> ANAIS <

WORKSHOP DIMAp 30 ANOS

24 A 28 DE AGOSTO DE 2015

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Workshop DIMAp 30 Anos

https://www.dimap.ufrn.br/dimap30anos/

24-28 de agosto de 2015
Universidade Federal do Rio Grande do Norte, Natal, Brasil

Organizadoras

Monica Magalhães Pereira
Thaís Vasconcelos Batista
Sílvia Maria Diniz Monteiro Maia
Márcia Jacyntha Nunes Rodrigues Lucena

Realização e Organização

Departamento de Informática e Matemática Aplicada
Centro de Ciências Exatas e da Terra
Universidade Federal do Rio Grande do Norte

Apoio e Patrocínio

HIRIX Engenharia de Software

HXD Smart Solutions

Cooperativa Cultural Universitária

>Prefácio<

É com satisfação que apresentamos os Anais do Workshop DIMAp 30 Anos, realizado na Universidade Federal do Rio Grande do Norte, Natal, RN, entre os dias 24 e 28 de agosto de 2015.

O Workshop DIMAp 30 Anos é uma iniciativa do Departamento de Informática e Matemática Aplicada (DIMAp) com o objetivo de promover o departamento e celebrar os 30 anos de sua existência completados no presente ano.

Esta é a 5ª edição do workshop. A primeira edição ocorreu em 2000, em comemoração aos 15 anos do departamento. A 2ª edição ocorreu em 2003, e a 3ª edição ocorreu no ano de 2004. Na celebração dos 20 anos do departamento, foi realizada a 4ª edição, cujo ano foi 2005. A sua realização ao longo das cinco edições tem possibilitado a divulgação dos projetos de pesquisa e inovação desenvolvidos pelos professores e alunos do departamento, principalmente no âmbito da UFRN, mas também entre outras instituições do estado do Rio Grande do Norte.

Nesta edição, contamos com a presença de palestrantes renomados convidados, o que amplia a discussão, envolvendo diversos tópicos de interesse para área, tais como inovação; mercado de trabalho e a busca por grandes empresas de TI; uma outra visão sobre a web; e o espaço das mulheres no mercado de trabalho na área de computação, TI e afins.

O Workshop também contempla a divulgação e o debate dos trabalhos científicos desenvolvidos pelos professores e alunos do departamento, através da apresentação das linhas de pesquisa do Programa de Pós-Graduação em Sistemas e Computação (PPgSC) do DIMAp.

Os Anais do Workshop DIMAp 30 Anos agregam excelentes trabalhos que apresentam temas inovadores, tecnológicos e científicos. Ao todo, são 22 trabalhos científicos abrangendo as mais diversas áreas do conhecimento relacionadas à computação: Engenharia de Software; Métodos Formais; Lógica; Arquitetura de Computadores; Algoritmos; Criptografia; Inteligência Computacional e Computação Gráfica.

Os mais de 40 autores envolvidos nesses trabalhos demonstram o potencial das áreas e a atuação do departamento na pesquisa e inovação nos mais diversos temas discutidos em todo o mundo na atualidade. Além disso, também são

apresentados trabalhos do tipo poster e demonstração. Os trabalhos de demonstração, de cunho mais tecnológico, atestam a atuação do DIMAp também em inovação tecnológica, através do desenvolvimento de projetos de hardware e software como potenciais produtos de mercado. Dentre os 8 trabalhos apresentados estão protótipos de sistemas embarcados, voltados à acessibilidade e automação residencial; ferramentas de software para o auxílio no desenvolvimento de sistemas; e soluções de algoritmos para os mais diversos problemas que exigem um alto poder computacional.

A extensa participação de docentes do DIMAp e de departamentos que atuam na área de computação, TI e afins, permite aumentar a interação entre os vários departamentos da UFRN. A grande participação de discentes, tanto de graduação como de pós-graduação, comprova como a busca por mais informações, o compartilhamento de conhecimento e o debate são essenciais no ambiente acadêmico e, devem ser constantemente fomentados por aqueles que nele atuam. Mais ainda, a participação de diversos discentes, de outras áreas de atuação, comprova que a computação é ubíqua e essencial para toda a sociedade. Por isso, a evolução e a busca por constante aprimoramento são foco essencial e prioritário para os docentes e discentes que atuam no DIMAp.

Agradecemos a todos os autores pela publicação e apresentação dos seus trabalhos e a todos os participantes que contribuíram para o sucesso de mais uma edição do Workshop DIMAp, especialmente, essa edição comemorativa dos 30 anos do departamento.

Também gostaríamos de agradecer a todos os setores da UFRN e organizações que apoiaram a realização deste evento, tornando-o possível.

Finalmente, este evento não seria possível sem os esforços das várias pessoas envolvidas em sua organização. Deste modo, agradecemos a todo o comitê organizador do Workshop DIMAp 30 Anos e a todas as pessoas que contribuíram para o sucesso deste evento.

>Sumário<

>Comissão Organizadora<

Coordenação Geral do Workshop DIMAp 30 Anos

Monica Magalhães Pereira

Thaís Vasconcelos Batista

Comitê de Organização Local

Ana Caroline Medeiros Brito

Débora Karoline Silva de Azevedo

Geovanni Cassemiro Rocha

Hiago Mayk Gomes de Araújo Rocha

João Pedro Alencar Rocha de Holanda

Jonathan Wanderley de Mesquita

Jorge Pereira da Silva

Katyanna Moura

Lucas Torres de Souza

Luís Tertulino da Cunha Neto

Luísa Rocha de Azevedo Santos

Raimundo Vítor de Godeiro Marques

Ronaldo de Figueiredo Silveira

Samanta Ferreira Aires

Samuel Natã de França Borges

Thales Aguiar de Lima

Viviane Costa Pinheiro

Wanderson Ricardo de Medeiros

Arte

Cláudia Fernanda O. K. Tavares

Página do Evento

Viviane Costa Pinheiro

Gabriela Cavalcante da Silva

>Comitê de Programa<

Coordenadoras

Sílvia Maria Diniz Monteiro Maia
Márcia Jacyntha Nunes Rodrigues Lucena

Membros do Comitê de Programa

Alba Sandyra Bezerra Lopes
Anderson Paiva Cruz
André Maurício Cunha Campos
Antonio Carlos Gay Thomé
Augusto José Venâncio Neto
Benjamín René Callejas Bedregal
Carlos Augusto Prolo
Cláudia Fernanda O. K. Tavares
Edgard de Faria Corrêa
Eduardo Henrique da Silva Aranha
Elizabeth Ferreira Gouvêa
Fernando Marques Figueira Filho
Gibeon Soares de Aquino Júnior
Givanaldo Rocha de Souza
Gustavo Girão Barreto da Silva
Hélida Salles Santos
Ivanovitch Medeiros Dantas da Silva
Jair Cavalcanti Leite
João Marcos de Almeida
Leonardo Bezerra
Lyrene Fernandes da Silva
Marcel Vinícius Medeiros Oliveira
Marcelo Ferreira Siqueira
Márcio Eduardo Kreutz
Marcos César Madruga Alves Pinheiro
Márjory Cristiany da Costa Abreu
Martin Alejandro Musicante
Matheus Menezes
Pedro Petrovitch Caetano Maia
Richard Walter Alain Bonichon
Selan Rodrigues dos Santos
Thatiana C. N. Souza
Uirá Kulesza

>DIMAp/UFRN<

Chefia do Departamento de Informática e Matemática Aplicada

Nélio Alessandro Azevedo Cacho (Chefe)

Antonio Carlos Gay Thomé (Vice-Chefe)

Direção do Centro de Ciências Exatas e da Terra

Djalma Ribeiro da Silva (Diretor)

Jeanete Alves Moreira (Vice-Diretora)

Reitoria da Universidade Federal do Rio Grande do Norte

Ângela Maria Paiva Cruz (Reitora da UFRN)

José Daniel Diniz Melo (Vice-reitor da UFRN)

>Artigos de Iniciação Científica<

Uma Arquitetura de Serviço de Descoberta para IoT e SoS

Porfírio Gomes[1], **Thais Batista**[1]

[1]Departamento de Informática e Matemática Aplicada (DIMAp) – Universidade Federal do Rio Grande do Norte (UFRN)
Natal – RN – Brasil

enghaw13@gmail.com, thais@ufrnet.br

Resumo. *A Internet das Coisas (IoT) é um paradigma no qual uma variedade de objetos físicos inteligentes são capazes de cooperar com outros objetos físicos e virtuais para a construção de sistemas que realizam tarefas através da interação entre esses objetos. Sistemas podem colaborar com outros sistemas independentes e heterogêneos visando alcançar um objetivo em comum, compondo Sistemas-de-Sistemas (SoS). O processo de descoberta de entidades (objetos ou sistemas) que atendam uma dada funcionalidade é essencial para que um sistema baseado em IoT ou um SoS realizem as tarefas para as quais foram construídos. Este artigo apresenta uma proposta de Arquitetura Federada para a descoberta de objetos no domínio de IoT e de sistemas no domínio de SoS.*

1. Introdução

Em um futuro próximo, com avanços em tecnologias de rede e na capacidade computacional dos dispositivos, será possível que cada objeto da Terra seja identificável, endereçável, controlado e monitorado através da Internet. A provisão de novas funcionalidades, resultantes da integração e cooperação desses objetos com outros recursos físicos e virtuais disponíveis na Web, constitui o paradigma conhecido como *Internet das Coisas* (do inglês *Internet of Things* - IoT) [Atzori et al. 2010].

A iniciativa atual em IoT demanda a construção de plataformas e serviços que possam capturar, comunicar, armazenar, acessar e compartilhar dados do mundo físico, criando oportunidades na criação de sistemas em uma série de domínios, como o de saúde, casas e cidades inteligentes, transporte, logística, monitoramento de ambientes e defesa militar [Barnaghi et al. 2012]. Esses sistemas oferecem funcionalidades que muitas vezes não são atendidas por uma solução monolítica, sendo necessário a comunicação e a interação entre diversos sistemas independentes e heterogêneos na provisão dessas funcionalidades.

A integração e interação entre sistemas independentes e heterogêneos está relacionada ao conceito de *Sistema-de-Sistemas* (do inglês *Systems-of-Systems* - SoS). Um Sistema-de-Sistemas pode ser entendido como um conjunto de sistemas constituintes complexos, independentes, distribuídos e heterogêneos que possuem suas próprias capacidades e cooperam para atender uma missão global em comum [Maier 1998]. Em engenharia de software, uma *capacidade* pode ser entendida como a habilidade de um sistema provê uma dada funcionalidade em termos de efeitos desejados, padrões, condições específicas e realização de tarefas.

A alta dinamicidade presente no domínio de IoT e de SoS faz com que os objetos e sistemas que os compõem não sejam, normalmente, conhecidos em tempo de *design*, sendo necessário a descoberta e composição em tempo de execução. O processo de descoberta é essencial para identificar a combinação de entidades (objetos ou sistemas) que atendam os objetivos desses dois tipos de sistemas.

Nesse contexto, este artigo apresenta uma proposta de Arquitetura que permite a descoberta de objetos e sistemas para os domínios de IoT e de SoS. A Seção 2 apresenta conceitos importantes em relação ao processo de Descoberta em IoT e SoS. A Seção 3 apresenta o Serviço de Descoberta proposto. Finalmente, a Seção 4 apresenta as conclusões e perspectivas de trabalhos futuros.

2. Processo de Descoberta em Internet das Coisas e em Sistemas-de-Sistemas

Em IoT, um *recurso* é definido como um dispositivo ou entidade que provê dados ou realiza atuação (e.g., um sensor ou um atuador), e um *serviço* é uma entidade de software que expõe as funcionalidades de um recurso [De et al. 2011]. Em IoT, *serviços de descoberta* são utilizados na identificação dos serviços adequados para a realização de uma tarefa em particular. Além disso, serviços de descoberta permitem a obtenção das informações necessárias para o uso desses serviços [Paganelli and Parlanti 2012]. O domínio de IoT consiste de uma miríade de recursos distribuídos e descentralizados que são providos e requeridos por diferentes usuários e organizações. Por isso, serviços de descoberta devem ser aplicados nesse contexto, permitindo que os serviços desejados sejam encontrados e que as informações necessárias sobre esses serviços sejam obtidas [Vermesan and Friess 2013].

Os serviços de descoberta em IoT precisam considerar que a capacidade de processamento e de energia dos recursos é limitada (e.g., tempo de carga da bateria) [Cassar et al. 2012], portanto, não pode haver processamento de requisições no próprio recurso. Também devem ser consideradas as questões de necessidade de escalabilidade e governança relacionadas com as diferentes políticas das organizações proprietárias dos recursos e fornecedoras dos serviços em IoT (e.g., os serviços providos por uma organização podem ser restritos a um certo grupo de usuários). Além disso, a natureza distribuída e heterogênea desses recursos impõe desafios de comunicação na construção de aplicações para esse domínio. A utilização de tecnologias semânticas é uma solução natural para promover interoperabilidade entre recursos, além disso, a utilização de anotações semânticas facilitam a descoberta de serviços uma vez que permitem a interpretação sem ambiguidade das funcionalidades e dos dados providos por esses serviços [Barnaghi et al. 2012]. Tal qual IoT, o domínio de SoS também enfrenta desafios de distribuição e descentralização, sendo recomendável o uso de semântica para prevenir ambiguidade de interpretação e garantir interoperabilidade entre os sistemas constituintes de um SoS.

Os serviços de descoberta para SoS precisam ser capazes de identificar a combinação apropriada de sistemas que contribuem para a realização da missão global do SoS, baseando-se nas capacidades desses sistemas. Desafios de escalabilidade e governança também estão presentes no domínio de SoS e devem ser considerados pelos serviços de descoberta. Além disso, devido ao ambiente dinâmico em que um SoS está normalmente inserido, os serviços de descoberta precisam estar preparados para mudanças, já que a missão global do SoS, o ambiente no qual o SoS opera e os sistemas constituintes que compõem o SoS podem sofrer alterações com o passar do tempo. Ademais, os sistemas constituintes podem se tornar indisponíveis ou outros sistemas mais adequados para a realização de uma tarefa podem ser adicionados ao seu ambiente, promovendo a necessidade de adaptação em um SoS.

3. Proposta de Arquitetura para um Serviço de Descoberta

A proposta de arquitetura apresentada nesta Seção é baseada em Sistemas Federados. Um *Sistema Federado* é uma coleção de sistemas computacionais independentes, cooperativos, possivelmente heterogêneos e autônomos que permitem a troca de parte ou de todos os seus dados [Sheth and Larson 1990]. Arquiteturas para Sistemas Federados são baseadas em *Federações de Registros*, compostas por uma coleção de registros autônomos, mas que cooperam entre si. Arquiteturas federadas espelham a estrutura de uma organização, agrupando módulos e aplicações independentes em domínios. Esses domínios são, em termos de administração e processamento, independentes, possuindo todas as capacidades necessárias para suportar suas próprias aplicações [Fernandez et al. 2003]. Essa divisão e independência de módulos, quando aplicado a IoT e a SoS, garantem escalabilidade e níveis de governança [Uckelmann et al. 2011]. Abordagens baseadas em Sistemas Federados também são usadas por garantir um certo nível de segurança e privacidade no contexto de IoT, além de interoperabilidade tanto em IoT como em SoS [Leo et al. 2014]. Em IoT, um recurso que expõe suas funcionalidades através de serviços se comunica por

interfaces. Em SoS, interfaces também são usadas para permitir que sistemas constituintes (SC) se comuniquem, permitindo a descoberta, seleção, composição e execução desses sistemas.

A Figura 1 apresenta a proposta de Arquitetura para Serviços de Descoberta para IoT e SoS. Essa arquitetura é dividida em quatro camadas, a saber: (i) Camada de Registros; (ii) Camada de Comunicação; (iii) Camada de Operação, e; (iv) Camada de Ontologia.

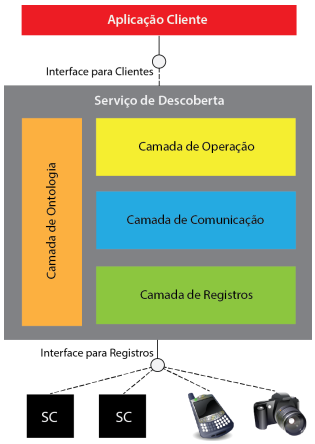


Figura 1. Arquitetura para Serviço de Descoberta

A *Camada de Registros* é composta por Diretórios e pelos Serviços de Diretórios que operam sobre esses Diretórios. As descrições dos serviços e sistemas, chamadas de *registros*, são armazenadas nesses *Diretórios*. Os hospedeiros desses serviços e sistemas devem enviar periodicamente uma mensagem de atualização de *status* para o Diretório onde estão cadastrados, confirmando sua disponibilidade.

A *Camada de Comunicação* é composta por uma rede P2P. Cada *peer* é um Serviço de Diretório que opera sobre um Diretório. Serviços de Diretório podem ser cadastradas em outros Serviços de Diretório de forma hierárquica ou de forma paralela. Adicionalmente, Serviços de Diretório podem ser agrupados em domínios de acordo com os modelos de descrição e ontologias de domínio usadas por esses Diretórios. Cada Serviço Diretório deve informar o Serviço Diretório Raiz sempre que um novo Serviço ou Sistema é adicionado ao seu Diretório. O Serviço Diretório Raiz é responsável por armazenar a *Extended Registries Ontology* (XTRO), que contém a descrição de todos os domínios e Diretórios cadastrados no Serviço de Descoberta.

Cada um dos Diretórios da Camada de Comunicação apresenta independência operacional, necessária tanto em IoT como em SoS, permitindo que diferentes políticas de governança sejam aplicadas pelo hospedeiro do Diretório. Além disso, a escalabilidade provida pelo uso de redes P2P e o agrupamento de Diretórios em domínios, otimizando o processo de descoberta e permitindo que diferentes modelos de descrição de serviços e ontologias sejam definidos, são características relevantes em IoT e em SoS.

A *Camada de Operação* é responsável por interagir com os Serviços Diretório, provendo a descoberta semântica de serviços e sistemas à aplicações cliente. A Camada de Operação se comunica com o Serviço Diretório Raiz para realizar essa descoberta.

A *Camada de Ontologia* é responsável por gerenciar as ontologias de domínio, permitindo a adição de novas ontologias de domínio para descrição de serviços e sistemas no XTRO. Ontologias de domínio são criadas baseadas nos conceitos e terminologias usados em um domínio específico e permitem a adição de semântica às descrições de serviços e sistemas. Os Diretórios deverão definir as suas próprias ontologias de domínio ou estender uma ontologia já cadastrada no XTRO. Uma vez que os atributos necessários para descrever um serviço são fundamentalmente diferentes dos atributos necessários para descrever um sistema, a Arquitetura em camadas proposta nesta seção permite que os Diretórios descrevam seus registros usando

diferentes modelos de descrição e ontologias de domínio.

Na proposta arquitetural apresentada neste trabalho, o estilo arquitetural REST (REpresentational State Transfer) [Fielding 2000] é adotado para permitir a comunicação com o Serviço de Descoberta através da criação de interfaces HTTP. O uso do estilo arquitetural REST permite que aplicações cliente possam solicitar a busca de registros ao Serviço de Descoberta através de mensagens HTTP e formatos de representação de dados - como o XML (*eXtensive Markup Language*) e o JSON (*Javascript Object Notation*) [JSON 2001] - sem a necessidade de nenhuma abstração adicional. Os serviços de IoT e sistemas constituintes que são registrados no Serviço de Descoberta também devem implementar interfaces HTTP para permitir a comunicação com o Serviço de Descoberta.

4. Conclusão e Trabalhos Futuros

Este artigo apresentou uma proposta de Arquitetura, baseada em Federações de Registros, para Serviços de Descoberta para os domínios de IoT e de SoS, permitindo que sejam usados vários Diretórios para armazenar as descrições dos serviços e sistemas. Essa distribuição garante escalabilidade e permite a independência desses Diretórios, que podem ser gerenciados por diferentes organizações com diferentes políticas para o cadastro e descoberta de registros. Como trabalho futuro, a implementação e avaliação de um Serviço de Descoberta usando essa Arquitetura deve ser realizada. Além disso, o teste e avaliação de ontologias para descrição semântica de serviços e sistemas e a descrição dos atributos necessários para permitir a descoberta dessas entidades no domínio de IoT e de SoS é necessário. Vale ressaltar que os atributos necessários para a descrição de serviços e sistemas são fundamentalmente diferentes e o Serviço de Descoberta deve permitir que essas entidades sejam descritas com diferentes atributos.

Referências

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

Barnaghi, P., Wang, W., Henson, C., and Taylor, K. (2012). Semantics for the internet of things: Early progress and back to the future. *Int. J. Semant. Web Inf. Syst.*, 8(1):1–21.

Cassar, G., Barnaghi, P., Wang, W., and Moessner, K. (2012). A hybrid semantic matchmaker for iot services. In *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications*, GREENCOM '12, pages 210–216, Washington, DC, USA. IEEE Computer Society.

De, S., Barnaghi, P., Bauer, M., and Meissner, S. (2011). Service modelling for the internet of things. In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, pages 949–955.

Fernandez, G., Zhao, L., and Wijegunaratne, I. (2003). Patterns for federated architecture. *Journal of Object Technology*, 2(1):135–149.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis.

JSON (2001). `http://www.json.org/`. Acessado em: 2015-07-19.

Leo, M., Battisti, F., Carli, M., and Neri, A. (2014). A federated architecture approach for internet of things security. In *Euro Med Telco Conference (EMTC), 2014*, pages 1–5.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Paganelli, F. and Parlanti, D. (2012). A dht-based discovery service for the internet of things. *Journal Comp. Netw. and Communic.*, 2012.

Sheth, A. P. and Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236.

Uckelmann, D., Harrison, M., and Michahelles, F. (2011). An architectural approach towards the future internet of things. In Uckelmann, D., Harrison, M., and Michahelles, F., editors, *Architecting the Internet of Things*, pages 1–24. Springer Berlin Heidelberg.

Vermesan, O. and Friess, P., editors (2013). *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers Series in Communication. River, Aalborg.

Um Framework para Funcionamento Offline em Aplicações Android

Jean Guerethes Fernandes Guedes[1], **Gibeon Soares de Aquino Júnior**[1]

[1]Departamento de Matemática e Informática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN)
Natal – RN – Brazil

guerethes@gmail.com, gibeon@dimap.ufrn.br

Abstract. *In the face of growing demand for creating mobile applications, driven by increasingly frequent use of smartphones and tablets by society, companies are facing difficulties in converting their traditional systems in applications for mobile computing. This process involves many challenges , in particular , the demand for connectivity and full remote data access to use mobile application on an environment that, due to its mobile characteristics, may not provide network access at all times. From the point of view of programming practices, this reflects in defining strategies for local data persistence, data replication and synchronization of data. On this reality, this work aims to develop a solution, through a framework, for providing features including, the provision of a object-relational mapping mechanism, replication and synchronization of data, as well as contemplating, removing, upgrading, and viewing if the mobile device is not connected, even if the mobile device without connectivity to the network.*

Resumo. *Diante da crescente demanda pela criação de aplicativos móveis, impulsionada pelo uso cada vez mais frequente de smartphones e tablets pela sociedade, as empresas vêm se deparando com um desafio de adaptar os seus tradicionais sistemas em aplicações para computação móvel. Esse processo envolve diversAs dificuldades, em particular, a demanda por conectividade e acesso a dados remotos durante o uso do aplicativo móvel em um ambiente que pela própria característica de mobilidade, pode não disponibilizar acesso à rede em todos os momentos. Do ponto de vista das práticas de programação e arquitetura, isso reflete em definir estratégias para persistência de dados localmente, replicação e sincronização de dados. Diante dessa realidade, esse trabalho tem como objetivo desenvolver uma solução, através de um framework, que apresente como principais funções o provimento de um mecanismo de persistência, replicação e sincronização dos dados, contemplando a criação, remoção, atualização e visualização dos dados persistidos ou requisitados, mesmo estando o dispositivo móvel sem conectividade com a rede.*

1. Introdução

Diante dos novos hábitos da sociedade em portar dispositivos móveis, empresas vêm enfrentando obstáculos na tentativa de adaptar seus sistemas e aplicações para o contexto da computação móvel. Uma das dificuldades recorrentes está em como lidar com a instabilidade do acesso à rede, na ausência de solução de replicação e sincronização de dados e, por fim, diante da não existência de um padrão de persistencia dos dados.

Um outro problema a ser abordado consiste na replicação de dados quando o dispositivo móvel se encontrar operando no modo offline. Foram identificadas algumas soluções fechadas e proprietárias para a replicação e sincronização de arquivos, tais como: [Smith 2013], [Microsoft] e [Drive 2015], mas estas foram consideradas inapropriadas para tratar dados, isso porque, dentre outros fatores, as soluções apresentaram apenas implementações voltadas para a replicação e sincronização de arquivos.

Para o desenvolvimento desse trabalho foi escolhida a plataforma Android, uma vez que esta possui licença open-source, por ser desenvolvida sobre a linguagem de programação java, que é uma das mais populares do mundo, e ainda pelo simples fato de existir uma maior diversidade de dispositivos móveis utilizando essa plataforma [Sowah and Fiawoo 2013].

O framework para o funcionamento offline das aplicações Android tem como intuito auxiliar o desenvolvimento de aplicações que necessitem se adaptar a inconstância do acesso a rede de forma mais rápida e com uma menor quantidade de código. Os aplicativos, de uma maneira geral, poderão realizar as operações sem a necessidade de conexão, apenas fazendo uso do banco de dados local.

O artigo está organizado da seguinte forma: a Seção 2 apresenta a arquitetura de software proposta para o framework. A Seção 3 destaca um estudo de caso de uso do Framework. Por fim, a Seção 4 apresenta as considerações finais.

2. Arquitetura do Framework

Diante da problemática descrita na introdução, foi necessária a construção de um framework que apresentasse as funções de persistência, replicação e sincronização dos dados, especialmente quando o dispositivo móvel se encontrar no modo offline, bem como a criação, remoção, atualização e visualização dos dados armazenados ou requisitando adotando com padrão a utilização do Web Service RESTful, como descrito por [Fielding 2000].

O framework foi desenvolvido baseado no uso de reflexão e na reutilização de padrões de software, com a possibilidade de se adaptar às necessidades específicas da aplicação que está sendo desenvolvida. Para que tal reutilização fosse possível o framework foi desenvolvido fazendo uso de padrões de projetos de criação, estruturais e comportamentais.

A arquitetura apresentada na Figura 1 é composta por três módulos: Comunicação com o servidor, Persistência, Sincronização. Os serviços da arquitetura são autônomos e cooperantes.

O framework conta com um banco de dados central, criado em sua instanciação, e é atualizado durante o uso do aplicativo. Os módulos Comunicação com o servidor é o módulo Persistência apresentam componentes flexíveis. Basta que seja implementada as interfaces que são responsáveis pelo funcionamento pelo comportamento do módulo. No caso do módulo Comunicação com o servidor a interfaces que é necessita realizar a implementação é a interface *WebService*, já no módulo Persistence a interfaces que deve ser implementada é a *IPersistenceManager*.

Para a inicialização do framework é possível ser realizada através da incialização padrão, ou seja, quando não foi extender a funcionalidade do framework para atender as

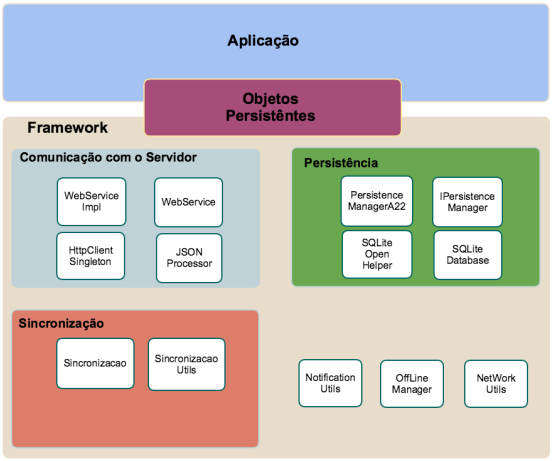


Figura 1. Arquitetura de Software

especificidades da aplicação que está sendo desenvolvida, ou pode ocorrer a inicialização passando algumas informações quando se fizer necessário extender as funcionalidades do framework, nesse caso a classe que está sendo estendida deverá ser informada no processo de inicialização, devendo essa classe ser uma extensão da interface *WebService* quando se fizer necessário estender funcionalidades de comunicação com o servidor ou da interface *IPersistenceManager* para quando o ponto de extensão for para o módulo de persistência.

3. Prova de Conceito

Essa seção tem como finalidade demonstrar a utilização do framework construído em vários casos reais. O framework foi utilizado nos aplicativos desenvolvidos na *SINFO* [1] são eles: o Coletor de Presença Mobile, UFRN Notify, *UFRN security* [2] *SIGAA 2.0* [3] e SigEventos.

Existe uma primeira versão do Coletor de Presença que está em uso desde 2012, sendo a sua implementação realizada sem o uso do framework desenvolvido. Por esse motivo o desenvolvedor precisou se ater a todas as situações de conectividade existentes, ou seja, foi necessário verificar, a cada requisição, o estado da conexão apresentada pelo dispositivo para selecionar a estratégia a ser realizada.

A primeira versão do coletor era subdivido em 10 pacotes, em que cada pacote era responsável por agrupar determinadas funcionalidade existentes no aplicativo, além de apresentar uma dependência de outros dois projetos para o seu funcionamento.

O projeto contabilizava um total de 30 classes Java e ainda 3427 linhas de código, para esse totalização não foi incluído as classes que fazem parte das Arquiteturas Mobile e da Arquitetura Android.

O novo projeto do aplicativo apresenta 3 pacotes, que corresponde na divisão das responsabilidades do aplicativo e ainda apresenta como um todo 4 classes java e apenas 767 linhas de código, que realizam o mesmo trabalho das 30 classes que foi necessário

[1] https://info.ufrn.br/html
[2] https://play.google.com/store/apps/details?id=ufrn.br.ufrnsecurity
[3] https://play.google.com/store/apps/details?id=br.ufrn.sinfo.sigaambeta

desenvolver para que a primeira versão apresentasse as funcionalidades almejadas.

A aplicação do Coletor de Presença Mobile sofreu substancial redução na sua quantidade de classes java, a redução foi um pouco superior a 77,6% na quantidade de linhas de código, entretanto, mesmo ocorrendo essa redução na quantidade de código java desenvolvido, confome mostrada na Tabela 1, apresentando as mesmas funcionalidades.

Tabela 1. Comparativo entre as versões do Coletor de Presença Mobile.

Versão	Qnt de Pacotes	Qnt de Classes	Qnt de Linhas de Código
Versão 1.0	10	30	3427
Versão 2.0	2	4	767

4. Considerações Finais

Com o intuito de colaborar para superar alguns óbices encontrados para a área do desenvolvimento de aplicações móveis, foi proposto um framework para o funcionamento offline de aplicações Android. O framework tem como intuito auxiliar o desenvolvimento de aplicações que demandam funcionamento offline de forma mais rápida e com uma menor quantidade de código. Dessa forma a complexidade foi reduzida bem como o custo para o desenvolvimento dos aplicativos, que, de uma maneira geral, poderá realizar as operações sem a necessidade de conexão, apenas fazendo uso do banco de dados local. Dessa forma o aplicativo pode ter uma maior abrangência, englobando tanto usuários que apresentam acesso contínuo a rede, como aqueles que não tem acesso em tempo integral.

Diante do estudo de caso apresentado, foi possível confirmar que o framework desenvolvido é adequado para solucionar os problemas listadas, pois ele já disponibiliza todas as funcionalidades de forma nativa, sem a necessidade de realizar nenhuma configuração adicional. O framework atende ainda a todos os requisitos funcionais (desempenho, confiabilidade) e aos requisitos não funcionais (criação do banco, sincronização dos dados e comunicação com o servidor), apresentando como seu ponto forte a fácil configuração e utilização, pois o mesmo não necessita de nenhuma configuração adicional, a não ser a utilização de anotações nas classes e da inicialização do framework na classe principal.

Referências

Drive, G. (2015). Google drive rest api overview.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine.

Microsoft. Develop with the onedrive api.

Smith, B. (2013). Introducing the dropbox sync api for mobile developers.

Sowah, R. A. and Fiawoo, S. Y. (2013). Design and development of a web service for android applications for extensive data processing. In *International Journal of Engineering Research Technology*.

Uma Estratégia Local e Composicional para a Verificação de Livelock em Sistemas Baseados em Componentes

M. S. Conserva Filho[1], **M. V. M. Oliveira**[1], **A. Sampaio**[2]

[1] Universidade Federal do Rio Grande do Norte (UFRN) - Natal, RN - Brasil

[2]Universidade Federal de Pernambuco (UFPE)– Recife, PE – Brasil

madiel@ppgsc.ufrn.br,marcel@dimap.ufrn.br,acas@cin.ufpe.br

Abstract. *Component-based development emerged as a paradigm to deal with the increasing complexity of software, improving reusability, maintenance and reliability of the components. However, in order to ensure trustworthy systems, a formal verification must be carefully performed when components are being integrated, especially in critical applications, during which classical problems can arise, such as livelock. In this paper, we propose a systematic approach to build livelock-free systems in BRIC, which is a component model that imposes some constraints on the components and how they interact with each other.*

Resumo. *O desenvolvimento baseado em componentes surgiu como uma técnica para lidar com a crescente complexidade dos softwares, facilitando a reutilização, manutenção e confiabilidade dos componentes. Entretanto, para garantir sistemas confiavéis, é necessário uma verificação formal durante o processo de integração dos componentes, principalmente em aplicações críticas, durante o qual podem surgir problemas clássicos, como livelock. Neste artigo, propomos uma abordagem composicional para possibilitar sistemas livres de livelock em BRIC, que é um modelo de componentes que impõe algumas restrições sobre os componentes e na integração entre eles.*

1. Introdução

Desenvolvimento de software baseado em componentes (DSBC) surgiu como uma promissora abordagem para lidar com a crescente complexidade e evolução das aplicações computacionais. A principal característica desta abordagem é desenvolver sistemas através da integração de partes independentes existentes, chamadas componentes de software. Além disso, esta técnica enfatiza a reutilização de componentes de software confiáveis, a fim de desenvolver sistemas de alta qualidade.

Com o objetivo de garantir confiabilidade no DSBC, uma questão fundamental é a composição dos componentes. Problemas podem surgir quando dois ou mais componentes são integrados pela primeira vez. Este problema é ainda mais relevante quando um grupo de componentes trabalham em conjunto com a finalidade de executar determinadas tarefas. Por estas razões, a integração dos componentes precisa ser cuidadosamente projetada, sistematizada e verificada; apenas isto pode garantir sistemas confiáveis. Portanto, para assegurar o sucesso do DSBC, é essencial realizar uma completa verificação formal durante a integração dos componentes, especialmente em aplicações críticas.

Em [Ramos 2011] é descrita uma estratégia que propõe a composição sistemática e segura de componentes baseada em um conjunto de regras de composição. Estas regras estão divididas em dois grupos: binárias (conecta canais de dois componentes distintos) e unárias (conecta dois canais distintos do mesmo componente). A aplicação sistemática destas regras possibilita desenvolver sistemas seguros, assegurando, por construção, a ausência de deadlock. Toda a abordagem é apoiada pela álgebra de processos CSP [Hoare 1985],[Roscoe 2010], que tem como foco a verificação de sistemas concorrentes. Esse trabalho também propõe um modelo de componentes, $\mathcal{BRIC}$, que impõe algumas restrições sobre os componentes e na forma de integração entre eles. Em $\mathcal{BRIC}$, cada componente é representado por um contrato, cujo comportamento é representado por um processo CSP.

A abordagem tradicional para provar que um sistema é livre de livelock realiza uma análise global a fim de verificar se algum estado com livelock não pode ser alcançado [Roscoe 2010]. Entretanto, esta verificação requer uma complexa análise e pode se tornar inviável para grandes sistemas. Esta estratégia é automatizada por FDR [Ltd 2012]. Uma alternativa consiste em realizar uma análise estática, implementada por SLAP [Ouaknine et al. 2013], o que possibilita uma verificação baseada apenas na sintaxe do processo. Além disso, ainda é possível realizar uma análise local, que permite avaliar apenas algumas partes do sistema, reduzindo o tempo e o esforço nesta verificação.

A principal contribuição deste trabalho é o desenvolvimento de uma estratégia composicional para o desenvolvimento de sistemas livres de livelock em $\mathcal{BRIC}$. A fim de alcançar este objetivo, propomos um conjunto de novas condições que devem ser adicionadas às originais regras de composição. Através destas novas condições, a ausência de livelock é garantida durante a composição dos componentes. Diante deste contexto, uma característica distinta da estratégia é que realizamos uma análise de livelock local, em vez da análise global. Além disso, as regras suportam um desenvolvimento sistemático capaz de identificar previamente a possibilidade de livelock no sistema.

2. Estratégia Local e Composicional para Livelock em $\mathcal{BRIC}$

Em [Ramos 2011], as regras de composição não introduzem livelock no comportamento do processo resultante porque os canais utilizados na composição não são escondidos do ambiente, mas apenas removidos do conjunto de canais de comunicação, evitando novas composições nesses canais. Com isto, as composições representam o estilo de composição caixa-cinza. No entanto, neste trabalho é realizado um estilo de composição caixa preta, em que se analisa o comportamento do componente resultante ao esconder os canais de comunicação que não estejam no conjunto de canais visíveis. Portanto, as composições agora podem introduzir livelock mesmo considerando que os componentes utilizados são livres de livelock. Por exemplo, o processo CSP $P = (a \rightarrow P) \setminus \{a\}$ executa infinitamente o evento interno a. Com isso, o processo P realiza um loop infinito de ações internas, introduzindo um livelock.

Neste contexto, o raciocínio sobre livelock está diretamente relacionado aos comportamentos infinitos de um processo. Portanto, o primeiro passo da estratégia consiste em identificar os comportamentos infinitos de um determinado componente. Isto significa identificar as sequências de eventos (traces) que levam o processo ao seu estado inicial. O conjunto dos mínimos padrões de interação (*MIP*) contém todas as sequências que levam o

processo para a sua primeira recursão. Por exemplo, vamos observar os seguintes processos CSP: $B_1 = a \rightarrow b \rightarrow B_1 \,\square\, n.1 \rightarrow n.2 \rightarrow B_1$, e $B_2 = c \rightarrow d \rightarrow B_2 \,\square\, s.1 \rightarrow B_2$, os quais representam os comportamentos dos contratos Ctr_1 e Ctr_2, respectivamente. Neste caso, os *MIP* são: $MIP(Ctr_1) = \{\langle a, b\rangle, \langle n.1, n.2\rangle\}$ e $MIP(Ctr_2) = \{\langle c, d\rangle, \langle s.1\rangle\}$.

Após a identificação destas sequências, é possível calcular quais canais podem ser utilizados na composição sem introduzir livelock apenas com base nos *MIP* de um contrato. Com base no exemplo anterior, o canal a de Ctr_1 pode ser utilizado em uma composição porque após remover o canal a dos $MIP(Ctr_1)$, os traces resultantes, $\{\langle b\rangle, \langle n.1, n.2\rangle\}$, não são vazios. O mesmo pode ser aplicado para o canal b de Ctr_1 e para os canais c e d de Ctr_2. Por esta razão, podemos concluir que os canais que não introduzem livelock de Ctr_1 e Ctr_2 são $\{a, b\}$ e $\{c, d\}$, respectivamente. Por outro lado, apenas com base nesta análise, não podemos usar o canal n de Ctr_1 pois após remover este canal o trace irá ficar vazio. O mesmo se aplica ao canal s de Ctr_2. Para estes casos, a verificação requer uma análise mais complexa no comportamento dos contratos para garantir uma composição livre de livelock.

A fim de garantir composições livre de livelock em $\mathcal{BRIC}$, propomos algumas condições adicionais que devem ser incorporadas às regras de composição. Essas novas condições estão classificadas em dois grupos: *Composições Seguras Simples* e *Composições Seguras Complexas*. A primeira representa a condição mais simples e requer apenas verificações básicas nos *MIP* dos componentes, como descrevemos no exemplo anterior.

O segundo grupo, entretanto, é responsável pelos casos que podem introduzir livelock. Neste contexto, garantir composições livres de livelock requer uma verificação complexa nos comportamentos dos contratos. Intuitivamente, devemos analisar se existe uma sequência formada pela concatenação dos *MIP* dos contratos envolvidos na composição. Esta sequência representa um possível comportamento da composição resultando em um loop infinito de ações internas. Caso esta sequência exista, a sincronização destes canais é viável e, consequentemente, livelock pode ser introduzido. Deste modo, uma composição complexa apenas é realizada caso não exista nenhuma sequência em comum. Portanto, com a inclusão destas novas condições, podemos garantir composições livres de livelock em $\mathcal{BRIC}$.

3. Estudo de Caso

Com a finalidade de demonstrar a eficiência da nossa estratégia, desenvolvemos uma análise comparativa entre as ferramentas que realizam a verificação de livelock. O estudo de caso utilizado é uma variação do jantar dos filósofos, que é um problema clássico de um sistema concorrente. Na Tabela 1, N representa o número de filósofos, # o número de composições executadas, FDR, SLAP e $\mathcal{BRIC}$ representam o tempo[1] da análise de livelock em FDR, SLAP (BDD e SAT), e da nossa análise local de livelock em $\mathcal{BRIC}$[2].

De acordo com a Tabela 1, podemos observar que a análise local foi capaz de executar todas as verificações de forma consideravelmente mais eficiente. Por exemplo, SLAP ultrapassou o tempo limite de 1 hora com 100 (ou mais) filósofos. Este tempo limite foi atingido mais cedo por FDR (5 filósofos). Por outro lado, a nossa abordagem foi capaz

[1] Tempo em segundos

[2] * Ultrapassou o tempo limite de 1 hora

N	#	FDR	SLAP (BDD)	SLAP (SAT)	BRIC
3	10	2.884	0.352	2.114	0.219
5	14	*	2.715	14.135	0.248
10	38	*	51.708	399.807	0.303
100	398	*	*	*	0.778
1.000	3.998	*	*	*	3.888
10.000	39.998	*	*	*	206.689

Table 1. Resultado do Estudo de Caso

de verificar um sistema composto por 10.000 filósofos e 10.000 garfos (20.000 processos e 39.998 composições) em menos de 4 minutos.

4. Conclusão

Neste artigo, apresentamos uma estratégia sistemática para garantir sistemas livres de livelock em $\mathcal{BRIC}$. A verificação tradicional de livelock realiza uma análise global. Para sistemas complexos, esta verificação usando FDR e SLAP pode facilmente se tornar inviável. Por outro lado, a verificação composicional e local de livelock se mostra promissora. O estudo de caso utilizado pode ser considerado como um bom teste para verificar o desempenho e o tempo de resposta da nossa estratégia. De acordo com o experimento realizado, a análise local demonstrou-se muito eficiente na verificação de livelock para o jantar dos filósofos com 10.000 filósofos e 10.000 garfos. Esta análise local de livelock ainda apresenta algumas limitações, por exemplo, parâmetros, recursão mútua e processos guardados, sendo deixados como trabalho futuro, o que também inclui outros estudos de caso.

References

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Ltd, F. S. (2012). *FDR2: User Manual, version 2.94*. Formal Systems Ltd.

Ouaknine, J., Palikareva, H., Roscoe, A. W., and Worrell, J. (2013). A static analysis framework for livelock freedom in CSP. *Logical Methods in Computer Science*, 9(3).

Ramos, R. T. (2011). *Systematic Development of Trustworthy Component-based Systems*. PhD thesis, Center of Informatics - Federal University of Pernambuco, Brazil.

Roscoe, A. (2010). *Understanding Concurrent Systems*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition.

Verificação de programas *Circus* executáveis usando Java Pathfinder

S.L.M. Barrocas[1] and Marcel Oliveira[1]

[1]Departamento de Informática e Matemática Aplicada (DIMAp) – Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal 1524 – 59078-900 – Rio Grande do Norte – RN – Brasil

samuellincoln@gmail.com, marcel@dimap.ufrn.br

Abstract. *This paper proposes a strategy for model checking executable programs generated from specifications (on the Circus notation) through JCircus, a formal translator from Circus to Java. In this paper, we focus on refinement model checking; thus, our strategy checks if the generated code refines its source specification.*

Resumo. *Este artigo propõe uma estratégia para verificar programas executáveis gerados a partir de especificações (na notação Circus) através de JCircus, um tradutor formal de Circus para Java. Neste artigo, nós focamos na verificação de refinamento; assim, nossa estratégia checa se o código gerado refina a sua especificação-fonte.*

1. Introdução

Métodos formais são técnicas para a especificação de sistemas usando notações formais sustentadas por uma semântica rigorosamente definida. Um esforço para minimizar os custos da aplicação de técnicas formais é a implementação de tradutores automáticos de notações formais para linguagens de programação. Entre as notações formais está Circus [Woodcock e Cavalcanti 2001], uma linguagem de especificação cuja sintaxe combina as sintaxes de Z [Woodcock e Davies 1996, Spivey 1992], CSP [Roscoe 1998] e de comandos guardados de Dijkstra [Dijkstra 1975]. Circus foi fornecida com uma semântica denotacional [Oliveira 2006], e também uma semântica operacional [Freitas 2005] que contém regras que podem ser aplicadas para gerar sistemas de transições rotuladas com predicado (LPTS) para uma determinada especificação.

JCircus [Barrocas 2011, Freitas 2005] é um tradutor automático para Circus que traduz especificações em Circus para código Java. O código de saída de JCircus usa a biblioteca JCSP [Welch et al 2010], que fornece abstrações para implementar algumas construções de CSP, tais como comunicação, multi-sincronização, paralelismo e escolha externa.

Como os tradutores automáticos são susceptíveis a erros de execução, vê-se necessária uma estratégia para aumentar a confiabilidade do código gerado. Assim, uma abordagem interessante é o uso de verificação de software. Neste artigo, nós descrevemos a implementação de uma estratégia para validar JCircus verificando a corretude do código que é gerado por JCircus. A estratégia inclui:

JCircus - Gerar código Java que implementa a especificação usando JCircus (será chamado de “código gerado por JCircus” ao longo deste artigo);

LPTS - Gerar o LPTS da especificação, que representa a semântica da especificação;

JPF - Gerar um modelo do LPTS em Java usando Java Pathfinder (JPF) [Visser et al 2003], que interage com o código gerado por JCircus para checar se ele implementa corretamente a especificação;

EclEmma - Aplicar a estratégia para parte de um banco de entradas (especificações de entrada) em Circus que assegurem cobertura de decisão (usando o EclEmma [EclEmma 2005], um medidor de cobertura de código) do código-fonte de JCircus;

O desenvolvimento desta estratégia exigiu a implementação de: 1) um gerador de LPTS, 2) um gerador de modelo em JPF, e 3) um gerenciador de banco de testes que assegura cobertura de Decisão da implementação em JCircus.

2. Conceitos básicos

Circus é uma linguagem formal que combina Z, CSP e a linguagem de comandos guardados de Dijkstra. Uma especificação em Circus é definida em termos de parágrafos. Cada um destes parágrafos pode ser um parágrafo Z, uma definição de canais, uma definição de conjuntos de canais, ou uma declaração de processo. Escrever especificações em Circus significa especificar processos. A declaração de um processo é composta pelo seu nome (possivelmente seguido de parâmetros) e a sua definição. Um processo pode ser explicitamente definido, ou deve ser definido em termos de outros processos. Quando um processo é explicitamente definido, além da definição do estado (usando a notação Z) e da ação principal, nós temos no seu corpo operações em Z e definições de ações (parametrizadas); elas são usadas para especificar a ação principal do processo. Uma ação pode ser uma expressão de esquema, um comando guardado, uma invocação para uma ação previamente definida, ou uma combinação destes construtores usando operadores CSP.

JCircus é uma ferramenta que traduz especificações em Circus para código Java. O código gerado usa a API JCSP [Welch et al 2000 e 2010], que implementa algumas construções da linguagem CSP, como paralelismo e multi-sincronização. JCircus é baseada em regras de transformação de uma estratégia de tradução proposta em [Oliveira 2006, Freitas 2005].

Java Pathfinder (JPF) começou como um verificador de software. Entretanto, ele tem várias extensões e modos de execução diferentes. Todos estes modos e extensões são comumente usados para verificar programas em Java. No nosso trabalho, nós focamos na característica original de JPF, que é a verificação clássica de software.

3. Estratégia de verificação do código executável em Circus

Se nós queremos provar que o comportamento de uma especificação Circus em Java (gerada por JCircus) é correto, é necessário checar se a especificação é refinada pelo código gerado por JCircus. Isto é, é necessário criar e implementar uma estratégia para automaticamente testar (checar a aceitação ou a recusa de um evento) todas as combinações de seqüências de eventos e checar se elas são aceitas ou recusadas corretamente. Para isto, devemos obter a semântica da especificação gerando o LPTS da especificação e depois, a partir do LPTS, gerar um modelo (em Java Pathfinder) que conduz os testes nos eventos.

A estratégia: Descrevemos a estratégia de conversão do LPTS para o modelo em JPF:

Pegue todos os caminhos do código gerado por JCircus que são ramos de uma escolha randômica (estes são gerados por ações de escolha interna e processos). Para cada caminho do código gerado por JCircus, proceda como segue:

- Para cada nó no LPTS, pegue todas as transições nas quais aquele nó é o nó-fonte. Se a transição não for silenciosa, então teste todos os eventos. Se o evento testado é o mesmo do arco e a restrição (cada nó tem uma restrição indicando se ele corresponde a um caminho alcançável ou não) do nó destino é verdadeira, então aquele evento entra na zona de expectativa de aceitação de eventos. Senão, se o evento testado tiver uma transição para ele com restrição verdadeira, ignore-o. Se as condições anteriores não forem satisfeitas, o evento entra na zona de expectativa de rejeição de eventos. Se o evento é recusado na zona de aceitação ou aceito na zona de recusa, então o refinamento falha para aquele caminho. Do contrário, ele é bem sucedido se o refinamento do nó-destino da transição analisada for bem sucedido também;

- Se a transição for silenciosa, não teste nenhum evento. Apenas cheque a restrição do nó-destino. Se ela evoluir para verdadeiro, então o sucesso do refinamento depende do sucesso do refinamento para o nó-destino da transição analisada. Caso contrário, o refinamento falha; o refinamento é bem sucedido apenas se ele for bem sucedido para todos os caminhos do código gerado por JCircus. Para cada caminho do código gerado por JCircus, o refinamento é bem sucedido apenas se ele for bem sucedido para o nó inicial do LPTS da especificação. Para o nó inicial, o refinamento é bem sucedido apenas se ele for bem sucedido para todos os nós-destino das transições não-silenciosas ou para ao menos um nó-destino de uma transição, e assim por diante;

Resultados: A estratégia descrita neste trabalho visa validar a ferramenta JCircus, isto é, provar que o código gerado por JCircus refina a especificação de entrada, seja ela qual for. A validação é dada pela aplicação da estratégia descrita neste artigo (de verificação de refinamento do código gerado por JCircus) para um conjunto de entradas que garanta alta cobertura de decisão do código-fonte de JCircus. Construímos um conjunto com 67 entradas, e este conjunto, quando executado junto com o EclEmma (um medidor de cobertura de código), garante mais de 95% de cobertura de decisão do código-fonte de JCircus. Até o presente momento, nós provamos o refinamento para o código gerado por JCircus para 17 especificações de um conjunto de entradas formado por 67 especificações. O objetivo final é aplicar a técnica descrita neste artigo para o conjunto completo, e as especificações restantes ainda estão tendo os seus refinamentos verificados.

4. Conclusões e trabalhos futuros

Neste trabalho, nós criamos e implementamos uma abordagem para verificar programas executáveis em Circus. Nós usamos uma ferramenta, chamada JCircus, para gerar o código Java executável de Circus e aplicar a técnica de verificação que nós implementamos. O processo de verificação envolveu a geração do LPTS da especificação e a geração de um modelo em JPF que conduz o código gerado por JCircus para a especificação. Além disso, validamos 17 especificações das 67 que cobrem o código.

Como trabalho futuro podemos citar a validação das especificações restantes (há

50 especificações pela frente) e a implementação completa do gerador de LPTS (as categorias sintáticas não traduzíveis por JCircus ainda não foram implementadas, bem como as de processos compostos). Outro trabalho futuro é a implementação de verificação para programas Circus executáveis recursivos (ainda não verificáveis, no momento). A estratégia para verificar programas recursivos possivelmente se dará por execução simbólica com JPF.

Referências

S. L. M. Barrocas. (2011) “JCircus 2.0: Uma extensão da Ferramenta de Tradução de Circus para Java”. Dissertação de mestrado, Programa de Pós-Graduação em Sistemas e Computação - Universidade Federal do Rio Grande do Norte.

E. W. Dijkstra. (1975) “Guarded commands, nondeterminacy and the formal derivation of programs”. Communication of the ACM, 18 (18):453–457.

A. Freitas. (2005) “From Circus to Java: Implementation and Verification of a Translation Strategy”. Master’s thesis, Department of Computer Science, The University of York.

L. Freitas. (2005) “Model-checking Circus”. PhD thesis, Department of Computer Science, The University of York, 2005. YCST-2005/11.

M. V. M. Oliveira. (2006) “Formal Derivation of State-Rich Reactive Programs using Circus”. PhD thesis, Department of Computer Science, University of York.

A. W. Roscoe. (1998) "The Theory and Practice of Concurrency". Prentice-Hall Series in Computer Science. Prentice-Hall.

J. M. Spivey. (1992) "The Z Notation: A Reference Manual". Prentice-Hall, 2nd edition.

W. Visser, K. Havelund, G. Brat, S. Park, and F. Lerda. (2003) "Model checking programs". Automated Software Engg, 10(2):203–232.

P. Welch, N. Brown, J. Moores, K. Chalmers, and B. Sputh. (2010) Alting barriers: synchronisation with choice in Java using JCSP. Concurr. Comput. : Pract. Exper., 22(8):1049–1062.

P. H. Welch. (2000) Process oriented design for Java: concurrency for all. In H. R. Arabnia, editor, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, pages 51–57. CSREA Press.

J. C. P. Woodcock and A. L. C. Cavalcanti. (2001) A concurrent language for refinement. In A. Butterfield and C. Pahl, editors, IWFM’01: 5th Irish Workshop in Formal Methods, BCS Electronic Workshops in Computing, Dublin, Ireland.

J. C. P. Woodcock and J. Davies. (1996) Using Z—Specification, Refinement, and Proof. Prentice-Hall.

EclEmma. (2005) Disponível em: <http://www.eclemma.org/>. Acesso em 16/07/2015.

Assistente para Verificação de Programas em Frama-C

Vitor A. Almeida[1]

[1] Departamento de Matemática e Informática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN) – Natal, RN – Brasil

vitoralcantara@ppgsc.ufrn.br

Abstract. *This article describes the early stage of an extension to Frama-C, WP plug-in. This extension enables a manual help to automated theorem provers by manipulating generated proof obligations. This paper proposes (yet another) tool to prove program correctness.*

Resumo. *Este artigo descreve a fase inicial de uma extensão para Frama-C, o plug-in WP. Esta extensão permite ajudar manualmente provadores automáticos de teoremas, através da manipulação das obrigações de prova geradas. Dando, ainda, uma nova opção para provar a corretude de programas.*

1. Introdução

O Frama-C [Cuoq et al. 2012] é uma plataforma de construção de analisadores para a linguagem C, que auxilia na verificação formal de sistemas baseados nesta linguagem. Ele possui um conjunto de *plug-ins* que, junto com os mesmos, provê análise estática e dedução de semântica do código C. A ferramenta foi implementada na linguagem OCaml.

Tanto funções quanto projetos completos em C podem ser especificados pelo Frama-C. Para tal, deve ser utilizada a linguagem ACSL [Baudin et al. 2015b], derivada da JML [Leavens et al. 1999], que permite adicionar anotações lógicas no código fonte. Estas anotações lógicas são usadas por vários *plug-ins* em suas análises.

Um dos *plug-ins* do Frama-C, o WP [Baudin et al. 2015a], permite que o Frama-C realize verificação dedutiva. Seu nome é uma sigla para *Weakest Precondition*, pois o WP se utiliza da técnica de pre-condição mais fraca para provar as propriedades do sistema, uma técnica baseada a partir de trabalhos de Hoare, Floyd e Dijkstra sobre corretude de programas [Cuoq et al. 2012]. Para provar as propriedades de um programa, o WP analisa o código C e as anotações ACSL correspondentes, gerando Obrigações de Provas (chamadas a partir deste ponto de OPs) para averiguar se a execução do sistema efetivamente seguirá à risca as condições exigidas pela anotação.

A lógica nas fórmulas das OPs é de primeira ordem/polissortida (*multi-sorted*) e cada OP pode assumir uma de três estruturas possíveis. Em duas destas estruturas a OP possui um conjunto de hipóteses e um objetivo, enquanto na terceira há somente um objetivo. E cada OP pode ser provada diretamente pelo WP, usando reescrita de fórmulas e tentativas de provas triviais, através de provadores SMT, como o Alt-Ergo [Bobot et al. 2008], assistentes de provas, como o Coq [Bertot and Castéran 2004] e gerenciadores de provas, como o Why3 [Filliatre and Paskevich 2013].

A forma que custa menos tempo ao usuário para se provar uma OP é através dos provadores automáticos. Porém, estes podem não ter sucesso, sendo duas as razões

possíveis: as anotações não condizem com a implementação; ou a OP é bastante complexa para ser resolvida nas restrições de tempo e espaço definidas pelo usuário.

A solução para o primeiro caso é a verificação e correção manual das anotações e/ou do código, uma vez que provadores automáticos não são capazes de tratar esse problema. Já na segunda situação, a opção para o usuário é modificar manualmente as OPs através de um assistente dedicado. Esse último caso pode ser complexo pois o usuário irá encontrar-se num contexto produzido automaticamente e, geralmente, com uso de ferramentas que trabalham em uma lógica de ordem superior. Desta forma, exige-se experiência do usuário, tanto no manuseio, quanto na estratégia de prova de OPs.

2. Extensão do WP

Propõe-se então, como auxílio ao usuário, uma extensão do WP que permita modificar as OPs antes de enviá-las aos provadores automáticos. As modificações consistem em aplicações de regras de reescrita, táticas e regras de inferência, fornecendo diferentes estratégias para que os provadores automáticos tenha uma maior possibilidade de sucesso.

A implementação atual contém 15 opções de modificação. Para exemplificar, duas modificações implementadas foram a remoção de quantificadores existenciais e universais em fórmulas e prova de OPs por casos. Estas modificações foram adaptadas de regras disponíveis tanto em Coq como também em Atelier-B [Mentré et al. 2012] e Rodin [Abrial et al. 2010], plataformas de especificação formal de sistemas em, respectivamente, B e Event-B.

Realizamos toda a implementação modificando o código original do WP presente na distribuição *Sodium* do Frama-C[1]. A opção de alterar diretamente o próprio WP foi escolhida pela necessidade de acessar as estruturas de dados internas do *plug-in*.

3. Análise experimental

A extensão se encontra na fase de desenvolvimento. Desta forma, ainda não foram realizados testes finais para provar a eficácia da ferramenta, mas uma metodologia para sua análise já foi definida.

Um conjunto de arquivos-fontes em C, com anotações em ACSL, foi escolhido para testes. Estes arquivos foram selecionados a partir da distribuição Sodium do Frama-C, dos exemplos no tutorial “ACSL By Example: Towards a Verified C Standard Library" [Burghardt and Gerlach 2015] e do sítio do Toccata[2]. Os passos para a análise experimental são:

1. Tentar provar as funções sem modificar as OPs, isto é, sem usar a extensão do WP, removendo os arquivos no qual todas as OPs foram provadas automaticamente.
2. Tentar provar as OPs restantes com a aplicação das táticas da extensão, separando as OPs que foram validadas somente com o uso da extensão.

Atualmente foram separados 34 arquivos-fontes com OPs não provadas automaticamente. À medida que os testes estiverem sendo executados, será feita uma análise da necessidade de inserir novas táticas para poder provar as OPs, tornando a realização de testes uma etapa paralela ao desenvolvimento.

[1] `http://frama-c.com/download.html`

[2] `http://toccata.lri.fr/gallery/jessieplugin.en.html`

4. Conclusão e trabalhos futuros

A próxima etapa será introduzir novas regras de acordo com a realização de testes como descrito na seção anterior. Será aprimorada, conjuntamente, a interface gráfica, permitindo o uso para avaliação e testes por outros usuários.

Uma outra funcionalidade, também planejada, é a obtenção de contra-exemplos nos casos em que os provadores refutem uma OP. As atuais alternativas de provadores no Frama-C não disponibilizam muitas informações além do resultado, sendo esta uma forma de permitir uma maior comunicação com os provadores.

Constatando-se a eficácia da extensão do Frama-C, planejamos discutir com os desenvolvedores do Frama-C e do WP a possibilidade de embutir integralmente esta extensão nas próximas distribuições do programa e do *plug-in*, com o intuito de agilizar a especificação formal, bem como diminuir custos no desenvolvimento de sistemas em C.

Referências

Abrial, J., Butler, M. J., Hallerstede, S., Hoang, T. S., Mehta, F., and Voisin, L. (2010). Rodin: an open toolset for modelling and reasoning in Event-B. *STTT*, 12(6):447–466.

Baudin, P., Bobot, F., , Correnson, L., and Dargaye, Z. (2015a). *WP Plug-in Manual*.

Baudin, P., Filliatre, J. C., Cuoq, P., Marché, C., Monate, B., Moy, Y., and Prevosto, V. (2015b). *ACSL: ANSI C Specification Language*.

Bertot, Y. and Castéran, P. (2004). *Interactive theorem proving and program development : Coq'Art : the calculus of inductive constructions*. Texts in theoretical computer science. Springer, Berlin, New York.

Bobot, F., Conchon, S., Contejean, E., and Lescuyer, S. (2008). Implementing Polymorphism in SMT Solvers. In *Proceedings of the Joint Workshops of the 6th International Workshop on Satisfiability Modulo Theories and 1st International Workshop on Bit-Precise Reasoning*, SMT '08/BPR '08, pages 1–5, New York, NY, USA. ACM.

Burghardt, J. and Gerlach, J. (2015). ACSL By Example: Towards a Verified C Standard Library. Acessado em: 14 maio 2015.

Cuoq, P., Kirchner, F., Kosmatov, N., Prevosto, V., Signoles, J., and Yakobowski, B. (2012). Frama-C: A Software Analysis Perspective. In *Proceedings of the 10th International Conference on Software Engineering and Formal Methods*, SEFM'12, pages 233–247, Berlin, Heidelberg. Springer-Verlag.

Filliatre, J.-C. and Paskevich, A. (2013). Why3 - Where Programs Meet Provers. In Felleisen, M. and Gardner, P., editors, *Programming Languages and Systems*, volume 7792 of *Lecture Notes in Computer Science*, pages 125–128. Springer Berlin Heidelberg.

Leavens, G., Baker, A., and Ruby, C. (1999). JML: A Notation for Detailed Design. In Kilov, H., Rumpe, B., and Simmonds, I., editors, *Behavioral Specifications of Businesses and Systems*, volume 523 of *The Springer International Series in Engineering and Computer Science*, pages 175–188. Springer US.

Mentré, D., Marché, C., Filliâtre, J., and Asuka, M. (2012). Discharging Proof Obligations from Atelier-B Using Multiple Automated Provers. In *Abstract State Machines, Alloy, B, VDM, and Z - Third International Conference, ABZ 2012, Pisa, Italy, June 18-21, 2012. Proceedings*, pages 238–251.

Negações, T-normas e T-conormas Difusas Hesitantes Típicas utilizando a Ordem Parcial de Xu-Xia

Hélida S. Santos, Benjamín C. Bedregal, Regivan N. Santiago[1]

[1]Departamento de Informática e Matemática Aplicada
Grupo de Lógica, Linguagens, Informação, Teoria e Aplicações (LoLITA)
Universidade Federal do Rio Grande do Norte (UFRN)
59.072-970 – Natal – RN – Brasil

{helida,bedregal,regivan}@dimap.ufrn.br

Abstract. *The theory of Fuzzy Sets has been applied to many fields to handle uncertainty, which can be reflected on the membership degree of the objects belonging to a set. Moreover, Hesitant Fuzzy Sets are useful whenever there is indecisiveness among several possible values for the preferences over objects in the process of decision-making. In this sense, the aim of this work is to consider the notion of aggregation functions using Xu-Xia partial ordering for typical hesitant fuzzy elements and study negations, t-norms and t-conorms and some of the properties usually demanded to such operators.*

Resumo. *A teoria dos conjuntos difusos têm sido aplicada em diversas áreas com o intuito de lidar com incertezas. Tais incertezas podem ser refletidas no grau de pertinência de objetos a um conjunto. Uma das extensões dessa teoria é a dos conjuntos difusos hesitantes que pode ser utilizada quando há indecisão entre vários possíveis valores acerca da preferência de um objeto sobre outro no processo de tomada de decisão. Neste sentido, o objetivo deste trabalho é considerar a noção de funções de agregação utilizando a ordem parcial de Xu-Xia para os elementos difusos hesitantes típicos além de estudar as negações, t-normas e t-conormas, e algumas das propriedades usualmente exigidas para tais operadores.*

1. Introdução

A teoria dos conjuntos difusos (fuzzy) apresentada em [Zadeh 1965] permitiu o surgimento de muitas outras extensões que lidassem com o problema de retratar incertezas do mundo real. Existem situações nas quais tais incertezas são difíceis de serem manipuladas porque elas podem surgir de diferentes fontes simultaneamente. Neste sentido, a literatura está repleta de propostas que tentam superar tal adversidade como os conjuntos difusos hesitantes (Hesitant Fuzzy Sets – HFSs) apresentados em [Torra and Narukawa 2009, Torra 2010] como uma nova extensão dos conjuntos difusos, onde o grau de pertinência é definido como um subconjunto do intervalo $[0, 1]$. Se, por exemplo considerarmos uma situação na qual não há um consenso entre vários especialistas que discordam entre si sobre o grau de pertinência de um elemento a um conjunto, uma forma de superar tal discordância é levar em consideração a opinião de todos os especialistas. Muitas pesquisas fazem uso desta teoria, principalmente na área de tomada de decisão (por exemplo [Farhadinia 2013, Wei 2012]). Em particular, vários tipos de operadores

OWA (ordered weighted average) foram propostos usando HFSs para realizar tomada de decisão (como podemos ver em [Bedregal, Reiser et al. 2014, Zhu et al. 2012]). Além disso, sistemas de inferência modelados utilizando a lógica difusa lidam com informações que podem ser formalmente discutidas e comparadas em termos de relações de ordem parcial definidas em conformidade com a estrutura da teoria dos reticulados. Em [Bedregal, Reiser et al. 2014], Bedregal et al. apresentaram funções de agregação hesitantes de acordo com a definição de funções de agregação valoradas em um reticulado limitado completo, ou seja, baseadas em uma ordem para valores difusos hesitantes típicos (typical hesitant fuzzy values – THFVs). Eles propuseram utilizar uma ordem para os THFVs a fim de definir as negações difusas hesitantes típicas. No entanto, observou-se que a negação difusa hesitante natural $\mathcal{N}(X) = \{1 - x : x \in X\}$ não satisfazia a propriedade de antitonicidade [Bedregal, Santiago et al. 2014], ou seja, a ordem utilizada naquele trabalho não era totalmente apropriada e por este motivo foi utilizada uma propriedade de antitonicidade enfraquecida em [Bedregal, Reiser et al. 2014]. Portanto, em [Santos et al. 2014], foi proposto considerar outra ordem, a saber, a ordem parcial de Xu-Xia apresentada em [Xu and Xia 2011] para assegurar a antitonicidade. Assim, nosso objetivo neste trabalho é realizar um estudo inicial das funções de agregação considerando a ordem parcial de Xu-Xia para os elementos difusos hesitantes típicos. Estudaremos aqui as negações, as t-normas e as t-conormas difusas hesitantes típicas, além de algumas das propriedades usualmente exigidas para tais operadores. Este trabalho está organizado da seguinte forma: na seguinte seção introduzimos o conceito dos conjuntos hesitantes típicos bem como a ordem parcial de Xu-Xia, a terceira seção está dedicada a apresentar os resultados obtidos e, por fim, temos as conclusões e referências.

2. Conjuntos Difusos Hesitantes Típicos

Assim como foi mencionado anteriormente, os conjuntos difusos hesitantes apresentados em [Torra and Narukawa 2009, Torra 2010] definem o grau de pertinência de um elemento a um conjunto através de um subconjunto no intervalo $[0, 1]$. Seja $\wp([0, 1])$ o conjunto das partes de $[0, 1]$, um conjunto difuso hesitante A definido sobre um conjunto não vazio U, é dado por:

$$A = \{(x, \mu_A(x)) : x \in U\} \tag{1}$$

onde $\mu_A : U \to \wp([0, 1])$. Existe um caso específico quando $\mu_A(x)$ é finito e não vazio para cada $x \in U$, que leva à definição dos conjuntos difusos hesitantes típicos (Typical Hesitant Fuzzy Sets – THFSs) apresentada em [Bedregal, Reiser et al. 2014].

Definição 1. *Seja $\mathbb{H} = \{X \subseteq [0, 1] : X$ é finito e $X \neq \emptyset\}$. Um conjunto difuso hesitante típico A definido sobre U é dado pela Eq. (1), onde $\mu_A : U \to \mathbb{H}$.*

Cada $X \in \mathbb{H}$ é chamado de elemento difuso hesitante típico de $\mathbb{H}$ e a cardinalidade de X (que corresponde ao número de elementos de X) é representada por $\#X$. Neste trabalho, optamos por utilizar a ordem parcial de Xu-Xia [Xu and Xia 2011] conforme justificativa mencionada na introdução. Esta ordem compara dois elementos difusos hesitantes de diferentes cardinalidades assumindo uma perspectiva pessimista, ou seja, os tomadores de decisão esperam por resultados desfavoráveis. Neste sentido, a fim de comparar elementos de cardinalidades diferentes, o menor valor é repetido no conjunto de menor cardinalidade até que as cardinalidades sejam iguais em ambos conjuntos. Em seguida, tais elementos são ordenados e comparados do maior ao menor

valor, um a um. A definição formal desta ordem, denotada por $\leq_{xx}$, foi apresentada em [Santos et al. 2014]. Alguns exemplos de conjuntos difusos hesitantes típicos, onde $X, Y \in \mathbb{H}$ são: $X = \{0.1, 0.6, 0.9\}$ e $Y = \{0.6, 0.5\}$, cujas cardinalidades são $\#X = 3$ e $\#Y = 2$, respectivamente. Quando aplicamos a ordem parcial de Xu-Xia temos que $X = \{0.1, 0.6, 0.9\}$ e $Y = \{0.5, 0.5, 0.6\}$ são incomparáveis, pois $0.9 > 0.6$ e $0.6 > 0.5$, no entanto $0.1 < 0.5$.

3. Operadores Difusos Hesitantes Típicos com a ordem parcial de Xu-Xia

3.1. Negações Difusas Hesitantes Típicas

Em [Santos et al. 2014], as negações difusas hesitantes típicas (NDHTs) foram definidas utilizando a ordem parcial de Xu-Xia. Dentre os principais resultados obtidos, podemos enfatizar a revisão das propriedades para as negações fortes, generalizando alguns importantes resultados como a caracterização das negações fortes de Trillas [Trillas 1979] no contexto dos elementos difusos hesitantes típicos.

Definição 2. *Seja $\mathcal{N} : \mathbb{H} \to \mathbb{H}$ uma função, dizemos que $\mathcal{N}$ é uma NDHT se satisfaz a condição de limite, $\mathcal{N}(\mathbf{0}_{\mathbb{H}}) = \mathbf{1}_{\mathbb{H}}$ e $\mathcal{N}(\mathbf{1}_{\mathbb{H}}) = \mathbf{0}_{\mathbb{H}}$; e se é decrescente, ou seja: se $X \leq_{xx} Y$ então $\mathcal{N}(Y) \leq_{xx} \mathcal{N}(X)$.*

Uma NDHT $\mathcal{N}$ é forte se é involutiva, ou seja, se para cada $X \in \mathbb{H}$, satisfaz $\mathcal{N}(\mathcal{N}(X)) = X$.

Em [Santos et al. 2015], foram apresentadas as t-normas e t-conormas difusas hesitantes típicas (TDHTs e SDHTs respectivamente) utilizando a ordem parcial de Xu-Xia conforme definições dadas nas subseções a seguir.

3.2. T-normas Difusas Hesitantes Típicas

Definição 3. *Seja $\mathcal{T} : \mathbb{H} \times \mathbb{H} \to \mathbb{H}$. $\mathcal{T}$ é uma TDHT, ou $\mathbb{H}$-t-norma, se é comutativa, associativa, monotônica e $\mathbf{1}_{\mathbb{H}}$ é o elemento neutro.*

Proposição 1. *[Santos et al. 2015] Seja $\mathcal{T} : \mathbb{H} \times \mathbb{H} \to \mathbb{H}$ uma $\mathbb{H}$-t-norma, então: se $\mathcal{T}$ satisfaz a lei do cancelamento, então $\mathcal{T}$ é estritamente monotônica. Além disso, se $\mathcal{T}$ é estritamente monotônica, então $\mathcal{T}$ apenas tem elementos idempotentes triviais e não possui zero divisores.*

3.3. T-conormas Difusas Hesitantes Típicas

Definição 4. *Seja $\mathcal{S} : \mathbb{H} \times \mathbb{H} \to \mathbb{H}$. $\mathcal{S}$ é uma SDHT, ou $\mathbb{H}$-t-conorma, se é comutativa, associativa, monotônica e $\mathbf{0}_{\mathbb{H}}$ é o elemento neutro.*

A fim de obter a $\mathbb{H}$-t-conorma dual ($\mathcal{S}_{\mathcal{T}}$) da $\mathbb{H}$-t-norma $\mathcal{T}$, usamos a noção de NDHTs.

Proposição 2. *[Santos et al. 2015] Sejam $\mathcal{N}$ uma NDHT forte e $\mathcal{T}$ uma $\mathbb{H}$-t-norma, então a função $\mathcal{S}_{\mathcal{T}} : \mathbb{H} \times \mathbb{H} \to \mathbb{H}$ definida por $\mathcal{S}_{\mathcal{T}}(X, Y) = \mathcal{N}(\mathcal{T}(\mathcal{N}(X), \mathcal{N}(Y)))$ é uma $\mathbb{H}$-t-conorma chamada $\mathbb{H}$-t-conorma dual de $\mathcal{T}$.*

4. Conclusões

O objetivo deste trabalho foi apresentar um estudo inicial das funções de agregação considerando a ordem parcial de Xu-Xia para os elementos difusos hesitantes típicos. Neste sentido, definimos as negações, as t-normas e as t-conormas difusas hesitantes típicas. Tais operadores são importantes pois podem generalizar, por exemplo, o complemento, a união e a inteseção nos conjuntos difusos, além de poderem ser utilizados para inferências nos sistemas de raciocínio aproximado ou controle. Futuramente, pretendemos continuar este estudo para outros operadores, como as implicações, bem como estudar pontos de equilíbrio e a noção de entropia neste mesmo contexto.

References

Bedregal, B.C. (2010). On interval fuzzy negations. *Fuzzy Sets and Systems*, 161: 2290–2313.

Bedregal, B., Santiago, R., Bustince, H., Paternain D. and Reiser, R. (2014). Typical hesitant fuzzy negations. *Int. Journal of Intelligent Systems*, 29: 525–543.

Bedregal, B., Reiser, R.H.S., Bustince, H., Lopez-Molina, C. and Torra, V. (2014). Aggregation Functions for Typical Hesitant Fuzzy Elements and the Action of Automorphisms. *Information Sciences*, 255(1) 82–99.

Bedregal, B. and Santiago, R.H.N. (2013). Interval representations, Łukasiewicz implicators and Smets-Magrez axioms. *Information Sciences*, 221: 192–200.

Farhadinia, B. (2013). A novel method of ranking hesitant fuzzy values for multiple attribute decision-making problems. *Int. Journal of Intelligent Systems*, 28(8): 752–767.

Santos, H., Bedregal, B., Santiago, R. and Bustince, H. (2014). Typical Hesitant Fuzzy Negations Based on Xu-Xia-partial order. In *IEEE Conf. Norbert Wiener in the 21st Century*, 1–6.

Santos, H., Bedregal, B., Santiago, R., Bustince, H. and Barrenechea, E. (2015). Construction of Typical Hesitant Triangular Norms regarding Xu-Xia-partial order. Submmited to *IFSA-EUSFLAT 2015*.

Torra, V. and Narukawa, Y. (2009). On hesitant fuzzy sets and decision. *In: proc. of FUZZ-IEEE*, pages 1378–1382.

Torra, V. (2010). Hesitant fuzzy sets. *Int. Journal of Intelligent Systems*, 25: 529–539.

Trillas, E. (1979). Sobre funciones de negación en la teoria de los conjuntos difusos. *Stochastica*, 3(1): 47–59.

Wei, G. (2012). Hesitant fuzzy prioritized operators and their application to multiple attribute decision making. *Int. Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 31: 176–182.

Xu, Z. and Xia, M. (2011). Distance and similarity measures for hesitant fuzzy sets. *Information Sciences*, 181 (11): 2128–2138.

Zadeh, L.A. (1965). Fuzzy sets. *Information and Control*, 8: 338–353.

Zhu, B., Xu, Z. and Xia, M. (2012). Hesitant fuzzy geometric Bonferroni means. *Information Sciences*, 205(1): 72–85.

Uma generalização de funções do tipo OWA para reticulados completos

Antonio Diego Silva Farias[1,2], **Regivan Hugo Nunues Santiago**[1]

[1]Departamento de Informática e Matemática Aplicada - DIMAp
Universidade Federal do Rio Grande do Norte - UFRN
Campus Universitário , Lagoa Nova- Natal - RN - Brasil

[2]Universidade Federal do Semi-Árido - UFERSA
Campus Universitário Pau dos Ferros - RN - Brasil

antonio.diego@ufersa.edu.br, regivan@dimap.ufrn.br

Abstract. *Ordered Weighted Averaging functions* (OWA) *are special types of the average aggregate function, with applicability in fields such as image processing and decision making. In this paper, we propose a new generalized form of* OWA *operators defined on complete lattices.*

Resumo. *Funções de médias ordenadas por pesos* (OWA) *são tipos especiais de função de agregação do tipo média, com aplicabilidade em campos como processamento de imagem e tomada de decisão. Neste trabalho, propomos uma nova forma generalizada de operadores* OWA *definidos sobre reticulados completos.*

1. Introdução

Algumas áreas da computação utilizam classes específicas de funções cuja finalidade é codificar em um único valor um conjunto de vários valores. Essas funções são denominadas funções de agregação.

As funções de agregação são classificadas em quatro subclasses: do tipo média, do tipo conjuntiva, do tipo disjuntiva e do tipo mista. As funções OWA, definidas inicialmente por Yager (1988), fornecem uma infinidade de exemplos de funções de agregação para o tipo média; as t-normas e t-conormas são exemplos de agregações conjuntivas e de agregações disjuntivas respectivamente.

Lizasoaim e Moreno (2013) construíram funções definidas sobre reticulados completos que estendem as funções OWA de Yager e estudaram suas propriedades. Essas funções são construídas a partir de elementos fixos $w_1, w_2, \cdots, w_n$, denominados de pesos, pertencentes ao reticulado.

O objetivo deste trabalho é propor uma nova forma generalizada de OWA, cujos pesos são definidos de acordo com cada entrada n-dimensional. Dividimos o artigo em duas partes: p rimeiramente trazemos algumas noções preliminares e em seguida apresentamos a proposta deste trabalho.

2. Noções Preliminares

2.1. Funções de Agregação

Uma função de agregação é uma ferramenta matemática que permite agrupar uma quantidade de informações em um único dado. Por exemplo, para determinar a altura média de

um conjunto finito $\{x_1, x_2, \cdots, x_n\}$ de pessoas, geralmente utilizamos a média aritmética. Formalmente temos:

Definição 1 *Uma* **função de agregação** n*-ária (ou operador de agregação* n*-ário) é uma função isotônica* $f : [0,1]^n \to [0,1]$ *com* $f(0,0,\cdots,0) = 0$ *e* $f(1,1,\cdots,1) = 1$.

Para maiores detalhes, o leitor pode consultar Beliakov, Pradera e Calvo (2007). Na tabela, a seguir, apresentamos alguns exemplos de funções de agregação:

Média aritmética	$MED(x_1, x_2, \cdot, x_n) = \dfrac{\sum_{i=1}^{n} x_i}{n}$
Máximo	$MAX(x_1, x_2, \cdots, x_n) = max\{x_1, x_2, \cdots, x_n\}$
Mínimo	$MIN(x_1, x_2, \cdots, x_n) = min\{x_1, x_2, \cdots, x_n\}$
Produto	$PROD(x_1, x_2, \cdots, x_n) = x_1.x_2.\cdots.x_n$

Tabela 1. Exemplos de funções de agragação

Podemos encontrar diversas aplicações para as funções de agregação, como por exemplo Warshawsky e Mavris (2013), Parternain et al (2012) e Paternain et al (2015).

2.2. Funções OWA

Dado um vetor n-dimensional $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, podemos reordenar suas coordenadas de modo a obter um novo vetor $s(\mathbf{x}) = (x_{(1)}, x_{(2)}, \cdots, x_{(n)})$, onde $x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(n)}$ e $\{x_{(1)}, x_{(2)}, \cdots, x_{(n)}\} = \{x_1, x_2, \cdots, x_n\}$. Para cada vetor de pesos $\mathbf{w} = (w_1, w_2, \cdots, w_n) \in [0,1]^n$, com $\sum_{i=1}^{n} w_i = 1$, definimos a função de agregação $OWA_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} x_{(i)} w_i$.

Funções como $MED(\mathbf{x})$, $MIN(\mathbf{x})$ e $MAX(\mathbf{x})$ definidas na tabela 1 são exemplos de funções do tipo OWA. Para obter $OWA_{\mathbf{w}} = MED$, por exemplo, basta tomar $u_w = \frac{1}{n}$, para $1 \leq i \leq n$.

2.3. t-normas e t-conormas

Alguns tipos especiais de funções de agregação fornecem um modelo para conjunções e disjunções em lógica difusa. Estes operadores são respectivamente denominados de t-normas e t-conormas, conforme definimos abaixo:

Definição 2 *Uma* **t-norma** *é uma função bidimensional* $T : [0,1]^2 \to [0,1]$ *isotônica, comutativa, associativa, com elemento neutro* 1.

Definição 3 *Uma* **t-conorma** *é uma função bidimensional* $S : [0,1]^2 \to [0,1]$ *isotônica, comutativa, associativa, com elemento neutro* 0.

Para conhecer melhor este assunto veja Klement, Messiar e Pap (2000). Abaixo apresentamos alguns exemplos de t-normas e t-conormas:

t-normas	**t-conormas**
$T_{mim}(x,y) = MIN(x,y)$	$S_{max}(x,y) = MAX(x,y)$
$T_P(x,y) = x.y$	$S_P(x,y) = x + y - xy$
$T_{LK}(x,y) = MAX(x+y-1,0)$	$S_{LK}(x,y) = MIN(x+y,1)$
$T_D(x,y) = \begin{cases} 0, & \text{se } x,y \in [0,1) \\ MIN(x,y), & \text{caso contrário} \end{cases}$	$S_D(x,y) = \begin{cases} 1, & \text{se } x,y \in (0,1] \\ MAX(x,y), & \text{caso contrário} \end{cases}$

Tabela 2. Exemplos de t-normas e de t-conormas

3. Forma generalizada da função OWA

Recentemente, Lizasoaim e Moreno (2013) utilizaram t-normas e t-conormas sobre reticulados completos[1] para definir o conceito de função OWA neste ambiente. As definições e propriedades das t-normas e t-conormas sobre reticulados podem ser encontradas em De Baets, R. Mesiar (1999).

Vale ressaltar que todo reticulado completo possui maior e menor elementos denotados respectivamente por $\top$ e $\bot$.

Dados um reticulado completo L, uma t-norma $T : L \times L \to L$ e uma t-conorma $S : L \times L \to L$, dizemos que $\mathbf{w} = (w_1, w_2, \cdots, w_n) \in L^n$ é:

(i) Um **vetor de pesos** se $S(w_1, w_2, \cdots, w_n) = \top_L$;
(ii) Um vetor de pesos é **distributivo** se para todo $a \in L$ temos que $a = T(a, S(w_1, \cdots, w_n)) = S(T(a, w_1), \cdots, T(a, w_n))$..

A ordenação de um vetor n-dimensional $\mathbf{a} = (a_1, A_2, \cdots, a_n) \in L^n$ é feita à partir dos seguintes valores:

- $b_1 = a_1 \vee a_2 \vee \cdots \vee a_n$
- $b_2 = [(a_1 \wedge a_2) \vee \cdots \vee (a_1 \wedge a_n)] \vee [(a_2 \wedge a_3) \vee (a_2 \wedge a_n)] \vee \cdots \vee [a_{n-1} \wedge a_n]$
- $\vdots$
- $b_k = \bigvee\{a_{j_1} \wedge \cdots \wedge a_{j_k} : \{j_1, \cdots, j_k\} \subseteq \{1, 2, \cdots n\}\}$
- $\vdots$
- $b_n = a_1 \wedge \wedge a_2 \wedge \cdots \wedge a_n$

Definição 4 (Definição 3.5 de Lizasoaim e Moreno (2013)) *Seja L um reticulado completo, T uma t-norma, S uma t-conorma e $\mathbf{w} = (w_1, \cdots, w_n) \in L^n$ um vetor distributivo de pesos. Para cada $(a_1, \cdots a_n) \in L^n$, considere o vetor ordenado $(b_1, \cdots, b_n)$ construído anteriormente, a função $F : L^n \to L$ dada por*

$$F_{\mathbf{w}}(a_1, \cdots, a_n) = S(T(w_1, b_1), \cdots, T(w_n, b_n))$$

é chamada de operador OWA n-ário sobre a tripla $\langle L, T, S\rangle$.

Observe que na definição acima cada vetor de pesos distributivo $\mathbf{w} \in L^n$, fixado, gera uma função $OWA_{\mathbf{w}}$. Neste sentido, propomos uma nova generalização de OWA,

[1]Em essência, um reticulado é um conjunto L munido de dois operadores binários $\vee$ e $\wedge$ (denominados de supremo e ínfimo) que satisfazem as propriedades da comutatividade, da associatividade, da idempotência e da absorção. Quando é possível determinar o supremo e o ínfimo de qualquer subconjunto $S \subseteq L$ dizemos que L é um reticulado completo.

onde o vetor de pesos se adapta ao vetor de entrada $\mathbf{x}$, isto é, para cada entrada $\mathbf{x}$ obtemos um vetor de pesos $\mathbf{w} = \mathbf{w}(\mathbf{x})$.

Para realizar este feito, precisamos instituir o conceito de função de pesos: Uma **função de pesos** associada a uma tripa $\langle L, T, S\rangle$ é uma função $f : L^n \to L^n$, digamos $f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_n(\mathbf{x}))$, tal que $S(f(\mathbf{x})) = \top_L$. Uma **função distributiva de pesos** é uma função de pesos f tal que $a = T(a, S(f(\mathbf{x}))) = S(T(a, f_1(\mathbf{x})), \cdots, T(a, f_n(\mathbf{x}))$ para qualquer $a \in L$.

Definição 5 *Sejam L um reticulado completo, T uma t-norma, S uma t-conorma e $f : L^n \to L^n$ uma função distributiva de pesos. A função*

$$F_f(\mathbf{x}) = S(T(f_1(\mathbf{x}), b_1), \cdots, T(f_n(\mathbf{x}), b_n)),$$

cujos b_i são como definidos anteriormente, é chamado de **operador OWA generalizado**.

Teorema 1 *Seja $L = [0, 1]$ munido com a t-norma produto T_P e a t-conorma S_{LK}. Então para cada vetor de pesos $\mathbf{w} \in [0, 1]^2$, no sentido de Yager, i.e., com $\sum_{i=1}^{n} w_i = 1$, existe uma função distributiva de pesos $f : [0, 1]^n \to [0, 1]^n$ tal que $OWA_{\mathbf{w}}(\mathbf{x}) = F_f(\mathbf{x})$ para todo $\mathbf{x} \in [0, 1]^n$.*

Demonstração: Basta definir $f(\mathbf{x}) = (w_1, \cdots, w_n)$ e ver que $b_1 = x_{(1)}$, $b_2 = x_{(2)}, \cdots, b_n = x_{(n)}$.

4. Conclusões

Neste trabalho introduzimos uma forma generaliza de função de agregação do tipo Média, denominada de OWA, para reticulados completos. As funções do tipo OWA, inicialmente estudadas por Yager (1988) são de grande importância no campo da computação por causa de suas ampla capacidade aplicabilidade. A maior contribuição deste artigo surge com a sua capacidade de estender o conceito de OWA para outro ambientes, como por exemplo o intervalar. Os próximos passos da pesquisas consistem em estudar as propriedades que operadores apresentam, bem como suas aplicações.

Referências

[1] B. D. Baets, R. Mesiar, Triangular norms on product lattices, Fuzzy Sets Syst. 104 (1999) 61-75.

[2] G. Beliakov, A. Pradera, T. Calvo, Aggregation functions: a guide for practitioners, Stud. Fuzziness Soft Comput. 221 (2007).

[3] E. P. Klement, R. Mesiar, E. Pap, Triangular Norms, Springer, V.8 (2000).

[4] I. Lizasoaim, C. Moreno, OWA operators defined on complete lattices, Fuzzy Set and Systems, 224 (2013), 36-53.

[5] D. Paternain, J. Fernandeza, H. Bustince, R. Mesiar ,G. Beliakov, Construction of image reduction operators using averaging aggregation functions, Fuzzy Sets and Systems 261 (2015) 87-111.

[6] D. Paternain, A. Jurio, E. Barrenechea, H. Bustince, B. C. Bedregal, E. Szmidt: An alternative to fuzzy methods in decision-making problems. Expert Syst. Appl. 39(9): 7729-7735 (2012).

[7] D. Warshawsky, D. Mavris, Choosing Aggregation Functions for Modeling System of Systems Performance, Procedia Computer Science 16 (2013) 236-244.

[8] R. R. Yager, Ordered weighted averaging aggregation operators in multicriteria decision making, IEEE Trans. Syst. ManCybern. 18 (1988) 183-190.

Uma abordagem *essencialmente* intervalar para o cálculo de pertinência fuzzy

Ronildo Moura[1], **Benjamin Bedregal**[1]

[1]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal – 59.072-970– Natal – RN – Brasil

ronildo@ppgsc.ufrn.br, bedregal@dimap.ufrn.br

Resumo. *O principal objetivo do uso de dados com natureza intervalar é representar medidas numéricas dotadas de imprecisões, que são normalmente encontradas em situações do mundo real. No entanto, é necessário adaptar técnicas usuais para serem aplicadas sobre dados do tipo intervalo. Para agrupamento de dados fuzzy, é necessário adaptar o cálculo do grau de pertinência fuzzy quando envolve o uso de distâncias* essencialmente *intervalares. Portanto, neste trabalho, nosso objetivo é apresentar uma extensão do cálculo da pertinência fuzzy do algoritmo Fuzzy C-Médias para o contexto intervalar.*

1. INTRODUÇÃO

Agrupamento de dados é a tarefa de organizar objetos em grupos seguindo algum critério, os dados são alocados em grupos ou subconjuntos de tal forma que os objetos em cada grupo compartilham alguma *similaridade*[Jain et al. 1999]. Agrupamento de dados tem sido utilizado em inúmeros problemas e aplicações, como na etapa de pré-processamento de outra técnica de análise de dados, como uma ferramenta de descoberta de conhecimento, e outros [Jain et al. 1999]. O uso mais conhecido para agrupamento é atribuir categorias para objetos não-rotulados, onde a estrutura dos grupos não é conhecida previamente.

O agrupamento Fuzzy C-Médias [Bezdek 1981] é um dos mais populares e simples algoritmos de agrupamento. No agrupamento fuzzy, um objeto pode pertencer a mais de um grupo ao mesmo tempo com diferentes graus de pertinência.

O aumento no volume e na complexidade dos dados tem requerido avanços na metodologia da análise de dados. Em agrupamento clássico, os objetos a serem agrupados são geralmente representados por variáveis categóricas ou numéricas. Entretanto, esse modelo é muito limitado para representar a variabilidade e incertezas inerentes aos dados, e variáveis valoradas em intervalos são necessárias.

Chakraborty [Chakraborty and Chakraborty 2006] apontou que a distância entre duas medidas imprecisas (dois intervalos) também pode ser uma medida imprecisa. Em [Santana and Santiago 2013], os autores deram a primeira noção sobre métricas baseadas em intervalos (i-métrica), utilizando o conceito da Representação Intervalar [Santiago et al. 2006]. Esses trabalhos mostram que é razoável utilizar medidas valoradas em intervalos para representar uma distância entre entidades com medidas intervalares.

De fato, o agrupamento de dados em objetos do tipo intervalo utilizando distâncias que também são valoradas em intervalos, para o cálculo da similaridade/dissimilaridade,

torna esses métodos capazes de preservar a variabilidade e incertezas inatas aos dados. A principal contribuição deste artigo é prover uma adaptação do cálculo da pertinência fuzzy do algoritmo de agrupamento Fuzzy C-Médias (FCM) para o contexto intervalar seguindo o paradigma da Representação Intervalar.

O artigo é organizado em 4 seções e estrutura da seguinte forma: Seção 2 introduz a noção de métrica *essencialmente* intervalar. Seção 3 descreve o método proposto, uma adaptação para construir graus de pertinência intervalar que mapeiam as distâncias essencialmente intervalares. Seção 4 apresenta a conclusão deste trabalho.

2. Distâncias essencialmente Intervalares

A matemática intervalar oferece um conjunto de métodos para manipular a falta de precisão nas operações numéricas, que normalmente envolve entradas de dados imprecisas, limitações físicas das máquinas, erros de arrendondamentos, entre outros. Os conceitos básicos da aritmética intervalar podem ser encontrados na literatura pertinente [Moore 1962] com mais detalhes.

Definição 2.1. Dados a e $b \in \mathbb{R}$ tal que $a \leq b$, é denotado de intervalo X, o conjunto $X = \{x \in \mathbb{R}\ a \leq x \leq b\}$ e representaremos por $X = [a; b]$. O conjunto dos intervalos assim definidos é representado por $\mathbb{I}(\mathbb{R})$ e o conjunto dos intervalos com extremos não-negativos por $\mathbb{I}(\mathbb{R})^+ = \{[a, b] \in \mathbb{I}(\mathbb{R}); 0 \leq a \leq b\}$.

O intervalo X pode ser denotado pelas projeções inferior ($\underline{X}$) e superior ($\overline{X}$) $X = [\underline{X}, \overline{X}]$ que representam o limite inferior e superior do intervalo X, respectivamente. O intervalo é chamado de *degenerado* quando $\underline{X} = \overline{X}$.

Extensão Intervalar foi proposta por [Moore 1962] como uma forma de generalizar funções reais em termos de intervalo. Santana e Bedregal [Santiago et al. 2006] avançaram nesse tema ao introduzir o conceito de Representação Intervalar. Os autores estudaram uma classe de funções intervalares que são associadas a algoritmos que satisfazem a propriedade de optimalidade. Dito isso, o objetivo dos autores foi oferecer um framework para criar funções intervalares a partir de funções reais assegurando os princípios da corretude da extensão intervalar e da optimalidade de algoritmos intervalares. [Santiago et al. 2006].

O paradigma da Representação Intervalar[Santiago et al. 2006] foi aplicado no conceito de distância por Santana e Santiago [Santana and Santiago 2013]. Eles propuseram uma generalização no conceito de métrica, com modificações nos axiomas e na valoração. Os autores desenvolveram uma generalização baseada em intervalos para métricas usuais, ao invés de funções na forma $d : M \times M \longrightarrow \mathbb{R}$ eles apresentaram funções com a assinatura: $d : M \times M \longrightarrow \mathbb{I}(\mathbb{R})^+$, onde $\mathbb{I}(\mathbb{R})^+$ é o conjunto dos intervalos fechados não negativos. Além disso, os autores apresentaram uma distância , chamada km-métrica, que captura a ordem Kulish-Miranker.

Definição 2.2 ([Santana and Santiago 2013])**.** Sejam dois intervalos $X, Y \in \mathbb{I}(\mathbb{R})$, considere o conjunto de distâncias euclidiana entre os elementos X e Y representada por $D_{XY} = \{d(x, y) : x \in X \text{ e } y \in Y\}$. Então a função $d_{KM} : \mathbb{I}(\mathbb{R}) \times \mathbb{I}(\mathbb{R}) \longrightarrow \langle \mathbb{I}(\mathbb{R})^+, \leq_{KM}, \ll, [0, 0] \rangle$ é definida por

$$d_{KM}(X, Y) = \begin{cases} [0; 0] & \text{, se } X = Y; \\ \left[\min(D_{XY}); \max(D_{XY})\right] & \text{, se } X \neq Y\ . \end{cases} \quad (1)$$

A distância entre os intervalos X e Y é codificada no conjunto D_{XY}, o qual contém a distância euclidiana exata de todos os pares $(x, y) \in X \times Y$.

3. ABORDAGEM INTERVALAR PARA O CÁLCULO DA PERTINÊNCIA FUZZY

O FCM clássico é um algoritmo iterativo que atribui graus de pertinência fuzzy a objetos do conjunto de dados e atualiza os centros dos grupos de acordo com o grau de pertinência. Os graus de pertinência funcionam como valores de pesos que representam o grau de contribuição de um objeto na estimativa dos centros dos grupos.

O grau de pertinência fuzzy do k-ésimo objeto do conjunto de dados em relação ao i-ésimo centro é descrito pela função f abaixo.

$$u_{ik} = \left[\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{2}$$

onde d_{ik} denota a distância entre o centro (i) e o objeto, e se $d_{ik} = 0$ para algum i, então para todo i com $d_{ik} \neq 0, u_{ik} = 0$ senão $u_{ik} = 1$.

É importante destacar que o cálculo da pertinência não depende somente da distância do k-ésimo objeto ao i-ésimo centro, mas também das distâncias do objeto (k) aos demais centros. Assim, o cálculo da pertinência recebe um vetor com as distâncias do objeto (k) a todos os protótipos (centros) e calcula para cada protótipo seus graus de pertinência.

No contexto de *i*-métricas, como discutido na Seção 2, pode ser definido, informalmente, uma função que projeta a *maior distância relativa* e *menor distância relativa*. A maior distância relativa é dada pelo conjunto contendo a maior distância (projeção superior) do objeto (k) ao i-ésimo protótipo e com as menores (projeção inferior) distâncias do objeto (k) aos demais protótipos. De forma dual é possível definir informalmente a função que projeta a *menor distância relativa*. As funções definidas acima, informalmente, são chamadas de $g_{\max}$ e $g_{\min}$.

Considere $\mathbb{D}_k = \{[\underline{d}_{ik}, \overline{d}_{ik}] | i = 1, \ldots, c\}$ como sendo um vetor com as medidas de dissimilaridades obtidas pela d_{KM} (Equação 1), do objeto $\mathbf{x}_k$ ao centros. As funções $g_{\min} : \langle \mathbb{I}(\mathbb{R})^+, [0,0] \rangle^c \longrightarrow \langle \mathbb{R}^+, 0 \rangle^c$ e $g_{\max} : \langle \mathbb{I}(\mathbb{R}) +, [0,0] \rangle^c \longrightarrow \langle \mathbb{R}^+, 0 \rangle^c$ são as projeções da *menor* e *maior* distância relativa para a $i-$ésima referência do conjunto de protótipos, respectivamente. As projeções podem ser definidas como $g_{\min}(\mathbb{D}_k) = \{\{\underline{d}_{jk} \mid i = j\} \cup \{\overline{d}_{jk} \mid i \neq j\}$ tal que $j = 1, \ldots, c\}$ para $\forall i, (1 \leq i \leq c)$ e $g_{\max}(\mathbb{D}_k) = \{\{\overline{d}_{jk} \mid i = j\} \cup \{\underline{d}_{jk} \mid i \neq j\}$ tal que $j = 1, \ldots, c\}$ para $\forall i, (1 \leq i \leq c)$.

A versão intervalar do cálculo dos graus de pertinência Fuzzy foi obtida utilizando o paradigma da Representação Intervalar [Santiago et al. 2006], $F(\mathbb{D}_k) = [\min f([\mathbb{D}_k]), \max f([\mathbb{D}_k])]$ e F é a **melhor representação intervalar** de f.

O cálculo da pertinência intervalar a partir de um vetor com medidas intervalares precisa mapear os extremos dos intervalos. Define-se F pelas projeções da *menor* e *maior distância relativa* apresentadas anteriormente. Dito isso, $\mathbb{U}_k = F(\mathbb{D}_k) = [f(g_{\max}(\mathbb{D}_k)), f(g_{\min}(\mathbb{D}_k))]$ tal que $\mathbb{U}_k \in \mathbb{U} = [\underline{u}_{ik}, \overline{u}_{ik}]$, note que f é uma função que inverte a ordem, pois $\mu_{ik} > \mu_{pk} \Leftrightarrow d_{ik} < d_{pk}$. Portanto, o menor valor de u_{ik} é computado por f com os extremos obtidos pela função $g_{\max}(\mathbb{D}_k)$ e o maior valor de u_{ik} com $g_{\min}(\mathbb{D}_k)$. Substituindo a expressão das projeções $g_{\max}$ e $g_{\min}$ em f (Eq. 2) e fazendo algumas simplificações, obtém-se:

$$u_{ik} = \begin{cases} \left[u_{\min_{ik}}, u_{\max_{ik}}\right] & \text{, caso } I_k = \emptyset \text{ e } E_k = \emptyset \\ \left[0, u_{\max_{ik}}\right] & \text{, caso } I_k \neq \emptyset \text{ e } i \notin I_k \text{ e } E_k = \emptyset \\ \left[u_{\min_{ik}}, 1\right] & \text{, caso } I_k \neq \emptyset \text{ e } i \in I_k \text{ e } E_k = \emptyset \text{ e } |I_k| = 1 \\ \left[0, 1\right] & \text{, caso } I_k \neq \emptyset \text{ e } i \in I_k \text{ e } E_k = \emptyset \text{ e } |I_k| > 1 \\ \left[0, 0\right] & \text{, caso } E_k \neq \emptyset \text{ e } i \notin E_k \text{ e } I_k = \emptyset \\ \left[\dfrac{1}{|E_k|}, \dfrac{1}{|E_k|}\right] & \text{, caso } E_k \neq \emptyset \text{ e } i \in E_k \text{ e } I_k = \emptyset \\ \left[0, \dfrac{1}{1+|E_k|}\right] & \text{, caso } I_k \neq \emptyset \text{ e } E_k \neq \emptyset \text{ e } i \in I_k \\ \left[\dfrac{1}{|I_k|+|E_k|}, \dfrac{1}{|E_k|}\right] & \text{, caso } I_k \neq \emptyset \text{ e } E_k \neq \emptyset \text{ e } i \in E_k \end{cases} \tag{3}$$

Onde os conjuntos $I_k = \{i | 0 \in d_{ik} \text{ e } d_{ik} \neq [0,0]\}$ e $E_k = \{i | d_{ik} = [0,0]\}$, considere $|I_k|$ e $|E_k|$ como as cardinalidades desses conjuntos, respectivamente.

4. CONCLUSÃO

Neste trabalho foi proposto um novo método para calcular os graus de pertinência fuzzy aplicando conceitos da matemática intervalar. Se o conjunto de dados for composto por dados que carregam com imprecisão e/ou variabilidade e forem representados em intervalos, pode ser adequado utilizar distâncias que carregam essa imprecisão/variabilidade. Portanto, a adaptação da função que obtêm a pertinência fuzzy dos objetos do conjunto aos centros dos grupos baseado no conceito do paradigma da Representação Intervalar foi necessária. Desta forma evitamos a perda da imprecisão/variabilidade no processo de agrupamento, pois evita os arrendondamentos ou outras transformações nas variáveis intervalares que são comuns nos métodos clássicos de agrupamento.

Referências

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Pleneum Press, New York.

Chakraborty, C. and Chakraborty, D. (2006). A theoretical development on a fuzzy distance measure for fuzzy numbers. *Mathematical and Computer Modelling*, 43(3):254–261.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

Moore, R. E. (1962). *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. PhD thesis, Department of Computer Science, Stanford University.

Santana, F. and Santiago, R. (2013). Interval metrics, topology and continuous functions. *Computational and Applied Mathematics*, 32(3):459–470.

Santiago, R. H. N., Bedregal, B. R. C., and Acioly, B. M. (2006). Formal aspects of correctness and optimality of interval computations. *Formal Aspects of Computing*, 18(2):231–243.

Suporte a Tolerância a Falhas para Aumentar o Desempenho de Redes em Chip

Alba S. B. Lopes[1], Marcio E. Kreutz[2], Monica M. Pereira[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do RN (IFRN)
Campus Natal-Zona Norte – Natal – RN – Brazil

[2] Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN) – Natal, RN – Brasil

alba.lopes@ifrn.edu.br, {kreutz, monicapereira}@dimap.ufrn.br

Abstract. *Replicated components in multi-processor systems is a natural potential to provide fault tolerance. Although, concurrent communication architectures - Networks-on-Chip - are not designed to support fault tolerance since all routers are used to compose paths for messages. However, it is possible to connect extra (spare) routers on the network topology to replace a faulty one to ensure the continuity of system functionality, even with failure. In this paper, we assume a Network-on-Chip already comprised by spare routers and propose a technique to explore the gains in performance that could be achieved by using them as regular routers when the system runs without failures. Results confirm the expected performance improvements, validating the technique.*

Resumo. *A existência de elementos replicados em sistemas multiprocessados proporciona um potencial natural para tolerância a falhas. Entretanto, em relação a comunicação, arquiteturas concorrentes, como as Redes em Chip, não são projetadas para provê suporte a tolerância a falhas, uma vez que todos os roteadores são usados no caminho das mensagens. Contudo, é possível conectar roteadores extras (spares) na topologia da rede de forma a substituir um roteador com falha, garantindo a funcionalidade do sistema. Neste trabalho, nós assumimos uma rede em chip que agrega roteadores spares e propomos uma técnica para explorar o ganho de performance que pode ser obtido através do uso desses roteadores quando a rede executa sem a presença de falhas. Resultados confirmaram o aumento de performance esperado, validando a técnica.*

1. Introdução

Os avanços nos processos de fabricação de circuitos integrados permitiram a inclusão de centenas de núcleos em um único chip. A era dos sistemas com múltiplos núcleos (em inglês *multicore/manycore*) proporcionam alcançar alto desempenho enquanto lidam com consumo de energia (HILL; MARTY, 2008). De forma a proporcionar comunicação entre os núcleos, as redes em chip (em inglês Network-on-Chips – NoCs) se apresentam como a solução mais rápida e escalável em substituição aos barramentos (Kumar et al, 2002). Junto com o aumento da densidade dos circuitos integrados, a falha

em transistores e interconexões também aumentou devido ao processo de miniaturização (Furber, 2006). Efeitos de envelhecimento que causam degradação no sistema e afetam progressivamente a confiança da NoC e desafiam os projetistas por soluções que considerem o tratamento de falhas permanentes, evitando a falha total do sistema (Pande et al, 2005).

Neste trabalho é proposta uma solução para tolerância a falhas em nível de hardware, para aumentar a confiança da rede em chip que pode ser usada para aumentar o desempenho em um cenário onde não há falhas na rede. A solução consiste em adicionar roteadores *spares* à rede que devem ser usados para substituir roteadores com falhas. Enquanto o roteador original não apresenta falhas, o *spare* é usado para proporcionar um segundo caminho para o processador que pode usá-lo para paralelizar o envio de pacotes e aumentar o desempenho da comunicação. A principal contribuição do trabalho é o uso do mecanismo de tolerância a falhas para aumentar o desempenho e não apenas a confiança do sistema.

Esse trabalho está organizado da seguinte forma: a seção 2 apresenta trabalhos relacionados. A seção 3 descreve a abordagem proposta. A seção 4 discute os resultados alcançados. Por fim, a seção 5 apresenta as conclusões.

2. Trabalhos Relacionados

Na literatura, é possível encontrar muitas soluções para lidar com falhas em NoC. As soluções abordam a detecção e tolerância a falhas em nível de software e hardware. As soluções em software são baseadas na detecção de erros e correção de códigos durante a transmissão, como é o caso do trabalho de Yu e Ampadu (2010). Além disso, alguns trabalhos utilizam algoritmos de roteamento adaptativos para evitar rotas com falhas, como o trabalho de Fick et al (2009). As soluções em hardware buscam incluir hardware extra para substituir componentes com falhas, como é o caso de Chatterjee et al (2014). Os autores propoem a inclusão de um roteador extra para cada roteador da rede, de forma realizar a substituição em caso de falhas.

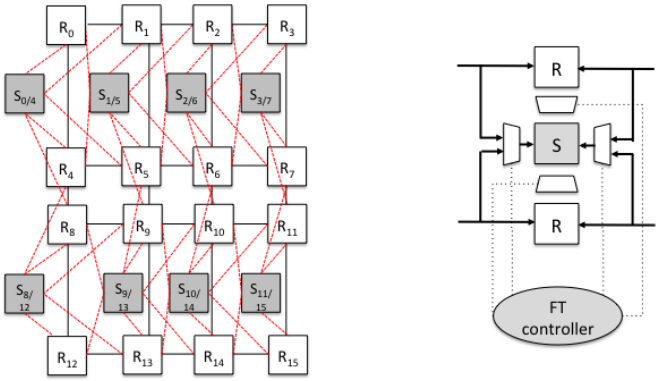
A maior parte dos trabalhos que usam redundância como mecanismo de tolerância a falhas não buscam utilizar o hardware extra para aumentar o desempenho da NoC, sendo essa a maior contribuição deste do trabalho proposto.

3. Rede em Chip Tolerante a Falhas

A solução proposta consiste em adicionar roteadores extras na rede para substituir roteadores que apresentarem falha. Para reduzir o custo em área e consumo de energia, a proposta consiste em adicionar um roteador extra para cada dois roteadores da rede, como apresentado na Figura 1. Dessa forma, considerando o cenário em que falhe um roteador de cada par, a rede é capaz de funcionar com até 50% de roteadores com falhas.

Na Figura 1, as caixas com R são os roteadores originais de uma rede 4x4 e as caixas com S são os roteadores *spare*. Os multiplexadores nos roteadores *spare* (ilustrados na Figura 1b) são necessários para selecionar de qual roteador original o *spare* irá receber a informação. Um mecanismo de controle (FT-controller) também foi incluído e é responsável por definir a seleção dos multiplexadores. Nesta abordagem o FT-controller seleciona os roteadores através de um passo offline. Dessa forma, é transparente para o

sistema se é o roteador original ou o *spare* que está ativo no momento, não sendo necessária a realização de nenhuma alteração na rota dos pacotes em caso de falhas.



a) NoC 4x4 Tolerante a Falhas b) Roteadores spare com muxes

Figura 1. NoC Tolerante a Falhas

Para explorar o uso dos roteadores *spares* enquanto da rede não apresenta falhas de forma a aumentar o desempenho, deve ser possível: 1) injetar mais pacotes na rede simultaneamente; e 2) explorar melhor o paralelismo da rede através da utilização dos caminhos alternativos oferecidos pelos roteadores *spares*. A segunda técnica tira vantagem da forma como os roteadores são conectados através dos multiplexadores. Ao observar as conexões na Figura 1, é possível perceber que quando os roteadores de destino estão distantes mais de um passo do roteador original, o roteador *spare* pode sempre passar por cima do primeiro roteador e alcançar o destino em um passo apenas.

Essas duas técnicas irão ajudar a aumentar a latência e a vazão através da exploração de caminhos concorrentes na NoC. E mesmo na presença de falhas que afetem os roteadores, além de tolerá-las, essa técnica irá manter um desempenho melhor do que uma rede comum. Embora esta técnica introduza área extra e eleve o consume de potência, esses custos são parcialmente compensados pelo aumento do desempenho quando o sistema opera sem falhas.

4. Resultados Experimentais

Para avaliar o impacto do uso dos roteadores *spares* na melhoria da latência e vazão, foram avaliadas três aplicações reais: TVOPD (Murali et al, 2009), NCS (Tino, 2011) e TMPEG4. Os experimentos foram executados em um simulador implementado em Java, onde a cada ciclo de *clock,* todos os componentes do roteador são executados concorrentemente. O algoritmo de roteamento utilizado nos experimentos foi o algoritmo XY. Os resultados podem ser observados na Figura 2.

Para todas as aplicações, foi observada uma melhoria de 10% para latência e 6% para vazão. Embora os resultados pareçam não apresentar uma melhora muito significativa, vale observar que esse ganho é praticamente a custo zero, uma vez que os roteadores extras estarão presentes na arquitetura de qualquer forma. E ainda, esses resultados são referentes a todas as mensagens das aplicações, e não apenas para as que os roteadores *spares* podem ser utilizados.

Levando em consideração esse fato, os resultados foram analisados levando em consideração apenas as mensagens em que puderam ser enviadas pelos roteadores extras. Nesse caso, o ganho chegou a ser de 5 a 6 vezes em relação a latência e cerca de 40% para vazão. Como esperado, com os roteadores *spares* não é possível aumentar o desempenho para todas as mensagens, apenas para as que são enviadas por caminhos distintos por um mesmo processador.

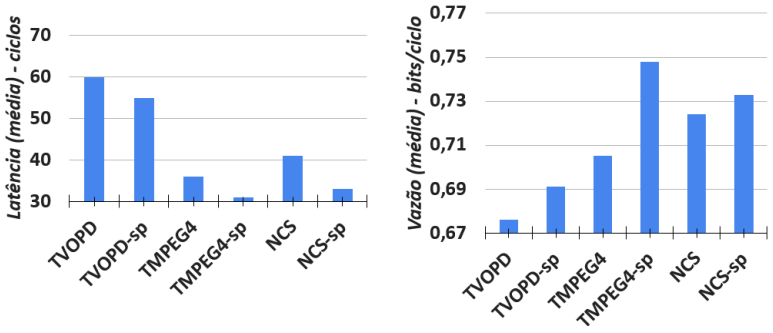


Figura 2. Resultados de Latência e Vazão

5. Conclusões

Este artigo apresentou uma proposta de tolerância falhas em NoC usando roteadores extras para substituir roteadores com falhas. A principal contribuição deste trabalho é utilizar os roteadores extras para aumentar o desempenho da NoC quando a rede não apresenta falhas. Os resultados demonstraram ganhos de ordem de 5 a 6 vezes em redução de latência e cerca de 40% em aumento da vazão.

Referências

Hill, M.D.; Marty, M.R., "Amdahl's Law in the Multicore Era," Computer , vol.41, no.7, July 2008, pp.33-38.

Kumar, S.; Jantsch, A.; Soininen, J.-P.; Forsell, M.; Millberg, M.; Oberg, J.; Tiensyrja, K.; Hemani, A., "A network on chip architecture and design methodology," *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, 2002, pp.105-112.

Furber, S., "Living with Failure: Lessons from Nature?," Test Symposium, 2006. ETS '06. Eleventh IEEE European , 21-24 May 2006, pp. 4-8., Pande, P.P.; Grecu, C.; Ivanov, A.; Saleh, R.; De Micheli, G., "Design, synthesis, and test of networks on chips," *Design & Test of Computers, IEEE* , vol.22, no.5, Sept.-Oct. 2005, pp.404-413.

Pande, P.P.; Grecu, C.; Ivanov, A.; Saleh, R.; De Micheli, G., "Design, synthesis, and test of networks on chips," *Design & Test of Computers, IEEE* , vol.22, no.5, Sept.-Oct. 2005, pp.404-413.

Yu, Q.; Ampadu, P., "Transient and Permanent Error Co-management Method for Reliable Networks-on-Chip," Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on, 3-6 May 2010, pp.145-154.

Fick, D.; DeOrio, A.; Chen, G.; Bertacco, V.; Sylvester, D.; Blaauw, D., "A highly resilient routing algorithm for fault-tolerant NoCs," Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09., 20-24 April 2009, pp.21-26.

Chatterjee, N.; Chattopadhyay, S.; Manna, K., "A spare router based reliable Network-on-Chip design," Circuits and Systems (ISCAS), 2014 IEEE International Symposium on, 1-5 June 2014, pp.1957-1960.

Murali, S. et al., Synthesis of Networks on Chips for 3D Systems on Chips, Proceedings ofAsia and South Pacific Design Automation Conference (ASP-DAC), Piscataway: IEEE Press, Jan. 2009, pp. 242-247.

Tino, A., Multi-Objective Tabu Search Based Topology Synthesis for Designing Power and Performance Efficient NoC Architectures. 102 p. 2011, Thesis (Masters of Applied Science in the Program of Electrical and Computer Engineering), Ryerson University, Toronto, 2011.

Bertozzi, D. et al. NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip, IEEE Transaction on Parallel and Distributed System, Piscataway: IEEE Press, Feb. 2005, pp. 113-129.

Exploração de Espaço de Projeto utilizando LM-NoC com Avaliação de Algoritmos de Roteamento

Jonathan W. de Mesquita [1], Marcos O. da Cruz[1], Monica M. Pereira[1], Ana L. Medeiros[1], Márcio E. Kreutz [1]

[1]LASIC - DIMAp - Universidade Federal do Rio Grande do Norte (UFRN)

{jonathan, marcos, monica, analuisafdm, kreutz}@lasic.ufrn.br

Abstract. *This article uses the LM-NoC simulation tool in the exploration of networks-on-chip design space, using the routing algorythm XYX, as an alternative to the traditional XY. The simulation results show that each application can benefit from a particular routing and simple changes in can result in significant performance improvements.*

Resumo. *O presente artigo utiliza a ferramenta de simulação LM-NoC na exploração de espaço de projeto de Redes-em-chip, utilizando o roteamento XYX, alternativo ao tradicional XY. Resultados de simulação mostram que cada aplicação pode se beneficiar de um determinado roteamento e, mudanças simples, podem acarretar em significativas melhorias no desempenho.*

1. Introdução

A exploração de espaço de projeto é uma etapa essencial para o projeto de arquiteturas eficientes. A implementação dessas arquiteturas em linguagens de descrição de hardware (HDL) é complexa e demorada. De acordo com [Coussy and Morawiec 2010], à medida que a tecnologia progride, o uso de abstrações de alto nível e métodos de síntese, se tornam mais e mais necessários. O uso de modelos de alto nível permite trabalhar com sistemas, ao invés de circuitos. No projeto de uma rede-em-chip, essa exploração é feita através de modificações em sua estrutura conceitual, para avaliar o seu comportamento e o seu desempenho. Dentre as modificações possiveis temos políticas de arbitragem e algoritmos de roteamento. O objetivo da exploração é otimizar o projeto de acordo com suas restrições e especificações, tais como, o consumo de potência, que, segundo [Kahng et al. 2009] tem-se tornado a restrição mais crítica em um projeto dessa natureza. Isso se dá pela sobrecarga de comunicação que uma rede-em-chip pode gerar, caso não seja bem projetada, e devido a relação direta entre comunicação e consumo [Dally and Towles 2001].

Esse artigo apresenta um estudo que demonstra o uso da ferramenta LM-NoC para exploração de espaço de projeto de redes em chip, utilizando o roteamento XYX, como alternativa ao tradicional XY. Resultados experimentais mostram que o uso do algoritmo XYX beneficia algumas aplicações com redução da latência média dos pacotes em até 50%, o que significa uma redução direta no consumo energético.

2. Trabalhos Relacionados

Muitas ferramentas foram desenvolvidas para se obter informações a respeito do desempenho de um dispositivo, ainda em fases iniciais do projeto e permitir a verificação do

efeito das otimizações em alto-nivel. A ferramenta Orion [Kahng et al. 2009] é um conjunto de modelos arquiteturais de potência para roteadores, proposto em 2002 e que foi largamente usado na indústria e na literatura, para estimativa de potência em estágios iniciais de projeto.

É possível ilustrar a natureza do espaço de projeto das arquiteturas de comunicação e também descrever uma metodologia de exploraçao que utilize algoritmos eficientes, que auxiliem a automatização de processos de mapeamento de um sistema de comunicaçao em um determinado template. É isso que [Lahiri et al. 2004] realiza em seu trabalho, além de demonstrar a importância de otimizar os protocolos de comunicação intra-chip, em vista de maximizar o desempenho do sistema.

3. LM-NoC

A LM-NoC [Mesquita et al. 2015] é uma linguagem de modelagem de domínio específico, no contexto de redes-em-chip usada para descrever os conceitos que definem uma determinada arquitetura de rede-em-chip, tais como: topologia, chaveamento, tamanho do buffer, definição do algoritmo de roteamento, dentre outros. A LM-NoC possui um editor para descrever redes em chip e a partir dessa descrição, ocorrer a geração de redes simuláveis em linguagem C++.

Para a simulação foi utilizado um gerador de tráfego que implementa o método genérico proposto por [Tedesco 2005]. O gerador define quais os roteadores de origem e destino, o tamanho dos pacotes e o intervalo em que eles são injetados na rede, em um arquivo de configuração. Durante a simulação, o arquivo de configuração é lido, e os pacotes são gerados e injetados. É gerado também um arquivo de saída contendo os dados relacionados a cada pacote: número, coordenadas (x,y) de origem e destino, ciclo de relógio em que o pacote foi injetado, o ciclo de relógio em que o cabeçalho foi recebido e a quantidades de ciclos gastos para a chegada do pacote. Obtém-se também a latência média e o tráfego aceito (número de flits por nodo por ciclo).

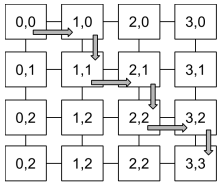
4. Roteamento Alternado XYX

O roteamento XY é um dos algoritmos determinísticos mais simples e mais utilizados [Zeferino 2003]. Nele, o pacote percorre totalmente os roteadores na linha X, até chegar a coluna onde se encontra o roteador destino. Em seguida percorre o caminho na coluna Y.

Este trabalho apresenta o roteamento XYX. A rede-em-chip é assim dividida em unidades funcionais que realizam roteamentos diferentes, dispostas alternadamente, como num tabuleiro de xadrez. O roteamento segue o algoritmo mostrado na Figura 1. Os roteadores representados pelas casas brancas, efetuam roteamento XY. Os que são representados pelas casas pretas, o roteamento YX. Desse modo, ao invés de percorrerem um movimento de “L”, os pacotes seguem uma trajetória que se aproxima de uma reta. Esse algoritmo não diminui a quantidade de roteadores no caminho. A ideia é modificar o trajeto dos pacotes e aplicá-lo, em exploração, a diferente aplicações, com diferentes padrões de tráfego, para encontrar melhorias no desempenho.

5. Resultados Experimentais

Para os testes foram usados três padrões de trafego representando três aplicações, mostrados na Figura 2. Algumas características da rede-em-chip foram mantidas fixas, e algu-




```
SE ((POSICAO_X + POSICAO_Y )% 2 == 0 )
      EXECUTAR ROTEAMENTO XY
SENÃO
      EXECUTAR ROTEAMENTO YX
```

Figura 1. Algoritmo de Roteamento XYX

mas combinações foram implementadas para obter diferentes configurações. Em todos os testes foram usadas redes Mesh-2D quadrada, de tamanho 8x8, com chaveamento por pacote. Nos dois conjuntos de testes foram usadas combinações de dois tipos de árbitros (Round Robin e Prioridade) com dois tipos de controle de fluxo (Store & Forward e Wormhole), obtendo assim, 4 combinações. Em cada uma foram testados 3 padrões de tráfego, e obtidos os valores referentes a comunicação sob os dois algoritmos de roteamento. Comparou-se, assim, a latência média dos pacotes sob o algoritmo de roteamento XY com os valores de latência média sob o algoritmo XYX.

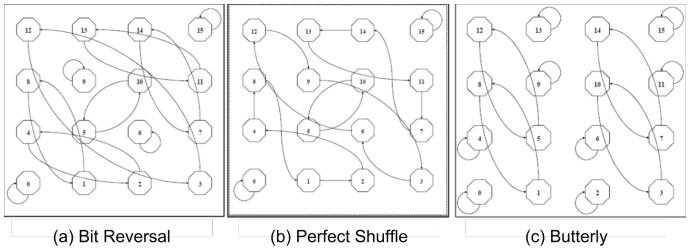


Figura 2. Padrões de tráfego

5.1. Conjunto de testes 1

O primeiro conjunto de testes utiliza três padrões de tráfego. Cada roteador gera 50 pacotes a cada 100 ciclos. Cada aplicação foi executada 6 vezes, cada uma contendo pacotes com o número de flits variando entre 50 e 500 flits. O resultado representado na Tabela 1 corresponde a diferença percentual entre as médias obtidas quando usados os dois algoritmos. Um percentual negativo significa que o algoritmo XYX obteve um resultado pior, em relação ao XY.

Table 1. Resultados Conjunto de Testes 1

	Bit Reversal	Perfect Shuffle	Butterfly
Round Robin + Wormhole	**2,59%**	-107,35%	**46,56%**
Prioridade + Wormhole	-8,16%	-106,87%	**9,01%**
Round Robin + Store & Forward	-0,68%	-95,10%	**7,35%**
Prioridade + Store & Forward	-32,69%	-102,63%	**12,20%**

Como visto nos resultados, o padrão de tráfego Butterfly é beneficiado pelo algoritmo gerando melhor desempenho, enquanto o Perfect Shuffle tem uma diminuição do desempenho. O Bit Reversal, por sua vez, na maior parte dos casos, não se beneficiou do roteamento XYX.

5.2. Conjunto de testes 2

O segundo conjunto de testes utiliza três padrões de tráfego. Cada roteador, desta vez gera 1000 pacotes a cada 100 ciclos. Cada aplicação foi executada 8 vezes, contendo pacotes com o número de flits variando entre 5 e 12 flits. O resultado representado na Tabela 2 corresponde a diferença percentual, similar ao conjunto de testes 1. Um percentual negativo significa que o algoritmo XYX obteve um resultado pior, em relação ao XY.

Table 2. Resultados Conjunto de Testes 2

	Bit Reversal	Perfect Shuffle	Butterfly
Round Robin + Wormhole	**52,21%**	-888,82%	**8,19%**
Prioridade + Wormhole	**35,87%**	-160,78%	**12,29%**
Round Robin + Store & Forward	**49,75%**	-1726,94%	**0,00%**
Prioridade + Store & Forward	**45,74%**	-945,67%	**0,00%**

Como visto nos resultados, os padrões de tráfego Butterfly e Bit Reversal foram beneficiados pelo algoritmo gerando melhor ou igual desempenho, enquanto o Perfect Shuffle tem uma diminuição do desempenho.

6. Conclusão

A exploração de espaço de projeto pode se beneficiar nas ferramentas que auxiliem como por exemplo a LM-NoC, aqui apresentada. Este trabalho realizou uma exploraçao baseada na comparaçao entre dois algoritmos de roteamento. Os resultados obtidos mostram que é possivel concluir que o algoritmo XYX pode beneficiar alguns tipos de aplicaçoes, sendo assim uma opção para obter melhoria de desempenho.

References

Coussy, P. and Morawiec, A. (2010). *High-level synthesis*. Springer.

Dally, W. J. and Towles, B. (2001). Route packets, not wires: on-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684–689. IEEE.

Kahng, A. B., Li, B., Peh, L.-S., and Samadi, K. (2009). Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, pages 423–428, 3001 Leuven, Belgium, Belgium. European Design and Automation Association.

Lahiri, K., Raghunathan, A., and Dey, S. (2004). Design space exploration for optimizing on-chip communication architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 23(6):952–961.

Mesquita, J., Medeiros, A. L., and Kreutz, M. (2015). Lm-noc: Uma linguagem de modelagem para redes em chip. *Iberchip XXI Workshop*.

Tedesco, L. P. (2005). *Uma proposta para Geração de Tráfego e Avaliação de Desempenho para NoCs*. PhD thesis, PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL.

Zeferino, C. A. (2003). Introdução às redes-em-chip. *V Escola de Microeletrônica Sul (livro texto). Porto Alegre: SBC*, pages 93–104.

Análise comparativa entre plataformas microcontroladas e de hardware reconfigurável na execução de algoritmos de apoio à detecção de melanoma

Thiago S. Marques[1], **Sidemar F. Cezario**[1]
Heliana B. Soares[2] **Monica M. Pereira**[3], **Edgard de F. Correa**[3]

[1]Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte (UFRN) – Natal, RN – Brasil

[2]Departamento de Engenharia Biomédica – UFRN – Natal, RN – Brasil

[3]Departamento de Informática e Matemática Aplicada – UFRN – Natal, RN – Brasil

{thiago.smarques.bti,sidemar.r7,helianabs}@gmail.com

{monicapereira,edgard}@dimap.ufrn.br

Abstract. *Nowadays, the development time of embedded systems projects is becoming shorter. To attend this change, a tendency is the development of a great part of these systems in software instead of the hardware, given the flexibility gain and the shorter development time. However, software components, in general, do not present the same performance of the hardware components. This paper presents the progress of the project that has the objective of building a biomedical application for the detection of cancer using a reconfigurable hardware platform.*

Resumo. *Hoje em dia, o tempo de desenvolvimento de projetos de sistemas embarcados é cada vez mais curto. Para atender a essa mudança, uma tendência é o desenvolvimento de grande parte desses sistemas em software em vez de hardware, dado o ganho de flexibilidade e tempo de desenvolvimento mais curto. No entanto, os componentes de software, em geral, não apresentam o mesmo desempenho que os componentes de hardware. Este artigo apresenta o andamento do projeto que tem como objetivo construir uma aplicação biomédica para a detecção de câncer usando uma plataforma de hardware reconfigurável.*

1. Introdução

Com o avanço das técnicas médicas no tratamento de doenças, a relação entre a medicina e a tecnologia torna-se mais estreita, pois com um aparato tecnológico avançado, pode-se detectar doenças antes mesmo delas se manifestarem.

Essas tecnologias envolvem aplicações médicas que necessitam de algoritmos específicos que possam extrair e analisar dados dos pacientes a fim de auxiliar o profissional médico a gerar um diagnóstico mais preciso e também mais precoce, o que traz diversos benefícios para o paciente, como menores gastos, expectativa de cura maior, pois a prevenção da doença, em geral, é mais barata comparado ao tratamento.

Este projeto envolve o desenvolvimento de algoritmos para aplicação biomédica de detecção de câncer usando uma plataforma com componentes heterogêneos (implementados em hardware reconfigurável e em software) na realização das técnicas necessárias. Tal plataforma híbrida permite reprogramar os algoritmos para se adaptarem a novas situações, permitindo uma maior flexibilidade.

Os algoritmos propostos estão relacionados com o método ABCD e implementados em software usando C++ e Matlab em [Soares 2008]. Esse método descreve as etapas que devem ser seguidas para classificar a lesão. Além disso, também é usado nesse trabalho um método algébrico para a análise dos dados extraídos.

Pesquisas relacionadas com detecção de câncer podem ser encontradas na literatura, tais como [Soares 2008], [Araujo et al. 2012] e [Sobieranski et al. 2007].

Este artigo está organizado da seguinte forma. A Sessão 2 apresenta a fundamentação. A Seção 3, descreve o problema a ser estudado. A proposta para a solução do problema será exposta na Seção 4. Resultados preliminares e algoritmos implementados serão apresentados na Seção 5. Considerações são dadas na Seção 6.

2. Fundamentação

Para obter-se dados a partir das imagens de lesões de pele, utiliza-se métodos e algoritmos que extraiam informações das imagens e analisem as mesmas. No projeto foi escolhido um método e um algoritmo que realizam tais funcionalidades, são eles respectivamente: ABCD proposto em [Stolz 1994] e Principal Component Analysis (PCA) [Pearson 1901].

O ABCD é um método utilizado para classificar lesões de pele pelos Especialistas/Dermatologistas. Ele é composto por etapas que descrevem uma maneira de detectar melanomas benignos e malignos. Essas etapas analisam quatro características da lesão, são elas: assimetria, bordas irregulares, coloração e diâmetro. A assimetria é analisada dividindo-se a imagem em duas partes. A imagem é assimétrica se uma metade não for igual a outra. As bordas irregulares são analisadas quanto a imperfeições, como irregularidades no contorno e no pigmento da lesão, que pode estender-se pela região vizinha. A coloração indica que a cor da lesão não é uniforme, apresentando diferentes tonalidades, e o diâmetro aponta que há uma modificação no tamanho dela.

O PCA é uma técnica algébrica usada para análise de dados. Assim, o algoritmo do PCA é capaz de determinar quais características são as mais importantes para caracterizar um conjunto de dados.

O PCA requer três etapas. Na primeira fase o conjunto de dados é fornecido e o algoritmo encarrega-se de organizar os dados na forma de uma matriz $m \times n$, onde m é número de características a serem analisadas e o n o número de amostras. Após esse processo, na segunda fase subtrai-se do conjunto de dados a média de cada característica e por último calcula-se a Decomposição em Valores Singulares (SVD) [Trefethen and III 1997] do conjunto resultante da fase 2.

O PCA contribui a medida que fornece as características mais importantes do conjunto de amostras. Consequentemente, é possível saber qual característica é a mais relevante na classificação das lesões de pele.

Tanto o ABCD, quanto o PCA requerem algoritmos que demandam um signifi-

cativo poder de processamento. Assim, para tentar melhorar a eficiência dos algoritmos, uma forma é através da implementação em hardware. Isso torna-se uma solução interessante usando uma plataforma de hardware reconfigurável, uma vez que com o auxílio da linguagem de descrição de hardware, tal processo de implementação fica menos complexo. Dessa forma, é possível gerar diversos protótipos com abordagens diferentes para uma posterior análise, a fim de se descobrir quais otimizações são mais interessantes, além de analisar o seu impacto no desempenho do sistema, no consumo de energia, etc.

3. Problema

Os algoritmos do PCA e de algumas etapas do ABCD demandam maior desempenho devido ao processamento de imagens. A implementação desses algoritmos em software permite uma maior rapidez no desenvolvimento, mas pode ocasionar um pior desempenho na execução.

É necessário, portanto, avaliar os custos de desenvolvimento e execução de ambas as soluções para os algoritmos: em software e em hardware. Os parâmetros a serem avaliados podem envolver: desempenho, consumo de energia, entre outros.

4. Proposta

Este trabalho tem o objetivo de implementar, em software e em hardware, os algoritmos do ABCD e do PCA, encontrados na literatura da área de processamento de imagens biomédica, para detecção do câncer de pele.

Assim, o fluxo de informação dos algoritmos segue a lógica apresentada na Figura 1.

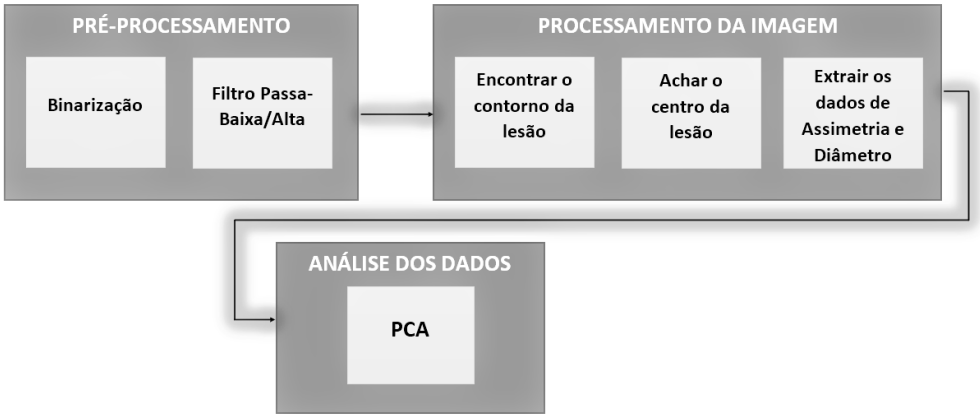


Figura 1. Fluxo dos algoritmos

É possível observar na Figura 1 as etapas que devem ser feitas para a análise da lesão. A etapa de binarização transforma uma imagem colorida em uma imagem em preto e branco, com o objetivo de identificar mais facilmente a área da lesão. Na etapa seguinte, os filtros de Passa-baixa/alta atenuam as partes indesejadas da imagem, ou seja, as que não fazem parte da região da doença. O próximo passo é encontrar o contorno da região lesionada para que seja possível analisar suas bordas e averiguar se são irregulares ou não, em seguida é preciso encontrar o centro da imagem, assim é possível comparar a

simetria e diâmetro. Por fim, a técnica algébrica do PCA é usada, sempre que as amostras satisfaçam as restrições da técnica, a fim de analisar os dados extraídos.

A metodologia a ser usada, parte da implementação dos algoritmos em software, usando a linguagem C e em paralelo a implementação em hardware, usando a linguagem VHDL.

A partir das duas versões de cada algoritmo será possível realizar a exploração do espaço de projeto, considerando o uso de algoritmos em software ou em hardware para cada uma das etapas e o seu impacto no desempenho do sistema, no consumo de energia, etc.

5. Resultados Preliminares

Atualmente estão sendo implementados algoritmos em hardware dedicado (VHDL) das etapas Análise dos Dados e Binarização. Posteriormente será efetuada a simulação desses algoritmos e comparação com versões que estão sendo implementadas alto nível (C/C++).

Essa comparação será tanto para validar os algoritmos implementados em hardware como também para avaliar se determinado algoritmo é melhor executado em software ou em hardware.

6. Considerações

Na análise de dados obtidos pelos algoritmos de processamento de imagens, nem sempre o PCA irá funcionar, partindo desse fato um método alternativo é o Independent Component Analysis (ICA) [Ans et al. 1985] que é um método mais poderoso e tem se mostrado bastante eficiente nos casos em que o PCA falha.

Referências

Ans, B., Hérault, J., and Jutten, C. (1985). Adaptive neural architectures: detection of primitives. pages 593–597. Proc. of COGNITIVA'85.

Araujo, A. F., Tavares, J. M. R. S., Oliveira, R. B., Rossetti, R. B., Marranghello, N., and Pereira, A. S. (2012). Análise e caracterização de lesões de pele para auxílio ao diagnóstico médico. Avanços em Visão Computacional.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. pages 559–572. Philosophical Magazine.

Soares, H. B. (2008). Análise e classificação de imagens de lesões da pele por atributos de cor, forma e textura utilizando máquina de vetor de suporte. Universidade Federal do Rio Grande do Norte.

Sobieranski, A. C., Coser, L., Comunelo, E., and von Von Wangenheim, A. (2007). Metodologia computacional para aplicação da regra abcd na avaliação de lesões pigmentadas. VII Workshop de Informática Médica - WIM.

Stolz, W. (1994). The abcd rule of dermatoscopy. high prospective value in the diagnosis of doubtful melanocytic skin lesion. pages 551–559. Journal American Academy Dermatology.

Trefethen, L. N. and III, D. B. (1997). Numerical linear algebra. Philadelphia: Society for Industrial and Applied Mathematics.

Ferramenta de Mapeamento em Tempo de Execução Para Arquiteturas ARM e VEX

Dogival F. Junior , Eliselma V. Santos, Monica M. Pereira

Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte - Natal, RN - Brasil

(dogival,eliselma)@lasic.ufrn.br, monicapereira@dimap.ufrn.br

Abstract. *This article describes an execution time mapping tool of instructions for ARM and VEX architectures. To guarantee the acceleration gain this tool uses a greed heuristcs with the modulo scheduling technique for rearranging the loop instructions in a pipeline fashion. Two scenarios are utilized for tests, the first one with data dependency in all instructions and the second with totaly independent instructions.*

Resumo. *Este artigo descreve uma ferramenta de mapeamento de instruções em tempo de execução para arquiteturas ARM e VEX. Para garantir o ganho de aceleração a ferramenta utiliza uma heurística gulosa aliada a técnica de modulo scheduling, utilizada para rearranjar instruções de laços em forma de pipeline. Dois cenários são usados para testes, o primeiro com dependência de dados em todas as instruções e o segundo sem dependência.*

1. Introdução

As aplicações vêm passando por um crescente aumento de complexidade, de modo que cada vez mais exigem dispositivos de hardware que possuam um alto desempenho além de flexibilidade para executar diferentes domínios de aplicações. Soluções como arquiteturas reconfiguráveis têm sido amplamente utilizadas para acelerar a execução das aplicações através de técnicas que exploram o paralelismo no nível de instruções [Compton 2002].

A maioria das arquiteturas reconfiguráveis são baseadas em um conjunto de blocos lógicos que são replicados de maneira a formar uma estrutura maior. A granularidade do bloco lógico refere-se ao tamanho e à complexidade deste bloco e pode ser fina ou grossa. A granularidade mais fina provê uma maior flexibilidade em adaptar o hardware à computação desejada. Componentes reconfiguráveis de granularidade grossa possuem diversas unidades funcionais, que executam operações à nível de palavras de bits [Compton 2002].

No contexto de sistemas embarcados, as arquiteturas reconfiguráveis são possíveis candidatas a serem usadas para acelerar aplicações. Já as arquiteturas de granularidade grossa são mais apropriadas pela economia de memória, tempo de reconfiguração e sua eficiência energética [Tanigawa. K., 2008], aspectos fundamentais no projeto de sistemas embarcados.

Esse trabalho propõe uma ferramenta de mapeamento de aplicações em uma arquitetura reconfigurável de granularidade grossa (CGRA) proposta em [Ferreira 2013] com dois conjuntos de instruções diferentes. O primeiro é o conjunto de instruções do

processador VEX [Wong 2008], sendo esse o processador original da arquitetura. O segundo é o conjunto de instruções do processador ARM [ARM 2008], uma arquitetura de processador que é utilizada principalmente em sistemas embarcados, possuindo como características um bom desempenho, de modo a ocupar pouca área e baixo consumo de energia [Dias 2012]. É importante destacar que o ARM foi escolhido nesse trabalho com o objetivo de propor um acelerador reconfigurável para ser utilizado em sistemas embarcados.

Esse artigo está organizado da seguinte forma. A Seção 2 apresenta a descrição da ferramenta. A Seção 3 apresenta os experimentos realizados e os resultados obtidos neste trabalho. Por fim, na Seção 4 são apresentadas as conclusões.

2. Descrição da Ferramenta

Para utilizar uma arquitetura reconfigurável como acelerador de execução de aplicações, é necessário um mecanismo que seja responsável por encontrar as oportunidades de paralelismo no código da aplicação e fazer o mapeamento das instruções na arquitetura para a execução em paralelo. Nesse trabalho, o mecanismo responsável por essas tarefas é a ferramenta de mapeamento.

A ferramenta de mapeamento em tempo de execução lê diretamente o código *assembly* das aplicações no momento de sua execução, analisa quais trechos de códigos podem ser acelerados e gera uma configuração equivalente a esses códigos. Depois que a ferramenta gera a configuração, esta é armazenada na memória de configuração do sistema. No momento da execução da configuração, esta é trazida da memória para o acelerador, carregada e executada. A etapa de geração de configuração é denominada mapeamento. O mapeamento consiste em selecionar as instruções *assembly* a serem aceleradas e realizar o posicionamento, o escalonamento e o roteamento dessas instruções na arquitetura. O posicionamento determina em qual unidade funcional a instrução será alocada. O escalonamento determina em que momento no tempo a instrução será executada. Por fim, o roteamento determina qual o destino dos resultados obtidos com a execução das instruções. Para realizar o mapeamento, é utilizada uma técnica para desenrolar laços chamada de *modulo scheduling* [Ferreira, 2013], uma técnica de *software pipelining* que sobrepõe diferentes iterações do laço para aumentar o grau de paralelismo a nível de instrução. Para realizar o mapeamento, é utilizada uma heurística gulosa, de forma que o posicionamento utiliza a próxima unidade livre disponível para alocar uma instrução, isso aliado ao roteamento que se utiliza de uma rede *crossbar* para comunicação entre as unidades funcionais, torna a complexidade de pior caso desse passo $O(1)$.

O algoritmo e a arquitetura para aceleração foram propostas originalmente em [Ferreira 2013]. A versão da ferramenta proposta nesse trabalho foi reescrita porém, sem mudanças no comportamento do algoritmo, agora com a adição da arquitetura ARM. Para adicionar a compatibilidade com o conjunto de instruções ARM, a ferramenta foi retrabalhada para funcionar em módulos, sendo estes: 1) o módulo de tradução ARM, que lê a entrada em *assembly* ARM e transforma para o formato compatível com a arquitetura aceleradora, e 2) o módulo de mapeamento, que recebe as instruções e mapeia de acordo com o algoritmo. Além disso, na ferramenta original, a arquitetura VEX era utilizada para ser acelerada, essa foi reimplementada na nossa versão, agora na forma de módulo.

3. Experimentos

Para validar a ferramenta de mapeamento, foram feitos experimentos utilizando a mesma aplicação compilada para o conjunto de instruções do ARM e do VEX. Inicialmente, foram utilizados duas aplicações sintéticas em dois cenários antagônicos. No primeiro todas as instruções são sequenciais, onde cada instrução depende do resultado da instrução anterior. No segundo cenário, temos todas as instruções independentes, sendo essa a situação perfeita para aplicação da aceleração. Em ambas as arquiteturas ARM e VEX foram utilizados instruções simples de *load*/*store*, *add*/*sub* e *mul* (multiplicação). A quantidade de unidades funcionais é mesma para todas as comparações sendo: 2 *load*/*store*, 12 *ALUs*, 2 *mul*.

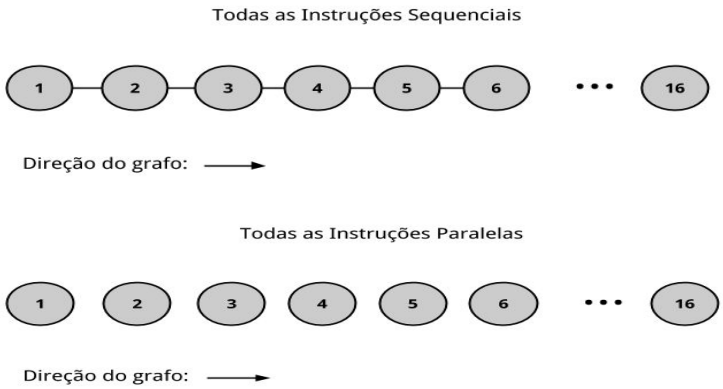


Figura 1: Grafo dos Experimentos

A Figura 2 ilustra como ficou o mapeamento das instruções nas unidades funcionais da arquitetura pela ferramenta. Ao todo, são 16 instruções numeradas de 1 a 16 e mapeadas nas unidades funcionais também numeradas, UF1 - UF16. Na Figura 2 é possível enxergar que, na aplicação I, tanto para o ARM quanto para o VEX, uma aplicação com dependência total seria o pior dos casos para nossa ferramenta. É possível observar isso pelo tempo *t* de cada instrução que é sequencial. Já que, como a próxima instrução depende da anterior, cada instrução será escalonada no tempo seguinte.

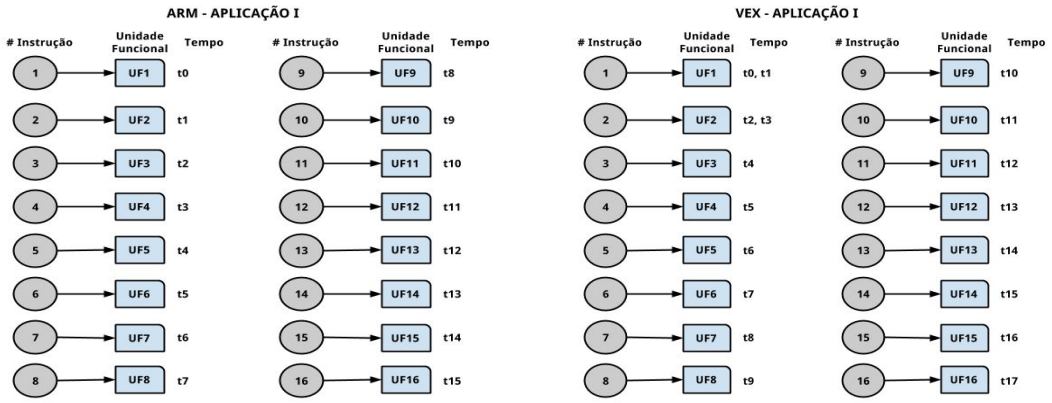


Figura 2. Teste de Mapeamento - Aplicação I: a)ARM b)VEX

Na Figura 3 observamos o cenário antagônico ao primeiro onde todas as instruções são totalmente independentes entre si. Este representa o melhor caso de mapeamento da ferramenta. Vale salientar que esse tipo de situação não é comum de

acontecer. Entre os cenários I e II, devemos ter um misto entre ambos, onde a ferramenta vai mapear as instruções em graus diferentes de paralelismo dependendo do tipo de aplicação a ser analisada. Por último vemos que a unidade de memória faz muita diferença, já que nessa configuração, usamos apenas duas unidade de *load*/*store* principalmente quando olhamos para o VEX que tem uma penalidade de dois ciclos para esse tipo de operação, ocupando a unidade funcional UF1 e UF2 no tempo t0, t1.

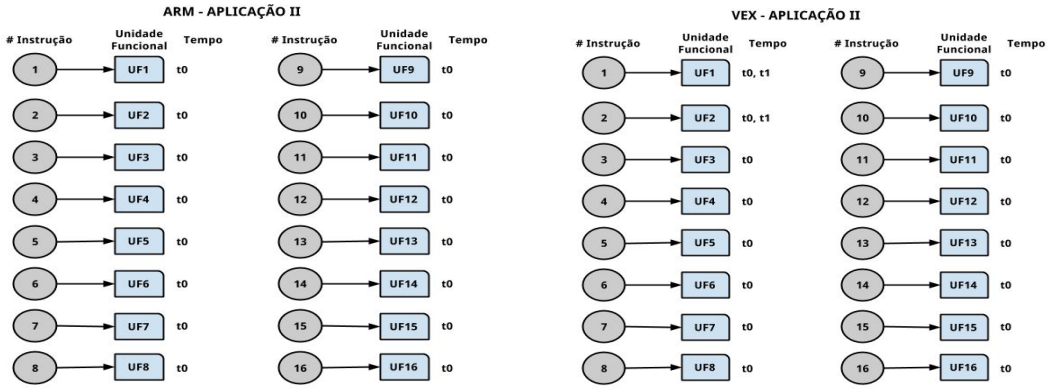


Figura 3. Teste de Mapeamento - Aplicação II: a)ARM b)VEX

4. Conclusão

Este artigo apresentou uma ferramenta para mapeamento em tempo de execução para as arquiteturas ARM e VEX, que utiliza um algoritmo eficiente para mapear laços em um CGRA que atua como acelerador. Com o experimento de dois cenários opostos quanto a dependência de dados, validamos e mostramos o potencial que a ferramenta tem para acelerar códigos com baixo grau de dependência de dados.

5. Referências

ARM, ARM. "Architecture Reference Manual (ARMv7-A and ARMv7-R edition)." ARM DDI C 406, 2008.

Compton, K. e Hauck, S. "Automatic Design of Reconfigurable Domain-Specific Flexible Cores", IEEE Transactions on VLSI Systems, pp. 493-503, v. 16, No. 5, 2008.

Dias, W. R. A.; Moreno, E. D.; das Barreto, R. Architectural Characterization and Code Compression in Embedded Processors. “Latin America Transactions, IEEE (Revista IEEE America Latina)”, v. 10, n. 4, 2012, p. 1865-1873.

Ferreira, R., et al. "A just-in-time modulo scheduling for virtual coarse-grained reconfigurable architectures." Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), International Conference on. IEEE, 2013.

Tanigawa. K., Zuyama, T., Uchida. T., Hironaka, T. “Exploring Compact Design on High Troughput Coarse Grained Reconfigurable Architectures”. Field Programmable Logic and Applications, 2008, vols. 543-546.

Wong, S.; Van As., T.; Brown, G. ρ-VEX: A reconfigurable and extensible softcore VLIW processor. In: “ICECE Technology, 2008. FPT 2008. International Conference on. IEEE”, 2008. p. 369-372.

Análise comparativa de DCT em arquiteturas de hardware reconfigurável e em GPU

Matheus Cassiano de Góes[1], Monica Magalhães Pereira[2], Bruno Motta Carvalho[2], Edgard de Faria Correa[2]

[1]Departamento de Engenharia Elétrica – Universidade Federal do Rio Grande do Norte(UFRN). Caixa Postal 1524 - 59078-970 - Natal - RN – Brasil

[2]Departamento de Informática e Matemática Aplicada – Universidade Federal do Rio Grande do Norte(UFRN). Caixa Postal 1524 - 59078-970 - Natal - RN – Brasil

cassianodegoes@gmail.com, {monicapereira, bruno, edgard}@dimap.ufrn.br

Abstract. *The digital TV transmission and video processing involve high amount of data, which may become a problem whether it demands long time and/or high resources consumption. In these situations the use of data compression algorithms becomes mandatory. The focus of the project mentioned in this work is the implementation of one of these algorithms, the DCT, in two different architectures, for comparative purpose. This article presents the proposals of the studies of similar work.*

Resumo. *Transmissões digitais de TV e processamento de vídeo envolvem grandes quantidades de dados, o que pode ser um problema caso isso demande muito tempo e/ou recursos. Nessas situações, o uso de algoritmos de compressão de dados torna-se obrigatório. O foco do projeto mencionado neste trabalho é a implementação de um destes algoritmos, a DCT, em duas arquiteturas distintas, com finalidade comparativa. Este artigo apresenta propostas de trabalhos similares.*

1.Introdução

As imagens hoje atingem enormes resoluções, de modo a torna-se imperativo o uso de algoritmos de compressão na transmissão ou no processamento gráfico em tempo real. Estes algoritmos tem como objetivo a redução da quantidade de dados a serem processados, porém sem provocar grandes perdas de qualidade.

Uma das técnicas mais utilizadas para a compressão de imagens e vídeos é a Transformada Discreta do Cosseno (DCT, do inglês *Discrete Cosine Transform*). A DCT leva um sinal ou imagem do domínio espaço-tempo para o domínio da frequência (ou espectro de frequência). Uma imagem no domínio da frequência codifica as variações de intensidades entre pixels em diversas frequências. Portanto, os coeficientes associados com as frequências presentes na imagem original são proporcionais às suas contribuições na formação da imagem. Quanto maior for a concentração de energia nas mais baixas frequências, maior será o nível de compressão alcançado. Após ser comprimida, a imagem original pode ser obtida aplicando a Transformada Discreta Inversa do Cosseno (IDCT, do inglês *Inverse Discrete Cosine Transform*). As soluções para a implementação da DCT devem aliar o desempenho do hardware e a flexibilidade do software.

A busca de equilíbrio entre essas características, por vezes antagônicas, leva a verificar alternativas de hardware com maior flexibilidade, como os hardwares reconfiguráveis, e de software com maior desempenho que exploram o paralelismo de arquiteturas, como as GPUs. O padrão H.265 apresenta, entre outras definições, as características que devem ser consideradas nos algoritmos e processos de transmissão e compressão de dados. Com base neste padrão, este trabalho propõe a implementação de um mesmo algoritmo em software paralelizado (CUDA ou OpenCV) e em hardware (VHDL) com a finalidade de comparar o desempenho entre as duas arquiteturas. Por se encontrar em estágio inicial, este artigo apresenta uma breve descrição sobre os conceitos básicos e o que se espera alcançar ao final do projeto.

Este artigo está organizado em outras 3 seções. A próxima Seção apresenta as etapas do padrão H.265/HEVC e explica a necessidade de otimização da DCT para hardwares reconfiguráveis. A Seção 3 descreve o algoritmo escolhido, e a ultima Seção aborda os trabalhos futuros.

2. Fundamentação Teórica

O padrão H.265 ou HEVC (*High Efficiency Video Coding*) é a atualização do padrão H.264/AVC utilizado atualmente pelo Sistema Brasileiro de Televisão Digital (SBTVD). Segundo Jesk [2013], a nova versão do padrão garante a mesma qualidade na compressão de imagens que o padrão anterior com economia de 35% de bytes para uma matriz 8x8. Para obter esse resultado, uma das modificações mais significativas está relacionada aos macroblocos, ou fotogramas, que são os blocos formados através do particionamento da imagem em blocos menores, identificando pixels repetidos ou de cores semelhantes. No padrão H.264 esses blocos tem tamanhos fixos. No novo modelo os blocos passaram a ter tamanhos variáveis, de modo a fazer um agrupamento mais inteligente e eficiente. O HEVC prevê diversas formas de particionamento do quadro para aumentar a eficiência da codificação, como: CTU (do inglês, *Coding Tree Units* ou Unidades de Codificação em Árvore), *slice*, *tile*, etc. Neste trabalho, focaremos nas Unidades de Transformada. Como mencionado na introdução, o padrão utiliza a DCT devido a acumulação da energia da imagem em poucos coeficientes, elevando muito a taxa de compressão. O cálculo da DCT-1D obedece a seguinte equação, que possui como componentes "p" (matriz unidimensional), "n" (tamanho da matriz unidimensional), "f" (0, 1, ..., n-1), "C_f" (coeficiente para o elemento da posição "f"), "p_t" (amostra t da matriz p) :

$$G_f = \frac{1}{2} C_f \sum_{t=0}^{n-1} p_t \cos\left(\frac{(2t+1)f\pi}{2n}\right), \text{ onde: } C_f = \begin{cases} \frac{1}{\sqrt{2}}, & f = 0 \\ 1, & f > 0 \end{cases} \text{ para } f = 0, 1, ..., n-1$$

A Figura 1 ilustra o resultado de uma matriz após o cálculo da DCT. Pode-se notar a concentração de energia na parte superior esquerda da matriz obtida como resultado. Para o cálculo da matriz, que é bidimensional, pode-se calcular através de aplicações sucessivas de DCT-1D. A Figura 2 mostra o processo de aplicação em uma matriz 2D. A DCT-1D é aplicada a matriz original, o resultado desta aplicação será outra matriz, na qual deverá ser aplicada uma nova DCT-1D, mas na direção da outra coordenada, resultando na DCT-2D da imagem original.

Para a implementação da DCT em hardware devem ser tomadas algumas medidas para maximizar a eficiência, como implementação de pipeline para explorar os ciclos de *clock*.

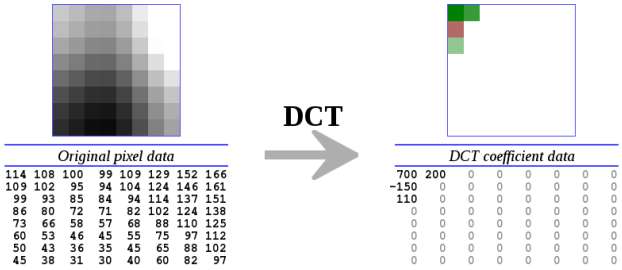


Figura 1 Na esquerda, matriz original. Na direita, resultado da aplicação da DCT.

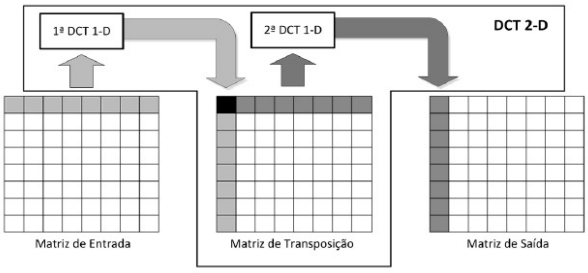


Figura 2 Aplicação da DCT para matrizes bidimensionais através de aplicação da DCT 1D duas vezes consecutivas.

3. Proposta

Na busca por soluções de alto desempenho para execução da DCT, é necessário levar em consideração diversos aspectos que vão além da velocidade de execução do algoritmo. Área de hardware e dissipação de potência são dois aspectos importantes que devem ser considerados, já que influenciam diretamente no custo do total do sistema e no seu consumo de energia. Por esse motivo, arquiteturas reconfiguráveis podem ser soluções eficientes, já que é possível projetar um hardware otimizado para a execução da DCT e sintetizá-lo em um dispositivo reconfigurável. Além desses aspectos, a programabilidade também é outro fator importante. Enquanto que a arquitetura reconfigurável é uma solução altamente otimizada, seu projeto é mais complexo e requer a implementação de toda a arquitetura em linguagem de descrição de hardware. Por outro lado, processadores de alto desempenho, como é o caso das GPUs (*Graphics Processing Unit*) utilizadas para processamento gráfico, são arquiteturas de prateleira, mais otimizadas do que os processadores de propósito geral, e que podem ser programadas através de uma linguagem de programação de alto nível, como é o caso da linguagem CUDA das GPUs da fabricante Nvidia [2009].

Considerando os fatores expostos acima, esse trabalho tem o objetivo de realizar uma comparação entre a solução otimizada, que utiliza hardware reconfigurável, e a solução de prateleira, que utiliza a GPU como processador de alto desempenho. Com o objetivo identificar qual a solução mais apropriada, considerando, além do desempenho, a área, dissipação de potência, programabilidade, dentre outros aspectos.

Para que a comparação entre a arquitetura de hardware reconfigurável e a implementação em GPU seja feita, ambos dispositivos devem executar o mesmo algoritmo. Um algoritmo que soluciona este problema foi proposto por Aray et al. [1988] e modificado por Kovac et al. [1995]. O mesmo é um método de cálculo da DCT-1D para um vetor de dimensão 8 com apenas 5 multiplicações e 28 adições. Este

algoritmo é escalonado e possui pipeline implícito. O algoritmo é dividido em 6 passos distintos e independentes. No quarto passo existem apenas multiplicações, enquanto os demais possuem apenas adições. Segundo Agostini e Bampi [2001], as entradas do algoritmo são a0, a1, ..., a7 e os coeficientes calculados da DCT 1D são S0, S1, ..., S7. Nas multiplicações, os valores dos multiplicandos m1, m2, m3, m4 são constantes e possuem, respectivamente, os valores: cos(4p 16), cos(6p 16), cos(2p 16)- cos(6p 16) e cos(2p 16)+ cos(6p 16). Na sequência estão descritas operações de todos os passos:

Passo 1:	**Passo 2:**	**Passo 3:**	**Passo 4:**	**Passo 5:**	**Passo 6:**
b0=a0 + a7	c0=b0 + b5	d0=c0 + c3	e0= d0	f0= e0	S0= f0
b1=a1 + a6	c1=b1 - b4	d1=c0 - c3	e1= d1	f1= e1	S1= f4 + f7
b2=a2 - a4	c2=b2 + b6	d2= c2	e2=m3* d2	f2= e5 + e6	S2= f2
b3=a1 - a6	c3=b1 + b4	d3=c1 + c4	e3=m1* d7	f3= e5 - e6	S3= f5 - f6
b4=a2 + a5	c4=b0 - b5	d4=c2 - c5	e4=m4* d6	f4= e3 + e8	S4= f1
b5=a3 + a4	c5=b3 + b7	d5= c4	e5= d5	f5= e8 - e3	S5= f5 + f6
b6= a2 - a5	c6=b3 + b6	d6= c5	e6=m1* d3	f6= e2 + e7	S6 = f3
b7= a0 - a7	c7= b7	d7= c6	e7=m2* d4	f7= e4 + e7	S7=f4-f4
		d8= c7	e8 = d8		

A partir da identificação do algoritmo, o próximo passo desse trabalho consiste em implementá-lo primeiramente em GPUs.

4. Conclusão

Este artigo descreveu brevemente a proposta de trabalho para realizar a comparação entre as soluções em hardware reconfigurável e em GPU do algoritmo da DCT. A próxima etapa do projeto prevê a implementação do algoritmo em uma arquitetura GPU, utilizando linguagem CUDA ou OpenCV, plataformas para o desenvolvimento de aplicações massivamente paralelas. Essa implementação será feita em pareceria com o laboratório IMAGINA do DIMAp, que atua com processamento de imagensO passo seguinte consiste em implementar o mesmo algoritmo em linguagem de descrição de hardware para realizar a comparação entre as duas soluções.

5. Referências

Agostini, L. e Bampi, S. Projeto de uma Arquitetura de DCT 1D para a Compressão de Imagens JPEG. In: VII Workshop IBERCHIP, 2001, Montevidéo: Uruguai. P. 2-3.

Jeske, R. G. Otimizações algorítmicas e desenvolvimento arquitetural para as DCTs do HEVC / Ricardo Garcia Jeske; Luciano Volcan Agostini, orientador; Júlio Carlos Balzano de Matos, coorientador. Pelotas, 2013. P.81.

Arai, Y., Agui, T., Nakajima, M. A Fast DCT-SQ Scheme for Images. Transactions of IEICE. Vol. E71, No. 11, 1988. P.1095-1097.

Kovac, M., Ranganathan, N. Jaguar: A Fully Pipeline VLSI Architecture for JPEG Image Compression Standard. Proceedings of the IEEE. Vol. 83, No. 2, Fevereiro 1995. p.247-258.

NVIDIA, NVIDIA CUDA Programming Guide, 2009. http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docsNVIDIA_CUDA_Programming_Guide_2.3.pdf

Uma Introdução a Criptografia: O Algoritmo RSA

Tamara T. Melo[1], Iago A. Dantas[2], Robson P. Sousa[3]

[1] Departamento de Informática e Matemática Aplicada (DIMAP) - Universidade Federal do Rio Grande do Norte (UFRN) - Natal - RN - Brasil

[2,3] Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN) - Mossoró – RN – Brasil

tamara-tavares@hotmail.com, iago.andrade1992@hotmail.com, robson.sousa@ifrn.edu.br

Abstract. *Cryptography is essential for the exchange of confidential information. In a globalized world where the Internet is the focus of attention, the process to become a message incomprehensible, that was originally written with clarity, is extremely important. In this paper we will discuss one of the most important public-key cryptosystems, RSA. Considered one of the most powerful cryptographic methods and commercially successful ever created, its operation is based on computational impossibility of factoring large integers.*

Resumo. *A criptografia é indispensável para a troca de informações confidenciais. Em um mundo globalizado onde a internet é o foco das atenções, o processo para tornar incompreensível uma mensagem que originalmente foi escrita com clareza é extremamente importante. Neste trabalho, abordamos um dos mais importantes sistemas criptográficos de chave pública, o RSA. Considerado um dos métodos criptográficos mais poderosos e comercialmente bem sucedidos já criados, seu funcionamento é baseado na impossibilidade computacional de fatorar números inteiros grandes.*

1. Introdução

Usada ao longo da história para enviar mensagens confidenciais, a criptografia tem como objetivo fazer com que apenas pessoas autorizadas possam compreender a mensagem enviada.

A criptografia surgiu há milhares de anos e é tão antiga quanto a própria escrita. Porém, foi somente após a Segunda Guerra Mundial, com a evolução da informática, que a tecnologia da criptografia apresentou grandes avanços incorporando complexos e sofisticados algoritmos matemáticos **[**Silva 2014**]**.

Neste trabalho apresentamos uma abordagem sobre a criptografia de chave pública, mais particularmente, trataremos de um dos métodos mais importantes de criptografia, conhecido como RSA.

O RSA foi criado em 1977 por pesquisadores do Massachussets Institute of Technology – MIT, e é uma das mais fantásticas aplicações da Teoria dos Números. O método é baseado na premissa de que embora seja fácil encontrar dois números primos de grandes dimensões, fatorar o produto destes dois números é uma atividade computacionalmente complexa [Melo 2014].

2. O Método da Criptografia RSA

O RSA é resultado do aperfeiçoamento do sistema de criptografia de chave pública desenvolvido nos anos 70 por Diffie e Hellman. Sendo hoje uma das mais bem sucedidas implementações do método de criptografia de chave pública, o RSA é o algoritmo mais utilizado do mundo. Por ser um método de chave pública, qualquer indivíduo pode codificar mensagens, mas somente a pessoa autorizada poderá decodificá-la.

Atualmente o RSA é implantado em muitos sistemas comerciais, e é usado por servidores e navegadores para proteger o tráfego da web, sendo utilizado também para assegurar a privacidade, a autenticidade do e-mail e para proteger sessões de login remoto, é considerado o coração dos sistemas de pagamento de cartão de crédito eletrônicos [Boneh 2014].

2.1. Método de Operação

Primeiramente é feita a ***pré – codificação***. Nessa etapa o propósito é transformar as letras em números inteiros positivos de dois algarismos. Para facilitar os cálculos, podemos usar a seguinte correspondência**:**

Tabela 1. Pré-condificação das letras do alfabeto português

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
11	12	13	14	15	16	17	18	19	20	21	22	23
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
24	25	26	27	28	29	30	31	32	33	34	35	36

Vamos considerar uma frase do grande filósofo e matemático **Platão**: ***Os números governam o mundo***. Convencionando que cada espaço entre as palavras seja convertido no número 10. A frase ficará representada pelo número:

$$M = 25291024312315282529101725321528241123102510233124 1425$$

Agora, precisamos usar os parâmetros do sistema RSA, que são dois números primos distintos denotados por p e q. Fazendo $n = pq$, a última etapa da pré-codificação consiste em separar o número encontrado em blocos, cujos valores sejam menores do que n. Uma alternativa seria escolher $p = 11$ e $q = 17$, obtendo assim, $n = 187$. Podemos então, organizar os blocos da seguinte forma:

$$2 - 52 - 9 - 102 - 43 - 12 - 3 - 15 - 2 - 82 - 5 - 29 - 10 - 17 - 25 - 3 - 21 - 52 - 82 - 4 - 11 - 23 - 102 - 5 - 10 - 2 - 33 - 12 - 4 - 14 - 25$$

Passaremos agora para o processo de ***codificação***.

Nesta nova etapa, precisamos de n, definido anteriormente, e de um inteiro positivo e de modo que :

$$mdc(\varphi(n), e) = 1 \text{ e } 1 < e < \varphi(n)$$

Se n é primo, como de 1 a n o único elemento que não é primo com n é o próprio n, então $\varphi(n) = n - 1$.

Por outro lado, temos que:

$$\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1).$$

Ao par de inteiros (n, e) chamaremos de ***chave de codificação*** do sistema RSA .

Inicialmente, vamos indicar a codificação de um bloco b por $C(b)$. Onde, $C(b)$ é o resto da divisão de b^e por n. Para esse calculo, denotaremos:

$$C(b) \equiv b^e (mod\ n)$$

Assim, temos: $p = 11$, $q = 17$, $n = 187$ e $\varphi(n) = (11-1)(17-1) = 160$. E escolhendo $e = 3$, temos $mdc(160,3) = 1$ e $1 < e = 3 < 160$. Logo $(n, e) = (187, 3)$ é a nossa chave de codificação.

Vamos então à codificação. O bloco $b_1 = 2$ da mensagem anterior é codificado da seguinte forma:

$$C(2) \equiv 2^3 (mod\ 187).$$

Daí temos: $C(2) = 8$. Executando o mesmo processo nos demais blocos, obtemos a seguinte mensagem codificada:

$$8-171-168-170-32-45-27-9-8-92-125-79-65-51-104-27-98-171-92-64-22-12-170-125-65-8-33-45-64-126-104$$

Assim,

$$N = 817116817032452798921257965511042798171926422121701256583345641261 04$$

é a nossa mensagem codificada.

Passaremos agora ao processo de decodificação. Para isso precisaremos de n e o inverso de e em $\varphi(n)$ que denotaremos por d. Ao par de números inteiros (n, d) chamaremos de ***chave de decodificação***. Conhecendo e e $\varphi(n)$ e aplicando o algoritmo de Euclides podemos encontrar d, além disso, precisamos que $D(C\ (b)) = b$.

Supondo que a seja um bloco da mensagem codificada, $a = C(b)$, o processo de decodificação se dá da seguinte maneira:

$$D(a) \equiv a^d (mod\ n);$$

ou seja, $D(a)$ é o resto da divisão de a^d por n.

Usando o algoritmo de Euclides, encontraremos d calculando o m$dc(\varphi(n), e) = mdc(\varphi(187), 3) = mdc(160,3)$, assim obtemos:

$$160 = 53 \cdot 3 + 1 \rightarrow 1 = 160 + (-53) \cdot 3$$

Logo o inverso de 3 módulo 160 é -53. Mas, precisamos que d seja positivo. Então, $d = 160 - 53 = 107$. Portanto, obtemos $(n, d) = (187,107)$.

Como exemplo, vamos decodificar o sexto bloco, $C(12) = 45$, temos que:

$$D(45)) \equiv 45^{107} (mod\ 187).$$

Portanto, queremos determinar o resto da divisão de 45^{107}por 187. Por congruência, segue que

$$12 \equiv 45^{107}(mod\ 187).$$

Logo, 12 é o nosso bloco original e, de forma análoga, conseguiremos decodificar os demais blocos obtendo assim a sequencia numérica inicial.

$$M = 25291024312315282529101725321528241123102510233124142 5.$$

e consequentemente, a mensagem original: ***Os números governam o mundo.***

Note que, o método de fato funciona e é seguro, pois se p e q forem primos de grande dimensão, consequentemente $n = pq$ será maior ainda. Como precisamos fatorar n em fatores primos. Dependendo do tamanho de n essa tarefa é impossível com os métodos de fatoração que hoje conhecemos.

3. Conclusão

Diante do que foi estudado neste trabalho, observamos que a busca pela comunicação de forma cada vez mais segura é o que move estudos e pesquisas em criptografia desde a antiguidade, tornado-a tão importante e estratégica para a humanidade.

Sendo facilmente implementado, o RSA é teoricamente simples, o obstáculo é de natureza tecnológica, pois em geral são utilizadas como chaves de codificação, números muito grandes. Fatorar um número inteiro n para encontrar números primos p e q com os métodos atuais poderia levar muitos anos. Sendo assim, a segurança do RSA é dada pela ineficiência dos métodos de fatoração atualmente conhecidos [Coutinho 2013].

Referências

Boneh, D. (2014) “Twenty Years of Attacks on the RSA Cryptosystem”. Disponível em: <http://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf>.

Coutinho, S. C. (2013) “Números Inteiros e Criptografia RSA”, Rio de Janeiro IMPA - SBM.

Silva, A. A. (2014) “Números, Relações e Criptografia”. Disponível em: <http://www.mat.ufpb.br/sergio/provas/me_i/livro-andrade.pdf>.

Melo, T. T. (2014) **“**Uma Introdução a Criptografia: O Algoritmo RSA”. 58 f. TCC (Graduação) - Curso de Licenciatura Plena em Matemática, Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, Mossoró.

Uma hibridização do Algoritmo Papílio com o Algoritmo Genético para a criptografia de imagens.

Luiz Ranyer[1], **Bénjamin Bedregal**[1], **Isaac Oliveira**[2]

[1]Grupo de Lógica Linguagem Informação Teoria e Aplicação (LoLITA)
Departamento de Informática e Matemática Aplicada (DIMAp)
Universidade Federal do Rio Grande do Norte (UFRN)
59.072-970 – Natal – RN – Brasil.

[2]Departamento de Ciências da Computação
Universidade do Estado do Rio Grande do Norte (UERN)
59120-200 – Natal – RN – Brasil.

ranyer@ppgsc.ufrn.br, bedregal@dimap.ufrn.br, isaacoliveira@uern.br

Abstract. *This article describes a proposal for a generalization of Papilio encryption algorithm with genetic algorithm (GA) in order to promote greater security in the image protection process. This PapílioGA combination will be compared with other hybrid algorithms, including the algorithm "BLOWGA"proposed by Sandeep, as its performance.*

Resumo. *Este artigo descreve uma proposta de generalização do algoritmo de criptografia Papílio com o algoritmo genético (AG), a fim de promover uma maior segurança no processo de proteção de imagens. Esta combinação PapílioGA será comparada com outros algoritmos híbridos, dentre eles o algoritmo "BLOWGA"proposto por Sandeep, quanto a sua performance.*

1. Introdução

O homem utiliza a criptografia por questões de privacidade, seja para obter uma vantagem sobre alguma informação ou pelo simples fato de protege-la de intrusos. Neste contexto surge a criptografia e seus algoritmos, que podem ser classificados simétricos ou assimétricos. O método simétrico, utiliza apenas uma chave, cuja a função é cifrar e decifrar. Já os algoritmos assimétricos, utilizam-se de duas chaves distintas, uma para cifrar e outra para decifrar, também conhecidas como sistema de chave privada e chave pública. Neste caso, dentre eles, o Papílio, utilizado neste trabalho, é um algoritmo de criptografia simétrico, baseado em rede Feistel.[Araujo et al. 2005].

Com a facilidade que a internet trouxe para o mudo dos negócios, hoje é imprescindível tratar de forma segura informações pela web, onde, algumas empresas tornaram alguns processos cotidianos mais ágeis e conseguiram reduzir custos, com o envio de projetos gráficos, documentos e informações sensíveis. No ambiente web é fundamental garantir a privacidade e a segurança destas imagens e documentos.

Neste sentido, existem alguns trabalhos, por exemplo, [Da Silva B. 2013], onde utilizam-se cifras simétricas e assimétricas sobre imagens. Devido ao grande compartilhamento de imagens que acontecem na web, faz-se necessário pesquisar e integrar as principais técnicas de proteção para melhorar a proteção destes arquivos.

Logo, neste artigo, propõe-se a utilização do algoritmo Papílio em conjunto com um algoritmo genético, de forma similar ao trabalho por [Bhowmik e Acharyya 2011]. Desta forma é possível desenvolver um algoritmo hibrido do PAPÍLIOGA que poderá ser utilizado em criptografia de imagens.

Com o desenvolvimento do Papílio, é interessante e necessário a realização de comparações do algoritmo Papílio, com o Blowfish (utilizado em [Bhowmik e Acharyya 2011])

e também o Algoritmo Genético para criptografia de imagens, onde as mesmas podem ser definidas, por exemplo, em blocos de 2x2, 3x3, 5x5 pixels, como encontrado em [Bhowmik e Acharyya 2011]. Desta forma a comparação será estruturada de forma a verificar se o algoritmo Papílio, desenvolvido por Ramos [Ramos 2002], tem um desempenho melhor quando combinando com o Algoritmo Genético (AG), conforme foi desenvolvido por [Bhowmik e Acharyya 2011], onde a combinação desses algoritmos pode melhorar significativamente a eficácia dos resultados em criptografia de imagens.

2. Papílio, Algoritmo Genético e Algoritmos Vinculados.

Nesta seção serão apresentados os algoritmos Papílio e Genético que serão utilizados para o desenvolvimento do algoritmo híbrido do Papílio. Existem alguns algoritmos em criptografia, que podem ser integrados em criptografia de imagem juntamente com Algoritmo Genético. Por exemplo, o RSA [Rivest et al. 1978], utiliza a implementação de sistemas de chaves assimétricas mais segura, baseada em teorias clássicas dos números.

Já o AES [Daemen e Rijmen 2002], funciona utilizando rodadas (Rounds) que dependem do tamanho da chave. Neste caso, o algoritmo possui uma chave principal e, a partir dela, são geradas N_r + 1 chaves, que são denominadas chaves de rodada, pois cada uma será usada em uma rodada diferente. Em cada etapa, são executados substituições e transposições. Neste caso, são realizadas a Permutação de bytes entre grupos, a substituição usando matrizes dos grupos, a execução de um XOR com a chave e finalmente a substituição de bytes.

Por fim existe outra cifra relevante em criptografia. O RC4 [Akgun et al. 2008], consiste em utilizar um *array* que a cada utilização tem os seus valores permutados, e misturados com a chave, o que provoca que seja muito dependente desta. Esta chave, utilizada na inicialização do *array*, pode ter até 256 bytes (2048 bits), embora o algoritmo seja mais eficiente quando é menor, pois a perturbação aleatória induzida no *array* é superior.

2.1. Papílio

O Papilio é um algoritmo de criptografia que utiliza a cifra Feistel onde a função F é uma função computada pelo algoritmo Viterbi Modificado. De acordo com [Araujo et al. 2005] os parâmetros são codificados em função das taxas de n/s é taxa de codificação, que é a razão entre a quantidades de bit que entra n e a quantidade de bit que sai s em um dado ciclo do codificador, Q é a indicação de quantas saídas futuras do decodificador, a entrada atual irá influenciar, e m é a indicação da profundidade do pipiline do codificador, e pode ser entendido como o comprimento da memória do codificador e os polinômios geradores [1]. Atualmente, os blocos de qualquer tamanho de bits podem ser utilizados. Além disso, o tamanho da chave é de 128 bits e pode variar em cada ciclo, onde o número de ciclos (*round*) pode variar entre 0 e 16. O processo de criptografia do Papílio fala da estrutura Feistel que são vários ciclos (Rounds).

A estrutura para decifragem do Papílio obedece inversamente o processo de criptografia. Neste caso as sub-chaves [2] são utilizadas em ordem inversa. Existe uma função F para cada (*round*) no Papílio para ambos os processos. Esta função F é a mesma para todos os ciclos e é considerada o componente principal do presente processo.

Os processos de criptografia e decriptografia sempre começam com a divisão do bloco de texto $m/2$. A parte direita é usada como entrada da função F, e a parte esquerda é usada junto com o texto de saída da função F em uma operação XOR. Esta operação XOR de saída é a função de entrada de F na próxima rodada e assim por diante até o último (*round*).

Diante dos algoritmos existentes na literatura, os critérios de avaliação destes algoritmos passam pelo tamanho do texto, após o processo de criptografia, pelo tamanho da chave utilizada

[1] definidos pelos componentes básicos do codificador: registradores de deslocamento e módulos somadores, são determinados os polinômios do par de bits da saída. [Araujo 2003]

[2] subchaves geradas a partir da chave inicial informada pelo usuário.

e pela complexidade do algoritmo que pode ser mensurada através de processo de criptoanálise [3]. Nesse sentido o Papílio te como característica o tamanho (em bits) do texto cifrado resultante é o mesmo do texto original. Em relação em sua complexidade o Papílio apresenta bons resultados quando comparado com outras cifras como visto em [Oliveira Filho 2010] o que significa uma vantagem do método do Papílio.

2.2. Algoritmo Genético - AGs

Os algoritmos genéticos formam uma classe de algoritmos destinados a encontrar soluções para problemas de otimização. AGs são uma classe particular de algoritmos evolutivos que usam técnicas inspiradas pela biologia evolutiva como herança, mutação, seleção natural e recombinação (ou cruzamento). A seguir será apresentado o algoritmo genético utilizado em [Bhowmik e Acharyya 2011]. Descrição do algoritmo genético.

2.2.1. Algoritmo Genético Sandeep

Com base no trabalho de [Bhowmik e Acharyya 2011], ao utiliza-se de uma chave adequada de forma a reorganizar um bloco de n-pixels, é necessário usar uma chave de comprimento n. Esta chave é na verdade um conjunto de n números em $n!$ de possíveis padrões.

Exatamente neste ponto, pode-se aplicar a técnica de busca avançada chamada como Algoritmo Genético para procurar um padrão no espaço de busca.Em [Bhowmik e Acharyya 2011], a busca concentra-se no melhor cromossomo que pode afetar a correlação dos pixels da imagem, onde é considerado um cromossomo de tamanho 16. Nos testes, o cromossomo é representado como uma sequência de 16 inteiros em um arranjo aleatório de $16!$ de possíveis opções, cada número inteiro no cromossomo está entre 1 e 16.

A função *fitness* dos cromossomos é calculada a partir da soma da variação total (comprimento do deslocamento) de cada gene. O objetivo é maximizar o valor da função *fitness* de modo que o cromossomo aplicado para embaralhar os pixels da imagem consiga diminuir as correlações existentes entre os pixels. A descrição do algoritmo pode ser vista detalhadamente em [Bhowmik e Acharyya 2011].

A pesquisa de [Bhowmik e Acharyya 2011] mostrou que o processamento inicial na imagem original pode melhorar a eficácia de qualquer algoritmo de criptografia significativamente.

3. Proposta

Uma imagem utilizada, é encriptada com Algoritmo Genético (AG), Blowfish e BlowGA. Neste artigo, a proposta é encriptar esta imagem utilizando o BlowGA, Papílio e PapílioGA e o PapílioGA proposto. Desta forma, o desempenho será comparado em função dos coeficientes de correlação [4] entre os pixels da imagem. A Figura 1 mostra a estrutura utilizada para compor o Algoritmo PapílioGA.

[3] técnica utilizada para analisar uma cifra durante seu desenvolvimento, [Oliveira Filho 2010]

[4] Média estatística do estado de uma imagem visto em [Bhowmik e Acharyya 2011].

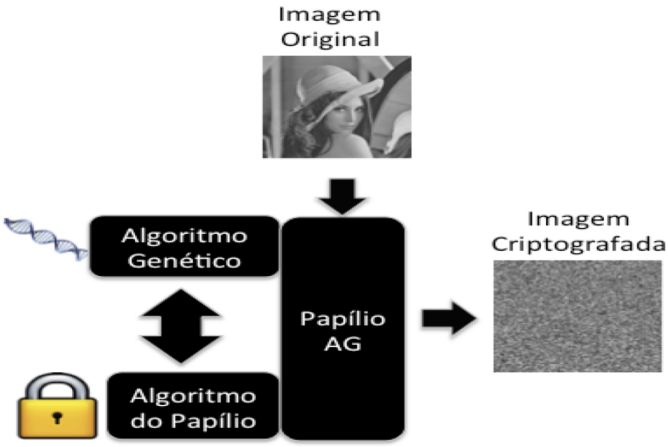


Figura 1. Estrutura Proposta.

4. Conclusão

Diante dos bons resultados do Papílio, apresentados nos trabalhos [Ramos 2002], [Araujo et al. 2005] [Neto 2009], [Oliveira Filho 2010],observou-se a viabilidade da utilização do mesmo em criptografia de imagens. Deste modo, esta nova abordagem pode ser um avanço e uma nova forma para alcançar uma melhora no nível de proteção de imagens com a utilização do PapilioGA.

Referências

Akgun, M., Kavak, P., and Demirci, H. (2008). New results on the key scheduling algorithm of rc4. In Chowdhury, D., Rijmen, V., and Das, A., editors, *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 40–52. Springer Berlin / Heidelberg.

Araujo, F. (2003). Papílioxp - uma exensão do algoritmo criptográfico papílio. Technical report, Universidade Federal do Rio Grande do Norte - UFRN.

Araujo, F. S.and Ramos, K. D., Bedregal, B. R., and Silva, I. S. (2005). Papílio cryptography algorithm. In *Computational and Information Science*, volume 3314 of *Lecture Notes in Computer Science*, pages 928–933. Springer Berlin Heidelberg.

Bhowmik, S. and Acharyya, S. (2011). Image cryptography: The genetic algorithm approach. In *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*, volume 2.

Da Silva B., Almeida H.P., O. C. C. e. O. D. C. (2013). Aplicação de técnicas de criptografia de chaves assimétricas em imagens.

Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer.

Neto, J. B. S. L. (2009). Papílio versátil: um algoritmo criptgráfico. Technical report, Universidade Federal do Rio Grande do Norte - UFRN.

Oliveira Filho, I. L. (2010). Criptoanálise diferencial do papílo. Master's thesis, Universidade Federal do Rio Grande do Norte - UFRN.

Ramos, K. D. N. (2002). Papílio: Proposta de um algoritmo de criptografia baseado no algoritmo viterbi e codificação convolucional. Master's thesis, Universidade Federal do Rio Grande do Norte - UFRN.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126.

Problema de Planejamento em Produção de Recursos em Tempo Real

Caio Freitas de Oliveira[1], **Elizabeth Ferreira Gouvêa Goldbarg**[1]

[1]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN)
Natal – RN – Brazil

caiofreitaso@gmail.com, beth@dimap.ufrn.br

Abstract. *This paper presents a new task scheduling problem based on real-time strategy games (RTS). Due to lack of models used in RTS that include peculiarities of a StarCraft race, a new model is proposed based on one that has already been applied successfully to other races.*

Resumo. *Este trabalho apresenta um novo problema de escalonamento de tarefas, com base em jogos de estratégia em tempo real (RTS). Devido à falta de modelos aplicados em RTS que incluam peculiaridades de uma raça de StarCraft, um novo modelo é proposto com base em um que foi já aplicado com sucesso às outras raças.*

1. Introdução

Jogos de estratégia em tempo real (RTS, em inglês) possuem um ambiente que motiva o jogador a criar planos de ação para derrotar seu oponente. Estes planos envolvem desde a criação de trabalhadores e construções para impulsionar a economia do império, até criação de unidades militares e as ordens que elas receberão. Os trabalhos envolvendo planejamento em jogos RTS tiveram como foco o desenvolvimento econômico do jogador dentro da partida, especialmente na sua fase inicial [Chan et al. 2007b, Chan et al. 2007a, Branquinho and Lopes 2010, Blackford 2014].

No contexto de planejamento em jogos RTS, os recursos de um jogo podem ser construções, unidades militares e civis, ou mesmo tecnologias. Para produzir um soldado, por exemplo, é necessário possuir (e usar) uma estrutura especializada (quartel), a qual, por sua vez, precisa de recursos brutos e unidades civis para ser construída. Cada ação possui uma duração. Assim, o objetivo do planejamento é encontrar a sequência que permita ao jogador alcançar um determinado objetivo no menor tempo possível.

Neste artigo, busca-se expandir o domínio do modelo apresentado em [Chan et al. 2007b] do ponto de vista da otimização multiobjetivo, em que cada recurso pode fazer parte tanto de um objetivo como de restrições. Este trabalho está dividido em três seções. Na próxima seção o novo modelo é descrito. A terceira seção mostra as conclusões e trabalhos futuros envolvendo este novo modelo.

2. Problema de Planejamento para Produção de Recursos

O problema é definido por três variáveis: o sistema S, o estado inicial I_0 e o modelo de otimização O. O sistema S é responsável por informar quantos recursos e ações existem

e como eles interagem entre si. O estado inicial I_0 informa a quantidade de recursos disponíveis bem como o andamento das ações não-concluídas. O modelo O informa quais são as restrições e objetivos referentes aos recursos.

De uma forma mais detalhada, o sistema S pode ser definido por três conjuntos: (a) o conjunto de recursos R, (b) o conjunto de tarefas T e (c) o conjunto de eventos E. Recursos representam os objetos e valores que são manipulados pelas tarefas durante o jogo. As tarefas são as ações que o jogador pode tomar para modificar os recursos. Já os eventos são modificações periódicas que são iniciadas ao criar um tipo de recurso.

2.1. Recursos

Os recursos possuem atributos que determinam algumas restrições sobre o quanto pode-se produzir. Seus atributos são:

- *Máximo produzível*: determina quantos recursos podem ter sido produzidos ao todo a qualquer momento do jogo. Precisa-se, portanto, saber quantos recursos foram produzidos até o momento.
- *Máximo utilizável por recurso*: indica a quantidade máxima de recursos utilizáveis que pode ter a qualquer estado. É preciso saber quantos foram gastos, para dizer quantos recursos utilizáveis existem em um determinado estado.
- *Equivalência*: é utilizada para determinar quando recursos equivalem, no papel de pré-requisitos de tarefas, que serão explicadas mais adiante.

Para facilitar a manipulação de recursos, três atributos são usados para determinar os recursos em um estado I_t.

- $C_r(I_t)$ indica o total produzido de um recurso r do estado I_0 ao estado I_t.
- $u_r(I_t)$ indica o total consumido de um recurso r do estado I_0 ao estado I_t.
- $U_r(I_t) = C_r(I_t) - u_r(I_t)$ indica a quantia utilizável de um recurso r nas mesmas condições.

Utilizando estes operadores, pode-se afirmar mais formalmente que qualquer recurso $r \in R$, em qualquer estado I_t, deve atender às seguintes restrições:

$$C_r(I_t) \leq M_r$$
$$U_r(I_t) \leq \sum_{s \in R}^{s \neq r} L_{rs} U_s(I_t)$$

Onde M_r indica o máximo produzível de r, e L_{rs} indica o máximo utilizável do recurso r por unidade do recurso s.

2.2. Eventos

Ao criar alguns recursos, pode-se ativar eventos cíclicos de produção de recursos, do mesmo tipo ou não. Eventos possuem três atributos:

- *Tempo de espera*: indica o período de cada ciclo do evento.
- *Produção*: é a quantidade de recursos produzidos a cada ciclo. Esta produção altera o C_r de cada recurso r gerado.
- *Gatilho*: o recurso que, quando criado, gera um novo evento deste tipo.

Ao consumir uma unidade de um tipo de recurso que produza eventos, o evento que demorará mais para ser terminado também é destruído no processo, de forma que para qualquer evento $e \in E$, qualquer estado I_t satisfaça à restrição: $|V_e(I_t)| = U_r(I_t)$. Onde V_e é o conjunto de todos os eventos do tipo e ativos no estado I_t. $U_r(I_t)$ é a quantia utilizável do recurso r, tal que r seja o gatilho do evento e.

2.3. Tarefas

As tarefas, ou ações, possuem relações com os recursos, ao invés de relações entre si, como os próprios recursos possuem. Além destas relações, tarefas possuem o atributo "duração". Ao todo, os atributos das tarefas são as seguintes:

- *Duração*: indica quanto o tempo entre o início da tarefa e sua conclusão.
- *Pré-requisito*: indica a quantidade de recursos específicos que é necessária para o início desta tarefa. Estes recursos não são usados. As equivalências de recursos são verificadas apenas neste contexto.
- *Custo*: é a quantidade de recursos a serem consumidos no início da tarefa.
- *Consumo*: é a quantidade de recursos a serem consumidos no término da tarefa. Estes recursos são imobilizados pela duração da tarefa e não poderão ser utilizados ou consumidos neste período.
- *Uso*: indica a quantidade de recursos que estarão imobilizados até o fim da tarefa porém sem serem consumidos. Estes recursos não poderão ser utilizados ou consumidos por outras tarefas.
- *Produção*: indica a quantidade de recursos que será gerada ao fim da tarefa.

Os atributos "custo", "consumo" e "produção" indicam alterações nos recursos consumidos (u_r) e nos recursos produzidos (C_r), respectivamente. Isto significa que é possível modelar sistemas que tenham valores negativos para certos atributos, que podem gerar sequências de ações não-triviais, ao se considerar os máximos de cada recurso (M_r, L_{rs}).

Para que um estado tenha a capacidade de executar uma determinada tarefa, ele deve possuir, pelo menos, a quantidade de recursos indicada em cada uma de suas relações de dependência: pré-requisitos, custos, consumos e usos. Importante notar que o consumo de pré-requisitos após o início de uma tarefa não a invalida neste problema. Junto com as dependências de recursos também existe a dependência temporal, em que uma ação deve demorar exatamente o tempo prescrito em sua definição. Estas afirmações podem ser expressas nas seguintes restrições:

$$
\begin{aligned}
&U_r(I_i) \geq Pr_t(r) \\
&U_r(I_i) - B_r(I_i) \geq Ct_t(r) + Cm_t(r) + Bo_t(r) \\
&tempo(I_f) - tempo(I_i) = D_t
\end{aligned}
$$

Onde $Pr_t(r), Ct_t(r), Cm_t(r), Bo_t(r)$ são os pré-requisitos, custos, consumos e usos da tarefa $t \in T$ em relação ao recurso $r \in R$, respectivamente. I_i indica o estado do jogo no tempo inicial da tarefa e I_f o estado imediatamente após a conclusão da tarefa. $B_r(I_i)$ indica a quantia do recurso r imobilizada no estado I_i. D_t indica a duração da tarefa t.

2.4. Modelos

A modelagem dentro dos sistemas apresentados se dá apresentando objetivos e restrições em que os recursos atinjam alguma cota. Desta forma, não teremos um estado final alvo único, mas podemos ter uma gama de estados finais decorrentes de diferentes ordens ótimas de ações, caso o modelo O seja multiobjetivo. Os modelos devem obedecer ao seguinte esquema:

$$
\begin{array}{ll}
\text{minimizar} & makespan(I_0, x) \\
\text{maximizar} & F_m(I_0, x), \forall m \in M \\
\text{minimizar} & F_n(I_0, x), \forall n \in N, N \cap M = \emptyset \\
\text{sujeito a} & \\
 & makespan(I_0, x) \leq T_f \\
 & F_k(I_0, x) < l_k, \forall k \in K \\
 & F_q(I_0, x) > g_q, \forall q \in Q \\
|M|, |N|, |K|, |Q| \geq 0 & \\
T_f > 0 &
\end{array}
$$

Onde os elementos de cada conjunto M, N, K, Q indicam quais variáveis referentes a certos recursos serão otimizadas. Elas consistem em $C_r, u_r, U_r (r \in R)$, ou seja, estas variáveis podem ser os totais produzidos, totais usados ou totais utilizáveis de certos recursos r. As funções $F_m(I_0, x)$ indicam o valor final da variável m ao final da execução das tarefas prescritas em x, começando a partir do estado inicial I_0. As constantes l_k e g_q são valores de restrição para as variáveis k e q, respectivamente.

Em cenários multiobjetivos, é por vezes importante utilizar uma janela de oportunidade T_f, que indica o tempo máximo para a conclusão das tarefas. Esta necessidade não se apresenta nos casos mono-objetivo, isto é, quando $M = N = \emptyset$.

Para determinar o $makespan(I_0, x)$ é necessário criar uma simulação de um ambiente que aplique as regras do sistema utilizado, criando os eventos e executando as tarefas e eventos. Esta simulação deve determinar o estado final como um todo, tanto os recursos utilizados e coletados como as tarefas não-concluídas.

3. Conclusão

Com este modelo é esperado que haja uma melhora nos otimizadores para jogos de estratégia em tempo real, além de outros ambientes. Trabalhos futuros mostrarão a complexidade do problema e comparação com problemas já consolidados, como o problema de escalonamento de projetos.

Referências

Blackford, J. M. (2014). Online build-order optimization for real-time strategy agents using multi-objective evolutionary algorithms.

Branquinho, A. A. B. and Lopes, C. R. (2010). Planning for resource production in real-time strategy games based on partial order planning, search and learning. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 4205–4211.

Chan, H., Fern, A., Ray, S., Wilson, N., and Ventura, C. (2007a). Extending online planning for resource production in real-time strategy games. In *Workshop on Planning in Games, ICAPS 2007*.

Chan, H., Fern, A., Ray, S., Wilson, N., and Ventura, C. (2007b). Online planning for resource production in real-time strategy games. In *International Conference on Automated Planning and Scheduling 2007*, pages 65–72.

Estudo Preliminar Sobre a Heurística de Lin-Kernighan para o Caixeiro Viajante Multiobjetivo

Emerson Bezerra de Carvalho
Orientadora: Elizabeth Ferreira Gouvêa Goldbarg

[1]Departamento de Informática e Matemática Aplicada – DIMAp
Centro de Ciências Exatas e da Terra – CCET
Universidade Federal do Rio Grande do Norte – UFRN

carvemerson@gmail.com

Resumo. *A heurística de Lin-Kernighan e conhecida como uma das melhores heurísticas para o Problema do Caixeiro Viajante. Neste trabalho, será apresentado um estudo preliminar comparativo entre duas propostas de implementações, ambas utilizando a Heurística de Lin-Kernighan Multiobjetivo, para realizar buscas locais em um conjunto de Pareto inicial.*

Abstract. *The Lin-Kernighan heuristic is known as one of the best heuristics for the Traveling Salesman Problem. In this work, a comparative study is presented between two preliminary implementations proposals, that use the Multiobjective LinKernighan heuristics to perform local searches in a set of initial Pareto.*

1. Indrodução

Uma das heurísticas mais poderosas para o Problema do Caixeiro Viajante (PCV) é a Heurística de Lin-Kernighan (LK) [Lin and Kernighan 1973]. Diversas versões e adaptações já foram propostas para o LK, tanto para a versão multiobjetivo quanto para a versão mono-objetivo do PCV [Helsgaun 2006] [Lust and Teghem 2010] [Applegate et al. 2003].

O PCV multiobjetivo pode ser definido formalmente como um grafo $G = (V, E, w)$, onde V é o conjunto de vértices, E o conjunto de arestas e w é uma função que atribui a cada aresta ($e_{ij} \in E$) um vetor $(w_{ij}^1, ..., w_{ij}^m)$. Cada elemento w_{ij}^k corresponde a uma determinada métrica, como por exemplo, distância e custo para uma aresta de um problema bi-objetivo [ó2004pareto].

Semelhante ao PCV mono-objetivo, para um conjunto de $n = |V|$ cidades, o PCV multiobjetivo consiste em determinar o ciclo hamiltoniano em G, mas minimizando ou maximizando as m funções objetivo do problema, simultaneamente. Entretanto, na maioria das vezes as funções objetivo são conflitantes entre si, fazendo com que o problema tenha mais de uma solução, ou seja, um conjunto de soluções de Pareto [Knowles et al. 2006]. Este conjunto contém todas as soluções que não são dominadas por qualquer outra solução. O problema de encontrar o conjunto Pareto ótimo é NP-difícil [Paquete and Stützle 2003].

O trabalho está organizado da seguinte maneira: na sessão dois serão mostrados os resumos das técnicas utilizadas; em seguida, na sessão 3 os experimentos e resultados, e, por fim, as considerações finais.

2. Técnicas Utilizadas

Os algoritmos propostos são compostos de duas fases. A primeira fase busca um Pareto inicial, e a segunda efetua um *Pareto Local Search* (PLS) [Paquete et al. 2004].

Na primeira fase os dois algoritmos utilizam a Heurística do Vizinho mais Próximo (HVP) [Goldbarg and Goldbarg 2012] com escalarização. A escalarização pondera os objetivos de forma a transformar o problema multiobjetivo em mono-objetivo [Ehrgott and Gandibleux 2000]. A ideia é aplicar pesos diferentes a cada objetivo, de forma que a soma dos pesos seja igual a 1 (um). Várias escalarizações são feitas, com intuito de gerar o conjunto inicial de soluções, e em seguida aplica-se a HVP, que pode ser resumidamente descrita da seguinte forma: a partir de um vértice aleatório, escolha a aresta de menor custo (de acordo com a escalarização feita) que o liga a um vértice não visitado. Em seguida, mova-se até ele, repetindo o mesmo passo até que não haja mais movimentos. No final, o caminho percorrido é a solução.

Os dois algoritmos propostos se diferenciam apenas em um ponto: o primeiro algoritmo (VP-LKM) utiliza apenas a HVP com escalarização, enquanto que o segundo algoritmo (VPLK-LKM) adiciona mais um passo na primeira fase, onde a heurística LK (versão mono-objetivo) é aplicada a cada solução do conjunto gerado, afim de tentar melhorar ainda mais o conjunto.

Na fase seguinte uma PLS é executada nos Paretos gerados na primeira fase. A PLS utiliza como busca local o algoritmo de Lin-Kernighan Multiobjetivo (LKM), que é uma adaptação do algoritmo original (proposto originalmente para a versão mono-objetivo). No LKM, ao invés de dar como resultado uma única solução, é retornado o Pareto gerado a partir de uma solução s, caso exista uma solução s^* vizinha de s que tenha os objetivos melhores ou iguais aos objetivos de s, ou seja, s^* domina s ($s^* \succ s$, [Knowles et al. 2006]), então o LKM irá retornar o Pareto de s^*, senão ele continua gerando o Pareto de s até que não haja mais melhoras ou movimentos a serem feitos. Os resultados entre esses dois algoritmos serão mostrados na sessão seguinte.

3. Experimentos

Para a comparação dos resultados será utilizado o indicador Hipervolume (I_H). O ponto de referência para o I_H foi definido a partir da análise de todos os conjuntos gerados pelos algoritmos, escolhendo-se então o pior valor dentre ambos os objetivos. A Figura 1, retirada de [Knowles et al. 2006], mostra o I_H de um conjunto A, o ponto de referência (20,20) e o espaço objetivo Z'.

A partir do valor I_H podemos calcular a porcentagem da área que é dominada pelo conjunto A; quanto maior a porcentagem, maior a o espaço dominado por A.

3.1. Ambiente Computacional e Casos de Testes

Cada um dos algoritmos foi executado 50 vezes, em um computador com sistema operacional Linux Ubuntu 14.04 LTS 64bits, 6GB de memória RAM e processador AMD A8-5500B APU Quad Core 3.2GHz. Os casos de testes foram retirados da TSPLIB. As instâncias utilizadas são a KroA100 e a KroB100, representando o primeiro e segundo objetivo, respectivamente.

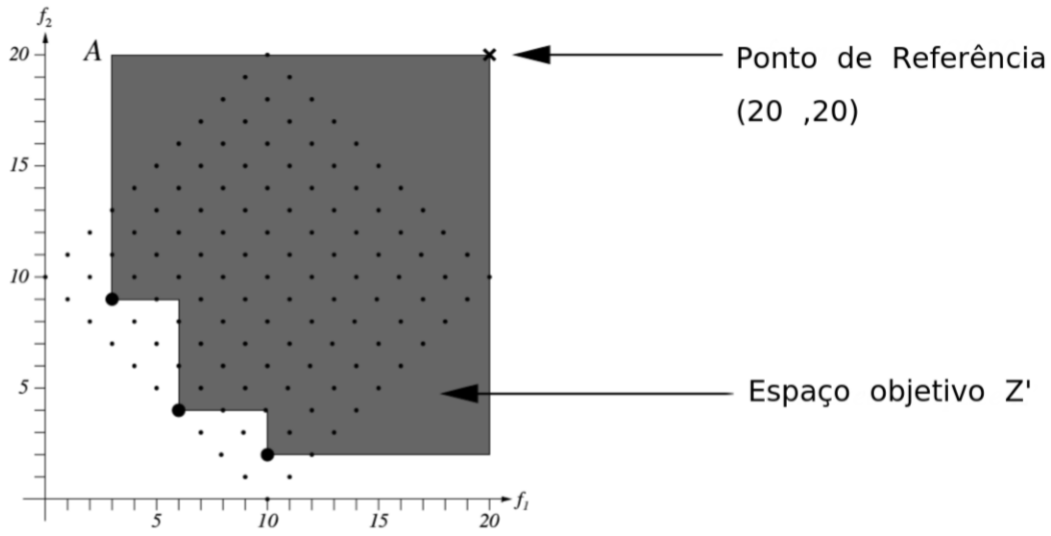


Figura 1. Hipervolume de A, $I_H(A) = 277$.

Tabela 1. Resultados dos testes do hipervolume

Hipervolume	VP-LK	VPLK-LK
Média	67,35%	67,36%
Mediana	67,35%	67,38%
Máximo	67,56%	67,55%
Mínimo	67,11%	67,03%

Tabela 2. Tabela de tempo em segundos

Tempo	VP-LK	VPLK-LK
Média	74,88	73,04
Mediana	72,04	70,19
Máximo	120,78	107,35
Mínimo	52,89	51,90

3.2. Resultados

As Tabelas 1 e 2 mostram a porcentagem e o tempo médio, mediano, mínimo e máximo alcançado por cada um dos algoritmos segundo os resultados do I_H após as 50 execuções. Na Tabela 1, vemos que as porcentagens são praticamente iguais; isso mostra que os algoritmos geram resultados bastante similares segundo o I_H. A Tabela 2 mostra o tempo de execução em segundos dos algoritmos, em média os resultados são bem parecidos, porém, mesmo possuindo um passo a mais na primeira fase de execução, o VPLK-LK conseguiu um tempo médio menor que o VP-LK, mesmo que pouco. A Figura 2 tem um Pareto gerado por uma execução de ambos algoritmos.

4. Considerações Finais

Este trabalho mostrou um estudo preliminar comparando duas versões simples de algoritmos que utilizam a heurística LK para o PCV multiobjetivo. Os resultados mos-

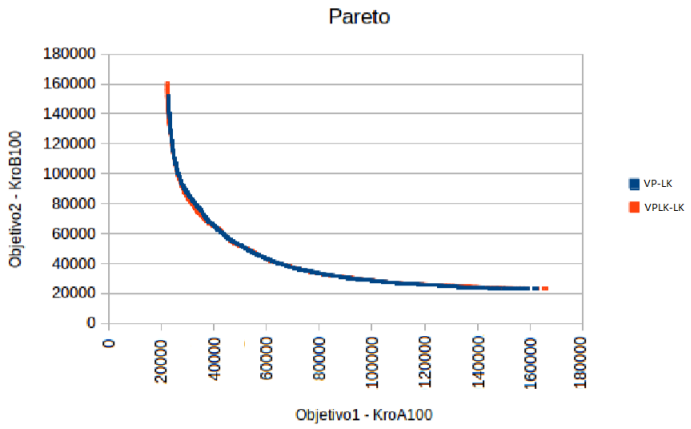


Figura 2. Pareto encontrado pelos algoritmos

tram que as duas versões são praticamente iguais com relação ao I_H, mostrando uma pequena diferença no tempo médio de execução. Para trabalhos futuros, serão propostas novas versões de algoritmos que utilizam o LK, afim de melhorar ainda mais o resultados, utilizar outros indicadores e comparar os resultados obtidos com os resultados da literatura.

Referências

Applegate, D., Cook, W., and Rohe, A. (2003). Chained lin-kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92.

Ehrgott, M. and Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460.

Goldbarg, M. and Goldbarg, E. (2012). *Grafos: Conceitos, algoritmos e aplicações*. Elsevier Brasil.

Helsgaun, K. (2006). *An effective implementation of K-opt moves for the Lin-Kernighan TSP heuristic*. PhD thesis, Roskilde University. Department of Computer Science.

Knowles, J., Thiele, L., and Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland. revised version.

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516.

Lust, T. and Teghem, J. (2010). The multiobjective traveling salesman problem: a survey and a new approach. In *Advances in multi-objective nature inspired computing*, pages 119–141. Springer.

Paquete, L., Chiarandini, M., and Stützle, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In *Metaheuristics for Multiobjective Optimisation*, pages 177–199. Springer.

Paquete, L. and Stützle, T. (2003). A two-phase local search for the biobjective traveling salesman problem. In *Evolutionary Multi-Criterion Optimization*, pages 479–493. Springer.

Algoritmos Exatos para o Problema Biobjetivo da Árvore Geradora Quadrática em Adjacência de Arestas

Lucas D. M. S. Pinheiro[1], Elizabeth F. G. Goldbarg[1], Silvia M. D. M. Maia[1], Marco C. Goldbarg[1]

[1]Departamento de Informática
e Matemática Aplicada – Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal 1524 – 59.078-900 – Natal – RN – Brazil

lucasd_monteiro@outlook.com, {beth,silvia,gold}@dimap.ufrn.br

Abstract. *This paper describes the Biobjective Adjacent Only Quadratic Spanning Tree problem (Bi-AQST), which is a variation of the classic problem of Minimum Spanning Tree (MST). In addition, it presents the exact algorithms developed to solve it, and compare them with another exact approaches present in literature.*

Resumo. *Este artigo descreve o problema Biobjetivo da Árvore Geradora Quadrática em Adjacência de Arestas (AGQA-bi), que é uma variação do problema clássico da Árvore Geradora Mínima (AGM). Além disso, apresenta os algoritmos exatos implementados para resolvê-lo, além de comparar com outras abordagens exatas existentes na literatura.*

1. Introdução

O problema da Árvore Geradora Mínima Quadrática (AGMQ) é uma versão do problema clássico da Árvore Geradora Mínima (AGM) no qual, além dos custos lineares das arestas são considerados custos quadráticos associados a cada par de arestas [Assad and Xu 1992]. No caso das interações ocorrerem somente entre arestas adjacentes, o problema é denominado Árvore Geradora Mínima Quadrática em Adjacência de Arestas (AGMQA). Tanto a AGMQ quanto a AGMQA são problemas NP-difíceis que procuram modelar problemas reais envolvendo projeto de redes de transporte e distribuição. Considere um grafo simples, não-direcionado e conexo $G = (V, E)$, onde $V = \{v_1, ..., v_n\}$ é um conjunto de n vértices e $E = \{e_1, ..., e_m\}$ um conjunto de m arestas, um valor associado a cada aresta, que representa seu custo linear $W = \{w_1, ..., w_m\}$, e um vetor de booleanos X onde $x_i, i = 1, ..., m$ possui valor 1 quando a i-ésima aresta faz parte da árvore e 0 caso contrário. Considere um subgrafo $T = (V_T, E_T), V_T = V, E_T \subseteq E$, tal que exista exatamente um único caminho entre cada par de vértices em T. A AGMQ considera custos de interação entre pares de arestas, também chamados intercustos, dados em uma matriz C_{ij} que representa o custo associado a cada par de arestas $(e_i, e_j), i, j = 1, ..., m$. O objetivo é encontrar uma árvore que minimize a equação 1.

$$Z(X) = \sum_{i=1}^{m} \sum_{j=1}^{m} C_{ij} x_i x_j + \sum_{i=1}^{m} w_i x_i | X \in T \tag{1}$$

Ao considerar apenas os intercustos quando as arestas são adjacentes, têm-se a AGMQA. Nos dois casos, os objetivos lineares e quadráticos são somados, e não se leva

em conta que muitas vezes a minimização de um deles pode ser conflitante com a do outro. É nesse quesito que a otimização multiobjetivo se torna atraente numa modelagem mais realista. No caso multiobjetivo (AGQA-bi) [Maia 2013], consideram-se os custos lineares e quadráticos como objetivos separados e, considerando A_i como o conjunto de arestas adjacentes à aresta de índice i, busca-se uma árvore T que minimize a equação 2.

$$Min\ Z(X) = (Z_1(X), Z_2(X))|X \in T \tag{2}$$

$$Z_1(X) = \sum_{i=1}^{m} w_i x_i \tag{3}$$

$$Z_2(X) = \sum_{i=1}^{m} \sum_{j \in A_i} C_{ij} x_i x_j \tag{4}$$

Existem dois algoritmos exatos relatados para a AGQA-bi por Maia (2013): *backtracking* e *branch-and-bound*. O que os diferem dos algoritmos apresentados neste artigo é o fato de terem sido implementados de maneira iterativa e o *branch-and-bound* utilizar um cálculo de estimativa quadrática diferente. As seções seguintes apresentam os algoritmos implementados, seguidos dos experimentos computacionais e da conclusão.

2. Algoritmos Exatos

Tanto no *backtracking* quanto no *branch-and-bound* as arestas do grafo são colocadas em uma lista, ordenadas pelo menor custo linear. A ideia do *backtracking* é explorar as maneiras de formar uma árvore através de uma combinação das arestas, sem necessariamente gerar todas as árvores geradoras do grafo. O algoritmo implementado mantém uma lista de arestas já selecionadas sol para fazer parte da árvore e um índice que indica qual a última aresta analisada da lista de arestas original do grafo, chamada $edges$. A medida que o índice vai avançando, verifica-se se a inserção da aresta não irá gerar um ciclo e, caso não gere, sua inserção é feita, continuando até que uma árvore seja formada. Em cada passo considera-se também a não inclusão da aresta na solução. Caso, em algum momento, a solução sendo montada já seja dominada por alguma presente no conjunto $fronteira$, que guarda as soluções não-dominadas até o momento, abandona-se esse ramo, e ocorre o *backtracking*. O algoritmo *branch-and-bound* desenvolvido é semelhante ao *backtracking*, apenas com a adição de duas funções que calculam uma estimativa para o objetivo linear e para o objetivo quadrático, a fim de saber se a melhor solução que poderia ser formada explorando aquele ramo da árvore de recursão será dominada por alguma outra no conjunto de fronteira obtido até o momento, com esperança de realizar podas antecipadas na árvore, diminuindo a quantidade de processamento utilizada. Na estimativa linear, calcula-se o custo da árvore geradora mínima considerando apenas os custos lineares das arestas, contando com o fato de que as arestas presentes em sol são obrigatórias. Na estimativa quadrática, calcula-se o custo de inserir as melhores arestas que faltam para completar a árvore, somente levando em conta as arestas que já estavam em sol e os intercustos das que estão fora com estas, sem a realização de uniões, como no caso da estimativa linear. Caso alguma aresta que esteja fora não seja adjacente a nenhuma que já faça parte da solução, seu valor estimado de contribuição é dado pelo menor intercusto entre ela e alguma das adjacentes. Uma segunda versão do

algoritmo *branch-and-bound* foi implementada, mas a única diferença se dá pelo fato de que a nova utiliza-se do mesmo procedimento de geração de soluções iniciais adotadas por Maia (2013), utilizando a heurística H2 de Assad e Xu (1992), a fim de reduzir o número de estados visitados pelo algoritmo, já que haverão mais soluções no conjunto de não-dominadas inicialmente.

3. Experimentos Computacionais

As instâncias utilizadas para a realização dos experimentos foram as mesmas utilizadas por Maia(2013). Consistem de grafos com 10, 15 e 20 vértices e com densidade 33%, 67% e 100%. Para cada número de vértices, 12 instâncias foram geradas. O intervalo de escolha dos custos lineares é ([0, 10] ou [0, 100]) e o intervalo de escolha dos custos quadráticos ([0,10] ou [0, 100]). Os algoritmos foram implementados em C++, usando o compilador g++ 4.8.1 e a flag de otimização -O3, sistema operacional Ubuntu 13.10 de 64 bits. Todos os experimentos foram realizados em máquinas HP Z400, com processador Intel Xeon QuadCore W3520 de 2.8 GHz e 8Gb de RAM. Os algoritmos foram comparados com as versões apresentadas por Maia (2013), que foram executadas numa máquina com as mesmas configurações acima, exceto pelo sistema operacional utilizado, que foi o Scientific Linux 5.5 de 64 bits. Por fim, foi estabelecido um limite de 3600 segundos para a execução dos algoritmos, e apenas as instâncias nas quais esse tempo limite foi cumprido são exibidas.

Instância	T(s)	visitados	AGMs	$\|\|Sol\|\|$	T(s) Back2	visitados Back2	AGMs Back2	$\|\|Sol\|\|Back2$
10.1	0,00	1164	80	5	0,00	7097	80	5
10.2	0,00	1157	80	5	0,00	7098	80	5
10.3	0,00	1290	85	7	0,00	7105	85	7
10.4	0,00	1290	87	7	0,00	7106	87	7
10.5	0,10	199919	9735	7	0,28	1220213	6816	7
10.6	0,09	175716	7015	11	0,25	1136869	5554	11
10.7	0,09	191475	7929	9	0,27	1207395	7585	9
10.8	0,08	173815	6706	14	0,26	1124700	6478	14
10.9	1,14	1592558	124892	13	3,03	14188485	96837	13
10.10	0,95	1203720	88251	21	2,58	11708418	77708	21
10.11	1,17	1485804	144602	17	3,19	13803348	109104	17
10.12	1,01	1175870	113415	30	2,66	11374045	87842	30
15.1	8,35	27328534	93805	16	71,29	232409214	84022	16
15.2	8,38	25017517	81163	26	70,10	221495709	79799	26
15.3	8,24	25511113	74976	21	68,69	217248471	73617	21
15.4	8,44	23668626	68969	40	70,34	205119467	68563	40

Tabela 1. *Backtracking* na AGQA-bi

São exibidos o tempo computacional em segundos, o número de estados visitados pelos algoritmos e a quantidade de árvores geradoras completas formadas, além do tamanho do conjunto de soluções não-dominadas obtido. Apenas os casos com 10 vértices e com 15 vértices (até o quarto caso dessa classe) foram possíveis de serem examinados, pois as outras não ficaram no limite de tempo estabelecido. A tabela 1 mostra a comparação entre o *backtracking* aqui implementado e o de Maia (2013), representado pelo nome Back2. Observa-se que o *backtracking* proposto neste trabalho foi mais rápido em relação ao tempo computacional e teve um número menor de nós visitados e de AGMs geradas, mas manteve o número de soluções não dominadas igual ao algoritmo de Maia (2013). A tabela 2 compara a segunda versão do algoritmo *branch-and-bound* com a versão de Maia (2013), representado pelo nome BBS. Os dados da tabela mostram que a

abordagem de Maia (2013) foi mais eficiente em termos de tempo computacional, mesmo tendo visitado mais nós e gerado mais árvores geradoras do que o algoritmo aqui proposto. A explicação pode estar no fato de que os propostos foram implementados de maneira recursiva, enquanto os de Maia (2013) foram feito de forma iterativa.

Instância	T(s)	visitados	AGMs	$\|\|Sol\|\|$	T(s) BBS	visitados BBS	AGMs BBS	$\|\|Sol\|\|BBS$
10.1	0,00	31	6	5	0,00	369	14	5
10.2	0,00	30	6	5	0,00	316	14	5
10.3	0,00	33	7	7	0,00	381	17	7
10.4	0,00	31	7	7	0,00	333	17	7
10.5	0,05	1972	78	7	0,04	18502	140	7
10.6	0,04	1505	77	11	0,03	19302	207	11
10.7	0,04	2008	53	9	0,03	16757	137	9
10.8	0,04	1700	90	14	0,03	20098	234	14
10.9	0,39	7661	1176	13	0,37	190445	1421	13
10.10	0,37	7554	1260	21	0,42	208648	2218	21
10.11	0,40	8008	1345	17	0,39	199599	1615	17
10.12	0,42	8744	1395	30	0,44	221473	2195	30
15.1	0,76	35515	290	16	0,39	248587	277	16
15.2	0,70	32606	262	26	0,50	325329	846	26
15.3	0,72	34350	306	21	0,41	264167	323	21
15.4	0,67	31364	380	40	0,54	349268	1077	40
15.5	196,05	2445723	1541	14	49,41	18811547	2123	14
15.6	129,86	1483179	1916	19	42,08	15379606	5134	19
15.7	158,18	1911810	1666	25	51,63	19803561	4145	25
15.8	100,86	1079234	2446	35	42,67	15567627	7485	35
15.9	3555,38	19867660	11382	25	2248,20	635894199	16984	25
15.10	3560,84	19022836	15854	33	2541,25	693058692	39226	33
15.11	3160,01	15786245	16609	43	1910,92	535708586	28292	43
15.12	3600,00	17541963	23761	58	2333,70	632304396	54178	58
20.1	1266,09	17123058	5386	31	804,86	375600900	16865	31
20.2	1033,50	13938164	5761	39	999,51	467708070	15339	39
20.3	1413,24	19136890	2895	60	1085,81	504485162	27701	60
20.4	1272,80	16917005	4793	86	1424,98	656363595	31717	86
20.5					3419,92	778515188	42455	26

Tabela 2. *Branch-and-bound* versão 2 e de Maia (2013) para AGQA-bi

4. Conclusão

Este artigo conduziu uma investigação de algoritmos exatos para o problema biobjetivo da Árvore Geradora Quadrática em Adjacência de Arestas. Foram desenvolvidos algoritmos *backtracking* e *branch-and-bound* recursivos, e os mesmos comparados com as abordagens exatas presentes em [Maia 2013]. O algoritmo *backtracking* proposto foi mais eficiente do que o de Maia (2013), enquanto o *branch-and-bound*, apesar de visitar menos nós e gerar menos árvores geradoras, levou mais tempo computacional do que o de Maia (2013). Trabalhos futuros podem envolver o desenvolvimento de novas maneiras de calcular as estimativas, a fim de melhorar a eficiência dos algoritmos.

Referências

Assad, A. and Xu, W. (1992). The quadratic minimum spanning tree problem. *Naval Research Logistics*, 39:399–417.

Maia, S. M. D. M. (2013). *O Problema Biobjetivo da Árvore Geradora Quadrática em Adjacência de Arestas Biobjetivo*. PhD thesis, Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte.

A probabilistic analysis of the biometrics menagerie existence in fingerprint data

Rayron Medeiros and Márjory Da Costa-Abreu

[1]DIMAp - UFRN
Campus Universitário Lagoa Nova - RN - Brasil

marjory@dimap.ufrn.br

Abstract. *Even though the use of biometrics has been restricted to high-security environments and criminal identification applications, with its popularisation, it has been noted that users within the system may have different degrees of accuracy. Some people may have trouble authenticating, while others may be particularly vulnerable to imitation. Recent studies have investigated and identified these types of users, giving them the names of animals: Sheep, Goats, Lambs, Wolves, Doves, Chameleons, Worms and Phantoms. The aim of this study is to evaluate the existence of these animals in a database of fingerprints and propose a new way of their investigation, based on the performance of verification between subjects samples.*

1. The biometrics menagerie

The recognition performance of a biometric-based system can vary significantly from one user to another. As a result, we have some users that are falsely rejected by the system, while others are easily impersonated by impostors. As already mentioned, several groups of problematic users were identified, and each one was given the name of an animal which, similarly, reflects its behaviour. The concept of biometrics menagerie was formalised and the first animals can be described as follows:

a **Sheep** represent the majority of the population and are usually easy to identify;
b **Goats** are users usually difficult to identify. These individuals tend to have a low match score when compared with themselves. They represent a disproportionate increase in the FRR;
c **Lambs** are individuals easy to impersonate. Other users tend to have a relatively high match score when compared with lamb users. They represent a disproportionate increase in the FAR;
d **Wolves** are good at impersonating users. When compared against other users, they tend to have a high match score. Like lambs, they represent a disproportionate increase in the FAR.

Only goats, lambs and wolves contribute to a negative impact on the system error rate, therefore users in these categories are called weak users. Different investigations [Doddington et al. 1998, Hicklin et al. 2005, Poh and Kittler 2008, Poh and Kittler 2009] have confirmed that the weak users constitute only a small fraction of the population of a biometric system, however, their contribution in the error rate can be disproportionately high. In a later recent investigation by Yager and Dunstone [Yager and Dunstone 2007], new animals were identified.

It is important to note that the animals do not necessarily represent a distinct and mutually exclusive subgroup of users [Doddington et al. 1998, Wayman 2004]. In fact, it is possible that they do not even exist in a real system. Animals can be better understood as a tendency of behaviour and, thus, an individual may be more susceptible to attack than another. Doddington conducted their study based on data from speech recognition, but the concept of animals can be applied to several areas of biometric identification. Subsequently, several other studies have shown the existence of animals in other biometrics. Wayman [Wayman 2004] demonstrated the existence of lambs and wolves in fingerprint data with a high degree of significance; Wittman [Wittman et al. 2006] examined the existence of animals in face recognition; Poh and Kittler [Poh and Kittler 2008] showed, individually, the phenomenon in different biometrics; and others, such as [Yager and Dunstone 2007].

This paper presents an investigation of the presence of animals in a fingerprint identification biometric system and evaluate the causes of their appearances. The main aim is to propose a new method of investigation and analysis of biometric systems, which differs from traditional methods which focus on statistical evaluation of global errors, such as ROC curves and EER. These statistics are useful for evaluating a biometric system as a whole, but they might ignore idiosyncrasies associated with users characteristics.

1.1. Results for the analysis of the animals existance based on samples

In order to perform our investigation, we have used the CASIA-FingerprintV5 [cas 2013] database, that contains 20.000 fingerprint images of 500 individuals, where each individual contributed with 40 impressions (5 for each finger, except the little finger). The individuals were instructed to rotate the fingers with various levels of pressure to generate significant intra-class variations.

Table 1 shows the results obtained from the experiment of samples analysis using the threshold of 20, while Table 2 and Table 3 show the results for $< 200b ><< 200b > 30$ and 40, respectively. The rows of each table are divided by the number of training samples which exceed the threshold when compared with the test samples of the same user. Thus, the first column of the first row represents the quantity of test sample that, when compared with training samples of the same user, did not exceed the threshold in any comparison. The first column of the second row represents the quantity of test samples which exceeded the threshold in the matching. The following columns show the number of attacks per number of test samples that exceeded its respective threshold.

Tables 1, 2 and 3 have regions representing the likelyhood of the presence of animals in the system. A significant number of samples with 0 match can be indicative of the presence of goats in the system or, at least, demonstrates that a considerable portion of the enrolled samples do have low quality. Similarly, a significant number of samples with 3 matches is an indicative of the presence of sheep in the system.

Tables 1, 2 and 3 show an increase the verification threshold decrease the amount of attacks and hence the false acceptance rate FAR. But on the other hand, also increased the number of verifications that not reached the threshold value, which leads to an increase in false rejection rate FRR.

In Figure 1, the doves are represented by the test users that had 3 matches and no attack; chameleons are the users that had 3 matches and at least one attack; the phantoms

Table 1. CASIAV5 with threshold 20.

	mtc	1atk	2atk	3atk	4atk	+5atk	total
0mtc	103	10	2	1	5	8	+77
1mtc	100	8	9	7	2	32	+215
2mtc	189	21	16	11	10	68	+466
3mtc	608	56	34	38	22	356	+2.096
tot	1000	95	122	171	156	+2.310	

Table 2. CASIAV5 with threshold 30.

	mtc	1atk	2atk	3atk	4atk	+5atk	total
0mtc	178	5	1	x	x	x	7
1mtc	152	12	2	x	1	x	20
2mtc	220	23	6	4	x	2	+57
3mtc	450	52	15	15	9	3	+178
tot	1000	92	48	57	40	+25	

Table 3. CASIAV5 with threshold 40.

	mtc	1atk	2atk	3atk	4atk	+5atk	total
0mtc	272	x	x	x	x	x	0
1mtc	183	1	x	x	x	x	1
2mtc	214	x	x	x	x	x	0
3mtc	331	2	x	x	x	x	2
tot	1000	3	0	0	0	0	

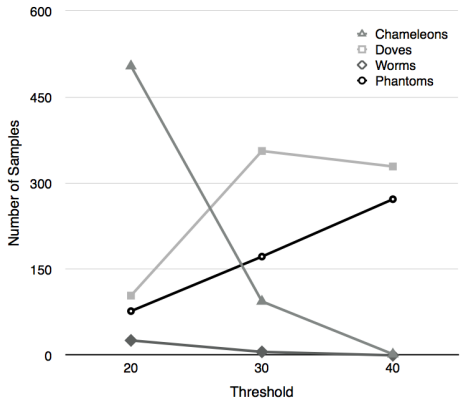


Figure 1. Animals indicative locations.

are the users that had neither matches nor attacks; and worms are the users that had not been matched and attacked.

In the case that we used the verification value of 20, more than half of the test users had 3 matches (doves and chameleons), and only 103 users have no matches (phantoms and worms). However, those that had more than 3 matches, 504 were also attacked. This can represent a disproportionately high false acceptance rate. When we increased the threshold for 40, the amount of animals which represents a false acceptance decreases (chameleons and worms), while the amount of phantoms, which represents a false rejection, increases. This behaviour proves that we are not able to decrease the rate of false

acceptance and false rejection rate simultaneously.

2. Final remarks

This paper has introduced an investigation of the presence of the biometric menagerie in a fingerprint-based biometric authentication system. It was shown that, despite what is claimed in the literature, there is no statistical significance evidence for the existence of goats, lambs and wolves. This is mainly due to the fact that, although the users tend to have individual match score distributions of genuine and impostor match score, some factors can reduce this effect, such as a good algorithm for the minutiae extraction.

On the other hand, we have found evidence of the existence of the other four animals (worms, doves, chameleons, and phantoms). These animals are defined in terms of the relationship between genuine and impostor match scores, and it was shown that they are present or absent significantly in real biometric fingerprints. This presence or absence of these animals do reflect the properties of the matching algorithm as well as the population of users.

We can, therefore, conclude that the reasons for the existence of a particular group of animals are varied and complex. They depend on a number of factors, including the processes used in the registration, feature extraction and matching algorithm, the quality of the captured fingerprint by the sensor and the intrinsic properties of the user population.

References

(2013). Casia-fingerprintv5.

Doddington, G., Liggett, W., Martin, A., Przybocki, M., and Reynolds, D. (1998). Sheep, goats, lambs and wolves a statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation. In *ICSLP'98*.

Hicklin, A., Watson, C., and Ulery, B. (2005). The myth of goats: How many people have fingerprints that are hard to match? Technical report, NIST.

Poh, N. and Kittler, J. (2008). A methodology for separating sheep from goats for controlled enrollment and multimodal fusion. In *Biometrics Symposium, 2008. BSYM '08*.

Poh, N. and Kittler, J. (2009). A biometric menagerie index for characterising template/model-specific variation. In *Advances in Biometrics*. Springer Berlin Heidelberg.

Wayman, J. L. (2004). Multifinger penetration rate and roc variability for automatic fingerprint identification systems. In *Automatic Fingerprint Recognition Systems*. Springer New York.

Wittman, M., Davis, P., and Flynn, P. (2006). Empirical studies of the existence of the biometric menagerie in the frgc 2.0 color image corpus. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*.

Yager, N. and Dunstone, T. (2007). Worms, chameleons, phantoms and doves: New additions to the biometric menagerie. In *Automatic Identification Advanced Technologies, 2007 IEEE Workshop on*.

Acquisition and analysis of a new hand-based biometric database: keystroke dynamics

Valmiro Ribeiro and Márjory Da Costa-Abreu

[1]DIMAp - UFRN
Campus Universitário Lagoa Nova - Natal - RN - Brasil

marjory@dimap.ufrn.br

Abstract. *The technologies of biometrics provide a variety of powerful tools to determine or confirm individual identity while, more recently, there has been considerable interest in using soft biometrics (personal information which is characteristic of, but not unique to, individuals) in the identification task. Although the area has seen an increase advance in the last decade, there is still the need to understand the existing demographics differences. Therefore, this paper presents a protocol for the collection of the first keystroke dynamics that will be part of a multimodal hand-based collection happening in Brasil.*

1. Introduction

The concern regarding our personal information which is now stored in computational systems, such as shopping websites, online bank services, etc has increased recently. The traditional data stored are usernames and passwords, but, this information can be easily lost, forgotten or stolen. In order to avoid this, the research field of biometrics has been studying and making progress to improve digital security.

Biometrics is the terminology used to refer to the field of computer science that uses human characteristics as a form of identification or access control, and usually is used in authentication systems. Biometric characteristics can be classified as either physiological, which can include retinal, iris patterns, face, palm topology, fingerprints and thermal images, or behavioural, which can include voice, handwritten signatures, and keystroke dynamics [Bergadano et al. 2002].

Since there are different types of biometrics, the selection of a particular biometric modality for use in a specific application involves analysing several factors. In a ideal environment, each modality should have some of the following characteristics: Universality, Distinctiveness, Permanence, and Collectability [Jain et al. 2004].

Among the listed characteristics, the collectability can have an enormous impact on the general performance of the system. If the data collected does not have good quality, a lot must be spent on pre-processing in order to have a reliable system. There are very little databases which were collected in real world situations and often they only have physiological characteristics [Ortega-Garcia et al. 2006].

In order to understand and developed more suitable authentication systems, it is crucial that we develop well accepted and reliable collection protocols. Differently from a physiological modality, when you are using a behavioural one, you have many more valiables to adjust and, the best way to do this is by collecting a new database in real world scenarios, and preferably with universal characteristics.

This paper presents the prototype of a new system for the collection of a new keystroke biometric database, which has a high acceptability and it is very cheap to collect and analyse and the main contribution for it is to investigate the best way (if it is by analysing digraphs or words, etc, for instance).

2. Keystroke dynamics

Keystroke dynamics (also known as keyboard dynamics) is the study of the unique timing patterns embedded in an individual's typing and most often developed in a way characteristic of that individual, hence the use of keyboard dynamics as a biometrics-based identification modality. Processing of such data typically includes extracting keystroke timing features such as the duration of a key press and the time elapsed between successive key presses [Epp et al. 2011].

Keystroke-based user identification compares current user activity against stored samples of similar activity usually captured during enrolment (known also as a user *profile*) ([Monrose and Rubin 1997] and [Bergadano et al. 2002]). Most applications aim at detecting significant differences in computer use in order to reduce inappropriate user authorisations to access and change valuable data, depending on the degree of certainty of negative identification ([Montalv and Freire 2006] and [Revett et al. 2007]).

Keystroke-related data can be collected without the aid of special tools or additional hardware but, nevertheless, user authentication through keystroke characteristics remains a difficult task because, as with all behavioural biometrics, typing dynamics are prone to a higher degree of variability than physiological biometrics, even without considering the psychological and physiological state of the individual under observation. Thus, the variability between two immediately consecutive samples can occur even if the subject providing the samples strives to maintain a uniform way of typing ([Chudá and Ďurfina 2009], [Kang and Cho 2009], [Hwang et al. 2009] and [Giot et al. 2009]).

Physiological characteristics tend to be more robust in this respect, although some modalities such as voice patterns and handwriting generally offer a manageable level of uniformity. On the other hand, in the act of typing on a keyboard, it is very challenging to maintain significant control over, for example, the number of milliseconds for which a key is depressed.

The main features typically used for user verification/identification based on their keystroke dynamics are:

- Latency between consecutive keystroke and
- Duration of the keystroke (hold time)

Most keystroke-based authentication systems use fixed-text models, in the sense that they use exactly the same static piece of text for authentication as was used during model construction. There have been fewer approaches that use models based on free text (text that is not prescribed to the user), and unsurprisingly these do not perform as well as fixed-text models. The length of the required training text varies between different studies; some require a few words or full pages of text, which can create better-performing models.

3. Proposed data collection protocol

Our data collection protocol is composed by a set of tasks chosen carefully and a questionnaire to be applied to every user. There are tasks in the chosen set that do not have anything to do with biometrics, because this protocol will be part of multimodal collectin a multi-modal of hand-biometrics (biometrics that can be collected from someone's hand).

Firstly, an user will be asked to answer some questions that we can not respond with our software, such as gender, age band, hand size and level of familiarity with an keyboard. This will make possible to categorise users and analyse specific characteristics of each group.

Secondly, the user will start with our main task related with keystroke dynamics. In order to make possible to other work compare their results with ours, without losing our ability to identify if someone is typing in Brazilian Portuguese, it was necessary to chose carefully every word from our fixed text. Then, we have chose 68 (sixty-eight) from the most frequently used words from Brazilian Portuguese. However, some of these words, called cognates, are written in the same way for both Brazilian Portuguese and English. Thus, we have reduced our set to only 15 (fifteen) words, to do not make a long collecting session. With our currently set of words, we may be able to compare features of a keystroke's database collected using this protocol and another database, collected using a different protocol with some of the same words. This set can be seen in the table below.

america	coisa	normal
internet	fazer	homem
primeiro	carro	porque
case	video	jesus
mouse	felicidade	pequeno

The main requirement for the selection of words for this database was that words must have very common digraphs from the Brazilian Portuguese, and in English, in order to this database be comparable. Once a first selection of words in complete, we can say that all the words from this protocol were carefully analysed to provide good information significance for both languages.

It is important to remember that we do not use capital letters, special characters or accent within this, because this kind of feature may interfere with our analysis. Also, we have used an universal QWERTY keyboard. With our collection process being applied to Brazilians, this kind of keyboard should probably confuse the native users if they were to look for accents or other special characters.

After the collection of the described words, there will be a task to collect keystroke information about a set of numbers. With numbers being universal, the only modifier to our task is if some individual is typing using the letters pad (left pad of the keyboard, containing the letters and numbers) or the number pad (right pad, containing numbers and arithmetical operators). This modifier will make possible to us to determine the differences between the two forms, if there is one. The set of numbers collected are "0168245739".

4. Final Remarks

This paper has presented a protocol for the collection of a new keystroke dynamics database. Current work is being developed in order to improve the protocol and to have a continuous multimodal data collection that will englobe four hand-based modalities (fingerprint, signature, keystroke dynamics and touch-screen dynamics).

References

Bergadano, F., Gunetti, D., and Picardi, C. (2002). User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397.

Chudá, D. and Ďurfina, M. (2009). Multifactor authentication based on keystroke dynamics. In *The International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, CompSysTech 2009, pages 1–6, New York, NY, USA. ACM.

Epp, C., Lippold, M., and Mandryk, R. (2011). Identifying emotional states using keystroke dynamics. In *The 2011 annual conference on Human factors in computing systems*, CHI 2011, pages 715–724, New York, NY, USA. ACM.

Giot, R., EI-Abed, M., and Rosenberger, C. (2009). Keystroke dynamics with low constraints svm based passphrase enrollment. In *The 3rd IEEE international conference on Biometrics: Theory, applications and systems*, BTAS 2009, pages 425–430, Piscataway, NJ, USA. IEEE Press.

Hwang, S.-s., Lee, H.-j., and Cho, S. (2009). Improving authentication accuracy using artificial rhythms and cues for keystroke dynamics-based authentication. *Expert Systems with Applications: An International Journal*, 36(7):10649–10656.

Jain, A., Dass, S., and Nandakumar, K. (2004). Can soft biometric traits assist user recognition? In *The 1st international conference on Biometric Authentication*, Lecture Notes in Computer Science, pages 561–572, Hong Kong.

Kang, P. and Cho, S. (2009). A hybrid novelty score and its use in keystroke dynamics-based user authentication. *Pattern Recognition*, 42(11):3115–3127.

Monrose, F. and Rubin, A. (1997). Authentication via keystroke dynamics. In *The 4th ACM conference on Computer and communications security*, CCS 1997, pages 48–56, New York, NY, USA. ACM.

Montalv ao Filho, J. and Freire, E. (2006). On the equalization of keystroke timing histograms. *Pattern Recognition Letters*, 27(13):1440–1446.

Ortega-Garcia, J., Alonso-Fernandez, F., Fierrez-Aguilar, J., Garcia-Mateo, C., Salicetti, S., Allano, L., Ly-Van, B., and Dorizzi, B. (2006). Software tool and acquisition equipment recommendations for the three scenarios considered. Technical Report Report No.: D6.2.1. Contract No.: IST-2002-507634, Universidad Politechnica de Madrid.

Revett, K., Gorunescu, F., Gorunescu, M., Ene, M., Magalhaes, S. d., and Santos, H. D. (2007). A machine learning approach to keystroke dynamics based user authentication. *International Journal of Electronic Security and Digital Forensics*, 1(1):55–70.

Métodos de Agrupamentos Para Dados Intervalares Híbridos

Liliane R. Silva[1], Ronildo Moura[1], Anne Canuto[1], Benjamin Bedregal[1], Regivan Santiago[1]

[1]Departamento de Informática e Matemática Aplicada,
Universidade Federal do Rio Grande do Norte(UFRN) Natal, RN, 59.072-970, Brazil.
E-mail: { liliane ronildo}@ppgsc.ufrn.br, {anne bedregal regivan}@dimap.ufrn.br

Resumo. *Neste artigo, busca-se verificar métodos para trabalhar com dados híbridos do tipo que contenham dados reais e intervalares. Para este propósito, são propostos algoritmos de agrupamentos crisp, IbKM e IbKMH que são adaptações do algoritmo K-means.*

1. Introdução

No mundo real podemos observar que existem uma grande variedade de tipos de dados categóricos, numéricos, fuzzy, intervalares, etc. além de propostas de medidas de similaridade para eles. No entanto, na maioria das vezes, os dados obtidos envolvem mais de um tipo de dados. O que normalmente é feito nesses casos é transformar os dados que possuem uma determinada natureza para dados numéricos, afim de obter uma base de dados homogênea.

Esse tipo de transformação pode acarretar em perda de informação, (por exemplo, imprecisão), ou no aumento do custo computacional. Assim, é necessário obter uma maneira de medir similaridade sem transformar os dados. O que é geralmente encontrado na literatura de algoritmos para agrupamento de dados híbridos são combinações de algoritmos que atuam em cada campo do dado.

Em 2012, a pesquisa realizada em [Pereira 2012] mostrou que existem poucos métodos que são capazes de extrair conhecimento a partir de dados híbridos. O autor em questão lidou com dados híbridos que são compostos por dados convencionais (numéricos e textuais) e dados geográficos (pontos, linhas e polígonos).

Essa abordagem é alterada neste trabalho. Em vez de aplicar apenas uma coersão nos dados para um único tipo e aplicar uma medida de similaridade, esse trabalho propõe também que se aplique cada medida de similaridade ligada aos dados originais a cada campo (o resultado, até aqui, serão números reais ou intervalos) e em seguida transforma-se esses valores para intervalos a fim de aplicar i-métricas.

Todo o embasamento teórico sobre i-métricas utilizados neste artigo, pode ser facilmente encontrado em [de Santana 2012], [da Silva 2015] e [Silva et al. 2015]. Neste trabalho iremos utilizar a distância d_{km} que é a uma das i-métricas propostas em [de Santana 2012] para aplicações em agrupamentos de dados intervalares.

A representação utilizada para que se possa calcular a distância intervalar entre dois intervalos X e Y, $d_{km}(X, Y)$, e dada abaixo:

Teorema 1.1 *Dados $X, Y \in \mathbb{I}(\mathbb{R})$, temos:*

$$d_{km}(X,Y) = \begin{cases} [0,0] & \text{, se } X = Y \\ [d(\overline{x},\underline{y}), d(\underline{x},\overline{y})] & \text{, se } \overline{x} < \underline{y} \\ [d\underline{x},\overline{y}), d(\overline{x},\underline{y}) & \text{, se } \overline{y} < \underline{x} \\ [0, d(\underline{x},\overline{y})] & \text{, se } X <_{km} Y \text{ e } X \cap Y \neq \emptyset \\ [0, d(\overline{x},\underline{y}) & \text{, se } Y <_{km} X \text{ e } X \cap Y \neq \emptyset \\ [0, \max(d(\overline{x},\underline{y}); d(\underline{x},\overline{y}))] & \text{, se } X \neq Y \text{ e } (X \subset Y \text{ ou} Y \subset X) \end{cases} . \quad (1)$$

Nas seções a seguir, apresentamos as variações do algoritmo K-Means. Na seção 4 são apresentados os resultados obtidos utilizando dados sintéticos e na seção 5 as conclusões.

2. Algoritmos Intervalar Baseado no K-Means - IbKM

O K-means é um algoritmo iterativo que repete dois passos: ou seja, no primeiro passo cada ponto é atribuído ao centróide mais próximo baseado em uma métrica específica escolhida; no segundo passo, uma vez que os grupos são formados, os centróides de cada grupo são atualizados . Assim, o algoritmo repete de forma iterativa esses dois passos até que os seus centróides não mudem. O Algoritmo K-Means Intervalar inicializa seus centros aleatoriamente, onde K é o número de grupos, então cada objeto é atribuido ao centro mais próximo, para realizar essa decisão uma ordem admissível total é utilizada. Em seguida é feita a atualização dos centroídes, sendo v_k o novo centroíde calculado pela equação (2), onde seu centro é um intervalo. Esse processo se repete iterativamente até que uma tolerança, ϵ seja satisfeita.

Algoritmo 1: k-Means Intervalar

Entrada: $\mathbf{X}$ - Conjunto de dados, K - n´umeros de grupos, ϵ - tolerança
Saída: $\mathbf{C} = \{C_1, \ldots, C_K\}$
Inicializa aleatoriamente os K centros: $\mathbf{v}^0 = \{v_1^0, \ldots, v_K^0\}$
repita
Classifique: Na iteração t, atribui cada objeto ($i \in \{1, \ldots, N\}$) para o agrupamento com o protótipo mais próximo:
$C^t(i) \leftarrow \arg\min_k d_{\mathbf{IMV}}(x_i, v_k)^2$; `// Usando uma ordem admissível total`
Atualização do protótipos: v_k é o centroide dos novos conjuntos:

$$\mathbf{v}_k^{t+1} = [\underline{v}_k^{t+1}, \overline{v}_k^{t+1}] = \left[\frac{\sum_{x_i \in C_k^t} \underline{x}_i}{\mid C_k^t \mid}; \frac{\sum_{x_i \in C_k^t} \overline{x}_i}{\mid C_k^t \mid} \right] \quad (2)$$

até $\|\mathbf{c}^t \ominus_{gH} \mathbf{c}^{t-1}\| \leq \epsilon$;

Na próxima seção será apresentado o K-Means Intervalar Híbrido, diferente do K-Means Intervalar ele não transforma os dados reais para dados intervalares.

3. Algoritmo K-Means Intervalar Híbrido - IbKMH

O algoritmo K-Means Intervalar Híbrido lida com conjuntos de dados híbridos, neste caso, dados que contém atributos intervalares e reais. Ele otimiza o tempo computacional do processo, pois no caso de bases híbridas o que é usualmente realizado é uma transformação de um tipo dado em outro para uma homogeneidade do conjunto de dados. Por exemplo, são utilizados dados reais e intervalares, para não perder as informações presentes nos dados intervalares é realizado uma intervalização dos dados reais, ou seja, eles são transformados em intervalos degenerados. Esse processo preserva as imprecisões presentes nesses dados, porém torna o processo muito mais caro computacionalmente. Assim uma forma de trabalhar com dados híbridos sem a necessidade de intervalar os dados reais, por exemplo, é por meio de um algoritmo que seja capaz de identificar o tipo dado e assim utilizar a distância que é computacionalmente mais adequada, ou seja, mais "barata".

Seja $m < s$, um **conjunto de dados híbrido** $C = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n\}$, onde cada objeto i descrito por s variáveis (m variáveis reais e $s - m$ variáveis intervalares), C é representado como um vetor $\mathbf{x}_i = (x_i^1, \ldots, x_i^s)$, $j \in \{1, 2, \ldots, s\}$, com $x_i^j = [\underline{x}_i^j, \overline{x}_i^j] \in \mathbb{I}(\mathbb{R})$ sendo $\underline{x}_i^j, \leq \overline{x}_i^j$, e $x_i^j \in \mathbb{R}$ onde j pertence ao conjunto de atributos reais.

$$d(\mathbf{x}_i, \mathbf{v}_k) = \sqrt{\sum_{j=1}^{s} d_{IMV_h}(x_i^j, v_k^j)^2}. \tag{3}$$

Assim, diferentemente do que é feito no algoritmo K-Means intervalar, ao calcularmos a distânciad_{IMV_h}, não é feita a transformação dos dados reais para o intervalos degenerados. Assim o algoritmo IbKMH sofre a alteração entre duas funções de distâncias quando se é calculado a d_{IMV_h}, sendo neste caso é a distância d_{km} , Eq.(1), e a distância Euclidiana, d_e.

4. Resultados e Análises dos Dados Sintéticos Híbridos

Nesta seção, são apresentados os resultados obtidos e as análises obtidas utilizando os algoritmos de agrupamentos crisp quando aplicados aos conjuntos de dados sintéticos híbridos, [da Silva 2015]: 10R90I, 30R70I, 50R50I, 70R30I e 90R10I.

A Tabela 1 abaixo, contém os resultados dos valores e o tempo gasto pelos algoritmos propostos, IbKM e IbKMH, utilizando as distâncias d_{km}, d_e e d_I. Dessa tabela, podemos concluir que o tempo computacional realmente diminui significativamente quando o algoritmo híbrido é utilizado. Na Segunda coluna da tabela 1 pode-se observar que a diferença existe apesar de pequena. No entanto na última coluna da tabela, é possível notar que o tempo diminui significativamente. Uma melhor forma de observar esse ganho de tempo computacional pode ser visto na Figura 1.

Da figura 1 pode-se observar a melhor maneira de manter as incertezas presentes nos dados intervalares e não aumentar o custo computacional transformando os dados reais em intervalos é utilizando algoritmos híbridos.

Tabela 1. Resultado dos algoritmos de agrupamento crisp com os conjuntos de dados sintéticos híbridos - CR(Std) e Tempo.

Algoritmos	10R90I	30R70I	50R50I	70R30I	90R10I
IbkM$_{d_{km}}$	1(0) 14.751	1(0) 14.412	1(0) 14.140	1(0) 14.204	1(0) 13.736
IbKM$_{d_I}$	1(0) 14.675	1(0) 14.548	1(0) 14.206	1(0) 14.261	1(0) 13.649
IbkMH$_{d_e,d_{km}}$	1(0) 13.175	1(0) 10.433	1(0) 7.982	1(0) 5.465	1(0) 2,83
IbKMH$_{d_e,d_I}$	1(0) 12.9501	1(0) 10.6188	1(0) 8.0670	1(0) 5.567	1(0) 2.822

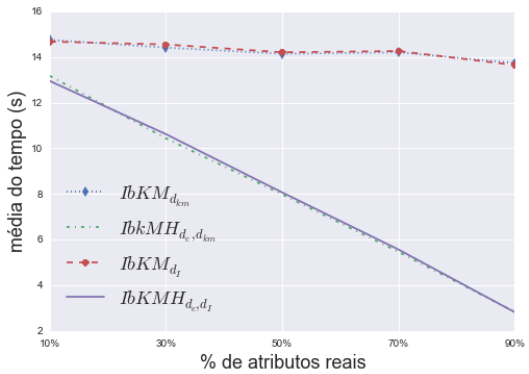


Figura 1. O gráfico descreve a relação entre tempo e a porcentagem de atributos reais, utilizando os algoritmos IbKM e IbKMH.

5. Conclusão

Os algoritmos Híbridos se mostram adquados para trabalhar com bases de dados que contenham dados reais e intervalares. Os resultados preliminares obtidos mostram um ganho computacional em relação ao tempo sem perder a qualidade das partições.

Referências

da Silva, L. R. (2015). *Uma Platarforma Intervalar para Agrupamentos de Dados.* PhD thesis, Programa de Pós-Graduação em Sistemas e Computação. Universidade Federal do Rio Grande do Norte, Natal-RN.

de Santana, F. L. (2012). *Generalizações do Conceito de Distância, i-Distâncias, Distâncias Intervalares e Topologia.* PhD thesis, Programa de Pós-Graduação em Sistemas e Computação. Universidade Federal do Rio Grande do Norte, Natal-RN.

Pereira, M. d. A. (2012). *Classificação de Dados Híbridos Através de Algoritmos Evolucionários*. PhD thesis, Universidade Federal de Minas Gerais.

Silva, L., Moura, R., Canuto, A., Santiago, R., and Bedregal, B. (2015). An interval-based framework for fuzzy clustering applications. *Fuzzy Systems, IEEE Transactions on*, PP(99):1–1.

SALSA – A Simple Automatic Lung Segmentation Algorithm

Addson Costa[1], **Bruno M. Carvalho**[1]

[1]Universidade Federal do Rio Grande do Norte - UFRN
Departamento de Matemática Aplicada - DIMAp - Brasil, Natal/RN

Abstract. *This article proposes the usage of SALSA (A Simple Automatic Lung Segmentation Algorithm), a simple and fast algorithm for the segmentation of CT volumes. The algorithm was tested with the data set of LOLA11, a lung segmentation challenge that happened at MICCAI 2011. The results are promising, being very close to the best results obtained in the challenge. We are currently improving the algorithm, in order to increase its accuracy level.*

Resumo. *Este trabalho propõe o uso do SALSA (A Simple Automatic Lung Segmentation Algorithm), um algoritmo simples e rápido para a segmentação de pulmões em volumes de CT. O algoritmo foi testado no conjunto de dados do LOLA11, um desafio de segmentação de pulmões que ocorreu durante o MICCAI 2011. Os resultados são animadores, estando bem próximos dos melhores resultados obtidos no desafio. Aprimoramentos estão sendo feitos no algoritmo atual para que possamos melhorar seu nível de acurácia.*

1. Introdução

Desde sua introdução, o uso de imagens de tomografia computadorizada (CT, do Inglês *Computerized Tomography*) no diagnóstico têm revolucionado a prática médica. Entretanto, a análise de imagens tomográficas pulmonares é uma tarefa de alto consumo de tempo devido ao grandes volumes de dados, que devem ser segmentados. Além disso, estes volumes podem ter sua qualidade afetada devido à diversas anomalias.

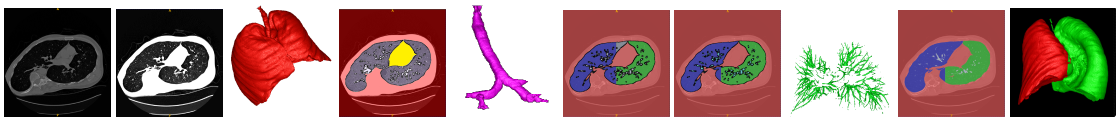
Existem vários algoritmos automáticos para resolver este problema, mas com diferentes acurácias e consumo de tempo associados. A técnica proposta neste artigo utiliza o algoritmo FloodFill [Gonzalez and Woods 2002] para segmentar a imagem baseado tanto em intensidades e vizinhanças espaciais. Nosso objetivo é conseguir obter uma segmentação simples, rápida e precisa. Para tal, desenvolvemos um método que é composto por operações simples e bem conhecidas de processamento de imagens.

2. Trabalhos Relacionados

Até este momento, 15 outros grupos publicaram resultados no website LOLA11[1]. Alguns grupos estão listados abaixo incluindo uma rápida descrição dos seus algoritmos. O grupo *Human*, usa um segundo observador para manualmente segmentar todas as imagens e estudar a variabilidade entre estas anotações e o padrão de referência do LOLA11.

Os métodos propostos por [Lassen et al. 2011] e [Weinheimer et al. 2011] se utilizam de algoritmos de crescimento de regiões para segmentar os pulmões após buscarem por estruturas tubulares para identificar a traqueia, seguidos de operações de fechamento para a inclusão de vasos e regiões com patologias. O método de [Pinho et al. 2011] faz

[1][van Rikxoort et al. 2014] http://lola11.com



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figura 1. Etapas do SALSA.

a localização da traqueia, crescimento de regiões adaptativo, segmentação baseada em corte gráfico, crescimento de regiões e operações morfológicas para produzir a segmentação final. Já o método de [Korn et al. 2011] faz uma segmentação inicial, que é refinada nas bordas com outros objetos para então refinar a separação entre pulmões. A técnica de [Sun et al. 2011] usa um algoritmo de detecção de costelas para iniciar um modelo de forma ativo para segmentar os pulmões de modo aproximado e depois adaptar o modelo para os dados da imagem por meio de um algoritmo de encontrar uma superficie ótima.

O método de [Gu and Pu 2011] executa uma segmentação de pulmões convencional e busca por erros. Se forem encontrados, o método inicia uma busca por referências importantes como costelas e bordas entre pulmões. Então, um método de registro elástico é aplicado para ajustar um *template* predefinido. O método de [Hosseini-Asl et al. 2011] remove o exterior do corpo humano, para depois modelar a imagem usando *NMF (Non-negative Matrix Factorization)*, onde uma matriz 4D é gerada incluindo para cada voxel, os sinais de sua vizinhança. A segmentação é então realizada usando o algoritmo *KMeans*.

3. SALSA

O método SALSA (acrônimo de *A Simple Automatic Lung Segmentation Algorithm*) é composto de várias fases que empregam operações simples de Processamento de Imagens, como filtros espaciais básicos, para realizar a tarefa de segmenção dos pulmões em volumes de CT. As etapas do método são descritas a seguir.

Primeiro, enquanto a imagem original (Figura 1a)[2] é carregada para a memória, é aplicado um algoritmo de limiarização (Figura 1b) seguido de uma busca por sementes, que segue as linhas sobre o plano axial tentando achar dois voxels com a maior quantidade de pulmão possível entre eles na melhor fatia encontrada.

As sementes encontradas no passo anterior são usadas para fazer uma busca pela borda [Gonzalez and Woods 2002] que isola o sistema respiratório (Figura 1c), mantendo os pulmões e traqueia dentro da borda. Os vasos estão dentro do volume do pulmão, porém fora da borda encontrada previamente (Figura 1c). Então nós primeiro pintamos o lado externo a borda usando processamento em fatias axiais, originando uma parte da segmentação do fundo (Figura 1d - vermelho). Depois adicionamos partes do fundo encobertas próximo a traqueia para completar a segmentação do fundo (Figura 1d - amarelo).

A traqueia é encontrada pela anatomia baseada em um cilindro, buscando-se em ambas as direções (Figura 1e). Para evitar vazamentos, fazemos um corte no meio da imagem e a partir de duas sementes (esquerda e direita da imagem), e pintamos os voxels conectados como pulmões esquerdo e direito, o que origina uma segmentação de pulmões parcial sem atravessar o corte (Figura 1f). Depois, os voxels no corte são analisados para complementar os pulmões otimizando-se a área do corte usando uma busca local (Figura

[2]Todas as imagens neste artigo foram geradas usando o ITK-SNAP [itk 2014].

1g). Vários cortes independentes podem ser movidos em direções diferentes em cada imagem porque os pulmões podem estar conectados em muitos pontos diferentes.

Depois, segmentamos os voxels que representam os vasos, evitando incluir áreas danificadas por patologias no pulmão, gerando uma imagem de melhor qualidade para busca por danos causados por doenças (Figura 1h). Repintando a borda como traqueia, pulmão esquerdo ou direito, nós temos a segmentação final (Figura 1i), que é ajustada para o padrão de referência usado no LOLA11 (Figura 1j).

4. Discussão e Resultados

Os resultados abaixo foram obtidos aplicando-se o SALSA às 55 imagens do banco de dados do LOLA11, usando um notebook I5-3337U CPU rodando a 1.8Ghz, com 4Gb de RAM DDR3-1600 e um disco SSD. A Tabela 1 mostra os índices de sopreposição, utilizando o método de Maximum Overlap, dos volumes segmentados em relação aos volumes de referência do LOLA11, onde *média* indica o valor médio, *dp* o desvio padrão, *min* o valor mínimo, *Q1* o valor do primeiro quartil, *med* o valor da mediana, *Q3* o terceiro quartil, e *max* o valor máximo dos índices de sobreposição para os 55 volumes.

Tabela 1. Sobreposição para cada pulmão, para 55 imagens em LOLA11.

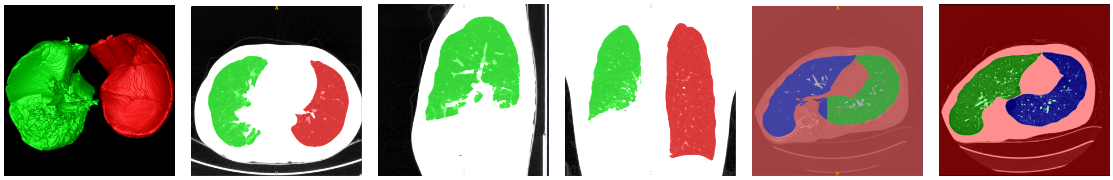
obj	média	dp	min	Q1	med	Q3	max
Pulmão Esquerdo	0.960	0.102	0.241	0.972	0.981	0.987	0.995
Pulmão Direito	0.959	0.132	0.020	0.977	0.984	0.989	0.994
score	0.959						

A Tabela 2 mostra os tempos de execução por volume e por voxel do SALSA quando aplicado aos volumes do LOLA11. Os tempos de execução estão expressos em segundos, para os volumes, e em microsegundos, para os voxels.

Tabela 2. Tempo de execução gasto pelo SALSA na segmentação dos 55 volumes do LOLA11.

obj	média	dp	min	Q1	med	Q3	max
volume (s)	47.13	38.62	10.32	31.03	37.17	52.45	248.66
voxel (μs)	0.435	0.393	0.156	0.413	0.359	0.354	3.040

As Figuras 4a, 4b, 4c e 4d mostram os resultados para o volume lola11-02, onde podemos ver o pulmão degradado na parte inferior. Usando o SALSA nós podemos ver isto claramente. Outros métodos encontram uma superfície suave ao redor do pulmão, mascarando possíveis doenças.



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

Figura 2. Resultados do SALSA. Segmentação de um pulmão com doença: (a) Renderização 3D vista de baixo para cima e fatias (b) axial, (c) sagital e (d) coronal do volume lola11-02. Utilização do filtro de Sobel para obter uma melhor separação dos pulmões. Resultado (e) sem uso do filtro na detecção de borda, (f) com o filtro de sobel. Imagem gerada usando a imagem lola11-05.

Na competição LOLA11, o melhor resultado foi alcançado pela segmentação totalmente manual executada pelo grupo *Human* seguindo o protocolo fornecido pela competição, o que resultou na sobreposição de 0.984. Os resultados atuais em LOLA11 usando segmentação automática são descritos na Tabela 3. O SALSA obteve um indíce

de sobreposição que o posicionou no meio da classificação, tendo ficado a apenas 0.021 pontos abaixo do índice de sobreposição do melhor método de segmentação automático.

Como os tempos de execução dos métodos foram calculados em máquinas diferentes, fica difícil de se comparar os tempos de execução de maneira justa. Porém, o SALSA só necessita de 47 segundos em média, sendo várias vezes mais rápido do que os concorrentes na maioria dos casos.

Tabela 3. Tempo de execução gasto pelo SALSA na segmentação dos 55 volumes do LOLA11.

obj	SALSA	Weinheumeier	Lassen	Pinho	Korn	Sun	Gu	Hosseini
acurácia	0.959	0.97	0.973	0.948	0.949	0.949	0.939	0.965
tempo (s)	47	684	60	NA	756	360	NA	NA

Para conseguirmos melhores resultados, nós estamos aperfeiçoando os métodos utilizados para executar as etapas descritas anteriormente. Especificamente, adicionamos a possibilidade de adicionar voxels utilizando o filtro de Sobel na busca da borda, o que melhorou consideravelmente os resultados, conseguindo uma definição perfeita separando os pulmões para 46 das imagens e uma definição mais precisa para as demais imagens. A Figura 4e mostra a separação dos pulmões obtida sem o uso do filtro de Sobel, enquanto que a Figura 4f mostra o resultado da separação dos pulmões obtida com o uso do filtro de Sobel. O SALSA+ está em fase de testes, onde alguns ajustes estão sendo feitos para que possamos melhorar sua acurácia.

5. Conclusões

O SALSA é um algoritmo simples, rápido, totalmente automático, e que funciona também em pulmões com alguns quadros patológicos, como demonstrado acima. O algoritmo não necessita de nenhuma entrada, e não mascara anomalias pulmonares. Trabalhos futuros serão concentrados em aumentar a acurácia obtida pelo algoritmo, bem como na segmentação dos lóbulos pulmonares, uma tarefa bem mais complicada, devido ao baixo contraste que as membranas que delimitam os lóbulos apresentam em exames de CT.

Referências

(Acessed in November, 2014). ITK-SNAP. www.itksnap.org.

Gonzalez, R. and Woods, R. (2002). *Digital Image Processing*. Prentice Hall, second edition.

Gu, S. and Pu, J. (2011). Automatic lung segmentation improvement using a registration framework. In *4th Int. Works. on Pulm. Im. Anal. (at MICCAI 2011)*.

Hosseini-Asl, E., Zurada, J. M., and El-Baz, A. (2011). Lung segmentation using incremental sparse nmf. In *4th Int. Works. on Pulm. Im. Anal. (at MICCAI 2011)*.

Korn, R., Kim, J., Schmidt, G., and Binnig, G. (2011). Description of a fully automatic lung segmentation algorithm based on the cognition network tecnology. In *4th Int. Works. on Pulm. Im. Anal. (at MICCAI 2011)*.

Lassen, B., Kuhnigk, J.-M., Schmidt, M., Krass, S., and Peitgen, H.-O. (2011). Lung and lung lobe segmentation methods at fraunhofer mevis. In *4th Int. Works. on Pulm. Im. Anal. (at MICCAI 2011)*.

Pinho, R., Delmon, V., and Vandemeulebroucke, J. (2011). Keuhkot: A method for lung segmentation. In *4th Int. Works. on Pulm. Im. Anal. (at MICCAI 2011)*.

Sun, S., Bauer, C., and Beichel, R. (2011). Robust active shape model based lung segmentation in ct scans. In *4th Int. Works. on Pulm. Im. Anal. (at MICCAI 2011)*.

van Rikxoort, E., van Ginneken, B., and Kerkstra, S. (Acessed in November, 2014). Lobe and lung analysis 2011.

Weinheimer, O., Achenbach, T., Heussel, C. P., and Düber, C. (2011). Automatic lung segmentation in mdct images. In *4th Int. Works. on Pulm. Im. Anal. (at MICCAI 2011)*.

>Artigos de Demonstração e Poster<

Abordagem heurística e exata para o problema da árvore geradora de rotulação mínima

Thiago S. Marques[1], **Sidemar F. Cezario**[1], **Lucas S. M. Cardozo**[1], **Elizabeth F. Gouvêa**[2]

[1]Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte (UFRN)
Natal, RN – Brasil

[2]Departamento de Informática e Matemática Aplicada
UFRN – Natal, RN – Brasil

{thiago.smarques.bti,sidemar.r7,lucassmcardoso}@gmail.com,

beth@dimap.ufrn.br

Abstract. *Many problems that exists today can be solved with computational aide, although some problems, more specifically the NP-Hard problems, do not have known algorithms that find the exact answer to the problem in a viable computational time. Thus, many researches have been made with the objective of finding approximated solutions to these kind of problems. This paper presents a comparative analysis between two approaches, the first being exact while the second being greedy, to the minimum labelling spanning tree problem.*

Resumo. *Muitos problemas da atualidade podem ser resolvidos com apoio computacional, entretanto para alguns problemas, mais especificamente os problemas da classe NP-Árduo, não são conhecidos algoritmos que encontrem a resposta exata para o problema em um tempo computacional viável. Assim, muitas pesquisas vem sendo feitas no sentido de encontrar soluções aproximadas para esses problemas. Esse trabalho apresenta uma análise comparativa entre duas abordagens, sendo uma exata e outra gulosa para o Problema da Árvore Geradora de Rotulação Mínima.*

1. Introdução

Esse trabalho tem como objetivo explorar algoritmos gulosos e exatos, mais especificamente que resolvam o problema da árvore geradora de rotulação mínima(AGRM). Assim, o problema da AGRM será descrito e, em seguida, serão apresentados os algoritmos propostos, bem como os resultados computacionais.

2. Descrição do problema

O problema da árvore geradora de rotulação mínima foi introduzido por [Chang and Leu 1997] o qual também demonstrou ser um problema NP-difícil. Ele pode ser definido da seguinte maneira: dado um grafo não dirigido $G = (V, E)$, onde V é o conjunto de vértices, E o conjunto de arestas rotuladas ou coloridas, construa uma árvore geradora com o menor número de rotulações ou cores possíveis. Exemplos de aplicações podem ser encontrados em [Consoli et al. 2008].

3. Descrição da técnica de solução

O algoritmo exato primeiramente gera todas as permutações de arestas possíveis, em seguida é aplicada para cada permutação uma versão adaptada do algoritmo de Kruskal [Kruskal 1956], para construir uma árvore geradora. Dessa forma, são construídas todas as árvores geradoras possíveis, depois basta contar o número de rotulações de cada árvore e guardar a que utilizou menos rotulações.

O algoritmo guloso consiste em ordenar as arestas de acordo com a quantidade de vezes que a rotulação aparece no grafo, ou seja, criar uma nova lista de arestas em ordem decrescente pelo número de rotulações. Depois, utiliza-se uma técnica semelhante ao Algoritmo de Kruskal para construir uma árvore geradora.

4. Resultados computacionais

Os experimentos computacionais exibidos na tabela 1 foram executados em uma máquina com as configurações a seguir: Processador Intel Core i5 1.60GHz, 6 GB de RAM, Sistema operacional Ubuntu e a linguagem Java. As rotulações das arestas das instâncias testadas foram gerados aleatoriamente com apoio do software Microsoft Excel.

Tabela 1. Resultados

V	E	Rotulações	Guloso (ms)	Exato (ms)	Guloso	Exato
4	6	4	3	15	2	2
5	5	3	2	5	2	2
6	11	4	2	15006	2	2
7	12	4	2	168169	3	2
6	13	5	3	2131128	1	1

As duas últimas colunas informam a quantidade de rotulações para gerar a árvore. Dessa forma, podemos perceber que a medida que o tamanho das instâncias aumenta, o tempo de execução do algoritmo exato cresce exponencialmente, enquanto que o heurístico toma um tempo polinomial.

5. Conclusões

Tendo em vista os aspectos observados, pode-se constatar a diferença de tempo computacional entre o algoritmo exato e o heurístico. O presente artigo mostra-se interessante para ser apresentado para estudantes que ainda não tiveram contato com problemas NP-difíceis. O próximo passo é desenvolver algoritmos meta-heurísticos para o problema.

Referências

Chang, R. S. and Leu, S. J. (1997). The minimum labeling spanning trees. page 277–282. Information Processing Letters 63.

Consoli, S., Darby-Dowman, K., Mladenovic, N., and Pérez, J. M. (2008). Greedy randomized adaptive search and variable neighbourhood search for the minimum labelling spanning tree problem. page 440–449. European Journal of Operational Research.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. page 48–50. Proceedings of the American Mathematical Society.

Desenvolvimento de Algoritmos para o Problema da Maior Subsequência Comum

Luis Tertulino da C. Neto[1], **Elizabeth F. G. Goldbarg**[2]

[1]Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte (UFRN)

[2]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN)

luistertulino95@gmail.com, beth@dimap.ufrn.br

Abstract. *This paper shows the work of building an exact algorithm and a greedy algorithm for the Longest Common Subsequence (LCS) problem, as well as the results obtained.*

Resumo. *Este artigo apresenta o trabalho de construção de algoritmos exato e guloso para o problema da Maior Subsequência Comum (MSC), bem como os resultados obtidos.*

1. Introdução

O Problema da Maior Subsequência Comum (MSC) foi apresentado por David Maier em 1978 [Maier 1978] e desde então é aplicado em diversas áreas de pesquisa e ferramentas computacionais.

O problema é definido como: dado um conjunto S de sequências S_i e um inteiro k, descobrir se existe uma subsequência comum a S cujo tamanho seja maior ou igual a k. Sua versão de otimização consiste em achar a maior subsequência. Em seu artigo sobre o problema, Maier mostrou que ele é NP-completo.

2. Técnicas utilizadas

Foram desenvolvidos dois algoritmos para resolver o problema: um exato e um guloso.

O algoritmo exato é enumerativo: todo o espaço de soluções é gerado, e calcula-se a maior subsequência comum. Esse espaço de soluções consiste em todas as subsequências da menor string. Então verifica-se quais delas são comuns ao conjunto, e sempre que uma nova é gerada, compara-se seu tamanho com as anteriores. Isso signifca que, caso exista mais de uma subsequência comum de tamanho máximo, todas elas sejam retornadas.

Como neste algoritmo geramos todas as subsequências da menor sequência, de tamanho n, temos uma complexidade inicial $\Theta(2^n)$. Além disso, para cada uma delas, verificamos se são uma subsequências comum em $\Theta(m * n)$, onde m é o número de sequências do conjunto. No fim, a complexidade é $\Theta(m * n * 2^n)$.

O algoritmo guloso adota a seguinte estratégia: dado que a maior subsequência comum possui tamanho menor ou igual ao tamanho da menor sequência do conjunto, definimos a seguinte função recursiva:

$$GreedyLCS(S_1, S_2, S_3..., S_n) = GreedyLCS(2LCS(S_1, S_2), S_3..., S_n)$$

onde $GreedyLCS$ é a função criada para aproximar uma maior subsequência comum, e $2LCS$ é o algoritmo de programação dinâmica para duas sequências, apresentado em [Cormen 2012].

Supondo que, no pior caso, as strings do conjunto tenham tamanho n e existam l strings nesse conjunto, o algoritmo $2LCS$ demora tempo $\Theta(n^2)$; como essa operação é realizada l vezes, a complexidade do algoritmo guloso é $\Theta(n^2 * l)$.

3. Resultados computacionais

Foram realizados testes computacionais de ambos os algoritmos utilizando sequências selecionadas aleatoriamente dos repositórios do UniProtKB/Swiss-Prot (`http://web.expasy.org/docs/swiss-prot_guideline.html`) e do National Center for Biotechnology Information: (`http://www.ncbi.nlm.nih.gov/genome/`). Os testes foram realizados em um PC com CPU de 1,80 GHz quad core e 6GB de memória, em um sistema Linux. O algoritmo foi implementado em C++. Os resultados são exibidos na tabela a seguir, com a seguinte descrição: $|S|$ é o tamanho do conjunto de sequências, $|S_1|$ é o tamanho da menor sequência do conjunto, $|LCS_{exato}|$ é o tamanho da LCS obtida com o algoritmo exato, $|LCS_{guloso}|$ é o tamanho da LCS obtida com o algoritmo guloso, t_{exato} é o tempo de execução do algoritmo exato, e t_{guloso} é o tempo de execução do algoritmo guloso.

Table 1. Resultados computacionais dos algoritmos

Instância	$\|S\|$	$\|S_1\|$	$\|LCS_{exato}\|$	$\|LCS_{guloso}\|$	t_{exato}	t_{guloso}
1	149	5	0	0	513 μs	189 μs
2	408	10	0	0	18.861 ms	283 μs
3	204	20	1	0	10 s	349 μs
4	43	30	0	0	1h 13min 35s 420ms	90 μs
5	44	24	0	0	1 min	155 μs
6	500	18	12	12	8.275 s	155 ms
7	68 542	18	6	6	11min 08s 500ms	206.806ms
8	20	14	17	16	375.854 ms	831 μs
9	158	12	8	8	382.696 ms	3.963 ms
10	103 699	10	0	0	3.722 s	56.038 ms
11	2 204 949	15	0	0	42 min 35 s 590 ms	1.72 s

4. Conclusão

Com base nos resultados apresentados na seção anterior, chegamos à conclusão de que o algoritmo guloso oferece uma resposta razoável para o Problema da Maior Subsequência Comum, em comparação à solução exata; embora na maioria dos casos não exista uma solução, quando esta existiu, o algoritmo guloso retornou um resultado com no máximo um caractere a menos.

Sistema em Arduino de Detecção de Queda para Idosos

João S. Belau[1], Diego O. Lemos[1], Monica M. Pereira[2]

[1]Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte (UFRN) – Natal, RN – Brasil

[2]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN) – Natal, RN – Brasil

{joaosb989,diegoifrn}@gmail.com, monicapereira@dimap.ufrn.br

Abstract. *This paper aims to present a fall detection system for the elderly using the Arduino technology and the MMA7361L accelerometer. The system is simple in construction and it was designed to be a wearable technology. The project is in progress, but it is already possible to detect free fall with the prototype built.*

Resumo. *Este artigo objetiva-se a apresentar um sistema de detecção de queda para idosos utilizando a tecnologia Arduino e o Acelerômetro MMA7361L. O sistema é de simples construção e foi pensado para ser uma tecnologia vestível. O projeto encontra-se em andamento, porém já é possível detectar queda livre com o protótipo construído.*

1. Introdução

A medida que uma pessoa vai envelhecendo seu organismo vai passando por alterações que refletem no cotidiano, especialmente o aumento da possibilidade de o indivíduo sofrer queda devido aos inúmeros fatores de risco que surgem (Machado et al., 2009). A ocorrência de queda em idosos pode trazer consequências drásticas a saúde dessas pessoas. O auxílio imediato a um idoso que sofreu uma queda é um fator determinante para sua saúde e pode até salvar a sua vida.

Nesse sentido, já existem pessoas que acompanham os idosos diariamente e também já foram criados sistemas de monitoramento de câmeras. Porém essas soluções são intrusivas e muitas vezes causam desconforto nos idosos que passam a se sentir cada vez mais incapazes. Assim, esse trabalho visa construir um sistema de detecção de queda para idosos de baixo custo, não intrusivo e que traga mais autonomia e segurança para essas pessoas.

2. Proposta

O sistema embarcado para detecção de quedas é composto por um Arduino, um acelerômetro, um *buzzer* e um software que ao detectar a queda, deve comunicar através de mensagem para algum usuário (parente ou profissional de saúde) sobre a queda do idoso. Arduino é uma plataforma eletrônica de código aberto baseado em hardware e software de fácil utilização (Arduino, 2015). Um acelerômetro é um dispositivo que mede a aceleração de um corpo em relação à gravidade. O acelerômetro é o modelo MMA7361L e foi escolhido por ter baixo consumo de energia e por permitir detectar queda livre linear em três eixos.

Para o funcionamento correto do sistema, o acelerômetro deve estar com um de seus eixos voltado para cima, mais próximo da inclinação perpendicular possível em relação ao chão. Essa seria a condição ideal na qual a pessoa estaria em pé ou sentada. Por isso definiu-se que o melhor local para dispor o protótipo e realizar os testes seria o tórax. A imagem a seguir mostra o esquema de ligação do protótipo.

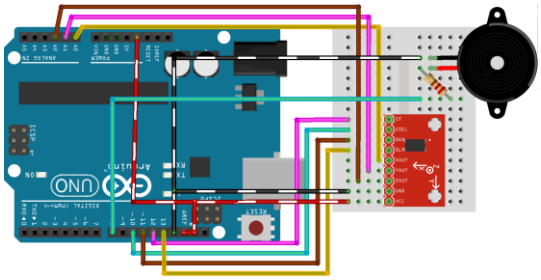


Figura 1. Esquema de ligação dos componentes do protótipo

A detecção da queda ocorre quando a pessoa sofre uma inclinação do seu tórax rapidamente em direção ao chão. Foi implementado um algoritmo no Arduino que permite o sensor detectar essa variação dentro de um intervalo de tempo e tomar uma ação caso a variação do eixo seja alta em um curto período de tempo. Caso ocorra a queda, um *buzzer* é acionado emitindo um som por um curto período de tempo indicando a queda. O Arduino também pode se conectar à Internet para enviar uma mensagem a outro usuário notificando sobre a queda.

Foram realizados alguns testes com o protótipo ligado a uma bateria de 9 volts no intuito de melhorar a precisão da leitura das variáveis e tentar diferenciar o que seria uma queda de um movimento normal. Nesses testes foi possível inferir que o protótipo responde bem a uma grande variedade de movimentos e que são raros os casos de falsos positivos para um sistema nas condições ideais.

3. Conclusão

Apesar do resultado desse trabalho ser parcial, conclui-se que eles já são de grande importância para o contexto de aplicação apresentado. O sistema já consegue identificar queda livre, porém é necessário, como próximo passo, melhorar a precisão na leitura e interpretação dos dados para que o maior número de formas de quedas possíveis seja identificável.

Também será objetivo nos próximos passos deste projeto integrar outros mecanismos de monitoramento ao protótipo, como sensor de temperatura e batimento cardíaco por exemplo, no intuito de proporcionar mais segurança e liberdade para os idosos. Embora o projeto esteja sendo testado para detecção de quedas de idosos, o sistema pode ser usado para a detecção de queda de pessoas em diferentes idades com problema de saúde, como o caso de epilepsia, desmaios, etc.

4. Referências

Machado, TR., Oliveira, CJ., Costa, FBC. e Araujo, TL. (2009) “Avaliação da presença de risco para queda em idosos”, http://www.fen.ufg.br/revista/v11/n1/v11n1a04.htm, Abril.

Arduino (2015), http://www.arduino.cc, Abril.

Connected Garden: Um Jardim Inteligente

Alysson R. O. de Lima[1], Raul S. Silva[2], Wanderson R. de Medeiros[3] , Monica M. Pereira[3]

[1]Escola de Ciências e Tecnologia - Universidade Federal do Rio Grande do Norte (UFRN) , Natal-RN-Brasil

[2]Instituto Metrópole Digital - Universidade Federal do Rio Grande do Norte (UFRN), Natal-RN-Brasil

[3]Departamento de Informática e Matemática Aplicada - Universidade Federal do Rio Grande do Norte (UFRN), Natal-RN-Brasil

alysson_lima@bct.ect.ufrn.br,{raul95,Wanderson.medeiros2}@gmail.com, monicapereira@dimap.ufrn.br

Resumo. *Connected Garden é um sistema voltado para ajudar no monitoramento e irrigação de jardins domésticos. Cada vaso possui sensores e atuadores para o monitoramento individual e irrigação, bem como conexão bluetooth usada para troca de informações. O sistema é responsável pelo gerenciamento das atividades relacionadas ao controle de irrigação. Além disso, também processa todos os dados coletados e notifica o usuário sobre as atividades e condições ambientais. Para proporcionar ainda mais flexibilidade, o sistema também contará com uma aplicação mobile que permite monitorar e configurar os vasos de qualquer lugar. O usuário ainda poderá optar pelo controle manual da irrigação.*

1. Introdução

Jardins ornamentais e hortas de pequeno porte estão ficando cada vez mais comuns em residências e prédios comerciais. Cada vez mais pessoas moram em apartamentos ou casas condominiais e trabalham em escritórios fechados, onde o terreno para cultivo de plantas é reduzido ou mesmo inexistente, isso faz com que se aumente a procura por lojas especializadas em plantas ornamentais de pequeno porte para se montar um jardim. Um dos problemas frequentes relacionados à essa prática é a falta de informação a respeito do cultivo das plantas. Embora existam plantas bastante resistentes a condições adversas, como excesso ou falta de água e exposição ao sol, como a flor gérbera [Marcondes 2012], muitas plantas são mais sensíveis e não resistem quando expostas a essas condições adversas [Ludwig 2013]. O resultado disso é um jardim que exige demasiada manutenção. Como consequência, o proprietário insatisfeito com o resultado do seu esforço e investimento desiste do cultivo do jardim. É para estes usuários que o Connected Garden foi pensado.

2. Proposta

O sistema consistirá de quatro módulos: os vasos embarcados, o concentrador, aplicação mobile e a aplicação web. Cada vaso embarcado possuirá um conjunto de sensores e atuadores. Os sensores irão captar as informações da planta (luminosidade,

umidade do solo/ar, temperatura e etc) e as enviar para o concentrador, e os atuadores serão parte do sistema de irrigação. O concentrador não apenas captará as informações dos vasos como também poderá mudar configurações dos sensores e atuadores. No concentrador, as informações das plantas serão pré-processadas e repassadas para um servidor. É nesse servidor que se encontrará uma aplicação web em Java onde o usuário poderá efetuar o login e consultar as informações de cada planta do jardim via browser ou aplicação mobile Android. Além de prover informações dos vasos, a aplicação web permitirá que o usuário cadastre "alertas" para lembrá-lo sobre cuidados que ele deve ter, por exemplo, adubação, dias de poda e etc.

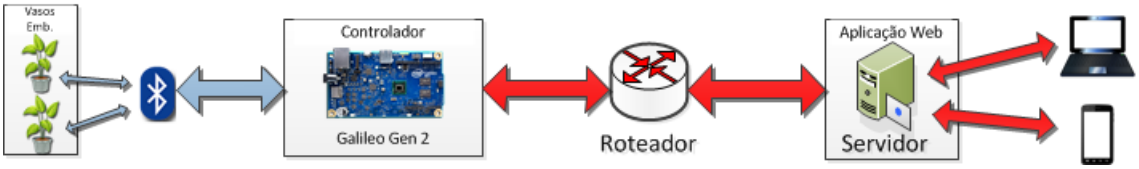


Figura 1: Arquitetura do Sistema

A Figura 1 apresenta a arquitetura proposta. A comunicação dos vasos com o concentrador será feita via bluetooth. O concentrador, implementado utilizando a placa Intel Galileo, irá enviar e receber as informações de um servidor web que armazenará estes dados, que por sua vez serão acessados por dispositivos rodando a aplicação mobile ou através de um web browser.

3. Resultados

Após alguns meses de trabalho e diversos testes (como leituras de umidade do ar, solo, luminosidade, e comunicação via bluetooth) obteve-se um protótipo capaz de medir a umidade do solo e a partir disso disparar a irrigação automática. Além disso, o protótipo consegue enviar informações para uma aplicação mobile através de uma intranet. Essas duas etapas do sistema já estão concluídas. Atualmente, está sendo implementada a interface web, e o projeto do vaso embarcado está sendo concluído. Alguns testes em relação à umidade do solo e escoamento da água estão sendo realizados para verificar a atuação do sistema de irrigação em casos reais.

4. Conclusão

O Connected Garden é um sistema que se propõe a auxiliar pessoas que desejam cultivar jardins e hortas de pequeno porte através do monitoramento e irrigação automatizada das plantas. A ideia do sistema é extrair informações de cada vaso do jardim e disponibilizar estas informações ao usuário através de uma aplicação mobile e uma interface web para que de posse destas informações o usuário possa tomar melhores decisões a respeito de como cuidar de cada planta de forma mais otimizada.

5. Referencias

MARCONDES, I. (2002) "A influência da urbanização na distribuição da vegetação na cidade de Curitiba - Paraná", Dissertação ao Curso de Pós-Graduação em Engenharia Florestal, Setor de Ciências Agrárias, UFPR, Curitiba - PR.

LUDWIG, F; et al. (2013) "Absorção de nutrientes em cultivares de gérbera cultivada em vaso", Horticultura Brasileira, p. 622-627.

O Problema do Caixeiro Viajante e Exemplos de Algoritmos

Ronaldo F. Silveira[1], **Elizabeth Ferreira Gouvêa Goldbarg**[2]

[1]Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte (UFRN)

[2]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN)

ronaldof1s1@gmail.com, beth@dimap.ufrn.br

Abstract. *This paper aims to present the Traveling Salesman Problem (TSP) and some algorithms implemented for it. Among those, there is an enumerative and a greedy algorithm*

Resumo. *O presente artigo tem por objetivo apresentar o Problema do Caixeiro Viajante (PCV) e alguns algoritmos implementados para tal. Dentre esses algoritmos, tem-se um enumerativo e outro guloso.*

1. Introdução

O Problema do Caixeiro Viajante (PCV) não possui data definida, mas estima-se que no século XIX ocorreu seu surgimento entre mercadores. Entretanto, o problema tornou-se mundialmente famoso em 1950 [Applegate 2006]. Este pode ser formulado como: dado um grafo $G = (V, E)$, onde V é o conjunto de vértices e E o conjunto de arestas, e um custo atribuído a cada aresta, encontrar o circuito hamiltoniano cujo somatório dos custos de suas arestas seja mínimo dentre todos os possíveis circuitos hamiltonianos de G. Richard Karp demonstrou que o problema do ciclo hamiltoniano é NP-Completo[Karp 1972], implicando em mostrar que o Caixeiro Viajante é um problema NP-Difícil. Este trabalho apresenta um algoritmo guloso para o problema e compara seus resultados aos de um algoritmo enumerativo implementado.

2. Descrição dos Algoritmos

Para a descrição destes algoritmos, leva-se em conta que N representa o número de vértices no grafo. O algoritmo enumerativo consiste em gerar todas as $N-1$ permutações de caminhos que formam um ciclo hamiltoniano começando pelo vértice V_0. Para cada uma das permutações, o algoritmo checa o tamanho da permutação (percorrendo a permutação e somando o peso das arestas) e armazena a menor de todas. O algoritmo de geração de permutações tem custo $O(N!)$, como é usado em $N-1$ vértices (excluindo o V_0), torna-se $O((N-1)!)$. Já o algoritmo usado para checar o tamanho do ciclo formado pela permutação executa em tempo quadrático, em relação ao número de vértices, para cada uma das permutações. Assim temos que o algoritmo total tona-se $O(N^2 \cdot (N-1)!)$, ou seja $O(N \cdot N!)$.

O algoritmo guloso trabalha utilizando uma alternância entre duas heurísticas. Até que todos os nós estejam no ciclo, o algoritmo alterna entre inserir o nó mais próximo ou

os dois nós que formam a aresta de custo mínimo no ciclo desejado, marcando, em seguida, os nós inseridos como visitados. O algoritmo realiza essas operações tomando cada vértice como o inicial, gerando resultados diferentes e comparando-os. O algoritmo supracitado, trabalhando com essas duas heurísticas de complexidades $O(N)$ e $O(N^2)$ respectivamente, possui complexidade da ordem de $O(N^4)$ por fazer essas operações alternadas N vezes e recomeçar com cada vértice sendo inicial (N vértices).

3. Resultados Computacionais

Para o algoritmo exato, foram criadas instâncias com números compatíveis de vértices, e então comparadas aos resultados obtidos pelo algoritmo guloso. Os experimentos computacionais foram feitos em uma máquina Acer, com processador intel i7-4510U 2.0GHz, com 8GB de memória. O sistema operacional é Windows 8.1 64bits. A tabela 1 apresenta o número de vértices de cada instância (sendo cada uma delas um grafo completo), o resultado exato (R_{exato}), o tempo que o algoritmo exato levou para ser executado (t_{exato}), o resultado obtido com o algoritmo guloso (R_{guloso}) e o tempo de execução deste (t_{guloso}).

instância (número de vértices)	R_{exato}	t_{exato}(ms)	R_{guloso}	$t_{guloso}(ms)$
3	145	0	145	0.4
4	168	0	168	0.5
5	200	0	200	2.6
6	114	16	151	1.7
7	209	47	246	1.8
8	153	250	167	2.9
9	197	2266	226	3.1
10	167	24579	234	5.11
11	158	299107	251	7

Tabela 1. Algoritmos Exato e Guloso

Assim, conclui-se que podem ser obtidos resultados muito diferentes entre os dois algoritmos. Com o primeiro (enumerativo) sendo executado muito mais lentamente em relação ao segundo (guloso), mas chegando a um resultado de caminho muito menor.

4. Conclusão

Pode-se observar, após a exposição de algoritmos com o propósito de solucionar o Problema do Caixeiro Viajante (PCV, um problema NP-Completo), certa discrepância entre os resultados verificados em termos de tempo de execução e de qualidade de solução obtida. Dessa forma, conclui-se que o algoritmo exato é capaz de apresentar a solução ótima quando aplicado a pequenas instâncias, entretanto, à medida que o algoritmo guloso resulta em respostas que apresentam valores maiores do que os ideais, a despeito de expressar tais soluções mais rapidamente.

Referências

Applegate, D. L. e. a. (2006). The travelling salesman problem: a computational study. *Princeton: Princeton University Press*.

Karp, R. M. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*.

>Resumos de Demonstração e Poster<

Trabalhos apresentados na sessão de Demonstração e Poster, cuja submissão ocorreu em chamada específica sem submissão de artigo.

Proposição de Tecnologia Assistiva de Auxílio à Leitura para Deficientes Visuais com Baixa Visão

Autores: Anderson Bezerra de Oliveira, Monica Magalhães Pereira e Gustavo Girão

Resumo: Atualmente a tecnologia já está imersa em praticamente todas as áreas da sociedade. Dentre diversas aplicações e ramificações de seu uso, há uma aplicação que se destaca por sua importância e impacto direto na sociedade, conhecida como tecnologia assistiva – ciência aplicada visando inclusão social e qualidade de vida. O presente projeto visa apresentar uma solução portátil e de baixo custo que amenize ou elimine (temporariamente) os efeitos dos sintomas de deficientes visuais com baixa visão, usando tecnologia OpenCV para Android com foco em melhoria da experiência de leitura. Será desenvolvido um aplicativo para dispositivos portáteis com transformações de vídeo, zoom ampliado e emparelhamento bluetooth com outros dispositivos (Notebook e Desktop), realizando-se, inicialmente, testes de simulação com smartphones e, posteriormente, com outros dispositivos portáteis. Ao final desse projeto, espera-se contribuir para a inclusão de deficientes visuais no âmbito acadêmico.

Inclusão de Suporte a Metadados a Uma Ferramenta de Suporte Formal ao Desenvolvimento Baseado em Componentes

Autores: Dalay Israel de Almeida Pereira e Marcel Vinícius Medeiros Oliveira

Resumo: O desenvolvimento de sistemas baseados em componentes foi uma evolução importante no processo de desenvolvimento de software. Com o uso dessa abordagem a manutenção dos sistemas foi facilitada, trazendo mais confiabilidade e reutilização. A composição dos componentes (e suas interações), entretanto, ainda constitui a principal fonte de problemas e requer uma análise detalhada. Os métodos formais são uma ferramenta precisa de especificação de sistemas que se utilizam de uma forte base matemática, trazendo, entre outros benefícios, mais segurança. O método formal CSP possibilita a especificação de sistemas concorrentes e a verificação de propriedades inerentes a tais sistemas, bem como o refinamento entre diferentes modelos. Uma abordagem de desenvolvimento de sistemas confiáveis baseados em componentes, BRIC, foi desenvolvida por Rodrigo Ramos em 2011. Esta abordagem utiliza CSP para especificar as restrições das interações entre os componentes de forma a permitir uma verificação formal do comportamento da composição deles. Para auxiliar esta abordagem, foi desenvolvida uma ferramenta, BTS (BRIC-Tool-Support), por Sarah Rocha em 2012, que automatiza a verificação da composição dos componentes, onde o conjunto de verificações CSP impostas é gerado e verificado automaticamente de maneira transparente ao usuário. Neste trabalho, nós apresentamos uma extensão a esta ferramenta que faz com que BTS ofereça suporte a uma otimização de BRIC, denominada BRICK, que enriquece os componentes com metadados que contém informações adicionais úteis na verificação da composição. Esses metadados da composição são derivados dos metadados dos componentes que a constitui.

Uma implementação verificada de abstração CSP

Autores: Francisco José Silva Macário e Marcel Vinícius Medeiros Oliveira

Resumo: Ao longo do desenvolvimento de sistemas concorrentes, a complexidade de desenvolvê-los pode facilmente crescer exponencialmente, dando origem a um processo muito complexo e propenso a erro. Ao fazer o uso de linguagens formais como CSP, é possível simplificar a tarefa de desenvolver sistemas complexos, aumentando o nível de confiabilidade no sistema resultante. Infelizmente, as linguagens formais não são executáveis, sendo necessário resolver o distanciamento entre a especificação formal e um programa executável. Em trabalhos anteriores, foi apresentado a ferramenta CSP2HC, ao qual traduz consideráveis subconjuntos de CSP para código fonte em Handel-C, que pode ser convertido para arquivos que rodam em FPGAs. Este subconjunto restringe o uso de estruturas de dados e hiding de CSP. Neste trabalho, será apresentado uma extensão do CSP2HC ao qual foi introduzido sequências do conjunto estruturas de dados aceitáveis e o completo tratamento do operador hiding de CSP. E finalmente é realizado uma validação da extensão, aplicando a tradução de uma especificação CSPm que possui hiding.

Organização e Realização