Usage," *32nd International Conference on Computers and Their Applications (CATA2017),* pp. 63-70, 2017.

[4] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Transactions on Reliability*, R-28(3):206-211, 1979.

[5] M. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1995.

[6] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology, Gaithersburg, 2011.

[7] J. D. Musa, "A Theory of Software Reliability and its Application," *IEEE Transactions on Software Engineering*, SE-1(3):312-327, 1975.

[8] J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[9] E. Nelson, "Estimating Software Reliability from Test Data," *Microelectronics Reliability*, 17(1):67-73, 1978.

[10] J. Offutt, "Quality Attributes of Web Software Applications," *IEEE Software*, 19(2):25-32, 2002.

[11] Organización Internacional de Normalización, *ISO-IEC 25010: 2011 Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-System and Software Quality Models*. ISO, 2011.

[12] M. Silic, G. Delac, I. Krka, and S. Srbljic, "Scalable and Accurate Prediction of Availability of Atomic Web Services," *IEEE Transactions on Services Computing*, 7(2):252-264, 2014.

[13] G. Svennerberg, *Beginning Google Maps API 3*, Paul Manning, New York, 2010.

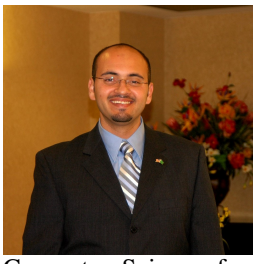
[14] J. Tian, "Reliability Measurement, Analysis, and Improvement for Large Software Systems," *Advances in Computers*, Elsevier, 46:159–235, 1998.

[15] J. Tian, S. Rudraraju, and Zhao Li, "Evaluating Web Software Reliability Based on Workload and Failure Data Extracted from Server Logs," *IEEE Transactions on Software Engineering*, 30(11):754–769, 2004.

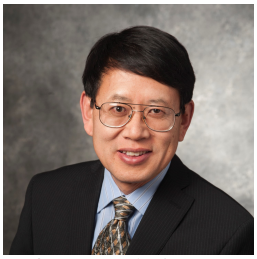
[16] S. Yamada, M. Ohba, and S. Osaki, "S-Shaped Reliability Growth Modeling for Software Error Detection," *IEEE Transactions on Reliability*, R-32(5):475–484, 1983.

[17] Z. Zheng and M. R. Lyu, "Collaborative Reliability Prediction of Service-oriented Systems," *32nd International Conference on Software Engineering*, ICSE'10, New York, NY, USA,1:35–44, 2010.

[18] Z. Zheng and M. R. Lyu, "Personalized Reliability Prediction of Web Services" *ACM Transactions on Software Engineering and Methodology*, 22(2):12:1–12:25, 2013.



Abdullah Bokhary is a Ph.D. Student in the Computer Science and Engineering Department of the Bobby B. Lyle School of Engineering at Southern Methodist University. He is also a faculty member at University of Jeddah, Jeddah, Saudi Arabia. Mr. Bokhary received a B.S. degree in Computer Science from King Abdulaziz University, Jeddah, Saudi Arabia 2001. He also earned a M.S. degree in Software Engineering and a M.S. degree in Engineering Management from Florida Institute of Technology in 2008. Mr. Bokhary worked between 2002 and 2005 as the data center manager in one of the divisions of the Hajj Ministry in Saudi Arabia. Mr. Bokharys research interests in cloud service reliability and usability.



Jeff (Jianhui) Tian received a B.S. degree in Electrical Engineering from Xi'an Jiaotong University in 1982, an M.S. degree in Engineering Science from Harvard University in 1986, and a Ph.D. degree in Computer Science from the University of Maryland in 1992. He worked for the IBM Software Solutions Toronto Laboratory between 1992 and 1995 as a software quality and process analyst. Since 1995, he has been with Southern Methodist University, Dallas, Texas, now as Professor of Computer Science and Engineering. Since 2012, he has also been a Shaanxi 100 Professor in the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. His current research interests include software quality, reliability, usability, testing, measurement, and applications in commercial, net-centric, web-based, service and cloud computing software and systems. He is a member of IEEE and ACM.

Computing Covers from Matchings with Permutations

Ariel Fernández*
Geographic Information Systems (GIS), Buenos Aires, ARGENTINA

Ryszard Janicki†
McMaster University, 1280 Main Street West, Hamilton, ON L8S 4L8, CANADA

Michael Soltys‡
California State University at Channel Islands, One University Drive, Camarillo, CA 93012, USA

Abstract

We present a matrix permutation algorithm for computing a minimal vertex cover from a maximal matching in a bipartite graph. Our algorithm is linear time and linear space, and provides an interesting perspective on a well known problem. Unlike most algorithms, it does not use the concept of alternating paths, and it is formulated entirely in terms of combinatorial operations on a binary matrix. The algorithm relies on permutations of rows and columns of a 0-1 matrix which encodes a bipartite graph together with its maximal matching. This problem has many important applications such as network switches which essentially compute maximal matchings between their incoming and outgoing ports.

Key Words: Minimal vertex cover; bipartite graph; maximal matching; König's Mini-Max theorem.

1 Introduction

In this paper we provide a new different solution to an old problem. The basic novelty is using matrix permutations (cf. [5]) instead of traditional graph-based techniques.

Suppose that we are given a bipartite graph $G = (V = V_1 \cup V_2, E)$, i.e., a graph where $V_1 \cap V_2 = \emptyset$ and $E \subseteq V_1 \times V_2$. Let A_G be the adjacency matrix of G, of size $|V_1| \times |V_2|$, and with 0-1 entries: $(i, j) \in E$ iff $(A_G)_{ij} = 1$. A matching M is a subset of E consisting of a "pairing" of the vertices of G in such a way that no two edges of M meet at the same vertex. We can represent a matching as a set of pairs of nodes of V, i.e., $M \subseteq E$, or as an adjacency matrix. A matching M is maximal if $|M|$ is as large as possible, i.e., when $|M|$ is maximum. We talk of bipartite graphs and their adjacency matrix representations interchangeably.

A vertex cover of a graph $G = (V, E)$ is a set of vertices $C \subseteq V$ such that each edge of the graph is incident to at least one vertex of the set C, i.e. for each $e = (v_1, v_2) \in E$, $\{v_1, v_2\} \cap C \neq \emptyset$. A vertex cover C is minimal if $|C|$ is minimum.

It is well known that given a general graph, a maximal matching can be computed with the classical Edmond's blossom algorithm ([10]) in $O(|V|^4)$ time, or the more complex $O(|V|^{\frac{1}{2}}|E|)$ algorithm by Micali and Vazirani [25]. For bipartite graphs, the easiest solution is to use the Ford-Fulkerson algorithm for flows [11] (c.f. [6]), either directly, or its modified version based on the concept of alternating paths, i.e., paths that alternate between edges that are in the matching and edges that are not in the matching (c.f. [1]), both run in $O(|V||E|)$ time; or we can use the more efficient (especially for sparse graphs) Hopcroft-Karp algorithm [16] which again runs in $O(|V|^{\frac{1}{2}}|E|)$, or, for dense graphs, the algorithm of [2] which runs in $O\left(|V|^{1.5}\sqrt{\frac{|E|}{\log|V|}}\right)$.

On the other hand, for general graphs, the problem of computing minimal vertex covers is **NP**-hard; in fact, it was one of Karp's 21 original **NP**-complete problems [17].

In 1916, in two nearly identical papers — one in German [20], the other in Hungarian [19] — König proved that every doubly stochastic matrix with non-negative entries must have a non-zero term in its determinant. In the same papers König also proved that every regular bipartite graph has a perfect matching. In the late 1950's Dulmagead and Mendelsohn published papers ([8, 9]) in which they worked out a canonical decomposition theory for bipartite graphs in terms of maximal matchings and minimal vertex covers.

For bipartite graphs, due to König's Mini-Max Theorem [19, 20], minimal vertex covers can be derived from maximal matchings in $O(|V| + |E|)$ time (c.f. [24, Lemma 3]). This means that for all algorithms known so far, the time complexity of computing a minimal vertex cover for bipartite graphs is the same as time complexity of computing an appropriate maximal matching. All widely known derivation methods use the idea of alternating paths (or equivalent concepts) (c.f. [14, 28]).

In this paper we use a different approach. Instead of traditional graph theory methods, we will use matrix permutation based techniques. The matrix permutation based techniques have recently become increasingly popular. They have been used for a variety of graph related problems including biology [3], trains scheduling in a railway traffic network [29] and clustering [21]. The basic advantage of permutation based methods is that, while 'big-Oh' complexity might be the same or slightly bigger than when graph based methods are used,

*Computing and Software. Email: agfern@gmail.com
†Computing and Software, Email: janicki@mcmaster.ca
‡Computer Science, Email: michael.soltys@csuci.edu

the computational overhead is usually much lower, and the implementation simpler.

We start with a matrix version of König's Mini-Max Theorem, instead of its more popular graph theory version. In fact we use graph theory terminology for readability only, as they are not really needed to present and implement our solution. All operations are simple matrix operations which can be implemented very efficiently.

Our algorithm is linear in both time and space with respect to the size of a binary matrix that represents a given *bipartite* graph. No assumptions are made about the method for computing maximal matchings.

We will also show a simple and very intuitive algorithm for a minimal vertex cover that runs in time $O(|V|^{\frac{3}{2}}|E|)$, and also illustrates well a fundamental difference between bipartite graphs and general graphs.

The paper is organized as follows. In Section 2 we present a problem formulation and König's Mini-Max Theorems. The basic concepts of our model are discussed in Section 3, while our main algorithm is presented and analyzed in Section 4. Another, more intuitive and natural, but slower, algorithm is discussed in Section 5. The last section, Section 6 contains conclusions.

This paper is a revised and extended version of the conference paper [12].

2 Problem Formulation and König's Mini-Max Theorems

Let $M_G = \mathsf{MA}(G)$ be the output of running an algorithm MA that computes a maximal matching for a given bipartite graph $G = (V_1 \cup V_2, E)$, i.e., M_G is the adjacency matrix of a maximal matching produced by the algorithm MA. MA could be Hopcroft-Karp algorithm [16], or any other algorithm of this type.

Let A_G be an adjacency matrix that defines the graph G, and let M_G denote the adjacency matrix for a maximal matching of G. Note that both A_G and M_G are of size $|V_1| \times |V_2|$, and thus at least four times smaller than a standard $|V| \times |V|$ representation. In what follows we will assume bipartite graphs to be represented by $|V_1| \times |V_2|$ adjacency matrices.

We find it useful to give two equivalent formulations of König's theorem. The first one is the standard formulation that uses the language of graphs.

Theorem 1 (König's Mini-Max version I)**.** *Given a bipartite graph G, if ρ_G is the size of the maximal matching of G and ρ'_G is the size of the minimal vertex cover of G, then $\rho_G = \rho'_G$.*

The proof of Theorem 1 (c.f. [5]) furnishes the basic ideas that will be used later in Section 3 to transform maximal matchings into minimal covers.

Given an $m \times n$ 0-1 matrix A, let S_A be a set of pairs of indices, i.e.,

$$S_A = \{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\} \subseteq \mathbb{N} \times \mathbb{N},$$

where $\mathbb{N}$ denote natural numbers and $A_{i_p j_p} = 1$ for every $p \in [k]$, and all the i_p's, as well as all the j_p's, are distinct. In other words, S_A is a set of positions in the matrix A containing 1s, and no two of those 1s are on the same row or the same column, i.e., no two of them are on the same *line*. Given A, the maximum possible size of such a set S_A is called the *term rank* of A ([5]). Notice that if A_G is the adjacency matrix of a bipartite graph G, then the term rank equals the size of a maximal matching in G.

On the other hand, given a 0-1 matrix A of size $m \times n$, a set C of lines, i.e., a collection of rows and columns of A, is called a *cover* if every 1 in A is in at least one row or column of C. Then, given a bipartite graph G, the size of the minimal vertex cover corresponds to the minimal cover of A_G.

We now restate König's theorem but using the language of matrices.

Theorem 2 (König's Mini-Max version II)**.** *Let A be a 0-1 matrix of size $m \times n$. The minimum number of lines in A that cover all of the 1s in A is equal to the maximum number of 1s in A, no two of the 1s on the same line.*

Since a bipartite graph G can be identified with its 0-1 matrix representation A_G of size $|V_1| \times |V_2|$, we may write $M_{A_G} = \mathsf{MA}(A_G)$ instead of $M_G = \mathsf{MA}(G)$.

Our goal is to design an algorithm, which we call PERALG, that takes an input $\langle A_G, M_{A_G} \rangle$ and produces a set of lines C_{A_G} that form a minimal cover of A_G. We want PERALG to compute C_{A_G} in $O(|A_G|)$, where $|A_G|$ is the size of the matrix A_G. We assume that the algorithm MA is given.

3 Preliminaries to our Algorithm

Our permutation-based algorithm, PERALG, runs in time $O(|V_1||V_2|)$. Since $|V_1||V_2| = |A_G|$, the size of the matrix representing G, PERALG runs in linear time in the size of $|A_G|$, assuming a model of computation (such as RAM) where the matrix entries can be accessed at cost $O(1)$.

The main idea behind PERALG is that, given a maximal matching M of a bipartite G, the minimal vertex cover C can be constructed by taking, for each $e \in M$, one of e's end point nodes. Of course, not all $2^{|M|}$ selections of end-points work, but at least one selection of end-points works. We show the details in Lemma 1.

We start with some terminology for denoting lines: given an $m \times n$ matrix A, we can denote the lines as $r_1, r_2, \ldots, r_m$ and $c_1, c_2, \ldots, c_n$, and the r's denote the rows and the c's denote the columns. It will also be advantageous to denote by $l^o_{(i,j)}$ a line going through entry i, j, where $o \in \{0, 1\}$, where $i \in [m]$ and $j \in [n]$, and

$$o = \begin{cases} 0 & l^o_{(i,j)} \text{ is vertical, i.e., } l^0_{(i,j)} = c_j \\ 1 & l^o_{(i,j)} \text{ is horizontal, i.e., } l^1_{(i,j)} = r_i \end{cases}.$$

A cover is a set of lines $C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}} = \{l^{o_1}_{(i_1,j_1)}, l^{o_2}_{(i_2,j_2)}, \ldots, l^{o_k}_{(i_k,j_k)}\}$, with *orientation* $\boldsymbol{o} = o_1 o_2 \ldots o_k$, and $\boldsymbol{i} = i_1, i_2, \ldots, i_k$, $\boldsymbol{j} = j_1, j_2, \ldots, j_k$, and it is such that any 1 in A is covered by (at least) one of these lines; i.e., if there is a 1 in position (i, j) of

the matrix A, then there exists a $p \in [k]$ such that $l^{o_p}_{(i_p,j_p)} \in C^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}_A$ and

$$[i = i_p \wedge o_p = 0] \vee [j = j_p \wedge o_p = 1].$$

If M_A is a maximal matching, the Mini-Max theorem says that there exist $\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}$ of length $k = |M_A|$ such that $C^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}_A$ is a cover. Recall that M_A represents a maximal matching as a 0-1 matrix, and that a 1 in position (i, j) means that (i, j) is an edge in the matching (i.e., $i \in V_1$ and $j \in V_2$ are "paired"). But in terms of "matrix combinatorics" this means that the 1s in M_A are positioned in such a way that no two 1s are on the same line (vertical or horizontal). Thus, we know that $C^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}_A$ must have lines through all the 1s of M_A; further, any such line cannot cover more than a single 1. Since we know that the size of $C^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}_A$ is the size of M_A, each 1 of M_A claims exactly one line. Hence $\boldsymbol{i},\boldsymbol{j}$ are directly determined by M_A, but $\boldsymbol{o}$ needs to be computed.

The result below shows the relationship between maximal matchings and minimal covers expressed using the notation described above.

Lemma 1. *Suppose that $G = (V = V_1 \cup V_2, E)$ is a bipartite graph, A its adjacency matrix, and M_A a maximal matching. Suppose*

$$M_A = \{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\},$$

i.e., M_A is a list of all the positions of M_A with a 1 in them ($k = |M_A|$). Then, it must be the case that

$$C^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}_A = \{l^{o_1}_{(i_1,j_1)}, l^{o_2}_{(i_2,j_2)}, \ldots, l^{o_k}_{(i_k,j_k)}\}$$

is a minimal cover for some $\boldsymbol{o} \in \{0,1\}^k$.

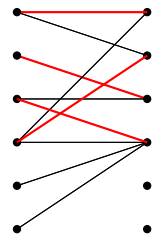
Proof. We know that for all $p \in [k]$, $A_{(i_p j_p)} = 1$, and so our cover must contain, for every $p \in [k]$, either r_{i_p} or c_{j_p}. By the Mini-Max theorem, there is a cover of size k, and so, by the pigeonhole principle, we can say something stronger: our cover *must consist*, for every $p \in [k]$, of either r_{i_p} or c_{j_p}. But that is the same as saying that our cover *must consist*, for every $p \in [k]$, of $l^{o_p}_{(i_p,j_p)}$, for $o_p = 0$ or $o_p = 1$. The lemma follows from that. □

Given permutations $\pi : [m] \longrightarrow [m]$ and $\tau : [n] \longrightarrow [n]$, let P_π and Q_τ be the corresponding permutation matrices. The matrices P_π and Q_τ are obtained from the identity matrix by exchanging the rows according to π and τ, respectively. Then: $(P_\pi M_A Q_\tau)_{ij} = (M_A)_{\pi^{-1}(i)\tau^{-1}(j)}$.

Given an $m \times n$ matrix A, and given a maximal matching M_A, which we represent as $M_A = \{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$, where $i_1 < i_2 < \cdots < i_k$, then the pair of permutations:

$$\begin{array}{ccc} \pi & & \\ i_1 & \mapsto & 1 \\ i_2 & \mapsto & 2 \\ & \vdots & \\ i_k & \mapsto & k \end{array} \qquad \begin{array}{ccc} \tau & & \\ j_1 & \mapsto & 1 \\ j_2 & \mapsto & 2 \\ & \vdots & \\ j_k & \mapsto & k \end{array}$$

are *order preserving permutations according to rows (for M_A)*. Note that the indices that are not specified are left fixed by π, τ.



(a) Graph G

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

(b) $A = A_G$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(c) $M_A = M_{A_G}$

$$\begin{bmatrix} \mathbf{1} & 0 & 0 & 1 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 1 & \mathbf{1} & 0 & 0 & 0 \\ 1 & 0 & 1 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

(d) $P_\pi A_G Q_\tau$

Figure 1: An example of calculating $P_\pi A Q_\tau$

That is, $P_\pi M_A Q_\tau$ place the 1s on the main diagonal, in the original order of the rows of M_A. Notice also that:

$$P_\pi A Q_\tau = \left[\begin{array}{c|c} T & A_1 \\ \hline A_2 & 0_{(m-k)\times(n-k)} \end{array}\right] = \left[\begin{array}{ccc|c} 1 & & & \\ & 1 & * & \\ & * & \ddots & A_1 \\ & & 1 & \\ \hline & & & \\ & A_2 & & 0 \\ & & & \end{array}\right] \tag{1}$$

That is, the 1s in M_A are permuted to be on the diagonal of the upper-left $k \times k$ quadrant; call this quadrant T. The first thing to observe is that the lower-right $(m-k) \times (n-k)$ quadrant consists entirely of zeros. This assertion is a consequence of the Min-Max theorem: all the lines in $C^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}_A$ pass through a 1 in T; none of these lines can possibly touch this lower-right quadrant, so it must be full of zeros.

For example, suppose that we have a graph G as in Figure 1; let us examine the values of M_A and $P_\pi A Q_\tau$.

In this case, π is the identity permutation, and τ fixes 1, moves 2 to 4, 3 to 2, 4 to 3, and fixes 5 and 6. The lines in red in Figure 1(a) indicate a maximal matching; the red ones in Figure 1(d) indicate the corresponding lines, now placed on diagonal. Note that the 2×2 lower-right submatrix is zero as it should.

The next Lemma relates the covering of the original A to the covering of permuted A, i.e., $P_\pi A Q_\tau$.

Lemma 2. *Suppose that π, τ are order preserving permutations according to rows. Then, if*

$$C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}} = \{l^{o_1}_{(i_1,j_1)}, l^{o_2}_{(i_2,j_2)}, \ldots, l^{o_k}_{(i_k,j_k)}\}$$

is a covering of A, then

$$C^{\boldsymbol{o},\pi(\boldsymbol{i}),\tau(\boldsymbol{j})}_{P_\pi A Q_\tau} = \{l^{o_1}_{(\pi(i_1),\tau(j_1))}, l^{o_2}_{(\pi(i_2),\tau(j_2))}, \ldots, l^{o_k}_{(\pi(i_k),\tau(j_k))}\}$$

is a covering of $P_\pi A Q_\tau$.

Proof. Suppose that $C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}} = \{l^{o_1}_{(i_1,j_1)}, l^{o_2}_{(i_2,j_2)}, \ldots, l^{o_k}_{(i_k,j_k)}\}$ is indeed a covering of A. Consider any entry (p,q) of $P_\pi A Q_\tau$, i.e., $(P_\pi A Q_\tau)_{pq} = A_{\pi^{-1}(p)\tau^{-1}(q)}$. If $A_{\pi^{-1}(p)\tau^{-1}(q)} = 1$, then either $r_{\pi^{-1}(p)} \in C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}$ or $c_{\tau^{-1}(q)} \in C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}$. This last statement means that there exists an $a \in [k]$ such that at least one of the following two statements is true:

- $l^{o_a}_{(i_a,j_a)} \in C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}$ where $i_a = \pi^{-1}(p) \wedge o_a = 1$, or
- $l^{o_a}_{(i_a,j_a)} \in C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}$ where $j_a = \tau^{-1}(q) \wedge o_a = 0$,

which in turn means that at least one of the following is true

- $l^{o_a}_{(\pi(i_a),\tau(j_a))} \in C^{\boldsymbol{o},\pi(\boldsymbol{i}),\tau(\boldsymbol{j})}_{P_\pi A Q_\tau}$ where $\pi(i_a) = \pi(\pi^{-1}(p)) \wedge o_a = 1$, or
- $l^{o_a}_{(\pi(i_a),\tau(j_a))} \in C^{\boldsymbol{o},\pi(\boldsymbol{i}),\tau(\boldsymbol{j})}_{P_\pi A Q_\tau}$ where $\tau(j_a) = \tau(\tau^{-1}(q)) \wedge o_a = 0$,

and as π, τ are permutations, they are bijections, and so $\pi(\pi^{-1}(p)) = p$ and $\tau(\tau^{-1}(q)) = q$, and so restating once again we obtain:

- $l^1_{(p,\tau(j_a))} \in C^{\boldsymbol{o},\pi(\boldsymbol{i}),\tau(\boldsymbol{j})}_{P_\pi A Q_\tau}$, or
- $l^0_{(\pi(i_a),q)} \in C^{\boldsymbol{o},\pi(\boldsymbol{i}),\tau(\boldsymbol{j})}_{P_\pi A Q_\tau}$.

In either case, this means that there is a line covering entry (p,q) of $P_\pi A Q_\tau$ if that entry is a 1. Hence, $C^{\boldsymbol{o},\pi(\boldsymbol{i}),\tau(\boldsymbol{j})}_{P_\pi A Q_\tau}$ is indeed a covering for $P_\pi A Q_\tau$. □

The purpose of Lemma 2 is to show that given A_G, we can reorder its rows and columns at will — which corresponds to a relabelling of V_1 and V_2 — and the resulting matrix has a maximal matching and minimal vertex cover of the same size. Furthermore, we can easily compute the maximal matching and vertex cover for the resulting matrix from the original one. Note that we assumed in Lemma 2 that the permutations are order preserving permutations (according to rows), and hence the orientation vector $\boldsymbol{o}$ is not affected. If the permutations are not order preserving, then we can still recompute the maximal matching and minimal vertex cover, but we must apply the corresponding permutation to the orientations. This is summarized in Corollary 1 below.

Corollary 1. *Suppose that π, τ are order preserving permutations according to rows, and that the diagonal 1s have been reordered by μ. Then, if $C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}$ is a covering of A, then*

$$C^{\mu(\boldsymbol{o}),\pi(\boldsymbol{i}),\tau(\boldsymbol{j})}_{R_\mu P_\pi A Q_\tau R_\mu} = \{l^{o_{\mu(1)}}_{(\mu(\pi(i_1)),\mu(\tau(j_1)))}, l^{o_{\mu(2)}}_{(\mu(\pi(i_2)),\mu(\tau(j_2)))}, \ldots, l^{o_{\mu(k)}}_{(\mu(\pi(i_k)),\mu(\tau(j_k)))}\}, \quad (2)$$

is a covering of $R_\mu P_\pi A Q_\tau R_\mu$.

4 Our Algorithm

On input $\langle A, M_A \rangle$, where M_A is a maximal matching for A, PERALG computes a minimal cover $C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}}$. More precisely, as was shown in Lemma 1, given M_A we know *a priori* that:

$$C_A^{\boldsymbol{o},\boldsymbol{i},\boldsymbol{j}} = \{l^{o_1}_{(i_1,j_1)}, l^{o_2}_{(i_2,j_2)}, \ldots, l^{o_k}_{(i_k,j_k)}\},$$

is a minimal covering for some $\boldsymbol{o}$, where the (i_p, j_p) are the non-zero entries of M_A. Hence, all that we need to compute in our algorithm is the orientation vector $\boldsymbol{o} = o_1 o_2 \ldots o_k$.

The analogy in the graph theoretic setting is the following: given a bipartite graph G and a maximal matching M the minimal vertex cover can be selected from M. This selection takes place by choosing for each edge $e \in M$, one of its end-points; a particular choice of end-points corresponds to a particular orientation. We now present PERALG for computing the orientations.

PERALG:

Input: A, M_A, $m \times n$ 0-1 matrices, where M_A is a maximal matching for A:

Step 1 If $k = |M_A| = \min\{m,n\}$, then

- $\{r_1, r_2, \ldots, r_m\}$ is a cover if $m \leq n$; i.e., return $\boldsymbol{o} = 1^m$ and exit
- $\{c_1, c_2, \ldots, c_n\}$ is a cover if $m > n$; i.e., return $\boldsymbol{o} = 0^n$ and exit

Step 2 Else, $k = |M_A| < \min\{m,n\}$, compute order preserving permutations π, τ that diagonalize M_A, so (recall the equation (1))

$$P_\pi A Q_\tau = \left[\begin{array}{c|c} T & A_1 \\ \hline A_2 & 0_{(m-k)\times(n-k)} \end{array}\right]$$

Step 2a If $A_1 = 0 \vee A_2 = 0$:

- If $A_1 = 0$ then return $\boldsymbol{o} = 0^k$ and exit
- If $A_2 = 0$ then return $\boldsymbol{o} = 1^k$ and exit

Step 2b Else, $A_1 \neq 0 \wedge A_2 \neq 0$. Group the 1s on the diagonal of T into two sets, the black and the green, of sizes k_1 and k_2, respectively, with $k = k_1 + k_2$. A black 1 in position (i,i) has the property that both row i of A_1 and column i of A_2 consist of zeros. The green 1s do not have this property. For each green 1 in position (j,j):

- If there is a 1 in row j of A_1, then we let $o_j = 1$.
- Else, we let $o_j = 0$.

This part is illustrated in Figure 2.

Let T' be T where the 1s under the lines covering the green 1s have been zeroed out (see Figure 3). In order to compute the orientations of the black 1s, repeat recursively Step 2a on T'.

Output orientations $\boldsymbol{o}$

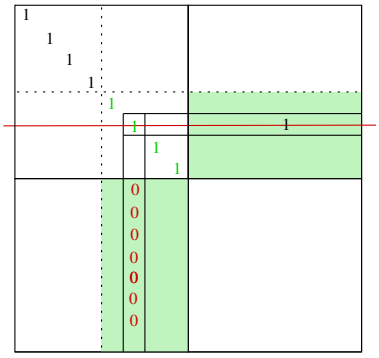


Figure 2: Step 2b

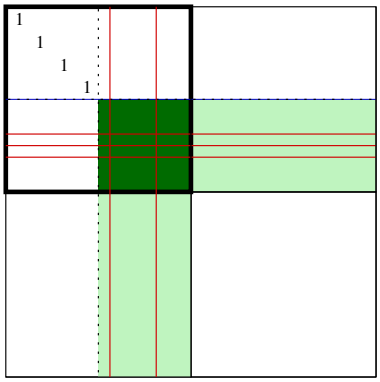


Figure 3: Repeat Step 2 with T', which is the upper-left quadrant, emphasized with a thicker border, with the "green square" zeroed out, as well as the entries under the red lines, arising from the cover of the "green square," zeroed out

Note that the situation, as represented in Figure 2, is simplified for the sake of clarity: the black 1s and the green 1s are depicted as two separate groups, but in general they are interspersed. We could block them together to be as in Figure 2, but that would require in general a permutation that is not order preserving; this could still be done by Corollary 1, but it introduces a technical overhead, as the orientations would no longer match (but we can always recover the original orientations by inverting the permutation).

Also note that in Step 2b, under the assumption $A_1 \neq 0 \wedge A_2 \neq 0$, we know that $k_2 > 0$, so we know that not all 1s in the upper-left quadrant are black; but it may well be the case that none are black, i.e., all the 1s are green, which would correspond to $k = k_2$.

Conceptually, the algorithm is rather simple. The technical complication is the permutations. We are computing the orientation of a covering for a permuted version of A; then, we must "extract" the correct orientation for the original version of A. This is what introduces a certain technical overhead. On the other hand, these permutations help to maintain a simple data structure (reconfigurations of the $|V_1| \times |V_2|$ matrix) that is essential for the computation.

We will now show that the algorithm PERALG really computes an appropriate vertex cover.

Lemma 3. *Algorithm* PERALG *is correct. That is, given A and M_A as input, it computes $\mathbf{o}$ so that $C_A^{\mathbf{o},i,j}$ is a vertex cover (of size $|M_A|$).*

Proof. By placing 1s on the diagonal of T, we ensure that each such 1 requires at least one line to be covered. By Lemma 1 we can conclude that exactly one line per 1 on the diagonal of T is sufficient to ensure a cover.

In Step 2b, if there is a 1 in row j of A_1, then we let $o_j = 1$, and otherwise we let $o_j = 0$. We know that this works because each 1 in T claims exactly one line in the cover. So it follows that it is not possible for both row j of A_1 to have a 1, and column j of the A_2 to have a 1, since that would require two lines through (j, j).

Further, the square that encloses the green 1s must be successfully covered by the above scheme: we have no choice as to the orientation of the lines covering the green 1s, and by the Min-Max theorem a successful covering exists, and thus the covering imposed by the green portions of A_1 and A_2 must necessarily work for the square enclosing the green 1s. □

In the following lemma, we show how to compute order preserving π, τ in Step 2. It is clear that it can be done in linear space, that is, in space $O(|A|)$. Except for the computation of order preserving π, τ once in Step 2, the recursive computation of the orientations is done inside A, with constantly many registers indexing A (space $O(\log(|A|))$) and hence also in space $O(|A|)$. Thus, PERALG requires linear space.

Lemma 4. *Algorithm* PERALG *runs in time $|A| = m \times n$.*

Proof. We show first the details of computing order preserving π, τ in Step 2. We initialize $r = 1$ and $q = 1$, and we also initialize two integer arrays i, j of size n. For every $p \in [n]$, if row p of M_A is not zero, we let $i[q] = p$, and let $q = q + 1$. On the other hand, if row p of M_A is zero, we let $j[r] = p$, and let $r = r + 1$.

We construct π from the two arrays i and j which encode the following mapping:

$$\begin{aligned} &i[1] \mapsto 1; \\ &i[2] \mapsto 2; \\ &\vdots \\ &i[k] \mapsto k; \\ &j[1] \mapsto k+1; \\ &j[2] \mapsto k+2; \\ &\vdots \\ &j[n-k] \mapsto n \end{aligned} \tag{3}$$

From π we construct P, where P has 1s in positions:

$$\begin{aligned}&(1,i[1]),(2,i[2]),\ldots,(k,i[k]),\\&(k+1,j[1]),(k+2,j[2]),\ldots,(n,j[n-k]),\end{aligned}\tag{4}$$

and zeros everywhere else. The permutation matrix Q is constructed in a similar manner from τ. This can be clearly done in time and space proportional to $|M_A|$, i.e., in linear time and space.

Once we obtain $P_\pi AQ_\tau$, we work, recursively, with this matrix, starting at each level of the recursion in Step 2a, in order to compute the orientations of the black 1s. As was mentioned above, if there are no green 1s, then the procedure terminates (outputting all horizontal or vertical orientations, according to which one of A_1 or A_2 is all zero). Thus, if $k_2 = 0$, we are done.

Otherwise, $k_2 > 0$, and the number of black 1s decreases by at least 1. Thus this loop, in the worst case, can repeat at most $k = |M_A| \leq \min\{m,n\}$ many times. Note therefore that if there are many green 1s, the procedure has fewer recursive calls; if there are few green 1s, the procedure has more recursive calls.

We make this argument a little bit more precise; let $R(n)$ be the maximum number of steps, in the worst-case, that our procedure takes on a matrix of size n (let n denote here $\max\{m,n\}$). Let a "step" be a single "atomic operation" which for us is one of the following two: check what is the value in position (i,j) of some matrix, and change the value in position (i,j) of some matrix to 0 or 1.

Suppose that k is the number of black 1's, and $n-k$ is the number of green 1's. Then, we can see that the worst-case analysis yields the following bound on the number of atomic steps:

$$R(n+1) = \max_{1<k<n}\{k(n+1-k)+R(k)\}.\tag{5}$$

First note that there must be at least one black 1, for otherwise, there are no more steps. There must also be at least two green 1's; if there were only one green 1, then either A_1 or A_2 would be all zero, which would also terminate the algorithm. Finally, no green 1's would imply termination as well. Hence the maximum is computed over $1 < k < n$ for size $n+1$.

Note that in the recursive equation (5), k represents the number of black 1's, and so the corresponding sizes of A_1 and A_2 are given by $k \times (n+1-k)$ and $(n+1-k) \times k$, and hence the term $k(n+1-k)$ — we are ignoring constants in (5); it should really be $2k(n+1-k)$, but this does not change the order of $R(n)$. The recursive step is repeated on the black 1's, and hence we add $R(k)$ in (5).

We now show by induction that $R(n) \leq n^2$, where R is initialized with $R(0) = R(1) = 1$. For $k < n+1$, by inductive assumption $R(k) < k^2$, and so by (5):

$$\begin{aligned}R(n+1) &\leq \max_{1<k<n}\{k(n+1-k)+k^2\} = \max_{1<k<n}\{kn+k\}\\&\leq n^2+n \leq n^2+2n+1 = (n+1)^2.\end{aligned}$$

Finally, we show how PERALG keeps track of the orientations $\boldsymbol{o}$ in each step of the recursive procedure. As was pointed out in the note following the presentation of PERALG, the situation, as represented in Figure 2, is simplified for the sake of clarity: the black 1s and the green 1s are depicted as two separate groups, but in general they are interspersed. This means that some orientations are computed in a given step, and some are not: $\boldsymbol{o} = o_1 o_2 \ldots o_k$, where $o_{i_1} o_{i_2} \ldots o_{i_{k_1}}$ correspond to the black 1s and are not yet computed, and $o_{j_1} o_{j_2} \ldots o_{j_{k_2}}$ correspond to the green 1s, and have just been computed. Note that the o_{i_p}'s and o_{j_q}'s are interspersed in $\boldsymbol{o}$, and $k = k_1 + k_2$. But it is easy to keep track of this, because the order is preserved throughout: let $\boldsymbol{o}$ be a string of 0s, 1s, and unset values. The unset values always appear in the same order as the black 1s on the diagonal to be dealt with in the next step. □

From Lemma 3 and Lemma 4 we get the main result of the paper.

Theorem 3. *Given a bipartite graph $G = (V = V_1 \cup V_2, E)$, and a maximal matching M_G,* PERALG *runs in time and space $O(|A_G|)$ to compute a minimal cover C_G. That is,* PERALG *runs in linear time and space to compute a minimal cover from a maximal matching.*

The time complexity of graph based algorithm for computing a minimal cover from a maximal matching for a bipartite graph $G = (V,E)$ is $O(|V|+|E|)$ (c.f. [24]) and clearly $O(|V|+|E|) = O(|A_G|)$, however, when it comes to real time complexity, $|A_G| \leq 4(|V|+|E|)$, and the overhead of our algorithm is minimal.

If the time complexity of $\mathsf{MA}(G)$ is $O(f_{\mathsf{MA}}(G))$ and $|E|^2 \leq O(f_{\mathsf{MA}}(G))$ (all known algorithms satisfy this and most likely always will), then the overall time complexity of finding a minimal vertex cover in a given bipartite graph G using our method is $O(f_{\mathsf{MA}}(G))$.

As was mentioned in the introduction, it is well known that given a general graph, a maximal matching can be computed with the classical Edmond's blossom algorithm ([10]) in $O(|V|^4)$ time, or the more complex $O(|V|^{\frac{1}{2}}|E|)$ algorithm by Micali and Vazirani [25]. As our transformation is linear, $O(|V|^2)$, it can be performed at a lesser cost than any algorithm currently on the market.

5 Another Simple Algorithm for Bipartite Graphs

As we have indicated in the Introduction, there are many polynomial time algorithms for finding maximal matching for both general and bipartite graphs. For example, a popular Hopcroft-Karp algorithm computes a maximal matching for a given bipartite graph in time $O(|V|^{\frac{1}{2}}|E|)$, where $G = (V,E)$ and $V = V_1 \cup V_2$. By König's Mini-Max theorem, we know that a bipartite graph G has a maximal matching of size k if and only if it has a minimal vertex cover of size k. Putting these elements together, we obtain a simple and natural algorithm NATALG, presented in Figure 4 for computing minimal vertex covers in a bipartite graph.

NATALG:
Input $G = (V = V_1 \cup V_2, E)$
$C \leftarrow \emptyset$
$k \leftarrow |\mathsf{MA}(G)|$ (= *size of minimal vertex cover of* G)
For every node $u \in V = V_1 \cup V_2$ do:
  If $u \in V_i$, then
    Let $G' = (V', E')$ be derived from G by:
      adding two new nodes u'_1, u'_2 to V_{3-i} to obtain V'
      adding two new edges $(u, u'_1), (u, u'_2)$ to obtain E'
  $k' \leftarrow |\mathsf{MA}(G' = (V', E'))|$
  If $k = k'$, then
    add u to C
    delete from G all edges incident to u
    delete all singleton nodes
    $k \leftarrow k - 1$
Output C

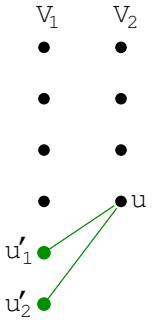


Figure 4: On input $G = (V = V_1 \cup V_2, E)$, NATALG repeatedly invokes an algorithm computing $|\mathsf{MA}(G)|$ for a bipartite graph G, (for example the Hopcroft-Karp algorithm

Lemma 5. *The algorithm* NATALG*, presented in Figure 4, computes a minimal vertex cover in time* $O(f_{\mathsf{MA}}(G)|V|)$*, where the time complexity of* $|\mathsf{MA}(G)|$ *is* $O(f_{\mathsf{MA}}(G))$.

Proof. If we add two new edges (u, u'_1) and (u, u'_2) to G — and obtain G' — the "cheapest" way to cover those two new edges is with their common vertex u. That is, by adding the edges $(u, u'_1), (u, u'_2)$, we force u to be part of a minimal cover of the resulting graph. If there was a cover of G that included u, then the same cover works for G'; essentially, we cover the two new edges "for free." This corresponds to the case $k = k'$, where we know that u was part of a cover, and so we add it to C, and we delete from G the edges incident to u. If, on the other hand, $k < k'$, then no minimal cover of G contained u, and thus we needed to add u to the cover of G' in order to take care of the two new edges; in this case we do not put u in C.

In either case, we delete the gadget, and repeat the procedure on the next unexamined node in $V = V_1 \cup V_2$. For added efficiency, if $k = k'$ then we delete all singleton nodes. We run the MA algorithm in each round, giving the stated running time bound of $O(|V|^{\frac{1}{2}}|E||V|)$. Note that it is immaterial in which order we examine the nodes of G; any ordering works. □

If Hoproft-Karp algorithm is used to compute $|\mathsf{MA}(G)|$ in NATALG, the time complexity is $O(|V|^{\frac{3}{2}}|E|)$.

Note that the reduction described in Lemma 5 would not work over general graphs (i.e., not necessarily bipartite). First, for the obvious technical reason that requires u'_1, u'_2 to be added to "the other vertex set," i.e., to V_{3-i} if $u \in V_i$, where $i = 1, 2$. But, more importantly, say that instead of the Hopcroft-Karp algorithm we invoke Edmond's algorithm that works over general graphs. Could we then modify somehow the algorithm in Figure 4 to make it work over general graphs? The answer is: "not in polytime, unless **P** = **NP**". The reduction given by the algorithm in Figure 4 relies deeply on the graph being bipartite.

6 Conclusion

PERALG is a matrix permutation based algorithm that runs in time $O(|V_1||V_2|)$ to transform a maximal matching of a bipartite graph into a minimal vertex cover. In particular, PERALG runs in linear time and space (as its input is a binary matrix of size $|V_1| \times |V_2|$), and using the properties in the famous König's Mini-Max theorem, it performs basic counting of zeros and ones to compute a minimal cover. Our algorithm is very simple, and does not employ the usual graph theoretic properties that are the foundation of classical algorithms in this area.

Our algorithm is one of many applications of König's Mini-Max theorem, which has also several equivalent formulations (cf. [18]): as *Menger's Theorem* [23], counting disjoint paths; as *Hall's Theorem* [15], giving necessary and sufficient conditions for the existence of a "system of distinct representatives" of a collection of sets; as *Dilworth's Theorem* [7], counting the number of disjoint chains in a poset. It has recently been shown in [13] that these different formulations are not only equivalent, but additionally this equivalence can be proven in weak fragments of arithmetic [27].

A survey of classical Mini-Max results can be found in [22].

There are many applications of the type of algorithms discussed in this paper. For example, [4] is a paper in a long tradition of studying switching routers, which effectively compute solutions (or approximate solutions) to the problem of matching incoming ports to outgoing ports.

Acknowledgment

This research was partially supported by NSERC Discovery Grant of Canada. Main parts of this work were done during the time that the first author was a Ph.D. student in the Department of Computing and Software, McMaster University, and the third author held a position in the same. We are grateful to the referees for a careful reading of this paper, and for suggesting thoughtful improvements.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1983.

[2] H. Alt, N. Blum, K. Mehlhorn, and M. Paul. *Computing a Maximum Cardinality Matching in a Bipartite Graph in Time* $O\left(n^{1.5}\sqrt{\frac{m}{\log n}}\right)$. *Information Processing Letters*, 37:92–99, 1991.

[3] A. T. Alexiou, M. M. Psiha, and P. M. Vlamos. *Combinatorial Permutation Based Algorithm for Representation of Closed RNA Secondary Structures. Bioinformation*, 7(2):91–95, 2011.

[4] Banerjee, Satyajit and Datta Chowdhury, Atish and Sinha, Koushik and Ghosh, Subhas Kumar. *Contention-Free Many-to-Many Communication Scheduling for High Performance Clusters.* In Natarajan, Raja and Ojo, Adegboyega, Distributed Computing and Internet Technology: 7th International Conference, ICDCIT 2011, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 6536:150–161, 2011.

[5] R. A. Brualdi and H. J. Ryser. *Combinatorial Matrix Theory.* Cambridge University Press, 1991.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms,* third edition. McGraw-Hill Book Company, 2009.

[7] R. P. Dilworth. *A Decomposition Theorem for Partially Ordered Sets. Annals of Mathematics*, 51(1):161–166, 1950.

[8] A. L. Dulmage and N. S. Mendelsohn. *Coverings of Bipartite Graphs. Canadian Journal of Mathematics*, 10:517–534, 1958.

[9] A. L. Dulmage and N. S. Mendelsohn. *Some Generalizations of the Problem of Distinct representatives. Canadian Journal of Mathematics*, 10:230–241, 1958.

[10] J. Edmonds. *Paths, Trees, and Flowers. Canadian Journal of Mathematics*, 17:449–467, 1965.

[11] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton Univ. Press, 1962.

[12] A. G. Fernández, R. Janicki and M. Soltys, *A Permutation-Based Algorithm for Computing Covers from Matchings. Proc. of CATA17 (32nd International Conference on Computers and Their Applications)*, Honolulu, Hawaii, USA, pp. 27–32, March 20–22, 2017.

[13] A. G. Fernández and M. Soltys. *Feasible Combinatorial Matrix Theory.* Krishnendu Chatterjee and Jirí Sgall, editors, *Mathematical Foundations of Computer Science 2013*, *Lecture Notes in Computer Science* Springer, 8087:777–788, 2013.

[14] F. Gavril. *Testing for Equality between Maximum Matching and Minimum Node Covering. Information Processing Letters*, 6(6):199–202, 1977.

[15] P. Hall. *On Representation of Subsets. Journal of London Mathematical Society*, 10:26–30, 1935.

[16] J. E. Hopcroft and R. M. Karp. *An* $n^{5/2}$ *Algorithm for Maximum Matchings in Bipartite Graphs. SIAM Journal on Computing*, 2(4):225–231, December 1973.

[17] R. M. Karp. *Reducibility Among Combinatorial Problems.* R. E. Miller and J. W. Thatcher, editors, Complexity of Computer Computations. Plenum Press, pp. 85–103, 1972.

[18] J. Kleinberg, É. Tardos. *Algorithm Design.* Pearson, 2006.

[19] D. König. *Gráfok és alkalmazásuk a determinánsok és a halmazok elméletére.* Matematikai és Természettudományi Értesítö, 34:104–119, 1916.

[20] D. König. *Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. Mathematische Annalen*, 77(4):453–465, 1916.

[21] I. Llatas, A. J. Quiroz, and J. M. Renóm, *A Fast Permutation-based Algorithm for Block Clustering. Test*, 9(2):397–418, 1997.

[22] L. Lovász and M. D. Plummer. *Matching Theory. Annals of Discrete Mathematics*, North-Holland, 1986.

[23] K. Menger. *Zur allgemeinen Kurventheorie. Fundamenta Mathematica*, 19:96–115, 1927.

[24] S. Mishra, V. Raman, S. Saurabh, S. Sikdar, and C. R. Subramnian. *The Complexity of König Subgraph Problems and Above-Guarantee Vertex Cover. Algorithmica*, 61(4):857–881, 2008.

[25] S. Micali and V. V. Vazirani. *An* $O(\sqrt{|V|}\cdot|E|)$ *Algorithm for Finding Maximum Matching in General Graphs.* 21st IEEE Symp. Foundations of Computer Science, pp. 17–27, October 1980.

[26] A. Schrijver. *Min-Max Relations for Directed Graphs.* Bonn Workshop on Combinatorial Optimization, *Ann. Discrete Math.*, North Holland, 16:261–280, 1982.

[27] M. Soltys, S. Cook. *The Proof Complexity of Linear Algebra. Annals of Pure and Applied Logic*, 130(1-3), 207–275, 2004.

[28] J. A. Storer. *An Introduction to Data Structures and Algorithms.* Progress in Computer Science and Applied Logic Series. Springer, 2001.

[29] T. J. J. Van den Boom, N. Weiss, W. Leune, R. M. P. Goverde, and B. De Schutter, *A Permutation-based Algorithm to Optimally Reschedule Trains in a Railway Traffic Network.* Proc. of the 18th World Congress The International Federation of Automatic Control, pp. 9537–9542, Milano, Italy, 2011.

Ariel Fernández holds a Ph.D. (2013) from McMaster University. His primary area of research is Proof Complexity and Algorithms, especially inspired by the subject of Combinatorial Matrix Theory, which combines Linear Algebra, Graph Theory, and Combinatorics, and has a rich algorithmic content. Recently he has become interested in Quantum Computing, and especially in the study of quantum concepts in Proof Complexity. Starting in 2013 he is working on Geographic Information Systems (GIS systems), and is currently the director of a maps making company called "Filcar SRL" located in Buenos Aires, Argentina.

Ryszard Janicki is a professor at McMaster University in the Department of Computing and Software. He received the M.Sc. degree in Applied Mathematics from the Warsaw University of Technology, Poland in 1975, and the Ph.D. and Habilitation in Computer Science from the Polish Academy of Sciences, Warsaw, Poland in 1977 and 1981 respectively. He taught computer science and mathematics at the Warsaw University of Technology, Poland in 1975-1984, Aalborg University, Denmark in 1984-86, before joining McMaster in 1986. He was a Visiting Scholar at University of Newcastle upon Tyne, U.K., in 1982 and a Visiting Professor at Bordeaux University, France, in 1994-95. He published more than 200 papers and co-authored a monograph. His research interests include concurrency theory, fundamentals of software engineering, ranking theory, abstract approximation, mereology and relational methods in computer science.

Michael Soltys is a faculty at California State University Channel Islands in the Department of Computer Science, where he is a full professor and chair of the department. He is also the director of IT Cybersecurity at Executek International. His research is in logic and algorithms, and he is especially interested in proofs of correctness. He is also working in the area of String Algorithms which involves combinatorial methods on finite words, and in Combinatorial Matrix Theory, which combines linear algebra, graph theory, and combinatorics, and has a rich algorithmic content. Recently he has become interested in Ranking Algorithms, and especially in the elegant Pairwise Comparisons Method. He is a member of the Centre for Combinatorics on Words and Applications (CCWA). For more information visit `soltys.cs.csuci.edu`.

A Software Development Environment for a Multi-chip Convolutional Network Accelerator

Tetsui Ohkubo, Mankit Sit, Hideharu Amano*
Keio University, 223-8522, JAPAN

Ryo Takata, Ryuichi Sakamoto, Masaaki Kondo†
The University of Tokyo, 113-8656, JAPAN

Abstract

A building block convolutional neural network accelerator consists of a host and multiple accelerator chips which can scale the performance by changing the number of stacked chips. In order to program the host and the accelerators, an integrated programming development environment called NAMACHA is proposed. It includes compilers for convolutional neural network accelerators and a system level simulator including inter-chip communication latency. On the simulator, the total application runs 4390x faster than that of the logic level simulation with 1.27% difference of clock cycle counts. The simulation results of implementing AlexNet, SIMD instructions provided in the accelerator improved the performance by 70% on average. It demonstrates that NAMACHA can be used for architectural exploration as well as development of practical software.

Key Words: Convolutional neural network; software develelopment kit; emulator.

1 Introduction

Image recognition with Convolutional Neural Networks (CNN) has been introduced in embedded systems for car electronics and robotics as well as data centers. SNACC[1][1] is a compact CNN accelerator targeted for embedded applications. It consists of four SIMD processors that individually support a dedicated instruction stream for CNN operations and provide 36 independently accessible tables for storing feature map, weight and temporal data. Moreover, SNACC provides ThruChip Interface (TCI)[9] for flexible chip stacking. By using TCI, multiple SNACC chips can be connected with each other through an escalator network formed by stacking the chips and also with an additional host processor for system coordination. Various trade-offs between performance and cost/power consumption can be chosen by changing the number of stacking chips. Such systems are called a building-block computation system[11]. A building-block computing system with SNACC chips is called the SNACC-Cube. It focuses on an image recognition for embedded systems. That is, the learning phase is assumed to be done in a cloud by a high performance system with multiple GPUs, and weights of the CNN have been preset. Compared with other ASIC examples[3][4][5][7], SNACC-Cube is energy efficient, flexible and scalable. It is developed with a low power Silicon-on-Thin BOX(SOTB) process[13] which works with an extremely low supply voltage. By making the use of wireless inductive TCI, the system size can be extended just by increasing the number of stacked chips.

One of the most important concerns about the building-block computation system including SNACC-Cube is its programming development environment, since the data transfer between the host and the chips must be taken into account when designing applications for this kind of system. In addition, it is crucial to accurately measure the communication latency between the chips because inter-chip communication can occupy a significant portion of the application execution time. Although Register transfer level (RTL) simulation based on hardware description language precisely simulates every single cycle of the hardware register state transitions, it is too slow and difficult to be used for most software developers.

To cope with the problem, we propose an integrated programming development environment NAMACHA. It includes a host programming environment, SNACC compiler, and a system level simulator. The programmer can develop programs for both host and SNACC using the evaluation results from a high-speed yet accurate system level simulation. The analysis results can be also used for improving the system architecture.

The rest of the paper is organized as follows. Section 2 introduces inductive coupling ThruChip Interface and building block computation systems with 3D chip stacking. Section 3 describes the brief introduction of SNACC, chip implementation and target multi-chip system SNACC-Cube. Then we introduce

*Faculty of Science and Technology, Email: snacc@am.ics.keio.ac.jp
†Graduate School of Information and Technology

[1]A comprehensive paper of SNACC will be published with real system evaluations.

NAMACHA in Section 4. Section 5 evaluates the performance and accuracy of the system level simulator. The implementation of AlexNet with NAMACHA is also presented. Section 6 mentions about related work and Section 7 concludes the paper.

2 Building Block Computing Systems

2.1 The First Example Cube-1

Building block computing systems are heterogeneous systems consisting of multiple chips connected by TCI. TCI is an electrically contactless 3D stacking technology, enabling wireless communication between various types of chips in a 3D-IC. Several wires to supply power and a system clock are needed for each chip except the host chip stacked on the top of the stack which requires I/O wires. Cube-1[11] is the first prototype consisting of a host microprocessor called Geyser[17] and at most three coarse-grained reconfigurable accelerator chips called CMA[12] . The tasks of image processing can be off-loaded to multiple CMA chips in the pipelined streaming manner. Now we are developing other accelerator chips and SNACC described here is one example.

2.2 Inductive Coupling Channels (TCI)

TCI[9] is a key technology of the building block computing systems. Square coils implemented with arbitrary metal layers are used as data transceivers, no special fabrication technology is needed. Data are transferred through magnetic field between two chips by overlapping a transmitter coil over a receiver coil that are placed in different chips, as shown in Figure 1. A pair of driver and inductor for sending data is called the TX channel, while a pair of receiver and inductor for receiving data is called the RX channel. An inductive coupling channel is usually formed by a coil for system clock and a coil for data transfer. A high-frequency clock (1 - 8 GHz) is generated by a ring oscillator, and the serialized data are transferred following the high-frequency clock directly through the driver.

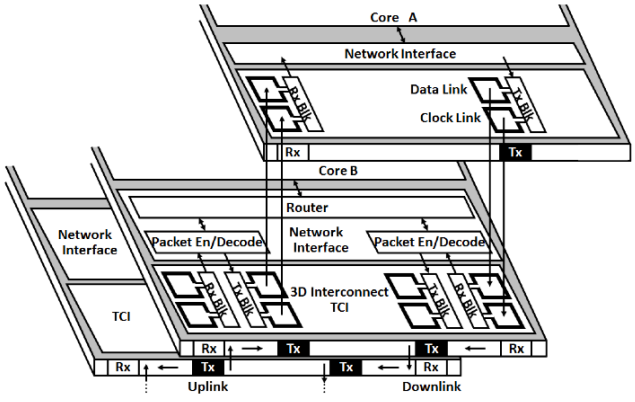


Figure 1: TCI with transmitter and receiver [15]

Although TCI requires a certain amount of logic to form a link between two chips, it has the following benefits.

- 8 Gbps of data at maximum can be transferred with low energy dissipation (0.14 pJ / bit) and low bit-error rate (BER$< 10^{-12}$) [15].
- Multiple chips can be stacked if the physical environment is allowed.
- Since pre-stacking testing of chip is possible, known-good-dies can be selected for stacking after the chip fabrication.
- Since TCI is electrically contact-less, electro-static-discharge (ESD) protection device is unnecessary.
- Since the coil uses the metal layers of conventional CMOS process, no extra process is needed. Although a coil has a large footprint, common digital circuits can be implemented inside the coil.

For simplifying the usage of TCI, we have developed IPs (Intellectual Property) on Renesas 65nm SOTB process and Rhom 0.18μm supported by VDEC Japan. TCI IP is consisting of coils, transmitter, receiver and SERDES (Serializer/De-serializer). The receiver coil and transmitter coil are duplex winding. That is, a channel of the IP is half-duplex. The direction can be switched within a few clock intervals. From the users' perspective, the IP can be treated as a simple 35-bit uni-directional registered channel. Flow-control for bi-directional communication, and a three-port router design are included in the IP.

3 The Target System

3.1 SNACC

3.1.1 Overview of SNACC SNACC is a CNN accelerator chip used as a building block of the proposed computing system. It consists of four SIMD processors, each of which provides support of dedicated instruction set for CNN acceleration and large local memory.

The block diagram of SNACC chip with local memory modules is shown in Figure 2.

SNACC implements a MIPS-like instruction set architecture to simplify the hardware. The instruction width is 16-bit, and there are 16 general purpose registers. R-type (register-register) and I-type (register immediate) instructions are provided, but unlike MIPS, only two operands are specified. Also, the instruction set does not support subroutine call, so the program size for each neural network layer is relatively small.

The most important instructions for CNN acceleration are two SIMD instructions called *mad (multiply add).* and *madlp (multiply add with loop).* The SIMD arithmetic unit is shown in Figure 3 which can handle four 16-bit data or eight 8-bit fixed point arithmetic data. Each multiplier recieves two inputs respectively from two 64-bit registers and the products are all summed together. The max unit outputs the largest value among all the inputs. The outputs from the adder and the max unit are multiplexed to register r13. Furthermore, the function unit,

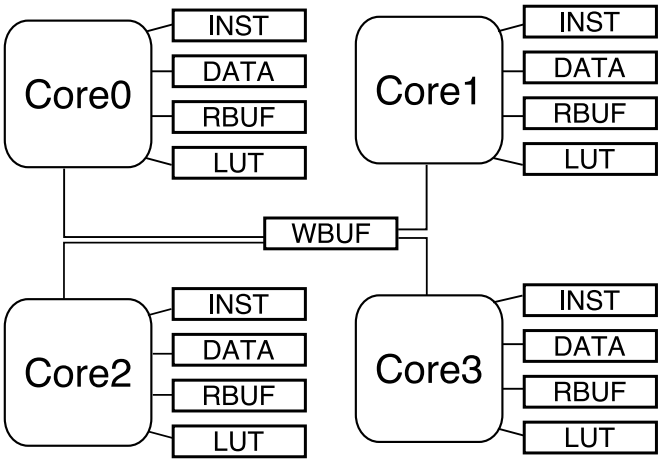


Figure 2: The overview of SNACC with local memory modules

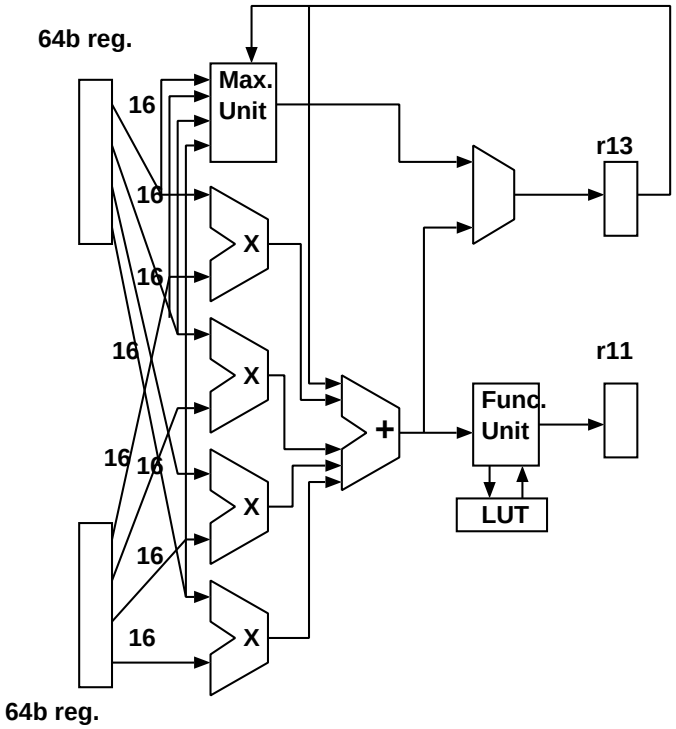


Figure 3: The multiply-add module in SNACC

which is implemented by a lookup table, is applied to the sum and stored into register r11. A *madlp* instruction executes multiply and add operations iteratively for a given number of times. In order to control the dedicated instructions, eight 8-bit control registers and two 32-bit SIMD registers are provided. By these instructions, SNACC offers efficiency and flexibility to CNN execution and their efficiency is evaluated by NAMACHA in Section 5.

The chip has relatively large local memory to effectively exploit data intensive nature of the task. These memory modules can be used for both dedicated and general purposes in local and shared manners. For each core, there is a local look-up table memory and a weight data memory that can be used along with SIMD instructions. There are total 36 independently accessible tables. Compared to other accelerators, the SNACC architecture imposes fewer restrictions over the implemementation of neural network as a relatively general instruction set is used. Therefore, there is no fundamental limitation in the SNACC architecture that prevents future implementation of other neural network architecutre likes recurrent neural network, while CNN is currently the primary focus.

SNACC also provides ThruChip Interface (TCI)[9] for flexible chip stacking. SNACC's local memory is relatively large but still not enough to preserve all the weight data used for practical large CNNs. In the future plan, we are going to place external large DRAM at the bottom of the SNACC-Cube through TCI. In that case, SNACC's SRAM local memory can be considered as a programmable cache.

3.1.2 The SNACC Chip SNACC chip was fabricated by silicon on thin buried oxide (SOTB)[13][14], a novel fully depleted silicon-on-insulator (SOI) developed by the low-power electronics association and project (LEAP) for a low-voltage power-supply operation. The transistors in SOTB are formed on a thin buried oxide layer. The detrimental short channel effect (SCE) is surpressed by using an ultra-thin fully depleted SOI (FD-SOI) layer and a buried oxide (BOX) layer. Since impurity doping (a halo implant) into the channel is not necessary, variations in the threshold voltage due to random dopant fluctuations (RDFs) can be reduced. A multi-threshold voltage design is easily made available by doping an impurity into the substrate directly under the thin BOX layer. Thus, we can extensively control the range of body (back-gate) bias and optimize performance and power consumption after fabrication.

The specifications for the SNACC chip are listed in Table 1, and the proposed software development environment NAMACHA is designed according to it. Figure 4 shows a photograph of the SNACC chip. The left side of the chip is TCI for inter-chip communication, and four SNACC cores are placed on the right side.

3.2 SNACC-Cube

SNACC-Cube is a building block computing system using SNACC chips. As shown in Figure 5, it consists of a host microprocessor Geyser[17] and up to three SNACC chips stacked under the host. The number of SNACC is varied from one to three. Geyser is a MIPS R3000 compatible processor with

Table 1: Specifications for SNACC.

Op. Cond.	Supply voltage Body biasing	0.55V 0.0V
Chip	Process Size I/O	LEAP 65nm SOTB 7-metal 5mm × 5mm 208pins
Tools	Design Synthesis P&R	Verilog HDL Synopsys Design Compiler H-2013.03-SP2 Synopsys IC Compiler G-2012.06-ICC-SP5

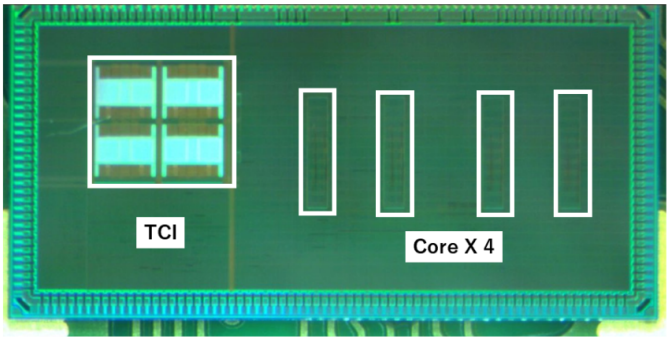


Figure 4: Photograph of CNACC chip.

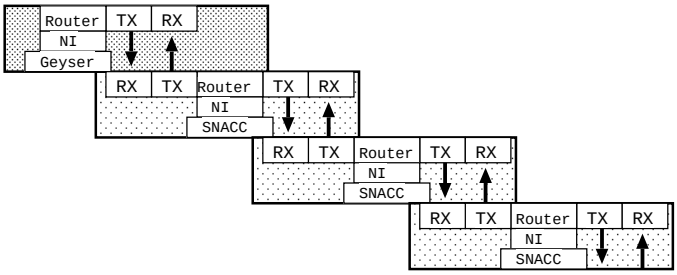


Figure 5: The structure of SNACC-Cube

separated instruction cache and data cache both with 4KB 2-way set associative mapping. Shared TLB with 16-entry is provided and the Linux operating system is available as well as other simpler embedded OS.

They are interconnected with TCI IP and an escalator network is formed as shown in Figure 6. A router with three IO ports is provided in the IP, and a full-duplex bi-directional interconnection is formed between the neighboring stacked chip. By using the opposite direction link, the sending router can detect the buffer status in the receiving router. The routing is controlled just by showing between up direction and down direction, so this simple linear network is called an escalator network.

The input port of each router has eight virtual channels. Since the escalator network is deadlock free without multiple virtual channels, they can be used by the system or application

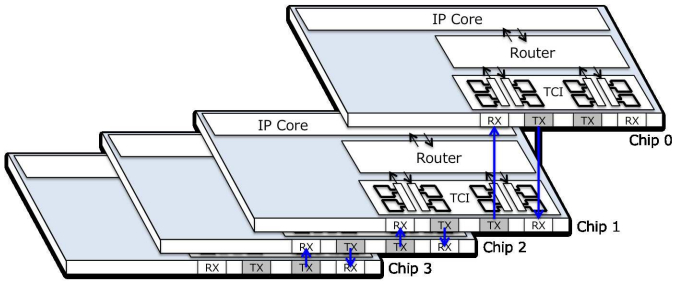


Figure 6: Escalator network in SNACC-Cube

software.

All memory modules of SNACC are mapped into two segments of R3000. When kseg1 is accessed, a single write or read request is issued to SNACC chips. When Geyser accesses kseg0 on which the cache is available, the block transfer using cache block write-back and replace-in is triggered. In order to avoid redundant cache accesses, operations which directly trigger write-back and replace-in are provided.

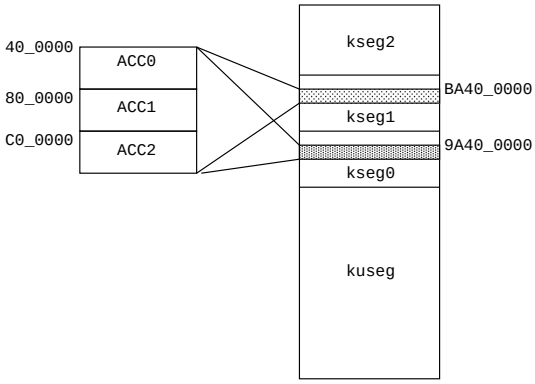


Figure 7: The memory map of SNACC-Cube

SNACC can send the interrupt request to Geyser. Also, the network interface in SNACC provides a DMA-controller, and the data transfer between SNACC chips is triggered just by the request from the Geyser. Figure 8 shows the typical execution of streaming tasks. Note that considerable amount of time is consumed by the data and command packet transfer.

4 NAMACHA Development Kit

NAMACHA is a development kit for CNN implementation running on SNACC-Cube. The goal of NAMACHA is to make programmers of SNACC-Cube able to port existing CNNs without concerning too much about the details of SNACC's architecture. To port existing networks, some degree of re-implementation is unavoidable. It is due to the architectural difference of the instruction set architecture between SNACC and general purpose CPU/GPU. For instance, SNACC has configurable look-up table hardware and performs calculations on fixed point arithmetic while general purpose CPU/GPUs

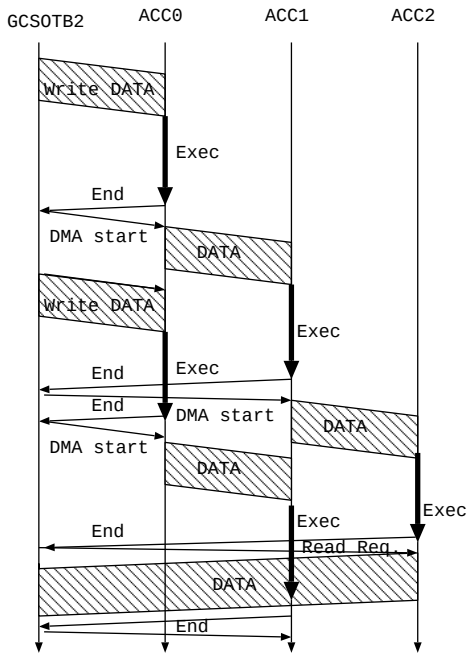


Figure 8: Streaming execution on SNACC-Cube

```
RegGuard channels_loop_counter(
    asm_, output_shape()[0]);
Label channels_start =
    asm_->NewLabelAndMark();
{
  RegGuard data_xs_ptr(asm_, data_ptr);

  RegGuard xs_loop_counter(
      asm_, output_shape()[1]);
  Label xs_start = asm_->NewLabelAndMark();
  {
    ...
  }
  data_xs_ptr += stride_ * input_ys_ * 2;

  xs_loop_counter -= 1;
  asm_->BranchNotZero(
      xs_loop_counter, xs_start);
}
channels_loop_counter -= 1;
asm_->BranchNotZero(
    channels_loop_counter, channels_start);
```

Figure 9: Example using SNACC assembler

do not have such dedicated hardware and provide extensive support on floating point arithmetic. In order to fully utilize the SNACC cores, low-level primitive such as the implementation of convolution layer and ReLU layer[6] must be done by a human programmer.

The NAMACHA development kit is composed of the following five main components: NAMACHA compiler, NAMACHA System Level Simulator, SNACC Runner, Emulating Runner and SNACC I/O library. NAMACHA compiler provides configurable CNN primitives that are well optimized by human programmers and an assembler to generate SNACC instruction according to the configuration. The general workflow of NAMACHA is first to write the configuration of CNN in C++, than compile it using NAMACHA compiler. Afterward, it is possible to identify performance bottleneck using NAMACHA System Level Simulator, and fix them by tuning network hyper parameters or even changing the underlying implementation of the primitives in NAMACHA Compiler. Now, NAMACHA development kit is full text base which can connect tool chain easily.

SNACC Runner and SNACC I/O Library are implemented in C while System Level Simulator, Emulating Runner and Compiler are implemented in C++.

4.1 NAMACHA Compiler

The NAMACHA Compiler consists of the assembler for the SNACC instruction set and the compiler that generates CNN assembly from given configuration of CNN layers.

SNACC assembly is implemented in the form of C++ function instead of usual plain text format for ease of use of the compiler, so the compiler generates SNACC instructions by executing the functions representing SNACC assembly. The assembler uses RAII C++ idiom to allocate registers, while general purpose compilers typically use more complex register allocation algorithms, which allows robust and readable yet fine tunable programming environment. Figure 9 shows a code example of SNACC assembly . (The code fragment is taken from convolution layer implementation.) RegGuard class allocates a register when declared, and frees it when the scope is finished. The class also overloads operators such as += to allow natural representation of arithmetic operations in the assembly.

Abstract C++ class *Layer*, shown in Figure 10, is a template for the implementation of different CNN layers. Each layer takes layer parameters in the constructor and generates SNACC assembly according to the given parameters. Generate() method creates SNACC assembly to the given assembler class which is responsible for SNACC instruction generation. If lut_used() returns true, LutFunction() will convert floating point function value to fixed point value using the look-up table.

Multiple layer classes are arranged to form a particular architecture of neural network. A layer configuration is set up using C++ vector. The layer configuration of AlexNet[8] in NAMACHA Compiler is shown in the Figure 11.

Therefore, programming in SNACC assembly is similar to conventional C++ programming, lowering the barrier for SNACC development. It is planned in the future to support conversion of Caffe or TensorFlow configuration to NAMACHA Compiler's configuration representation.

```
class Layer {
 public:
  virtual void Generate() = 0;
  virtual double LutFunction(double x) = 0;
  virtual void GenerateRbuf(
      vector<uint8_t> *rbuf) = 0;
  virtual bool lut_used() const = 0;
  virtual void set_assembler(Assembler *
      assembler) = 0;
};
```

Figure 10: Abstract Layer class

```
vector<pair<string, vector<Layer*>>> layers =
{
  {"conv1", {
    new Conv(/* kernels = */conv1_kernels,
             /* bias = */conv1_bias,
             /* stride = */4),
    new Relu()}},
  {"norm1", {
    new Lrn(/* local_size = */5,
            /* alpha = */0.0001,
            /* beta = */0.75,
            /*k = */1.0)}},
  {"pool1", {
    new MaxPool(/* kernel_size = */3,
                /* stride = */2)}},
  ...
  {"fc8", {
    new InnerProduct(
        /* kernels = */fc8_kernels,
        /* bias = */fc8_bias)}},
  {"prob", {
    new SoftmaxStage1(),
    new SoftmaxStage2()}}
};
```

Figure 11: AlexNet configuration in SNACC Compiler

4.2 NAMACHA System Level Simulator

The NAMACHA System Level Simulator simulates the entire SNACC-Cube system on a workstation. The system level simulation can be used for finding performance bottlenecks when programming SNACC-Cube, as well as debugging and profiling NAMACHA compiler.

In order to simulate the entire system, two kinds of interpreters are required for Geyser core and accelerator cores simulations due to their different instruction set architectures. To emulate the Geyser chip, we used VMIPS, a GPL licensed MIPS R3000 emulator. For our accelerator core, we built our own interpreter.

The simulator can run in the single core mode and the system level mode. The single core mode simulates only the single core SNACC environment while the system level mode simulates entire SNACC-Cube.

To make the NAMACHA System Level Simulator, a few memory mapped devices are added to the Geyser host simulated by VMIPS. Physically, the SNACC cores are connected to Geyser through TCI channels, but as mentioned above, each local memory of the SNACC chip, as well as the SNACC's control register, is memory mapped to the Geyser's global memory space. Moreover, the simulator generates RS232C compatible output, which allows SNACC Runner running on Geyser, which will be discussed later, to generate debug outputs.

As explained earlier, accurate measurement of Geyser host execution cycles, accelerator execution cycles, and TCI transfer overhead is critical to precisely evaluate the target software efficiency. The following are the analysis of the possible sources of inaccuracy for these measurements.

Geyser Host Cycle Counting. There are several sources of inaccuracy for cycle counting on Geyser MIPS host. Geyser employs traditional five stage in-order pipeline. Simply counting executed instruction number fails to accurately count stalls in the pipeline. Also, Geyser's cache flush mechanism is not accurately implemented in the system level simulator. As a result, it can be another source of cycle count inaccuracy.

SNACC Cycle Counting. Current accelerator design does not implement pipelining but it can be considered as four stage multi-cycle microprocessor. Some SIMD instructions have complex control flow, which may not finish within four cycles, but they are still not difficult to simulate in terms of cycle counting. The hardest part to simulate is counting cycles when taking the arbitration of shared memory into account. SNACC employs the relatively simple mechanism of arbitration for its shared memory, which gives each of the four cores a chance to access the shared memory every four cycles. However, when those SIMD instructions with complex control flow access the shared memory, the resulting timing of arbitration becomes completely unpredictable. Therefore, this may constitute a significant portion of cycle counting inaccuracy for SNACC core simulation.

TCI Communication Latency Modeling. As Cube uses packet routing mechanism to communicate through TCI, there is no guarantee over the total latency observed in the real hardware. Leaving the complex and precise modeling of that latency to future work, current implementation of the NAMACHA System Level Simulator took a simpler approach by stalling a constant number of cycles for reading and writing respective memory region. The constant number is a reasonable number given by the knowledge of TCI router implementation and significantly larger than writing to the Geyser's main memory. It assumes one-byte transfer through the TCI takes 25 cycles and each router takes an additional 50% margin.

4.3 Emulating Runner

NAMACHA also provides Emulating Runner written in C++. The purpose of Emulating Runner is to offer an integrated

test environment for SNACC using the SNACC single core simulator. Emulating Runner supplies input data to run SNACC single core simulator and then copies emitted output as the next layer's input. The Runner is able to compare the output to that of a general purpose deep learning framework such as Caffe to debug the generated SNACC instructions of the implemented CNN architecture.

4.4 SNACC Runner

NAMACHA Compiler generates SNACC instructions, but the control from the Geyser host to the SNACC cores is also necessary. SNACC Runner on the host core sends SNACC program from the host to each layer and orchestrates the data transfers between the chips. Current NAMACHA implementation assumes there is enough large local memory, therefore what the SNACC Runner does is simply copying input data to SNACC local memory and transferring output data back to the host for every layer when SNACC execution is completed. But in the future, when larger DRAM is attached at the bottom of the SNACC-Cube and smaller local memory is used as a programmable cache, the control flow will become complex and how the SNACC Runner performs the task effectively will be important to the entire system performance.

4.5 SNACC I/O Library

As mentioned above, SNACC is controlled by the Geyser host core through memory-mapped I/O. SNACC I/O library gives simple abstraction over this memory mapped I/O. The library is used by the SNACC Runner. The enrichment of the abstraction library is planned.

4.6 GCC Cross Compiler Toolchain for Geyser

The Geyser host core implements general MIPS R3000 instruction set. For the host software development, a cross compiler of GNU Compiler Collection running on x86 Linux is used. The output image can be supplied to both register transfer level simulation and system level simulation without any change.

5 Evaluation

5.1 NAMACHA Simulator

The RTL of the system is implemented in Verilog, and the simulation is done on Cadence's NC-Sim 10.20-s131. The NAMACHA System level simulator is C++ program compiled with GCC Version 6.2.1 and -O2 option. Both simulators ran on CentOS 6.5 machine with its CPU Intel Xeon E5-2667. The code used for the simulator evaluation is hand-assembled small four layer CNN.

The simulation results of NAMACHA System Level simulator and the Verilog-HDL RTL simulation are shown in Table 2. NAMACHA System Level time includes the simulation of SNACC Runner mentioned above. The value of the simulation time difference in a single core is unreliable as the time consumed is too small, but at the system level, it is clear that the use of a system level simulator gives significant performance improvement, thus giving far shorter round time of the software development.

The unnatural difference of the entire system simulation time between RTL and NAMACHA is actually reasonable as the RTL simulator offers far more information about the hardware signals during simulation. The RTL simulator is rather a tool for hardware development of processor and accelerator, but not for the development of embedded software that runs on these hardware systems. The NAMACHA System Level Simulator cannot be used for debugging the accelerator hardware. Considering the information granularity given by the RTL simulator, it is running an acceptable speed.

Table 2: NAMACHA Simulator vs. Verilog HDL simulation

	Simulation Time	Cycles Counted
NAMACHA System Level	0.04s (4390x faster)	375253 (1.27% error)
NAMACHA Single Core	0.01s (188x faster)	31793 (7.85% error)
RTL Entire System	175.60s	380097
RTL Single Core	1.88s	34501

5.2 AlexNet on NAMACHA Simulator

AlexNet[8] was implemented on NAMACHA simulator, and the clock cycle counts of a single core were evaluated. The results of each layer are shown in Table 3. It shows that the convolutional layers in the later stage occupies a large part of computation time. While convolution layers are computationally intensive, inner product layers are communication intensive, thus the acceleration in SNACC should be only applied to the convolution layers.

By using the NAMACHA simulator, the contribution of SIMD instructions (*mad/madlp*) shown in Section 3 was analyzed. Figure 12 shows the performance with and without SIMD instructions respectively for all layers in AlexNet. The performance improvement by SIMD instructions can be observed in all layers. Especially, fc6-8 corresponding to all-reduce operation can be much reduced. It appears that NAMACHA is useful as a tool for improving SNACC architecture.

6 Related Work

General purpose Deep Learning framework e.g. TensorFlow typically offers CPU and GPU implementation. However, accelerators using FPGA and ASIC are advantageous in terms of power consumption and sometimes in total performance.

Table 3: Cycle counts for each layer of AlexNet

layer	sub-layer	cycle counts	ratio
conv1	conv	673052316	6.4%
	relu	10454492	
norm1	lrn	138596152	1.3%
pool1	max_pool	9583548	0.1%
conv2	pad	2423464	10.3%
	conv	1087449148	
	relu	6718556	
norm2	lrn	91633788	0.9%
pool2	max_pool	5973056	0.1%
conv3	pad	1584172	23.1%
	conv	2461776440	
	relu	2336344	
conv4	pad	2376236	34.6%
	conv	3691166268	
	relu	2336344	
conv5	pad	2376236	23.1%
	conv	2460777528	
	relu	1557592	
pool5	max_pool	1299520	0.0%
fc6	inner_product	9662556	0.1%
	relu	147544	
fc7	inner_product	4419672	0.0%
	relu	147544	
fc8	inner_product	1079092	0.0%

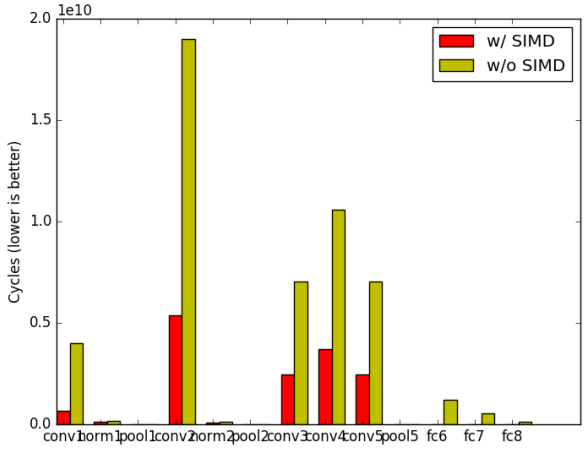


Figure 12: The performance improvement with mad/madlp instructions

An example of accelerator using FPGA is Zhang et al., 2015[16]. Recently, there are ASIC examples including DianNao[3], DaDianNao[4], Eyeriss[5] and EIE[7]. DianNao and DaDianNao are SIMD accelerators, and DaDiaNao provides a large eDRAM in the chip. Eyeriss uses a two dimensional many core architecture which can reuse the convolution layer. It also provides run-length compression mechanism to reduce the traffic between the outside DRAM. EIE focuses on the sparse structure of CNN and tries to reduce the total amount of data without degrading the recogition accuracy. Cognitive chip[10] also takes the completely different approach as they are trying to emulate biological neurons.

Unfortunately, no software system like NAMACHA has been reported for such ASIC CNN accelerators. The main reason comes from a high degree of flexibility of SNACC-Cube. The number of connected chips can be freely changed in SNACC-Cube, thus the system configuration including overhead of the network is changed. The programmers must develop their program considering the system configuration, thus the integrated software system including accurate simulator is needed. In other ASIC CNN accelerators, the system configuration and target CNN system are mostly fixed and the programmer just develops code specialized for the target systems.

7 Conclusions and Future Work

In this article, we propose an integrated program development environment NAMACHA for a building-block computation system SNACC. The NAMACHA System Level Simulator supports development by offering 4390x faster yet 1.27% accurate simulation environment. The NAMACHA Compiler enables easier porting of existing CNNs. The simulation results of implementing AlexNet, SIMD instructions provided in the accelerator improved the performance by 70% on average. It demonstrates that the NAMACHA can be used for architectural exploration as well as development of practical software.

We have planned to improve SNACC-Cube as well as NAMACHA environment. We are going to attach large external DRAM to the bottom of the system. On the compiler side, implementation of Caffe/TensorFlow configuration converter, and also the support of RNN and other DNN variants are planned. For the simulator, precise latency modeling of communication through TCI is left to future work. For this time we selected VMIPS for simplicity, but we can consider Just-In-Time emulators such as QEmu[2] when the interpreter performance becomes a bottleneck as the software gets larger. Finally, the development of GUI for showing the result of simulator is also our future work.

Acknowledgments

This work is partially supported by JSPS KAKENHI S grant number 25220002.

References

[1] "A Study on Building-Block Computing Systems using Inductive Coupling Interconnect". URL {http://www.am.ics.keio.ac.jp/kaken_s/}.

[2] F. Bellard. "QEMU , A Fast and Portable Dynamic Translator". *USENIX Annual Technical Conference. Proceedings of the 2005 Conference on*, pp. 41–46, 2005.

[3] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. "DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning". *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*, pp. 269–284, 2014.

[4] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam. "DaDianNao: A Machine-Learning Supercomputer". *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, pp. 609–622, 2015.

[5] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze. "An Energy Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks ". *Proceedings of the IEEE International Solid-State Circutirs Conference (ISSCC)*, pp. 262–263, 2016.

[6] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks". *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 15:315–323, 2011.

[7] S. Han, X. Lin, H. Mao, J. Pu, A. Pedram, M. Horowitz, and W. Dally. "EIE: Efficient Inference Engine on Compressed Deep Nerutal Network". *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*, pp. 243–254, 2016.

[8] A. Krizhevsky, I. Sutskever, and H. Geoffrey E. "ImageNet Classification with Deep Convolutional Neural Networks". *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pp. 1–9, 2012.

[9] T. Kuroda. "ThruChip Interface (TCI) for 3D Networks On Chip". *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, pp. 238–241, Oct 2011.

[10] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha. "A Digital Neurosynaptic Core Using Embedded Crossbar Memory with 45pJ per spike in 45nm". *Proceedings of the Custom Integrated Circuits Conference*, pp. 1–4, 2011.

[11] N. Miura, Y. Koizumi, Y. Take, H. Matsutani, T. Kuroda, H. Amano, R. Sakamoto, M. Namiki, K. Usami, M. Kondo, and H. Nakamura. "A Scalable 3D Heterogeneous Multicore with an Inductive ThruChip Interface". *IEEE Micro*, 33(6):6–15, 2013.

[12] N. Ozaki, Y. Yoshihiro, Y. Saito, D. Ikebuchi, M. Kimura, H. Amano, H. Nakamura, K. Usami, M. Namiki, and M. Kondo. "Cool Mega-Array: A Highly Energy Efficient Reconfigurable Accelerator". *2011 International Conference on Field-Programmable Technology*, pp. 1–8, 2011.

[13] R. Tsuchiya and M. Horiuchi and S. Kimura and M. Yamaoka and T. Kawahara and S. Maegawa and T. Ipposhi and Y. Ohji and H. Matsuoka . "Ultralow-power LSI Technology with Silicon on Thin Buried Oxide (SOTB) CMOSFET". *Solid State Circuits Technologies, Jacobus W. Swart (Ed.), InTech*, pp. 146–156, 2010.

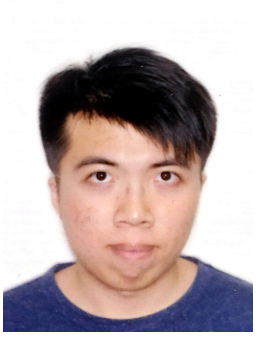
[14] R. Tsuchiya and M. Horiuchi and S. Kimura and M. Yamaoka and T. Kawahara and S. Maegawa and T. Ipposhi and Y. Ohji and H. Matsuoka. "Silicon on thin BOX : A New Paradigm of the CMOSFET for Low-Power and High-Performance Application Featuring Wide-Range Back-Bias Control". *Tech. Dig. Int, Electron Devices Meet.*, pp. 631–634, 2004.

[15] Y. Take, H. Matsutani, D. Sasaki, M. Koibuchi, T. Kuroda, and H. Amano. "3D NoC with Inductive-Coupling Links for Building-Block SiPs". *IEEE Transactions on Computers*, 63(3):748–763, 2014.

[16] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong. "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks". *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '15*, pp. 161–170, 2015.

[17] L. Zhao, D. Ikebuchi, Y. Saito, M. Kamata, N. Seki, Y. Kojima, H. Amano, S. Koyama, T. Hashida, Y. Umahashi, D. Masuda, K. Usami, K. Kimura, M. Namiki, S. Takeda, H. Nakamura, and M. Kondo. "Geyser-2: the Second Prototype CPU with Fine-grained Run-time Power Gating". *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, pp. 87–88, 2011.

Tesui Okubo graduated from the Department of Information and Computer Science, Keio University at 2017. He currently works at Google.



Mankit Sit is a master student at Keio University. He received a BEng degree from the Chinese University of Hong Kong in 2016.

Hideharu Amano is a Professor at the Department of Information and Computer Science, Keio University. He received a Ph.D degree from the Department of Electronic Engineering, Keio University, Japan in 1986. His research interests include the area of parallel architectures and reconfigurable systems.

Ryo Takata is a master student at the University of Tokyo.

Ryuichi Sakamoto is a postgraduate researcher at the University of Tokyo. He received MS and PhD degree from Tokyo University of Agriculture and Technology.

Masaaki Kondo is an Associate Professor at the Graduate School of Information Systems at the University of Electro-Communications in Tokyo. His research interests include computer architectures, high-performance computing, and dependable computing. Kondo has a PhD in electrical engineering from the University of Tokyo.

Instructions for Authors

The International Journal of Computers and Their Applications is published multiple times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

A. Procedure for Submission of a Technical Paper for Consideration

1. Email your manuscript to the Editor-in-Chief, Dr. Fred Harris, Jr., Fred.Harris@cse.unr.edu.

2. Illustrations should be high quality (originals unnecessary).

3. Enclose a separate page (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.

B. Manuscript Style:

1. The text should be **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. Figures should be captioned and referenced.

C. Submission of Accepted Manuscripts

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-Chief.
2. The submission may be on a CD/DVD or as an email attachment(s) . **The following electronic files should be included:**
    - Paper text (required).
    - Bios (required for each author). Integrate at the end of the paper.
    - Author Photos (jpeg files are required by the printer, these also can be integrated into your paper).
    - Figures, Tables, Illustrations. These may be integrated into the paper text file or provided separately (jpeg, MS Word, PowerPoint, eps).

3. Specify on the CD/DVD label or in the email the word processor and version used, along with the title of the paper.

4. Authors are asked to sign an ISCA copyright form (http://www.isca-hq.org/j-copyright.htm), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

Publication Charges

After a manuscript has been accepted for publication, the contact author will be invoiced for publication charges of **$50.00 USD** per page (in the final IJCA two-column format) to cover part of the cost of publication. For ISCA members, $100 of publication charges will be waived if requested.

January 2014