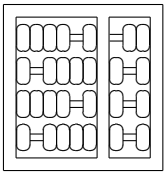


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Anais do XI Workshop de Teses, Dissertações e Trabalhos de Iniciação Científica do IC-Unicamp

Profa. Ariadne Maria Brito Rizzoni
Profa. Esther Luna Colombini
Profa. Juliana Freitag Borin
Allan da Silva Pinto
Amanda Cristina Davi Resende
Carlos Alberto Petry *Eliana Alves Moreira*
Emanuel Felipe Duarte *Ícaro Cavalcante Dourado*
José Valderlei da Silva
Luís Augusto Martins Pereira
Lucas Augusto Carvalho *Samuel Botter Martins*
Sueny Messias *Vanessa R. M. L. Maike*

Technical Report - IC-16-05 - Relatório Técnico

August - 2016 - Agosto

Apresentação

Este relatório técnico contém os resumos de 26 trabalhos cujos artigos foram autorizados a serem publicados no *XI Workshop* de Teses, Dissertações e Trabalhos de Iniciação Científica (WTD), do Instituto de Computação (IC) da Universidade Estadual de Campinas (Unicamp), edição 2016.

O *XI Workshop* ocorreu entre os dias 3 e 4 de Agosto de 2016 e contou com cerca de 180 participantes, entre ouvintes, apresentadores de trabalhos e organizadores. Na ocasião houve 30 apresentações orais e 16 pôsteres. Aos alunos foi dada a possibilidade de escolher a forma de apresentação (oral, pôster), bem como escolher se desejasse publicar ou não seu trabalho nos anais do evento. A publicação dos resumos, sob forma de relatório técnico, tem por objetivo divulgar os trabalhos em andamento e concluídos e registrar, de forma sucinta, o estado da arte da pesquisa do Instituto de Computação no ano de 2016.

Neste ano ocorreram duas palestras. A primeira, intitulada *"Ciência e Tecnologia no Estado de São Paulo"*, foi proferida pelo Professor Doutor Carlos Henrique de Brito Cruz, diretor científico da *FAPESP*. A segunda, intitulada *"Desafios da Robótica Inteligente"*, foi proferida pela Professora Doutora Esther Luna Colombini, docente do *IC - Unicamp*, além de coordenadora da Olimpíada Brasileira de Robótica (OBR). Também foi oferecido o minicurso intitulado "Planejamento das Condições de Ensino e Aprendizagem", ministrado pelo Professor Doutor Sérgio Antonio da Silva Leite, da Faculdade de Educação (FE) da Unicamp, e coordenador do EA2 - Espaço de Apoio ao Ensino e Aprendizagem da Unicamp. Tanto as palestras como o minicurso foram gravados e transmitidos *online* via Web.

Agradecemos aos alunos que participaram do evento, em particular àqueles que se dispuseram a apresentar seus trabalhos, seja oralmente ou na forma de pôsteres, bem como aos orientadores que os incentivaram a fazê-lo. Agradecemos, também, aos professores e alunos de doutorado do IC que compuseram as bancas de avaliação dos trabalhos. Agradecemos ao Professor Doutor Ricardo da Silva Torres, diretor do IC, e ao Professor Doutor Julio César López Hernández, coordenador da Pós-Graduação, pelo incentivo, apoio e patrocínio ao evento. Finalmente, agradecemos imensamente aos alunos do programa de Pós-Graduação do IC que efetivamente organizaram o evento e que são coeditores deste relatório – Allan da Silva Pinto, Amanda Cristina Davi Resende, Carlos Alberto Petry, Eliana Alves Moreira, Emanuel Felipe Duarte, Ícaro Cavalcante Dourado, José Valderlei da Silva, Luís Augusto Martins Pereira, Lucas Augusto Carvalho, Samuel Botter Martins, Sueny Messias e Vanessa R. M. L. Maike. A eles dedicamos o XI *Workshop* de Teses, Dissertações e Trabalhos de Iniciação Científica do Instituto de Computação da Unicamp.

Dra. Ariadne Maria Brito Rizzoni Carvalho

Dra. Esther Luna Colombini
Dra. Juliana Freitag Borin
Coordenadoras do XI WTD
Professoras do Instituto de Computação - Unicamp

Sumário

1. Programação do WTD

"Ou nos contentamos com a falta de algumas coisas em nossa vida, ou lutamos para realizar todas a nossas loucuras.".

Mário Quintana

Neste capítulo, apresentamos a programação e algumas estatísticas do *XI Workshop de Teses, Dissertações e Trabalhos de Iniciação Científica* (WTD) do Instituto de Computação (IC) da Unicamp.

Na edição deste ano, correram no dia 03 de Agosto duas palestras de abertura e um minicurso, conforme detalhado na tabela 1.1. A primeira palestra, intitulada *"Ciência e Tecnologia no Estado de São Paulo"*, foi ministrada pelo Professor Doutor Carlos Henrique de Brito Cruz da *Universidade Estadual de Campinas, Instituto de Física Gleb Wataghin* e a segunda palestra, intitulada *"Desafios da Robótica Inteligente"*, foi ministrada pela Professora Doutora Esther Luna Colombini da *Universidade Estadual de Campinas, Instituto de Computação*. O Professor Doutor Sérgio Leite, da *Universidade Estadual de Campinas, Faculdade de Educação*, ministrou o minicurso intitulado *"Planejamento das Condições de Ensino e Aprendizagem"*. As palestras e o minicurso foram transmitidos *online* pelo *website* do WTD (www.ic.unicamp.br/wtd/2016).

Tabela 1.1: Programação das palestras e minicurso do XI WTD realizados no dia 03 de Agosto de 2016. As apresentações foram realizadas no Centro de Convenções da Unicamp.

Hora	**Título**	**Palestrante**
09:00 – 12:00	Minicurso: Planejamento das Condições de Ensino e Aprendizagem	Professor Doutor Sérgio Antonio da Silva Leite
14:00 – 16:00	Palestra: Ciência e Tecnologia no Estado de São Paulo	Professor Doutor Carlos Henrique de Brito Cruz
16:30 – 18:30	Palestra: Desafios da Robótica Inteligente	Professora Doutora Esther Luna Colombini

As apresentações orais dos trabalhos foram realizadas no dia 04 de Agosto em três salas (351, 352 e 353) e os pôsteres foram exibidos no *hall* do IC3, cuja programação está detalhada nas Tabelas 1.2 a 1.5). Os trabalhos foram avaliados por bancas formadas por dois professores e um aluno de doutorado do Instituto de Computação, os alunos também tiveram participação como *chairs* das sessões.

O XI WTD contou com um total de 46 participações, sendo 30 apresentações orais e 16 apresentações na forma de pôsteres. A distribuição dos trabalhos com relação às quatro áreas de pesquisa do IC (Figura 1.1) estão assim distribuídas: 5 para Teoria da Computação, 20 para Sistemas de Informação, 19 para Sistemas de Computação e 2 para Sistemas de Programação. Outra estatística medida se refere ao número de trabalhos quanto ao nível do aluno (Figura 1.2): 11 trabalhos apresentados foram de alunos de iniciação científica, 15 foram de mestrado e 19 de doutorado. Entre as participações 12 foram de iniciação científica, 15 foram de mestrado e 19 de doutorado (Figura 1.3). A Figura 1.4 apresenta a representação visual em formato

de *Tag Cloud* das palavras-chave relacionadas aos trabalhos apresentados. Houve a participação de cerca de 160 pessoas no evento ao longo de todo o *Workshop*, entre participações presenciais e online. A organização do XI WTD contou com 15 pessoas que atuaram como membros do comitê de organização do evento. Os certificados de participação das apresentações orais e pôsteres foram confeccionados em formato eletrônico e disponibilizados a todos os participantes.

Tabela 1.2: Programação das apresentações orais do XI WTD realizadas na sala 351 do Instituto de Computação (IC-3.5), no dia 04 de Agosto de 2016.

Início	Aluno	Trabalho	Orientador	Nível	Banca
09:00	Pedro Paulo Libório Lima do Nascimento	Characterizing GPS Outages: Geodesic Dead Reckoning Solution for VANETs and ITS	Leandro Aparecido Villas	Mestrado	Lucas Wanner, Lehilton, Allan Mariano de Souza
09:30	Frances Albert Santos	A Roadside Unit-Based Localization Scheme to Improve Positioning for Vehicular Networks	Leandro Aparecido Villas	Doutorado	Lucas Wanner, Lehilton, Allan Mariano de Souza
10:30	Leonardo Alves de Melo	Um sistema para contagem de pessoas em passagens	Edson Borin	Iniciação Científica	Flávio Miyazawa, Mário Cortes, Takeo Akabane
11:00	Flávia Pisani	A platform for data analytics on the IoT with declarative languages	Edson Borin	Doutorado	Flávio Miyazawa, Mário Cortes, Takeo Akabane
14:00	Marcus Felipe Botacin	Análise de malware utilizando suporte de hardware	Paulo Lício de Geus	Mestrado	Islene C. Garcia, Neucimar J. Leite, Celso Brennand
14:30	Marcelo Invert Palma Salas	Análise de Segurança para a Descoberta, Bloqueio e Rastreamento de Tráfego Malicioso sobre a Rede Tor	Paulo Lício de Geus	Doutorado	Islene C. Garcia, Neucimar J. Leite, Celso Brennand
15:30	Daniel Rodrigues Carvalho	Dynamically Skewed Compressed Cache	Rodolfo Jardim de Azevedo	Mestrado	Tomasz Kowaltowski, Diego Aranha, Luan Cardoso dos Santos
17:00	João Moreira	Feel the Flow: Riding Kernels and Krakens With Control-Flow Integrity	Sandro Rigo	Doutorado	Esther Luna Colombini, Rodolfo J. de Azevedo, Vitor Afonso
17:30	Pedro De Nigris Vasconcellos	Caracterização do Raspberry Pi	Lucas Wanner	Iniciação Científica	Esther Luna Colombini, Rodolfo J. de Azevedo, Vitor Afonso

Tabela 1.3: Programação das apresentações orais do XI WTD realizadas na sala 352 do Instituto de Computação (IC-3.5), no dia 04 de Agosto de 2016.

Início	**Aluno**	**Trabalho**	**Orientador**	**Nível**	**Banca**
09:00	Alexandre Yukio Yamashita	Detection of the Alzheimer disease using differences of center of mass in amygdala-hippocampal regions	Neucimar Jerônimo Leite	Mestrado	Jacques Wainer, André Santanchè, John Vargas
09:30	Darwin Saire Pilco	Multi-scale Morphological Image Simplification Based on Extrema Relationships: Improvements and Applications	Neucimar Jerônimo Leite	Mestrado	Jacques Wainer, André Santanchè, John Vargas
10:30	Lucas Carvalho Leal	Testes online baseados em modelos para arquitetura baseada em serviços dinâmicos	Eliane Martins	Mestrado	Ricardo Torres, Cecilia Rubira, Marcelo Palma Salas
11:00	Wallace Felipe Francisco Cardoso	Um estudo experimental sobre testes funcionais e de robustez de acordo com a análise de mutantes	Eliane Martins	Doutorado	Ricardo Torres, Cecilia Rubira, Marcelo Palma Salas
14:00	Jacqueline Midlej do Espírito Santo	Interoperabilidade de dados na área da saúde	Claudia Bauzer Medeiros	Doutorado	Zanoni Dias, Júlio C. dos Reis, Jael Zela
14:30	Lucas Augusto Montalvão Costa Carvalho	Provenance-Based Retrieval: Fostering Reuse and Reproducibility Across Scientific Disciplines	Claudia Bauzer Medeiros	Doutorado	Zanoni Dias, Júlio C. dos Reis, Jael Zela
15:00	Leandro Tacioli	Automatic Bioacoustic Identification Comparison Using Different Features and Classifiers	Claudia Bauzer Medeiros	Mestrado	Edmundo Madeira, Guilherme Pimentel, Felipe Cardoso
15:30	Márcio de Carvalho Saraiva	Extraction of implicit relationships among courseware content	Claudia Bauzer Medeiros	Doutorado	Edmundo Madeira, Guilherme Pimentel, Felipe Cardoso

17:00	Anderson Rossanez	Checklist Quality Assessment of Natural Language Requirements for Space Applications	Ariadne M. B. R. Carvalho	Mestrado	Heiko Hornung, Eliane Martins, Wallace Felipe
17:30	Juliana Medeiros Destro	Fatores relevantes no mapeamento de ontologias em línguas diferentes	Ariadne M. B. R. Carvalho	Doutorado	Heiko Hornung, Eliane Martins, Wallace Felipe
18:00	Fernando J. V. da Silva	Identifying Emotions in Tweets related to the Brazilian Stock Market	Ariadne M. B. R. Carvalho	Doutorado	Heiko Hornung, Eliane Martins, Wallace Felipe

Tabela 1.4: Programação das apresentações orais do XI WTD realizadas na sala 353 do Instituto de Computação (IC-3.5), no dia 04 de Agosto de 2016.

Início	Aluno	Trabalho	Orientador	Nível	Banca
09:00	Atilio Gomes Luiz	Duas famílias de caterpillars 0-rotativos	Christiane Neme Campos	Doutorado	Orlando Lee, Cid Carvalho, Rafael Arakaki
09:30	Mauricio Jose de Oliveira Zambon	Exact Solutions for the Geometric Firefighter Problem	Pedro Jussieu de Rezende	Doutorado	Orlando Lee, Cid Carvalho, Rafael Arakaki
10:30	Lucas Zanco Ladeira	Ataques de Canal Lateral em Aplicações de Internet das Coisas	Ricardo Dahab	Mestrado	Fábio Usberti, Rafael Schouery, Felipe A. Louza
11:00	Tiago Reis	Secure and Efficient Software Implementations of Neeva Hash Function for ARM Architectures	Julio López	Mestrado	Fábio Usberti, Rafael Schouery, Felipe A. Louza
14:00	Rafael de Oliveira Werneck	A bag-of-graphs approach for objects with multiple modalities	Ricardo da Silva Torres	Doutorado	Ricardo Dahab, Eliane Martins, Lin Tzy Li
14:30	Greice Cristina Mariano	Uso de Técnicas de Visualização de Informação para Detecção de Padrões Temporais Cíclicos Associados a Estudos de Fenologia	Ricardo da Silva Torres	Doutorado	Ricardo Dahab, Eliane Martins, Lin Tzy Li
15:00	Ícaro Cavalcante Dourado	An accurate, efficient, and general-purpose graph-based text representation model	Ricardo da Silva Torres	Mestrado	Ricardo Dahab, Eliane Martins, Matheus Mota
15:30	Juan Felipe Hernández Albarracín	Aprendizado de Índices Espectrais com Programação Genética	Ricardo da Silva Torres	Mestrado	Ricardo Dahab, Eliane Martins, Matheus Mota

17:00	Marleny Luque Carbajal	Design de um ambiente de Programação Tangível para crianças no contexto educativo brasileiro	Maria Cecília Calani Baranauskas	Doutorado	Edson Borin, Alexandre Mello Ferreira, Márcio de Carvalho Saraiva
17:30	Laís Minchillo	Programação como ferramenta de ensino de pensamento computacional	Juliana Freitag Borin	Mestrado	Edson Borin, Alexandre Mello Ferreira, Márcio de Carvalho Saraiva

Tabela 1.5: Programação das apresentações de pôsteres do XI WTD realizadas no *Hall* do IC3.5

Aluno	**Título**	**Orientador**	**Nível**	**Área**
Agnaldo Aparecido Esmael	Multiscale Multiclass Classification of Remote Sensing Images	Ricardo da Silva Torres	Doutorado	Sistemas de Informação
Akari Ishikawa	How a drug can make your brain forget the words? A language functional MRI study with topiramate in epilepsy and headache	Clarissa L Yasuda	Iniciação Científica	Sistemas de Programação
Alysson Bolognesi Prado	Quid.net.br - Uma Ferramenta para Visualização e Expansão de Revisões Bibliográficas	Maria Cecilia Baranauskas	Doutorado	Sistemas de Informação
Caio Salvador Rohwedder	Uma classificação de aplicações Android para detecção de energy bugs	Edson Borin	Iniciação Científica	Sistemas de Computação
Carlos Eduardo Millani	COISA Bot: Uma plataforma para apoio ao ensino de pensamento computacional	Edson Borin	Iniciação Científica	Sistemas de Computação
Diogo Hideki Shiraishi	Projeto de um dispositivo IoT de baixo custo para sensoriamento de ambientes	Edson Borin	Iniciação Científica	Sistemas de Computação
Eliana Alves Moreira	A comunicação alternativa e o lúdico via interfaces tangíveis	Maria Cecilia Baranauskas	Doutorado	Sistemas de Informação
Felipe dos Santos Pinto de Andrade	Three Scales of Evolutionary Computation	Ricardo da Silva Torres	Doutorado	Sistemas de Informação
Gabriel Souza Franco	Uma análise do consumo energético de processadores heterogêneos	Edson Borin	Iniciação Científica	Sistemas de Computação
Henrique Machado Gonçalves	análise de superfícies de tempo de trânsito na análise de dados sísmicos	Edson Borin	Iniciação Científica	Sistemas de Computação
Hercules Cardoso da Silva	Aceleração de métodos de processamento síncrono com OpenCL	Edson Borin	Mestrado	Sistemas de Computação
Jeferson Rech Brunetta	Uma plataforma para execução de código como serviço em dispositivos IoT	Edson Borin	Mestrado	Sistemas de Computação

João Henrique Speglich	análise de heurísticas de busca para a estimação de parâmetros geofísicos	Edson Borin	Iniciação Científica	Sistemas de Computação
Luan Egidio Ferreira	Uma análise do consumo energético de uma plataforma para dispositivos IoT	Edson Borin	Iniciação Científica	Sistemas de Computação
Luís Fernando Vieira Silva	Geração automática de callgraphs para detecção de energy bugs	Edson Borin	Iniciação Científica	Sistemas de Computação
Maria B Kersanach	Projeto de uma Interface Cérebro-Computador Baseada em Imaginação de Movimento	Romis Attux	Iniciação Científica	Sistemas de Computação

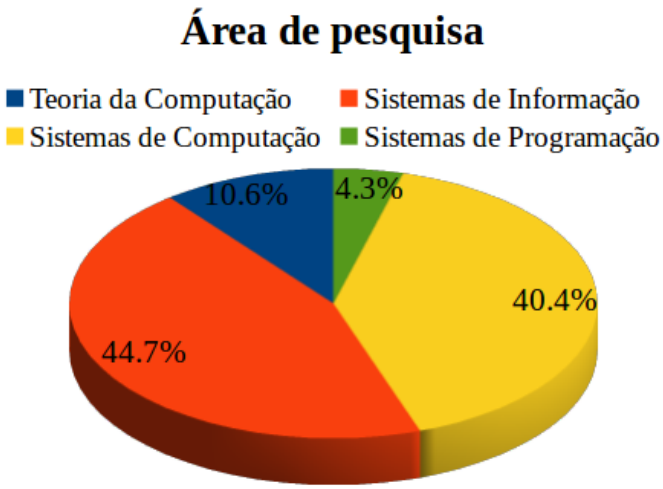


Figura 1.1: Distribuição dos trabalhos apresentados considerando as áreas de pesquisa.

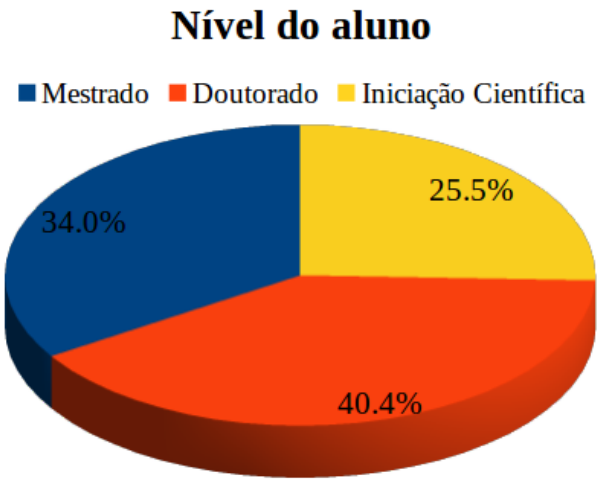


Figura 1.2: Quantidade relativa de alunos nos níveis Iniciação Científica, Mestrado e Doutorado.

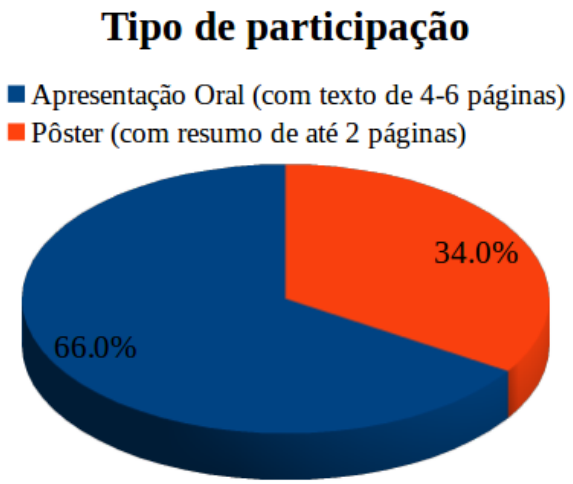


Figura 1.3: Percentual de trabalhos relativos ao tipo de participação (Apresentação oral e Poster).

Figura 1.4: *Tagcloud* com as palavras-chave relacionadas nos trabalhos dos participantes.

2. Resumos estendidos

"Deus não escolhe os capacitados, capacita os escolhidos. Fazer ou não fazer algo só depende de nossa vontade e perseverança".

Albert Einstein

Análise de *malware* utilizando suporte de *hardware*

Marcus Botacin[1], André Grégio[1,2], Paulo Lício de Geus[1]

[1]Instituto de Computaçao (IC) – Universidade Estadual de Campinas (UNICAMP)

[2]Departamento de Informática – Universidade Federal do Paraná (UFPR)

Abstract. *Malware infection is one of main current threats to computer systems, causing image and financial damage to users. Dynamic analysis systems have been applied as forensic and countermeasure-development tools. However, such systems have their effectiveness limited due to samples capable of evasive actions, which are able to detect monitoring systems. In this work, we propose a transparent analysis technique, hardware supported through branch monitoring, which is able to analyze evasive malware.*

Resumo. *A infecção por malware é uma das maiores ameaças atuais a sistemas computacionais, causando prejuízos, de imagem e financeiros, aos usuários. Sistemas de análise dinâmica têm sido usados como forma de forense e de desenvolvimento de contra-medidas frente a tais infecções. Contudo, tais sistemas têm tido sua eficácia limitada devido a exemplares capazes de ações evasivos, capazes de detectar monitoramento. Neste trabalho, propomos uma técnica de análise transparente, suportada por recursos de hardware via monitores de branch, capaz de analisar malware evasivo.*

1. Introdução

Software malicioso (*malware*) é uma das maiores ameaças aos sistemas computacionais. A infecção por *malware* é responsável por causar danos de imagem e financeiros a empresas e usuários. Fraudes bancárias causadas por *malware*, por exemplo, estão entre as maiores preocupações dos bancos do país [FEBRABAN 2012].

Para lidar com infecções de *malware*, técnicas de análise são utilizadas. Estas permitem identificar o comportamento do artefato malicioso e, assim, desenvolver contramedidas e soluções de forense e prevenção.

A análise de *malware* pode se dar de forma estática e/ou dinâmica [Sikorski and Honig 2012]. Considerando que a primeira sofre de limitações teóricas [Moser et al. 2007], nos ateremos, neste trabalho, à segunda.

A análise dinâmica consiste em executar o artefato em um ambiente controlado (*sandbox*) e obter um traço de execução, que pode envolver instruções executadas, acesso a memória e chamadas de funções do sistema.

Contudo, a análise dinâmica pode sofrer limitações ao analisar exemplares evasivos [Marpaung et al. 2012], que detectam a monitoração pela injeção de código ou pelos efeitos colaterais de emulação.

Neste trabalho, propomos uma abordagem para analisar exemplares evasivos. Esta consiste em utilizar recursos de monitoração por *hardware* para analisar exemplares sem a necessidade de se injetar código.

Este artigo é dividido da seguinte maneira: a Seção 2 apresenta trabalhos relacionados ao uso de recursos de *hardware*; a Seção 3 apresenta o monitor de *hardware* utilizado; a Seção 4 apresenta a solução proposta; a Seção 5 apresenta resultados preliminares obtidos com a solução proposta; por fim, a Seção 6 discute os resultados obtidos e os trabalhos futuros.

2. Trabalhos Relacionados

Abordagens como Ether [Dinaburg et al. 2008] e HyperDBG [Fattori et al. 2010] fazem uso de recursos de *hardware* para a virtualização de sistemas para implementar sistemas de instrospeção capazes de executar instruções no processador real, sendo, portanto, livres de efeitos colaterais. Tais abordagens, contudo, exigem grande esforço de implementação, já que requerem a escrita de um *hypervisor*. Além disso, técnicas de instrospecção são limitadas ao sistema para o qual foram desenvolvidos, não sendo portáveis.

Abordagens como MALT [Zhang et al. 2015] e SPECTRE [Zhang et al. 2013] fazem uso do modo SMM do processador para inserir código de monitoração de forma protegida. Dado que o modo SMM é isolado do modo real, a análise se dá de forma totalmente transparente. Entretanto, o uso do modo SMM exige a reescrita do BIOS do sistema, o que é inviável em algumas plataformas.

Abordagens alternativas, como [Willems et al. 2012] e [Kompalli and Sarat 2014], fazem uso de recursos de monitoramento de *hardware* para detectar ataques no sistema. A vantagem deste tipo de abordagem é sua facilidade de implementação, dado que não requerem a reescrita do BIOS ou de uma máquina virtual. Tais sistemas, contudo, não foram ainda utilizados para a análise de *malware*.

3. Monitoração de *branch*

Processadores Intel [1] possuem uma plataforma de monitoramento de desempenho composta de diversos módulos [Intel 2011]. Neste trabalho, nos focaremos no uso do módulo BTS (*Branch Trace Store*), responsável por realizar o *log* de instruções de desvios (`JMP`,`CALL`,`RET`). Os endereços das instruções de desvio são armazenadas pelo processador em registradores especiais (MSR) ou em uma página da memória. Neste caso, uma interrupção é gerada quando o número de desvios armazenados atinge um limiar estabelecido. Dado que a monitoração é realizada por *hardware* específico dentro do processador, tem-se um *overhead* teoricamente nulo. A coleta dos dados armazenados é realizada em espaço de *kernel*, através de um módulo/*driver*.

4. Proposta de solução

Para nossa solução, propomos utilizar o mecanismo BTS com armazenamento em páginas do sistema operacional. O sistema utilizado é o Microsoft Windows, já que este é um dos maiores alvos para infecções por *malware*. Os dados são coletados por um *driver* de *kernel* e repassados ao modo usuário para análise.

Definimos um *threshold* de captura de 1 instrução, ou seja, toda instrução de desvio é registrada e gera uma interrupção. Dada a interrupção, pode-se obter o processo

[1] Utilizados no escopo deste trabalho devido à disponibilidade

ativo naquele instante, podendo-se relacionar a instrução executada ao processo, monitorando, assim, apenas processos de interesse. Deve-se destacar que o processador não tem qualquer mecanismo para identificar o processo em execução, o que justifica nossa abordagem.

Obtidos os endereços das instruções de desvio de um dado processo, pode-se realizar a introspeção nestes. A partir do mapeamento de bibliotecas em memória, podemos identificar em qual porção de código a execução se encontra.

O programa cliente, em modo usuário, tendo estas informações, pode implementar diferentes politicas e algoritmos de análise.

5. Resultados Preliminares

Esta seção apresenta alguns resultados preliminares obtidos com a aplicação do *framework* proposto. Para facilitar a compreensão, utilizaremos o código apresentado na Listagem 1 como exemplo.

Listagem 1. Código de exemplo.

```
1 int main()
2 char name[MAX_NAME];
3 scanf("%s",name);
4 printf("Hello, %s",name);
```

O início do procedimento de análise se dá com a obtenção dos endereços e do tamanho das seções de código das bibliotecas e do código do programa sob análise, como mostrado na Listagem 2.

Listagem 2. Introspecção.

```
1 Binary: Sample.exe 7f7ccac0000 e000
2 Library: C:\Windows\System32\msvcr110d.dll
      7fc05e30000 1e2000
3 Library: C:\Windows\System32\KernelBase.dll
      7fc24050000 f3000
4 Library: C:\Windows\System32\kernel32.dll
      7fc252e0000 136000
5 Library: C:\Windows\System32\ntdll.dll
      7fc26f90000 1c0000
```

O início da análise em si se dá com o início do mecanismo de coleta. No exemplo da Listagem 3, a execução do programa já se encontra em uma biblioteca do sistema.

Listagem 3. Início do monitoramento.

```
1 Started analyzing at 7fc26f92c4a:
2 C:\Windows\System32\ntdll.dll
```

Dado que o mecanismo de monitoramento analisa todo desvio, sem qualquer filtragem, podemos capturar informações das implementações internas das bibliotecas e transições entre elas, como mostrado na Listagem 4.

Listagem 4. Código interno à biblioteca.

```
1 Code swapping from 7fc240561b9:
2 C:\Windows\System32\KernelBase.dll
3 (Unknown Function)
4 TO 7fc05f3f11b:
5 C:\Windows\System32\msvcr110d.dll
6 (_read+0xf4b)
```

Através da captura de desvios do tipo `RET` podemos identificar o fim da execução de uma função de biblioteca e a retomada da execução do código do binário sob análise, como mostrado na Listagem 5.

Listagem 5. Buffer lido.

```
1 LIB C:\Windows\System32\msvcr110d.dll
2 at 7fc05e5d4af (scanf+0x3f)
3 returned to Binary Sample.exe
4 at 7f7ccac1037
```

Observa-se, na Listagem 6, a chamada para a função `printf`, responsável por imprimir a mensagem de "Hello".

Listagem 6. Chamada para impressão da string.

```
1 Binary Sample.exe at 7f7ccac1079 called lib
2 C:\Windows\System32\msvcr110d.dll
3 at 7fc05e5c7b0 (printf)
```

Após a impressão, tem-se o retorno da função da biblioteca para o código, como mostrado na Listagem 7. Omitimos, propositalmente, as instruções internas à implementação da função `printf`.

Listagem 7. String impressa.

```
1 LIB C:\Windows\System32\msvcr110d.dll
2 at 7fc05e5c924 (printf+0x174)
3 returned to Binary Sample.exe
4 at 7f7ccac107f
```

Finalmente, como mostrado na Listagem 8, chama-se a função `exit`, que encerra o programa. Após esse ponto, nenhuma instrução mais é capturada para o programa `Sample`.

Listagem 8. Fim da execução.

```
1 Binary Sample.exe at 7f7ccac15d2 called lib
2 C:\Windows\System32\msvcr110d.dll
3 at 7fc05e35520 (exit)
```

Os exemplos apresentados ilustram a viabilidade de eficácia da solução proposta. A partir destes exemplos, construções adicionais podem ser implementadas. Casos como a omissão de instruções internas às bibliotecas serão tratados com a implementação de um mecanismo de filtragem semelhante aos mecanismos de *step into* e *step over* de *debuggers*.

6. Conclusões

Neste trabalho, apresentamos uma solução para análise de *malware* evasivo, baseada em um recursos de monitoração de performance em *hardware*. Este recurso tem *overhead* teórico nulo e não requer qualquer injeção de código no processo monitorado. Os resultados obtidos preliminarmente mostram a aplicação prática da solução proposta. Trabalhos futuros, construídos a partir do *framework* já implementado, incluem a reconstrução dos grafos de chamadas de função e de controle, bem como uma ferramenta para detecção de ataques.

Referências

Dinaburg, A., Royal, P., Sharif, M., and Lee, W. (2008). Ether: Malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 51–62, New York, NY, USA. ACM.

Fattori, A., Paleari, R., Martignoni, L., and Monga, M. (2010). Dynamic and transparent analysis of commodity production systems. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 417–426, New York, NY, USA. ACM.

FEBRABAN (2012). FEBRABAN dá dicas de segurança eletrônica. `https://febraban.org.br/Noticias1.asp?id_texto=1886`. Data de Acesso: 12/05/2016.

Intel (2011). *Intel® 64 and IA-32 Architectures Software Developer's Manual*.

Kompalli and Sarat (2014). Using existing hardware services for malware detection. In *Proceedings of the 2014 IEEE Security and Privacy Workshops*, SPW Í4, pages 204–208, Washington, DC, USA. IEEE Computer Society.

Marpaung, J., Sain, M., and Lee, H.-J. (2012). Survey on malware evasion techniques: State of the art and challenges. In *Advanced Communication Technology (ICACT), 2012 14th International Conference on*, pages 744–749.

Moser, A., Kruegel, C., and Kirda, E. (2007). Limits of static analysis for malware detection. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 421–430.

Sikorski, M. and Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, San Francisco, CA, USA, 1st edition.

Willems, C., Hund, R., Fobian, A., Felsch, D., Holz, T., and Vasudevan, A. (2012). Down to the bare metal: Using processor features for binary analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 189–198, New York, NY, USA. ACM.

Zhang, F., Leach, K., Stavrou, A., Wang, H., and Sun, K. (2015). Using hardware features for increased debugging transparency.

Zhang, F., Leach, K., Sun, K., and Stavrou, A. (2013). Spectre: A dependable introspection framework via system management mode. In *Proceedings of the 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, DSN '13, pages 1–12, Washington, DC, USA. IEEE Computer Society.

Análise de Segurança para a Descoberta, Bloqueio e Rastreamento de Tráfego Malicioso sobre a Rede Tor

Marcelo I. P Salas, Paulo Lício de Geus

[1]Laboratory of Security and Cryptography (LASCA), Institute of Computing
University of Campinas (UNICAMP)
Av. Albert Einstein, 1251, Room 84, Campinas, Brazil - CEP 13083-852

{mpalma, paulo}@lasca.ic.unicamp.br

Abstract. *Tor is an overlay network that provides anonymous communication on the Internet for TCP applications. This network serves hundreds of thousands of users, allowing them to decide when they wish to be identified or not. Although this network is substantially used to circumvent Internet censorship in countries under dictatorial regimes, the anonymity offered by the Tor network also supports, in a way, access to hidden services (e.g. Silk Road 2.0) for selling drugs, pedophilia, trafficking people, among others. This service that ensures privacy also hides a whole new side of violence, allowing botnets to go undercover, send SPAM, perform distributed denial of service attacks (DDoS), among other cybercrimes. This project proposes the research of methods and techniques to detect and classify malicious traffic in order to block potential threats and the development of techniques to trace the origin of malicious code over the Tor network. The goal is to design and implement a solution to the growing problem of malicious traffic on this network, researching forensic techniques and methods to try and protect the Tor network from malicious traffic, whilst also trying to preserve the privacy and anonymity of non-malicious traffic.*

Resumo. *Tor é uma rede de sobreposição que fornece comunicação anónima na Internet para aplicações TCP. Esta rede atende centenas de milhares de usuários, permitindo-lhes decidir quando desejam identificar-se ou não. Apesar de ser substancialmente utilizada para contornar censura na Internet em países sob regimes ditatoriais, esta rede de anonimato dá suporte, de certo modo, ao acesso a serviços ocultos (por exemplo Silk Road 2.0) fornecendo vendas de drogas, pedofilia, tráfico de pessoas, entre outros. O serviço que garante a privacidade também esconde, por trás todo um lado oculto de violência, possibilitando a proteção de botnets, envio de SPAM, ataques distribuídos de negação de serviço (DDoS), entre outros cibercrimes. Neste contexto, o presente projeto de pesquisa propõe o estudo de métodos e técnicas para detecção e classificação de tráfego malicioso, bloqueio de possíveis ameaças e desenvolvimento de técnicas de rastreamento da origem do código malicioso sobre a rede Tor. O objetivo é projetar e implementar uma solução ao crescente tráfego de código malicioso sobre esta rede, pesquisando técnicas forenses e métodos para proteger a rede Tor do tráfego malicioso, preservando a privacidade e anonimato do tráfego não malicioso.*

1. Introdução

Tor (anteriormente um acrônimo para *The Onion Router*) é uma rede de sobreposição que fornece comunicação anônima na Internet para aplicações TCP [Ling et al. 2015a]. De código aberto, esta rede atende centenas de milhares de usuários, transportando terabytes de informação cifrada, permitindo-lhes decidir quando desejam identificar-se ou não, evitando rastreamento dos seus dados online e protegendo a privacidade das suas atividades online contra tentativas de adversários de encontrá-los e destruí-los [Simons 2014].

Com mais de 6.700 servidores [Metrics 2015], a rede Tor é propensa a transportar mais de 30 vezes o tráfego malicioso em comparação com servidores que não são parte desta rede [Akamai]. Assim, o dinamismo de Tor torna a tarefa de bloquear o tráfego malicioso em um trabalho complexo para os voluntários [Ling et al. 2015a]. Este problema abre a possibilidade que os voluntários sejam legalmente processados pelo tráfego que circula por seus roteadores.

Infelizmente, os atacantes estão utilizando Tor por causa da sua proteção da privacidade nas comunicações, obtido como descrito a seguir. Através de uma aplicação, Tor seleciona, geralmente, 3 roteadores[1] da sua rede e constrói uma rota anônima utilizando um subconjunto desses roteadores. O tráfego entre o atacante e o destino é retransmitido ao longo desse percurso de forma cifrada e impossibilita o rastreamento, já que cada roteador (*onion Tor* ou *relay*) apenas conhece seu emissor anterior e receptor posterior das mensagens. Por último, o roteador de saída, atua como um *proxy*, comunicando-se diretamente com o destino de forma clara e anônima, sendo uma das poucas opções para os pesquisadores de analisar o tráfego de saída Tor.

Neste contexto, o presente projeto de pesquisa propõe uma análise de segurança para o estudo e implementação de métodos e técnicas para detecção e classificação de tráfego malicioso, bloqueio de possíveis ameaças e desenvolvimento de técnicas de rastreamento da origem do código malicioso sobre a rede Tor. O objetivo é projetar e implementar soluções ao crescente tráfego de código malicioso sobre esta rede, pesquisando técnicas forenses e ferramentas para protegê-la do tráfego malicioso, preservando a privacidade e anonimato dos usuários. A combinação de técnicas (p.ex. análise de padrão do tráfego malicioso e análise forense de ataques) e ferramentas (p.ex. IDSs[2] e analisadores de tráfego) coadjuvarão no desenvolvimento de uma plataforma para avaliar em tempo real possíveis ameaças circulando pela rede Tor, ajudarão no desenvolvimento de técnicas para a análise forense dos ataques e colaborarão na prevenção de futuras ameaças de segurança contra a Internet no Brasil e no mundo [Martins et al.].

Dadas as políticas da *Digital Millennium Copyright Act* (DMCA), que monitora constantemente os roteadores de saída Tor e envia notificações contra o compartilhamento de materiais com direitos autorias. Nossa arquitetura encaminhará o tráfego do roteador de saída através de um firewall para outro roteador de entrada (*guard onion*). Desta forma, haverá condições de bloquear tráfego capaz de possível imputação de responsabilidade legal sobre a universidade. Lamentavelmente, o reencaminhamento do tráfego gerará atraso na comunicação entre os usuários e os servidores, no entanto este é um custo quase

[1] A configuração padrão da rede Tor é composta por 3 roteadores: um de entrada (*entry guard*), outro de saída (*exit router*) e um roteador intermediário. Existem outras configurações que serão descritas nos capítulos seguintes.

[2] Intrusion Detection Systems, i.e. sistemas de detecção de intrusão

sempre presente ao se pesquisar tráfego malicioso em ambientes sensíveis.

Pretende-se enfatizar três frentes de pesquisa, envolvendo: **i) análise de tráfego malicioso em Tor**, o que inclui identificação, classificação e bloqueio deste tráfego utilizando técnicas e ferramentas; **ii) botnets**, com o objetivo de detectar, analisar e bloquear suas atividades maliciosas sobre a rede Tor, tais como DDoS, SPAM, roubo de informação e outros [Wang et al. 2015]; e **iii) Rastreamento de tráfego malicioso sobre Tor**, com o objetivo de tentar localizar os centros de C&C das botnets e outros servidores na *Deep Web*. O desenvolvimento da plataforma permitirá fazer análise estatística de tráfego malicioso sobre a rede Tor. Isto será possível pela utilização de ferramentas de IDS e análise de reconhecimento de padrões de tráfego para botnet, SPAM e outros. Além de mitigar tráfego malicioso, projeta-se implementar mecanismos para identificar novas ameaças, bloqueá-las e tentar estimar futuros cenários. De certo modo, isto permitirá categorizar o tráfego malicioso sobre a rede Tor.

2. Proposta

O propósito do projeto de pesquisa é melhorar a segurança de Tor e ajudar a reduzir a ciberdelinquência desta rede de anonimato e privacidade através de técnicas de análise de tráfego de um roteador de saída Tor.

Isto inclui a pesquisa para configurar uma rede de captura e reencaminhamento do tráfego, além dos sistemas de análise, monitoração, localização e proteção contra o tráfego malicioso. Para alcançar esse objetivo, propõe-se utilizar ferramentas como analisadores de tráfego, IDS e técnicas de aprendizado de máquina para capturar comportamento malicioso. Além disso, é necessário desenvolver uma ferramenta de identificação e bloqueio de tráfego malicioso [Cavalcante et al. 2014], que possa ser utilizada por voluntários da rede Tor. Enquanto esses recursos são compatíveis para os objetivos propostos, podem também ser de interesse para outras redes de privacidade e anonimato.

Esta seção apresenta as atividades já realizadas, referentes a cada uma das etapas citadas no capítulo anterior, ou seja, coleta, análise e classificação, e bloqueio e rastreamento de tráfego malicioso em Tor, seguindo nossa propostas descrita na Figura 1.

2.1. Coleta

A seguir, serão apresentados os métodos de coleta de tráfego da rede Tor utilizados. Estas estão classificadas por amostras de fontes públicas, fontes privadas e amostras coletadas por analisador de tráfego de rede.

Fontes Públicas. Foram obtidas amostras de tráfego malicioso e benigno de páginas Web, tais como "Contagiodump"[3], "Codeplex"[4], "Netresec"[5], "Pcapr"[6], "Packetlife"[7] e "wireshark"[8]. Estas páginas publicam novos exemplares continuamente. Neste caso, faremos o processo de obtenção deles contínuo durante o período do doutorado.

[3] `http://contagiodump.blogspot.com.br`

[4] `http://sysdoccap.codeplex.com/wikipage?title=System\%20Overview\%20Document\%20Scenario\%20Captures`

[5] `http://www.netresec.com/?page=PcapFiles`

[6] `http://www.pcapr.net/home`

[7] `http://packetlife.net/captures/`

[8] `https://wiki.wireshark.org/SampleCaptures\#Sample\_Captures`

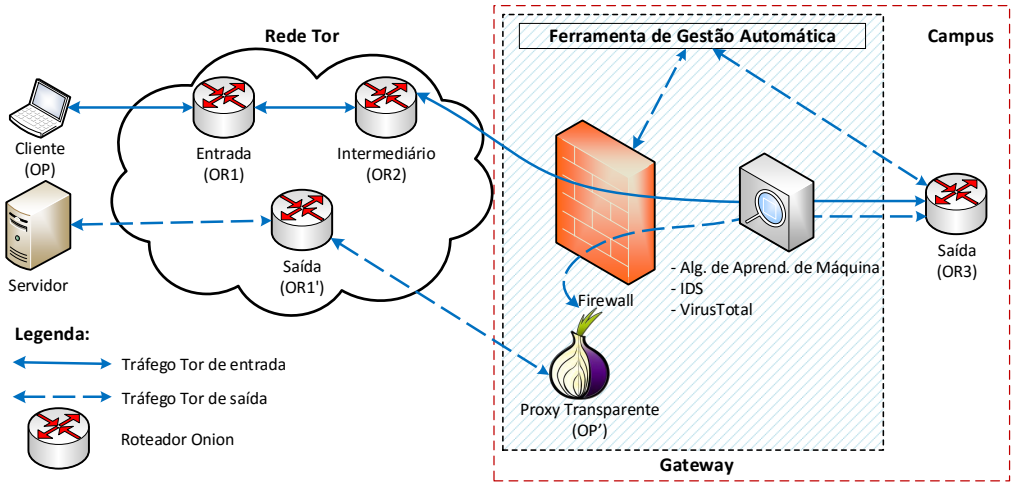


Figure 1. Arquitetura da plataforma de Coleta de Tráfego Malicioso para a rede Tor [Ling et al. 2015b].

Fontes Privadas. Compostas por uma combinação de amostras de tráfegos maliciosos e benignos, pertencentes ao banco de dados ISOT [Net 2011]. O tráfego malicioso inclui amostras de botnets Zeus, Storm e Waledac. Outras fontes de tráfego benigno foram obtidas de dois diferentes bancos de dados, um a partir do projeto *"Traffic Lab"* do centro de pesquisa Ericsson da Hungria [ICIR] e outro do *"Lawrence Berkeley National Lab (LBNL)"* [Orebaugh et al. 2006].

Coleta por Analisador de Tráfego de Rede. Será implementada uma interface de coleta de amostras de tráfego malicioso e benigno fazendo uso de um roteador conectado à rede Tor, de preferência para um roteador de saída (*exit router*).

2.2. Análise e Classificação

Foi desenvolvido um sistema de análise dinâmica, em ambiente emulado (honeypot), para analisar o padrão de comportamento dos *malware* nas plataformas Windows XP e 7. Para isso, foi selecionado um conjunto de 125 amostras de malware pertencentes a 21 famílias dentre vírus, cavalos de Troia e botnets, e as 125 aplicações mais baixadas do repositório CNET. Foram aplicados dois algoritmos de aprendizado de máquina, *Support Vector Machine* (SVM) e *Boosting* com Árvores de Decisão. Utilizou-se a técnica *Principal Component Analysis* (PCA) para evitar a Maldição da Dimensionalidade e eliminar as características que não contribuíssem para a separação das classes.

O software malicioso é caracterizado por um comportamento complexo e diversificado, que vai desde simples modificações de recursos do sistema até atividades vinculadas a redes externas. Variantes de malware de uma mesma família compartilham padrões de comportamento comuns, tais como o uso de mutações de sua carga útil, modificações de arquivos de sistema específicos, uso não típico de bibliotecas do sistema e acesso a redes externas, caracterizadas pelos seus endereços IP.

Assim, nosso objetivo foi explorar esses padrões comuns para análise automática com vistas à detecção de malware baseada no seu comportamento perante o sistema op-

eracional Windows, versões XP e 7.

A técnica de Boosting mostrou ser superior a SVM para este caso, alcançando melhores resultados tanto para o Windows 7 quanto para o Windows XP, com acurácias de até 92% e 94%, respectivamente.

Esta pesquisa demostrou que a utilização de poucas amostras combinadas com PCA para redução de dimensionalidade permite obter resultados satisfatórios e aumenta a credibilidade dos resultados da pesquisa. Além disso, é possível aplicar esta metodologia para análise de *malware* recuperado do tráfego malicioso da rede Tor. Também esperamos obter *malware* para outros plataformas, tais como Android.

2.3. Outras atividades

Os Testes de Segurança permitem avaliar as vulnerabilidades em aplicações e serviços frente a diversos tipos de ataques e descobrir novas vulnerabilidades antes que sejam exploradas por atacantes [Salas and Martins 2012]. Nesta pesquisa, estas técnicas permitiriam injetar diversos tipos de ataques contra os roteadores Tor e seus serviços ocultos, permitindo encontrar vulnerabilidades para poder rastrear serviços ilícitos da *Deep Web*, bloquear o tráfego malicioso das botnets ou injetar ataques nos roteadores de saída.

Entre as técnicas de Testes de Segurança que podem ser utilizadas estão: Testes de Penetração e Injeção de Falhas. Os Testes de Penetração (TP) emulam ataques, com o objetivo de revelar vulnerabilidades. Estes testes são automatizados pelo uso de ferramentas denominadas *vulnerability scanners* (VS). Já a Injeção de Falhas é uma técnica que pode ser utilizada para avaliar aspectos de dependabilidade dos sistemas de computação, podendo ser implementada em hardware ou software, para emular anomalias, defeitos ou erros no sistema alvo e observar seu comportamento sob um ambiente estressante.

A vantagem de usar Injeção de Falhas com Testes de Penetração é que a primeira permite maior cobertura de ataques. Neste contexto, as falhas são introduzidas por um injetor—software responsável por injetar falhas no sistema—antes ou durante a execução. Assim, é possível modificar os ataques dinamicamente para encontrar diversas vulnerabilidades utilizando diferentes ambientes de testes, i.e. analisar as vulnerabilidades em função do tipo de serviço que presta o roteador Tor (entrada, intermediário, saída, diretório, Ponto de Encontro e Ponto de Introdução). Por enquanto, o principal objetivo de atacar roteadores de saída é bloquear o tráfego malicioso, ao passo que o de atacar serviços ocultos é revelar os endereços IP.

Para utilizar estes ataques por Injeção de Falhas, utilizaremos testes constituídos por dois conjuntos de entrada: a carga de trabalho (workload) e a carga de falhas (faultload) [Hsueh et al. 1997]. A primeira representa as entradas usuais do sistema, que servem para ativar suas funcionalidades, enquanto a segunda representa as falhas a serem introduzidas no sistema. Para isso, contamos com um conjunto de 36 tipos de ataques que serão testados contra um dos ambientes Tor.

References

Akamai. The q2 state of the internet security report. `www.stateoftheinternet.com/security-report`.

Cavalcante, T., Rocha, A., and Carvalho, A. (2014). Large-scale micro-blog authorship attribution: Beyond simple feature engineering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 399–407. Springer.

Hsueh, M.-C., Tsai, T. K., and Iyer, R. K. (1997). Fault injection techniques and tools. *Computer*, 30(4):75–82.

ICIR. Lbnl/icsi enterprise tracing project. `http://www.icir.org/enterprise-tracing/`.

Ling, Z., Luo, J., Wu, K., Yu, W., and Fu, X. (2015a). Torward: Discovery, blocking, and traceback of malicious traffic over tor. *Information Forensics and Security, IEEE Transactions on*, 10(12):2515–2530.

Ling, Z., Luo, J., Wu, K., Yu, W., and Fu, X. (2015b). Torward: Discovery, blocking, and traceback of malicious traffic over tor. *IEEE Transactions on Information Forensics and Security*, 10(12):2515–2530.

Martins, Gilbert B, S. E., de Freitas, R., and Feitosa, E. Estruturas virtuais e diferenciação de vértices em grafos de dependência para detecção de malware metamórfico.

Metrics, T. (2015). Tor metrics. `https://metrics.torproject.org/`.

Net, H. (2011). The honey project france chapter. `http://www.honeynet.org/chapters/france`.

Orebaugh, A., Ramirez, G., and Beale, J. (2006). *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress.

Salas, M. I. P. and Martins, E. (2012). *Metodologia de testes de seguranÃ§a para anÃ¡lise de robustez de Web services por injeÃ§Ã£o de falhas (Security testing methodology for robustness analysis of Web services by fault injection)*. PhD thesis.

Simons, J. W. (2014). A rede secreta. `http://www.publico.pt/tecnologia/noticia/a-rede-secreta-1673221`.

Wang, A., Mohaisen, A., Chang, W., and Chen, S. (2015). Delving into internet ddos attacks by botnets: Characterization and analysis. In *IEEE International Conference on Dependable Systems and Networks (DSN)*.

Aprendizado de Índices Espectrais com Programação Genética

Juan Felipe Hernández Albarracín, Ricardo da Silva Torres

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 13083-852 – Campinas – SP – Brazil

juan.albarracin@students.ic.unicamp.br, rtorres@ic.unicamp.br

Abstract. *This work introduces a genetic-programming-based approach for spectral-index learning through band selection and combination in remote sensing images. This technique aims to solve both classification and regression problems. Preliminary experiments regard binary pixel-wise classification of hyperspectral scenes are described. Experimental results demonstrate that the proposed method is effective in separating samples of two different classes. Ongoing work focused on the ensemble of the binary classifiers into a multi-class one is presented as well.*

Resumo. *Apresenta-se uma abordagem de programação genética para aprendizado automático de índices espectrais através de seleção e combinação de bandas em imagens de sensoriamento remoto. A técnica objetiva resolver problemas de classificação e regressão. Os resultados preliminares obtidos até agora são referentes à classificação binária de pixels em cenas hiper-espectrais e mostram que o método proposto é efetivo para separar amostras de duas classes diferentes. Apresenta-se também trabalho em andamento voltado à combinação de classificadores binários em um multi-classe.*

1. Introdução

Sensoriamento remoto é a prática de deduzir informação da superfície terrestre usando imagens adquiridas à distância com radiação refletida ou emitida em diferentes regiões do espectro eletromagnético [Campbell and Wynne 2011]. O grupo de imagens de uma mesma região geográfica em diferentes bandas espectrais, é conhecido como *imagem (ou cena) espectral*. A adquisição de imagens em regiões além das regiões visíveis do espectro eletromagnético é de uma grande importância, dado que substâncias químicas podem ser detectadas através da sua *assinatura espectral* [Campbell and Wynne 2011].

Índices espectrais são funções que associam bandas espectrais com características físicas e químicas de objetos do entorno. Um exemplo relevante é o NDVI (acrônimo em inglês para *índice de vegetação de diferença normalizada*) para medição de biomassa, baseado na diferença ente a reflectância de um objeto no canal vermelho e no canal infra-vermelho próximo. Apesar de serem normalmente formulados por especialistas, os índices espectrais podem ser deduzidos automaticamente por algoritmos de reconhecimento de padrões, sustentados por dados de treinamento.

A técnica proposta consiste de um arcabouço de programação genética cujo objetivo é fazer seleção e combinação das bandas de cenas espectrais, criando um índice apropriado para um objetivo específico. Classificação de imagens espectrais normalmente

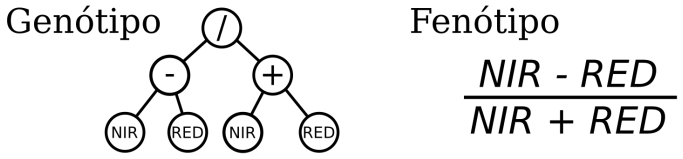


Figura 1. Exemplo de indivíduo GP com o índice NDVI. O genótipo corresponde à sua representação computacional (árvore sintática) e o fenótipo, á sua interpretação (fórmula).

tem que lidar com os problemas da alta dimensionalidade e ruído. A busca de um índice espectral visado a resolver o problema de classificação é uma técnica de extração de características que resolve estes problemas. Índices espectrais aprendidos automaticamente são capazes de fornecer informação de maior utilidade para propósitos específicos do que índices formulados por especialistas. A principal hipótese deste trabalho é definida da seguinte forma: contando com informação de treinamento, é possível encontrar uma combinação de bandas espectrais que consiga separar entidades de categorias diferentes (e.g., presença/ausência de vegetação, alta/baixa concentração de nitrogênio, região com espécie *X* ou *Y*).

2. Background e trabalhos relacionados

Programação genética (GP) é uma técnica de aprendizado automático que pertence à família dos algoritmos evolutivos [Koza 1992], em que soluções candidatas de um problema complexo são representadas como indivíduo dentro de uma população. Um *fitness* (ou função objetivo) é estabelecido como critério para selecionar e manter as soluções que melhor resolvem o problema, conforme o princípio darwiniano de supervivência do mais apto. Em GP, particularmente, os indivíduos são programas de computador ou fórmulas, cujo *fitness* depende do resultado da sua execução. Normalmente, os indivíduos são representados como árvores sintácticas (Figura 1), e a sua evolução é feita fundamentalmente por dois operações, para a criação de novos indivíduos: recombinação e mutação. Recombinação é o operador pelo qual a informação genética de dois ou mais indivíduos é intercambiada; no caso da GP, dois indivíduos intercambiam subárvores. Os novos indivíduos resultantes têm uma chance de sofrer mutação, ao trocar arbitrariamente um dos seus subárvores por um gerado aleatoriamente.

Várias pesquisas têm se focalizado no uso da GP para combinação de bandas espectrais. Normalmente, visam a encontrar formulas que medem concentração de uma substância química ou a presencia de um objeto. O método proposto em [Fonlupt and Robilliard 2000], por exemplo, usa a GP para avaliar a concentração de fitoplâncton, sedimento e matéria orgânica dissolvida em oceanos e águas costeiras. Fazendo uso de uma função-objetivo variante, com a qual se optimizava por cada classe em etapas, resultados superiores às aproximações polinomiais tradicionais foram conseguidos.

Em [Chion et al. 2008], apresenta-se o índice espectral de vegetação por programação genética (GP-SVI): um método baseado em GP para evoluir um modelo de regressão que descreve o nível de nitrogênio em vegetação. O *fitness* considera tanto a força da correlação com a informação adquirida *in situ*, quanto o tamanho da fórmula. Os resultados foram bastantes satisfatórios, mostrando a GP-SVI como o me-

lhor, quando comparado com outros métodos. Um trabalho bastante similar foi apresentado em [Puente et al. 2011], em que foi apresentado um índice espectral de vegetação por programação genética (GPSVI) para estimar o fator de cobertura vegetal em solos. O *fitness* baseava-se na covariância entre a fórmula do indivíduo e o *fator C* (*cover-management factor*) adquirido analiticamente a partir de outros sub-fatores medidos *in situ*. Os resultados foram comparados com uma seleção de índices espectrais de vegetação reconhecidos, aos quais a técnica proposta ultrapassou.

Em [Ross et al. 2005], apresenta-se uma abordagem para resolver o problema de classificação de minerais: várias execuções do algoritmo de GP treina separadamente classificadores binários individuais para cada uma das classes. Os indivíduos em vez de representar fórmulas, são árvores de decisão sobre combinações aritméticas das bandas espectrais.

Diferente das iniciativas já mencionadas, a técnica proposta usa GP para classificação e regressão, baseando-se na qualidade do espaço de características gerado pela combinação de bandas fornecida pelo algoritmo.

3. Método proposto

Nós propomos uma técnica baseada em GP que visa encontrar um índice de vegetação apropriado para propósitos de classificação o regressão. Neste artigo, apresentamos a configuração do método para classificação de pixeis em imagens espectrais, e encontrar o índice espectral que maximize a acurácia.

A Figura 2 descreve o fluxograma geral do método. Dado uma conjunto de cenas espectrais com regiões etiquetadas, o algoritmo de GP procura uma combinação aritmética das bandas. Cada fórmula gera uma única imagem em escala de cinza, quando aplicada sobre as bandas da imagem. O critério para avaliar cada fórmula baseia-se nas distâncias intra e inter-classe dos pixels resultantes. A melhor fórmula encontrada é usada como entrada a um sistema simples de classificação que é treinado com o conjunto de treino e teste.

3.1. Arcabouço de GP

Cada fórmula candidata é representada por uma árvore sintáctica, cujos nós internos representam operações aritméticas, e as folhas representam constantes ou bandas espectrais. A função de *fitness* determina como as amostras de diferentes classes são separáveis dentro do espaço de características gerado pela fórmula. O *fitness* permite ordenar os indivíduos e atribuir uma maior chance de serem selecionados aos melhores.

3.1.1. Função de *fitness*

Considerando cada classe como uma distribuição de valores de pixels, a medida usada para avaliar a sua separação no espaço criado por um índice espectral é a Silhueta [Rousseeuw 1987].

Seja a *dissimilaridade média* de um objeto x a uma distribuição C_i, a média das distâncias de x a todos os objetos que pertencem a C_i. Seja $a(x)$ a dissimilaridade média

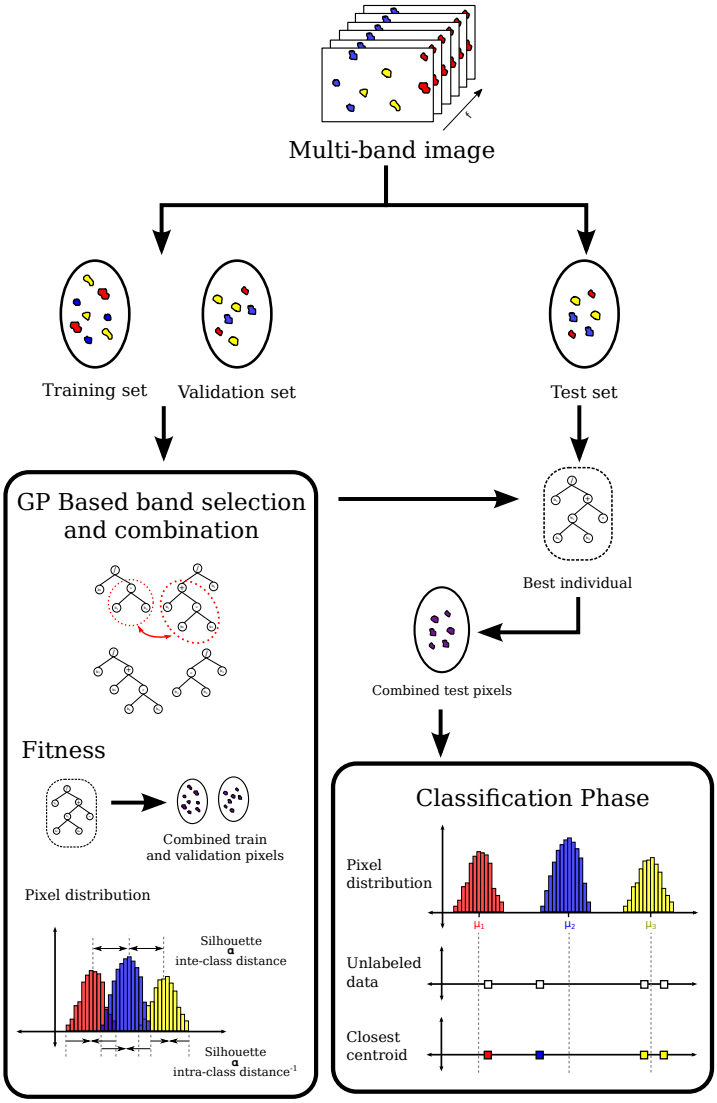


Figura 2. Fluxograma do método. Os pixels etiquetados de uma imagem espectral são separados em conjuntos de treino, validação e teste. Treino e validação são usados no arcabouço de GP. O melhor indivíduo é usado para combinar as bandas dos pixels de teste e atribuir a uma classe cada pixel conforme à distância aos centroides das classes.

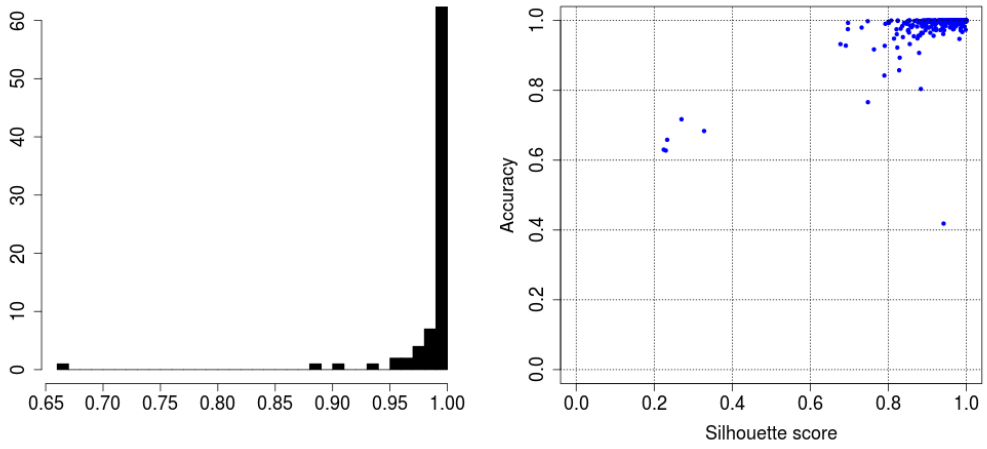
de x a sua própria distribuição e $b(x)$, a menor dissimilaridade média de x às demais distribuições. A silhueta do objeto x é definida por:

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \tag{1}$$

acarretando valores entre -1.0 e 1.0. O valor geral da silhueta é a média das silhuetas de todos os objetos etiquetados. Valores próximos a 1.0 significam distribuições mais compactas e afastadas entre elas. O objetivo é, portanto, maximizar este valor.

Tabela 1. Parâmetros do arcabouço de GP

Parâmetro	**Valor**	**Parâmetro**	**Valor**
Tam. da pop.	100	Parâmetros	$\{b_i : 0 \leq i < n\} \cup \{c_j \in [0, 10^6]\}$
Num. de gens.	200	Operadores	{+, -, *, %}
Taxa de recombinação	0,9	Método	Torneio $\times$ 3
Taxa de mutação	0,1	de Seleção	



(a) (b)

Figura 3. Histograma de acurácias para os pares de classes (a), e visualização da correlação entre a medida de silhueta e a acurácia da classificação.

3.2. Etapa de classificação

A classificação é feita com o índice encontrado na fase anterior. Um pixel sem classe, após ser combinado conforme ao índice, é atribuído à classe cuja média for a mais próxima. A média de uma classe é calculada a partir dos dados de treinamento que pertencem à mesma.

4. Protocolo experimental

Neste artigo apresentamos resultados para classificação binária no data set Salinas, uma cena hiper-espectral de 224 bandas e etiquetada com 16 classes, coletada no vale Salinas, Califórnia[1]. A Tabela 1 mostra a configuração do arcabouço de GP. O operador $\%$ significa *divisão protegida*, robusta no caso de o denominador for igual a 0. Foi utilizada validação cruzada em 5 *fold* para avaliar a efetividade da classificação.

5. Resultados

Um total de 120 execuções do algoritmo foram feitas, correspondentes a cada par de classes que podem ser selecionadas do conjunto de 16. A Figura 3-a mostra a distribuição das

[1]Esta coleção foi obtida em `http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes` (Último acesso em 22/06/2016).

acurácias resultantes, após a validação 5-fold. Como pode ser visto, a metade dos pares obtiveram uma separação com acurácia de 100%, é só 4 obtiveram uma acurácia menor que 95%, dos quais um par obteve uma acurácia baixa. A média total da distribuição é $0,99026$. A Figura 3-b mostra a correlação entre o valor da silhueta do melhor indivíduo com a acurácia correspondente na fase de classificação. O comportamento observado indica que a correlação é positiva e poderia estabelecer-se empiricamente que, para este data set, silhuetas de $0,8$ são suficientes para obter acurácias por cima de 95%.

Posteriormente, os classificadores para cada par foram integrados a partir do useo de uma abordagem de fusão do tipo OVO (*one versus one*), resultando em uma acurácia de $0,8444$ e uma acurácia normalizada de $0,88776$.

6. Conclusões e trabalho a seguir

Um método baseado em programação genética de aprendizado de índices espectrais para classificação de pixeis em imagens multi-banda foi apresentado. Experimentos executados para classificação binária mostram que é possível encontrar combinações de bandas para diferenciar elementos das imagens, segundo um critério específico. Existem classes cuja separação não é tão trivial, e demandam uma exploração mais exaustiva do espaço de soluções. Os passos a seguir compreendem o ajuste dos parâmetros do algoritmo GP para mudar a exploração do espaço, analise de outras funções objetivo, e o estudo de métodos de combinação de classificadores visando tarefas de classificação multi-classe.

7. Agradecimentos

Os autores agradecem CAPES, CNPq e o Instituto Virtual FAPESP-Microsoft (proc. #2013/50155-0 e #2013/50169-1) pelo apoio.

Referências

Campbell, J. B. and Wynne, R. H. (2011). *Introduction to Remote Sensing, Fifth Edition*. Guilford Publications, New York, NY, USA.

Chion, C., Landry, J.-a., and Da Costa, L. (2008). A genetic-programming-based method for hyperspectral data information extraction: Agricultural applications. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(8):2446–2457.

Fonlupt, C. W. B. and Robilliard, D. (2000). Genetic Programming with Dynamic Fitness for a Remote Sensing Application. *PPSN*, pages 191–200.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Puente, C., Olague, G., Smith, S. V., Bullock, S. H., Hinojosa-Corona, A., and González-Botello, M. A. (2011). A genetic programming approach to estimate vegetation cover in the context of soil erosion assessment. *Photogrammetric Engineering & Remote Sensing*, 77(4):363–376.

Ross, B. J., Gualtieri, A. G., Fueten, F., and Budkewitsch, P. (2005). Hyperspectral image analysis using genetic programming. *Applied Soft Computing*, 5(2):147–156.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65.

Ataques de Canal Lateral em Aplicações de Internet das Coisas

Lucas Zanco Ladeira[1] **, Ricardo Dahab**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brazil

lucas.zanco.ladeira@gmail.com

Abstract. *A known class of attacks in cryptography is that of side-channel attacks, where it is possible to analyze algorithms from the point of view of tiny fluctuations of the running time, energy consumption and magnetic field fluctuations, among other telltale signs. In this kind of attack, the greater the amount of information available to the adversary, the easier it is to obtain secret data; therefore, it is a critical factor in devices running Internet of Things' applications, which trade a huge amount of information with other devices. The goal of this article is to explore and evaluate, at a conceptual level, the security level of a few common Internet of Things' applications from the point of view of side-channel threats.*

Resumo. *Uma classe de ataques conhecida é a de canais laterais, onde é possível analisar algoritmos do ponto de vista de pequenas variações no tempo de execução, consumo de energia, e variação no campo magnético, entre outros sinais indicadores. Nesse tipo de ataque, quanto maior a quantidade de informação ao adversário, mais fácil é obter dados secretos; sendo assim, é crítico em dispositivos utilizados em internet das coisas, que transmitem grande quantidade de informação com outros dispositivos. O intuito desse artigo é de explorar e avaliar, de forma conceitual, a segurança existente em algumas aplicações de internet das coisas do ponto de vista de ameaças de canal lateral.*

1. Introdução

Dependendo de como são implementados algoritmos criptográficos, existem certas vulnerabilidades que podem "vazar" informações sobre o texto claro, ou até mesmo da chave privada em uso. Um tipo de ataque conhecido é o de canal lateral, que faz a análise de informações físicas do sistema e não por meio da força bruta ou exploração de vulnerabilidades teóricas do esquema. Esse ataque se vale de dados vazados pela análise do tempo de execução das operações matemáticas, análise da variação no consumo de energia do circuito, campo magnético criado durante a execução, entre outros [Kang et al. 2016]. Esses ataques se beneficiam tanto de algoritmos implementados em hardware como em software.

Levando em consideração as vulnerabilidades exploradas por esse tipo de ataque, as implementações devem utilizar técnicas para se tornarem resistentes. Essas técnicas compreendem a execução de operações que apenas aumentam o tempo de execução sem modificar o resultado, operações matemáticas que não diferenciam a execução dependendo de uma determinada informação ou até mesmo a utilização de máscaras

[Kang et al. 2016].

Para esse trabalho é necessário analisar o impacto da utilização desses ataques em sistemas de internet das coisas (IoT). Esses sistemas utilizam sensores para analisar ambientes e geralmente criam grande quantidade de informação [Yan et al. 2014]; para ataques de canal lateral, quanto maior a quantidade de informações mais fácil é obter dados que deveriam ser secretos. Sendo assim, são críticos dentro de aplicações de internet das coisas.

O intuito desse artigo é de explorar, de forma conceitual, a segurança existente em aplicações de internet das coisas, e também avaliar o impacto da falta dela em cenários reais. É necessário levar em consideração a capacidade de um sistema embarcado de executar técnicas de segurança em relação a velocidade de processamento, gasto de energia, e o custo sobre a implementação de ser resistente a ataques de canal lateral [Trappe et al. 2015]. Sendo assim, é possível avaliar cada dispositivo dentro de um sistema separadamente, e quais ataques de canal lateral podem ser utilizados para obter informações sobre a chave secreta.

2. Ataques de Canal Lateral

Como já foi dito na introdução, ataques de canal lateral utilizam dados físicos do dispositivo para obter informações utilizadas no processamento [Kang et al. 2016]. O vazamento dessas informações ocorre pela análise do tempo de execução, potência, campo magnético, ou até mesmo pelo som. Levando em consideração, a grande quantidade de ataques diferentes [Ambrose et al. 2015], iremos focar apenas em três: ataque por tempo, potência elétrica e campo magnético.

2.1. Ataque por Tempo

Ataque por tempo é baseado na análise do tempo de execução de rotinas criptográficas para obter a chave secreta [Kang et al. 2016]. Isso é possível partindo de várias entradas diferentes e analisando a diferença no tempo de execução do algoritmo. Sendo que, o tempo total é o tempo final subtraído o tempo inicial da primitiva criptográfica, e não de cada operação executada pela rotina.

2.2. Ataque por Potência Elétrica

De acordo com [Kang et al. 2016] o ataque de canal lateral mais eficiente é o de análise de potência elétrica. Esse ataque utiliza o modelo de peso e de distância de Hamming para analisar o consumo de energia, e obter a relação entre a variação de potência e os *bits* computados. O peso de Hamming, de forma simplificada, é a quantidade de *1*'s em um vetor de *bits* [Morancho 2014]. Caso o vetor não possua apenas *1*'s e *0*'s, o peso é a quantidade de elementos diferentes de *0*. Já a distância de Hamming é a quantidade de elementos que diferem entre si entre dois vetores de bits [Fan et al. 2013].

O ataque de potência faz a análise do perfil da potência durante a cifração de um dado, em busca de padrões para advinhar a chave secreta. Esse ataque é dividido em três tipos diferentes: simples (SPA), diferencial (DPA) e correlacional (CPA). Cada um deles se difere na análise efetuada no perfil, aumentando a complexidade de implementação e a eficiência do ataque.

2.3. Ataque por Campo Magnético (EM)

Dispositivos computacionais, sejam embarcados ou não, utilizam CMOS para armazenar suas informações de inicialização. O fluxo de dados que passa por esse semicondutor gera radiação eletromagnética. Essa radiação pode ser obtida e analisada para modelar ataques de canal lateral [Shah et al. 2015]. Ataque por campo magnético, como o próprio nome já infere, faz a análise do campo magnético gerado pelo circuito durante a execução do módulo de criptografia para obter a chave secreta. A grande diferença entre ataques pela análise da potência elétrica e pela análise do campo magnético é a de que o segundo não necessita de contato com o circuito.

3. Contramedidas à Ataques de Canal Lateral

Como as vulnerabilidades exploradas por ataques de canal lateral estão ligadas à como o algoritmo foi implementado, existem contramedidas para tornar resistentes essas implementações [Kang et al. 2016]. As que serão abordadas nesse trabalho compreendem: *code injection*, *masking* e *shuffling*. Essas contramedidas podem aumentar substancialmente o tempo para cifrar um dado, sendo que, no caso de *masking* a cada operação é necessário mascarar e desmascarar a chave.

3.1. *Code Injection*

A primeira contramedida que iremos abordar é a adição de operações matemáticas e variáveis que não modificam o resultado final, e tornam o tempo constante não importando o bit da chave [Shah et al. 2015]. Um exemplo é o caso da soma de um bit da chave com algum bit do texto claro, levando em consideração que a operação de soma pode variar o tempo de execução dependendo do bit da chave, é possivel adicionar operações na tentativa de manter o tempo variando independente desse dado.

3.2. *Masking*

Essa contramedida funciona da seguinte maneira: antes de cada operação é gerado um número aleatório, esse número aleatório é utilizado para mascarar a chave privada. A operação matemática é executada, e então, é a chave privada é desmascarada. Caso ocorra um ataque de canal lateral a informação obtida pelo ataque é a chave mascarada, sendo que em cada operação diferente é necessário gerar um número pseudo-aleatório novo, não é possível descobrir qual é a máscara e desmascarar a informação otida para recuperar a chave privada [Kang et al. 2016].

3.3. *Shuffling*

A contramedida de *shuffling* é efetiva contra ataques de análise de potência, pois embaralha as instruções do código a serem executadas. Existem dois passos importantes nessa contramedida: o primeiro é a extração de dependências, avaliando cada instrução para verificar se ela pode ser trocada de lugar com outra sem mudar o resultado final da execução. O segundo é a geração de uma lista aleatório das instruções para a execução [Luo et al. 2015].

4. Aplicações de Internet das Coisas

Iremos apresentar algumas aplicações de IoT, sendo que essas soluções possuem alta taxa de quebra de privacidade caso não utilizem segurança. É necessário considerar que muitas aplicações de IoT ainda não apresentam segurança no estado da arte, sendo assim, iremos avaliar a inserção de algoritmos criptográficos e contramedidas à ataques.

4.1. Aplicação 1 - *Ambient Assisted Living*

O primeiro sistema de internet das coisas que iremos abordar foi apresentado por [Tragos et al. 2015], o mesmo tem o objetivo de gerenciar um edifício para dar apoio a soluções médicas, focando em pessoas idosas e/ou com deficiências. Esse tipo de aplicação é classificada como *Ambient Assisted Living* (AAL), e um dos principais aspectos é o gerenciamento do sistema de acordo com regras para minimizar a interação humana.

4.2. Aplicação 2 - Passaporte Eletrônico

O trabalho de [Hamad et al. 2015] faz uma análise do passaporte eletrônico, e apresenta uma maneira de autenticar vários passaportes ao mesmo tempo. Os passaportes eletrônicos fazem o uso de *chips* RFID para armazenar dados biométricos do portador, e utilizar esses dados para autenticar com um terminal no aeroporto. Esses *chips* não necessitam de contato para se comunicar com algum dispositivo, sendo assim, é possível utilizar sensores próximos. Para autenticar um passaporte é necessário que o mesmo se autentique com pelo menos três sensores próximos, e assim diminuindo a margem de erro da aplicação.

5. Análise de Vulnerabilidades

No sistema de internet das coisas de AAL são utilizados múltiplos sensores para observar as mudanças no ambiente. Considerando a utilização de um esquema simétrico é necessário utilizar contramedidas para ataques de canal lateral para manter o sistema seguro. Sendo que, caso seja utilizado apenas um par de chaves para todos os dispositivos, isso torna possível a decifração e modificação de qualquer informação trocada com o servidor caso algum dos sensores não utilizar contramedidas eficasez. Devemos citar que o ataque a ser utilizado depende da necessidade do atacante, sendo que com maior taxa de acerto dos *bits* da chave requer um ataque com maior complexidade.
O outro trabalho a ser analisado faz o uso da tecnologia de RFID em passaportes eletrônicos. Considerando a escassez de energia que ocorre no RFID e o custo de ciclos do processador apresentado no trabalho de [David McCann 2015] a implementação de contramedidas se torna cara, sendo que os *chips* não possuem grande quantidade de memória, e o processamento deve ser eficiente para autenticar com três sensores próximos.

6. Nossa Contribuição

Nossa contribuição parte do princípio de que aplicações de IoT não tem especificada a segurança utilizada, isso para as que utilizam algum tipo de algoritmo de criptografia. Queremos propor um modelo para análise da segurança de uma arquitetura IoT dividindo a necessidade de cada dispositivo, levando em consideração ataques de canal lateral.

Sendo assim, propomos três níveis de segurança: o primeiro chamado de NS (*not*

secure) é o vermelho da figura 3, que significa a ausência de algoritmos criptográficos. Nesse nível qualquer dado enviado pela rede sem fio pode ser obtido, alterado, ou apenas lido. O segundo, chamado de CS (*cryptographic secure*), é representado pelo círculo verde, que significa que o dispositivo utiliza algoritmos criptográficos, não considerando se é assimétrico ou simétrico. Dessa maneira, não é possível decifrar o dado sem ter conhecimento da chave secreta. O último nível, chamado de SCAS (*side-channel attack secure*), o circulo azul, possui implementações seguras de algoritmos criptográficos contra ataques de canal lateral.

Levando em consideração os três tipos de sensores da figura 3, é possível observar que caso o esquema utilizado seja assimétrico, a segurança do servidor deve suprir vulnerabilidades contra ataques de canal lateral. Já no cenário de um esquema simétrico, dois dos sensores podem comprometer todas as informações transmitidas caso a chave seja a mesma para todos.

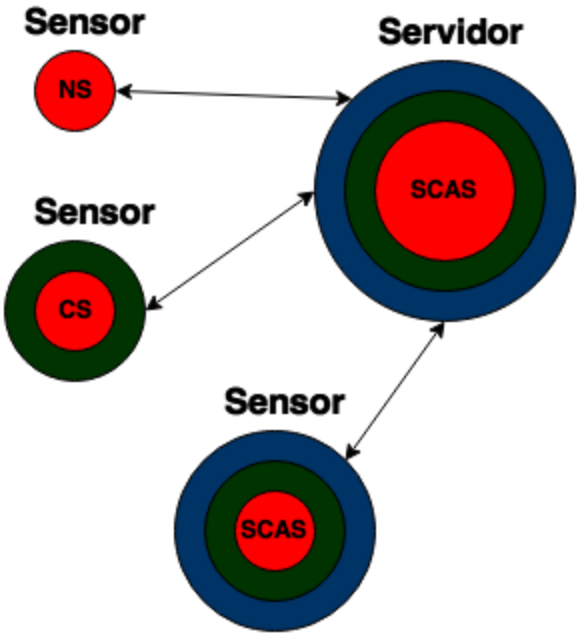


Figura 1. Níveis de Segurança. Fonte: Própria

7. Conclusões

Nesse trabalho foram apresentados ataques de canal lateral e contramedidas para os mesmos, sendo que, é necessário avaliar onde podem ser executados os ataques, quais ataques são mais eficientes e o custo das contramedidas. As conclusões que tiramos desse trabalho são as de que muitos sistemas de internet das coisas não possuem segurança implementada. Dos sistemas que utilizam algoritmos de criptografia, alguns dispositivos não tem a capacidade de implementar as contramedidas, considerando memória necessária, energia e tempo de processamento.

Sendo assim, ataques de canal lateral apresentam grande capacidade de utilização em sistemas com uso de sensores com pouco poder computacional, podendo apenas im-

plementar algoritmos criptográficos. Isso leva um risco à confidencialidade de dados obtidos por sensores que podem monitorar o nosso dia a dia, e também abre vulnerabilidade a modificação na tomada de decisão de sistemas críticos.

Referências

Ambrose, J. A., Ragel, R. G., Jayasinghe, D., Li, T., and Parameswaran, S. (2015). Side channel attacks in embedded systems: A tale of hostilities and deterrence. In *Sixteenth International Symposium on Quality Electronic Design*, pages 452–459.

David McCann, Kerstin Eder, E. O. (2015). Characterising and comparing the energy consumption of side channel attack countermeasures and lightweight cryptography on embedded devices. Cryptology ePrint Archive, Report 2015/832. `http://eprint.iacr.org/`.

Fan, B., Kong, Q., Yuan, X., Wang, Z., and Pan, C. (2013). Learning weighted hamming distance for binary descriptors. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2395–2399.

Hamad, F., Zraqou, J., Maaita, A., and Taleb, A. A. (2015). A secure authentication system for epassport detection and verification. In *Intelligence and Security Informatics Conference (EISIC), 2015 European*, pages 173–176.

Kang, Y. J., Bruce, N., Park, S., and Lee, H. (2016). A study on information security attack based side-channel attacks. In *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pages 61–65.

Luo, P., Zhang, L., Fei, Y., and Ding, A. A. (2015). Towards secure cryptographic software implementation against side-channel power analysis attacks. In *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 144–148.

Morancho, E. (2014). A hybrid implementation of hamming weight. In *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 84–92.

Shah, P. G., Ara, T., Ambareen, J., Huang, X., and Hegde, V. V. (2015). Prevention of simple power analysis attacks in elliptical curve cryptography on wsn platform. In *2015 7th International Conference on Emerging Trends in Engineering Technology (ICETET)*, pages 51–55.

Tragos, E. Z., Foti, M., Surligas, M., Lambropoulos, G., Pournaras, S., Papadakis, S., and Angelakis, V. (2015). An iot based intelligent building management system for ambient assisted living. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 246–252.

Trappe, W., Howard, R., and Moore, R. S. (2015). Low-energy security: Limits and opportunities in the internet of things. *IEEE Security Privacy*, 13(1):14–21.

Yan, Z., Zhang, P., and Vasilakos, A. V. (2014). A survey on trust management for internet of things. *Journal of Network and Computer Applications*.

Caracterização do Raspberry Pi

Lucas Wanner[1], **Pedro De Nigris Vasconcellos**[2]

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251 – 13083-852 – Campinas – SP – Brasil

{ifwanner, pedronvas18}@gmail.com

Abstract. *This paper reports and analyses the progress accomplished in characterising the performance of 9 Parsec benchmarks running on the Raspberry Pi, a low cost, single-board embedded computer. The experiments were performed in a second generation model with the aid of the Perf tool. The data obtained through Parsec and Perf reflect that the platform achieves a good performance-cost tradeoff in computing-intensive applications, and that performance for most of the benchmarks tested scales proportionally to the number of active cores in the Raspberry Pi. Ongoing work consists in characterising energy consumption and implementing runtime libraries capable of dynamically changing application quality in real time in order to minimize energy consumption.*

Resumo. *Este artigo reporta e analisa os progressos realizados na caracterização do desempenho de 9 benchmarks do Parsec sendo executados na plataforma Raspberry Pi. Os experimentos foram executados em um modelo da segunda geração com o auxílio da ferramenta Perf. Os dados obtidos através do Parsec e do Perf refletem a capacidade elevada de processamento que a plataforma atinge em aplicações de alto nível na proporção em que seus quatro núcleos assim como dados esclarecedores a respeito dos contadores de CPU. A próxima etapa consiste em caracterizar o gasto energético e com base nessa informação implementar bibliotecas que alteram a qualidade dos dados em tempo de execução a fim de minimizar o gasto energético.*

1. Contexto da pesquisa

Na última década, tornou-se insustentável aumentar o processamento sem que o consumo de energia e dissipação de calor mantivessem níveis manejáveis, acarretando no desenvolvimento de máquinas que através da paralelização e seus múltiplos núcleos conseguissem aumentar a taxa de processamento. Contudo, percebe-se que as tecnologias relacionadas à conceitos energéticos foram negligenciadas em prol de mais desempenho. É necessário considerar outras soluções energicamente viáveis e inteligentes, vide a tendência da redução de tamanho dos aparelhos eletrônicos e tecnológicos. Uma possível solução é considerar a utilização de plataformas eficientes, que carregam juntamente os conceitos de baixo consumo de energia com considerável poder de processamento.

O Raspberry Pi está incluído neste conjunto devido às suas amplas qualidades: é extremamente versátil na medida em que oferece funcionalidades semelhantes à de um computador desktop ocupando um espaço similar ao de um cartão de crédito e com um preço bastante acessível. Além do que um usuário pode esperar de um computador/laptop, apresenta um custo baixo de energia – em torno de 2W – e é capaz de rodar sistemas operacionais inteiros, principalmente distribuições Linux.

Todas essas características tornam o Raspberry Pi uma plataforma que vale investigar. Devido à sua breve existência – aproximadamente 4 anos – a literatura é bastante escassa, principalmente quando se trata das suas capacidades de lidar com aplicações de alto desempenho, uma vez que ele é amplamente referenciado como uma plataforma para controlar um pequeno conjunto de ações.

2. Objetivos

Na atual conjectura da computação, existe um número reduzido de pesquisas que visam otimizar o consumo de energia. Temos uma plataforma que apresenta um consumo energético extremamente reduzido quando comparada com outras plataformas e uma carência na literatura a respeito do seu processamento. Com isso em mente, temos como objetivo principal, analisar de forma extensa as capacidades do Raspberry Pi, principalmente em aplicações que exigem maior poder computacional. Esperamos, através desta caracterização, obter indicadores capazes de apontar as suas vantagens e desvantagens e como ele se compara com outras plataformas, para que no âmbito do consumo energético seja possível fazer a melhor escolha sobre qual plataforma utilizar para realizar alguma carga de trabalho. A pesquisa também contempla implementar métodos para melhor utilizar a energia durante a aplicação.

3. Método

Utilizamos um modelo Raspberry Pi 2 Model B, com um processador de 900MHz quad-core ARM Cortex-A7 CPU com 1GB de RAM. Foi instalado o sistema operacional Raspbian através da distribuição oficial, em sua versão mais recente, conhecida como Jessie – imagem completa de um sistema operacional baseado no Debian Jessie. A vantagem de utilizar um processador ARMv7 é que podemos implementar aplicações no ambiente Linux, garantindo a compatibilidade.

A caracterização do Raspberry Pi, nesta fase, se deu exclusivamente através dos benchmarks presente no Parsec. Vale a ressalva: o pacote do Parsec apresenta também

dois outros conjuntos de benchmarks – SPLASH-2 e SPLASH-2x – que não foram utilizados de forma algum durante esta fase. Ao compilar cada um dos 16 benchmarks do Parsec, apenas 9 foram compilados com êxito, sendo estes o que compuseram as aplicações testadas: Blackscholes, Bodytrack, Facesim, Ferret, Fluidanimate, Freqmine, Streamcluster. Visto que vários destas aplicações necessitaram um número considerável de bibliotecas para efeitos de compilação, não houveram tentativas extensas a fim de descobrir o motivo pelo qual as outras aplicações não compilaram.

O manual do Parsec contém todos os comandos suportados, assim como todos que foram utilizados ao longo dos experimentos. Indicaremos no artigo apenas os comandos que consideramos indispensáveis.

A execução de cada aplicação requer seu nome, o tamanho da entrada e a quantidade de threads ativas.

```
1 parsecmgmt -a run -p <PACKAGE> -i <INPUT> -n <THREADS>
```

Muito embora o Parsec ofereça 4 tipos de arquivos de entrada, – Simsmall, Simmedium, Simlarge e Native – focamos nas entradas nativas. Esta decisão deriva do fato que todas as outras entradas são subconjuntos ($Simsmall \subseteq Simmedium \subseteq Simlarge \subseteq Native$) e assim, não seria produtivo testar subconjuntos repetidos.

A fim testar a escalabilidade e eficiência de processamento do Raspberry Pi, o parâmetro referente ao número de threads ativas variou de 1 até 4. Dessa forma, conseguimos obter o desempenho da plataforma considerando o número de núcleos ativos – fixamos em 4 justamente por ser uma plataforma quad-core.

Para complementar os resultados, garantindo uma qualidade elevada dos dados obtidos, utilizamos a ferramenta Perf. Devido à sua capacidade poderosa de coleta de informações a respeito dos contadores de desempenho de CPU, sua utilização permitiu uma análise detalhada da CPU em cada aplicação, sendo crucial em casos pontuais nos quais os resultados obtidos diferenciavam do que era esperado. É necessário garantir que a versão do Perf seja a mesma do kernel do sistema operacional para que ele possa ser utilizado.

O Perf permite explicitar quantas repetições serão executadas de um determinado comando e quais eventos de software/hardware desejamos obter informações (por exemplo: branch-misses, número de instruções, miss rate da L1). Assim, todos eventos habilitados no Raspberry Pi foram considerados. Por suportar repetições de comandos, os resultados providenciados pela ferramenta são a média aritmética dos eventos juntamente com seu desvio padrão.

```
1 perf stat -r 5 -e <EVENTS> parsecmgmt -a run -p <PACKAGE> -i <INPUT> -n <THREADS>
```

Dessa forma, cada experimento, que consiste em uma aplicação, uma entrada e um número de threads, seria executado 5 vezes.

4. Resultados

Na representação dos resultados, consideramos o tempo real providenciado pelo Parsec, por representar o tempo de execução prático no ponto de vista do usuário. O gráfico da Figura 1 não possui nenhuma barra de erro pois os erros ficaram na faixa de 0-3% e as barras de erros eram insignificante considerando a grandeza dos valores obtidos.

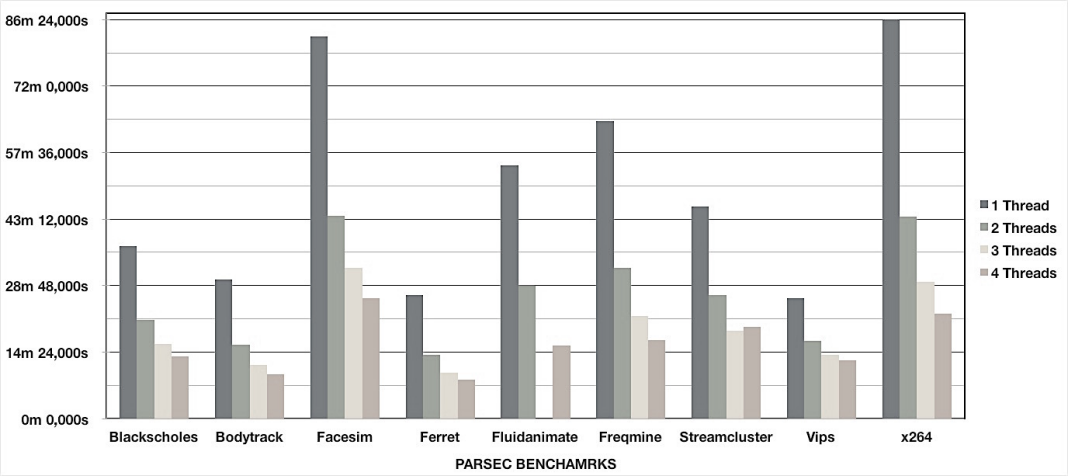


Figura 1. Médias de *Tempo Real* para os 9 benchmarks variando o número de threads ativas

Como nesta etapa não consideramos o consumo energético das aplicações, não somos capazes, no momento, de providenciar resultados concretos sobre as vantagens e desvantagens a respeito do trade-off entre tempo de execução e consumo energético, nem o impacto no custo energético em ativar cada núcleo. Esperamos obter estes resultados na próxima etapa, quando mediremos de fato este consumo.

A Figura 1 mostra claramente que para todos benchmarks, exceto para o Streamcluster, aumentar o número de núcleos ativos, o tempo de execução diminui consideravelmente, garantindo uma melhora de desempenho de pelo menos 50% em todos os benchmarks. Estes resultados comprovam a teoria quando consideramos multithreading, contudo é importante ressaltar a importância deste ganho. Considerando a natureza dessas aplicações, que variam desde cálculos matemáticos (Blackscholes), manipulação de imagens (Ferret), modelagem 3D (Facesim) e até mesmo manipulação de partículas (Fluidanimate), que demandam cálculos complicados e requerem um poder computacional bastante elevado, atingir um ganho de desempenho em torno de 70% é extremamente relevante. Desta forma, sem que haja uma limitação energética, ativar todos os núcleos do Raspberry Pi garante uma redução em mais da metade do tempo de execução das aplicações testadas, sendo o tempo de execução mais longo, com 4 núcleos, é de 30 minutos.

Podemos agrupar os ganhos de desempenho em 3 grupos.

As aplicações Blackscholes, Bodytrack e Facesim obtiveram ganhos de desempenho de 64,0046%, 68,2829% e 68,5331%, respectivamente. Os contadores obtidos através do Perf não demonstram alguma disparidade notável, contudo vale o comentário que os

contador de ciclos de CPU se manteve constante, indicando que as aplicações apresentam uma natureza paralelizável.

As aplicações Freqmine, x264 e Fluidanimate obtiveram ganhos de desempenho de 73,7005%, 73,6924% e 71,1487%, respectivamente. Este grupo corresponde aos maiores ganhos de desempenho da plataforma. Para as aplicações Freqmine e x264, os ciclos de CPU se mantiveram constantes apesar do aumento de misses na cache LLC das instruções store e load.

No caso da aplicação Fluidanimate, os contadores **iTLB-load-misses** e **L1-icache-load-misses** apontaram algo interessante da aplicação. Ambos contadores dizem respeito à cache de instruções, o que pode significar que, justamente por ser altamente paralelizável, a aplicação utiliza em determinado trechos – divididos entre as threads – um conjunto grande de instruções, tal que, quando uma thread necessita de identificar uma instrução do seu trecho, a cache possui instruções de outro trecho referente à outra thread. Iso implica no aumento do número de ciclos de CPU que também foi constatado.

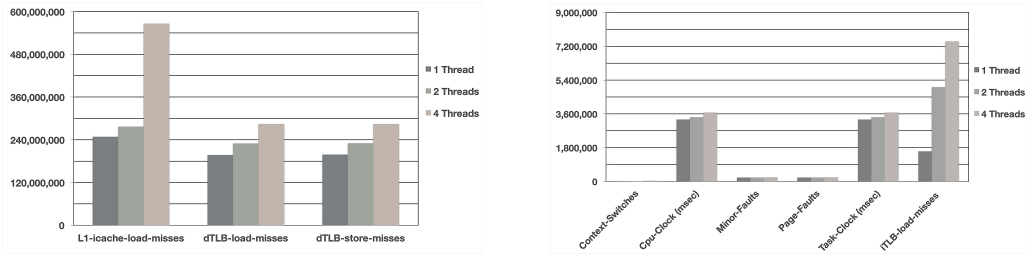
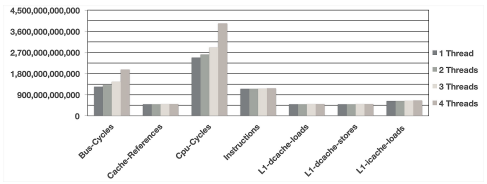
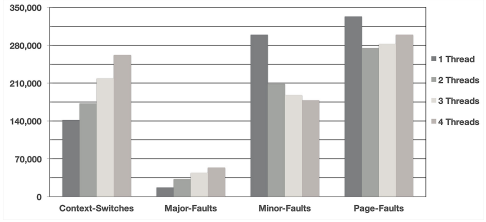


Figura 2. Contadores de CPU do *Fluidanimate*

As aplicações Streamcluster e Vips obtiveram ganhos de desempenho de 56,8973% e 51,2504%, respectivamente. Consistem nas aplicações com o menor ganho de desempenho e apresentam os contadores mais inconsistentes, principalmente referente ao ciclos de CPU do Streamcluster e da troca de contexto do Vips, fatores que impactaram drasticamente o desempenho. Esses contadores indicam que estas aplicações apresentam maiores dificuldades quando são paralelizadas, acarretando diretamente no tempo de execução.



(a) Contadores de CPU do ***Streamcluster***



(b) Contadores de CPU do ***Vips***

5. Em andamento

A pesquisa a respeito do Raspberry Pi ainda não atingiu a sua completude. No atual estágio, estamos implementando bibliotecas compartilhadas que visam analisar as

condições nas quais o raspberry está submetido através de sensores externos (temperatura, bateria) a fim de alterar, em tempo de execução, a qualidade dos dados gerados, maximizando a eficiência energética e a qualidade dos resultados das aplicações. Em seguida, pretendemos medir o consumo energético em dois cenários distintos – com e sem a utilização destas bibliotecas – a fim de determinar quanto foi ganho em termos energéticos e a que custo na qualidade dos resultados das aplicações, permitindo avaliar a eficácia da otimização de energia.

6. Conclusões

Apesar de estarmos no início da pesquisa, apenas tendo analisado o desempenho de 9 aplicações, os resultados são significativos, pois provam que o Raspberry Pi esta capacitado para lidar com aplicações que exigem um alto nível de processamento, vide a natureza das aplicações presentes no Parsec, garantindo um tempo razoável de execução de no máximo 30 minutos. Também foi constatado o impacto elevado quando aumentamos o número de núcleos utilizado na aplicação, representando ganhos de desempenho que variam de 50% até 74%.

O Perf é indispensável para detectar detalhes específicos de aplicações e conseguem apontar disparidades de forma bastante eficiente, que a longo prazo, servem como base para caracterizar a plataforma de uma maneira completa.

Esperamos com a próxima fase da pesquisa podermos comparar resultados de desempenho com consumo energético a fim de compreender ainda melhor quais cenários a utilização desta plataforma providencia a maior quantidade da vantagens, assim como rodar testes com as bibliotecas que visam minimizar o consumo energético enquanto tentando garantir resultados com qualidade elevada.

Referências

Bienia, C. (2011). *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University.

McKinley, Blackburn, E. (2011). Looking back on the language and hardware revolutions. In *Measured Power, Performance, and Scaling*.

Checklist Quality Assessment of Natural Language Requirements for Space Applications

Anderson Rossanez[1], Ariadne M. B. R. Carvalho[1], Marco Vieira[2]

[1]Institute of Computing – University of Campinas
Campinas-SP, Brasil

[2]Department of Informatics Engineering – University of Coimbra
Coimbra, Portugal

anderson.rossanez@gmail.com, ariadne@ic.unicamp.br, mvieira@dei.uc.pt

Abstract. *Problems with the specification of software requirements documents are a common cause of software defects. In the space applications domain, such defects are very costly, especially when detected after deployment in the field. It is imperative to ensure that software requirements are well written to avoid the introduction of these defects. The quality of software requirements is frequently assessed via checklists, based on standards for space application software, and on problems found in previous projects. Given the importance of quality assessment, and the fact that it is manually performed by domain experts, we propose to develop a semi-automatic, natural language processing tool, to diminish the reviewer's effort in the assessment, and to reduce errors in this process.*

Resumo. *Problemas com a especificação de requisitos são uma causa comum de defeitos de software. No domínio das aplicações espaciais, esses defeitos tem um custo muito alto, especialmente quando detectados em campo. É imperativo garantir que os requisitos de software estão bem escritos, de modo a evitar a introdução de tais defeitos. A qualidade de documentos de requisitos de software é comumente verificada utilizando-se checklists, baseados em padrões para software do domínio espacial e em problemas encontrados em projetos anteriores. Dada a importância dessa validação e do fato que ela é feita de maneira manual por especialistas do domínio, nós propomos o desenvolvimento de uma ferramenta semi-automática de processamento de língua natural, para diminuir o esforço de um revisor na validação e para reduzir erros nesse processo.*

1. Introduction

Software defects, when detected at later stages of software development, can be very expensive. If they are detected when the software product is already deployed in the field, the cost is even higher and, depending on the application, the consequences of such defects can be catastrophic. A considerable number of software defects are related to problems in the software requirements, that could have been avoided if the Software Requirements Specification (SRS) document were well written and properly verified. In areas such as space applications, SRS documents are written almost entirely in Natural Language (NL), which makes them understood by a higher number of people, not only

by the domain specialists. But, unfortunately, this may result in issues inherent to natural languages, such as ambiguity, incompleteness, misinterpretation, among others.

In order to detect problems in the software requirements, and to avoid finding software defects at the later development phases, researchers have proposed methods to assess the quality of the SRS documents. In the space applications domain, Véras et al. [Véras et al. 2010] provide an approach for SRS review, to be performed by domain specialists, guided by checklists consisting of several yes/no questions used to measure the quality of the software requirements. They use three types of checklists: the first one based on Packet Utilization Standards (PUS) [Secretariat 2003], the second on Conformance and Fault Injection (CoFI) methodology [Ambrosio et al. 2006], and the third on problems found in previous projects.

The PUS-based checklist is used to assure the compliance to the standard, defined by the European Cooperation for Space Standardization (ECSS). It contains questions elaborated from two services present in the standard: 91 questions from the telecommand verification service, and 260 questions from the on-board operations scheduling service.

The purpose of the CoFI-based checklist is to verify if the SRS handles situations related to software failures. In the CoFI methodology, Finite State Machines (FSMs) are generated from the SRS to produce test cases. Questions for the checklist are generated from the FSMs, which are built from the PUS services: 35 questions for the telecommand verification service, and 23 questions for the on-board operations scheduling service.

The objective of the error-based checklist is to detect typical specification mistakes. It comes from SRS problem reports of seven past projects. Review Item Discrepancies (RIDs) were found and categorized: external conflict/inconsistency, lack of traceability, external incompleteness, incorrectness, internal conflict/consistency, application knowledge, readability, domain knowledge, and non-usage of standard. 22 questions were generated from each of the problem descriptions. The 22 questions from the error-based checklist are divided into four categories:

1. *Lack of Traceability*: 5 questions for detecting problems related to missing or wrong traceability, such as whether all requirements are traced to at least a system or an interface requirement, or to the correct system or interface requirement;
2. *Requirement Incompleteness*: 4 questions for detecting if the requirements do not contain terms like “TBD” (To Be Defined) or “TBC” (To Be Confirmed), and other indications of incompleteness;
3. *Incorrectness*: 10 questions aiming to detect words or expressions indicating incorrectness, such as “should”, “might”, or “in preference”;
4. *Internal Conflict/Consistency*: 3 questions for checking if there are conflicts or other inconsistencies, like a reference to a figure or table which is not related to the requirement, or showing conflicting information with what the requirement states.

As stated in the work by Véras et al. [Véras et al. 2010], the checklists were manually used in three different projects, by six different domain specialists. They answered some questions differently due to different levels of expertise and, also, due to different interpretations of both the questions and the requirements. The discrepant answers direclty influenced the quality assessment of the SRS document under analysis. This problem

was pointed out by the authors as an important issue for future work. We believe that, if this process were semi-automatic, the number of divergent answers from the specialists could be minimized, and the necessary effort in the process would be reduced.

2. Objectives

The aim of our work is to provide a semi-automatic method for helping in the quality assessment of SRS documents through Natural Language Processing (NLP). Specifically, we want to provide an automated solution for some of the problems described at Véras et al. [Véras et al. 2010], by developing a tool to be used in the quality analysis of natural language SRS documents. In this way, we expect to mitigate the influence of the human factor by reducing the effort in the process, preventing errors, and also, speeding up the quality assessment process. Our work will focus on the checklist generated from errors detected in past projects.

3. Literature Review

We searched the scientific literature for existing works regarding automated software requirements analysis using NLP. We found several works using NLP in requirements analysis, but none using NLP applied to the automation of checklists, similar to the one we propose here. We found works in three different areas: Quality Assessment, Quality Indicators, and Model Generation.

3.1. Quality Assessment Tools

These works describe tools that implement methods to assess the quality of natural language SRS documents, automatically or semi-automatically, using NLP techniques. Some of these tools are: NASA's ARM tool [Carlson and Laplante 2014], which detects ambiguity, incompleteness, and lack of understanding; the EA-Analyzer [Sardinha et al. 2013], which identifies conflicting requirements; and the Requirements Quality Analyzer (RQA) [Genova et al. 2013], which detects a wide range of problems on the requirements.

3.2. Quality Indicators

These works describe some indicators used to measure the quality of NL SRS documents, such as ambiguity, incompleteness, lack of atomicity, among others. Some of them present methods for detecting such indicators using NLP techniques, such as ambiguity, incompleteness, and lack of understanding [Lamar and Mocko 2010]; and the identification of equivalent requirements [Falessi et al. 2010].

3.3. Model Generation

These works show a different use of NLP on SRS documents: the generation of models that are commonly used to produce test cases and source code. Models, such as UML diagrams or state machines, can also be very helpful for assisting the quality assessment of SRS documents. Some examples are: UML models for generating test cases and source code [Tichy and Koerner 2010]; state machines for identifying incompleteness and inconsistencies [Kof 2009]; and extended finite state machines from SRS documents for generating test cases [Greghi et al. 2015].

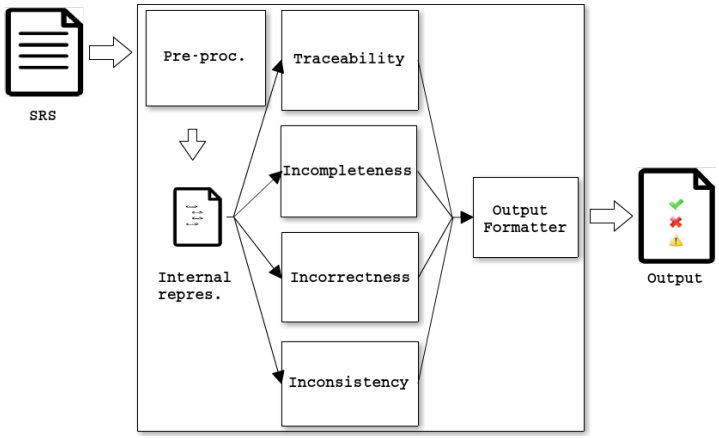


Figure 1. Tool's architecture.

4. Our Approach

We want to develop a tool to help the reviewer apply the error-based checklist to assess the quality of SRS documents. This characteristic makes it different from the other tools that we found in the literature review, which perform software requirements analysis using NLP. Nevertheless, we will consider some characteristics and techniques used by those tools in this work.

When applying the error-based checklist, the reviewer must answer 22 questions similar to the following:

1. *Are all the requirements without the word "should" indicating a non-mandatory feature of the system?*
2. *Are all the variables or numeric values with the proper units?*

Some of the questions will be automatically answered by the tool (e.g., question [1.]), in which the tool must check if the requirements contain words or expressions that should not be there. For the other questions (e.g., question [2.]), the tool will generate a warning and leave the question unanswered. The warnings will explain to the reviewer what to do for answering the question.

The tool will be divided into modules representing the question's categories from the error-based checklist. Each module will be responsible for specific actions, which will eventually result in either answers to questions, or warnings. The modules will use NLP tools available in the Stanford CoreNLP toolkit [Manning et al. 2014]. Figure 1 illustrates the tool's architecture in a high level.

The first module is responsible for pre-processing the SRS document, which may be written in different formats, such as PDF, DOC, PPT, among others. Therefore, the first task is to convert the document into plain text for information extraction. This may be achieved through libraries such as Tika, from the Apache foundation, which extracts text and metadata from several document formats. Once plain text is obtained, the statements from the requirements can be retrived and converted to an internal standard representation, like XML (eXtensive Markup Language). Once the text is in the standard representation, it is used as input by the other modules.

The traceability module handles questions related to the lack of traceability from the error-based checklist. It must check if the requirements traceability matrix is complete, and raise warnings to help the reviewer in the traceability checking. To achieve this, the requirements traceability information must be retrieved from the internal representation, extracted in the pre-processing module.

In the incompleteness module, the questions from the requirement incompleteness category are considered. The module must raise warnings related to requirements with events and numeric values to be checked by the reviewer. Also, it must automatically answer questions related to terms that should not be present in the requirements, like "TBD" or "TBC".

The incorrectness module deals with questions related to the incorrectness category. In this module, most of the questions must be automatically answered, since they are mostly concerned with the identification of terms and expressions that should not be present in the requirements, indicating incorrectness. The Stanford TokensRegex, and a set of rules for identifying the undesired terms and expressions, will be used for this task.

The inconsistency module deals with questions from the internal conflict/inconsistency category of the checklist. It is responsible for raising warnings when finding references to images, tables, and lists in the requirements. Thus, the reviewer must check if such references are according to the requirements.

The last module is responsible for gathering all the warnings and answers from the previous modules and formatting the final output in form of tables, lists, or XML files.

5. Conclusions

In the current state of our work, we can say that providing a semi-automatic tool to help reviewing SRS documents is feasible. Nevertheless, there is still a lot to be done in terms of the tool's development.

One point under investigation is the SRS document pre-processing, specifically the extraction of the requirements and other relevant information. Up to now, this task needs manual intervention, but we are investigating some alternatives, such as to use machine learning and named entity recognition combined with parse trees to automate the task. Besides, we will consider the methodology used in the work by Greghi et al. [Greghi et al. 2015], whose input is also SRS documents written in different formats.

We also need to decide on the tool's output. It could be customized to fit different views, for example: a checklist view, showing the original checklist with the answers and warnings; a requirements view, showing the list of requirements where, for each requirement, we could expand a list showing all the answers and warnings from the checklist's questions; and views containing only requirements with warnings, positive or negative answers.

Finally, we must decide on the tool's validation. We expect to compare the results obtained from the manual analysis of the documents, performed by Véras et al. [Véras et al. 2010] using the error-based checklist, with the results obtained with the results of the analysis assisted by our tool. Also, we expect to run the tool on SRS documents containing injected defects, in order to see if they are properly caught either with warnings, or with automatically generated answers.

References

Ambrosio, A. M., Martins, E., Vijaykumar, N. L., and Carvalho, S. V. (2006). A conformance testing process for space applications software services. *Journal of Aerospace Computing, Information, and Communication*, 3(4):146–158.

Carlson, N. and Laplante, P. (2014). The nasa automated requirements measurement tool: A reconstruction. *Innovations in Systems and Software Engineering*, 10(2):77–91.

Falessi, D., Cantone, G., and Canfora, G. (2010). A comprehensive characterization of nlp techniques for identifying equivalent requirements. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pages 18:1–18:10, New York, NY, USA. ACM.

Genova, G., Fuentes, J., Llorens, J., Hurtado, O., and Moreno, V. (2013). A framework to measure and improve the quality of textual requirements. *Requirements Engineering*, 18(1):25–41.

Greghi, J. G., Martins, E., and Carvalho, A. M. B. R. (2015). Semi-automatic generation of extended finite state machines from natural language standard documents. In *Dependable Systems and Networks Workshops (DSN-W), 2015 IEEE International Conference on*, pages 45–50.

Kof, L. (2009). Translation of textual specifications to automata by means of discourse context modeling. In Glinz, M. and Heymans, P., editors, *Requirements Engineering: Foundation for Software Quality*, volume 5512 of *Lecture Notes in Computer Science*, pages 197–211. Springer Berlin Heidelberg.

Lamar, C. and Mocko, G. (2010). Linguistic analysis of natural language engineering requirement statements. In *Proceedings of the 8th International Symposium on Tools and Methods of Competitive Engineering, TMCE 2010*, volume 1, pages 97–111.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Sardinha, A., Chitchyan, R., Weston, N., Greenwood, P., and Rashid, A. (2013). Ea-analyzer: automating conflict detection in a large set of textual aspect-oriented requirements. *Automated Software Engineering*, 20(1):111–135.

Secretariat, E. (2003). Ground systems and operations–telemetry and telecommand packet utilization ecss-e-70-41a.

Tichy, W. and Koerner, S. (2010). Text to software: Developing tools to close the gaps in software engineering. *Proceedings of the FSE/SDP Workshop on the Future of Software Engineering Research, FoSER 2010*, pages 379–383.

Véras, P. C., Villani, E., Ambrósio, A. M., Pontes, R. P., Vieira, M., and Madeira, H. (2010). Benchmarking software requirements documentation for space application. In *Proceedings of the 29th International Conference on Computer Safety, Reliability, and Security*, SAFECOMP'10, pages 112–125, Berlin, Heidelberg. Springer-Verlag.

Design de um ambiente de Programação Tangível para crianças no contexto educativo brasileiro

Marleny Luque Carbajal[1], M. Cecília C. Baranauskas[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6176 CEP 13081-970– Campinas – SP – Brazil

cecilia@ic.unicamp.br, marleny.carbajal@ic.unicamp.br

Abstract. *Learning programming has positive effect on children's development. Tangible User Interfaces (TUI) are an easy way for them to learn computer programming. TaPrEC is a tangible programming environment designed for children to learn basic programming concepts. Children build tangible programs on a table using wooden blocks like puzzle pieces. The TaPrEC evaluation in a real educational scenario with elementary school children suggests that it is an interesting programming environment for children, easy to learn and use. Furthermore, the results demonstrate that the children had a preliminary understanding of the basic concepts of programming.*

Resumo. *A aprendizagem da programação tem um efeito positivo no desenvolvimento das crianças. Uma forma potencialmente facilitada de introduzir as crianças na programação são as Interfaces Tangíveis. TaPrEC é um ambiente de programação tangível projetado para que crianças aprendam conceitos básicos de programação. As crianças constroem programas tangíveis sobre uma mesa usando blocos de madeira semelhantes a peças de quebra-cabeças. Os resultados da avaliação do TaPrEC num cenário educacional real com crianças do ensino fundamental sugerem que é um ambiente de programação interessante para as crianças, fácil de aprender e de utilizar. Além disso, os resultados demostram que as crianças obtiveram um entendimento preliminar dos conceitos básicos da programação.*

1. Introdução

A pesquisa mostrou que o ensino da programação de computadores pode beneficiar o desenvolvimento das crianças em muitos aspectos, não só em áreas como matemática e ciências, mas também nas habilidades de linguagem, interação emocional social e criatividade [Clements 1999]. A aprendizagem da programação é uma forma eficiente de cultivar o pensamento computacional, que tem sido descrito como uma habilidade fundamental para todos, não apenas para cientistas da computação [Wing 2006]. Então, resulta altamente importante projetar ambientes apropriados para ajudar as crianças a aprender a programação.

A teoria de [Piaget 1959] oferece uma base teórica para a promoção da utilização das Interfaces Tangíveis, ao mostrar que a interação com o mundo físico é essencial para as crianças durante seu processo de desenvolvimento da inteligência. Uma Interface de Usuário Tangível (TUI-*Tangible User Interface*) [Ishii and Ullmer 1997] permite que o usuário possa interagir com a informação digital por meio da

teclado ou o mouse. Há autores [Horn, Solovey, Crouser and Jacob 2009] que afirmam que as linguagens de programação tangíveis têm o potencial de facilitar a aprendizagem de sintaxes complicadas, de promover a colaboração, e facilitar aos professores manter um ambiente positivo de aprendizagem.

Conforme ao exposto, projetamos e desenvolvemos o ambiente de programação tangível denominado TaPrEC (*Tangible Programming Environment for Children*) envolvendo no processo de design as principais partes interessadas (pesquisadores, professores, alunos) para garantir que a solução criada faça sentido para elas. Para atingir esse objetivo, utilizamos o modelo Semio-participativo de Design [Baranauskas, Martins e de Assis 2012], inspirado na Semiótica Organizacional, que articula ao mesmo tempo o desenvolvimento de sistemas interativos e práticas sociais com as partes interessadas.

TaPrEC permite às crianças criarem programas de computador organizando objetos tangíveis aplicando três conceitos básicos de programação: Sequências, Repetições e Procedimentos. A arquitetura do TaPrEC está composta pelo *Raspberry Pi*, um microcomputador de baixo custo, tecnologia RFID incorporada nos blocos de programação e um software de processamento de dados desenvolvido na linguagem de programação *Scratch*.

Neste artigo apresentamos o desenvolvimento incremental do nosso protótipo inicial, alimentado pela participação das partes interessadas. Assim, o texto é organizado como segue: na próxima seção apresentamos os trabalhos relacionados ao projeto. Em seguida apresentamos em detalhe o ambiente de programação tangível TaPrEC. Finalizamos com a conclusão e próximos passos.

2. Trabalhos Relacionados

O conceito de programação tangível, o uso de técnicas de interação tangível para construir programas de computador, nasceu na década de 1970 quando Radia Perlman criou o *Slot Machine* [Perlman 1976] que permitia a criação de programas físicos do Logo [Papert 1980]. Desde então, diversos grupos de investigação criaram uma variedade de sistemas de programação tangíveis que suportam a construção física da estrutura de um algoritmo. O ambiente *AlgoBlock* [Suzuki e Kato 1995] permitia aos usuários manipular blocos de alumínio que podiam ser conectados entre si para formar um programa executável com o objetivo de deslocar um submarino virtual na tela do computador. Outro sistema é o *Tangible Programming Bricks* [McNerney 2004] que possuía peças Lego com um microprocessador incorporado que podiam ser empilháveis e usadas para a construção de programas simples. O ambiente *Electronic Blocks* [Wyeth 2008] está conformado por blocos Lego com circuitos eletrônicos embutidos neles projetados para criar programas de computador tangíveis empilhando os blocos. *Tern* [Horn and Jacob 2007] é uma linguagem de programação tangível que combina materiais de baixo custo e visão computacional. *T-Maze* [Wang, Wang and Liu 2014] possui um conjunto de blocos de madeira interligados por ímãs e utiliza tecnologia de visão computacional para converter automaticamente os programas físicos em código digital. *E-Block* [Wang, Zhang, and Chen 2013] está composta por um conjunto de blocos de programação que servem como dados de entrada e permite escrever programas tangíveis. A informação da sequência de blocos é transferida para o

computador por meio de microcomputadores e, em seguida, é traduzido em informação semântica.

Em geral podemos distinguir dois tipos de tecnologias usadas nos objetos tangíveis dos ambientes de programação apresentados: i) circuitos eletrônicos e microcomputadores; e ii) visão computacional. No primeiro caso, para montar uma peça tangível é preciso ter conhecimentos básicos de eletrônica. No caso dos blocos que utilizam visão computacional, ambos os projetos apresentados (Tern e T-Maze) utilizam códigos TopCodes [Horn 2012] que precisam conhecimento de programação na linguagem Java. As diferentes propostas oferecidas para a programação tangível usam tecnologias que requerem investimentos diferentes. No caso do grupo de microprocessadores ou circuitos eletrônicos, requerem um investimento maior por tratar-se de tecnologia mais sofisticada. Para o caso dos ambientes que utilizam visão computacional, o investimento se eleva se consideramos os computadores especializados necessários para processar o software de visão computacional e uma câmera com boa resolução para garantir a captura dos códigos de visão computacional.

Com base nas contribuições de esses projetos, desenvolvemos o ambiente de programação tangível TaPrEC com a ideia de usar tecnologia acessível a populações socioeconomicamente menos favorecidas sem fazer um enorme investimento. Com esse propósito, escolhemos usar um microcomputador de baixo custo que possui todas as funcionalidades básicas de qualquer computador como hardware principal, a tecnologia RFID para construir os objetos tangíveis e a linguagem de programação *Scratch* para facilitar a personalização do ambiente.

3. O TaPrEC: *"Tangible Programming Environment for Children"*

Com base nos *feedbacks* das crianças e professores obtidos ao longo das nossas primeiras oficinas experimentais [Carbajal e Baranauskas 2015], fizemos diferentes mudanças em nosso protótipo inicial para torná-lo o mais eficaz possível para a sua finalidade. A seguir detalhamos os componentes, a sintaxe e a arquitetura do ambiente de programação tangível TaPrEC na sua segunda versão.

TaPrEC está composto por um hardware de baixo custo, os blocos de programação e um software de processamento de dados conforme ilustra a Figura 1.

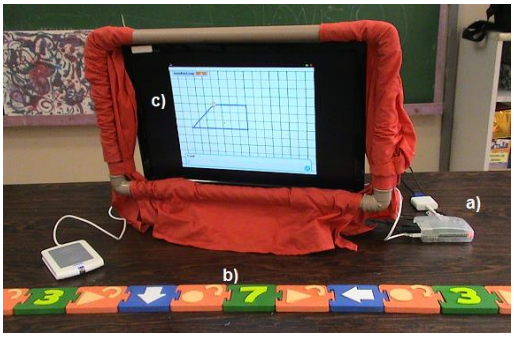
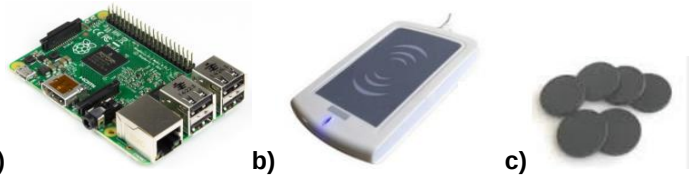


Figura 1. Componentes do ambiente TaPrEC: a) hardware de baixo custo; b) blocos de programação; e c) software de processamento de dados.

TaPrEC utiliza o microcomputador *Raspberry Pi* (Figura 2a) como hardwar principal. Na segunda versão do ambiente TaPrEC utilizamos o modelo de segunda geração *Raspberry Pi 2 Modelo B.* A outra tecnologia usada no TaPrEC é a Identificação por

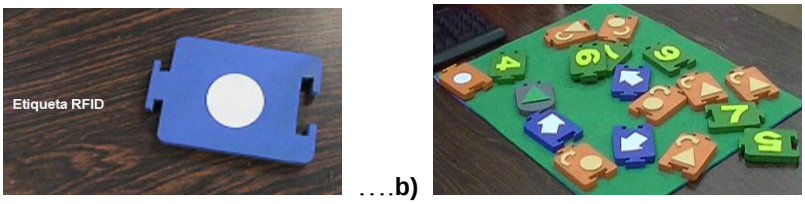
Radiofrequência (RFID - *Radio Frequency Identification*) que pode ser utilizada para fornecer um identificador único a objetos físicos. O funcionamento dos sistemas RFID é simples: a etiqueta RFID (Figura 2c), que contem os dados de identificação, gera um sinal de radiofrequência com esses dados. Esse sinal é detectado por um leitor RFID (Figura 2b), que é responsável pela leitura da informação e o envio em formato digital para uma aplicação específica de processamento de dados.



a) **b)** **c)**

Figura 2. Hardware de baixo custo: a) Raspberry Pi 2 Modelo B; b) leitor RFID e c) etiquetas RFID

Os blocos de programação do ambiente TaPrEC são blocos de madeira coloridos semelhantes a peças de quebra-cabeças que contém em um dos lados uma etiqueta RFID (Figura 3a) para que sejam identificados dentro do ambiente TaPrEC e no outro lado um símbolo em alto-relevo para representar sua funcionalidade (Figura 3b). O Software de Processamento (desenvolvido na linguagem de programação *Scratch* 1.4) se encarrega de armazenar os identificadores das etiquetas RFID de todos os blocos de programação. Todos os códigos que representam uma mesma ação são agrupados em uma lista. Quando o software recebe uma sequência de identificadores verifica se o identificador corresponde a alguma das listas de ações; executa as ações e; mostra os resultados.



a) ….**b)**

Figura 3. Blocos de Programação: a) bloco de programação com a etiqueta RFID incorporada; b) símbolos em alto-relevo dos blocos de programação.

Para criar um programa no ambiente TaPrEC é necessário colocar os blocos de programação numa sequência específica: primeiro o bloco de início, depois os blocos de ações e finalmente o bloco de fim (Figura 4). Os blocos de controle indicam o início e o fim do programa. As informações do programa tangível são enviadas ao software de processamento de dados por meio do leitor de RFID.

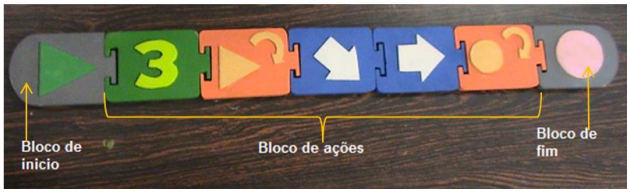


Figura 4. Exemplo de programa no ambiente TaPrEC

Na Figura 5 apresentamos a arquitetura do ambiente TaPrEC.

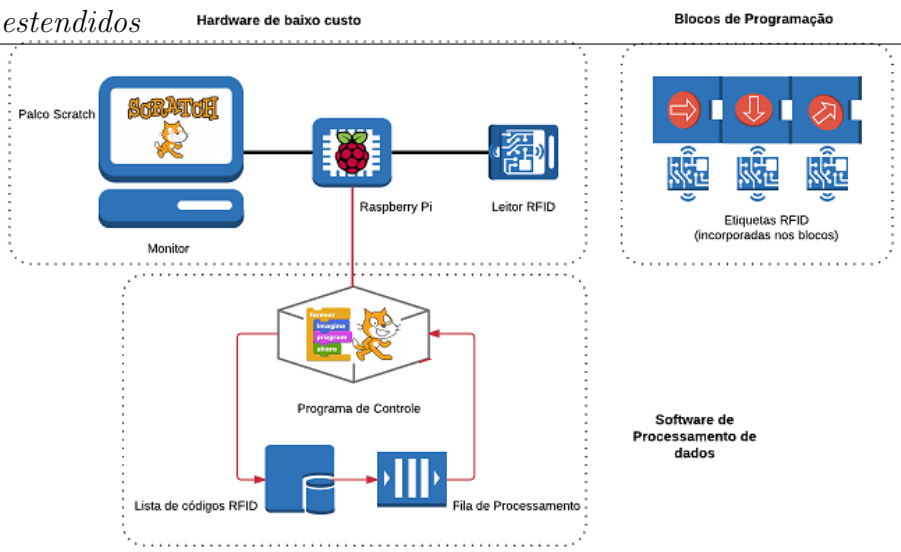


Figura 5. Arquitetura do ambiente TaPrEC

Quando o usuário passa o leitor RFID sobre o bloco de inicio, o software verifica se o identificador existe na lista de códigos RFID (registrados previamente). Caso exista o código, o software habilita uma fila de processamento. Depois quando o usuário passa o leitor pelos blocos de ações, novamente o software verifica se os códigos existem e depois são guardados na fila de processamento. Finalmente quando o usuário passa o leitor pelo bloco de fim, o software executa o código *Scratch* associado a cada um dos blocos de ações e os resultados são mostrados no palco de *Scratch*.

Como nosso método de design envolve a participação ativa das diferentes partes interessadas, realizamos um Estudo de Caso e contamos com a participação de professores e alunos da escola complementar ao ensino fundamental onde esta pesquisa se desenvolveu.

6. Conclusões

Acreditamos que as soluções computacionais devem envolver as partes interessadas em um permanente ciclo de discussão, teste e aperfeiçoamento dos ambientes que se propõem a apoiar esse processo de ensino. O ambiente TaPrEC foi experimentado ao com crianças e professores em contexto educativo, conseguindo que os participantes operassem facilmente os objetos tangíveis, criassem programas tangíveis e aprendessem os conceitos básicos de Sequências, Repetições e Procedimentos, objetivo do TaPrEC. Em termos de usabilidade, os resultados das Oficinas Experimentais demostraram que o ambiente TaPrEC é fácil de aprender, eficiente no uso, e de acordo com os resultados da Autoavaliação de Emoções, os usuários mostraram uma resposta afetiva positiva ao usa-lo. Em termos de tempos de resposta podemos afirmar que a tecnologia usada na segunda versão do TaPrEC mostrou-se mais adequada para a programação tangível. Uma vez que a média do tempo de resposta do sistema melhorou (2s) em comparação do protótipo inicial (5s). Os trabalhos futuros estão relacionados com a melhoria do ambiente TaPrEC. Para uma versão futura do ambiente planejamos acrescentar novos conceitos de programação como as estruturas condicionais, parâmetros, eventos e trabalhar mecanismos de *debugging* no ambiente.

Baranauskas, M. C. C., Martins, M. C., & de Assis, R. (2012). XO na escola e fora dela: uma proposta semio-participativa para tecnologia, educação e sociedade.

Carbajal, M. L., & Baranauskas, M. C. C. TaPrEC: Desenvolvendo um ambiente de programação tangível de baixo custo para crianças. CEP, 13083, 852.

Clements, D. H. (1999). The future of educational computing research: The case of computer programming. Information Technology in Childhood Education Annual, 1999(1), 147-179.

Horn, M. S., & Jacob, R. J. (2007, April). Tangible programming in the classroom with tern. In CHI'07 extended abstracts on Human factors in computing systems (pp. 1965-1970). ACM.

Horn, M. S., Solovey, E. T., Crouser, R. J., & Jacob, R. J. (2009, April). Comparing the use of tangible and graphical programming languages for informal science education. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 975-984). ACM.

Horn, M. (2012). TopCode: Tangible Object Placement Codes. Retrieved from httpz/l users. eecsnorthwester. edit/mmhorn/topcodes.

Ishii, H., & Ullmer, B. (1997, March). Tangible bits: towards seamless interfaces between people, bits and atoms. In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems (pp. 234-241). ACM.

McNerney, T. S. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. Personal and Ubiquitous Computing,8(5), 326-337.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

Perlman, R. (1976). How to Use the Slot Machine. Logo Working Paper 43, Logo Laboratory, Massachusetts Institute of Technology, Cambridge.

Piaget, J. (1959). The language and thought of the child (Vol. 5). Psychology Press.

Suzuki, H., & Kato, H. (1995, October). Interaction-level support for collaborative learning: AlgoBlock—an open programming language. In The first international conference on Computer support for collaborative learning (pp. 349-355). L. Erlbaum Associates Inc.

Wang, D., Wang, T., & Liu, Z. (2014). A Tangible Programming Tool for Children to Cultivate Computational Thinking. The Scientific World Journal, 2014.

Wang, D., Zhang, Y., & Chen, S. (2013). E-Block: A tangible programming tool with graphical blocks. Mathematical Problems in Engineering, 2013.

Wing, J. M. (2006). Computational thinking. Communications of the ACM,49(3), 33-35.

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. The Journal of the learning sciences, 17(4), 517-550.

Xu, D. (2005). Tangible user interface for children-an overview. In Proceedings of the SIXTH Conference in the Department of Computing (pp. 579-584).

Detection of the Alzheimer disease using differences of center of mass in amygdala-hippocampal regions

Alexandre Yukio Yamashita[1], **Neucimar Jerônimo Leite**[1]

[1]Institute of Computing – University of Campinas (UNICAMP)
Campinas – SP – Brazil

alexandre.yamashita@liv.ic.unicamp.br, neucimar@ic.unicamp.br

Abstract. *Recent studies show that the Alzheimer's disease (AD) can be automatically detected using magnetic resonance imaging with high accuracies. In this paper, we propose a method to detect AD using a new feature based on the differences between the brain and its center of mass in amygdala-hippocampal regions. This feature was designed to enhance the peripheral areas of the brain affected by the disease. Experiments were performed using 1265 volumetric images from the Alzheimer's Disease Neuroimaging Initiative dataset. Tests achieved a high accuracy of 90.26% in the distinction of healthy and AD patients.*

Resumo. *Estudos recentes mostram que a doença de Alzheimer (DA) pode ser detectada automaticamente usando imagens de ressonância magnnética com alta acurácia. Esse trabalho propõe um método para detectar DA usando um nova característica baseada em diferenças entre imagens do cérebro e seu centro de massa em regiões do hipocampo e amígdala. Essa característica foi projetada visando realçar as áreas periféricas do cérebro afetadas pela doença. Experimentos foram feitos usando 1265 imagens volumétricas da base de dados da Alzheimer's Disease Neuroimaging Initiative. Os testes realizados apresentaram uma alta acurácia de 90.26% na distinção de pacientes saudáveis e com DA.*

1. Introduction

Alzheimer's disease (AD) is a neurodegenerative disorder that causes problems in memory, reasoning and behavior. It is the most common form of dementia, accounting for 60% to 80% of the dementia cases [Association 2016]. Although there is no cure, there are treatments to relieve the symptoms and reduce the aggravation of the disease. Thus, the early diagnosis of AD is important to provide better quality of life to the patient.

Machine learning techniques can help understand the effects of the disease in the different parts of the brain. In recent years, several classification methods were proposed for the diagnosis using magnetic resonance imaging (MRI). Studies suggest that these methods can predict AD more accurately than clinicians [Klöppel et al. 2008].

Many approaches use Support Vector Machines (SVM) to discriminate AD patients from healthy controls. The methods differ in the selection and type of features extracted. Klöppel et al. [Klöppel et al. 2008] uses the gray matter of the whole brain to classify 68 subjects, reporting an accuracy of 96%. In [Yang et al. 2011], features were extracted with independent component analysis (ICA). The method correctly classifies

87.5% of 366 individuals. Batmanghelich et al. [Batmanghelich et al. 2009] create a sparse basis to represent the images, having a correct classification rate of 89% obtained from images of 115 individuals.

Other methods [Casanova et al. 2011][Rao et al. 2011] use logistic regression (LR) for classification. Differently from SVMs, LRs provide class probabilities that can be useful as diagnosis metrics or to adjust decision thresholds. Casanova et al [Casanova et al. 2011] tested the performance of a penalized logistic regression to classify 98 subjects, resulting in an accuracy of 85.7%. In [Rao et al. 2011], a sparsity penalty is incorporated into the log-likelihood aiming at automatic feature selection. 85.26% of the images from 129 subjects were correctly classified .

Some recent works [Ambastha 2015][Payan and Montana 2015] use deep learning techniques with images from a large database provided by the Alzheimer's Disease Neuroimaging Initiative (ADNI) [Jack et al. 2008]. Ambastha [Ambastha 2015] extracted features with a 3D convolutional neural network (ConvNet), reporting an accuracy of 81.79%. The method from [Payan and Montana 2015] is also based in a 3D ConvNet, but the training is done using a sparse auto-encoder. A high accuracy of 95.39% is reported, but experiments from [Ambastha 2015] indicate bias in the data. Analyzing the performance of these works, we can observe that there is a higher difficulty for classification when large databases are considered.

In this paper, we propose a method to discriminate AD patients from healthy controls using a new feature based on the differences between the brain and its center of mass. This method is evaluated using the ADNI database and considering four classifiers: LR, Linear SVM, SVM with polynomial kernel and Artificial Neural Network (ANN).

The paper is organized as follows. Section 2 describes the database, techniques and methodology used in our experiments. Results are described and discussed in Section 3. Finally, the conclusions and directions for future researches are outlined in Section 4.

2. Material and methods

The ADNI database used to validate our method is described in Subsection 2.1. The images were preprocessed with a method described in the Subsection 2.2. The feature proposed in this work is presented in Subsection 2.3.

2.1. ADNI database

The ADNI is a worldwide project aiming to collect MRI and positron emission tomography for research on AD. T1-weighted volumetric MP-RAGE scans were acquired at different sites using GE, Philips, or Siemens systems. The database consists of 3025 images from 796 individuals categorized in three classes: normal control subjects (NC), patients diagnosed with AD and patients with mild cognitive impairment (MCI). In our experiments, we considered 1265 images from 365 subjects. 436 images belong to subjects diagnosed with AD and 829 to healthy subjects.

2.2. Preprocessing

The preprocessing method used in this work consists of four steps. Firstly, the brains are extracted using the BET2 tool from the FSL package [Jenkinson et al. 2005]. In the

second step, the image is spatially normalized into an International Consortium for Brain Mapping template. The normalization is done using the Symmetric Diffeomorphic Registration method implemented by the ANTs tool [Avants et al. 2009]. In the third step, the values of the image voxels are mapped in the range $[0, 1]$ using the min-max normalization. The new values are calculated by:

$$x_{norm} = \frac{x - X_{min}}{X_{max} - X_{min}}. \tag{1}$$

Each voxel x is subtracted to the minimum value X_{min} of all voxels. Then, they are divided by the difference between the maximum X_{max} and the minimum X_{min} of their original values. At last, the image is resized to the size 74x92x78, which results in 531024 voxels.

2.3. Proposed feature

The process two compute the features has four steps. In the first step, we compute the center of mass locally in regions of the image. Each region is a 3D block with same width, height and depth. The center of mass C is computed by:

$$C_{\mathrm{x}} = \frac{\sum_{\mathrm{x} \in R} \mathrm{x} I(\mathrm{x})}{\sum_{\mathrm{x} \in R} I(\mathrm{x})}. \tag{2}$$

where R is the region, x is the voxel position and $I(\mathrm{x})$ is the voxel value in the position x. The region of a voxel in a position $\mathrm{x} = (x, y, z)$ is defined by all voxels in the valid positions between $(x - s/2, y - s/2, z - s/2)$ and $(x + s/2, y + s/2, z + s/2)$, where s is the block size.

In the second step, the center of mass is smoothed by a mean filter M defined by:

$$M_{\mathrm{x}} = \frac{\sum_{\mathrm{x} \in R} I(\mathrm{x})}{w + h + d}, \tag{3}$$

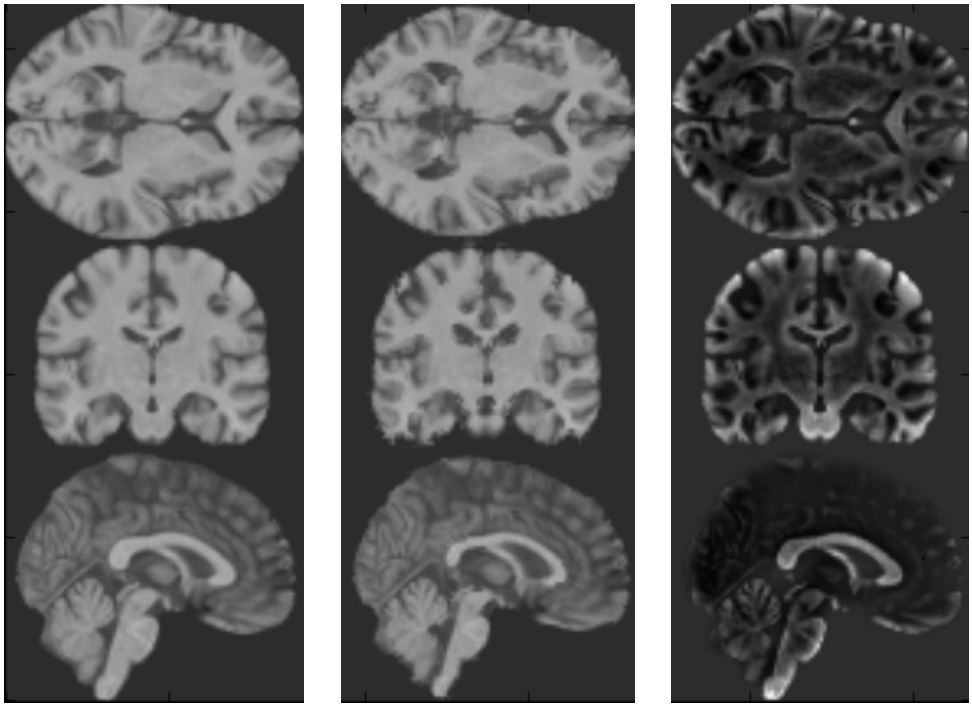
where w, h, d are the width, height and depth of the region.

In the third step, we compute the difference between the brain image and the center of mass smoothed by the mean filter. Lastly, this difference is voxel-wise multiplied by the brain image. This multiplication increases the contrast of the image and reduces the intensity of the voxels in the background.

Fig. 1 presents the output images resulted from the feature extraction. The first image is the input image used to extract the feature. The second image is the center of mass computed from the brain image. The third image is the feature extracted by computing the differences between the brain image and the smoothed center of mass.

2.4. Segmentation

The use of the whole brain for classification may not be the best data to carry out classification. The selection of regions of interest (ROI) can improve the performance of the classifiers [Ambastha 2015]. In this work, we segmented the brain images to use only the amygdala-hippocampal regions.



(a) Brain image (b) Center of mass (c) Feature extracted

Figure 1. Example of feature extraction

The images were segmented with a mask created using the software MARINA (MAsks for Region of INterest Analysis) [Walter et al. 2003]. This software allows to create, smooth, threshold, edit, and save masks in the SPM-ANALYZE format. The mask was created, selecting six neuroanatomical regions: hippocampus-L, hippocampus-R, parahippocampal gyrus-L, parahippocampal gyrus-R, amygdala-L and amygdala-R. Then, the mask was smoothed using an isotropic Gaussian filter with full width at half maximum (FWHM) = 5 mm. A threshold was applied to keep just the values above 0.25 in the mask. The segmentation removed all the voxels with zero values in the mask, resulting in images with 6005 voxels.

3. Results and Discussion

In this section, we present the experiments and results obtained in this work. First, we describe the methodology used for training and testing the classifiers. Then, we describe the experiments considered and present the results obtained for each classifier.

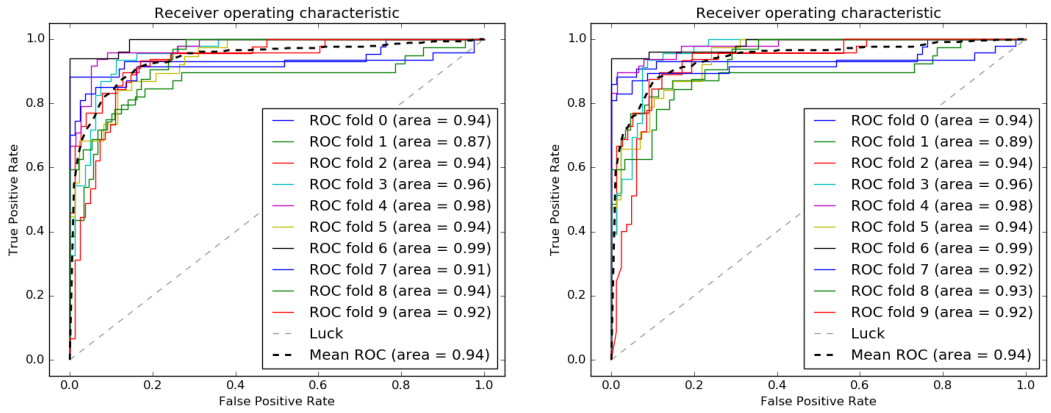
3.1. Validation method

The experiments were evaluated using the 10-fold cross-validation method. All images of a subject were assigned to a unique fold. This is very important to avoid having images of the same subject in both training and testing sets, generating biased results.

First, one fold is selected to be the test set. The other nine folds are used in the training and validation of the classifiers. The validation set is used to adjust the classifier parameters and decide where to stop the training process. The validation and training

Table 1. Performance results

Classifier	Accuracy	Precision	Recall	Specificity
ANN	90.26 ± 1.19	89.20 ± 3.25	81.15 ± 2.76	94.80 ± 1.55
LR	88.61 ± 1.54	86.96 ± 3.55	79.98 ± 3.66	93.16 ± 2.09
Linear SVM	87.59 ± 1.10	82.44 ± 3.13	81.69 ± 2.10	90.70 ± 1.71
P-SVM	85.91 ± 1.19	81.17 ± 3.17	80.58 ± 1.98	89.07 ± 1.82



(a) Logistic Regression (b) Artificial Neural Network

Figure 2. ROC plots

data are also splitted without putting images from the same subject in both sets. After splitting the data, we have about 10% of the data for testing, 85% for training and 5% for validation. This method is repeated for all 10 folds.

3.2. Experiments and results

We performed experiments with four type of classifiers: LR, ANN, Linear SVM and SVM with polynomial kernel (P-SVM). The parameters of each classifier were adjusted by grid search with the validation set. For the ANN, we used 6005 units in the input layer, 200 units in the hidden layer and 2 units in the output layer. The degree of the polynomial used in the P-SVM was 2. And the proposed features were computed using a box size of 7x7x7.

Table 1 presents the peformance metrics obtained for each classifier. The classifier with the highest performance metrics was the ANN, reporting a mean accuracy of 90.26%. The Receiver Operating Characteristic (ROC) plots of the statistical classifiers LR and ANN are presented in Fig. 2(a) and Fig. 2(b), respectively. Both classifiers reported an area under the curve equal to 0.94.

4. Conclusion

In this paper, we proposed a method to detect AD with a new type of feature. The feature is based on the differences between the brain image and its center of mass. Unlike other methods in the literature, we performed experiments with a large MRI dataset and

tested four classifiers. The results obtained show that the feature have a good discrimination power to classify AD patients and health controls. In future works, we intend to test the classification using images from patients with MCI. Also, we want to perform experiments adding other type of features and type of images.

References

[Ambastha 2015] Ambastha, A. K. (2015). Neuroanatomical characterisation of alzheimers disease using deep learning.

[Association 2016] Association, A. (2016). Alzheimer's disease and dementia.

[Avants et al. 2009] Avants, B. B., Tustison, N., and Song, G. (2009). Advanced normalization tools (ants). *Insight J*, 2:1–35.

[Batmanghelich et al. 2009] Batmanghelich, N., Taskar, B., and Davatzikos, C. (2009). A general and unifying framework for feature construction, in image-based pattern classification. In *Information Processing in Medical Imaging*, pages 423–434. Springer.

[Casanova et al. 2011] Casanova, R., Whitlow, C. T., Wagner, B., Williamson, J., Shumaker, S. A., Maldjian, J. A., and Espeland, M. A. (2011). High dimensional classification of structural mri alzheimers disease data based on large scale regularization. *Front Neuroinform*, 5:22.

[Jack et al. 2008] Jack, C. R., Bernstein, M. A., Fox, N. C., Thompson, P., Alexander, G., Harvey, D., Borowski, B., Britson, P. J., L Whitwell, J., Ward, C., et al. (2008). The alzheimer's disease neuroimaging initiative (adni): Mri methods. *Journal of Magnetic Resonance Imaging*, 27(4):685–691.

[Jenkinson et al. 2005] Jenkinson, M., Pechaud, M., and Smith, S. (2005). Bet2: Mr-based estimation of brain, skull and scalp surfaces. In *Eleventh annual meeting of the organization for human brain mapping*, volume 17, page 167.

[Klöppel et al. 2008] Klöppel, S., Stonnington, C. M., Barnes, J., Chen, F., Chu, C., Good, C. D., Mader, I., Mitchell, L. A., Patel, A. C., Roberts, C. C., et al. (2008). Accuracy of dementia diagnosisa direct comparison between radiologists and a computerized method. *Brain*, 131(11):2969–2974.

[Payan and Montana 2015] Payan, A. and Montana, G. (2015). Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. *arXiv preprint arXiv:1502.02506*.

[Rao et al. 2011] Rao, A., Lee, Y., Gass, A., and Monsch, A. (2011). Classification of alzheimer's disease from structural mri using sparse logistic regression with optional spatial regularization. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 4499–4502. IEEE.

[Walter et al. 2003] Walter, B., Blecker, C., Kirsch, P., Sammer, G., Schienle, A., Stark, R., and Vaitl, D. (2003). Marina: An easy to use tool for the creation of masks for region of interest analyses. In *9th International Conference on Functional Mapping of the Human Brain*, volume 19.

[Yang et al. 2011] Yang, W., Lui, R. L., Gao, J.-H., Chan, T. F., Yau, S.-T., Sperling, R. A., and Huang, X. (2011). Independent component analysis-based classification of alzheimer's disease mri data. *Journal of Alzheimer's disease*, 24(4):775–783.

Duas famílias de caterpillars 0-rotativos

Atílio G. Luiz[1], C. N. Campos[1], R. Bruce Richter[2]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas – São Paulo – Brazil

[2]Department of Combinatorics and Optimization – University of Waterloo
Waterloo – Ontario – Canada.

gomes.atilio@gmail.com, campos@ic.unicamp.br, brichter@uwaterloo.ca

Abstract. *A graceful labelling of a tree T is an injective function $f\colon V(T) \to \{0,\dots,|E(T)|\}$ such that $\{|f(u)-f(v)|\colon uv \in E(T)\} = \{1,\dots,|E(T)|\}$. A tree T is said to be 0-rotatable if, for any $v \in V(T)$, there exists a graceful labelling f of T such that $f(v)=0$. In this work, the 0-rotatability of caterpillars is investigated and it is proved that the following two families of caterpillars are 0-rotatable: caterpillars with perfect matching and caterpillars obtained by identifying one leaf of the star $K_{1,s-1}$ to a leaf of P_n, with $n \geq 4$ and $s \geq \lceil \frac{n-1}{2} \rceil$. This result reinforces the conjecture that all caterpillars with diameter at least five are 0-rotatable.*

Resumo. *Uma rotulação graciosa de uma árvore T é uma função injetora $f\colon V(T) \to \{0,\dots,|E(T)|\}$ tal que $\{|f(u)-f(v)|\colon uv \in E(T)\} = \{1,\dots,|E(T)|\}$. Uma árvore T é dita 0-rotativa se, para todo $v \in V(T)$, existe uma rotulação graciosa f de T tal que $f(v)=0$. Neste trabalho, provamos que as seguintes famílias de caterpillars são 0-rotativas: caterpillars com emparelhamento perfeito e caterpillars obtidos a partir da identificação de uma folha da estrela $K_{1,s-1}$ com uma folha do caminho P_n, com $n \geq 4$ e $s \geq \lceil \frac{n-1}{2} \rceil$. Este resultado reforça a conjetura de que todos os caterpillars com diâmetro maior ou igual a cinco são 0-rotativos.*

1. Introdução

Seja $G = (V(G), E(G))$ um grafo com conjunto de vértices $V(G)$ e conjunto de arestas $E(G)$. Uma *rotulação graciosa* de G é uma função injetora $f\colon V(G) \to \{0,\dots,|E(G)|\}$, tal que a rotulação de arestas induzida $g\colon E(G) \to \{1,\dots,|E(G)|\}$, dada por $g(uv) = |f(u)-f(v)|$, $uv \in E(G)$, é bijetora. Um grafo que admite uma rotulação graciosa é chamado *gracioso*. Alguns exemplos de grafos graciosos são exibidos na Figura 1.

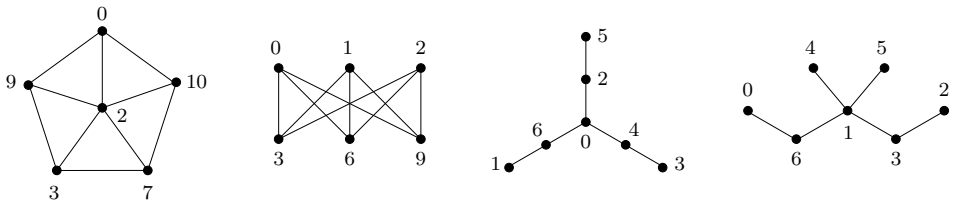


Figura 1. Exemplos de grafos com uma rotulação graciosa.

A rotulação graciosa foi introduzida em 1967 por Rosa [Rosa 1967] e sua origem está relacionada com outra conjetura denominada *Conjetura de Ringel-Kotzig*, que afirma que todo grafo completo com $2m+1$ vértices possui uma decomposição cíclica em $2m+1$ subgrafos, cada um dos quais isomorfo a uma determinada árvore com m arestas. Como uma forma de abordar a Conjetura de Ringel-Kotzig, Rosa introduziu quatro tipos de rotulação em grafos, incluindo rotulações graciosas, e propôs a seguinte conjetura.

Conjetura 1 ([Rosa 1967])**.** *Todas as árvores são graciosas.*

Esta conjetura é comumente conhecida como *Conjetura das Árvores Graciosas*. Em seu artigo seminal, Rosa provou que se a Conjetura das Árvores Graciosas é verdadeira, então a Conjetura de Ringel-Kotzig é verdadeira. Desde que foi proposta, a Conjetura das Árvores Graciosas tem mantido vários pesquisadores empenhados em sua solução, embora pouco ainda se saiba sobre sua validade para árvores arbitrárias. Resultados recentes envolvendo rotulações graciosas podem ser consultados na resenha dinâmica mantida por Gallian [Gallian 2015].

Assim que se começa a investigar rotulações graciosas de árvores, torna-se clara a importância de saber como construir rotulações graciosas com o rótulo 0 atribuído a um vértice específico. A importância do rótulo 0 em uma rotulação graciosa de uma árvore T deve-se ao fato de que é fácil expandir T adicionando k novas folhas ao vértice que foi rotulado com 0, posteriormente, atribuindo rótulos $|E(T)|+1, \ldots, |E(T)|+k$ a estas novas folhas. Uma árvore T é 0-*rotativa* se, para todo $v \in V(T)$, existe uma rotulação graciosa f de T tal que $f(v) = 0$.

A 0-rotatividade em árvores foi primeiramente investigada por Rosa [Rosa 1967]. Em 1977, o autor provou que todos os caminhos são 0-rotativos [Rosa 1977]. Entretanto, em 1969, alguns exemplos de árvores que não são 0-rotativas foram descobertas [Duke 1969]. Por exemplo, a menor árvore que não é 0-rotativa é exibida na Figura 2.

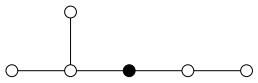


Figura 2. Uma árvore que não é 0-rotativa. Não existe rotulação graciosa desta árvore que atribua o rótulo 0 ao vértice preto.

Posteriormente, Chung e Hwang investigaram a 0-rotatividade de um produto de duas árvores T_1 e T_2 denominado Δ-*construção*, denotado $T_1 \Delta T_2$, e provaram que se T_1 e T_2 são 0-rotativas, então $T_1 \Delta T_2$ também o é [Chung and Hwang 1981]. Em 2004, Bussel mostrou que todas as árvores com diâmetro no máximo três são 0-rotativas [Bussel 2004]. O autor também mostrou que existem árvores com diâmetro quatro que não são 0-rotativas. De fato, ele caracterizou completamente as árvores com diâmetro quatro que não são 0-rotativas usando o seguinte resultado.

Teorema 2 ([Bussel 2004])**.** *Seja T uma árvore com diâmetro quatro tal que o seu centro v tem grau dois. Sejam v_1, v_2 vértices adjacentes a v e sejam m_1, m_2 o número de folhas adjacentes a v_1, v_2, respectivamente. Considere $m_1 \geq m_2$. A árvore T tem uma rotulação graciosa f com $f(v) = 0$ se e somente se existem inteiros x e r tais que $m_1 = (m_2 + 2 - x)(r-1) - x$ e tais que: (i) no máximo um elemento em $\{x, r\}$ é ímpar; (ii) $2 \leq r \leq |E(T)|/2$; e (iii) $0 \leq x \leq \min\{r-1, m_2\}$.* □

Seja $\mathcal{D}$ a classe das árvores com diâmetro quatro cujo centro tem grau dois e que não satisfazem as condições do Teorema 2. Seja $\mathcal{D}'$ a classe das árvores obtidas a partir da identificação de uma folha de um caminho P_n arbitrário, $n \geq 1$, com o centro de uma árvore em $\mathcal{D}$. Bussel provou que, dada uma árvore T com diâmetro quatro, T é 0-rotativa se e somente se $T \notin \mathcal{D}'$. Adicionalmente, ele mostrou que todas as árvores com no máximo 14 vértices que não são 0-rotativas pertencem à classe $\mathcal{D}'$. Baseado nestes resultados, o autor propôs a seguinte conjetura.

Conjetura 3 ([Bussel 2004])**.** *A classe $\mathcal{D}'$ contém todas as árvores que não são 0-rotativas.*

Desde que a 0-rotatividade de árvores foi estudada pela primeira vez, há 50 anos, ela tem sido considerada uma forma de abordar a Conjetura das Árvores Graciosas e, também, um problema desafiador em si mesmo. Em particular, uma família de árvores para a qual a propriedade de 0-rotatividade não é conhecida é a família dos *caterpillars*, definida a seguir. Uma árvore T é um *caterpillar* se T é um caminho ou o subgrafo obtido pela remoção de todas as suas folhas é um caminho.

De fato, note que, se a Conjetura 3 for verdadeira, então ela implica que todo *caterpillar* com diâmetro pelo menos cinco é 0-rotativo. Considerando esta observação, neste trabalho, nós investigamos a Conjetura 3 restrita a *caterpillars* e provamos que as seguintes famílias de *caterpillars* são 0-rotativas: (i) *caterpillars* com emparelhamento perfeito; e (ii) *caterpillars* obtidos a partir da identificação de uma folha da estrela $K_{1,s-1}$ com uma folha do caminho P_n, com $n \geq 4$ e $s \geq \lceil \frac{n-1}{2} \rceil$. Estes resultados fortalecem a Conjetura 3 e, particularmente, a conjetura de que todos os *caterpillars* com diâmetro pelo menos cinco são 0-rotativos.

2. Preliminares

Um *emparelhamento* em um grafo G é um subconjunto de arestas de $E(G)$, duas a duas não adjacentes. Dizemos que um vértice $v \in V(G)$ é *saturado* por M se v é extremo de alguma aresta de M. Um *emparelhamento perfeito* é um emparelhamento que satura todos os vértices do grafo. Seja T uma árvore com um emparelhamento perfeito M. A *árvore contraída* de T é a árvore T' obtida a partir de T pela contração de todas as arestas de M.

Broersma e Hoede [Broersma and Hoede 1999] introduziram o conceito de rotulação fortemente graciosa, definido a seguir. Seja T uma árvore com um emparelhamento perfeito M. Uma rotulação f de T é *fortemente graciosa* se f é uma rotulação graciosa e se $f(u) + f(v) = |E(T)|$ para toda aresta $uv \in M$. Um exemplo de rotulação fortemente graciosa é exibida na Figura 3.

Broersma e Hoede provaram que a Conjetura das Árvores Graciosas é verdadeira se e somente se toda árvore com emparelhamento perfeito tem pelo menos uma rotulação fortemente graciosa. Eles também estudaram o rótulo 0 em rotulações fortemente graciosas, como apresentado no próximo lema. Este resultado é importante para a demonstração do Teorema 5.

Lema 4 ([Broersma and Hoede 1999])**.** *Seja T uma árvore com emparelhamento perfeito M e seja $uv \in M$, $u, v \in V(T)$. Seja T' a árvore contraída de T e seja $x \in V(T')$ o*

vértice correspondente à aresta uv. Se T' tem uma rotulação graciosa f' com $f'(x) = 0$, então T tem duas rotulações fortemente graciosas f_1 e f_2 tais que: (i) $f_1(u) = 0$ e $f_1(v) = |E(T)|$; (ii) $f_2(u) = |E(T)|$ e $f_2(v) = 0$. □

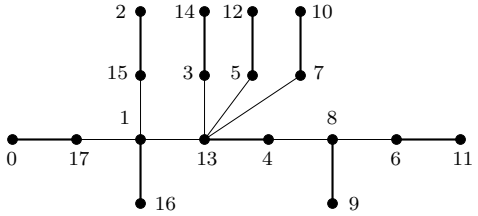


Figura 3. Rotulação fortemente graciosa de uma árvore.

3. Resultados

Nesta seção, os resultados principais são apresentados. Em particular, o Teorema 6 mostra que, para todo inteiro $d \geq 5$, existem *caterpillars* 0-rotativos com diâmetro d e com um número arbitrariamente grande de vértices. Os Teoremas 5 e 6 fortalecem a conjetura de que todos os *caterpillars* com diâmetro pelo menos cinco são 0-rotativos.

Teorema 5. *Todo caterpillar com emparelhamento perfeito é 0-rotativo.*

Demonstração. Seja T um *caterpillar* com emparelhamento perfeito M e seja $uv \in M$, $u, v \in V(T)$. Seja T' a árvore contraída de T e seja $x \in V(T')$ o vértice correspondente à aresta uv. Como T tem um emparelhamento perfeito, temos que T' é um caminho. Rosa provou que todo caminho é 0-rotativo [Rosa 1977]. Logo, T' é 0-rotativo. Portanto, T' tem uma rotulação graciosa f' tal que $f'(x) = 0$. Pelo Lema 4, T tem duas rotulações fortemente graciosas f_1 e f_2 tais que: $f_1(u) = 0$ e $f_1(v) = |E(T)|$; $f_2(u) = |E(T)|$ e $f_2(v) = 0$. Portanto, existem rotulações fortemente graciosas de T que atribuem o rótulo 0 ao vértice u ou ao vértice v. Como a aresta uv foi arbitrariamente escolhida, concluímos que o caterpillar T é 0-rotativo. □

Teorema 6. *Sejam $n \geq 4$ e $s \geq \lceil \frac{n-1}{2} \rceil$. Se T é o caterpillar obtido a partir da identificação de uma folha da estrela $K_{1,s-1}$ com uma folha do caminho P_n, então T é 0-rotativo.*

Esboço da demonstração. Seja T um *caterpillar* como definido na hipótese e seja $v \in V(T)$ um vértice arbitrário. Primeiro, escolhemos adequadamente uma aresta wz do P_n para ser removida de T. A componente que contém o vértice v é denominada T_1 e a outra componente é denominada T_2. Ajuste a notação para que $w \in V(T_1)$. Considere uma bipartição $\{V_1, V_2\}$ de $V(T_1)$ em conjuntos independentes tal que $v \in V_1$ e defina $k = |V_1|$. Assim, nós construímos duas rotulações injetoras f_1, f_2 para T_1, T_2, respectivamente, tais que $f_1 \colon V(T_1) \to \{0, \ldots, k-1\} \cup \{k + |E(T_2)| + 1, \ldots, |E(T)|\}$, $f_2 \colon V(T_2) \to \{k, k+1, \ldots, k + |E(T_2)|\}$, tais que: (i) $f_1(v) = 0$; (ii) os rótulos das arestas induzidos por f_2 são $1, 2, \ldots, |E(T_2)|$; (iii) os rótulos das arestas induzidos por f_1 são $|E(T_2)| + 2, \ldots, |E(T)|$; e (iv) $f_1(w)$ e $f_2(z)$ são tais que $|f_1(w) - f_2(z)| = |E(T_2)| + 1$. Finalmente, nós definimos uma rotulação f de T do seguinte modo:

$$\text{para } u \in V(T), f(u) = \begin{cases} f_1(u), & \text{if } u \in T_1; \\ f_2(u), & \text{if } u \in T_2. \end{cases}$$

Portanto, f é uma rotulação graciosa de T tal que $f(v) = 0$ e, como v é um vértice arbitrário, concluímos que T é 0-rotativo. □

4. Agradecimentos

Esta pesquisa foi financiada pela FAPESP, processos 2014/16987-1, 2014/16861-8, 2015/03372-1; e pela NSERC, processo 41705-2014 057082.

Referências

Broersma, A. J. and Hoede, C. (1999). Another equivalent of the graceful tree conjecture. *Ars Combinatoria*, 51:183–192.

Bussel, F. V. (2004). 0-Centred and 0-ubiquitously graceful trees. *Discrete Mathematics*, 277:193–218.

Chung, F. R. K. and Hwang, F. K. (1981). Rotatable graceful graphs. *Ars Combinatoria*, 11:239–250.

Duke, R. A. (1969). Can the complete graph with $2n + 1$ vertices be packed with copies of an arbitrary tree having n edges? *The American Mathematical Monthly*, 76(10).

Gallian, J. A. (2015). A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics*, DS6, 1–389.

Rosa, A. (1967). On certain valuations of the vertices of a graph. *Theory of Graphs (Internat. Sympos., Rome, 1966) Gordon and Breach, New York; Dunod, Paris*, 349–355.

Rosa, A. (1977). Labeling snakes. *Ars Combinatoria*, 3:67–73.

Dynamically Skewed Compressed Cache

Daniel R. Carvalho[1], **Rodolfo de Azevedo**[1]

[1]Computer Systems Laboratory – Universidade Estadual de Campinas (Unicamp)
Campinas – SP – Brazil

ra180791@students.ic.unicamp.br,rodolfo@ic.unicamp.br

Abstract. *Many levels of on-chip cache reduce the average memory latency at the cost of extra die area and power. To decrease the outlay of these extra components, cache compression techniques are used to store compressed data and allow a cache capacity boost. This project introduces Dynamically Skewed Compressed Cache, a modification of the Skewed Compressed Cache that minimizes the amount of invalid bytes inside the cache by allowing higher flexibility of both the compression and allocation stages. The Dynamically Skewed Compressed Cache will be tested and evaluated using ZSim, and the performance results will be compared and appraised to determine the validness of the use of the proposed technique.*

Resumo. *Muitos níveis de caches on-chip reduzem a latência média de memória ao custo de área e energia extra no die. Para diminuir o dispêndio desses componentes extras, técnicas de compressão de cache são usadas para armazenar dados comprimidos e permitir um aumento de capacidade de cache. Este projeto apresenta Cache Comprimida Distorcida Dinamicamente, uma modificação da Cache Comprimida Distorcida que minimiza a quantidade de bytes inválidos dentro da cache através da permissão de maior flexibilidade de ambos os estágios de compressão e de alocação. A Cache Comprimida Distorcida Dinamicamente será testada e avaliada usando ZSim, e os resultados de desempenho serão comparados e avaliados para determinar a válidade de utilização da técnica proposta.*

1. Context

Over the years electronic hardware has become faster and more efficient, but devices do not improve at equal rates. Off-chip memories, although increasing in capacity, present lower bandwidth and latency improvements than what is requested by current microprocessors [Hennessy and Patterson 2011] [Rogers et al. 2009]. Besides that, these memories require huge amounts of energy when compared to local accesses (*i.e.*, register access) [Nguyen and Wentzlaff 2015]. To try to overcome these problems, on-chip memories (caches), lower level memories with small latency and energy cost, but small storage capacity and high cost per bit, have been created [Mahapatra and Venkatrao 1999]. This dichotomy has led designers to try to increase effective cache size by compressing and compacting data blocks before inserting them on caches, a technique called cache compression.

The main goal of cache compression techniques is to virtually increase the size of the cache without the disadvantages of doing so, that is, improve power consumption and

cache capacity with as low latency, area and metadata overhead as possible. These overheads are inherent to such methods due to the extra hardware and wiring required, and the extra stages every access must go through: decompression on lookups and compression on writes. Besides, extra metadata is also necessary to inform either data compression state or location. These techniques must also be lossless so as to maintain correct processor behavior.

Skewed Compressed Caches (*SCC*) [Sardashti et al. 2014] is one such technique that relies on the fact that most workloads tend to have compression and spatial locality, that is, neighboring blocks tend to have similar compressibility and be inside the cache simultaneously, respectively, to gather neighboring blocks with the same compressibility in a single physical entry.

It does so by grouping up to 8 adjacent blocks (*super-blocks*), each with its corresponding super-block tag. These super-blocks are sparse, so they can be distributed along many data entries, according to its blocks' compressibilities (blocks with the same compressibility are placed on the same data entry). When they are placed in a data entry, a state tag informs the validness of each of the data entry's 8 sub-blocks (8 bytes each, totaling 64 bytes).

To execute cache lookups, first the compression factor must be determined, which is done using Equation 1 with all cache ways. Then, for each way, Equations 2 and 3 are used to find out the set index and byte offset, respectively. h_i represent the hash functions used for each compression factor. The tags of each of the possible data entries are then checked to verify if the super-block tag matches and the offset is valid. By using the state tag and the address of the block to determine the offset information, the block's location within the data entry will be known without using many extra bits. Figure 1 shows an example of possible mappings for block A for all its different compression factors using SCC. It is important to notice that a way can allocate blocks with different compression factors.

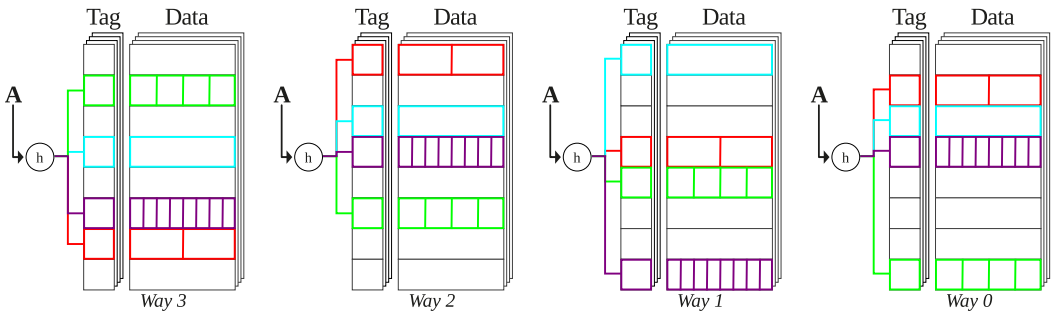


Figure 1. Different mappings for block A for all its different compression factors.

$$CF_1CF_0 = A_{10}A_9 \wedge W_1W_0 \tag{1}$$

$$SetIndex = \begin{cases} h_0(\{A_{47} - A_{11}, A_8, A_7, A_6\}); & CF == 0 \\ h_1(\{A_{47} - A_{11}, A_8, A_7\}); & CF == 1 \\ h_2(\{A_{47} - A_{11}, A_8\}); & CF == 2 \\ h_3(A_{47} - A_{11}); & CF == 3 \end{cases} \quad (2)$$

$$ByteOffset = \begin{cases} none; & CF == 0 \\ A_6 << 4; & CF == 1 \\ A_7A_6 << 3; & CF == 2 \\ A_8A_7A_6 << 2; & CF == 3 \end{cases} \quad (3)$$

For cache writes the computation of the set index, byte offset and way can be done in parallel using Equations 2, 3 and 4, respectively. As can be seen the compression factor of a block determines the way to be used, that is, SCC are highly dependent on compression locality. This lack of flexibility can become a limitation in some cases because it may generate a high rate of internal fragmentation in case a workload's neighboring blocks do not compress similarly.

$$W_1W_0 = A_{10}A_9 \wedge CF_1CF_0 \quad (4)$$

2. Objective

To try to overcome SCC's dependence on compression locality we propose the *Dynamically Skewed Compressed Cache* (*DSCC*), a method that dynamically changes the compression factor according to the data entries' free space in order to reduce internal fragmentation. By doing so it also allows more blocks to be placed simultaneously in the cache, as blocks are replaced when suitable.

2.1. The Dynamically Skewed Compressed Cache

As mentioned before, the Skewed Compressed Cache scheme uses a fixed compression rate (*i.e.*, a given block will always be compressed to the same compression factor) to allocate memory blocks, in which a block is compressed to its maximum compressibility before being written to the cache. This approach, along with the distribution of compression factors along the ways, manages to increase cache efficiency and reduce accesses to the main memory, and thus improves system performance and energy usage.

However, by using this method most of the super-blocks will have holes (bits within data entries without valid contents). For example, if we were to sequentially allocate two blocks A and B belonging to the same super-block, where A is compressible to 32 bytes, and B is compressible to 16, two data entries would be used using the SCC scheme. Figure 2.1 (left) illustrates the example. For the sake of clarity, all data in the cache is assumed to be invalid, unused ways are omitted, and the rest of the cache is left blank.

The problem in this approach is that it focuses on maximizing block compression, which does not necessarily maximize the quantity of blocks on the cache. In this paper we propose *Dynamically Skewed Compressed Cache* (*DSCC*), a variant of the Skewed Compressed Cache that minimizes the amount of holes in the cache. It does so by dynamically changing the size of the compressed block in order to find the best fit.

When a block is compressed to be written, instead of directly writing the block to the way given by its compression factor as in the SCC scheme, all ways are checked so that a data entry with a worse compression factor, but which already contains an entry from the block's super-block, is favored to a data entry with the given compression factor, but that belongs to another super-block, and thus writing to it will require an eviction. If the worse compression factor way is chosen, the block is compressed to its maximum compression factor and the extra bits can be filled with leading zeroes.

Like SCC, DSCC is organized in super-blocks, and each data entry is associated to a sparse super-block tag. Each tag contains valid bits, bits that determine the state of its referred data entry. The SCC scheme allows a data entry to be divided in up to 8 blocks of 8 bytes, and each tag entry contains 8 bits. This allows a one-to-one mapping of the states to the blocks.

Nonetheless, in the proposed scheme the compression factor of a block is not necessarily its maximum compression factor, and thus extra information is required in order to decompress data correctly. Instead of using a binary representation for the states (*valid* and *invalid*), a second bit must be added to the state information, such that the valid bits can also represent the compression factor. Equation 5 shows the possible values and their corresponding meanings.

$$ValidBits \begin{cases} I & = 00 = Invalid \\ V & = 01 = Valid \quad (CF) \\ V^{-} & = 10 = Valid \quad (CF-1) \\ V^{--} & = 11 = Valid \quad (CF-2) \end{cases} \tag{5}$$

Note that a block with compression factor of 3, that is, a block that occupies 8 bytes can be used as an uncompressed block, in other words, use all 64 bytes of the data entry (CF = 0). In this representation, it would be a valid block with CF-3. and thus would not be represented by the above valid bits. Although it is a possibility, it is an undesired one, as using 64 bytes would later require a recompression as another block from the super-block is writen. Thus, for efficiency reasons, no blocks that can be compressed are left uncompressed, and the third bit for state representation is not needed.

Using the DSCC scheme, the previous example would result in the usage of a single data entry, as B would be compressed to 32 bytes in order to fill the hole on A's data entry. A's valid bits will be set to 01 (V), that is, valid with maximum compression, and B will have 10 (V^{-}) as its valid bits, since its maximum compression factor is 1 level higher than what it has been compressed to. Figure 2.1 (right) shows the allocation process for this example.

3. Methodology

In order to emulate typical working conditions, ZSim [Sanchez and Kozyrakis 2013], a x86-64 simulator, will be used. ZSim is a fast, scalable and accurate multi-core architectural simulator that does dynamic binary translation to perform instruction decoding and timing analysis during the instrumentation phase and thus reduce the amount of tasks to be repeated during simulation. We chose ZSim because it outperforms current popular simulators like Sniper [Carlson et al. 2011], which uses approximation techniques to

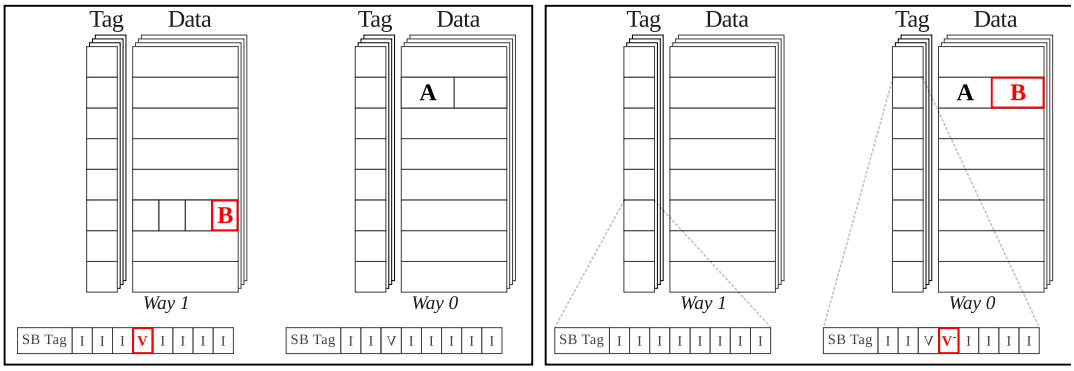


Figure 2. Allocation of blocks A and B, where A compresses to 32, and B to 16 bytes bytes. *Left:* Using SCC block B is compressed to its maximum and written to the cache on way 1. *Right:* Using DSCC block B is sub-compressed to 32 bytes and written to the cache on way 0.

do timed interval instead of instruction-wise simulation, and gem5 [Binkert et al. 2011], which emulates the instructions, that is, it decodes and invokes functional and timing models for every instruction.

The validation stage will be performed at the Computer Systems Lab of the Institute of Computing at Unicamp, as it provides ZSim and a cluster, which allows high parallel computation and, thus, faster results.

The processor design will be the same for all compression schemes, and will take into account common specifications for current processors. The systems will be tested using the benchmarks from Parsec [Bienia 2011] and SPEC 2006 [Henning 2006]. Tests will focus on common computer operations under the same environmental conditions.

As a final step the results from both systems will be compared regarding *1)* cache evictions, that is, which system evicted the least amount of memory blocks; *2)* power efficiency, that is, which system provided the least amount of energy usage to finish the tasks; *3)* chip die area used, in other words, how much space is required to synthesize each CPU; *4)* development complexity, a comparison between the difficulties and trade-offs of the systems; *5)* time taken to perform the tasks; and *6)* estimated overall cost. Usage validness of the Dynamically Skewed Compressed Cache will be determined via an analysis of these factors to verify at which conditions it could be a better scheme than the ones being used in processors nowadays.

For area and power efficiency evaluation we will use Cacti 6.5 [Muralimanohar et al. 2009], an integrated cache and memory access time, cycle time, area, leakage, and dynamic power model.

4. Conclusion

In this project we aim to improve the Skewed Compressed Cache by under-compressing data to minimize the amount of holes in the cache. We are currently implementing the SCC and DSCC techniques using ZSim. Afterwards tests will be conducted, and the results will be evaluated regarding cache evictions, power efficiency, die area, complexity,

time and overall cost.

We expect the cache to be used more efficiently and to generate less evictions, resulting in an overall improvement. Due to the extra hardware required to analyze and relocate blocks according to their compression factors the energy cost of a single cache level will increase, but the reduction of accesses to the lower level caches will likely equalize energy usage.

References

Bienia, C. (2011). *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University.

Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., et al. (2011). The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7.

Carlson, T. E., Heirman, W., and Eeckhout, L. (2011). Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 52. ACM.

Hennessy, J. L. and Patterson, D. A. (2011). *Computer architecture: a quantitative approach*. Elsevier.

Henning, J. L. (2006). Spec cpu2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17.

Mahapatra, N. R. and Venkatrao, B. (1999). The processor-memory bottleneck: problems and solutions. *Crossroads*, 5(3es):2.

Muralimanohar, N., Balasubramonian, R., and Jouppi, N. P. (2009). Cacti 6.0: A tool to model large caches. *HP Laboratories*, pages 22–31.

Nguyen, T. M. and Wentzlaff, D. (2015). Morc: a manycore-oriented compressed cache. In *Proceedings of the 48th International Symposium on Microarchitecture*, pages 76–88. ACM.

Rogers, B. M., Krishna, A., Bell, G. B., Vu, K., Jiang, X., and Solihin, Y. (2009). Scaling the bandwidth wall: Challenges in and avenues for cmp scaling. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 371–382, New York, NY, USA. ACM.

Sanchez, D. and Kozyrakis, C. (2013). Zsim: Fast and accurate microarchitectural simulation of thousand-core systems. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, pages 475–486, New York, NY, USA. ACM.

Sardashti, S., Seznec, A., Wood, D., et al. (2014). Skewed compressed caches. In *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pages 331–342. IEEE.

Multi-scale Morphological Image Simplification Based on Extrema Relationships: Improvements and Applications

Darwin Saire Pilco[1], **Neucimar J. Leite**[1]

[1]Institute of Computing – University of Campinas, UNICAMP
Campinas, SP, Brazil, 13083-852

Abstract. *Image simplification has been proved useful in several image processing applications as an additional step for more complex tasks, such as segmentation and feature extraction. In this work, we explore a graph-based simplification method that guarantees a well-behaved suppression of the image extrema (maxima/minima) by taking into account both information of distance and contrast, as well as some interesting aspects of the scale-space theory. By highlighting some new properties of the method, we define a local update of the graph which implies in an interesting bypass in the whole algorithm structure which, originally, is very time-consuming. Finally, we illustrate how to combine this simplification process with well-known morphological tools to approach problems related mainly with multi-scale image segmentation and homogenization.*

Resumo. *A simplificação de imagens tem mostrado ser útil em diversas aplicações de processamento de imagens como um passo adicional para tarefas mais complexas, como a segmentação e extração de características. Neste trabalho, exploramos um método de simplificação baseado em grafos que garante o bom comportamento da supressão nos extremos da imagem (máximas / mínimas), tendo em conta a informação da distância e contraste, bem como alguns aspectos interessantes da teoria do espaço-escala. Para destacar algumas novas propriedades do método, nós definimos uma atualização local do grafo que implica um desvio interessante em toda a estrutura do algoritmo que, inicialmente, é custoso. Finalmente, nós mostramos como combinar este processo de simplificação com ferramentas morfológicas bem conhecidas para abordar problemas relacionados, principalmente, com segmentação de imagens multi-escala e homogeneização.*

1. Introduction

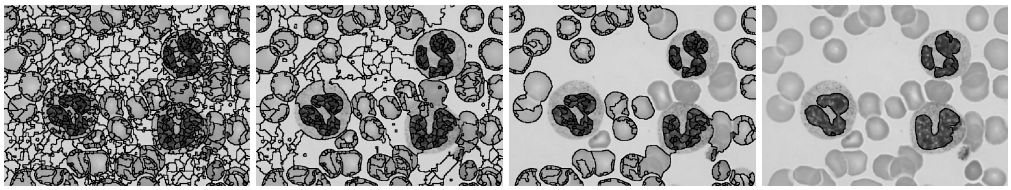
The area of image processing and analysis includes a large amount of applications ranging from the lowest level tasks (e.g. extremum points detection) to the more specialized ones, such as segmentation and classification, requiring the deletion of unnecessary details of the input images. In such a case, it is common to apply a pre-processing step to the original image before any further consideration. Often, this step aims not only to filter out noisy components, for instance, but also to simplify the image through the elimination of non-significant details, while keeping the information necessary to the achievement of the desired outcome.

Image simplification is commonly referred to as a pre-processing step and, in particular, Mathematical Morphology [Serra 2003], [Soille 2013], [Heijmans and Roerdink 1998],

[Najman and Talbot 2013] introduces interesting low-level simplification filters exhibiting well-known properties [Serra and Vincent 1992]. The typical filtering by opening and closing [Soille 2013] and their combination as alternate sequential filters are commonly used to eliminate undesirable components of an image while preserving its main content. The multi-scale approach based on the scale-space theory [Witkin 1984] defines a multi-level processing (from finer to coarser scales) related to different representations of the original signal. In such a case, the simplification should be well-behaved in the sense that, given a certain feature of interest (e.g., the zero-crossing of a function), one seeks to track its representation along the different scales. This multi-level transformation should satisfy some properties of $monotonicity, continuity, fidelity$ and $euclidean\ invariance$ [Witkin 1984].

Morphologically, the leveling approach in [Meyer 2004][Meyer et al. 1997] defines a reconstruction-based simplification without changes in the final contours w.r.t the original image. Another example of morphological simplification is the dynamic measure [Bertrand 2007] which selects components of an image according to the notion of extinction values (e.g., area or volume) [Vachier 2001]. This procedure is closely related to the measure of persistence of a signal and is used to eliminate image components regarded as non-significant. Another method based on scale-dependent non-flat structuring function is discussed in [Dorini and Leite 2013], where a toogle-like transformation simplifies an image in a self-dual way.

Recently, the work in [Polo and Leite 2013] introduced a non-self-dual simplification method taking into account the relationship between image extrema. More specifically, it considers the distance and contrast between the various regional maxima (minima) and define a total order relation closely linked to the degree of simplification one wants to impose. As we will see later, this multi-level process establishes a non-decreasing and well-behaved activity from which the least significant extrema merge successively with the most significant ones, in a given neighborhood.



(a) σ=24.5 ℓ=4 (b) σ=4.58 ℓ=17 (c) σ=1.67 ℓ=55 (d) σ=0.03 ℓ=27

Figura 1. Monotonic reduction of image extrema with final watershed segmentation.

In this work, we explore this graph-inspired simplification process aiming at improvements in terms of algorithm and applications. We highlight some new properties concerned with the local update of the graph configuration that yields, among others, a huge speed-up in execution time. We also propose new means to combine this algorithm with different morphological tools and explore the homogenization aspect led by the well-behaved merging of our features of interest (the image extrema). An overview of our work can be seen in Figures 1 and 2. The Figure 1, we show some tuples of the aforementioned order relation. These tuples were used here to define a meaningful segmentation of the image represented mostly by its deepest regional minima. The Figure 2 shows the bypass introduced here to approach the high computational cost of the original algorithm.

2. Methodology

The main concepts behind the simplification process explored in this work consider the notion of scale-dependent erosion an dilation, recently explored, for example, in [Dorini and Leite 2013], [Polo and Leite 2013].

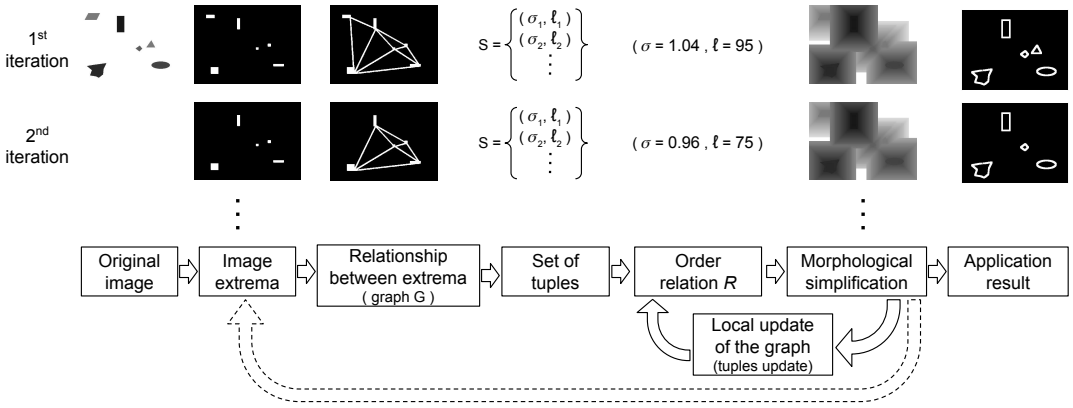


Figura 2. The original algorithm and the bypass obtained from a local update of the graph.

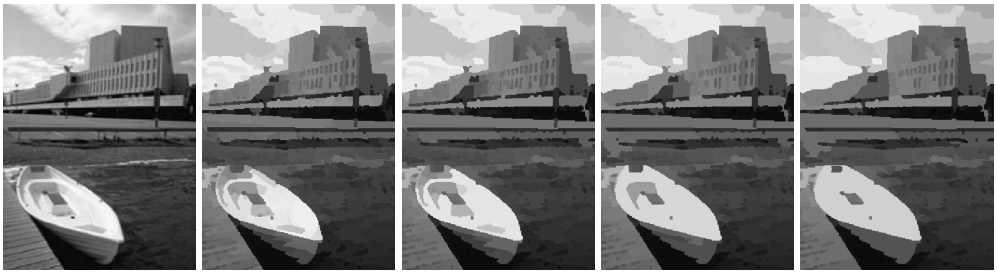
For this work we have used several scale define by $\sigma = \frac{|f(u)-f(v)|}{\ell(X,Y)}$ where $\ell(X,Y)$ is the separation between the regional minima(maxima) X,Y, $u \in X$ and $v \in Y$. Also is used a non-flat pyramid-shaped structuring function g_σ, which is 0 in the position $(0,0)$ and $-\sigma$, otherwise. Beside, the order relationship R is defined by $(\sigma_p, \ell_p) \prec (\sigma_o, \ell_o) \Longleftrightarrow \sigma_p > \sigma_o \vee (\sigma_p = \sigma_o \wedge \ell_p < \ell_o)$, which is responsible for selecting the tuple to remove the extrema least significant.

Figure 2 depicts the main steps of image simplification algorithm. First, a graph G expressing the neighborhood connectedness of the regional maxima (minima) is defined. Further, the set $S = \{(\sigma_1, l_1), \ldots (\sigma_n, l_n)\}$ of tuples is defined and an order relation R is obtained for the current simplification step represented, for instance, by the highest tuples generating very few simplification activity. Finally morphological operations of erosion and dilation are used for simplification process.

Since we reiterate this whole process, from the least to the most significant extrema, the main drawback of this algorithm is its high computational cost given by the graph construction and tuples sorting. The dotted lines represent the original process and the bypass connection defined by a local update of the graph shows the new method providing an speed up in simplification process, achieved through the discovery of new properties, which we have not mention by the limited space. And finally we have a application using as marker function for watershed segmentation the extrema of the simplification process.

3. Results and Applications

In these examples, a regular 8-connected grid graph is considered and the function z is given by the well-known watershed transform [Najman and Talbot 2013].

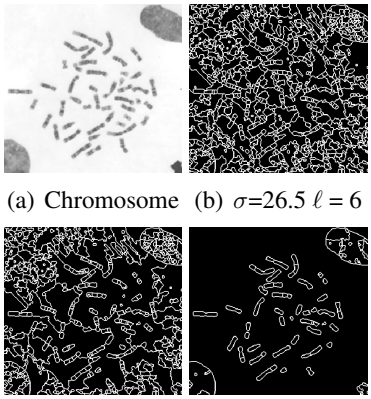


(a) Image (b) σ=59.0 ℓ=3 (c) σ=10.9 ℓ=11 (d) σ=6.4 ℓ=9 (e) σ=5.1 ℓ=17

Figura 3. Successive simplifications obtained from morphological reconstruction of the original image, using the extrema defined at each scale as markers.

In Figure 3 shows a set of simplified images obtained from morphological reconstructions [Soille 2013].

In such case, the marker functions are represented by the input regional minima simplified with the tuples disposed in descending order. Note the significant reduction of the number of image extrema with level and position preservation of the remaining contours.
In Figure 4 we illustrate how the watershed transform can be combined with the simplification method to approach its typical over-segmentation problem. In these both cases, we use the regional minina obtained at each iteration as markers for the watershed function. Note the monotonic reduction of the over-segmented regions and the convergence of the method to a more meaningful and treatable segmentation result.



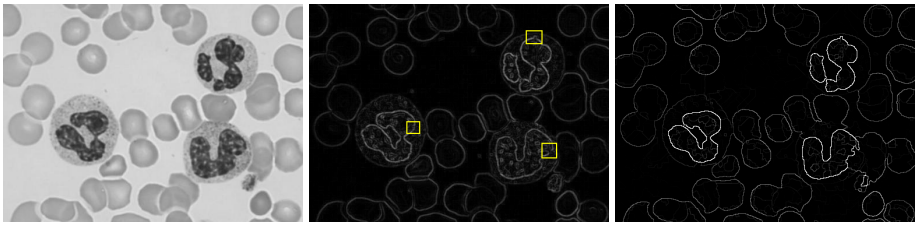
(a) Chromosome (b) σ=26.5 ℓ = 6 (c) σ=5.11 ℓ=28 (d) σ=0.9 ℓ=11

Figura 4. Simplification combined with the watershed.

(a) (b) (c) (d) (e)

Figura 5. Image homogenization example. (a) Tomography lung, (b) Laplacian contours, (c) High boost contours, (d) Homogenization and (e) Homogenized contours.

Figure 5 shows the effect led by the homogenization of the regions when merged in a well-behaved manner. 5(e) depicts the contours of the homogenized



(a) Cell. (b) Morphological gradient. (c) Enhancement gradient.

Figura 6. Enhancement gradient example.

image 5(d), highlighting its better quality w.r.t methods such as Laplacian and high-boost [Gonzalez and Woods 2006].

Figure 6 shows an enhancement gradient, consisting of the average of all gradients obtained by the reconstructed morphological image at each scale. The yellow box in Figure 6(b) represents the weak gradients from which a leaking can occur in a morphological segmentation.

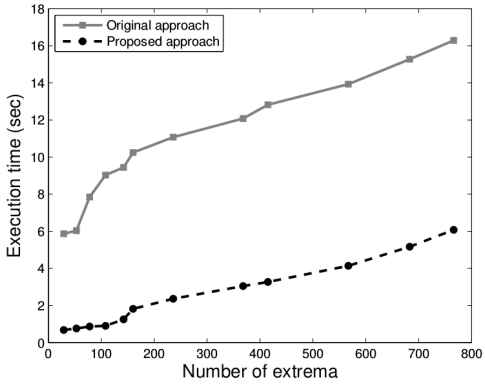


Figura 7. Computational time comparison.

The improvement of the computational time achieved by considering the new properties described in Section 2 is illustrated in Figure 7, where the number of generated images is proportional to the number of image extrema. All the experiments were conducted using a system equipped with Intel Core i7-3770K processors per node, running at 3.50 Ghz with RAM 31Gb. In this case, the test image is of size 358×481 pixels (Figure 3(a)) and the varying number of extrema was obtained by a simple blurring process.

4. Conclusion

In this paper, we dealt with a morphological simplification taking into account the information of distance (separation) and contrast between the extrema of a signal. We explored scale-space properties, such as monotonicity and continuity, as well as the graph-based implementation, to introduce new results concerned with the local update of a graph providing structure to a given image. These results led to significant improvements in terms of computational cost. Unlike the work in [Polo and Leite 2013], which discussed mainly theoretical aspects of the original approach, we showed through some examples how to combine the simplification step with well-known morphological tools in applications related to image segmentation and homogenization.

Future works on this matter include, for example, the automatic and adaptive learning of the considered tuples, aiming at less supervised multi-scale tasks, and the investigation of the use of these tuples in problems related to image description and representation.

Studies with different shapes of the structuring functions and other scaled operations such as erosion and dilations should also be of interest.

Referências

Bertrand, G. (2007). On the dynamics. *Image and Vision Computing*, 25(4):447 – 454. International Symposium on Mathematical Morphology 2005.

Dorini, L. B. and Leite, N. J. (2013). A scale-space toggle operator for image transformations. *International Journal of Image and Graphics*, 13(04):1350022.

Gonzalez, R. C. and Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Heijmans, H. J. and Roerdink, J. (1998). *Mathematical morphology and its applications to image and signal processing*, volume 12. Springer Science & Business Media.

Meyer, F. (2004). Levelings, image simplification filters for segmentation. *Journal of Mathematical Imaging and Vision*, 20(1-2):59–72.

Meyer, F., Vachier, C., Oliveras, A., and Salembier, P. (1997). Morphological tools for segmentation : connected filters and watersheds. *Annales Des Telecommunications*, 52(7-8):367–379.

Najman, L. and Talbot, H. (2013). *Mathematical morphology*. John Wiley & Sons.

Polo, G. and Leite, N. (2013). From extrema relationships to image simplification using non-flat structuring functions. In Hendriks, C., Borgefors, G., and Strand, R., editors, *Mathematical Morphology and Its Applications to Signal and Image Processing*, volume 7883 of *Lecture Notes in Computer Science*, pages 377–389. Springer Berlin Heidelberg.

Serra, J. (2003). *Image Analysis and Mathematical Morphology*, volume 1. Academic Press.

Serra, J. and Vincent, L. (1992). An overview of morphological filtering. *Circuits, Systems and Signal Processing*, 11(1):47–108.

Soille, P. (2013). *Morphological image analysis: principles and applications*. Springer Science & Business Media.

Vachier, C. (2001). Morphological scale-space analysis and feature extraction. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 3, pages 676–679. IEEE.

Witkin, A. P. (1984). Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 150–153. IEEE.

Programação como ferramenta de ensino de pensamento computacional

Laís V. Minchillo[1], Juliana F. Borin[1], Edson Borin[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 – Cidade Universitária, Campinas/SP - Brasil

lminchillo2@gmail.com, juliana@ic.unicamp.br, edson@ic.unicamp.br

Abstract. *Computational Thinking is a useful skill to solve problems in all fields of science. Efforts around the world aim to teach this skill to children, and in some countries it's already part of the curriculum. We propose using programming as a tool to teach computational thinking, including the development of a mobile application. This task brings multiple challenges: definition of programming concepts to teach, teaching methods, choosing target age group, choosing a methodology for testing and final evaluation. We expect to find that teaching programming is indeed a valid way to teach computacional thinking.*

Resumo. *Pensamento Computacional é uma habilidade útil para resolução de problemas em todas as áreas de conhecimento. Iniciativas no mundo todo têm como objetivo o ensino desta habilidade para crianças, e em alguns países isto já é parte do currículo escolar básico. Propomos o uso da programação como ferramenta de ensino de pensamento computacional, com o desenvolvimento de um aplicativo para plataformas móveis. Esta tarefa traz diversos desafios: definição de conceitos de programação a ensinar, métodos de ensino, escolha da faixa etária alvo, criação de uma metodologia de testes e avaliação final. Esperamos constatar ao final deste trabalho que o ensino de programação de fato traz o aprendizado de pensamento computacional.*

1. Introdução

Pensamento computacional, do inglês computational thinking, é uma ferramenta na resolução de problemas que se aplica a todas as áreas de conhecimento. O termo foi popularizado Wing (2006), que caracteriza pensamento computacional como uma habilidade para resolver problemas, projetar sistemas e entender o comportamento humano através de conceitos fundamentais para a Ciência da Computação.

O pensamento computacional não deve ser visto como uma habilidade técnica, e sim como uma forma de organização de pensamento e de resolução de problemas, e é natural considerar o seu ensino na escola [Wing 2006, Barcelos e Silveira 2012, França e Amaral 2013, HITSA 2012, Code.org 2015, CSTA 2011]. Com a difusão do ensino de pensamento computacional - matéria que em alguns países já faz parte do currículo escolar -, é esperado que a nova geração e as seguintes tenham um melhor entendimento de tecnologia e suas diferentes aplicações.

O ensino de pensamento computacional a crianças não é uma necessidade nova - Papert (1975) publicou um artigo sobre o tema. Segundo ele, crianças aprendem fazendo

e pensando sobre o que fazem, e a inovação no ensino deve trazer melhores coisas para se fazer e melhores jeitos de se pensar. O autor também afirma que a computação é de longe a fonte de inovação mais rica que se pode encontrar no campo de educação. Papert foi também um dos criadores de Logo, uma linguagem de programação cujo objetivo era prover um ambiente divertido para crianças aprenderem matemática e conceitos de programação [Logo Foundation 1991].

Os currículos escolares de diversos países europeus já inclui o ensino de programação e outros conceitos relacionados ao pensamento computacional (como pensamento lógico, resolução de problemas, abtração, entre outros) [HITSA 2012, UK Department for Education 2013, European Schoolnet 2015]. Nos Estados Unidos, o presidente Obama lançou no início do ano uma iniciativa para o ensino de Ciência da Computação no currículo escolar K-12[1] [Smith 2016]. A Sociedade Brasileira de Computação está trabalhando para incluir o ensino de Computação no currículo escolar brasileiro [SBC 2016]. Pensamento computacional não é igual a saber programar, mas é uma habilidade utilizada por programadores. Este projeto tem, portanto, o objetivo de desenvolver uma ferramenta de apoio ao ensino de pensamento computacional através do ensino de programação.

O restante deste documento está organizado da seguinte forma: a seção 2 discute o ensino: quais conceitos deverão ser ensinados e como este conteúdo poderá ser dividido em módulos. A terceira seção mostra uma proposta de desenvolvimento de aplicativo para plataformas móveis. Finalmente, a quarta e última seção traz uma conclusão e resultados esperados.

2. Programação: conceitos e módulos de ensino

Embora o termo pensamento computacional tenha sido amplamente discutido recentemente, não existe uma definição absoluta de o que é ou como deve ser ensinado. O ensino através de programação é recorrente na literatura. Nesta seção serão explorados os conceitos de programação mais citados por diferentes autores. Podemos considerar que eles formam uma base de fundamentos para o aprendizado e ensino de pensamento computacional. Podem ser citados:

- Sequência: uma série de passos individuais.
- Algoritmo: uma sequência de instruções para resolver determinada tarefa.
- Repetição: a execução da mesma sequência múltiplas vezes.
- Evento: uma ação externa que dispara uma sequência de comandos.
- Condicional: tomada de decisões baseada em uma condição pré definida.
- Depuração, teste: executar um algoritmo para encontrar eventuais falhas.
- Decomposição de problemas, função, modularização: dividir um problema grande em outros menores que podem ser resolvidos mais facilmente. Cada subproblema pode ser resolvido com uma função: um conjunto de instruções que produzem uma saída a partir de uma entrada.
- Repetição e condicional aninhados: dentro de uma repetição pode ser encontrada outra repetição. Analogamente, um condicional também pode ter outro condicional aninhado.

[1] K-12 corresponde ao Ensino Fundamental no Brasil

- Recursão: uma função que chama a si mesma.
- Paralelismo: executar mais de uma instrução ao mesmo tempo, ou executar mais de uma sequência de instruções ao mesmo tempo. Tarefas paralelas podem ser independentes ou não.

A Tabela 1 relaciona a lista de conceitos e os autores que os citaram.

Tabela 1: Conceitos de pensamento computacional e autores que os citaram

	França e Amaral 2013	Code.org 2015	CSTA 2011	Brennan 2011	Brennan e Resnick 2012	Hambrusch et al 2009
Sequência	●	●	●	●	●	
Algoritmos		●	●			
Repetição	●	●	●	●	●	●
Evento	●	●	●	●	●	
Condicional	●	●	●	●	●	●
Depuração	●	●	●		●	
Teste	●	●	●			
Decomposição, funções, modularização	●	●	●		●	●
Repetição aninhada, condicional aninhado		●	●			
Paralelismo	●		●	●		
Recursão	●		●			●

Existem propostas que organizam estes conceitos em módulos de ensino [Code.org 2015, CSTA 2011]. Cada módulo tem uma faixa etária alvo e aborda diferentes conjuntos de conceitos de pensamento computacional. As Tabelas 2 e 3 mostram duas propostas de divisão em módulos para ensino.

No âmbito da educação, o mapeamento entre conceitos a serem ensinados e faixas etárias é controverso. Diferentes crianças têm diferentes níveis de aprendizado, o que Piaget (1972) chama de estágios cognitivos de conhecimento. Desse modo, mais importante do que fazer uma divisão dos conceitos por faixa etária é fazer uma ordenação dos conceitos. Assim, os conceitos podem ser estudados por crianças de diferentes idades que estejam na mesma faixa de aprendizado.

Tabela 2: Módulos para ensino de pensamento computacional, Code.org

Nível	Idades	Pré-requisitos	Conceitos
1	4 a 6	Leitura (iniciantes)	Sequência, repetições, eventos, resolução de problemas
2	6 ou mais (Indicado 7 a 10)	Leitura, matemática	Condicionais, algoritmos, depuração
3	6 ou mais (Indicado 9 e 10)	Módulo 2 completo	Decomposição de problemas, funções, repetições aninhadas, condicionais aninhados

Tabela 3: Módulos para ensino de pensamento computacional, CSTA K-12

Nível	Idades	Conceitos
1	8 a 11	Algoritmos, resolução de problemas, design e implementação, teste, divisão em subproblemas
2	11 a 14	Algoritmos, design e implementação, paralelização, abstração, divisão em subproblemas, verificar diferentes algoritmos que resolvem o mesmo problema
3	14 a 17	Uso de funções, parâmetros, classes e métodos pré-definidos na resolução de problemas, descrever os passos para desenvolver um software, explicar como sequências, iterações e recursões são usados para construir algoritmos, modelagem e simulação de ambientes, abstração, programação paralela

A Tabela 4 mostra uma proposta de ordenação dos conceitos de pensamento computacional em três níveis. Será necessário definir, para cada conceito, um método para seu ensino. Além disso, uma faixa etária alvo deverá ser escolhida para os testes e desenvolvimento inicial, tendo em mente a pretensão de futuramente expandir a interface do aplicativo para outras faixas etárias.

Tabela 4: Divisão dos conceitos de pensamento computacional em níveis

Nível 1	Nível 2	Nível 3
Sequência Algoritmo Teste Depuração	Decomposição de problemas Evento Condicional Repetição	Paralelismo Recursão Repetições aninhadas Condicionais aninhados

3. Aplicativo para plataformas móveis

Já existem ferramentas, aplicativos e jogos que oferecem o ensino de pensamento computacional, programação ou Ciência da Computação. A maioria delas, entretanto, está disponível somente em inglês, não traz um plano de ensino abrangente e não oferece produtos por preços acessíveis.

Propomos o desenvolvimento de um aplicativo gratuito, em português e que contará com exercícios que cobrem integralmente os conceitos de programação listados na seção anterior. O aplicativo será baseado em programação por blocos, e um diferencial será a possibilidade de comunicação com plataformas robóticas. A título de testes, usaremos um robô em desenvolvimento pelo mesmo grupo de pesquisa[2].

O desenvolvimento tem as seguintes fases: criação da interface, integração de uma biblioteca de programação por blocos, definição dos blocos a serem utilizados, comunicação externa com a plataforma robótica e criação das fases do jogo.

A biblioteca escolhida para a programação por blocos é a Blockly, desenvolvida pelo Google [Google for Education 2012]. Ela permite a criação de blocos e customização do código criado a partir deles.

4. Conclusão e resultados esperados

O ensino de pensamento computacional ainda apresenta diversos desafios. Dentre eles, podemos destacar:

- **O que é pensamento computacional?**
  Conforme discutido neste artigo, não há um consenso quanto a definição de pensamento computacional ou como seu ensino deve ser feito.
- **Quais conceitos devem fazer parte de um plano de ensino de pensamento computacional?**
  São poucos os artigos científicos que listam conceitos para o ensino de pensamento computacional. A segunda seção deste artigo contribui nesta direção, ao explorar o ensino de programação como meio de ensinar pensamento computacional.
- **Como avaliar a aprendizagem de pensamento computacional?**
  O processo de ensino-aprendizagem sempre culmina em alguma forma de avaliação, de modo que se possa diagnosticar a evolução do conhecimento e da habilidade dos alunos com relação aos objetivos estabelecidos. Da mesma forma como não está claro como e o quê ensinar, não está definido como será o processo de avaliação.

De forma geral, é esperado que no fim deste trabalho seja constatado que o ensino de programação é uma forma válida de ensino de pensamento computacional. As diversas iniciativas que utilizam este meio nos indicam que esta talvez seja uma boa maneira de prosseguir. Além disso, o aplicativo desenvolvido poderá servir como um guia de como o conteúdo é ensinado.

---

[2] Este projeto tem sido financiado pela empresa Tecsinapse. O grupo de pesquisa inclui um aluno de Iniciação Científica, orientado pelo Prof. Edson Borin, que está desenvolvendo o robô.

Referências

Barcelos, T. S., Silveira, I. F. (2012) Pensamento Computacional e Educação Matemática: Relações para o Ensino de Computação na Educação Básica, XX WorkShop sobre Educação em Computação, XXX Congresso da Sociedade Brasileira de Computação.

Brennan, K. (2011) Creative computing: a design-based introduction to computational thinking.

Brennan, K., Resnick, M. (2012) New frameworks for studying and assessing the development of computational thinking, AERA.

Code.org. (2015) Curriculum, disponível em https://code.org/educate/curriculum.

CSTA. (2011) K-12 Computer Science Standards, disponível em https://csta.acm.org/Curriculum/sub/K12Standards.html.

European Schoolnet. (2015) Computing our future, disponível em http://www.eun.org/publications/detail?publicationID=661.

França, R. S., Amaral, H. J. C. (2013) Proposta Metodológica de Ensino e Avaliação para o Desenvolvimento do Pensamento Computacional com o Uso do Scratch, Congresso Brasileiro de Informática na Educação.

Google for Education. (2012), Blockly, disponível em https://developers.google.com/blockly/.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., Hosking, A. L. (2009) A multidisciplinary approach towards computational thinking for science majors, SIGCSE, 183-187.

HITSA, Information Technology Foundation for Education. (2012) Programming at Schools and Hobby Clubs, disponível em http://www.innovatsioonikeskus.ee/en/programming-schools- and-hoby-clubs.

Logo Foundation. (1991) Logo, disponível em http://el.media.mit.edu/logo-foundation/.

Papert, Seymour. (1975) Teaching Children Thinking, Innovations in Education & Training International 9.5, 245-55.

Piaget, J. (1972) The Psychology of the Child, New York: Basic Books.

SBC. (2016) SBC realiza reunião para discutir a nova versão da Base Nacional Comum Curricular, disponível em http://www.sbc.org.br/noticias/.

Smith, Megan. (2016) Computer Science For All, The White House, disponível em https://www.whitehouse.gov/blog/2016/01/30/computer-science-all.

UK Department for Education. (2013) National curriculum in England: computing programmes of study, disponível em https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study.

Wing, Jeannette M. (2006) Computational Thinking, Communications of the ACM 49.3, 33.

Secure and Efficient Software Implementations of Neeva Hash Function for ARM Architectures

Tiago Reis[1], **Julio López**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brazil

Abstract. *This paper presents a series of software implementations and benchmarkings for Neeva, a novel design of lightweight hash function, on ARM-based devices. We show how to optimize the given algorithm for 32-bit architectures and how to protect it against certain side-channel attacks, besides comparing its final performance to that of other hash functions.*

Resumo. *Este artigo apresenta uma série de implementações em software e medidas de desempenho para Neeva, um novo projeto de função de hash leve, em plataformas ARM. Nós mostramos como otimizar este algoritmo para arquiteturas de 32 bits e como protegê-lo contra certos ataques de canal lateral, além de comparar seu desempenho final àquele de outras funções de hash.*

1. Introduction

The rise of the Internet of Things (IoT) [Atzori et al. 2010] has brought to the area of cryptography new challenges, since typical IoT platforms suffer from computational constraints that call for specially designed algorithms and protocols. The subarea of cryptography responsible for proposing and studying such algorithms is known as *Lightweight Cryptography*, as a reference to the use of fewer resources than usual [Dinu et al. 2015].

Among many novel designs on this area, we focus here on Neeva [Bussi et al. 2016], a hash function with security level of 112 bits whose only documented implementation attempt has been done by its authors and has no remarks about optimization efforts. Therefore, it is relevant to conduct a study to optimize Neeva implementations and analyze how its performance fares in comparison to other hash functions on the same platforms, which is our proposal to this paper.

On next sections, we present the specification for the algorithm itself, discuss the security properties of our implementations, describe our target architecture, detail relevant implementation aspects, show the benchmarking results and, finally, conclude our work.

2. The Neeva Hash Function

The Neeva hash function [Bussi et al. 2016], or Neeva-hash, is an instance of a sponge function [Bertoni et al. 2008], a structure commonly used to support hash functions designs consisting of three steps:

1. *Initialization phase*. The input message should be padded so it can be divided in an integral number of r-bit blocks. For Neeva, we have $r = 32$.
2. *Absorbing phase*. Here, the r-bit message blocks are XORed into the first r bits of the state, one by one, interleaved with applications of a permutation f.

3. *Squeezing phase*. As long as the output is requested, the first r-bits of the state are returned as output blocks, always interleaving applications of f over the state. Since Neeva has output length $|Z| = 224$ bits and $r = 32$, this process has to be done 7 times.

Having described the sponge framework, we shall now proceed to the specifics of Neeva-hash design. The main aspect left to describe the hash function is the permutation f, which is specified in Algorithm 1. The S-box (substitution box) this algorithm refers to is a 4-bit to 4-bit permutation designed for the block cipher PRESENT [Bogdanov et al. 2007] and is shown in Table 1.

Table 1. S-box used by Neeva-hash, given in hexadecimal notation.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Algorithm 1: Neeva-hash permutation f.

Data: $B = B_0B_1...B_{256}$, a 256-bit state
for $i = 0$ **to** 31 **do**
  $B \leftarrow S(B)$ Applying S-box 64 times
  $B = b_0||b_1||...||b_{15}$, b_m is a 16-bit word, $0 \leq m \leq 15$
  for $j = 0$ **to** 3 **do**
    for $k = 0$ **to** 2 **do**
      $b_{4j+k} \leftarrow b_{4j+k} \oplus b_{4j+3}$
    end
  end
  $B \leftarrow rotl_8(B)$
  for $j = 0$ **to** 15 **do**
    $b_j \leftarrow b_j \boxplus_{2^{16}} RC_{i,j}$
  end
end
return B

3. Security of Implementations

In our work, we are concerned with side-channel attacks, that is, attacks crafted based on information obtained from the physical implementation of a cryptographic primitive [Chen et al.]. For instance, an attacker may gather data such as execution time of an algorithm [Kocher 1996] or power consumption [O'Flynn and Chen 2014] and, through these data, gain access to sensitive information processed by a machine. Henceforth, we focus on timing attacks, since they are the ones most closely related with the software implementation. Furthermore, we have access to the FlowTracker [Silva et al. 2015] tool, which points out vulnerabilities that may be used in timing attacks, and thus have helped us guaranteeing the security of all implementations presented in this work.

At this point, it is relevant to clarify what we consider to be sensitive information. Since our object of study is a hash function, we treat as secret the input message, for the fact that, in various applications, critical data is directly hashed and the leak of the hash function input could compromise the security of a whole system.

4. Target architecture and its features

Currently, there is a vast variety of processors well suited to be integrated to the IoT. Although, here we will focus on some representatives of ARM architecture, since it is the world leader in the market of microprocessors and, thus, attracts most relevant academic works, as well as commercial interest [ARM]. In particular, we will test our implementation on four platforms with different 32-bit processors: a Cortex-M3, a Cortex-M4, a Cortex-A9 and a Cortex-A15.

A relevant feature of the Cortex-A is that they have access to a Single Instruction Multiple Data (SIMD) extension called NEON, whose instructions act upon 64 or 128 bit vectors, applying the same operation to a series of lanes. Thus, one can organize a great amount of data in one vector and use NEON to process them in parallel.

5. Implementation

In this section, we discuss the most relevant details of our implementations and present the time measurements made in each platform studied. Since there is no publicly available reference implementation for Neeva hash function, we start analyzing a naive implementation and then we comment on what can be improved and how relevant are the gains.

5.1. Naive implementation

As a baseline, we start off with a simple working code, which we call a *naive implementation*. Our intention was to produce C-language code as close as possible to the pseudocode in Algorithm 1. However, minding the protection against side-channel attacks, we are required to make some adaptations.

The main adaptation regards the S-box, which, if implemented as a simple lookup table, may result in different memory access patterns and, thus, different execution times for different inputs. A secure alternative follows:

```
uint8_t neeva_sbox(uint8_t x) {
    uint8_t A, B, C, D;
    uint8_t E, F, G, H;

    /* Input x has four bits: A, B, C and D */
    A = (x>>0) & 0x1;     C = (x>>2) & 0x1;
    B = (x>>1) & 0x1;     D = (x>>3) & 0x1;

    /* Output bits are calculated */
    E = (A&B&C) ^ (A&B&D) ^ (A&C&D) ^ A ^ B ^ (B&C) ^ D ^ 1;
    F = (A&B) ^ (A&B&D) ^ (B&D) ^ C ^ (A&D) ^ (A&C&D) ^ D ^ 1;
    G = (A&B&C) ^ (A&B&D) ^ (B&D) ^ B ^ (A&C&D) ^ (C&D) ^ D;
    H = A ^ C ^ D ^ (B&C);

    /* Result is combined into a four-bit integer */
    return (H<<0) | (G<<1) | (F<<2) | (E<<3);
}
```

Table 2 presents the performance results for this naive implementation.

Table 2. Performance measured for naive implementation of Neeva hash function.

Microprocessor	Cortex-M3	Cortex-M4	Cortex-A9	Cortex-A15
Performance (cycles / byte)	55228.31	55260.63	22719.26	16966.72

5.2. S-box optimization

The first improvement we can make to the previously shown S-box algorithm comes from the realization that that function only makes use of four bits of the input, not needing the remaining bits to operate upon or to store data. Hence, these other bits can be used to perform the same operations simultaneously. Using a 32-bit word as input and output to the S-box function should, then, drop the number of calls to this function by a factor of eight, since each group of four bits can calculate an application of the S-box. After this improvement, the program yielded much faster times, as can be seen in Table 3.

Table 3. Performance measured for Neeva hash function with optimized S-box function.

Microprocessor	Cortex-M3	Cortex-M4	Cortex-A9	Cortex-A15
Performance (cycles / byte)	9550.41	6967.17	3934.47	2954.93

5.3. Further improvements

As suggested by the analysis of the S-box optimization, carrying out operations in 32-bit variables may reduce the overall number of instructions needed to run the code, even though some steps of Neeva seem adequate to be operated on 16-bit variables. Besides the application of the S-box to 32-bit words, summing the state to round constants also need interesting techniques to be carried out on 32-bit variables. The best solution we found is represented on Algorithm 2.

Algorithm 2: Summing 16-bit words modulo 2^{16} in 32-bit registers.

Input: Two integers $A = A_1||A_2$ and $B = B_1||B_2$, where A_i, and B_i are 16-bit integers, for $i = 1, 2$.
Output: An integer $C = C_1||C_2$ such that $A_i + B_i \equiv C_i \mod 2^{16}$, for $i = 1, 2$.
$T_0 \leftarrow (A + B) \mod 2^{32}$
$T_1 \leftarrow A \oplus B$
$T_2 \leftarrow T_0 \oplus T_1$
$T_3 \leftarrow T_2$ & 0x00010000
$C \leftarrow T_0 - T_2$
return C

The performance results after these optimizations are listed in Table 4. These are the best results we could obtain without using SIMD instructions, and, therefore, the best results overall for the Cortex-M microprocessors.

Table 4. Performance measured for fully optimized Neeva hash function, but with no SIMD instructions.

Microprocessor	Cortex-M3	Cortex-M4	Cortex-A9	Cortex-A15
Performance (cycles / byte)	4970.42	3996.94	2349.09	2025.61

5.4. Implementation using NEON

Each round of the Neeva-hash permutation f presents a structure that make parallelism possible. Dividing the 256-bit state as a sequence of sixteen 16-bit words, we can allocate

these words into NEON vectors by grouping up those whose indexes are congruent modulo 4. For example, the first vector would contain words 0, 4, 8 and 12. This arrangement has no impact on the performance of the S-box appliance or on the sums to the round constants, but greatly simplifies the XORing and rotating steps. For the S-box calculation, the NEON instruction VTBL can be used to simulate a lookup table efficiently.

Table 5 shows performance measurements for Neeva implemented with NEON instructions. The speed up obtained here is very relevant, especially on the Cortex-A15, which is 3.7 times faster than our best code that does not make use of SIMD instructions.

Table 5. Performance measured for Neeva hash function using NEON instructions.

Microprocessor	Cortex-A9	Cortex-A15
Performance (cycles / byte)	979.31	546.62

6. Results analysis and comparison with literature

As the only performance results publicly available for Neeva come from its own authors, we refer to them, who measured a speed of 12067 cycles/byte in an Intel Core 2 duo processor [Bussi et al. 2016]. We cannot compare these numbers directly with our results because we are working on different architectures, although, since we were able to produce faster code in all of our low-end platforms compared to a higher-end processor, we have reasons to believe our code is better optimized than that of the authors.

However, when comparing to state-of-the-art implementations of other hash functions, we are able to find faster alternatives, even among those with higher security levels. For example, both SHA-256 and Keccak-256 claim to have security level of 128 bits and, when implemented on Cortex-A9 and Cortex-A15 processors, have been able to process messages in less than 100 cycles/byte [eba]. Numbers from this order of magnitude have also been achieved on Cortex-M3 for SHA-256 [Tschofenig and Pegourie-Gonnard].

7. Conclusion

In this work we have investigated various different strategies for implementing Neeva hash function as efficiently as possible while still minding the security of the implementation against some sorts of side-channel attacks. We were able to produce very efficient code relative to a straightforward implementation, especially when using NEON vector instructions.

Even though we were able to devise interesting techniques for optimization and have indications that our code is better optimized than the version written by Neeva designers, our performance measurements do not fare well in comparison with the state-of-the-art implementations for other hash functions. Therefore, unless further improvements to the implementations are made, there are better choices for lightweight hash functions to be used in platforms similar to those we analyzed.

We can, then, consider as one of the main contributions of our work the publication of a benchmarking of Neeva hash function in relevant platforms for the context of IoT. Furthermore, we showed how to implement efficiently and securely important pieces of code that could be reused, in particular, the PRESENT S-box. At last, we can also

consider our NEON implementation as a relevant result, since it showcases a practical example of how vector instructions can provide high speed up if the vectors are arranged conveniently to take advantage of structural properties of the algorithm.

References

eBACS: ECRYPT Benchmarking of Cryptographic Systems. `https://bench.cr.yp.to/results-hash.html`. Accessed: June, 2016.

ARM. ARM — The Architecture for the Digital World. `http://www.arm.com/`. Accessed: June, 2016.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

Bertoni, G., Daemen, J., Peeters, M., and Assche, G. V. (2008). On the indifferentiability of the sponge construction. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer.

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B., Seurin, Y., and Vikkelsoe, C. (2007). PRESENT: an ultra-lightweight block cipher. In Paillier, P. and Verbauwhede, I., editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer.

Bussi, K., Dey, D., Biswas, M. K., and Dass, B. K. (2016). Neeva: A lightweight hash function. *IACR Cryptology ePrint Archive*, 2016:42.

Chen, S., Wang, R., Wang, X., and Zhang, K. Side-channel leaks in web applications: a reality today, a challenge tomorrow. `http://research.microsoft.com/pubs/119060/WebAppSideChannel-final.pdf`. Accessed: June, 2016.

Dinu, D., Corre, Y. L., Khovratovich, D., Perrin, L., Großschädl, J., and Biryukov, A. (2015). Triathlon of lightweight block ciphers for the internet of things. *IACR Cryptology ePrint Archive*, 2015:209.

Kocher, P. C. (1996). Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Koblitz, N., editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer.

O'Flynn, C. and Chen, Z. (2014). Side channel power analysis of an AES-256 bootloader. *IACR Cryptology ePrint Archive*, 2014:899.

Silva, B. R., Pereira, F. M. Q., and Aranha, D. F. (2015). Uma técnica de análise estática para detecção de canais laterais baseados em tempo. In *XV Simpósio Brasileiro de Segurança da Informação e Sistemas Computacionais (SBSeg 2015)*, pages 16–29.

Tschofenig, H. and Pegourie-Gonnard, M. Performance of state-of-the-art cryptography on ARM-based microprocessors. `http://csrc.nist.gov/groups/ST/lwc-workshop2015/presentations/session7-vincent.pdf`. Accessed: June, 2016.

F

Taming Kernels and Krakens With Control-Flow Integrity

João Moreira[1], **Sandro Rigo**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251 - Cidade Universitária, Campinas - SP, 13083-852

joao.moreira@lsc.ic.unicamp.br, sandro@ic.unicamp.br

Abstract. *This paper describes the implementation of a control-flow integrity (CFI) framework that focuses on protecting the Linux kernel. By combining source-code and binary analysis, our framework maps valid execution paths into a fine-grained control-flow graph, which is later used to instrument the kernel with label-based CFI checks that prevent control-flow hijacking attacks. The system induces an average overhead of 17% on system call latency and 5% on I/O throughput, while its impact on real-world applications is ≈1%.*

Resumo. *Esse artigo descreve a implementação de um mecanismo de controle de fluxo (CFI) para Kernel do Linux. Através da combinação de técnicas de análise de binários e de compilação, este framework mapeia caminhos de execução válidos em um grafo de fluxo de controle de fina granularidade, que depois é utilizado no processo de instrumentação do código do Kernel. Neste processo, verificações para previnir ataques ao controle de fluxo do sistema são inseridas. O sistema introduz um custo computacional de 17% nas latências de chamadas ao sistema e 5% na banda de operações de comunicação. No entanto, seu impacto em aplicações reais é de ≈1%.*

1. Introduction

Control-flow attacks target the modification of the regular instruction sequence of a computer system, leading to unexpected behaviors. These attacks usually take advantage of misplaced memory operations to overwrite control data, such as function pointers or return addresses. By modifying these data, attackers redirect control-flow to impose a malicious program logic, built obtain privileges on the target computer system.

Different defenses have been proposed since the appearance of the first control-flow hijacking attacks. These defenses, one after another, were proven inefficient as more attack methodologies were discovered and exposed by researchers. Amongst the solutions, Control-Flow Integrity (CFI) [Abadi et al. 2005] stands out as a practical alternative, compatible with most existent software and hardware, capable of preventing powerful attacks to which other solutions are inefficient, such as return-oriented programming (ROP) based attacks [Checkoway et al. 2010].

Here we present a fine-grained CFI framework focused on the Linux kernel that is capable of preventing control-flow attacks. To the best of our knowledge, this is the first fine-grained CFI mechanism that supports the Linux kernel. The system presents small overheads, presenting averages of 17% on micro-benchmarks focused on stressing kernel code and 0.9% on user-land benchmarks running over the protected kernel.

2. Background

Control-Flow Integrity (CFI) [Abadi et al. 2005] is a technique originally proposed to ensure the correctness of control-transfer instructions on software. The general idea behind CFI consists in instrumenting the program with instructions (guards) that will validate the targets of indirect control-flow transfers before its execution happens. This way, these mechanisms are capable of deflecting ROP attacks variants [Shacham 2007, Checkoway et al. 2010, Bletsch et al. 2011]. Different techniques for verifying the validity of a target exist, but the most generic one consists in dereferencing the pointer used in the indirect control transfer and verifying if it is actually pointing towards pre-defined values (tags). In this sense, enhancing software with CFI consists in correctly calculating the set of valid targets for every indirect control-transfer instruction, instrumenting these instructions with guards and the destinations with the respective tags.

CFI is said to be coarse-grained [Zhang et al. 2013, Mingwei and Sekar 2013] if its control-flow target validation is loose, allowing returns to any site after a *call* instruction and indirect calls to the first instruction of any function. This approach was proven vulnerable as valid malicious flows could be traced inside protected applications [Göktas et al. 2014, Carlini and Wagner 2014]. Motivated by that, fine-grained CFI mechanisms were developed [Tice et al. 2014, Mashtizadeh et al. 2015], tightening the control-flow target validation through only allowing returns to instructions after a *call* instruction to the respective returning function and indirect calls to the first instruction of functions that follow a pre-defined rule, such as having matching prototype with the code pointer used in the call. Despite these efforts, the problem remains open, specially on what concerns specific domains, such as the operating system (OS) kernel. In this direction, all the implementations known are coarse-grained [Criswell et al. 2014] or are yet incomplete to support large code bases such as the Linux Kernel [Ge et al. 2016].

The first challenge on implementing an efficient fine-grained CFI scheme for a program consists in building a CFI-specific control-flow graph (CFG) from where the sets of valid targets for a control-transfer will be extracted. For CFI-specific we mean a specialized CFG that brings information about valid targets for indirect control-transfers between functions from different modules. We call this graph an *indirect control-flow graph (ICFG)*, in order to make it clearly different from the CFG in the traditional compiler theory, which concerns on mapping direct transitions between basic-blocks.

Nodes in the ICFG represent functions or groups of functions and edges represent indirect control-transfers normally seen in code that was written in C. These transfers are regular return instructions, indirect calls through function pointers, and indirect jumps. From now on, we will refer to the control-transfers decurrent of return instructions as *back-edges* and to the other control-transfers as *forward-edges*.

On what concerns the implementation of a CFI for the Linux Kernel, it is important to notice that most of its code is written in C, with a significant part written in Assembly and frequent interactions between code written in both languages. This is specially important because the underlying infrastructure used to implement the described framework imposes limitations on what concerns instrumenting Assembly code.

3. CFI for the Kernel

Our framework is a fine-grained CFI mechanism for the Linux kernel. Its implementation was mostly written as compilation passes inside the LLVM compiler [Lattner and Adve 2004], making use of its API and Intermediate Representation (IR) in order to instrument kernel code. It supports modules, as long as they were also compiled and instrumented together with the whole system compilation. Due to differences between the Kernel source-code and LLVM syntax standards, the kernel code was patched with llvm-linux project [Linux Foundation] patches. Despite its focus on the Linux kernel, the principles behind this framework are general enough to be applied on any OS.

The approach used relies on two primitives, which are *tags* and *guards*. Tags are used to mark special spots in the binary. Guards are used to make sure that a certain pointer points to a specific spot, marked by a pre-defined tag, before it is actually used. Theses primitives consist in sequences of assembly instructions exemplified in Figure 1.

Figure 1(a) shows the standard tag we use for marking code throughout the whole kernel binary. The tag is actually encoded as the value being copied in a `movl` instruction. The destination address of this instruction is a global variable reserved by the compiler for this unique purpose. As the value in this address is never read, the whole scheme benefits from write-back cache strategies.

Figure 1(b) shows a forward-edge where the highlighted instruction `callq *%rcx` is protected by the instructions above it. In line 1, the value pointed to by `rcx` is verified through the `cmpl` instruction. This instruction will dereference the value pointed to by `rcx` plus an offset and compare it to the expected tag. The offset exists because the actual tag in the target address is encoded inside the `movl`, which has a few first bytes representing the instruction opcode. The following instruction, `je`, will branch the code to line 6 if the previous comparison matches. If the comparison does not match, `je` does not branch and the code flows to line 3, that will push the attempted target address into the stack for analysis by a violation handler function, which is called in the next line. Once invoked, the violation handler reports a control-flow mismatch and returns to the next instruction, which will restore the stack and execute the original call. Figure 1(c) is a back-edge guard example, where the highlighted `ret` instruction is similarly protected.

For the scheme to work properly, the instruction tags must be correctly placed in the destination of the control-transfers. This way, the correspondent tag must be placed at the prologue of functions that can be called through an indirect call and also right after each call instruction, as the instruction that will be executed right after the callee returns.

4. Evaluation

We evaluate the framework from perspectives of performance, security and coverage. First, we assess the performance overheads with the benchmarks, LMBench [McVoy and Staelin 1996] and SPEC2006 [Henning 2006]. Second, we evaluate the security provided by the framework on protecting the system against control-flow attacks.

The system used on all tests was an Intel(R) Core(TM) i7-6700 8 core CPU @ 3.40GHz, with 32GB RAM memory, and running Debian Linux with GNU kernel v3.19.0 on VMware Workstation 12 Player. Every test unit was executed 10 times, and an average was calculated. No tests were executed in parallel. In the results described below, we

(a) tag

```
1 movl $0x3a66ac66,0xffffffff8585e000
```

(b) forward-edge guard

```
1 cmpl  $0x56777818,0x7(%rcx)
2 je    <6>
3 push  %rcx
4 callq <violation_handler>
5 pop   %rcx
6 callq *%rcx
```

(c) back-edge guard

```
1 mov   (%rsp),%rdx
2 cmpl  $0x17c204b6,0x7(%rdx)
3 je    7
4 push  %rdx
5 callq <violation_handler>
6 pop   %rdx
7 retq
```

Figure 1. Example Tags and Guards

compared: (i) a regular compilation of the Linux vanilla kernel in minimal configuration (Vreg) and (ii) the same kernel compiled with CFI instrumentation (Vcfi).

4.1. Performance: LMBench and SPEC2006

In order to verify the effects of CFI on the kernel performance, we picked the LMBench [McVoy and Staelin 1996] micro-benchmark. This benchmark stresses the system in its OS domain, allowing a close observation of the overhead side-effects generated on tasks such as performing system calls or page-fault handling.

We evaluated the latencies of OS capabilities for executing: null syscalls; I/O critical syscalls (which are read/write, fstat and select); open/close syscalls; the installation of a signal handler; process creation followed by exit, execve and /bin/sh; context switching between processes; select syscall on 100 file descriptors; page fault handling and inter-process communication with socket and pipe. The bandwidth was measured while communicating through pipe, unix sockets (AF_UNIX) and TCP sockets. The results for these tests can be seen in Figure 2, which shows the overheads on latency and bandwidth while comparing Vcfi to Vreg.

We measured an average latency overhead of 17.75% when comparing Vcfi against Vreg. The smallest overheads appeared in *null syscall*, which did not show discernible values, and in *fork+exec*, which had a 9%overhead. A *select* operation on 100 file descriptors (*select 100 fd's*) presented the biggest overhead, which is 42%.

The bandwidth tests describe efficiency on communication. Vcfi presented an average bandwidth overhead of 5.33%. The biggest value (9%) was verified on *pipe*, while *AF_Unix* presented 7% overhead and *TCP* did not present significative overhead.

In computer systems, most useful work is usually performed by user-land applications and, to understand CFI overheads on such applications we tested the benchmark SPEC 2006 [Henning 2006] which is mostly composed by user-land CPU-intensive applications. As in these cases time spent on kernel code is minimal, we expect SPEC overheads to be even less than the overheads verified on LMBench. For all applications present in the SPEC-INT 2006 suite, Vcfi showed an average overhead of 0.9%, with a maximum of 1.5% for *456.hmmer* and a minimum of 0.3% for *473.astar*. So, the low overhead expectation was confirmed by the experiments.

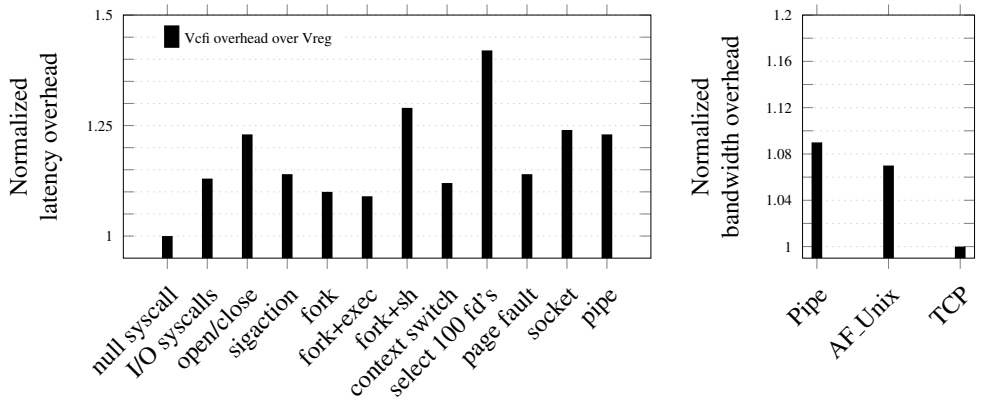


Figure 2. CFI performance on LMBench

4.2. Security

Our framework extends existent coarse-grained CFI mechanisms. As a fine-grained implementation, it has a smaller set of valid paths for the binary, imposing harder obstacles for exploitation. If the same kernel we used on these tests was compiled with both approaches, the function pointers with the prototype `void ()` would have 1130 valid target functions in our system while on a coarse-grained system this set would be ≈2533% more loose, allowing 28632 functions as valid targets. On the compiled system, the function prototype `void ()` is the most common one, thus 1130 functions is the biggest allowed set, what makes this proportion even larger when comparing other prototypes. The average number of allowed destinations for a function pointer on our framework is 10.6.

This work also differs from other OS CFIs [Ge et al. 2016] by providing support to the Linux Kernel, which is the larger supported OS code-base; by not relying on code modifications which harm kernel features such as module support and for building its ICFG differently, in a way which does not restrict the use of language features.

4.3. Size, Code Coverage and Exploitation Feasibility

We verified a code-size overhead of 15.9% when comparing Vcfi to Vreg.

Assembly is not processed by LLVM in its IR form and, thus, is not protected by CFI. Statistics about unprotected code show that, in total, the compiled kernel has 28537 functions from which 351 (1.2%) are written in assembly. The final binary has 28865 return instructions from which 85 (0.3%) were not protected, and 6053 indirect calls, from which 8 (0.2%) were not protected. The majority of the code was covered by CFI, imposing relevant difficulties to attackers intended to compromise such system.

5. Conclusions

In this paper we presented a CFI solution for the Linux kernel. To implement CFI capable of protecting an OS, specific challenges were faced, such as defining a control-flow graph that describes valid flows and is compatible domain requirements.

Our system proved to be an efficient solution against control-flow attacks and presented low overheads, showing an average of 17% on kernel micro-benchmarks. We also shown that this overhead is significantly amortized for regular applications, reaching an average of 0.9%, when measured in the SPEC 2006 benchmark.

References

Abadi, M., Budiu, M., Erlingsson, U., and Ligatti, J. (2005). Control-flow integrity. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, pages 340–353, New York, NY, USA. ACM.

Bletsch, T., Jiang, X., Freeh, V. W., and Liang, Z. (2011). Jump-oriented programming: A new class of code-reuse attack. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 30–40, New York, NY, USA. ACM.

Carlini, N. and Wagner, D. (2014). Rop is still dangerous: Breaking modern defenses. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 385–399, San Diego, CA. USENIX Association.

Checkoway, S., Davi, L., Dmitrienko, A., Sadeghi, A.-R., Shacham, H., and Winandy, M. (2010). Return-oriented programming without returns. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 559–572, New York, NY, USA. ACM.

Criswell, J., Dautenhahn, N., and Adve, V. (2014). Kcofi: Complete control-flow integrity for commodity operating system kernels. In *2014 IEEE Symposium on Security and Privacy*, pages 292–307.

Ge, X., Talele, N., Payer, M., and Jaeger, T. (2016). Fine-grained control-flow integrity for kernel software. In *IEEE European Symposium on Security and Privacy 2016*, Euro S&P, Washington, USA. IEEE Computer Society.

Göktas, E., Athanasopoulos, E., Bos, H., and Portokalidis, G. (2014). Out of control: Overcoming control-flow integrity. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 575–589, Washington, DC, USA. IEEE Computer Society.

Henning, J. L. (2006). SPECCPU 2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17.

Lattner, C. and Adve, V. (2004). LLVM: A compilation framework for lifelong program analysis & transformation. In *Proceedings of the International Symposium on Code Generation and Optimization: Feedback-directed and Runtime Optimization*, CGO '04, Washington, USA. IEEE Computer Society.

Linux Foundation. LLVMLinux. `http://llvm.linuxfoundation.org/`. Accessed 2016-05-22.

Mashtizadeh, A. J., Bittau, A., Boneh, D., and Mazières, D. (2015). Ccfi: Cryptographically enforced control flow integrity. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 941–951, New York, NY, USA. ACM.

McVoy, L. and Staelin, C. (1996). Lmbench: Portable tools for performance analysis. In *Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*, ATEC '96, pages 23–23, Berkeley, CA, USA. USENIX Association.

Mingwei, Z. and Sekar, R. (2013). Control flow integrity for cots binaries. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 337–352, Washington, D.C. USENIX.

Shacham, H. (2007). The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 552–561, New York, NY, USA. ACM.

Tice, C., Roeder, T., Collingbourne, P., Checkoway, S., Erlingsson, Ú., Lozano, L., and Pike, G. (2014). Enforcing forward-edge control-flow integrity in gcc & llvm. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 941–955, San Diego, CA. USENIX Association.

Zhang, C., Wei, T., Chen, Z., Duan, L., Szekeres, L., McCamant, S., Song, D., and Zou, W. (2013). Practical control flow integrity and randomization for binary executables. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, Washington, DC, USA. IEEE Computer Society.

Testes online baseados em modelos para arquitetura baseada em serviços dinâmicos

Lucas C. Leal [1], **Eliane Martins**[1], **Andrea Ceccarelli**[2]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 Cidade Universitária, Campinas/SP - Brasil, CEP 13083-852

[2]Dipartimento di Matematica e Informatica – Università degli Studi di Firenze
Viale Morgagni, 67/a - 50134 Firenze, Italia.

Abstract. *Service oriented architecture (SOA) applications can evolve and change during their execution, and their dynamicity and unpredictability are the biggest challenges to overcome in order to execute online tests. Since offline tests do not completely assure the correct behavior of the system during runtime, finding ways to perform online tests are not just desired but needed. This project's goal is to generate and perform online tests on a dynamic SOA application. In order to achieve this goal, model-based engineering and and self-adaptive systems solutions will be studied and combined. A self-adapted testing service (SATS) will be created, which will be integrated to a testing platform previously developed.*

Resumo. *Aplicações de arquiteturas baseadas em serviços (SOA) podem evoluir e mudar durante sua execução, e sua dinamicidade e imprevisibilidade são os maiores desafios a serem superados para que testes online possam ser executados. Como testes offline não garantem completamente o comportamento correto de uma aplicação durante sua execução, meios para executar testes online não são só desejados mas também são necessários. O objetivo desse projeto é a geração e execução de testes em uma aplicação baseada em SOA. Para alcançar esse objetivo, soluções de engenharia de modelos e sistemas auto-adaptativos serão estudadas e combinadas. Um sistema de teste auto-adaptativo (SATS) será desenvolvido, o qual será integrado a uma plataforma de testes previamente desenvolvida.*

1. Introdução

SOA é um padrão de desenvolvimento de serviços muito bem estabelecido e documentado. Serviços baseados em SOA seguem um padrão para interface de comunicação, e trabalhando sob um acordo que deixa claro o que e como uma coisa pode ser solicitada [Siau 2009]. Com o crescente número de serviços baseados em SOA, cria-se um ambiente próspero para que aplicações mais complexas e distribuídas surjam, assim como cresce a necessidade de se assegurar a qualidade dessas aplicações como um todo.

Esse trabalho apresenta na segunda sessão uma descrição do que é SOA e como são os testes em aplicações baseadas no paradigma. Na terceira sessão, são mencionadas ferramentas e trabalhos que abordam os problemas na área de teste online em SOA. Na quarta sessão, é exibida uma proposta de sistema de execução de testes online em SOA,

utilizando modelos e sistemas auto-adaptativos. Na quinta e sexta sessões, são apresentados os resultados e conclusões obtidos até agora com o desenvolvimento do trabalho, respectivamente.

2. Arquitetura orientada a serviços (SOA)

Arquitetura orientada a serviços (SOA) é um paradigma para organizar a utilização de capacidades distribuídas entre diferentes domínios e donos [Oasis 2005]. É um padrão de construção de serviços web, cujo objetivo é promover inter-operatividade, escalabilidade e reutilização de soluções de software. Aplicações SOA podem ser classificadas em dois tipos: orquestração e coreografia. Orquestração segue um paradigma de mestre-escravo, enquanto coreografias são compostas de serviços independentes que operam sobre um mesmo protocolo [Wang 2013].

SOA é comumente usada como uma solução empresarial, e vista como uma maneira de reduzir custos de desenvolvimento e manutenção [Tselentis et al. 2009]. A grande maioria das aplicações SOA são orquestradas, e são compostas progressivamente por aplicações terceiras disponíveis na internet. Cada um desses serviços pode ser descrito, publicado, categorizado, descoberto e dinamicamente montado, ser ligado e desligado a qualquer momento quando necessário. Uma das consequências dessas características é o desempenho baixo das técnicas clássicas de teste para avaliar uma aplicação SOA enquanto online. [Kalamegam and Godandapani 2012].

3. Testes em SOA

Testes em SOA se diversificam em diversos níveis e podem cobrir tanto aspectos funcionais e não-funcionais de serviços, quanto individualmente e integrados a uma composição [Kalamegam and Godandapani 2012]. Teste de unidade, integração, sistema e regressão são necessários para garantir que aplicações SOA se comportem de acordo com o planejado. Entretanto, devido à dinamicidade e à imprevisibilidade de aplicações SOA, muitas técnicas de teste se tornam inadequadas ou inaplicáveis diretamente [Ribarov et al. 2007].

A execução de testes em aplicações SOA pode ser offline e online. Testes offline executados antes do lançamento de uma aplicação não asseguram que ela vá ter o mesmo comportamento durante sua execução [Ceccarelli et al. 2011]. Entretanto, testes online são geralmente evitados por serem muito intrusivos.

3.1. Desafios de teste SOA

Estratégias de teste para aplicações SOA devem envolver componentes de infraestrutura, serviços individuais e compostos e processos de negócio ponta-a-ponta, abrangendo múltiplos serviços e aplicações. Os testes feitos nos web-services podem visar a interface de comunicação, a implementação, a comunicação e os requisitos da camada de acordo de serviço (SLA) [Kalamegam and Godandapani 2012].

Planejar testes para aplicações SOA envolve superar uma série de desafios. Aplicações SOA orquestradas possuem diferentes abordagens de teste comparando com coreografias, por exemplo. Nesse trabalho, vamos estudar uma aplicação centralizada (orquestrada), os desafios mais comuns associados a testes nesse tipo de aplicação são: necessidade de conhecimentos técnicos sólidos pelos testadores; falta de visibilidade de uma estratégia no geral; Mudanças rápidas e dinamicidade da aplicação; indisponibilidade de

códigos fontes e serviços estruturais; falta de informação sobre componentes integrados; evolução imprevisível de serviços terceiros; dependência de provedores de serviço; ambientes de teste, que não conseguem simular fielmente o ambiente funcionamento final; os testes devem ser executados mesmo com a aplicação online.

4. Trabalhos relacionados

4.1. Ferramentas

Existem várias ferramentas para geração de casos de teste a partir de modelos, todas elas focando em uma característica específica dos sistemas sob teste (SUTs), em diferentes modelos de entrada, em diferentes algorítimos de geração [Utting et al. 2012], algumas das mais relevantes são: (i) GRAPHWALKER, é uma ferramenta de testes baseados em modelos escrita em JAVA. Funciona lendo modelos parecidos com diagramas de estados finito. Os testes gerados podem ser executados tanto online quanto offline [Olsson 2014]; (ii) MODEL J-UNIT, é uma biblioteca livre que estende o funcionamento do JUnit para testes baseados em modelos. Modelos interagem diretamente com o sistema sob teste em JAVA ou com um adaptador. Testes gerados podem ter diferentes métricas [Utting 2015]; (iii) TOP CASED, é uma ferramenta de desenvolvimento baseado em modelos e ambiente de teste para sistemas críticos de hardware e software. É uma ferramente livre para desenvolvimento de sistemas embarcados, atualmente conhecido como PolarSys [Polarsys 2015].

4.2. Pesquisas

Alguns trabalhos foram escolhidos para servirem de referência para esse projeto:

Dranidis e colaboradores em seu trabalho apresentam uma técnica automatizada que determina quando adaptações são necessárias em aplicação SOA, evitando assim compensações desnecessárias [Dranidis et al. 2010].

Cao e colaboradores apontam que testes de conformidade são comumente usados para aumentar a confiança no sistema, e isso consome muito tempo e requer muita experiencia sobre o sistema alvo. Usando modelos os autores propõem um algorítimo responsável pela criação dos casos de teste, execução e avaliação dos resultados em aplicações SOA [Cao et al. 2010].

Ceccarelli e colaboradores propõem um sistema de execução de testes de robustez para aplicações SOA, monitorando se serviços passam ou não por atualizações e executando testes nas interfaces do serviço original ou um clone, caso seja necessário [Ceccarelli et al. 2011].

Dranidis et al. e Cao et al. utilizam modelos estáticos em seus trabalhos. O projeto apresentado nesse trabalho usa modelos dinâmicos para geração de casos de testes e sistema de teste similar ao apresentado por Ceccarelli et al.

5. Proposta

5.1. Objetivo

Como existe uma quantidade muito grande de desafios atrelados à execução de testes online em aplicações SOA, algumas suposições foram feitas para restringir a complexidade

do problema e facilitar o desenvolvimento de uma possível solução, sendo elas: execução de testes ponta-a-ponta em SOA, os quais terão como objetivo verificar se o processo de negócio está de acordo com os requisitos; a geração de casos de testes deve ser dinâmica e acompanhar a evolução da aplicação SOA; os testes devem ser executados mesmo com a aplicação online. Sendo assim, o objetivo final desse projeto é a geração automática e execução de testes online a partir de modelos em uma aplicação SOA.

5.2. Metodologia

O serviço de teste será desenvolvido usando como base o sistema de teste disponibilizado por Ceccarelli et al. [Ceccarelli et al. 2011]. O sistema de teste base possui ferramentas de detecção de mudança nos serviços que estão atreladas à aplicação, assim como possui uma maneira de clonar os serviços que estão online para que os testes sejam executados sem interferir na aplicação real. Para contornar os problemas relacionados à dinamicidade da aplicação e à geração de casos de testes que sejam coerentes com o novo arranjo, um levantamento das técnicas disponíveis nos campos de engenharia de software e de modelos será executado. Isto será feito pois ambas as áreas têm soluções interessantes para os problemas abordados nesse projeto.

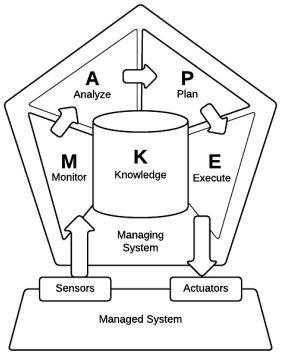


Figura 1. Ciclo auto-adaptativo MAPE-K

Adaptação em ambientes inconstantes é um problema documentado e estudado na área de engenharia de software, sendo a adaptação autônoma uma característica desejada pra muitos sistemas de software. Sistemas auto-adaptativos (SAS) são estudados por diferentes áreas da engenharia de software, um exemplo é a engenharia de componentes [Brun et al. 2009].

Model based testing (MBT) é um campo da engenharia baseado em modelos e é uma variante de testes caixa preta, cujo objetivo é a geração de testes a partir de modelos comportamentais do sistema e de sua execução [Apfelbaum and Doyle 1997]. MBT apresenta algumas maneiras para gerar casos de testes para a aplicação de forma automatizada.

6. Resultados

Para a geração dos casos de teste, foi selecionada uma ferramenta chamada GraphWalker, a qual é baseada em modelos e desenvolvida em Java. Esta ferramenta funciona lendo modelos como diagramas de estados finitos. Os testes gerados a partir do modelo podem rodar tanto offline quanto online [Olsson 2014]. Os modelos usados pela ferramenta são facilmente descritos usando um arquivo XML e possuem uma ferramenta para serem gerados manualmente se necessário.

Para que o modelo gere os resultados desejados, é necessário que ele seja atualizado concomitantemente com o sistema. O SATS será desenvolvido baseado em um ciclo auto-adaptativo centralizado, baseado na solução da IBM para sistemas autoadaptativos, conhecido como MAPE-K [Arcaini et al. 2015] conforme ilustrado na figura 1.

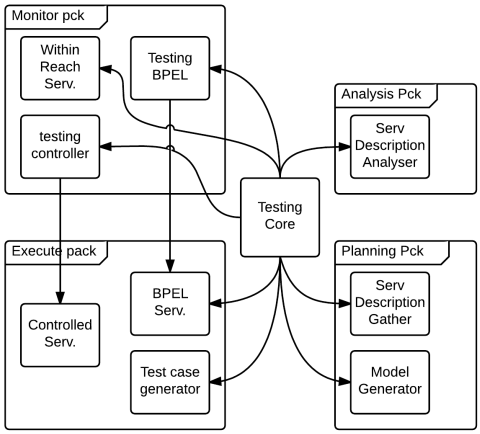


Figura 2. Pacotes e componentes do SATS

O sistema autoadaptativo de teste está planejado para ter os componentes exibidos na figura 2. Os componentes do pacote de monitoração e dois dos componentes (Controlled Services e BPEL Service) no pacote de execução serão herdados da plataforma de teste desenvolvida pela equipe de Ceccarelli.

O núcleo de testes será responsável por coordenar o ciclo MAPE-K ilustrado na figura 3. O ciclo é dividido em dois processos, conforme apresentado na 3. O processo principal é iniciado quando uma atualização é detectada por algum dos componentes do pacote de monitoração. Em seguida, o núcleo de teste chama o componente Service Description Analyser, responsável por decidir se é necessário ou não executar uma atualização no modelo e, consequentemente, a execução de testes. Caso o resultado da análise seja positivo, o núcleo de teste dá início ao segundo processo, que consiste basicamente em atualizar as informações necessárias para atualizar o modelo, gerar um novo modelo do sistema, criar clones dos serviços caso necessário, executar os testes e criar um relatório.

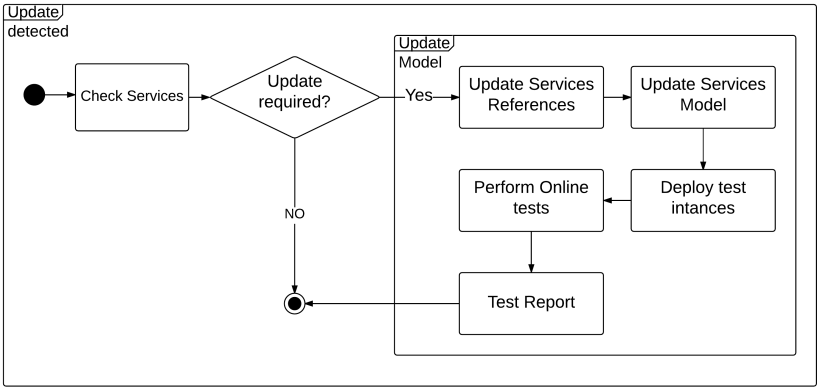


Figura 3. Diagrama de sequencia do ciclo MAPE-K do SATS

6.1. Conclusão

O serviço de teste ainda está incompleto, os componentes dos pacotes de análise e planejamento ainda estão em desenvolvimento. O componente responsável por coordenar todo o ciclo MAPE-K já tem sua lógica interna descrita, e os componentes dos pacotes de análise e planejamento já estão em desenvolvimento. O desafio atual é encontrar uma maneira de medir o desempenho da abordagem quando ela estiver concluída. Existem ferramentas em desenvolvimento por outros grupos de pesquisa, entretanto essas ferra-

mentas não estão disponíveis para serem usadas, o que dificulta qualquer comparação entre os resultados obtidos.

Referências

Apfelbaum, L. and Doyle, J. (1997). Model based testing. In *Software Quality Week Conference*, pages 296–300.

Arcaini, P., Riccobene, E., and Scandurra, P. (2015). Modeling and analyzing mape-k feedback loops for self-adaptation. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2015 IEEE/ACM 10th International Symposium on*, pages 13–23. IEEE.

Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., and Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*, pages 48–70. Springer.

Cao, T.-D., Felix, P., Castanet, R., and Berrada, I. (2010). Online testing framework for web services. In *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*, pages 363–372. IEEE.

Ceccarelli, A., Vieira, M., and Bondavalli, A. (2011). A testing service for lifelong validation of dynamic soa. In *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on*, pages 1–8. IEEE.

Dranidis, D., Metzger, A., and Kourtesis, D. (2010). Enabling proactive adaptation through just-in-time testing of conversational services. In *Towards a Service-Based Internet*, pages 63–75. Springer.

Kalamegam, P. and Godandapani, Z. (2012). A survey on testing soa built using web services. *International Journal of Software Engineering and Its Applications*, 6(4).

Oasis, S. (2005). Reference model tc. *OASIS Reference Model for Service Oriented Architecture*, 1.

Olsson, N. (2014). Graphwalker - model-based testing @ONLINE.

Polarsys (2015). Eclipse titan@ONLINE.

Ribarov, L., Manova, I., and Ilieva, S. (2007). Testing in a service-oriented world.

Siau, K. (2009). *Principle Advancements in Database Management Technologies: New Applications and Frameworks: New Applications and Frameworks*. IGI Global.

Tselentis, G., Domingue, J., and Galis, A. (2009). *Towards the Future Internet: A European Research Perspective*. IOS press.

Utting, M. (2015). Modeljunit test generation tool@ONLINE.

Utting, M., Pretschner, A., and Legeard, B. (2012). A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability*, 22(5):297–312.

Wang, Y. (2013). A survey on formal methods for web service composition. *arXiv preprint arXiv:1306.5535*.

Um estudo experimental sobre testes funcionais e de robustez de acordo com a análise de mutantes

Wallace Felipe Francisco Cardoso [1] **e Eliane Martins** [1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6.176 – 13.081-970 – Campinas – SP – Brasil

{wallace.cardoso, eliane}@ic.unicamp.br

Abstract. *Testing whether a system satisfies the specified behavior is not enough, it's important also testing if the system has mechanisms which dealing with invalid or innoportune inputs. Our work considers functional and robustness test cases generated from behavioral models, and assesses them, according to its potential of fault detection, through of mutation analysis technique. Our results shows that the robustness test cases aid to improve the quality of test activity as they has a higher probability to detect some kinds of faults than the functional test cases.*

Resumo. *Testar se um sistema satisfaz apenas o comportamento especificado não é suficiente, é importante também testar se o sistema possui mecanismos que tratam de entradas inválidas ou inoportunas. Nossa proposta considera casos de teste funcionais e de robustez gerados a partir de modelos comportamentais, e os avaliam, quanto ao potencial de detecção de defeitos, através da técnica análise de mutantes. Os resultados mostram que os casos de teste de robustez ajudam a melhorar a qualidade da atividade de teste dado que têm uma probabilidade maior de detectar determinados tipos de defeitos em relação aos casos de teste funcionais.*

1. Introdução

O teste é o método principal para revelar defeitos de um determinado produto de software [Papadakis and Malevris 2012]. Um sistema com muitos defeitos é pouco confiável e, em determinados casos, oferece riscos à vida, às finanças e ao meio ambiente.

O processo clássico do teste consiste em executar um sistema de acordo com uma determinada entrada e observar se suas saídas correspondem ao esperado. Uma abordagem de teste é testar o sistema de acordo com entradas válidas que verificam se o sistema se comporta como especificado. Outra abordagem consiste em testar como o sistema reage quando entradas inválidas ou inesperadas são introduzidas, ou ainda, executar o sistema sob condições ambientais estressantes. São conhecidas como testes funcionais e testes de robustez, respectivamente.

Visando verificar se o sistema contém mecanismos para tratar entradas inválidas ou inoportunas, e com isso evitar consequências catastróficas, os testes de robustez consistem em executar um sistema em presença destas entradas anômalas [IEEE Std 24765:2010]. Uma estratégia é introduzir parâmetros inválidos na interface do sistema com o seu ambiente [Micskei et al. 2012]. Entretanto, por não levar em conta o estado do sistema, esta estratégia apresenta uma baixa controlabilidade dos testes.

Nossa proposta considera utilizar modelos de software para gerar testes funcionais e de robustez. Através dos modelos é possível gerar casos de teste antes mesmo da implementação do sistema existir, o que implica em uma redução significativa dos custos de projeto [Prenninger and Pretschner 2005]. Teste baseado em modelo (TBM) é muito utilizado para testes funcionais e, utilizando-se do mesmo arcabouço, existem algumas propostas para testes de robustez.

Neste trabalho pretendemos responder a seguinte pergunta, levando em consideração o teste baseado em modelo: Qual conjunto apresenta um potencial de detecção de defeitos melhor, o conjunto dos casos de teste funcionais ou dos de robustez?

O restante deste trabalho está organizado da seguinte forma: A Seção 2 apresenta conceitos básicos, o estudo experimental é apresentado na Seção 3, os trabalhos relacionados estão na Seção 4, e as conclusões estão na Seção 5.

2. Fundamentação Teórica

Nesta seção serão apresentados alguns conceitos básicos para o entendimento do restante deste trabalho.

2.1. Teste Baseado em Modelo

Teste baseado em modelo é uma abordagem onde o modelo é utilizado para extrair requisitos de teste visando melhorar a qualidade e também reduzir custos da atividade de teste [Prenninger and Pretschner 2005].

Diagramas de estados, por exemplo, são comumente utilizados pela indústria de software para o desenvolvimento de código. O modelo pode ser tanto para a geração de código quanto de testes [Martins et al. 1999, Shirole et al. 2011]. Os testes gerados a partir de um modelo abstrato do sistema são conhecidos como testes abstratos e, para serem executados, devem ser concretizados de acordo com as tecnologias e linguagens escolhidas de desenvolvimento do sistema.

2.2. Análise de Mutantes

Análise de mutantes é uma técnica que consiste em criar versões de um determinado artefato e introduzir defeitos artificiais em cada versão, visando avaliar se um determinado conjunto de testes é capaz de revelar a presença destes defeitos introduzidos artificialmente [Woodward 1993].

Cada versão é conhecida como mutante e, na análise de mutantes tradicional, contém apenas uma alteração sintática. Os mutantes são criados através de operadores de mutação, os quais contêm regras de onde (ponto de mutação) e como a mutação deverá ocorrer.

Um mutante é dito morto quando pelo menos um caso de teste é capaz de produzir saídas diferentes no mutante e no original. Caso contrário, ele é dito vivo. Se não existir um caso de teste que consiga produzir saídas diferentes entre o mutante e o original, o mutante é dito equivalente [Papadakis and Malevris 2012]. Mutantes equivalentes são descartados, pois não denotam versões defeituosas, mas versões alternativas. Contudo, o problema de detectar se um mutante é equivalente ou não é indecidível, realizado manualmente ou através de heurísticas.

O cálculo do escore de mutação [Woodward 1993], a métrica que permite determinar o potencial de detecção de defeitos de um conjunto de teste, é dada pela Equação 1: onde TS é um conjunto de casos de teste, M é um conjunto de mutantes, DM é uma função que retorna a quantidade de mutantes mortos por TS, e EM é uma função que retorna a quantidade de mutantes equivalentes em M. O escore de mutação varia entre 0 e 1, onde 1 significa 100% adequado à mutação.

$$MS(TS, M) = \frac{DM(TS, M)}{|M| - EM(M)} \tag{1}$$

3. Estudo Experimental

O estudo experimental realizado busca responder a pergunta: Qual conjunto apresenta um potencial de detecção de defeitos melhor, o conjunto dos casos de teste funcionais ou dos de robustez?

Para descobrir o potencial de detecção de defeitos de cada conjunto de teste foi utilizada a técnica análise de mutantes (Seção 2.2) implementada na ferramenta State-Mutest [Cardoso 2015]. Foram considerados 11 modelos, 11 conjuntos de casos de teste funcionais e 11 de robustez. Os modelos são parte de um *benchmark* de teste, enquanto que os casos de teste foram gerados em trabalho prévio do grupo [Yano et al. 2011b].

Todos os modelos e os respectivos casos de teste foram importados para a State-Mutest, a ferramenta então cria um relatório da análise de mutantes para cada conjunto avaliado, automaticamente. Os modelos e os resultados da análise de mutantes são apresentados na Tabela 1.

Os modelos utilizados são modelos comportamentais, um subconjunto do diagrama de estados da UML. Os casos de teste funcionais e os de robustez exercitam o mesmo caminho de transições nos modelos, entretanto os casos de teste de robustez contém sequências de entrada adicionais (entradas anômalas). Quando o modelo não deixa explícito o comportamento do sistema em resposta a um estímulo anômalo, baseado na semântica da UML, seu estado permanece inalterado ignorando o estímulo recebido [Rumbaugh et al. 2004, p. 81].

Os casos de teste de robustez (SEI) contêm tanto as entradas nominais quanto as anômalas, enquanto que os casos de teste funcionais (SEN), neste estudo, são um subconjunto dos casos de teste de robustez contendo apenas as nominais.

Os resultados foram analisados estatisticamente utilizando o software *R* [R Core Team 2016]. Os testes estatísticos realizados consideram a significância estatística $\alpha = 0,05$. Segundo o teste de normalidade *Shapiro-Wilk test*, as amostras comparadas se aproximam a uma distribuição normal ($p > \alpha$, não rejeita a hipótese nula H_0 = "a distribuição se aproxima à normal"). Dado que as amostras são pareadas e se aproximam à normal, utilizou-se o teste para médias *paired test-t*, para verificar se, em média, existe diferença significativa entre o escore médio dos conjuntos (hipótese alternativa H_a). De acordo com o teste, em média, o conjunto SEI tem o melhor potencial de detecção de defeitos em relação ao conjunto SEN ($p < \alpha$, aceita a hipótese alternativa H_a = "existe diferença significativa entre as médias").

Ainda que as entradas anômalas não causem alterações no estado do sistema, de

Tabela 1. Resultados da análise de mutantes para os modelos do estudo.

	Escore de Mutação			
Modelo	**#SEN (#MV)**	**#SEI (#MV)**	**Casos de Teste**	**Mutantes (#EQ)**
ATM	0,9626 (13)	1 (0)	70	348 (2)
Cashier	0,7197 (105)	0,8071 (74)	153	389 (3)
CruiseControl	0,8269 (9)	0,8615 (0)	94	260 (5)
FuelPump	0,9030 (17)	0,9229 (0)	71	588 (5)
Inres	0,9314 (12)	0,9838 (0)	261	248 (13)
Lift	0,9338 (10)	0,9779 (0)	52	136 (1)
VendingMachine	0,9191 (26)	0,9570 (0)	454	396 (5)
InFlight	0,7449 (136)	0,8708 (59)	326	938 (45)
Class2Protocol	0,7379 (70)	0,7964 (48)	322	393 (27)
ATM2	0,8954 (14)	0,9291 (0)	202	593 (7)
Inres2	0,8468 (9)	0,8899 (0)	188	209 (10)
Média	**0,8556**	**0,9100**		

#MV Mutações não exercitadas
#SEN Casos de teste funcionais
#SEI Casos de teste de robustez
#EQ Mutantes equivalentes

acordo com a Tabela 1 considerando as mutações não exercitadas, elas contribuem para um melhor alcance dos pontos de mutação do que as entradas de teste funcionais. Quando uma mutação não é exercitada, na maioria dos casos, o mutante não é morto, pois os resultados do mutante e do original podem divergir apenas após o ponto de mutação.

4. Trabalhos Relacionados

Foram propostas algumas estratégias e abordagens no contexto de teste baseado em modelo e testes de robustez, as quais diferem na forma em que as entradas inválidas são introduzidas no modelo.

Uma estratégia consiste em completar o modelo com entradas inválidas e suas respectivas saídas esperadas [Saad-Khorchef et al. 2007]. Consequentemente, o modelo pode se tornar muito grande levando a uma explosão combinatória de casos de teste.

Com o objetivo de evitar a explosão de estados, a estratégia CoFI (*Conformance and Fault Injection testing*) propõe o uso de diferentes modelos para representar o comportamento do sistema diante de entradas válidas e inválidas [Ambrosio et al. 2007]. Entretanto, a quantidade de modelos pode crescer proporcional à complexidade do sistema. Existem propostas que aplicam mutação na sequência de teste criando sequências com entradas anômalas [Rollet and Salva 2009], mas é possível que alguns testes tornem-se infactíveis.

SABRINE (*StAte-Based Robustness testIng of operatiNg systEms*) [Cotroneo et al. 2013] é uma proposta que extrai um diagrama de estados através

da execução do sistema, e a partir do modelo gera os casos de teste de robustez. A limitação desta proposta é que o modelo reflete o que foi implementado, assim falhas de omissão de comportamento, por exemplo, não são reveladas.

Em trabalho prévio do grupo, foi proposto um método de geração de casos de teste de robustez a partir de diagramas de estados [Yano et al. 2011b, Yano et al. 2011a]. O método proposto considera entradas de teste funcionais e anômalas para gerar conjuntos de teste de robustez, não havendo necessidade de completar o modelo.

Outro trabalho prévio do grupo realizou uma avaliação do potencial de detecção de defeitos dos casos de teste funcionais e dos de robustez em relação à efetividade e ao custo [Cardoso and Martins 2016]. Os resultados mostram que os casos de teste de robustez têm um custo-benefício melhor do que os casos de teste funcionais.

5. Conclusões

Os testes de robustez são uma das formas de garantir que um sistema possui mecanismos que tratam de entradas anômalas. Portanto, é possível determinar se um sistema comporta-se adequadamente tanto nas condições previstas quanto para as anomalias que podem ocorrer.

Realizamos um estudo experimental visando descobrir se existe diferença significativa entre o potencial de detecção de defeitos de casos de teste funcionais em relação aos de robustez. Os resultados mostram que os casos de teste de robustez são melhores em revelar defeitos do que os casos de teste funcionais. Testadores podem preferir casos de teste menores acreditando reduzir o tempo gasto com a execução de um conjunto grande de casos de teste, entretanto é importante que um conjunto de casos de teste contenha sequências anômalas dado que a presença delas melhora o potencial de detecção de defeitos dos casos de teste.

Não podemos generalizar nossos resultados dado que os modelos foram retirados de um *benchmark* de teste, mesmo que presentes em outros trabalhos, e não há nenhum modelo de um sistema real, sendo baixa também a quantidade de modelos utilizada em nosso estudo. Pretendemos incluir modelos de sistemas reais em um estudo futuro, e considerar uma quantidade significativa de modelos para um experimento visando melhorar a representatividade de nossos resultados.

Referências

[Ambrosio et al. 2007] Ambrosio, A. M., Mattiello-Francisco, F., Santiago Jr, V. A., Silva, W. P., and Martins, E. (2007). Designing fault injection experiments using state-based model to test a space software. In *Dependable computing*, pages 170–178. Springer.

[Cardoso 2015] Cardoso, W. F. F. (2015). Statemutest: uma ferramenta de apoio ao teste baseado em modelos de estado estendidos. Master's thesis, Universidade Estadual de Campinas.

[Cardoso and Martins 2016] Cardoso, W. F. F. and Martins, E. (2016). Uma avaliação do potencial de detecção de defeitos dos casos de teste de robustez de acordo com a análise de mutantes. In *XVII Workshop de Testes e Tolerância a Falhas (WTF), XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2016*. Sociedade Brasileira de Computação (SBC).

[Cotroneo et al. 2013] Cotroneo, D., Di Leo, D., Fucci, F., and Natella, R. (2013). Sabrine: State-based robustness testing of operating systems. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 125–135. IEEE.

[Martins et al. 1999] Martins, E., Sabião, S. B., and Ambrosio, A. M. (1999). Condata: a tool for automating specification-based test case generation for communication systems. *Software Quality Journal*, 8(4):303–320.

[Micskei et al. 2012] Micskei, Z., Madeira, H., Avritzer, A., Majzik, I., Vieira, M., and Antunes, N. (2012). Robustness testing techniques and tools. In *Resilience Assessment and Evaluation of Computing Systems*, pages 323–339. Springer.

[Papadakis and Malevris 2012] Papadakis, M. and Malevris, N. (2012). Mutation based test case generation via a path selection strategy. *Information and Software Technology*, 54(9):915–932.

[Prenninger and Pretschner 2005] Prenninger, W. and Pretschner, A. (2005). Abstractions for model-based testing. *Electronic Notes in Theoretical Computer Science*, 116:59–71.

[R Core Team 2016] R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Rollet and Salva 2009] Rollet, A. and Salva, S. (2009). Testing robustness of communicating systems using ioco-based approach. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 67–72. IEEE.

[Rumbaugh et al. 2004] Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual, The*. Pearson Higher Education.

[Saad-Khorchef et al. 2007] Saad-Khorchef, F., Rollet, A., and Castanet, R. (2007). A framework and a tool for robustness testing of communicating software. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 1461–1466. ACM.

[Shirole et al. 2011] Shirole, M., Suthar, A., and Kumar, R. (2011). Generation of improved test cases from uml state diagram using genetic algorithm. In *Proceedings of the 4th India Software Engineering Conference*, pages 125–134. ACM.

[Woodward 1993] Woodward, M. R. (1993). Mutation testing—its origin and evolution. *Information and Software Technology*, 35(3):163–169.

[Yano et al. 2011a] Yano, T., Martins, E., and De Sousa, F. L. (2011a). A model-based approach for robustness test generation. In *Dependable Computing Workshops (LADCW), 2011 Fifth Latin-American Symposium on*, pages 33–34. IEEE.

[Yano et al. 2011b] Yano, T., Martins, E., and De Sousa, F. L. (2011b). Most: a multi-objective search-based testing from efsm. In *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on*, pages 164–173. IEEE.

Um sistema para contagem de pessoas em passagens

Leonardo Alves de Melo[1], **Edson Borin**[2]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6176 – 13.081-970 – Campinas – SP –Brasil

leonardo.alves.melo.1995@gmail.com, edson@ic.unicamp.br

Abstract. *This paper was based on a research focusing on development of low-cost people detection and counting techniques in passages. In order to this, was investigated the Ultrasonic Sensor HC-SR04 and it was discovered that its detection border has an average angle of 15° relative to its center. Based on that, was proposed an architecture of a prototype containing HC-SR04 sensors and its data was captured by an embedded software on a microcontroller, which determines the passage direction. Initial tests showed that the software on this architecture had 78% of accuracy.*

Resumo. *Este trabalho foi baseado na busca de métodos de baixo custo e consumo para contagem de pessoas em passagens. Para tanto, foi investigado o sensor ultrassônico HC-SR04 e descoberto que sua borda de detecção apresenta uma angulação média de 15º em relação ao seu centro. A partir disso, foi proposta uma montagem de um protótipo contendo sensores HC-SR04 e seus dados captados por um* software *embarcado em um microcontrolador, o qual determina a direção em que a pessoa passa. Testes iniciais revelaram que o* software *nessa montagem identificou corretamente 78% das passagens.*

1. Introdução

Uma das aplicações esperadas para a Internet das Coisas é a gestão inteligente de prédios, incluindo residências, escritórios e industrias. Neste contexto, a instalação de "passagens inteligentes", as quais identificam a direção e o número de pessoas que passaram pelo local, permitirá um monitoramento mais preciso e otimização do uso dos recursos do prédio. Como exemplo, este mecanismo poderia ser utilizado para o desenvolvimento de um banheiro inteligente, que monitoraria seu uso e alertaria a equipe de limpeza a partir de um determinado número de pessoas que o usaram. Outra aplicação seria a contagem de pessoas para determinar de forma precisa quantas pessoas estão presentes em ambientes de grandes eventos e evitar um possível problema de superlotação.

Um equipamento de contagem de pessoas que tenha custo e consumo baixos permite sua utilização mesmo em locais que não tenham acesso a tomadas, visto que se pode usar uma bateria para alimentação por vários dias. Por ser barato, mais pessoas poderão adiquiri-lo e implementá-lo em seus estabelecimentos, levando a Internet das Coisas para mais locais.

2. Trabalhos relacionados

Existem atualmente diversos métodos de detecção e contagem de pessoas em passagens, como no caso da utilização de uma câmera 3D fixada acima de um portão [Terada 1999],

que apresenta uma boa acurácia mesmo quando há um grande fluxo de pessoas. O problema da utilização de uma câmera é o alto custo, além de necessitar de uma infraestrutura no local, como a presença de uma tomada. Existem métodos voltados para locais com ausência de infraestrutura, como o aplicado nas áreas montanhosas da Coreia do Sul [Son 2007], em que foi utilizado um sensor fotoelétrico para a detecção e contagem de pessoas, no intuito de auxiliar na preservação do meio ambiente do local.

3. Objetivos

Este projeto tem como objetivo o desenvolvimento e aprimoramento de métodos de detecção de passagens de pessoas por meio de vãos, como portas, corredores, portões, entre outros, de forma que o produto final tenha baixo custo e consumo de energia. Para tanto, utilizaremos sensores os quais apresentam essas mesmas propriedades, e tudo será gerenciado por um *software* embarcado para análise dos dados dos sensores.

4. Metodologia

Será investigado o sensor de distância ultrassônico *HC-SR04* para entender seu funcionamento e, com base nisso, propor algoritmos para a contagem de pessoas. Também serão analisados os tipos de passagens e de que forma um protótipo poderia contar pessoas no determinado local.

5. Experimentos

Foram realizados dois experimentos para entender o funcionamento do sensor ultrassônico.

5.1. Dados brutos de uma passagem

O primeiro experimento foi feito colocando dois sensores ultrassônicos lado a lado distando 20cm no batente de uma porta, e controlados por um microcontrolador, conforme consta na figura 1.

A partir dessa montagem, ligamos o microcontrolador via USB para o computador, e coletamos ininterruptamente as distâncias captadas por cada sensor. A distância máxima de cada sensor foi inicializada com 75cm, e qualquer distância maior que essa, como a da largura da porta, é considerada como zero. A partir disso, realizamos diversas passagens planejadas por essa porta e salvamos os valores de distância captados pelos sensores ultrassônicos em um arquivo de texto, para posteriormente gerar gráficos de distâncias captadas versus tempo, como podemos ver na figura 2.

Podemos perceber a partir do gráfico que o sensor em que a pessoa primeiro passou é o primeiro sensor a detectar uma distância diferente de zero. Percebemos também que a detecção do sensor pode apresentar falhas, saltando de um valor referente à distância da pessoa até zero, representando a perda da pessoa por parte do sensor por alguns milésimos de segundo.

5.2. Ângulos

O segundo experimento foi o de determinação do ângulo máximo em que o sensor ultrassônico consegue detectar um objeto em relação ao seu centro. O experimento consistiu em colocar um objeto de dois modos diferentes a uma dada distância e ângulo em

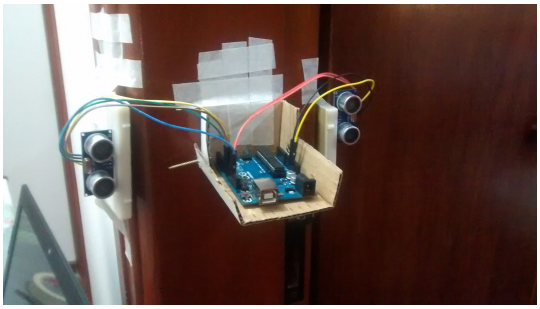


Figura 1. Foto da montagem do experimento.

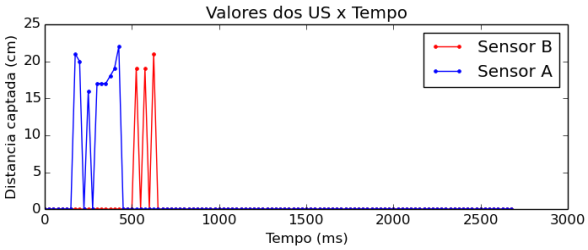


Figura 2. Gráfico gerado pelos dados do experimento.

relação ao eixo do sensor, e foi testado se era possível detectá-lo. Os dois modos que se foi colocado o objeto são: paralelo ao eixo horizontal do sensor e perpendicular a direção de propagação da onda emitida pelo sensor. Ambos os modos podem ser vistos no gráfico da figura 3.

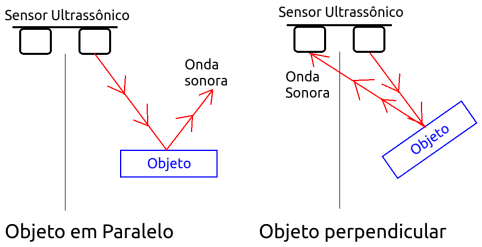


Figura 3. Esquema dos modos de colocação do objeto.

A partir dos dados, pudemos gerar o gráfico da figura 4 e, com ele, estimamos um ângulo intermediário entre as deteções paralela e perpendicular, e com isso chegamos ao valor de 15º para ser considerado o ângulo máximo de abertura em que o sensor consegue detectar um objeto.

6. Protótipo

Com base nos experimento feitos, foi desenvolvido um protótipo de um módulo contador.

6.1. Montagem

Foi determinado que o protótipo deveria conter exatamente dois sensores ultrassônicos, em que a direção da passagem seria determinada dependendo de qual sensor detectou

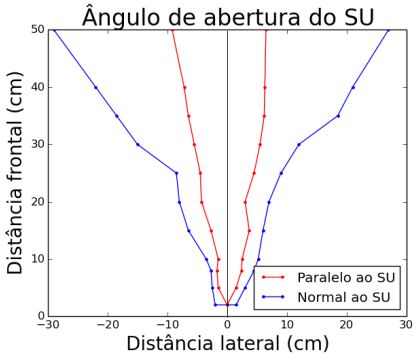


Figura 4. Gráfico da distância máxima lateral detectada pelo sensor a partir da distância frontal.

primeiramente um objeto com base em um algoritmo que analisa as médias de distância detectada por cada sensor.

Sabendo que o ângulo de detecção máximo é de 15º, propomos que os dois sensores devem estar 30º um em relação ao outro. Assim, foi possível colocá-los lado a lado, criando um protótipo pequeno e que tenha o mínimo de região de intersecção de detecção dos dois sensores, permitindo que um detecte a passagem antes do outro. O protótipo é constituído por uma caixa de papelão branco medindo 9,5cm x 11cm x 10,5cm com os componentes internos, e um espaço por onde os sensores podem fazer suas medidas, e pode ser na foto da figura 5.

Figura 5. Foto do protótipo.

6.2. Eletrônica

Por dentro do protótipo, além dos dois sensores ultrassônicos, temos um microcontrolador, um módulo *bluetooth JY-MCU* para a transferência de dados via *bluetooth* para o computador, uma *protoboard* e uma bateria 9V.

6.3. Algoritmo

Tendo como base o experimento dos dados brutos, concluímos que o algoritmo de detecção de pessoas deve ter algum recurso para tolerar possíveis erros do sensor, mas sem perder informações importantes que poderiam alertar o início ou o termino de uma

passagem. Optamos por utilizar um vetor de trinta e duas medidas para cada sensor, em que tiraríamos uma média para ser levada em conta na detecção de uma passagem. Essa média é atualizada a cada nova medida calculada.

Determinamos que toda medida na qual não foi detectado um objeto tem um valor máximo predeterminado. Dessa forma, quando o sensor detectar a aproximação de uma pessoa, a média irá reduzir seu valor, e a partir do ponto em que ela ultrapassar um limite mínimo, o programa considera que está ocorrendo uma passagem, e o sentido da passagem é definido do primeiro sensor que detectou para o outro sensor que ainda não detectou.

Após a detecção do primeiro sensor, o algoritmo prevê que o outro sensor detectará a pessoa, e a partir de então, dois eventos podem ocorrer: se o outro sensor detectar a pessoa, o algoritmo determina que a passagem está correta e espera a pessoa sair da zona de detecção dos sensores para contar a passagem. Se demorar mais que um limite arbitrário de tempo e o outro sensor não detectar a pessoa, o algoritmo determina que a passagem não foi correta, pois não é possível que a pessoa tenha passado apenas na frente de um dos sensores, então conta uma incerteza.

Um gráfico representativo dos valores das médias de cada sensor com a margem limite quando ocorre uma passagem bem sucedida é visto na figura 6. Note que qualquer valor de média maior que 70cm não é considerado uma detecção de passagem, e a partir do ponto que que a média se torna menor que 70cm (aproximadamente 500ms), o programa considera que está ocorrendo uma passagem, e como o sensor A foi o primeiro a detectar, então o sentido da passagem é do sensor A para o sensor B. O programa passa a esperar que o sensor B seja detectado, o que ocorre a partir de 600ms, então temos uma passagem bem sucedida, agora o algoritmo espera a pessoa sair da região de detecção para contar, o que ocorre em 1500ms.

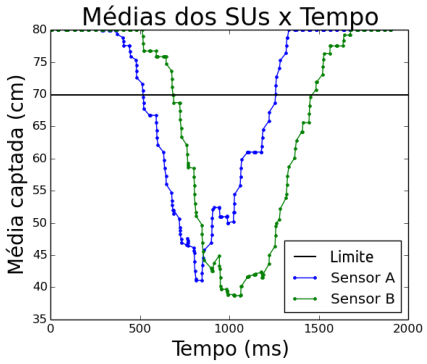


Figura 6. Gráfico das médias dos sensores em uma passagem.

7. Teste

Foi realizado um teste com o protótipo no dia 23 de maio de 2016 no corredor do IC 3 que dá acesso ao bebedouro. O experimento durou 45 minutos e ocorreram 100 passagens, dessas, o detector acertou em 78% dos casos. Os 22% de erro por parte do algoritmo estão divididos em: 4% de incertezas, que é quando uma única pessoa passou, mas apenas um dos sensores detectou e alertou incerteza, e 18% de passagens de mais de uma pessoa ao

mesmo tempo, seja lado a lado ou em sequência. Podemos ver no gráfico da figura 7 a resposta do algoritmo de contagem em comparação com a contagem feita pelo usuário. Percebemos que o programa tende a detectar menos pessoas que deveria, e esse erro aumenta com o passar do tempo.

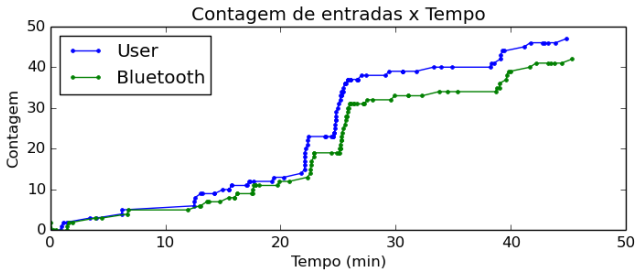


Figura 7. Gráfico comparativo do número de entradas registradas pelo usuário e enviadas via bluetooth pelo microcontrolador.

8. Criação de Massa de Dados

No intuito de melhorar a eficiência do algoritmo, realizamos um experimento no mesmo corredor da figura 10 no dia 6 de Junho de 2016 durante uma hora, mas dessa vez ao invés do microcontrolador enviar a contagem, ele envia os dados brutos das leituras dos sensores, que são guardados em um arquivo de texto. Foi gerado um arquivo desse tipo para cada passagem realizada, ou seja, como tivemos 18 entradas e 12 saídas, geramos 18 arquivos com os dados que os sensores detectariam em uma entrada, e 12 arquivos com os dados que os sensores detectariam em uma saída. Houve 5 casos em que mais de uma pessoa passou em um curto intervalo de tempo, que são exatamente os casos em que o algoritmo contaria menos pessoas, e eles foram devidamente registrados.

Essa massa de testes é usada para agilizar o processo de teste e otimização do algoritmo de contagem de pessoas, pois com ela podemos usar os casos de teste em um programa no computador ao invés de realizar experimentos reais com o microcontrolador, que podem ser demorados.

9. Conclusões e Trabalhos Futuros

A partir de dois sensores ultrassônicos de baixo custo, um microcontrolador e um algoritmo simples conseguimos contar a passagem de pessoas em um corredor com acerto em 78% dos casos. Como trabalho futuro, pretendemos aprimorar o algoritmo e/ou explorar novas posições para os sensores para melhorar a precisão do sistema. Para tanto, nos basearemos na massa de dados para otimizar o processo de teste do algoritmo.

Referências

Son, B. (2007). Development of an automatic people-counting and environment monitoring system based on wireless sensor network in real time. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 579 – 584. IEEE.

Terada, K. (1999). A counting method of the number of passing people using a stereo camera. In *Industrial Electronics Society, 1999. IECON '99 Proceedings. The 25th Annual Conference of the IEEE (Volume:3)*, pages 1318 – 1323 vol.3. IEEE.

3. Resumo dos pôsteres

"Para mudar o mundo, você precisa antes mudar a sua cabeça".

Jimi Hendrix

.

Aceleração de métodos de processamento sísmico com OpenCL

Hércules C. Silva[1], **Edson Borin** [1], **Jorge H Faccipieri**[2], **Tiago A. Coimbra**[2],

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
CEP 13083-852 – Campinas – SP – Brasil

[2]Centro de Estudos de Petróleo – Universidade Estadual de Campinas (UNICAMP)
CEP 13083-970 – Campinas – SP – Brasil

{hercules.cardos.silva1,tgo.coimbra}@gmail.com, edson@ic.unicamp.br

jorge@ggaunicamp.com

Resumo. *Este trabalho tem como objetivo a aceleração de dois métodos de imageamento sísmico: CMP (*Common midpoint*) e o CRS (*Common reflection surface*), para isso nos utilizaremos de aceleradores de* hardware *como o Xeon Phi e GPUs com o auxílio do OpenCL.*

1. Introdução

O conhecimento da subsuperfície é de fundamental importância para a exploração de hidrocarbonetos. Para isso, diversas técnicas foram desenvolvidas para mapear a subsuperfície e resgatar o máximo de informação possível do meio geológico através de medidas realizadas na superfície. Os métodos CMP (do inglês, *Common midpoint*) [Mayne 1962] e CRS (do inglês, *Common Reflection Surface*) [Hubral et al. 1998] foram desenvolvidos para esse propósito. Tais métodos realizam buscas de eventos (curvas no método CMP e superfícies no método CRS) que representam as respostas do meio geológico (camadas) ao sinal enviado por uma fonte (explosivo ou canhão de ar comprimido).
O processamento do dado sísmico adquirido na superfície, em ambos os métodos, se dá pelo cômputo de uma medida de coerência entre os sinais obtidos. Tal processamento requer um grande volume de operações computacionais, tornando-se necessário o uso de sistemas de computação de alto desempenho. Neste trabalho mostraremos as vantagens do uso de aceleradores de *hardware* como o Xeon Phi e GPUs e para isso utilizaremos o modelo de programação OpenCL.

2. Metodologia

Na tabela 1 temos as configurações que utilizamos em nossos experimentos, todos os experimentos utilizaram a *flag* de otimização -O3.
Comparamos a implementação em OpenCL usando os aceleradores de *hardware* que estão descritos na tabela 1 com a implementação OpenMP usando a CPU do *host* da plataforma 1 mostrada na tabela 1.

		Plataforma 1	Plataforma 2	Plataforma 3	Plataforma 4
Host	Processador	2 x Intel Xeon E5-2670	1 x Intel Xeon E5-2630 v2		
	Memória	DDR3 64 GB	DDR3 32 GB		
	SO	Red Hat 4.4.7-16	Ubuntu 14.04 - LTS 64 bits		
	Compilador	gcc 4.4.7	gcc 4.8.4	nvcc V7.0.27	
	Runtime Drivers	Xeon Phi Driver 3.1.2-1	AMD OpenCL 2.0 Driver (14.41)	OpenCL 1.1 CUDA 7.0.28 Driver(346.46)	
Acelerador	Modelo	Xeon Phi 3120	Radeon R9 290x	GTX 770	GTX Titan
	Mem. Band.	240 GB/s	320 GB/s	224.3 GB/s	288.4 GB/s
	Núcleos	57 cores	2816 cores	1536 cores	2688 cores
	Frequência	1.1 GHz	1.04 GHz	1.05 GHz	0.84 GHz

Tabela 1. Plataformas computacionais usadas nos experimentos.

3. Resultados

Os resultados obtidos com os nossos experimentos podem ser resumidos pela figura 1 onde podemos ver claramente que obtivemos bons ganhos de desempenho para o CMP e CRS com a implementação em OpenCL.
Para o CMP obtivemos ganhos de desempenho que variam de 2 a 11 vezes e para o CRS obtivemos ganhos de desempenho que variam de 6 a 49 vezes, sendo que obtivemos o melhor resultado para o CMP usando a GPU GTX Titan, enquanto que para o CRS o melhor resultado foi com a GPU R9 290X.

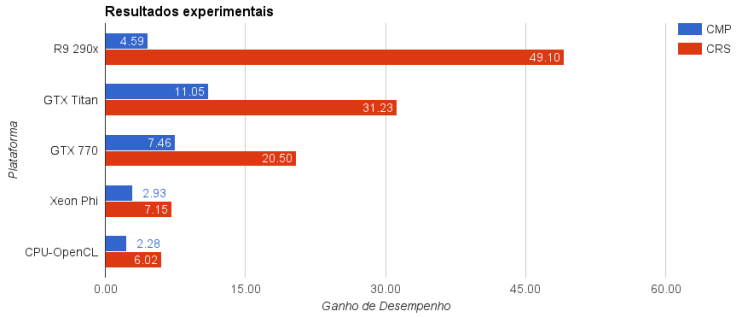


Figura 1. Resultados experimentais para o CMP e CRS, ganho de desempenho em relação a implementação em OpenMP usando o *host* da plataforma 1 descrito na tabela 1.

4. Conclusão

Como discutido na seção de 3, fica claro que se pode ganhar muito com o uso de aceleradores de *hardware* para resolver problemas como o CMP e CRS.

Referências

Hubral, P., Höcht, G., and Jäger, R. (1998). An introduction to the common reflection surface stack. In *60th EAGE Conference and Exhibition*.

Mayne, W. H. (1962). Common reflection point horizontal data stacking techniques. *Geophysics*, 27(6):927–938.

Análise de heurísticas de busca para a estimação de parâmetros geofísicos

João Henrique Speglich[1], Edson Borin [1], Jorge H Faccipieri[2], Tiago A. Coimbra[2],

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
CEP 13083-852 – Campinas – SP – Brasil

[2]Centro de Estudos de Petróleo – Universidade Estadual de Campinas (UNICAMP)
CEP 13083-970 – Campinas – SP – Brasil

{speglichj,tgo.coimbra}@gmail.com, edson@ic.unicamp.br

jorge@ggaunicamp.com

Resumo. *Estudar o espaço de busca definido por uma equação multi-paramétrica e através dos resultados obtidos definir heurísticas que melhor se adaptem ao espaço definido.*

1. Introdução

A sísmica de reflexão tem como objetivo fornecer a melhor imagem possível da subsuperfície e uma de suas principais aplicações é a exploração de hidrocarbonetos. O método *Common Reflection Surface* (CRS) fornece uma seção de afastamento nulo simulada, porém esse o faz utilizando uma maior quantidade de informações contidas no dado de multicobertura, informações essas ignoradas pelo método convencional, *Common Midpoint* (CMP) [Faccipieri 2012].

Para se obter o tempo de trânsito multi-paramétrico CRS será utilizada uma expansão em polinômio de Taylor até segunda ordem, que aproxima uma função numa vizinhança de pontos. Nesse caso, a função a ser aproximada é o tempo de trânsito de um par fonte receptor com ponto médio m na vizinhança do ponto central, m_0, e abertura h. Portanto podemos descrever a superfície de tempo de trânsito multi-paramétrico CRS (Equação 1) no qual A, B e C definem o espaço de busca a ser explorado.

$$t^2(m,h) = [t_0 + A(m - m_0)]^2 + B(m - m_0)^2 + Ch^2 \quad (1)$$

2. Metodologia

Heurísticas de busca são técnicas que exploram subconjuntos do espaço de soluções possíveis de um problema, segundo critérios arbitrários, podendo estes serem baseados em processos físicos, em dinâmicas de populações, em processos bioinspirados, etc.

As heurísticas estudadas:

- Simulated Annealing [Garabito et al. 2012]: Um método de otimização global que distingue entre diferentes ótimos locais. Ao maximizar uma função, qualquer ponto que possua um valor maior do que o atual é aceito e o processo se repete

a partir deste novo ponto. Um ponto de valor menor também pode ser aceito. Assim, pode escapar de ótimos locais. A decisão de escolher um ponto menor é feita através da analogia a equação diferencial do calor. Com interações do processo de otimização o algoritmo produz soluções mais próximas ao máximo global.

- Differential Evolution e Modificações [Garabito et al. 2012]: O DE baseia-se em mecanismos evolutivos biológicos, onde populações de pontos, de tamanho NP, são espalhadas pelo espaço de busca e se interagem através de combinações lineares, gerando novos descendentes em cada geração, somente os NP que obtiverem os melhores resultados permanecem, convergindo ao final das gerações a um máximo global.

3. Resultados

Investigando-se o espaço de busca foi possível determinar que o parâmetro B não altera significativamente o resultado da função objetivo, desde que consideradas aberturas na direção de m suficientemente pequenas. Isso permitiu que as análises fossem realizadas em um plano, onde apenas os parâmetros A e C variam.

Dentre as heurísticas estudadas, o DE, contando com uma população de pontos explorando o espaço de busca, apresentou os melhores resultados, uma vez que foi capaz de identificar o máximo global de coerência na maioria das vezes, ao contrário do SA que, contando apenas com um ponto explorando o espaço, precisa que seus parâmetros sejam guiados a fim de se evitar máximos locais.

4. Conclusão

O espaço de busca agora passa a ser bidimensional. Heurísticas de Busca que percorrem o espaço de busca de maneira mais abrangente e com um número maior de pontos se mostraram mais eficazes quando estamos lidando com espaços de busca que contém máximos locais, portanto para o problema estudado o DE se mostrou superior ao SA.

Referências

Faccipieri, J. H. (2012). *Separação e Processamento de difrações em dados geofísicos de refleção*.

Garabito, G., Stoffa, P. L., Lucena, L. S., and Cruz, J. C. (2012). Part i — {CRS} stack: Global optimization of the 2d crs-attributes. *Journal of Applied Geophysics*, 85:92 – 101.

HOW A DRUG CAN MAKE YOUR BRAIN FORGET THE WORDS? A LANGUAGE FUNCTIONAL MRI STUDY WITH TOPIRAMATE IN EPILEPSY AND HEADACHE.

Akari Ishikawa*, Tátila M. Lopes, Tamires A. Zanão, Rubens Mariano Jr.*, Brunno M. Campos, Danielle S. Garcia, Bárbara Braga, Alberto L. C. Costa, Fernando Cendes, Clarissa L. Yasuda.

Neuroimaging Laboratory CEPID BRAINN – University of Campinas (UNICAMP)

ra163282@students.ic.unicamp.br, cyasuda@g.unicamp.br

Abstract. *To demonstrate how TOPIRAMATE (TPM) hampers words formation we performed cognitive tests and language functional MRI (fMRI) study for healthy volunteers, patients (taking TPM) with headache and others with epilepsy. We identified poor cognitive performance and alterations on both brain activations and deactivations during language paradigm fMRI in subjects taking TPM.*

1. Introduction

Despite the excellent control of both seizures and headache, the drug TOPIRAMATE (TPM) can cause language (specially word finding difficulties) and memory impairment, mostly unnoticed by subjects [1, 2]. So far, it is unknown how this drug changes brain function as well as the mechanism for these side effects. Here we applied cognitive tests and language fMRI to healthy controls, subjects with migraine/headache (HEAD) and others with temporal lobe epilepsy (TLE) to investigate dysfunction on brain activations and deactivations. We hypothesize that TPM disrupts normal pattern of activations and deactivations, resulting in poor cognitive performance.

2. Materials and Methods

2.1 Subjects

After Ethical approval, we performed a cross-sectional study, recruiting 24 healthy controls (18 women, mean age 42±13 years), 15 patients with migraine/headache taking TPM (HEAD-TPM, 12 women, 39±12 years) and 12 patients with epilepsy using TPM (TLE-TPM, 11 women, 39±13 years). Patients were recruited from both Epilepsy and Headache outpatient clinics at UNICAMP Hospital.

2.2 fMRI acquisition

All subjects performed a language fMRI study in a 3T PHILIPS scanner with a blocked-design language paradigm (alternating task and rest every 20 seconds); subjects were instructed to covertly (silently, not loud) think about words beginning with different letters (phonemic task) or think about the name of figures they were visually presented (animals and fruits- categorical task).

2.3 Cognitive tests

All subjects underwent cognitive testing which consisted in generating words beginning with letter F, A and S and then, animals' names (one minute for each task).

2.4 Imaging Processing and statistics

First, all functional images were realigned, normalized, smoothed and co-registered with the structural image (T1 weighted) of the brain using a MATLAB toolbox UF²C[3]. We then performed a first-level (individual analysis) on SPM12 (http://www.fil.ion.ucl.ac.uk/spm/software/spm12/), in order to model phonemic task against categorical task and rest to obtain individual contrasts for each *stimuli*; to expedite the whole process, we developed some scripts in MATLAB platform to automatically process the first-level for all 100 subjects. To accelerate the visual checking of each individual collection of activations and deactivations, a second MATLAB script was created to generate slice view figures for each subject containing a set of figures. Using another script in Perl, we created HTML pages which showed individual sets of SPM maps´ figures in a web browser to optimize the checking step by a neurologist. Lastly, these maps were carried to a second-level analysis (group analysis) in SPM with full-factorial model. Here, we also used scripts in MATLAB to automatically perform series of group comparisons. Statistical analyses of cognitive tests were performed with SPSS22.

3. Results

Groups were balanced for age ($p=0.8$) and gender ($p=0.5$). A multivariate analysis of language tests([$F_{(4,92)}=6$, $p<0.0001$, Pillai's Trace=0.4, partial $\eta 2=0.2$]) showed significant reduction of word production with letters (FAS test) for both HEAD-TPM and TLE-TPM ($p<0.001$) compared to controls; no significant differences were observed for categorical (animals) word production ($p<0.05$) for these 2 groups, compared to controls.

On fMRI results we observed less activations and deactivations for both HEAD-TPM and EPI-TPM.

4. Conclusions

Taking the advantage of a series of scripts for MATLAB/SPM12 and Perl, we were able to process data from 100 subjects in very short time. In addition to avoid human error due to repetitive manual processes, the automation of such analysis allowed us to reach very relevant clinical results. Our data suggest that TPM prevents normal brain activations and deactivations during language production, resulting in significantly impaired language performance. Careful attention is necessary to prescribe such drug to avoid excessive cognitive dysfunction, despite its efficacy.

References

1.Yasuda CL, Centeno M, Vollmar C, Stretton J, Symms M, Cendes F, et al. The effect of topiramate on cognitive fMRI. Epilepsy research. 2013;105(1-2):250-5.

2.Loring DW, Williamson DJ, Meador KJ, Wiegand F, Hulihan J. Topiramate dose effects on cognition: a randomized double-blind study. Neurology. 2011;76(2):131-7.

3. de Campos, B. M., Coan, A. C., Lin Yasuda, C., Casseb, R. F. and Cendes, F. (2016), Large-scale brain networks are distinctly affected in right and left mesial temporal lobe epilepsy. Hum. Brain Mapp.. doi: 10.1002/hbm.23231

Análise de superfícies de tempo de trânsito na análise de dados sísmicos

Henrique Machado Gonçalves, Edson Borin, Jorge Henrique Faccipieri Jr , Tiago A. Coimbra

e-mail: henriquemg93@gmail.com,edson@ic.unicamp.br, jorge.faccipierri@gmail.com, tgo.coimbra@gmail.com

Resumo. *O método CRS (do inglês, common reflection surface) que consiste em um operador de empilhamento multiparamétrico, o qual admite interpretações geofísicas de seus parâmetros. Para realizar o empilhamento, utiliza-se uma equação de tempo de trânsito para corrigir a influência do afastamento da fonte-receptor. Existem diversas curvas de tempo de trânsito que possuem características diferentes. Este trabalho consiste em investigar as diferentes curvas e avaliar suas características.*

1. Introdução

A exploração de hidrocarbonetos requer um reconhecimento da subsuperfície, pois as regiões de fácil acesso foram exauridas ou estão em fase de exploração. Para isso, diversas técnicas foram desenvolvidas para mapear a subsuperfície e têm como abordagem resgatar o máximo de informação possível do meio geológico.

Desenvolveu-se o método CRS (do inglês, *Common Reflection Surface*) [1] para esse propósito. A obtenção do dado utilizado no CRS consiste em um conjunto composto por uma fonte e receptores que registram a resposta da subsuperfície; esse conjunto é rearranjado após cada detonação da fonte, gerando vários conjuntos de dados para cada conjunto de receptores-fonte. A obtenção dos dados é possível devido à reflexão do sinal, dada as mudanças de densidade nas camadas da subsuperfície.

O método CRS, utilizado pelo GGA, aplica uma superfície hiperbólica como tempo de trânsito. Neste trabalho, pretende-se investigar como diferentes superfícies de tempo de trânsito [2, 3] podem afetar os resultados do processamento do dado sísmico. Para avaliar as características e os resultados, foi desenvolvido um programa para renderizar a imagem da subsuperfície e variar a equação do tempo de trânsito mantendo-se o mesmo espaço de busca para analisar os efeitos na imagem final. O programa é executado com diferentes dados de entrada e a qualidade da imagem obtida é avaliada por geofísicos. Além da equação presente no CRS[2] do GGA, foi testada a equação do método *Non-hyperbolic* CRS [3], doravante NCRS.

2. Resultados Experimentais

O método CRS apresentou a melhor taxa de SemblanceTraces/s [4], sendo assim, o método mais rápido. Em relação ao resultado do processamento, observou-se que a equação do método NCRS permitiu encontrar uma região de refletores que a equação do método CRS não identificou (Figura 1).

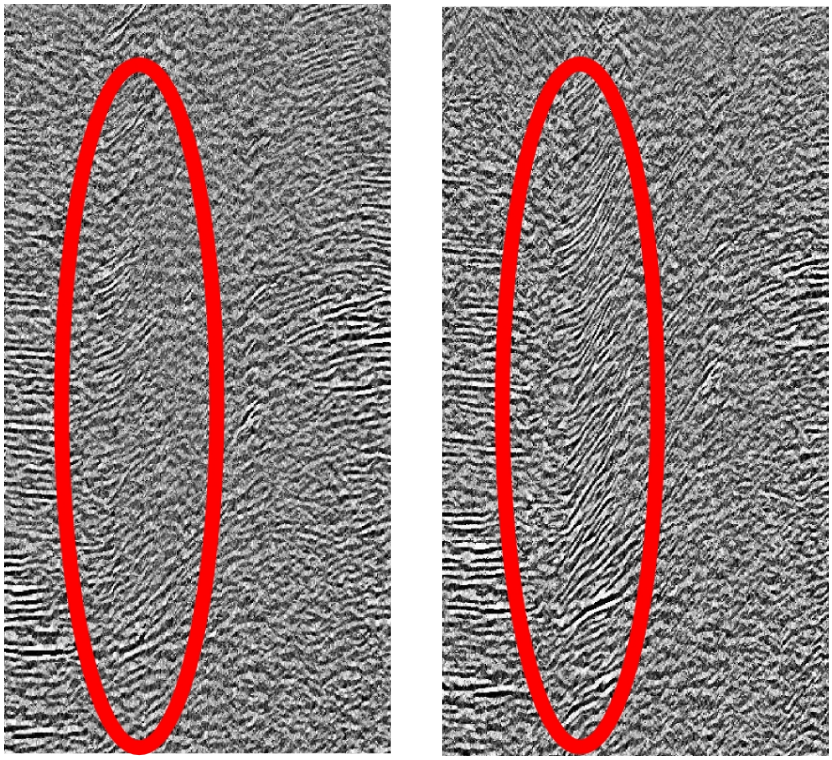


Figura 1. Comparação das imagens renderizadas utilizando o CRS (esquerda) e o NCRS (direita).

3. Conclusões

A análise dos dados obtidos indicam que a curva apresentada pelo método CRS permite uma computação mais rápida, mas apresenta uma qualidade menor de resolução. A superfície não hiperbólica [3] (NCRS) apresentou melhor qualidade de imagem, mas sua computação é um pouco mais lenta que o CRS convencional. Também foi constatado que a superfície não hiperbólica é significantemente mais precisa para grandes afastamentos de fonte-receptor.

4. Referências

[1] An introduction to the common reflection surface stack - Hubral et al. 1998.

[2] Common Reflection Surface Stack: Images and attributes. Geophysics - Jägger et al. 2001.

[3] Non-hyperbolic common reflection surface - NCRS Sergey Fomel. 2013.

[4] Accelerating semblance computations on heterogeneous devices using OpenCL - Borin et al. 2015.

COISA Bot: Uma plataforma para apoio ao ensino de pensamento computacional

Carlos Eduardo Millani[1], **Edson Borin**[1], **Juliana Freitag Borin**[1], **Augusto Fernandes Vellozo**[2]

[1]Instituto de Computação - Universidade de Campinas (UNICAMP)
Campinas - SP - Brazil

[2]TecSinapse - São Paulo - SP - Brasil

{carlos.millani,augusto.vellozo}@tecsinapse.com.br

{edson,juliana}@ic.unicamp.br

Resumo. *Novas plataformas robóticas e interfaces online para o ensino de programação e de pensamento computacional focadas em crianças estão sendo lançadas a cada ano, e o número de países que investe no ensino dessas tecnologias também aumenta. Um empecilho porém é o elevado custo da implantação da maioria dessas plataformas em escolas. Através da utilização de uma plataforma virtual (COISA [1]) e uma biblioteca para programação em blocos (Blockly [2]), propomos o COISA Bot, uma plataforma robótica de baixo custo que possibilita a programação através de blocos em aplicativos para dispositivos móveis.*

1. Motivação

Observa-se a cada ano o aumento no número de plataformas robóticas focadas no ensino de programação e pensamento computacional. O ensino desta tecnologia recebe atualmente bastante atenção, com países da Europa e nos Estados Unidos incluindo o tópico na grade de suas escolas [3, 4]. Um grande empecilho entretanto é o custo da maioria dessas plataformas. Desde as mais completas até as mais simples, a implantação em larga escala exige um alto investimento, que pode ser inviável para a realidade das escolas brasileiras. A Tabela 1 nos mostra o preço de compra de uma unidade de diversas plataformas.

Nome	Preço ($)	Nome	Preço ($)
mBot	90	Shield Bot	69.90
Kibo	229	Snap	99
Cubetto	225	OzoBot	59
COISA Bot	40		

Tabela 1. Comparativo de preço entre COISA Bot e diversas outras plataformas

2. Objetivos

Utilizando *hardware* de baixo custo, visamos criar uma plataforma que supra essa lacuna no mercado e possibilite o uso de brinquedos educacionais em escolas de todo o Brasil. Podemos ver na Tabela 1 que o preço atual de produção de um COISA Bot é inferior ao de outros produtos no mercado. Esse preço foi contabilizado considerando a compra individual de componentes, sendo possível reduzir ainda mais esse valor com um processo de fabricação em escala.

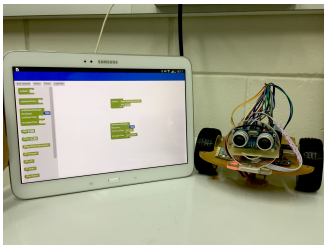


Figura 1. COISA Bot

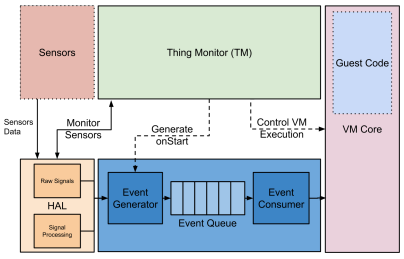


Figura 2. Esquemático do COISA

3. Método

A plataforma utiliza o COISA, uma plataforma virtual para dispositivos compactos que abstrai a complexidade de se manipular o *hardware* em baixo nível, em conjunto com uma adaptação da biblioteca Blockly da Google, que permite a criação de códigos por blocos.

4. Resultados Experimentais

Uma camada de tratamento de eventos foi adicionada ao COISA, como pode ser visto na Figura 2. Dessa maneira, o usuário pode registrar blocos de código para serem executados quando uma condição de algum sensor for satisfeita. O Blockly é um ambiente para a criação de algoritmos usando blocos e que possibilita a tradução dos mesmos para linguagens como JavaScript e Python. Diversos blocos foram adaptados para gerar código na linguagem de montagem da plataforma e um montador JavaScript foi desenvolvido, permitindo que todo o processo de compilação fosse executado no navegador. Para o envio do binário gerado, procuramos manter a portabilidade. Com a biblioteca do Web Bluetooth, tentamos utilizar mais uma vez o próprio navegador, mas por se tratar de uma biblioteca em desenvolvimento, várias restrições foram encontradas. A alternativa encontrada foi o desenvolvimento de um aplicativo para dispositivos móveis. Toda a complexidade da geração do binário permaneceu no navegador através do uso de uma Web View e o aplicativo é responsável apenas pelo envio das informações para o robô. O robô juntamente com o aplicativo criado estão demonstrados na Figura 1.

5. Conclusões

Apesar de uma camada de código, para o tratamento de eventos, ter sido adicionada, muito pouco se alterou do *footprint* do COISA. Além da adaptação do Blockly para gerar código para o COISA também foram estudados mecanismos de comunicação, visando a portabilidade do sistema.

References

[1] C. MILLANI, A. LINHARES, R. AULER, and E. BORIN, *"COISA: A Compact OpenISA virtual platform for IoT devices"*. Anais do XVI WSCAD, 2015.

[2] "Blockly." `https://developers.google.com/blockly`. Último acesso 7 Jun. 2016.

[3] R. Paddick, *"Primary Schools Urged to Do More"*. Education Technology, 2013.

[4] "Computer science for all." `https://www.whitehouse.gov/blog/2016/01/30/computer-science-all`. Último acesso 7 Jun. 2016.

Geração automática de *callgraphs* para detecção de *energy bugs*

Luís Fernando Vieira Silva[1], **Edson Borin**[1], **Sandro Rigo**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brasil

ra173170@students.ic.unicamp.br,{edson,sando}@ic.unicamp.br

Resumo. *Este trabalho faz parte de um projeto maior que possui como objetivo modificar o compilador Android de modo que ele passe a buscar, em tempo de compilação de um aplicativo, por* energy bugs *gerados pelo uso incorreto de* `wakelocks`*. Foi construída uma ferramenta de geração de* callgraph *para auxiliar a análise manual de aplicações Android, essa análise, por sua vez, visa corroborar os resultados obtidos pelo compilador. Foram detectadas dificuldades ao se usar a técnica* reaching definition*.*

1. Motivação

Com o avanço tecnológico em processadores e dispositivos periféricos de *smartphone* e o não acompanhamento da área de baterias tornou-se essencial administrar de maneira eficiente o consumo desses dispositivos. Para isso o sistema operacional Android adota um mecanismo onde todos os componentes permanecem desligados ou em estado de hibernação até que sejam necessários. Desse modo, o programador de um aplicativo deve informar ao sistema quando deseja usar um componente específico e, depois, quando ele não é mais necessário. Essa solução apresenta uma falha pois depende do uso correto desse sistema pelo programador, o que não ocorre em todos os casos. Quando ocorre um erro na programação que leva ao consumo desnecessário de energia é caracterizado um *energy bug* [Pathak et al. 2012].

2. Objetivo

O projeto pretende modificar o compilador Android de forma que ele possa, estaticamente, reconhecer quando uma aplicação possui um *energy bug*. Neste contexto, o objetivo desse trabalho foi analisar o código de aplicações manualmente visando corroborar os resultados do compilador e adquirir conhecimentos pertinentes ao seu desenvolvimento.

3. Método

Foram selecionadas 233 aplicações que pedem permissão para o uso de `wakelocks` como base para nossos estudos. Realizamos uma busca por programas que decompilariam essas aplicações e optamos por utilizar a ferramenta *open source* jadx [Skylot], assim pudemos analisar o código delas.

Desenvolvemos também uma ferramenta que recebe como entrada um aplicativo Android qualquer e cria um grafo de chamadas (*callgraph*) com os métodos que interagem com os `wakelocks` permitindo uma análise mais rápida de cada aplicação. A figura 1 é um exemplo de PDF gerado pela ferramenta.

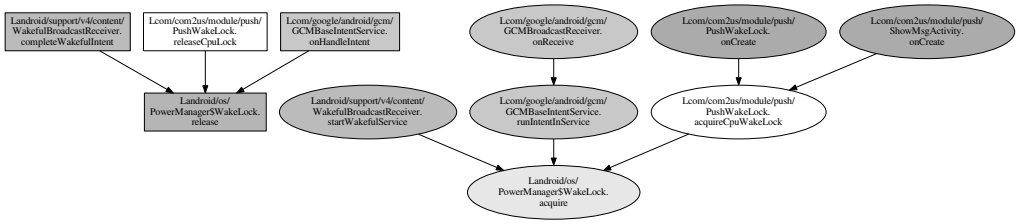


Figura 1. Exemplo de *callgraph* gerado pela ferramenta.

4. Resultados

Dos 233 aplicativos iniciais, 105 foram analisados e, entre eles, 32 possuem *energy bugs* em alguma parte do código.

Notamos que, em 45% dos aplicativos que usam `wakelock`, ocorre uma checagem similar ao exemplo 1. Isso causa um problema para a *data flow analysis* já que ela acusa que o `wakelock` pode estar ativo após a execução do `if`, podemos notar que isso não é verdade pois ele será liberado toda vez que estiver ativo.

Descobrimos também que os `wakelocks` possuem por padrão um contador fazendo com que, caso tenha sido feito, por exemplo, 3 `acquires`, serão necessários 3 `releases` para que ele seja desativado. Esse fato impossibilita o uso da análise por *reaching definition* já que nela uma definição desativa outra.

```
1 if(wakelock!=null && wakelock.isHeld()){
2     wakelock.release();
3 }
```

Exemplo 1. Checagem comum ao liberar o wakelock

5. Conclusões

Percebemos que ao usar a *data flow analisys* devemos fazer uma otimização para que a análise não infira que códigos como o do exemplo 1 sejam um *energy bug*. Além disso percebemos que uma análise por *reaching definition* é incapaz de lidar com `wakelocks` que usam um contador de referências, podendo levar o compilador a um falso negativo.

Podemos também notar que uma alta porcentagem de aplicações possuem algum erro de programação que pode ocasionar um *energy bug* (aproximadamente 30%).

6. Agradecimentos

Gostaríamos de agradecer à Samsung pelo suporte financeiro recebido para esse trabalho.

Referências

Pathak, A., Jindal, A., Hu, Y. C., and Midkiff, S. P. (2012). What is keeping my phone awake?: Characterizing and detecting no-sleep energy bugs in smartphone apps. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 267–280, New York, NY, USA. ACM.

Skylot. Jadx: Dex to java decompiler. `https://github.com/skylot/jadx`. Acessado: 16-06-2016.

Projeto de um Dispositivo IoT de Baixo Custo para Sensoriamento de Ambientes

Diogo Hideki Shiraishi[1], Luan Egidio Ferreira[1], Edson Borin[1], Juliana Freitag Borin[1]

[1]Instituto de Computação – Universidade Estadual de Campinas
Campinas, SP – Brasil

diogohshiraishi@gmail.com, luanegidioferreira@gmail.com,
edson@ic.unicamp.br, juliana@ic.unicamp.br

Resumo. *A revolução da Internet das Coisas é um fenômeno em ascensão. Sobre este conceito e no contexto de* smart buildings *e* smart homes*, é apresentado um projeto de um dispositivo de baixo custo para sensoriamento de ambientes. Assim, utilizando a plataforma de desenvolvimento NodeMCU que integra um SoC ESP8266 com rádio Wi-Fi são conectados sensores de temperatura, umidade e luminosidade para coleta de dados. Estas informações são enviadas para um serviço de coleta de dados na nuvem computacional da Google que integra os dados da rede de sensores que pode ser estabelecida com os dispositivos propostos.*

1. Contexto

Nos últimos anos, é inegável a crescente utilização de sensores no contexto de Internet das Coisas ou IoT (do inglês, *Internet of Things*). Essa revolução ocorre devido à redução do custo e aos avanços em tecnologias de comunicação, processamento e consumo energético em sistemas computacionais.

Uma das aplicações principais na IoT é o sensoriamento de ambientes, como prédios comerciais (*smart buildings*) e residências (*smart homes*) com o objetivo de monitorar o consumo de recursos e fornecer controle de acesso. Para isto, a aplicação de *motes* (ou *sensor nodes*) em rede é necessária de modo a prover coleta de dados para análise, interpretação e decisão, dando significado e finalidade a eles. Desse modo, uma característica relevante a ser considerada em um projeto de dispositivo IoT é o baixo custo. Essa diretriz é factível de ser alcançada dada a emergência e disponibilidade de diversas soluções de plataforma de desenvolvimento, de modo que o projetista deve ser capaz de identificar as necessidades e demandas do projeto a serem atendidas.

A utilização de dispositivos em IoT de baixo custo torna acessível a implementação de uma rede de nós de sensores em um contexto a ser monitorado, estudado e controlado.

2. Objetivo

Objetiva-se projetar um modelo de um dispositivo IoT de baixo custo para sensoriamento de ambientes que possibilite e auxilie o desenvolvimento de *smart buildings* e *smart homes*.

3. Métodos

Inicialmente, deverá ser determinada a plataforma de desenvolvimento a ser empregada no dispositivo. Para isso, determinadas especificações como consumo de energia, conectividade, robustez, versatilidade e tamanho deverão ser considerados. Desse modo, são candidatas plataformas compactas, com conectividade sem fio, interface para acoplamento de sensores e processador para execução de programas.

Em seguida, de posse dos sensores a serem utilizados e do pequeno gabinete plástico será realizada proposta de montagem dos elementos, juntamente com a bateria para alimentação do sistema. Os componentes para sua montagem também deverão ser considerados. Por fim, será avaliada sua funcionalidade a partir de *software* desenvolvido especificamente para o projeto.

4. Resultados

Para o dispositivo foi escolhida a plataforma de desenvolvimento NodeMCU devido ao baixo custo, tamanho reduzido e um SoC ESP8266 que inclui um processador e um rádio Wi-Fi. De posse da plataforma computacional, foi desenvolvido um circuito integrando um circuito de alimentação com três pilhas AA, foram acoplados sensores de temperatura, umidade (DHT11) e luminosidade (LDR), e estes foram integrados em um pequeno gabinete plástico. Em seguida, foi desenvolvido um *software* que realiza a leitura dos sensores e envia os dados pela interface Wi-Fi para um serviço de coleta de dados na nuvem computacional da Google (Google Cloud Platform).

O protótipo do dispositivo desenvolvido tem baixo custo, aproximadamente US$ 15, considerando sua funcionalidade, aplicabilidade e versatilidade. Nos experimentos realizados, com uma baixa taxa de amostragem (p.ex: 1 amostra a cada 30 minutos), esse dispositivo pode operar por longos períodos de tempo, na ordem de meses, alimentado apenas por 3 pilhas comuns alcalinas de 1,5 V.

5. Conclusões

O dispositivo IoT projetado contempla as principais características de baixo custo e conectividade. A versatilidade do dispositivo possibilita diferentes configurações de funcionamento, com diferentes sensores, dada sua abordagem modular. Isso permite uma abrangência maior frente a demandas específicas distintas. Por outro lado, adquiriuse conhecimento necessário para que possam ser desenvolvidos também dispositivos com finalidades específicas, atendendo prioridades individuais de projeto.

Em seguida, serão realizados estudos de modelos alternativos de montagem com aplicações para finalidades específicas, de modo a comportar a utilização de novos sensores e o acréscimo de novas funcionalidades, a fim de obter dispositivos personalizados para cada contexto de aplicação.

Referências

NodeMCU DEVKIT V1.0. (16 de maio de 2015). Disponível em: github.com/nodemcu.

The Internet of Things Is Far Bigger Than Anyone Realizes. (Novembro de 2014). Disponível em: wired.com/insights/2014/11/the-internet-of-things-bigger/.

Projeto de uma Interface Cérebro-Computador Baseada em Imaginação de Movimento

Maria B. Kersanach[1] [2], Luisa F. S. Uribe[2], Thiago B. S. Costa[2], Romis Attux[2]

[1] Instituto de Computação – UNICAMP – Campinas, SP – Brasil
[2] DCA/FEEC – UNICAMP – Campinas, SP – Brasil
ra156571@students.ic.unicamp.br, attux@dca.fee.unicamp.br

Resumo. Neste trabalho, apresenta-se a implementação de uma Interface Cérebro-Computador baseada em Imaginação de Movimento, e discutem-se os resultados obtidos por meio da técnica de wrappers para a seleção de eletrodos. Esses resultados mostram a relevância de se customizar o conjunto de atributos para cada usuário, justificando assim o paradigma adotado.

1. Introdução

Interface Cérebro-Computador (BCI, do inglês *brain-computer interface*) é um sistema de comunicação no qual comandos que um indivíduo envia para o mundo externo não passam por vias biológicas normais [1], mas sim pela tradução da atividade eletroquímica do tecido neural, evocada ou espontânea. Será abordado neste trabalho o paradigma de imaginação de movimento, o qual é definido como a simulação mental de um movimento cinestésico que modula atividades no córtex sensório-motor sem qualquer movimento físico do corpo [2].

2. Materiais e Métodos

As aquisições foram realizadas com o sistema de eletrodos secos g.SAHARAsys e o amplificador de sinais biológicos g.USBamp (GTEC). A montagem dos eletrodos privilegiou a região do córtex motor. Convém ressaltar que o experimento foi aprovado pelo comitê de ética da UNICAMP (protocolo número 791/2010). Realizaram-se 20 sessões de 8 segundos cada, para cada atividade de imaginação de mão esquerda, direita e repouso. A frequência de amostragem do sinal foi de 256 Hz, havendo janelamento de 3 s com sobreposição de 0,5 s. As bandas utilizadas para representar a informação do espectro de potência do sinal foram 8 a 12 Hz, 12 a 16 Hz e 16 a 20 Hz.

3. Construção da Interface

O pré-processamento se dá pela aplicação do filtro de Referência de Média Comum (CAR) [3]. Para a extração de características, utiliza-se o Método do Periodograma de Welch [3], gerando-se para cada janela um vetor de atributos com o conteúdo das três bandas supracitadas. Na etapa de classificação, foi aplicado um classificador linear baseado no método de mínimos quadrados [3].

70% das sessões são aleatoriamente escolhidas para o treinamento do classificador e as outras 30% para o teste. A escolha dos eletrodos é realizada por um wrapper [3] progressivo, o qual utiliza o próprio classificador para selecionar os eletrodos mais representativos. A característica progressiva (iniciar com um eletrodo e iterativamente criar conjuntos de até 16 eletrodos ao testar os subgrupos possíveis)

possui a vantagem de ser computacionalmente pouco custosa e se mostrou satisfatória no contexto de BCIs.

4. Resultados

Devido a limitações de espaço, serão apresentados apenas os resultados de um dos usuários que testaram a interface, sendo este saudável, com 20 anos de idade e do sexo feminino. Vale ressaltar que não houve treinamento prévio com o paradigma.

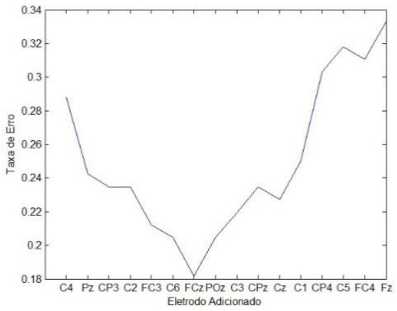


Figura 1. Variação da taxa de erro para a ordenação de eletrodos por wrapper

O gráfico ilustra a relevância da seleção de atributos, visto que a taxa de erro varia desde aproximadamente 30% iniciais até o ponto mínimo de 18% com o conjunto ótimo de 7 eletrodos (na ordem, C4, Pz, CP3, C2, FC3, C6, FCz). Percebe-se ainda que, posteriormente, atinge-se quase 35% ao utilizar todos os 16 eletrodos na classificação. A média das taxas de erro para este sujeito, em quatro sessões, é de 15,7%, com desvio padrão de 5,2%. Os resultados são compatíveis com abordagens da literatura [4].

5. Conclusão

Nota-se que a escolha dos eletrodos tem grande impacto no desempenho atingível pelo usuário. Isso justifica a técnica de seleção de atributos, confirmando a validade das hipóteses, técnicas e paradigmas adotados.

Agradecimentos

Os autores agradecem ao CNPq o apoio financeiro.

Referências

[1] Wolpaw, J. R, Birbaumer, N., McFarland, D. J., Pfurtscheller, G. e Vaughan, T. M. (2002) “Brain–computer interfaces for communication and control”. Clinical Neurophysiology, Vol. 113, pp. 767-791.

[2] Hwang, H. J, Kwon K., Im, C. (2008) “Neurofeedback-based motor imagery training for brain-computer interface (BCI)”, Journal of Neuroscience Methods, Vol. 179, pp. 150-156.

[3] Leite, S. N. C. (2016) Contribuições ao Desenvolvimento de Interfaces Cérebro-Computador Baseadas em Potenciais Evocados Visualmente em Regime Estacionário, Tese de Doutorado, UNICAMP.

[4] Dornhege, G., Millán, J. R., Hinterberger, T., McFarland, D. J., Müller, K.-R. (2007) Toward Brain-Computer Interfacing, MIT Press.

Quid.net.br - Uma Ferramenta para Visualização e Expansão de Revisões Bibliográficas

Alysson Bolognesi Prado[1], Maria Cecília Calani Baranauskas[1]

[1] Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
CEP 13083-852 – Campinas – SP – Brasil

{aprado, cecilia}@ic.unicamp.br

Resumo. *Na produção de conhecimento científico, referências bibliográficas desempenham importante papel mediador entre os cientistas, que usam resultados publicados anteriormente para contextualizar, apoiar, ou mesmo confrontar suas novas afirmações. Revisões bibliográficas são uma forma de localizar, selecionar e sumarizar publicações cobrindo um determinado tema. Neste trabalho apresentamos quid.net.br, uma ferramenta que explora conceitos da Actor-Network Theory – ANT – para visualizar como os elementos de uma coleção de publicações se relacionam, identificando outros itens bibliográficos com potencial de interesse ao cientista usuário.*

1. Introdução

Thomas Kuhn definiu uma comunidade científica como um grupo de pessoas que estudam um assunto específico compartilhando um paradigma, isto é, um conjunto de princípios, conceitos, linguagem e métodos. A literatura científica tem um papel fundamental para fornecer a um novato os fatos estabilizados e amplamente aceitos pela comunidade, bem como divulgar descobertas e propostas mais recentes. O paradigma corrente é formalizado, disseminado e evolui através da literatura.

A disponibilidade pública na Internet de dados bibliográficos e sociais de cientistas apresenta expressivo potencial de apoio a pesquisadores, ao mesmo tempo em que constitui um desafio para a seleção, organização e apresentação destes dados de forma a constituir informação útil para projetos de pesquisa.

Neste trabalho apresentamos **quid.net.br**, uma ferramenta para produzir visualizações que representam sinteticamente um conjunto de publicações e seus relacionamentos, permitindo compreender como seus elementos se relacionam, e identificar outras publicações com potencial de interesse ao cientista usuário.

2. Referencial Teórico

Como parte de seu trabalho em Estudos Sociais da Ciência e Tecnologia, Latour (2011) identificou que a construção de um fato científico é um fenômeno sociológico que vai além da mera interação face-a-face das pessoas envolvidas. A literatura científica, entre outros atores não-humanos, desempenha importante papel mediador e agregador na estruturação de uma comunidade científica, segundo a Actor-Network Theory – ANT.

Quando usada como premissa em uma argumentação, uma citação bibliográfica ajuda o citador a ter seus argumentos aceitos, enquanto confere ao citado maior

visibilidade, estabelecendo uma associação de benefício mútuo. Quando uma alegação feita em um artigo é posteriormente aceita por muitos outros e incorporada a eles na forma de citações, é progressivamente reconhecida como fato científico e incorporada ao paradigma vigente (Figura 1).

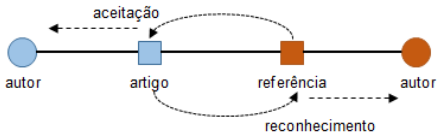


Figura 1. Relação de benefício mútuo entre cientistas, mediada por publicações

3. A ferramenta

Quid[1] explora a relação entre uma publicação e suas referências como uma forma de capturar o contexto sociotécnico em que está inserida. Considerando que Revisões Bibliográficas Sistemáticas (Prado *et al.*, 2014), ou mesmo agrupamentos não sistemáticos de fontes bibliográficas, têm foco em um tema de pesquisa específico, localizar os elementos centrais daquele contexto pode apontar para as bases que auxiliam na compreensão daquele tópico de pesquisa.

Uma Revisão Bibliográfica Sistemática realizada com apoio do software START pode ser diretamente importada no sistema *Quid*. Dados de autores e publicações também podem ser cadastrados manualmente. O sistema conta, entretanto, com uma base de dados de publicações atualizada periodicamente de forma automatizada, buscando em repositórios com API pública como DOI, DBLP, IEEExplore, Springer, Biblioteca Digital da Unicamp via OAI-PMH, e também no Currículo Lattes sob demanda do usuário.

O sistema constrói um grafo de associações, a partir das publicações selecionadas, contendo suas referências e autores. Esta estrutura permite a extração de métricas, como *betweeness centrality* e tamanho da maior componente conexa, bem como a visualização com algoritmo de *spring forces layout* para uma análise qualitativa.

4. Conclusão

O suporte de uma teoria sociológica mostra-se fundamental para a melhor compreensão do fenômeno que os dados representam, e consequentemente o seu uso. O sistema vem sendo utilizado no apoio a revisões sistemáticas e na análise de conferências científicas.

Referências

Latour, B. (2011). Ciência em ação – como seguir cientistas e engenheiros sociedade afora. Editora da Unesp.

Prado, A. B.; Baranauskas, M. C. C.; Bittencourt, I. I.; Gonçalves, F. M.; (2014) "Expandindo revisões bibliográficas sistemáticas pela análise de redes sócio-técnicas científicas". XLI Seminário Integrado de Software e Hardware (SEMISH) - CSBC.

[1] Da expressão em latim *quid pro quo*, que significa "isto por aquilo" e remete às relações de benefício mútuo destacadas pela ANT.

Uma análise do consumo energético de processadores heterogêneos

Gabriel Souza Franco[1], Edson Borin[1], Sandro Rigo[1], Alexandro José Baldassin[2], Lucas Francisco Wanner[1]

[1]Instituto de Computação –UNICAMP
Av. Albert Einstein, 1251 – Campinas – SP – Brasil

gabriel.franco@students.ic.unicamp.br

{edson,sandro,lucas}@ic.unicamp.br

[2]Instituto de Geociências e Ciências Exatas de Rio Claro – UNESP
Av. 24-A, 1515 – Rio Claro – SP – Brasil

alex@rc.unesp.br

Abstract. *In the last years an exponential growth of mobile devices was seen, which brought new problems to light. Due to the dependance on batteries, is isn't enough to run every load at max frequency, instead trying to minimise energy consumption bearing in mind time deadlines present in many cases. This work analyses perfomance and energy consumption of a processor and processor architecture found in mobile devices.*

Resumo. *Nos últimos anos houve um crescimento exponencial de dispositivos móveis, que trouxeram novos problemas e considerações à tona. Devido à dependência de baterias, não é mais suficiente executar em frequência máxima qualquer código que requira recursos computacionais, mas sim tentar minimizar o consumo de energia tendo em vista o tempo limite muitas vezes existente. Esse trabalho analisa o desempenho e o consumo de energia de um processador e arquitetura em uso em dispositivos móveis.*

1. Metodologia

Para a realização dos experimentos foi empregada uma placa de desenvolvimento *ODROID XU+E*. A mesma possui um processador *Exynos 5410*, em configuração *big.LITTLE* de 4 núcleos *Cortex-A7* de baixo consumo de energia e mais 4 núcleos *Cortex-A15* de alto desempenho. Também possui medidores de energia conectados diretamente às linhas de alimentação do processador, memória e *GPU*, permitindo assim uma coleta acurada do consumo instantâneo de cada componente durante os experimentos. A aquisição dos dados foi feita de maneira automatizada, pela montagem de *scripts* que gerenciaram a execução de cada experimento e a organização dos arquivos resultantes, que foram posteriormente movidos para outro sistema para análise. Cada aplicativo foi executado com todas as frequências disponíveis ao processador, sendo que o escalonamento dinâmico de frequência foi desabilitado.

Foi utilizada a suíte de aplicativos PARSEC 3.0 para a realização dos experimentos, descritas por Bienia [Bienia et al. 2008]. A suíte foi ajustada para a plataforma ARM e os programas foram compilados usando o GCC 5.1.0. Nem todos os aplicativos puderam ser testados por erros de compilação.

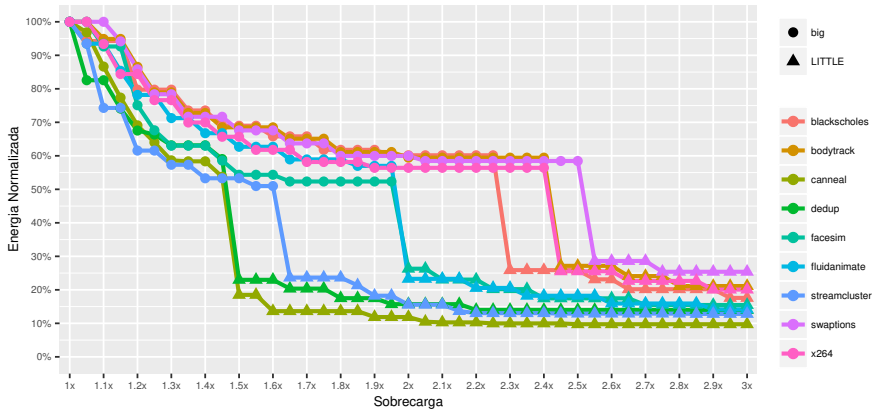


Figura 1. Consumo energético normalizado em função da sobrecarga máxima de desempenho aceitável

2. Resultados

Os resultados obtidos demonstram que existem ganhos possíveis de eficiência energética caso um programa não precise ser executado o mais rapidamente possível. Na figura 1 está representado a razão entre o menor consumo de energia dado um tempo limite e o consumo do ponto mais rápido. Pode-se notar que há sempre uma queda acentuada no consumo, que corresponde à passagem do *big* para o *LITTLE*. No entanto, o ponto onde ocorre essa troca depende do aplicativo, e há grande variação entre eles. Dependendo dos requerimentos, uma sobrecarga dessas pode ser aceitável ou não.

Uma aplicação que demonstra esse meio-termo é o processamento de entrada do usuário em dispositivos móveis. O tempo de resposta de uma pessoa é muito maior do que o tempo necessário para processar a entrada, então é benéfico diminuir a frequência de operação até o limite de resposta, já que isso acarretará uma grande diminuição do consumo de energia.

3. Conclusão

Neste projeto foram conduzidos experimentos para averiguar a relação entre o desempenho e o consumo de energia em uma série de aplicações, utilizando-se da arquitetura *big.LITTLE*. Foi possível avaliar a separação das características de cada *cluster*, o *LITTLE* com ênfase no consumo de energia e o *big* com foco no desempenho.

Os resultados indicam que os núcleos heterogêneos permitem um aumento significativo na eficiência energética em situações onde uma perda de desempenho entre $1,5\text{x}$ e $2,6\text{x}$ possa ser tolerada. Este aumento significativo também sugere que a introdução de um terceiro tipo de núcleo, intermediário, pode ser interessante, principalmente nos casos onde a utilização dos processadores *LITTLE* causam um grande impacto no desempenho do sistema.

Referências

Bienia, C., Kumar, S., Singh, J. P., and Li, K. (2008). The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM.

Uma análise do consumo energético de uma plataforma para dispositivos IoT

Luan Egidio Ferreira[1], Edson Borin[1], Juliana Freitag Borin[1]

[1]Instituto de Computação
Universidade Estadual de Campinas (UNICAMP) – Campinas, SP - Brazil

luanegidioferreira@gmail.com, edson@ic.unicamp.br, juliana@ic.unicamp.br

Resumo. *Diante do cenário emergente da Internet das Coisas, foi estudado uma plataforma de desenvolvimento de baixo custo e com capacidade WiFi. Conhecida como NodeMCU, essa plataforma possui o chip ESP8266, que por sua vez tem um processador e um rádio WiFi. Os estudos abordados nesse artigo tiveram como foco o seu perfil energético, visto que a duração da bateria é um ponto crítico no desenvolvimento de um dispositivo da Internet das Coisas, com o propósito de quantificar essa duração em situações de interesse, assim como identificar possíveis otimizações.*

Contexto

Com o propósito de ampliar a pesquisa no tema da Internet das Coisas ou IoT (do inglês "*Internet of Things*") estudamos a aplicação das plataformas de desenvolvimento de baixo custo emergente nos últimos anos. Entre as plataformas, o protagonista desse artigo é o NodeMCU, que, constituído pelo chip ESP8266, conta com um processador integrado a um rádio WiFi e entradas digitais/analógicas por menos de US$5, tornando-o um forte candidato para aplicações de baixo custo [Kumar, Abhijeet, e Sharma 2015].

A aplicação referência durante os estudos foi a da criação de um Prédio Inteligente (do inglês "*Smart Building*"), onde dispositivos *IoT* (chamados de "*motes*") seriam espalhados por salas, corredores e portas, por exemplo, para que diversas funcionalidades convenientes possam ser implementadas. Nesse contexto é fundamental que vários tipos de *motes* sejam alimentados a partir de baterias devido à sua mobilidade, e nesses casos o tempo de vida do dispositivo é um ponto crítico no seu desenvolvimento, uma vez que essa característica tem impacto direto sobre a viabilidade de seu uso e também no seu custo de manutenção. Sendo assim, o estudo aqui descrito é focado no consumo energético da plataforma NodeMCU.

Objetivos

O objetivo do estudo é traçar um perfil energético da plataforma, a fim de se quantificar o tempo de bateria e entender de que maneiras o consumo pode ser otimizado.

Método

Utilizando o DAQ (do inglês "*Data Acquisition System*") para medir e coletar voltagem em um resistor de valor conhecido conectado em série com o dispositivo podemos inferir o valor da corrente que passava pelo sistema, visto que estamos usando uma fonte de voltagem fixa. Com esses dados, utilizamos a ferramenta R para criar gráficos de Corrente *vs* Tempo e calculamos o uso médio de corrente utilizada pelo mote em modo *sleep* e em modo ativo, ou seja, com o rádio WiFi desligado e

ligado, respectivamente. Como o tempo de duração dos ciclos ativo e *sleep* também foram medidos, é possível obter o consumo por hora que foi utilizado, para então finalmente comparar esse resultado com as cargas oferecidas por baterias obtendo-se o tempo de vida do *mote*.

Resultados

Antes de estimarmos o tempo de bateria, percebemos que um dos fatores mais impactantes e também mais variáveis é o tempo de duração do ciclo ativo do *mote*. Isso é devido ao tempo gasto para se estabelecer uma conexão entre o *mote* e o Ponto de Acesso WiFi (abreviado por PA) e o fato que em um ambiente no qual o PA se encontra congestionado esse tempo chega perto de um minuto, o que fortemente aumenta a carga por hora, diminuindo o tempo de vida em uma dada bateria. Utilizando um PA residencial com baixo fluxo de usuários, observamos um tempo muito menor (próximo de 8 segundos) e muito mais estável para se estabelecer uma conexão.

Utilizando a ferramenta R conseguimos constatar que durante o modo *sleep* o consumo de corrente é muito baixo, mas quando o *mote* acorda para efetuar seu ciclo ativo vemos picos de até 400 mA. Fazendo a média da corrente nos estados *sleep* e ativo por vários gráficos coletados, obtemos, respectivamente, 0.00017A e 0.08A.

Agora precisamos apenas definir um perfil de uso para estimarmos o tempo de bateria do mote. Em um cenário de ambientes inteligentes faz sentido possuir um dispositivo que fica em modo *sleep* por 30 minutos e depois entra em modo ativo apenas para enviar dados de sensores à Internet. Nesse cenário podemos estimar que, sob condições de um PA não congestionado, o *mote* passaria cerca de 10 segundos no modo ativo para cada 30 minutos no modo *sleep*. Portanto, em um ciclo completo de ativo + *sleep*, temos uma carga de 0.00031 Ah. Comparando isso com uma bateria AA comum de 2.2 Ah, o mote teria um tempo de vida estimado em 295 dias, equivalente a 9 meses e meio.

Conclusões

Os resultados encontrados são satisfatórios, pois com três simples baterias AA podemos usar um dispositivo de baixo custo para uma aplicação sofisticada como a de criar um ambiente inteligente por mais de 9 meses. Portanto fica reforçado o potencial que dispositivos de baixo custo possuem em um contexto da Internet das Coisas.

Em outros estudos também encontramos algumas maneiras de otimizar o gasto de bateria do *mote*, como por exemplo criar ciclos em que o mesmo liga o processador sem ligar o rádio WiFi, armazenando os dados dos sensores em sua memória flash para em outro momento enviar esses dados para a rede [Pellepl 2015]. Essas otimizações melhorariam ainda mais o resultado que encontramos aqui.

Por fim, vale notar que o aprendizado obtido sobre o impacto que o estado de congestionamento do Ponto de Acesso tem no consumo energia do mote é valiosa para a continuidade dos estudos sobre um ambiente inteligente na Internet das Coisas.

Referencias

Kumar, Abhijeet, and Apoorva Sharma (2015) "Internet of Life (IOL).",

ISBN 978-93-5156-328-0.

Pellepl (2015) "Wear-leveled SPI flash file system for embedded devices", https://github.com/pellepl/spiffs, April.

Uma classificação de aplicações Android para detecção de *energy bugs*

Caio S. Rohwedder[1], **Edson Borin**[1], **Sandro Rigo**[1],

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brasil

ra157754@students.ic.unicamp.br, {edson,sandro}@ic.unicamp.br

Resumo. *Analisamos manualmente aplicações Android procurando por* energy bugs *(*bugs *que causam gasto excessivo de bateria). Como a inspeção manual não é um processo trivial, primeiramente identificamos quais aplicativos trariam resultados melhores, classificamos e ordenamos eles. Os resultados foram utilizados no desenvolvimento de uma análise estática em tempo de compilação que identifica se um aplicativo contém um* energy bug *ou não.*

1. Motivação

Para reduzir o gasto de energia em dispositivos móveis, o sistema Android tenta deixar cada componente ligado somente enquanto ele está em uso. Os aplicativos informam ao sistema que o componente de interesse (CPU, tela, ...) deve ficar ligado programaticamente através da obtenção de `wakelocks`. Enquanto um `wakelock` está adquirido, o componente em questão fica ligado até que o `wakelock` seja liberado [Google].

O uso indevido ou mau uso dos `wakelocks`, como por exemplo adquirir um e nunca liberá-lo, pode levar ao consumo exagerado de bateria do dispositivo, caracterizando-se um *energy bug* [Pathak et al. 2012].

2. Objetivos

O objetivo do projeto é modificar o compilador do Android para que faça uma análise estática em busca de *energy bugs*. Nosso objetivo foi analisar manualmente aplicações Android para validar os resultados do compilador, melhorar sua análise e encontrar erros comuns relacionados a `wakelocks`.

3. Método

Para realizar a análise manual, fizemos o *download* de 233 aplicações Android (entre as mais populares e as com possíveis *bugs*). Como o número de aplicações *open source* era muito baixo, utilizamos um *decompiler* para inspecionar seus códigos.

Classificamos os aplicativos em 4 tipos para separá-los pelo nível de dificuldade de análise: (0) os que não usam `wakelocks`, apesar de solicitar permissão para tal; (1) os que utilizam `wakelocks` exclusivamente nos métodos base das *activities* (`onCreate`, `onStart`, ...); (2) os que utilizam `wakelocks` somente a partir dos métodos base (direta ou indiretamente) e (3) os que utilizam `wakelocks` em outras situações. A Figura 1 mostra um resumo dos resultados. Como houve uma concentração muito grande de aplicativos do tipo (3), ordenamos as aplicações a partir dos seus grafos de chamada

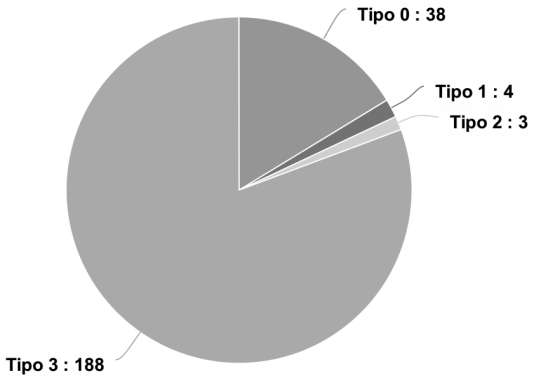


Figura 1. Classificação dos aplicativos

(*callgraphs*), dando preferência para *callgraphs* menores e que contém métodos base de *activity*.

Implementamos também *scripts* para automatizar a comparação dos resultados da análise manual com os resultados da análise realizada pelo compilador.

4. Resultados Experimentais

Com a comparação dos resultados manuais com os do compilador, criamos um conjunto de aplicações sintéticas para investigação e observamos que dentro do ciclo de vida de uma *activity* um `wakelock` que foi adquirido deve ser liberado até o método `onStop`. Liberar um `wakelock` no `onDestroy` foi considerado um *energy bug* pois esse método pode não ser chamado, por exemplo, ao sair de um aplicativo clicando no botão *home*.

Dos 233 aplicativos iniciais, 105 foram analisados manualmente, dos quais 32 apresentaram *energy bugs*.

5. Conclusões

Pudemos observar que a análise manual de *energy bugs* em aplicações reais nem sempre é viável por conta da complexidade do código, mas a classificação realizada nos ajudou a selecionar os aplicativos mais suscetíveis à análise manual. A implementação dos *scripts* de teste aliado aos nossos resultados permitiu que identificássemos limitações e oportunidades de melhoria na análise do compilador de forma ágil, a cada nova versão.

Por fim, nossa análise manual revelou que aproximadamente 30% dos aplicativos analisados possuíam potenciais *energy bugs*. Gostaríamos de agradecer à Samsung pelo suporte financeiro recebido para esse trabalho.

Referências

Google. Keeping the device awake. `https://developer.android.com/training/scheduling/wakelock.html`. Acessado: 18-06-2016.

Pathak, A., Jindal, A., Hu, Y. C., and Midkiff, S. P. (2012). What is keeping my phone awake?: Characterizing and detecting no-sleep energy bugs in smartphone apps. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 267–280, New York, NY, USA. ACM.