IEEE
Computational Intelligence
MAGAZINE

MAY 2012
VOLUME 7 NUMBER 2
WWW.IEEE-CIS.ORG

16 Evolutionary and Swarm Computing for the Semantic Web

32 Large-Scale Storage and Reasoning for Semantic Data Using Swarms

45 Exploiting Tractable Fuzzy and Crisp Reasoning in Ontology Applications

54 Reasoning with Large Scale Ontologies in Fuzzy pD* Using MapReduce

IEEE Computational Intelligence Society

IEEE

This is one of the two flagship biennial international meetings sponsored by the IEEE Computational Intelligence Society promoting all aspects of computational intelligence. The IEEE SSCI 2013 co-locates several technical meetings at one location. This event attracts top researchers, professionals, and students from around the world. Regular registration includes attendance of any session of any technical meeting, complete set of the proceedings of all meetings, lunches, coffee breaks and the banquet. The IEEE SSCI meeting features **a large number of keynote, tutorial, panel and special sessions** all of which are open to all participants. Each meeting will consider awarding **best student paper** and **best overall paper awards**. The IEEE SSCI 2013 will also offer a number of **travel grants for students** and **travel grants for researchers from developing countries**. The conference proceedings of the IEEE SSCI have always been included in the IEEE Xplore and indexed by all other important databases.

Organizing Committee

General Chair:
P N Suganthan, Nanyang Technological University, Singapore
General Program Chair: David B Fogel, Natural Selection Inc., USA
Finance Chair: Robert Kozma, The University of Memphis, USA
Keynote-Tutorial Chair: Swagatam Das, Jadavpur University, India
Local Arrangement Chairs:
Meng-Hiot Lim, Nanyang Technological University, Singapore
Yew Soon Ong, Nanyang Technological University, Singapore
Publication Chair:
Suresh Sundaram, Nanyang Technological University, Singapore

Advisory Committee

Piero P Bonissone, USA
Bernadette Bouchon-Meunier, France
Kalyanmoy Deb, India
Garry Greenwood, USA
Chin-Teng (CT) Lin, Taiwan
Derong Liu, China
Simon M Lucas, UK
Sushmita Mitra, India
Nikhil R Pal, India
Vincenzo Piuri, Italy
Marios M Polycarpou, Cyprus
Xin Yao, UK
Gary Yen, USA

Publicity Committee

Ashish Anand, India
S Baskar, India
Uday K Chakraborty, USA
Tomasz Cholewo, USA
Sung-Bae Cho, Korea
Leandro d S Coelho, Brazil
Andries P Engelbrecht, South Africa
Haibo He, USA
Francisco Herrera, Spain
Amir Hussain, UK
Hisao Ishibuchi, Japan
Xiaodong Li, Australia
Jane J Liang, China
Bijaya K Panigrahi, India
Millie Pant, India
Mike Preuss, Germany
Alex K Qin, France
Suresh C Satapathy, India
Dipti Srinivasan, Singapore
Fuchun Sun, China
Ke Tang, China
Fatih M Tasgetiren, Turkey
Huaguang Zhang, China

Important Dates

Paper submission: 10 Oct 2012
Decision: 05 Jan 2013
Final submission: 05 Feb 2013
Early Registration: 05 Feb 2013

Symposia and Workshops

ADPRL 2013, IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, Marco Wiering, The Netherlands; Jagannathan Sarangapani, USA; Huaguang Zhang, China.

ALIFE 2013, IEEE Symposium on Artificial Life, Hussein Abbass, Australia.

CCMB 2013, IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain, Leonid Pervlovsky, USA; Damien Coyle, UK; Kai-Keng Aung, Singapore; Jose F Fontanari, Brazil.

CIASG 2013, IEEE Symposium on Computational Intelligence Applications in Smart Grid, Ganesh K Venayagamoorthy, USA; Jung-Wook Park, Korea; Haibo He, USA.

CIBCB 2013, IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, Jung-Hsien Chiang, Taiwan; Pau-Choo (Julia) Chung, Taiwan.

CIBIM 2013, IEEE Symposium on Computational Intelligence in Biometrics and Identity Management, David Zhang, Hong Kong; Qinghan Xiao, Canada; Fabio Scotti, Italy.

CICA 2013, IEEE Symposium on Computational Intelligence in Control and Automation, Xiao-Jun Zeng, UK.

CIComms 2013, IEEE Symposium on Computational Intelligence for Communication Systems, Maode Ma, Singapore; Paolo Rocca, Italy; Sasitharan Balasubramaniam, Ireland.

CICS 2013, IEEE Symposium on Computational Intelligence in Cyber Security, Dipankar Dasgupta, USA; Justin Zhan, USA; Kumaraguru Ponnurangam, India.

CIDM 2013, IEEE Symposium on Computational Intelligence and Data Mining, Barbara Hammer, Germany; Zhi-Hua Zhou, China; Lipo Wang, Singapore; Nitesh Chawla, USA.

CIDUE 2013, IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, Yaochu Jin, UK; S Yang, UK; Robi Polikar, USA.

CIES 2013, IEEE Symposium on Computational Intelligence for Engineering Solutions, Michael Beer, UK; Vladik Kreinovich, USA; Rudolf Kruse, Germany.

CIEL 2013, IEEE Symposium on Computational Intelligence and Ensemble Learning, Nikhil R Pal, India; Xin Yao, UK; P N Suganthan, Singapore.

CIFEr 2013, IEEE Symposium on Computational Intelligence for Financial Engineering and Economics, Robert Golan, USA; Han La Poutre, The Netherlands; Ronald R. Yager, USA; Shu-Heng Chen, Taiwan.

CIfIoT 2013, IEEE International Workshop on Computational Intelligence for the Internet of Things, Vincenzo Piuri, Italy.

CII 2013, IEEE Symposium on Computational Intelligence in Industry, Piero Bonissone, USA; Keeley Crockett, UK; Cristian Figueroa, Chile; Emilio Corchado, Spain.

CIMI 2013, IEEE International Workshop on Computational Intelligence in Medical Imaging, Gerald Schaefer, UK; Sergio Damas, Spain.

CIMSVP 2013, IEEE Symposium on Computational Intelligence for Multimedia Signal and Vision Processing, K M Iftekharuddin, USA; S Bouzerdoum, Australia.

CIPLS 2013, IEEE Symposium on Computational Intelligence in Production and Logistics Systems, Bülent Çatay, Turkey; Raymond Chiong, Australia; Patrick Siarry, France.

OC 2013, IEEE Workshop on Organic Computing, Rolf Würtz, Germany.

CIRAT 2013, IEEE Symposium on Computational Intelligence in Rehabilitation and Assistive Technologies, Guilherme N DeSouza,USA; Shuzhi Sam Ge, Singapore.

CISched 2013, IEEE Symposium on Computational Intelligence in Scheduling, Rong Qu, UK; Ling Wang, China; Quanke Pan, China.

CISDA 2013, IEEE Symposium on Computational Intelligence for Security and Defense Applications, Akira Namatame, Japan; Nur Zincir-Heywood, Canada; Rami Abielmona, Canada.

CIVI 2013, IEEE Workshop on Computational Intelligence for Visual Intelligence, Guilherme N DeSouza, USA; Yuanqiang (Evan) Dong, USA.

CIVTS 2013, IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems, Dimitar Filev, USA; Danil Prokhorov, USA.

CompSens 2013, IEEE Workshop on Merging Fields of Computational Intelligence and Sensor Technology, Ivo Bukovsky, Czech Republic; Torsten Wagner, Japan

EAIS 2013, IEEE Symposium on Evolving Adapting Intelligent Systems, Plamen Angelov, UK; Dimitar Filev, USA; Nikola Kasabov, New Zealand.

FOCI 2013, IEEE Symposium on Foundations of computational Intelligence, Manuel Ojeda-Aciego, Spain; Carlos Cotta, Spain.

GEFS 2013, IEEE International Workshop on Genetic and Evolutionary Fuzzy Systems, Rafael Alcala, Spain; Yusuke Nojima, Japan.

HIMA 2013, IEEE Workshop on Hybrid Intelligent Models and Applications, Patricia Melin, Mexico; Sanghamitra Bandyopadhyay, India.

IA 2013, IEEE Symposium on Intelligent Agents, Hani Hagras, UK; Vincenzo Loia, Italy.

ICES 2013, IEEE International Conference on Evolvable Systems – From Biology to Hardware, Andy M Tyrrell, UK; Pauline C Haddow, Norway.

MC 2013, IEEE Symposium on Memetic Computing, Zexuan Zhu, China; Maoguo Gong, China; Ji Zhen, China; Yew-Soon Ong, Singapore.

MCDM 2013, IEEE Symposium on Multicriteria Decision-Making , Carlos A Coello Coello, Mexico; Piero Bonissone, USA; Yaochu Jin, UK.

QCCI 2013, IEEE Symposium on Quantum Computing and Computational Intelligence, William N N Hung, USA; Swagatam Das, India; Marek Perkowski, USA.

RiiSS 2013, IEEE Workshop on Robotic Intelligence in Informationally Structured Space, Honghai Liu, UK; Naoyuki Kubota, Japan.

SDE 2013, IEEE Symposium on Differential Evolution, Janez Brest, Slovenia; Swagatam Das, India; Ferrante Neri, Finland; P N Suganthan, Singapore.

SIS 2013, IEEE Swarm Intelligence Symposium, Yuhui Shi, P. R. China.

T2FUZZ 2013, IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems, Hani Hagras, UK; Jerry Mendel, USA; Bob John, UK.

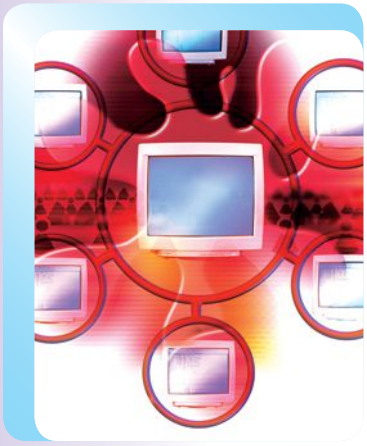
WACI 2013, Workshop on Affective Computational Intelligence, Marie-Jeanne Lesot, France; Maria Rifqi, France.

http://www.ieee-**ssci**.org/

Digital Object Identifier 10.1109/MCI.2012.2188599

IEEE Computational Intelligence MAGAZINE

Volume 7 Number 2 ❑ May 2012
www.ieee-cis.org



on the cover
©IMAGE 100

Features

16 **Evolutionary and Swarm Computing for the Semantic Web**
by Christophe Guéret, Stefan Schlobach, Kathrin Dentler, Martijn Schut, and Gusz Eiben

32 **Large-Scale Storage and Reasoning for Semantic Data Using Swarms**
by Hannes Mühleisen and Kathrin Dentler

45 **Exploiting Tractable Fuzzy and Crisp Reasoning in Ontology Applications**
by Jeff Z. Pan, Edward Thomas, Yuan Ren, and Stuart Taylor

54 **Reasoning with Large Scale Ontologies in Fuzzy *pD** Using MapReduce**
by Chang Liu, Guilin Qi, Haofen Wang, and Yong Yu

Columns

67 Review Article
Semantic Web Meets Computational Intelligence: State of the Art and Perspectives
by Huajun Chen, Zhaohui Wu, and Philippe Cudré-Mauroux

75 Book Review
by Piotr Lipinski

Departments

2 Editor's Remarks
by Kay Chen Tan

3 President's Message
by Marios M. Polycarpou

4 Society Briefs
Newly Elected CIS Administrative Committee Members (2012–2014)
by Marios M. Polycarpou

IEEE Fellows—Class of 2012
by Piero P. Bonissone

11 Publication Spotlight
by Derong Liu, Chin-Teng Lin, Garry Greenwood, Simon Lucas, and Zhengyou Zhang

14 Guest Editorial
Special Issue on Semantic Web Meets Computational Intelligence
by Huajun Chen and Philippe Cudré-Mauroux

78 Conference Calendar
by Gary B. Fogel

IEEE Computational Intelligence Magazine (ISSN 1556-603X) is published quarterly by The Institute of Electrical and Electronics Engineers, Inc. **Headquarters:** 3 Park Avenue, 17th Floor, New York, NY 10016-5997, U.S.A. +1 212 419 7900. Responsibility for the contents rests upon the authors and not upon the IEEE, the Society, or its members. The magazine is a membership benefit of the IEEE Computational Intelligence Society, and subscriptions are included in Society fee. Replacement copies for members are available for $20 (one copy only). Nonmembers can purchase individual copies for $119.00. Nonmember subscription prices are available on request. Periodicals postage paid at New York, NY and at additional mailing offices. Postmaster: Send address changes to IEEE Computational Intelligence Magazine, IEEE, 445 Hoes Lane, Piscataway, NJ 08854-1331 U.S.A. PRINTED IN U.S.A. Canadian GST #125634188.

Digital Object Identifier 10.1109/MCI.2012.2191837

Kay Chen Tan
National University of Singapore, SINGAPORE

Editor's Remarks

Empowering Semantic Web with Computational Intelligence

After the New Year issue, I am now closing this issue before the Lunar New Year in Asia. Like previous years, I had to make reservations for air tickets and plan for my family holidays. Most of the time, I have to check against my kids' co-curricular activities, my wife's work schedule and my own appointments before going online to book the tickets.

How nice if I could simply activate a programme on my mobile device that can automatically synchronize with my wife and children's calendars, remind me to reschedule any important appointments that I might have overlooked, book the air tickets with our preferred seats and perhaps even update my accounting system and calendars automatically. This example sounds simple but it remains a long-term vision of intelligent Semantic Web, which essentially is the topic of this issue.

While the World Wide Web has largely changed our way of lives, the next phase of web evolution can be described by Tim Berners-Lee's vision of the Semantic Web, which is "a web of data that can be processed directly and indirectly by machines." In this issue, we showcase several successful deployments of computational intelligence (CI) technologies into Semantic Web applications, as guest-edited by Huajun Chen and Philippe Cudré-Mauroux.

In the "Society Briefs" column of this issue, we congratulate the new IEEE Fellows in the Class of 2012 elevated through CIS, and welcome our five newly elected AdCom members who will assist in governing and administering of CIS.

(continued on page 10)

KC Tan and fellow CIS members at the Winter School on Advances in Computational Intelligence: Algorithms and Applications in Bhubaneswar, India.

Digital Object Identifier 10.1109/MCI.2012.2188559
Date of publication: 13 April 2012

President's Message

Marios M. Polycarpou
University of Cyprus, CYPRUS

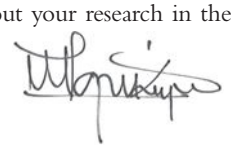
Data Flood

Recently, I have been thinking about data—there is so much of it. Supposedly, everyday we create more than 2 quintillion bytes of data (yes, that's 18 zeros after the one!). The amount of data generated is increasing by 50 percent a year. Imagine the amount of data that will be generated in 5 years, or 10 or 20 years. You may be wondering, where is all this data coming from? Well, it comes from everywhere. Starting from traditional sources, there are sensors that measure temperature, humidity, movement; there are more advanced sensors that measure chemical/biological changes, heart rate, vibrations, etc. There are data coming from records of online purchases, video data, audio, click streams, etc. Moreover, every day new sensors are being developed in almost all application domains for generating even more data. We are moving towards the so-called "Internet of Things", where sensors and actuators are embedded in physical systems (e.g., appliances, mobile phones, automobiles) and connected by networks to computing devices.

What does one do with so much data? A few years ago there was the issue of data storage. Some of the hottest companies of the time dealt with developing technologies for fast data storage/access. Nowadays, data storage has become extremely cheap. According to Kevin Kelly of *Wired* magazine, for less than $600, we can purchase a disk drive with the capacity to store all of the world's music. With cloud computing technology, data access will become even cheaper and more convenient. The big challenge in the years ahead is to be able to process and make sense out of the available data, and ultimately make the generated knowledge easily available to authorized users. The objective is to be able to find correlations and patterns in time and space, to develop models in real time and to extract useful knowledge that can be used, for example, to improve productivity, to enhance safety or to create more energy efficient environments.

Computational intelligence is at the heart of these technological developments. The data analysis tools that are being developed by researchers in our community will be crucial in the handling of the big data surge. Computational intelligence methods such as real-time learning, evolutionary computation, data mining, neural networks, fuzzy systems, data visualization can be applied in many applications related to real time data analysis. According to a report by the McKinsey Global Institute, there is a significant shortage of people with deep analytical skills for handling big data and making decisions based on their findings. I believe that the Computational Intelligence research and development community is well positioned to be a key player in the upcoming data revolution.

I look forward to seeing many of you in Brisbane, Australia at our flagship conference, the IEEE World Congress on Computational Intelligence (WCCI 2012). I also look forward to your presentations and to learning more about your research in the theory and applications for intelligent data processing.



Digital Object Identifier 10.1109/MCI.2012.2188560
Date of publication: 13 April 2012

Digital Object Identifier 10.1109/MCI.2012.2192189

Marios M. Polycarpou
University of Cyprus, CYPRUS

Newly Elected CIS Administrative Committee Members (2012–2014)

Cesare Alippi, Politecnico di Milano, ITALY

Cesare Alippi received the Dr. Ing. degree in electronic engineering (summa cum laude) in 1990 and the Ph.D. degree in computer engineering in 1995, both from Politecnico di Milano, Milan, Italy. He has been a visiting researcher at the University College London, London, U.K., the Massachusetts Institute of Technology, Cambridge, USA and the École Supérieure de Physique et de Chimie Industrielles, Paris, France.

Alippi acted as research scientist at the Italian National Research Council (1996–98) then Reader and Associate Professor at Politecnico di Milano (1998–2002). Since 2002, he has been a Full Professor in Information Processing Systems at the same institution.

Alippi is a Fellow of the IEEE, Associate editor of *IEEE Computational Intelligence Magazine*, Associate Editor of the *IEEE Transactions on Neural Networks* (2005–11), Associate Editor of the *IEEE Transactions on Instrumentation and Measurements* (2003–09), Chair of the IEEE Neural Networks Technical Committee of the IEEE Computational Intelligence Society (2008–10), Chair of the IEEE Computational Intelligence Society Research Grant Sub-committee (2011) and member and Chair of other IEEE committees including the IEEE Rosenblatt Award Committee.

In 2004 he received the IEEE Instrumentation and Measurement Society Young Engineer Award; in 2011, he was awarded Knight of the Order of Merit of the Italian Republic. Current research activity addresses adaptation and learning in non-stationary environments and intelligent embedded systems. Research is also carried out on the industrial front with several well-known companies.

Cesare has organized several special issues and sessions in international journals and conferences, given keynote and plenary talks and has been program (co) chair and general chair in several IEEE conferences (recently, IJCNN11, IEEE CIMSA 09, IEEE ROSE 09, IEEE HAVE 09). Cesare holds 5 patents and has published more than 150 papers in international journals and conference proceedings.

Currently, Cesare is the General chair of the IEEE INNS International Joint Conference on Neural Networks 2012, June 10–15, 2012, Brisbane, Australia, and guest editor of the special issue on "Learning in Nonstationary and Evolving Environments" to be published in *IEEE Transactions on Neural Networks and Learning Systems* in 2013.

Pau-Choo (Julia) Chung, National Cheng Kung University, TAIWAN

Pau-Choo (Julia) Chung received the Ph.D. degree in electrical engineering from Texas Tech University, USA, in 1991.

She then joined the Department of Electrical Engineering, National Cheng Kung University (NCKU), Taiwan, and has become a full professor since 1996. She served as the Vice Director (2001–2005), then the Director (2005–2008) of the Center for Research of E-life Digital Technology, the Director of Electrical Laboratory (2005–2008), and the Director of Institute of Computer and Communication Engineering (2008–2011), NCKU. She was selected as Distinguished Professor of NCKU, 2005. She currently is serving as the Chair of Department of Electrical Engineering, NCKU.

Dr. Chung's research interests include medical image analysis, video analysis, pattern recognition, computational intelligence, biosignal analysis, neural networks, and computer vision. Particularly she applies most of her research results on healthcare and medical applications. She received many awards, such as the annual best paper award in Chinese Journal Radiology 2001, the best paper awards from World Multiconference on Systemics, Cybernetics, and Informatics (SCI) 2001 and International Computer Symposium (ICS) 1998, Acer's Best Research Award in 1994 and 1995, the best paper awards from the Conference of Computer Vision, Graphics, and Image Processing (CVGIP), in 1993, 1996, 1997, 1999, and 2001, Best Research Young Innovator Award of National Science Council, Taiwan, in 1999. Dr. Chung has served as the

Digital Object Identifier 10.1109/MCI.2012.2188561
Date of publication: 13 April 2012

Program Co-Chair, the Publicity Co-Chair, the Special Session Co-Chair, and program committee member in many international conferences. Dr. Chung was the Chair of IEEE Computational Intelligence Society (CIS) (2004–2005), Tainan Chapter. She is the Chair of the IEEE Life Science Systems and Applications Technical Committee, and a member of the BioCAS Technical Committee in the CAS Society. She served as the Editor of Journal of Information Science and Engineering, the Guest Editor of *IEEE Transactions on Circuits and Systems–I,* and the Secretary General of Biomedical Engineering Society of the Republic of China. Currently she is the Associate Editor of *IEEE Transactions on Neural Networks and Learning Systems*, the *Multidimensional Systems and Signal Processing*, and *Soft Computing*. She is one of the co-founders of Medical Image Standard Association (MISA) at Taiwan and currently is in the Board of Directors of MISA, and BoG of Image Processing and Pattern Recognition (IPPR) Association, Taiwan.

Pau-Choo Chung currently is serving the second term in BoG of CAS Society (2007–2009, 2009–2012) and the ADCOM member of IEEE CIS. She also served as one IEEE Distinguished Lecturer (2005–2007) in CASS. She is in Member of Phi Tau Phi honor society and has been an IEEE Fellow since 2008.

Yaochu Jin, University of Surrey, UK

Yaochu Jin received the B.Sc., M.Sc., and Ph.D. degrees all in Automatic Control from Zhejiang University, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree in Computer Engineering from Ruhr University Bochum, Germany, in 2001.

He is currently Professor of Computational Intelligence and Head of the Nature Inspired Computing and Engineering (NICE) Group, Department of Computing, University of Surrey, UK. Before joining Surrey, he was a Principal Scientist and Project Leader with the Honda Research Institute Europe in Germany. His research interests range from computational intelligence to computational neuroscience and computational systems biology. He has authored one monograph, (co)-edited books and conference proceedings and is the author of over 150 peer-reviewed journal and conference papers.

He is presently Chair of the Intelligent Systems Applications Technical Committee and was the founding Chair of the Evolutionary Computation in Dynamic and Uncertain Environments Task Force. He is General Chair of 2013 UK Workshop on Computational Intelligence, Program Chair of 2013 IEEE Congress on Evolutionary Computation (CEC 2013), and General Chair of the 2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. He was Co-Chair of 2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, Co-Chair of the 2007, 2009 and 2011 IEEE Symposium on Multi-Criterion Decision-Making, Program Chair of the 2005 International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'2005). He is the Area Co-Chair of FUZZ-IEEE 2011 and Tutorial Chair of CEC 2007 and CEC 2010.

Dr. Jin is an Associate Editor of *BioSystems*, *IEEE Transactions on Neural Networks*, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *IEEE Transactions on NanoBioscience*, and *IEEE Computational Intelligence Magazine*. He is an Area Editor of the *Soft Computing Journal*. He has served as a guest editor of nine international journal special issues, including six IEEE transactions. He is an Invited Plenary/Keynote Speaker of several international conferences on various topics, including modeling and analysis of gene regulatory networks, multi-objective machine learning, computational modeling of neural development, morphogenetic robotics and evolutionary aerodynamic design optimization.

He is a Fellow of British Computer Society and Senior Member of the IEEE.

James M. Keller, University of Missouri-Columbia, USA

James M. Keller received the Ph.D. in Mathematics in 1978. He holds the University of Missouri Curators' Professorship in the Electrical and Computer Engineering and Computer Science Departments on the Columbia campus. He is also the R. L. Tatum Professor in the College of Engineering. His research interests center on computational intelligence: fuzzy set theory and fuzzy logic, neural networks, and evolutionary computation with a focus on problems in computer vision, pattern recognition, and information fusion including bioinformatics, spatial reasoning in robotics, geospatial intelligence, sensor and information analysis in technology for eldercare, and landmine detection. His industrial and government funding sources include the Electronics and Space Corporation, Union Electric, Geo-Centers, National Science Foundation, the Administration on Aging, The National Institutes of Health, NASA/JSC, the Air Force Office of Scientific Research, the Army Research Office, the Office of Naval Research, the National Geospatial Intelligence Agency, the Leonard Wood Institute, and the Army Night Vision and Electronic Sensors Directorate. Professor Keller has coauthored over 350 technical publications.

Jim is a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) and the International Fuzzy Systems Association (IFSA), and a past President of the North American Fuzzy Information Processing Society (NAFIPS). He received the 2007 Fuzzy Systems Pioneer Award and the 2010 Meritorious Service Award from the IEEE Computational Intelligence Society. He finished a full six year term as

Editor-in-Chief of the *IEEE Transactions on Fuzzy Systems*, followed by being the Vice President for Publications of the IEEE Computational Intelligence Society from 2005–2008, and since then an elected CIS Adcom member. He is the IEEE TAB Transactions Chair and a member of the IEEE Publication Review and Advisory Committee. Among many conference duties over the years, Jim was the general chair of the 1991 NAFIPS Workshop and the 2003 IEEE International Conference on Fuzzy Systems.

Jacek M. Zurada, University of Louisville, USA

Jacek M. Zurada serves as a University Scholar and Professor with the Electrical and Computer Engineering Department at the University of Louisville, Kentucky. He was Department Chair from 2004 to 2006. He has published 360 journal and conference papers in the areas of neural networks, computational intelligence, data mining, image processing and VLSI circuits. He also has authored or co-authored three books and co-edited a number of volumes in Springer *Lecture Notes on Computer Science*.

He has held visiting appointments at Princeton, Northeastern, Auburn, and overseas in Australia, Chile, China, France, Germany, Hong Kong, Italy, Japan, Poland, Singapore, Spain, South Africa and Taiwan. Dr. Zurada was an Associate Editor of *IEEE Transactions on Circuits and Systems, Pt. I and Pt. II*, and served on the Editorial Board of the *Proceedings of IEEE*. From 1998 to 2003 he was the Editor-in-Chief of *IEEE Transactions on Neural Networks*. He is an Associate Editor of *Neurocomputing, Schedae Informaticae, International Journal of Applied Mathematics and Computer Science, Advisory Editor of Int'l Journal of Information Technology and Intelligent Computing*, and Editor of Springer Natural Computing Book Series.

He has served the profession and the IEEE in various elected capacities, including as President of IEEE CIS in 2004–05 and the ADCOM member in 2009–11 and earlier years. He has been member and Chair of various IEEE CIS and IEEE TAB committees, and is also now serving as Vice-Chair of PSPB and PSPB Strategic Planning Committee (2010–11). He is also Chair of the IEEE TAB Periodicals Committee (2010–11) and will chair the IEEE TAB Periodicals Review and Advisory Committee in 2012–13. He is also on the 2012 ballot of IEEE-wide elections for 2013 VP-TAB Elect.

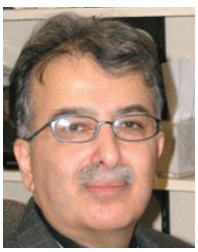
Dr. Zurada has received a number of awards for distinction in research, teaching, and service including the 1993 Presidential Award for Research, Scholarship and Creative Activity, 1999 IEEE Circuits and Systems Society Golden Jubilee Medal, and the 2001 Presidential Distinguished Service Award for Service to the Profession. He is a Distinguished Speaker of IEEE CIS. In 2003 he was conferred the Title of National Professor by the President of Poland. In 2004, 2006 and 2010, he received three Honorary Professorships from Chinese universities. Since 2005 he has been a Member of the Polish Academy of Sciences, and has been appointed as a Senior Fulbright Specialist for 2006–12.

IEEE Fellows—Class of 2012

Piero P. Bonissone
General Electric Global Research, USA

Hojjat Adeli, The Ohio State University, Columbus, OH, USA

for contributions to computational intelligence in infrastructure engineering



Hojjat Adeli is Abba G. Lichtenstein Professor of Civil Engineering, and via courtesy appointment Professor of Biomedical Engineering, Biomedical Informatics, Electrical and Computer Engineering, Neuroscience, and Neurological Surgery at The Ohio State University. He has authored over 500 research and scientific publications in various fields of computer science, engineering, applied mathematics, and medicine since 1976 when he received his Ph.D. from Stanford University at the age of 26. His research has been published in 83 different journals including *IEEE Transactions on Neural Networks*, and *IEEE Transactions on Biomedical Engineering*. He has authored fourteen books including *Machine Learning—Neural Networks, Genetic Algorithms, and Fuzzy Systems*, Wiley, 1995; and *Wavelets in Intelligent Transportation Systems*, Wiley, 2005; and *Automated EEG-based Diagnosis of Neurological Disorders—Inventing the Future of Neurology*, CRC Press, 2010. He has also edited 23 books such as *Knowledge Engineering-Volume One-Fundamentals*, McGraw-Hill, 1990, Adeli, H., Ed., *Knowledge Engineering—Volume Two—Applications*, McGraw-Hill, 1990, and *Supercomputing*

Digital Object Identifier 10.1109/MCI.2012.2188563
Date of publication: 13 April 2012

in Engineering Analysis, Marcel Dekker, 1992. He is the Founder and Editor-in-Chief of the international research journals *Computer-Aided Civil and Infrastructure Engineering*, now in 27th year of publication and *Integrated Computer-Aided Engineering*, now in 20th year of publication He is also the Editor-in-Chief of *International Journal of Neural Systems*. He is the quadruple winner of The Ohio State University Lumley Outstanding Research Award. In 1998 he received The Ohio State University's highest research honor, the Distinguished Scholar Award "in recognition of extraordinary accomplishment in research and scholarship". In 2005 he was elected Distinguished Member of American Society of Civil Engineers "for wide-ranging, exceptional, and pioneering contributions to computing in civil engineering and extraordinary leadership in advancing the use of computing and information technologies in many engineering disciplines throughout the world." In 2007, he received The Ohio State University Peter L. and Clara M. Scott Award for Excellence in Engineering Education, "for sustained, exceptional, and multi-faceted contributions to numerous fields including computer-aided engineering, knowledge engineering, computational intelligence, large-scale design optimization, and smart structures with worldwide impact," and Charles E. MacQuigg Outstanding Teaching Award. He is a Fellow the American Association for the Advancement of Science. He has been a Keynote Lecturer at 83 research conferences held in 43 different countries. In 2010 Wiley-Blackwell established the Hojjat Adeli Award for Innovation in Computing to be bestowed annually with a cash prize of $1,000. The same year he was featured as an Engineering Legend in ASCE journal of *Leadership and Management in Engineering*. In 2011 World Scientific established the Hojjat Adeli Award for Outstanding Contribution in Neural Systems given annually with a cash prize of $500. He is also featured in Buckeye Wall of Brilliance at The Ohio State University, *"*a permanent exhibition honoring the achievements of individuals who have passed through its halls.*"*

Pierre Baldi, University of California, CA, USA

for contributions to machine learning and its applications in the life sciences

Pierre Baldi received his PhD from the California Institute of Technology in 1986. He was a postdoctoral fellow at the University of California, San Diego from 1986 to 1988, and a member of the technical staff at the Jet Propulsion Laboratory from 1988–1994. He was the founder and CEO of a start-up company from 1995 to 1999 and joined the University of California, Irvine in 1999. At UCI, he is currently Chancellor's Professor in the School of Information and Computer Sciences and the Department of Biological Chemistry and the Director of the Institute for Genomics and Bioinformatics.

Baldi's research work is at the interface of the computational and life sciences, in particular the application of artificial intelligence and statistical machine learning methods to problems in chemoinformatics, genomics, proteomics, and systems biology. He is credited with pioneering the use of Hidden Markov Models (HMMs), graphical models, and recursive neural networks in bioinformatics. Dr. Baldi has published over 250 peer-reviewed research articles and four books: *Modeling the Internet and the Web–Probabilistic Methods and Algorithms*, Wiley, (2003); *DNA Microarrays and Gene Regulation–From Experiments to Data Analysis and Modeling*, Cambridge University Press, (2002); *The Shattered Self–The End of Evolution*, MIT Press, (2001); *Bioinformatics: the Machine Learning Approach*, MIT Press, Second Edition (2001).

His group has developed widely used databases, software, and web servers including the ChemDB database and chemoinformatics portal for the prediction of molecular properties and applications in chemical synthesis and drug discovery, the SCRATCH suite of protein feature predictors, the Cyber-T program for the differential analysis of gene expression data using Bayesian statistical methods, and the MotifMap system for charting transcription factor binding sites on a genome-wide scale and for supporting gene regulatory mechanisms inferences in healthy and disease system. He is the recipient of the 1993 Lew Allen Award, the 1999 Laurel Wilkening Faculty Innovation Award, a 2006 Microsoft Research Award, and the 2010 E. R. Caianiello Prize for research in machine learning. He is also a Fellow of the Association for the Advancement of Science (AAAS) and the Association for the Advancement of Artificial Intelligence (AAAI).

Kalyanmoy Deb, Indian Institute of Technology-Kanpur, INDIA

for contributions to evolutionary multi-criterion optimization techniques

Kalyanmoy Deb is a Professor of Mechanical Engineering at Indian Institute of Technology Kanpur in India. He is also an Adjunct Professor at the Business Technology Department in the Aalto University in Finland and a visiting professor at University of Skovde, Sweden. Prof. Deb received his Bachelor's degree in Mechanical Engineering from Indian Institute of Technology Kharagpur in India in 1985. After a short industrial experience of two years, he completed his master's and PhD degrees in Engineering Mechanics from University of Alabama, Tuscaloosa, USA in 1989 and 1991, respectively.

Prof Deb's main research interests are in evolutionary optimization algorithms and their application in optimization and machine learning. He is largely known for his seminal research in developing and applying Evolutionary Multi-Criterion Optimization (EMO). He has led and has been very active in getting the EMO and MCDM fields together through joint conference organizations, book writing, executing collaborative

research projects. For his pioneering research in multi-criterion optimization and decision making, he was awarded the IEEE Fellow, Infosys Prize in Engineering and Computer Science, CajAstur Mamdani Prize in Soft Computing, JC Bose fellowship, and 'Edgeworth-Pareto' award. He received the prestigious Shanti Swarup Bhatnagar Prize in Engineering Sciences for the year 2005. He has also received the 'Thomson Citation Laureate Award' from Thompson Scientific for having the highest number of citations in Computer Science during 1996–2005 in India. His 2002 IEEE-TEC NSGA-II paper is now judged as the Most Highly Cited paper and a Current Classic by Thomson Reuters having more than 2,900 citations. He is a fellow of Indian National Science Academy (INSA), Indian National Academy of Engineering (INAE), Indian Academy of Sciences (IASc), and International Society of Genetic and Evolutionary Computation (ISGEC). He has received Friedrich Wilhelm Bessel Research award from Alexander von Humboldt Foundation in 2003. He has written two text books on optimization, 11 edited books, and more than 300 international journal and conference research papers. He is associate editor and in the editorial board on 18 major international journals. More information about his research can be found from http://www.iitk.ac.in/kangal/deb.htm.

Gerard Dreyfus, ESPCI-PARISTECH, FRANCE

for contributions to machine learning and its applications

Gérard Dreyfus is a professor of electronics, automatic control and machine learning, head of the SIGnal processing and MAchine learning (SIGMA) lab of ESPCI ParisTech. He joined ESPCI ParisTech as an assistant professor in computer science after receiving his PhD in physics from Université Pierre et Marie Curie (Paris). After 10 years as a researcher in physics, he became involved in statistical machine learning and created the SIGMA lab (formerly Electronics lab). As early as 1985, he published pioneering articles on neural networks (mainly in physics journals), which are still being cited in the current literature. At the beginning of the 1990s, he was among the few machine learning researchers who first realized the strong interrelation between machine learning and adaptive filtering algorithms. In 2001, he devised an innovative methodology that allows the model designer to integrate prior knowledge, in the form of differential or partial differential equations, into a machine-learning model. In the same methodological vein, Gérard Dreyfus and his group devised innovative methods for model and variable selection. These contributions were successfully applied in industry, in the framework of research contracts in many different areas of technology.

Gérard Dreyfus authored or co-authored more than 200 publications in international journals and conferences, and three books on machine learning and neural networks. He holds some 20 international patents. He co-founded two companies, and has been acting as scientific advisor of several French companies; two of his PhD students created successful companies, active in machine learning in France and in the USA.

Since 2003, Gérard Dreyfus has oriented his research towards designing new machine learning approaches in areas related to medicine (computer-aided diagnosis and therapy, silent speech interface for speech impaired patients), computer-aided drug design, natural risk assessment, and computational models of biological neural networks.

Gérard Dreyfus holds the responsibility of a continuing education program in machine learning, which he created in 1991. He founded the French chapter of the IEEE Computational Intelligence Society in 2003. He is a member of the Machine Learning for Signal Processing (MLSP) Technical Committee of the IEEE, and of the Multimedia Tutorials Committee of the IEEE Computational Intelligence Society; he was a member of the Neural Networks technical committee. He belongs to the editorial board of several international journals and to the organizing committee of various international conferences.

Gary Bryce Fogel, Natural Selection, Inc., CA, USA

for contributions to computational intelligence and its application to biology, chemistry, and medicine

Gary B. Fogel received a B.A. in Biology from U.C. Santa Cruz in 1991 with a minor in Earth Sciences, and the Ph.D. in Biology from U.C. Los Angeles in 1998 where he was a Fellow of the Center for the Study of the Evolution and Origin of Life (CSEOL). As an employee at Natural Selection, Inc., he has produced numerous computational intelligence applications to biological, chemical, and medical problems. These include tools built for companies such as Eli Lilly & Co., Isis Pharmaceuticals, Beckman Coulter, Inc., and Althea Technologies, Inc., and under funding from the National Science Foundation, National Institutes of Health, U.S. Army, and other agencies. He has also led internal projects at Natural Selection, Inc. that have had broad interest in the application of computational intelligence to biomedical problems from sequence analysis and alignment, to phylogenetics, protein folding, and small molecule toxicity, as well as models in ecology, theoretical biology, and other disciplines. Through these activities, Dr. Fogel has broadened the appreciation of these approaches within the biological community. Simultaneously, he has helped computer scientists understand how computational intelligence can be best applied in biomedicine.

Among his many projects, he led the development of a novel computational intelligence drug discovery platform for the U.S. Army to accelerate high-throughput generalized protein-targeted

drug discovery. This technology results in collections of novel small molecule compounds that bind to protein targets and was first tested for anti-malarial drug discovery. He has also recently been using computational intelligence tools for pattern identification in the HIV genome associated with co-receptor usage and late-stage pathogenesis such as dementia and lymphoma. The resulting models can be used as biomarkers for these conditions, and hold great promise for pre-screening HIV patients for the likelihood of late-stage disease. He continues to work in the area of microRNA analysis, helping researchers understand the evolvability of microRNAs in HIV and their role in late-stage dementia and lymphoma in HIV patients.

Dr. Fogel currently serves as editor-in-chief for *BioSystems*, serves as an associate editor for the *IEEE Transactions on Evolutionary Computation*, and has served as a member of the editorial board of 6 other journals. He has published over 90 papers in peer reviewed journals, conferences, and books and also co-edited the books *Computational Intelligence in Bioinformatics* and *Evolutionary Computation in Bioinformatics*. He established the IEEE CIS Bioinformatics and Bioengineering Technical Committee, and also the annual IEEE Symposium on Computational Intelligence on Bioinformatics and Computational Biology serving as general chair in 2004 and 2005. He currently serves IEEE CIS as VP Conferences and manages Natural Selection, Inc. as its Chief Executive Officer.

Sushmita Mitra, Indian Statistical Institute, INDIA

for contributions to neuro-fuzzy and hybrid approaches in pattern recognition

Sushmita Mitra is currently Professor and Head of Machine Intelligence Unit, Indian Statistical Institute, Kolkata. She received her B.Tech. and M. Tech. in Computer Science from the University of Calcutta in 1987 and 1989, respectively, and the University Gold Medal. Moving to Indian Statistical Institute on a research fellowship, she was formally recruited there in 1991. From 1992 to 1994 she was in the RWTH, Aachen, Germany, on a DAAD Fellowship. She received her Ph.D. in 1995 from Indian Statistical Institute, with the thesis being subsequently published as an authored book.

Dr. Mitra has made pioneering contribution to neuro-fuzzy pattern recognition, both with theoretical analysis and practical applications. The generic family of systems developed by her, particularly those integrating rough sets into the computational intelligence framework, have significant practical applications in data mining and bioinformatics. Her contribution in neuro-fuzzy rule generation enabled the extraction of classificatory linguistic rules from fuzzy neural networks, thereby promoting their understandability in the background of the 1990's notion of their black-box nature. This led to the development of neuro-fuzzy expert systems. In recognition of her excellent research, she was made a Fellow of INAE – the highest engineering academy in India. It also culminated in the *IEEE Neural Networks Council Outstanding Paper Award* in 1994, for her pioneering work on neuro-fuzzy pattern recognition. Two of her papers belong to the list of Top-cited papers in Engineering from India.

She is a co-author of three books, *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing* (Wiley), *Data Mining: Multimedia, Soft Computing, and Bioinformatics* (Wiley), and *Introduction to Machine Learning and Bioinformatics* (CRC), beside several edited books. Dr. Mitra is an Associate Editor of *IEEE/ACM Transactions on Computational Biology and Bioinformatics*", *Neurocomputing*, and a Founding Associate Editor of *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (WIRE DMKD)*. She is the author of more than 75 research publications in referred international journals. Her two patents are pending with US and Indian Patent office. She has been listed among the top 100 women scientists of India in the prestigious volume *Lilavati's Daughters: The Women Scientists of India*, Bangalore: Indian Academy of Sciences (2008). Her current research interests include soft computing, data mining, pattern recognition, image processing, and Bioinformatics.

Marimuthu Palaniswami, The University of Melbourne, AUSTRALIA

for contributions to computational intelligence, learning systems, and nonlinear modeling

M. Palaniswami received his BE (Hons) from the University of Madras, ME from the Indian Institute of science, India, and Ph.D. from the University of Newcastle, Australia before joining the University of Melbourne, where he is a Professor of Electrical Engineering and Director/Convener of a large ARC Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP) with about 200 researchers and interdisciplinary themes as focus for the centre. Previously, he was a Co-Director of Centre of Expertise on Networked Decision & Sensor Systems. He served various international boards and advisory committees including a panel member for National Science Foundation (NSF). He has published more than 300 refereed journal and conference papers, including a number of books, edited volumes and book chapters. He was given a Foreign Specialist Award by the Ministry of Education, Japan in recognition of his contributions to the field of Machine Learning. He received the University of Melbourne Knowledge Transfer Excellence Award and Commendation Awards. He served as associate editor for Journals/transactions including IEEE Transactions on Neural Networks and Computational Intelligence for Finance. He is the Subject Editor for

International Journal on Distributed Sensor Networks. Through his research, he supported various local and international companies. As an international investigator, he is involved in FP6 and FP7 initiatives in the areas of smart city and Internet of Things (IoT). In order to develop new research capacity, he founded the international conference series on sensors, sensor networks and information processing. His research interests include Smart Sensors and Sensor Networks, Machine Learning, Neural Networks, Support Vector Machines, Signal Processing, Biomedical Engineering and Control.

Hao Ying, Wayne State University, USA

for contributions to theory and biomedical applications of fuzzy control

Hao Ying is Professor and Undergraduate Program Director in the Department of Electrical and Computer Engineering at Wayne State University, Detroit, Michigan, USA. Prior to joining WSU in 2000, he was an associate professor at the Department of Physiology and Biophysics at The University of Texas Medical Branch, Galveston, Texas. He received his Ph.D. degree in biomedical engineering from The University of Alabama at Birmingham in 1990, and obtained B.S. and M.S. degrees in electrical engineering from Donghua University, Shanghai, China, in 1982 and 1984, respectively.

Professor Ying started fuzzy control research in the 1980s. In an *Automatica* paper published in 1990, he presented a novel mathematical analysis technique that derived and revealed the mathematical structure of a fuzzy controller as well as its precise relationship with, and performance advantages over, the widely-used PI controller. It was the first technique in the field that derived the mathematical structure of a fuzzy controller and linked it to a conventional controller. Shortly after, he developed numerous analytical techniques to analyze and design various fuzzy control systems, making it possible to conduct fuzzy control analysis and design systematically within the framework of the nonlinear control theory. Recently, he showed this analytical approach to be equally valuable in rigorously studying the emerging type-2 fuzzy control. Professor Ying has also made contributions to another fundamental issue in fuzzy control theory - the universal applicability of fuzzy controllers. In a 1994 *Automatica* paper, he showed the first quantitative results in the field—formulas that could calculate the numbers of fuzzy sets and rules needed to satisfy any given approximation error. This and other related findings of his not only theoretically assure universal applicability of a broad variety of fuzzy controllers, but put the fuzzy approximation theory on par with traditional approximation theories (which are always quantitative). Lastly, Professor Ying developed an innovative fuzzy control system in 1989 and successfully used it for the real-time closed-loop control of mean arterial pressure in post-surgical patients in a Cardiac Surgical Intensive Care Unit. According to PubMed, this challenging exploration is the world's first attempt on real-time fuzzy control in biomedicine

Professor Ying has published one single-author book entitled *Fuzzy Control and Modeling: Analytical Foundations and Applications* (IEEE Press, 2000, 342 pages; foreword by Professor Lotif Zadeh), which contains solely his own research results. He has also published more than 90 peer-reviewed journal papers (38 on fuzzy control theory and 9 on its biomedical applications). His work has been widely cited—his h-index of 30 reflects over 2,500 citations (he is the sole author of 17 of the 30 publications and is the lead author for another 4).

Editor's Remarks *(continued from page 2)*

Besides the book review, we also include a review article on empowering Semantic Web with computational intelligence to complement the theme of the issue.

This year, the WCCI 2012 is moving Down Under to Brisbane in June, so start practicing your G'day Mates! as we look forward to catching up with fellow researchers from far and wide. This congress of our society not only serves as a global forum for scientists, engineers, educators, and students to discuss and present new research findings on computational intelligence, it also includes technical activities, social functions and networking sessions, which are excellent channels for making new connections and fostering friendship among our fellow CI researchers.

Meanwhile, we seek out your constructive feedback and valuable comments from all of you, which I am sure, will contribute in maintaining the high standards of this magazine.

Enjoy the issue!

K. C. Tan

Derong Liu, Chin-Teng Lin, Garry Greenwood, Simon Lucas, and Zhengyou Zhang

CIS Publication Spotlight

IEEE Transactions on Neural Networks

Optimal Tracking Control for a Class of Nonlinear Discrete-Time Systems With Time Delays Based on Heuristic Dynamic Programming, by H. Zhang, R. Song, Q. Wei, and T. Zhang, *IEEE Transactions on Neural Networks*, Vol. 22, No. 12 pt 1, Dec. 2011, pp. 1851–1862.

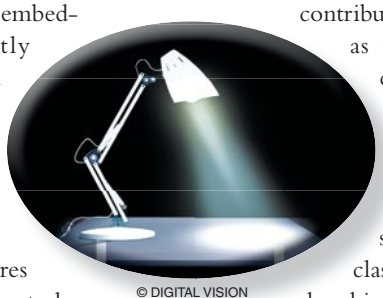
Digital Object Identifier: 10.1109/TNN.2011.2172628

"In this paper, a novel heuristic dynamic programming (HDP) iteration algorithm is proposed to solve the optimal tracking control problem for a class of nonlinear discrete-time systems with time delays. The novel algorithm contains state updating, control policy iteration, and performance index iteration. To get the optimal states, the states are also updated. Furthermore, the 'backward iteration' is applied to state updating. Two neural networks are used to approximate the performance index function and compute the optimal control policy for facilitating the implementation of HDP iteration algorithm. At last, two examples are presented to demonstrate the effectiveness of the proposed HDP iteration algorithm."

Generalized Constraint Neural Network Regression Model Subject to Linear Priors, by Y.J. Qu and B.G. Hu, *IEEE Transactions on Neural Networks*, Vol. 22, No. 12 pt 2, Dec. 2011, pp. 2447–2459.

Digital Object Identifier: 10.1109/TNN.2011.2167348

"This paper reports the study of adding transparency to neural networks. A class of linear priors (LPs) are considered, such as symmetry, ranking list, boundary, monotonicity, etc., which represent either linear-equality or linear-inequality priors. A generalized constraint neural network-LPs (GCNN-LPs) model is studied. Unlike other existing modeling approaches, the GCNN-LP model exhibits its advantages. First, any LP is embedded by an explicitly structural mode, which may add a higher degree of transparency than using a pure algorithm mode. Second, a direct elimination and least squares approach is adopted to study the model, which produces better performances in both accuracy and computational cost over the Lagrange multiplier techniques in experiments. Specific attention is paid to both 'hard (strictly satisfied)' and 'soft (weakly satisfied)' constraints for regression problems. Numerical investigations are made on synthetic examples as well as on the real-world datasets. Simulation results demonstrate the effectiveness of the proposed modeling approach in comparison with other existing approaches."



© DIGITAL VISION

IEEE Transactions on Fuzzy Systems

Stability Analysis and Control of Discrete Type-1 and Type-2 TSK Fuzzy Systems: Part I Stability Analysis, by S. Jafarzadeh, M.S. Fadali, and A.H. Sonbol, *IEEE Transactions on Fuzzy Systems*, Vol. 19, No. 6, Dec. 2011, pp. 989–1000.

Digital Object Identifier: 10.1109/TFUZZ.2011.2158218

"This paper provides novel sufficient conditions for the exponential stability of type-1 and type-2 Takagi–Sugeno–Kang (TSK) fuzzy systems. Three major contributions of this paper are as follow: (1) The new conditions do not require the existence of a common Lyapunov function, (2) The results for TSK type-2 systems cover a wider class of uncertain membership functions than those considered in earlier results on type-2 stability, such as triangular membership with uncertainty in the primary membership and in the support, (3) The stability conditions are formulated as LMI problems that can be conveniently solved with the LMI commands of the MATLAB Robust Control Toolbox. The use of the conditions in stability testing is demonstrated using several numerical examples and shown well-performance. This paper gives novel and broadening stability conditions for readers to further

Digital Object Identifier 10.1109/MCI.2012.2188565
Date of publication: 13 April 2012

apply to their researches to solve the stability problem."

On the Equivalence Conditions of Fuzzy Inference Methods—Part 1: Basic Concept and Definition, by Hirosato Seki and Masaharu Mizumoto, *IEEE Transactions on Fuzzy Systems*, Vol. 19, No. 6, Dec. 2011, pp. 1097–1106.

Digital Object Identifier: 10.1109/TFUZZ.2011.2160268

"This paper addresses equivalence of fuzzy inference methods. It first presents several well-known fuzzy inference methods. In the following, three fuzzy inference methods: the product–sum–gravity method, simplified fuzzy inference method, and fuzzy singleton-type inference method, are shown to be equivalent to each other. Thirdly, the equivalence conditions between the single input rule modules connected type fuzzy inference method and the single input connected fuzzy inference method are demonstrated. Finally, it also gives the equivalence conditions between the single input type fuzzy inference methods and the previous three fuzzy inference methods. Investigating the equivalence among various fuzzy inference methods is helpful to understand the relationship of those fuzzy inference methods."

IEEE Transactions on Evolutionary Computation

Orthogonal Learning Particle Swarm Optimization, by Z. Zhan, J. Zhang, Y. Li, and Y. Shi, *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 6, Dec. 2011, pp. 832–847.

Digital Object Identifier: 10.1109/TEVC.2010.2052054

"Particle swarm optimization relies on its learning strategy to guide its search direction. Traditionally, each particle utilizes its historical best experience and its neighborhood's best experience through linear summation. This paper proposes an orthogonal learning strategy to discover more useful information. The new learning strategy and the new swarm algorithm were tested on a set of 16 benchmark functions. Results suggest this enhanced swarm algorithm exhibits improved convergence."

Neighborhood Knowledge-Based Evolutionary Algorithm for Multiobjective Optimization Problems, by Z. Yu, H. Wong, D. Wang, and M. Wei, *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 6, Dec. 2011, pp. 812–831.

Digital Object Identifier: 10.1109/TEVC.2010.2051444

In this paper a new evolutionary algorithm, referred to as the neighborhood knowledge-based evolutionary algorithm, is proposed for multiobjective optimization problems. This new algorithm considers non-dominated sorting and the horizontal transmission of information in a candidate solution. It also exploits neighborhood knowledge acquired during the search process. The authors introduce a new notion, known as the measure space, which integrates multiple measures, such as the convergence metric and the diversity metric, to evaluate the performance of the algorithm. The new algorithm is tested on a large number of multiobjective optimization problems."

IEEE Transactions on Computational Intelligence and AI in Games

Search-Based Procedural Generation of Maze-Like Levels, by D. Ashlock, C. Lee, and C. McGuinness, *IEEE Transactions on Computational Intelligence and AI in Games,* Vol. 3, No. 3, Sept. 2011, pp. 260–273.

Digital Object Identifier: 10.1109/TCIAIG.2011.2138707

"This paper investigates the use of evolutionary algorithms for automatically generating maze-like levels for games. The work shows how interesting mazes with very different natures can be produced by changing the representation used by the algorithm, even while keeping the fitness function constant. The method easily allows a designer to incorporate fixed content (such as a set of rooms) within a larger evolved maze and to ensure that certain constraints on the final solutions are satisfied."

IEEE Transactions on Autonomous Mental Development

A Model of Neuronal Intrinsic Plasticity, by C. Li, *IEEE Transactions on Autonomous Mental Development*, Vol. 3, No. 4, Dec. 2011, pp. 277–284.

Digital Object Identifier: 10.1109/TAMD.2011.2159379

"Recent experimental results have accumulated evidence that the neurons can change their response characteristics to adapt to the variations of the synaptic inputs, which is the so-called neuronal intrinsic plasticity mechanism. In this paper, we present a new model on neuronal intrinsic plasticity. We first show that the probability distribution of the neuronal firing rates is more suitable to be represented as a Weibull distribution than an exponential distribution. Then, we derive the intrinsic plasticity model based on information theory. This study provides a more realistic model for further research on the effects of intrinsic plasticity on various brain functions and dynamics."

IEEE Transactions on Computational Intelligence and AI in Games (T-CIAIG)

Call for papers: Special Issue on Computational Narrative and Games

Special issue editors: Ian Horswill, Nick Montfort and R. Michael Young

Stories in both their telling and their hearing are central to human experience, playing an important role in how humans understand the world around them. Entertainment media and other cultural artifacts are often designed around the presentation and experience of narrative. Even in video games, which need not be narrative, the vast majority of blockbuster titles are organized around some kind of quest narrative and many have elaborate stories with significant character development. Games, interactive fiction, and other computational media allow the dynamic generation of stories through the use of planning techniques, simulation (emergent narrative), or repair techniques. These provide new opportunities, both to make the artist's hand less evident through the use of aleatory and/or automated methods and for the audience/player to more actively participate in the creation of the narrative.

Stories have also been deeply involved in the history of artificial intelligence, with story understanding and generation being important early tasks for natural language and knowledge representation systems. And many researchers, particularly Roger Schank, have argued that stories play a central organizing role in human intelligence. This viewpoint has also seen a significant resurgence in recent years.

The *T-CIAIG* Special Issue on Computational Narrative and Games solicits papers on all topics related to narrative in computational media and of relevance to games, including but not limited to:

- Storytelling systems
- Story generation
- Drama management
- Interactive fiction
- Story presentation, including performance, lighting, staging, music and camera control
- Dialog generation
- Authoring tools
- Human-subject evaluations of systems

Papers should be written to address the broader *T-CIAIG* readership, with clear and substantial technical discussion and relevance to those working on AI techniques for games. Papers must make sufficient contact with the AI for narrative literature to provide useful insights or directions for future work in AI, but they need not be limited to the documentation and analysis of algorithmic techniques. Other genres of papers that could be submitted include:

- Documentation of complete implemented systems
- Aesthetic critique of existing technologies
- Interdisciplinary studies linking computational models or approaches to relevant fields such as narratology, cognitive science, literary theory, art theory, creative writing, theater, etc.
- Reports from artists and game designers on successes and challenges of authoring using existing technologies

Authors should follow normal *T-CIAIG* guidelines for their submissions, but clearly identify their papers for this special issue during the submission process. *T-CIAIG* accepts letters, short papers and full papers. See http://www.ieee-cis.org/pubs/tciaig/ for author information. Extended versions of previously published conference/workshop papers are welcome, but must be accompanied by a covering letter that explains the novel and significant contribution of the extended work.

Deadline for submissions: September 21, 2012
Notification of Acceptance: December 21, 2012
Final copy due: April 19, 2013
Expected publication date: June or September 2013

Digital Object Identifier 10.1109/MCI.2012.2188600

Guest Editorial

Huajun Chen
Zhejiang University, CHINA
Philippe Cudré-Mauroux
Massachusetts Institute of Technology, USA

Special Issue on Semantic Web Meets Computational Intelligence

The Semantic Web (SW) [1] carries out the vision of a global web of data directly consumable and understandable to machines. This vast data space,consisting of open-linked data annotated with possibly expressive ontologies, promises significant opportunities for a variety of new web applications. The current Semantic Web languages and technologies are mainly built on traditional Artificial Intelligence approaches such as Description Logic, Logic Programming, Ontology Reasoning, etc. However, in a highly open, decentralized, dynamic, and vast Web environment, the building and development of a global data space calls for more expressive languages capable of dealing with fuzziness and vagueness in web semantics [2][3], and more efficient computational approaches to reduce the complexity of a number of problems inherent to the Semantic Web such as Web-scale querying and reasoning [4], vast decentralized semantic storage [5], and massive linked data analysis, etc.

The Computational Intelligence (CI) communities have accumulated a wealth of adaptive approaches such as fuzzy logic system, evolutionary computation, artificial neural networks that are particularly adept in tackling difficult problems in a highly dynamic and decentralized environment such as the Web. More and more researchers from the Semantic Web community are aware of the importance of introducing these CI methodologies and approaches to study the complexity of such a huge web of data. For example, Fuzzy Logic has inspired the design of variant fuzzy extensions of several Semantic Web languages. Nature-inspired optimization methods such as Genetic Algorithms (GA), Swarm Intelligence (SI), Artificial Immune Systems (AIS) have been witnessed in optimizing query answering and reasoning over such a vast, decentralized data space.

This special issue aims at promoting the discussion on current trends in the marriage of the Semantic Web and Computational Intelligence. Our goal is not only to introduce applying CI approaches into variant Semantic Web applications, but also present cutting-edge perspectives and visions to highlight future development. After a rigorous peer-review processes, we have finally selected four articles that cover the different aspects of Computational Intelligence applied in the Semantic Web research and development.

In the opening article, Christophe Guéret et al., describe how evolutionary and swarm computing can be applied to address the most typical application problems on the Semantic Web. First, they perform a systematic analysis on the most typical inference tasks including query, storage, entailment checking, consistency and satisfiability checking, and mapping. They then argue that the existing approaches to address underlying inference tasks necessarily fail given the vastness, dynamicity and complexity of the data generated by the Semantic Web. For each of these tasks they discuss possible problem-solving methods grounded in evolutionary and swarm computing. Finally, they highlight two successful case studies; the eRDF framework that provides evolutionary algorithms for querying, and a swarm algorithm for logical entailment computation.

In the second article, Hannes Mühleisen et al., give a positive answer to the question of whether swarm-based approaches are useful in creating large-scale distributed storage and reasoning system for the Semantic Web. They present a scalable, adaptive and robust semantic storage system, inspired by Swarm Intelligence, to store and analyze the massive amounts of semantic data. They also emphasize the problem of web-scale reasoning and propose the idea of reasoning over a fully decentralized and self-organized storage system that is based on the collective behavior of swarm individuals. Experiments of storage and reasoning performance are illustrated as well to show the general feasibility and scalability of proposed approaches.

In the third article, Jeff Z. Pan et al., study the problem of fuzzy representation and reasoning in the Semantic Web. This article presents how to make use of tractable profiles in OWL 2 and some of their fuzzy extension to provide tractable reasoning services for ontology applications. They describe a reusable reasoning infrastructure called TrOWL, and show how it can be used

Digital Object Identifier 10.1109/MCI.2012.2188579
Date of publication: 13 April 2012

to support several use cases such as mashups, process refinements validation, software engineering guidance for tractable applications of fuzzy and crisp ontologies.

In the the fourth paper, Liu et al., investigate how cloud infrastructure can help solve scalability issue of fuzzy reasoning in OWL. It reports a MapReduce-based framework that enables scalable reasoning on top of semantic data under fuzzy pD^* semantics (i.e., an extension of OWL pD^* semantics with fuzzy vagueness). This type of research is important as the content in the Semantic Web is akin to be fuzzy and the size of fuzzy information in it is extraordinary huge, leading to a number of challenges dealing with both the hugeness and vagueness for the Semantic Web content.

In summary, the first two articles focus on the applications of variant evolutionary computation approaches in reducing the complexity of a number of computational problems inherent to the Semantic Web such as query processing, semantic storage, and web-scale reasoning. The next two articles move to the use of fuzzy logic to represent and reason over fuzzy information in the Semantic Web with particular focus on the scalability issue of dealing with vast, fuzzy data in a tractable way.

We hope the four selected papers can cover the most essential aspects in the application of CI approaches in the Semantic Web. These aspects range from the state of the art technologies and solutions to tackle the critical challenges faced by the building and development of the Semantic Web, to typical case studies and visions for future development. Moreover, we hope this special issue can shed light on the importance of Computational Intelligence research in the Semantic Web, and stimulate further research in relevant fields.

Lastly, we would like to express our appreciation to our distinguished reviewers whose expertise and professionalism has certainly contributed significantly to the high quality of the special issue. We would also like to thank Dr. Kay Chen Tan, the Editor-In-Chief, for his helpful guidance and contructive suggestions in the whole process of organizing this special issue. Finally, we would like to thank the authors of the selected papers for their innovative research results.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Sci. Amer. Mag.*, 2001.

[2] G. Stoilosa, G. Stamoub, and J. Z. Pan, "Fuzzy extensions of OWL: Logical properties and reduction to fuzzy description logics," *Int. J. Approx. Reason.*, vol. 51, no. 6, pp. 656–679, July 2010.

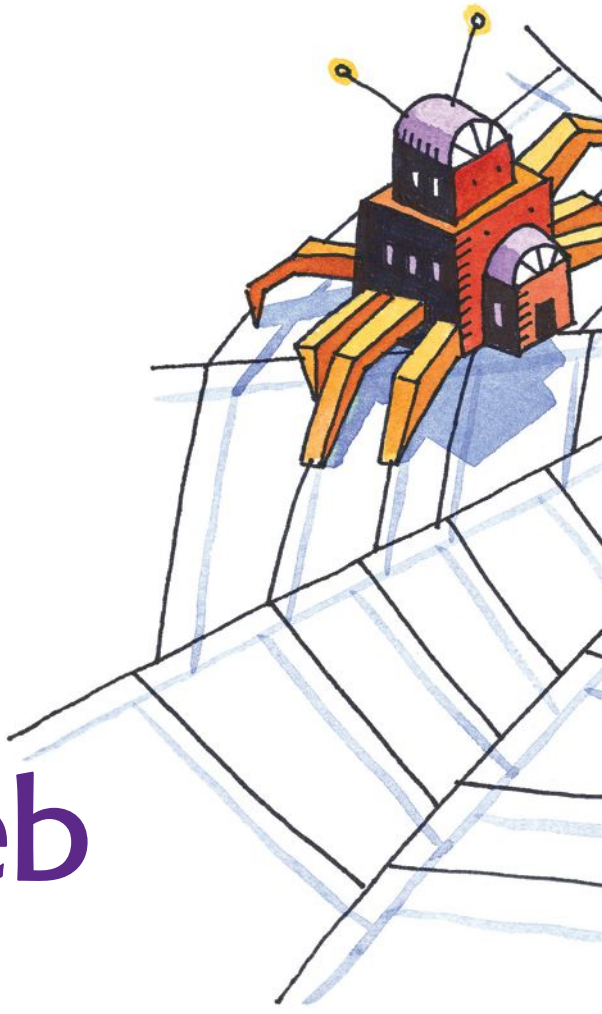
[3] T. Lukasiewicz and U. Straccia, "Managing uncertainty and vagueness in description logics for the Semantic Web," *J. Web Semantics*, vol. 6, pp. 291–308, 2008.

[4] D. Fensel and F. van Harmelen, "Unifying reasoning and search to web scale," *IEEE Internet Comput.*, vol. 11, no. 2, pp. 96–95, 2007.

[5] H. Mühleisen, A. Augustin, T. Walther, M. Harasic, K. Teymourian, and R. Tolksdorf, "A self-organized semantic storage service," in *Proc. 12th Int. Conf. Information Integration and Web-Based Applications and Services*, Paris, France, Nov. 2010, pp. 357–364.

Christophe Guéret, Stefan Schlobach, Kathrin Dentler, Martijn Schut, and Gusz Eiben
Vrije Universiteit Amsterdam, THE NETHERLANDS

Evolutionary and Swarm Computing for the Semantic Web



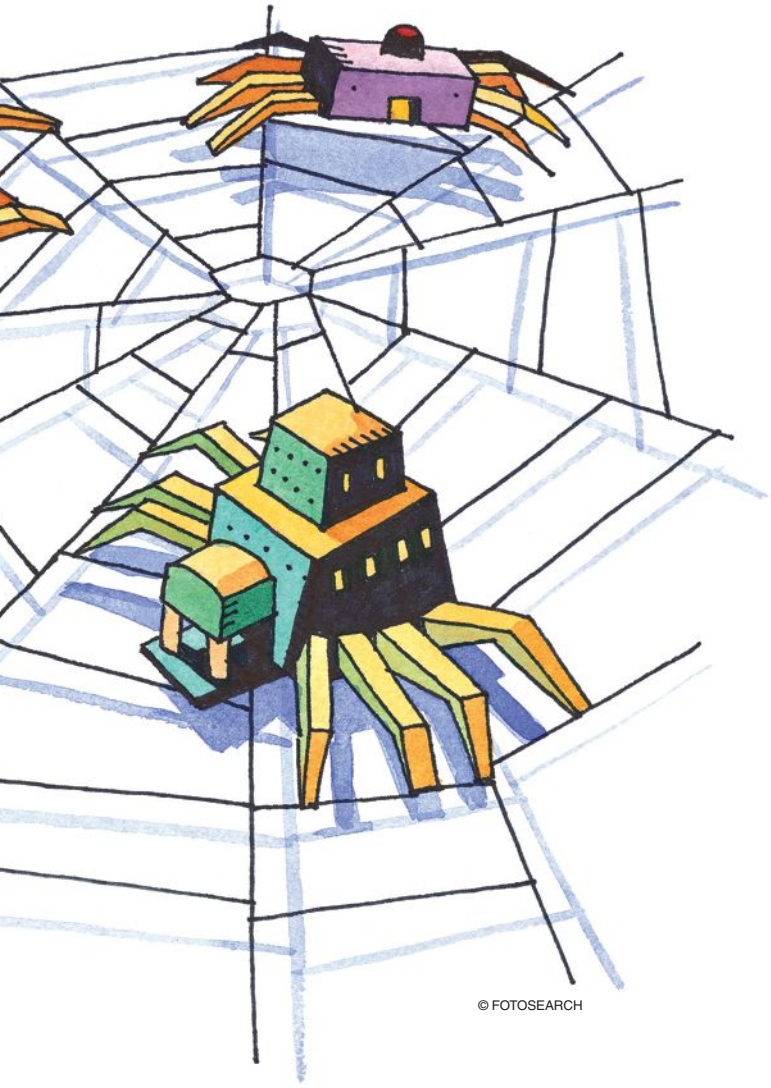
***Abstract*—The Semantic Web has become a dynamic and enormous network of typed links between data sets stored on different machines. These data sets are machine readable and unambiguously interpretable, thanks to their underlying standard representation languages. The expressiveness and flexibility of the publication model of Linked Data has led to its widespread adoption and an ever increasing publication of semantically rich data on the Web. This success however has started to create serious problems as the scale and complexity of information outgrows the current methods in use, which are mostly based on database technology, expressive knowledge representation formalism and high-performance computing. We argue that methods from computational intelligence can play an important role in solving these problems. In this paper we introduce and systemically discuss the typical application problems on the Semantic Web and argue that the existing approaches to address their underlying reasoning tasks consistently fail because of the increasing size, dynamicity and complexity of the data. For each of these primitive reasoning tasks we will discuss possible problem solving methods grounded in Evolutionary and Swarm computing, with short descriptions of existing approaches. Finally, we will discuss two case studies in which we successfully applied soft computing methods to two of the main reasoning tasks; an evolutionary approach to querying, and a swarm algorithm for entailment.**

Digital Object Identifier 10.1109/MCI.2012.2188583
Date of publication: 13 April 2012

I. Introduction

Since its creation some 20 years ago, the Web has been the place of major evolutions. Started as a publication mechanism for connected text, it has now become a rich content platform where everyone can publish and consume various types of media. Yet, all the content published is only accessible to humans, not machines. The next evolution of the Web, sometimes referred to as Web 3.0, will be that of a Web accessible to both machines and humans: a *Semantic Web* (SW) whose content can be interpreted by machines as it is explicitly modeled using languages with clearly defined *Semantics*. In practice, the Semantic Web connects data in a similar way as the WWW connects documents. Atomic data-units called *resources* are connected via labeled links with other resources and together these so called *triples* form a gigantic graph of *Linked Data*. The label of these links is where the Semantics are: links not only connect two resources but also convey the *meaning* of such connection.

1556-603X/12/$31.00©2012IEEE



The meaning of the types can be fixed using standardized schema and ontology languages such as RDFS and OWL [1]. Traditionally, the formal meaning of these languages is based on logical paradigms that were designed for small and hand-made knowledge bases, and come with a classical model-theory assigning truth to formula, and entailment based on this truth. Accordingly, the popular problem solving methods have their origin in database technology and classical logical paradigms, with centralized storage of, and access to, data; and with algorithms aiming at sound and complete results. The search is performed assuming full access to the complete dataset from a single location.

It is increasingly visible that in a highly complex, dynamic, context-dependent, opinionated and contradictory data space such as the Semantic Web, these approaches are insufficient. They are often prone to logical fallacies, usually intractable and can not easily cope with contextual information. The Semantic Web is a decentralized data space, a market place of ideas where contradictions and incoherence are common things for the data it contains. It is widely recognized that new adaptive approaches are required to exploit the ever growing amounts of dynamic Semantic Web data [2]. These insights have triggered research into the Semantic Web as a Complex System [3] and into more expressive formal languages capable of dealing with the multidimensionality and dynamicity [4] to name but a few. There is also a significant body of research about the potential of applying methods from Computational Intelligence to address a diverse set of problems that are inherent to the Semantic Web.

In this paper, we introduce the Semantic Web (Section II) and its typical usage and problem space (Section III). We then describe how evolutionary and swarm computing can be applied to address these problems and illustrate our point with two case studies (Section IV).

II. The Semantic Web

The World Wide Web is a decentralized system enabling the publication of documents and links between these documents on the Internet. A document is a piece of text, usually written in HTML and made available at a particular address (the URI). The links between documents are based on anchors put in these texts (hypertext links) and express a relation whose meaning depends on the interpretation made of the anchoring text. The Semantic Web uses the Web as a platform to publish and interlink data, rather than documents.

A. The Design of the Semantic Web

The Semantic Web replaces the HTML documents found in the documents by descriptions in RDF (Resource Description Framework) [5] about resources and uses typed relations instead of generic hyperlinks. On the Semantic Web, a URI is a **resource** representing a "real world" entity outside the internet or a document within it. For instance, the resource http://dbpedia.org/resource/Amsterdam is a Semantic Web representation of the city of Amsterdam in the Netherlands. The description associated to this resource contains factual information about it, expressed using triples. A **triple** is the combination of a subject, a property and an object. The subject being a resource from the Semantic Web, the property another resource and the object either a resource or a textual value (a **literal**). An example of such description for `dbpedia:Amsterdam` is shown in Table 1. In this table, we follow a common practice of using a compact syntax for the URIs, the interested reader will find the translation of these prefixes on the site http://prefix.cc.

Table 1 shows the two main advantages of the Semantic Web. Firstly, properties are also URIs and, thus, also have a description associated to them. These descriptions are defined in vocabularies (also called "ontologies") that provide a set of properties targeted to specific knowledge representation domains. For instance, the vocabulary "Friend Of A Friend

TABLE 1 Part of the description given for `dbpedia:Amsterdam`.

SUBJECT	PROPERTY	OBJECT
DBPEDIA:AMSTERDAM	RDF:TYPE	DBO:CITY
DBPEDIA:AMSTERDAM	FOAF:NAME	"AMSTERDAM"
DBPEDIA:AMSTERDAM	DBO:COUNTRY	DBPEDIA:NETHERLANDS

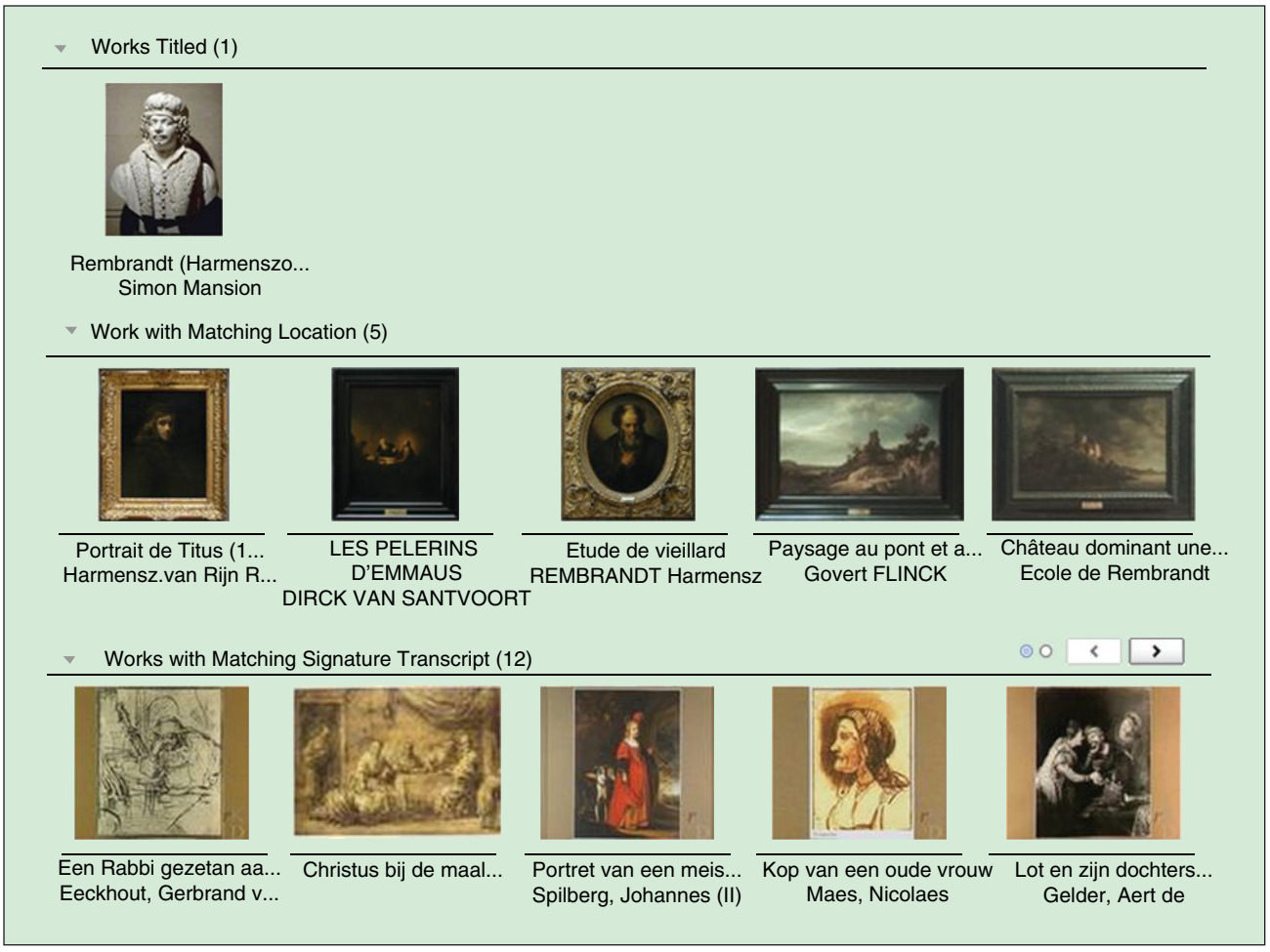


FIGURE 1 The cultural heritage portal "eCulture" shows how the Semantic Web impacts search, browsing and data integration. This image shows example results for "Rembrandt."

(FOAF)" contains properties used to describe social network oriented relationships. Secondly, URIs can be used in subject and object position and yield a network of relationships when URIs that are used as a subject/object in a triple are re-used as a object/subject in an other. For instance, the resource `dbpedia:Netherlands`, used as an object in Table 1, contains the triple ⟨`dbpedia:Netherlands, dbo:currency, dbpedia:Euro`⟩ as part of its description. In addition to this, the Semantics defined for some particular triples allows "reasoners" to derive new information by combining existing data. For instance, a triple expressing the fact that Amsterdam is in the Netherlands and another one saying that the Netherlands are part of Europe can be combined to deduce that Amsterdam is in Europe. This reasoning process will lead to the materialization of an explicit triple for an information that was otherwise implicit. The Semantic Web thus provides a way to express interconnected, structured knowledge about things in a Semantically well understood way.

Having introduced the basic notions needed to get a better understanding of what the Semantic Web is, we now have a closer look at the problems that need to be addressed when one wants to use Semantic Web data in an application. For this, we briefly introduce two web sites which leverage Semantic Web technologies for two different usages.

1) *Access to cultural heritage:* The site "eCulture"[1] (*c.f.* Figure 1) integrates data from different Dutch museums into a common content portal. This application shows how search, browse and data integration tasks are better achieved using Semantic Web technologies. Data integration is performed by re-using identifiers to refer to the same things. In doing so, graphs from different data sources are merged into a single graph. Search and browsing features are provided to explore the content of this graph. The usage of particular properties to describe the resources enables targeted search that can differentiate between finding keywords in the title or in the description of an artwork. Faceted browsing provides the user with a way to find relevant artwork based on sought properties. What cannot be seen in Figure 1 is the underlying usage of

[1]http://eculture.cs.vu.nl/europeana/session/search, accessed January 3, 2012

standards: the thesauri expressed in SKOS [6] and OWL [1], the integration with other Linked Data and the representation of manuscripts as unambiguous resources described in RDF.

2) *Music recommendation:* The site Seevl[2] (*c.f.* Figure 2) shows an example of personalization and recommendation made with Semantic Web technologies. This site provides a search interface to find musical artistes and then propose matching similar artistes. The matching is done using shared properties which are found in the graph. For instance, every band singer born in a particular city will be connected to that city in the data graph. All other artistes connected to that city are similar in the sense they share a common birth place. The usage of Linked Data to encode and link the data makes it possible for the site to integrate data from different sources and perform properties-based comparisons. Different resources can be compared based on their respective, semantically rich, descriptions in RDF.

FIGURE 2 Seevl is a musical search and recommendation system using the RDF data published by DBpedia (an RDF version of Wikipedia) based on dbrec.net.

B. The Nature of the Semantic Web

RDF allows for the creation of triples about anything on the Semantic Web. In fact, anyone is free to combine any subject, predicate and object, eventually coming from three different data sources, to create and publish a new triple. The decentralized data space of all the triples is growing at an amazing rate since more and more data sources are being published as semantic data, in RDF. But the size of the Semantic Web is not the only parameter of its increasing complexity. Its distributed and dynamic character, along with the coherence issues across data sources, and the interplay between the data sources by means of reasoning contribute to turning the Semantic Web into a Complex System.

1) **The massive amount of data and data sources**
Triples can be served by many data sources, eventually in large quantities. Modern triple stores, databases that are optimized to store triples, claim to be able to store and serve as many as a trillion of triples[3]. A popular view of the Semantic Web data space, the "LOD Cloud", referred to around 300 data sources [7], accounting for more than 30 billion of triples as of late 2011; A number that has not taken into account the data published by Facebook and several other web sites using embedded RDF content (RDFa). All this data being served is dynamic and may change on a frequent basis. It is even more so for data that is produced by sensors, such as those found in environmental monitoring networks [8].

These scale and dynamicity properties are similar to that of the Web of Documents, upon which the Semantic Web is grounded, but, unlike it, and because of the Semantics of the links, a triple published by one data provider can have a direct impact in combination with triples provided by many others. This emphasizes the importance of being able to deal with the *entire* Semantic Web, whatever its scale is.

2) **Opaque data locality**
As a set of triples, the content of the Semantic Web can be served by many different data sources at the same time. The global picture of a giant graph of information, is obtained by aggregating all the, possibly duplicated, triples served by all these data sources. From that global point of view, the locality of the triples is opaque and does not (or should not) affect the operation of the tasks making use of that content.

There is currently a large body of work, and an ongoing normalization work lead by the W3C[4], on the definition of provenance on the Semantic Web. The fact that any triple can be potentially served by anyone makes a strong case for tracking the origin of the triples. Opaque data locality also yields data synchronization and coherence issues as several servers can be serving different versions of a similar data set.

3) **Privacy and coherence concerns**
Even if triples from the Semantic Web can be served by many different data sources, some of these may be restricted to only being served by some particular server. Privacy concerns are a first motivation for this restriction: some data providers control their data and prevent any massive data acquisition process on their data set. Provenance is another motivation related to being seen as an authority to the data served. Finally, there is also a technical restriction associated to the dereferencing of the resources (URIs) used to get their description.

These specific constraints are a hiccup for algorithms that require having a local copy of the data they operate on. Whatever the motivation is, incoherence or limitations, the Semantic Web data should not be centralized but be available for use as is, directly from where it is being served.

[2]http://seevl.net, accessed January 5, 2012
[3]http://www.w3.org/wiki/LargeTripleStores, accessed January 6, 2012
[4]http://www.w3.org/2011/prov/wiki/Main_Page, accessed January 6, 2012

TABLE 2 Tasks and traditional solving methods to make use of a set of triples T. In the table, $\vdash$ stands for logical entailment and $t < Q$ implies that t is an instance of Q.

TASK	FORMAL PROBLEM DEFINITION	TRADITIONAL SOLVING METHODS
QUERYING	GIVEN T AND A QUERY Q, RETURN THE SET OF TRIPLES $\{t \in T\}$ SUCH THAT $T \vdash t < Q$	LOOKUP AND JOIN
STORAGE	GIVEN T AND A TRIPLE t RETURN $T \cup t$	CENTRALIZED INDICES, DISTRIBUTED HASH-TABLES
ENTAILMENT	GIVEN T. DERIVE $t \notin T$ WITH $T \vdash t$	CENTRALIZED AND PARALLELIZED DEDUCTION (RULES).
CONSISTENCY	GIVEN T. CHECK WHETHER $T \vdash \bot$ (FALSE)	LOGICAL REASONING
MAPPING	GIVEN T AND A MAPPING CONDITION C. RETURN $S,O \in T \times T$ SUCH THAT $C(S,O)$ LIKELY HOLDS WITH RESPECT TO T	SIMILARITIES SEARCH BETWEEN RESOURCES AND CLASSES. INDUCTIVE REASONING.

C. What Kind of Problem Solving Methods are Required?

To summarize: the Semantic Web data space is distributed, dynamic and inconsistent, sensitive to privacy issues, and opaque. This has immediate impact on typical characteristics expected of problem-solving methods applied to it: adaptivity, scalability and approximation.

- **Learning, adaptation and interactivity:** The Semantic Web is a market place of ideas, filled with contradicting and/or locally relevant facts. Requests expressed over this data are typically contextualized and algorithms should adapt to both the context and the data in order to provide accurate answers. Interactive re-evaluation of the user goal (i.e. the fitness function of the problem at hand) is also a desired capability.
- **Scalability and robustness:** The data is both decentralized and available in large quantities. Algorithms must be scalable to be able to deal with the amount of data available, they must also be robust enough to cope with the potential failure of data providers. The data from the Semantic Web is served by Web servers which may interrupt serving the data without prior notification. They can also change the content being served at any time. Problem solving methods should thus be able to cope with dynamic search spaces.
- **Anytime behavior and approximation:** When considering a distributed publication system such as the Web, one has to give up on completeness, soundness and determinism of answers. These values traditionally ensured by knowledge systems have to be traded for algorithms returning answers as they are found (anytime behavior) on a "best so far" basis (approximation). Additionally, one should not expect to get *all* the answers matching a given query and expect instead a non deterministic subset of them.

An additional desired feature for problem solvers is the ability to be parallelisable. The data model of the Semantic Web is a set of triples, which can easily be split into subsets of triples. Parallelisable strategies can hence benefit from that characteristic to provide increased computational performances.

III. The Semantic Web: A Problem Space for Evolutionary and Swarm Computing

The nature of the Semantic Web, and the typical problems that need to be addressed before one can make use of its richness of information, are precisely those that Computational Intelligence methods are good at solving. We will substantiate this claim in this section by analyzing a number of key tasks required for building Semantic Web applications.

In their systematic analysis [9] Harmelen *et. al* argued that typical Semantic Web applications require a rather restricted set of basic reasoning tasks ("Entailment", "Consistency", "Mapping"). However, the Semantic Web combines data with Semantics, *i.e.* provide extra meaning to data, and thus yields problems related to data management as well. We therefore extend the analysis of [9] with a set of basic data manipulation tasks ("Querying", "Storage"). Table 2 contains a formal description of the different tasks Semantic Web applications have to perform, along with a short explanation of the traditional solving techniques.

Every algorithm currently being put to use on Semantic Web data has been designed as a logic based method operating over a finite set of curated triples T (a "knowledge base"). As discussed in Section II-B, a typical triple set T on the Semantic Web is not static, nor curated, and all the guarantees of logical reasoning (completeness, soundness, determinism, …) are lost. Instead, one can only *aim* at them and thus *optimize* towards these ideals. The consensual approach is to fit the Semantic Web data into a knowledge base by downloading, aggregating and curating subsets of its content. The problem is therefore adapted to fit the currently available solving methods, rather than being addressed with novel techniques.

In order to find solving methods capable of dealing with the complex character of the Semantic Web we propose to rephrase the logical formulations of the tasks as optimization problems. Proper solving algorithms have then to be found for these problems. Evolutionary and Swarm algorithms are known to perform well on optimization problems with large, and eventually dynamic, search spaces. In the subsections, we go through all the tasks, look at them from an optimization point of view and discuss the approaches that have already been taken to tackle them. A summary of the outcome of this discussion is provided in Table 3.

A. Querying

1) *Problem description:* The task of querying the content of the Semantic Web consists of finding the set of triples $\{t \in T\}$ that matches a given query Q (we denote this by $t < Q$). The query language for Semantic Web data is SPARQL [10]. SPARQL queries are constructed from so called triple patterns usually connected in a graph. Those triple patterns are triples with free variables. Solving such query means looking for parts of the data set that match the given set of triple patterns. Figure 3 shows a given query (on the left) and a matching solution (on the right).

TABLE 3 The interaction with the Semantic Web is made through different tasks which can be phrased either as logic or optimization problems.

TASK	LOGIC PROBLEM	OPTIMIZATION PROBLEM	RELATED WORK
QUERYING	CONSTRAINT SATISFACTION	CONSTRAINED OPTIMIZATION	ERDF [13]
STORAGE	CONSTRUCTION OF SETS	CLUSTERING	SWARMLINDA [20]
ENTAILMENT	LOGICAL DEDUCTION	MULTI-OBJECTIVE OPTIMIZATION	SWARMS [24]
CONSISTENCY	(UN)SATISFIABILITY CHECKING	CONSTRAINED OPTIMIZATION	-
MAPPING	LOGICAL DEDUCTION	CLASSIFICATION	PSO [36], GOSSIPING [31], EVOLUTIONARY STRATEGY [37]

The nodes starting with a question mark are variables that have to be instantiated.

2) *Standard formulation and solving approach:* Solving a SPARQL query is a constraint satisfaction problem. The query defines the constraints and the triple set T is the domain in which the variables can pick their binding from. Taking inspiration from the work done in databases, the resolution of SPARQL queries is made of iterated lookup and join operations. A lookup consists of getting the set of triples that match a single triple pattern (an edge of the query graph) and, typically, loads the results in memory. A join is the finding of overlapping resources amongst two lookup result sets in order to produce a connected result. This construction mechanism ensures that all the possible answers are found and validated against all the constraints. This approach is adopted by the majority of existing triple stores such as Sesame [11] and YARS [12].
3) *Optimization problem:* The constraints expressed by the query can be relaxed into an objective function to optimize. In particular, considering the size of the Semantic Web, one may not expect to get all the answers to a particular query. Furthermore, considering the risk of incoherence and data source high churn rate, one may not assume having all the information at hand at a particular time. We thus relax the constraints of the query patterns into an objective function. The goal is to find bindings validating as many of the constraints as possible. This function has to be optimized under the constraint that the bindings used generate a connected solution graph.
4) *Solving methods:* With "eRDF"[13], [14], [15], we investigated using evolutionary computing to solve this problem. Solutions to the SPARQL query are found in an iterative guessing process based on evolutionary computing. More information on eRDF will be provided in Section IV-A.

B. Storage

1) *Problem description:* Storing a new information into the Semantic Web boils down to inserting a new triple t into a set of triples T. After the insertion, the new set of triples is equal to $T \cup t$. Because of the size of the data sets, this storage has to be achieved so as to facilitate the retrieval of the newly inserted triple. In fact, triples have to be inserted in such a way that their retrieval is optimized for lookup operations (*c.f.* previous section on querying). It is thus desirable that triple sharing subject and/or properties and/or objects are grouped together.
2) *Standard formulation and solving approach:* The standard insertion strategy of databases is to save the data and index it. The indexes ensure efficient lookup operations at the cost of an higher insertion time and more space needed for the data structures. In a distributed setting, a pre-determined indexing strategy applied by a central entity ensures an optimal load balancing between different storage nodes [16].

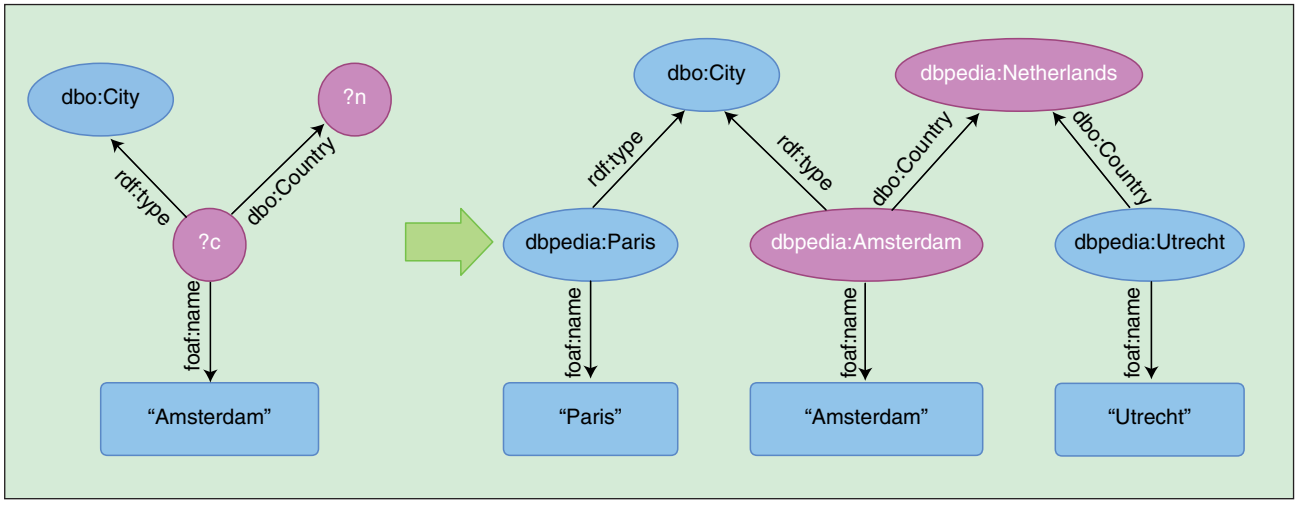


FIGURE 3 Example of a simple query (left part) and a matching sub-graph within an RDF graph (right part). The query reads as "Find a city ?*c* whose name is "Amsterdam" and which is in the country ?*n*", the result found is ?*c* = `dbpedia:Amsterdam` in ?*n* = `dbpedia:Netherlands`.

3) *Optimization problem:* In an optimization setting, the indexing task leaves place to clustering. The single optimum guaranteed of the pre-determined grouping rules defined by the indexes is replaced by a measure of the clustering efficiency. Considering a set of triples and a set of storage nodes, different clusterings (assignment of triples to nodes) of different quality can be established. The problem solver will strive to produce a clustering as efficient as possible by re-allocating data among the storage nodes.
4) *Solving methods:* Collective intelligence has already proven to be efficient for network routing [17] and data clustering [18]. Two features that combined, lead to triples clustering and query routing capabilities. Additionally, swarm algorithms can run continuously and adapt to changes in the data. SwarmLinda [19] shows how an artificial ants algorithm can be leveraged to create a self-organizing system with emerging triple clusters and optimized routes for querying. SwarmLinda applies a sorting method inspired by the brood sorting ability of real ants: artificial ants continuously take triples from one location and move them to another where similar triples are found. Traces left by the triples on the network connections taken during their journey are used for routing purposes at query time [19]. The basic concept has also been extended to provide a self-organized semantic storage service ("S4"), where RDF triples can be stored and queried in a fully distributed fashion [20]. Following a similar idea, PIAF [21] makes use of traces left by facts moved among the peers of a peer-to-peer system. These traces are used to efficiently disseminate data and route queries.

C. Entailment Checking

1) *Problem description:* Inference is the task of extracting information that is implicitly encoded in the formal representations of the data. In most knowledge representation languages, and also in those used to model knowledge on the Semantic Web, entailment is one of the core semantic notions. It is the fact that a formula logically follows from other formulas, *e.g.* a set of triples. As formulas that can be entailed are not necessarily explicit in the original data base, inference algorithms must derive those entailed formulas according to some calculi. The most prominent kind of entailment in Semantic Web is *relation checking* (whether two resources are related by some property), with the special case of *instance checking* (when this property is the `rdf:type` property). The reasoning tasks of realization and retrieval defined in [9] are combinations of the entailment we discuss here.
2) *Standard formulation and solving approach:* The state of the art approach for entailment checking consists of applying a given set of predefined rules over a static set of triples T. This process assumes that T has been curated to remove logical inconsistencies. It is incremental and has to be repeated from scratch every time the content of T changes. The search process is stopped as soon as no new facts are derived and added to the data set. The most efficient entailment checking engine to date, WebPIE, implements this strategy to derive the implications of a large number of triples [22]. A similar approach is also found in more common reasoners [23].
3) *Optimization problem:* In a dynamic and incomplete environment such as the Semantic Web, the goal of creating all the triples that can be derived from T by entailment (the "closure" of T) is actually that of deriving as many new triples as possible. Besides, on the Semantic Web it is unrealistic to consider that T will contain no inconsistencies. A closure will inevitably contain contradicting facts, a situation that has to be avoided. Entailment can thus be defined as an optimization problem around two opposite goals: generate as many new triples as possible and generate few of them in order to avoid deriving inconsistent information. More objectives could also be considered to, for instance, take into account the locality of the data, or its reliability. On the Semantic Web, entailment is hence a multi-objective optimization problem.
4) *Solving methods:* In [24], we applied swarm computing to the generation of the closure of a set of triples. Each of the swarm entity ("agent") is responsible for one reasoning rule used to derive new triples. Agents browse the graph in search of triples that match their rule. They follow properties to move from resource to resource, effectively visiting nodes in the graph. When an agent encounters a matching triple pattern, it fires its rule and writes down the newly derived triple to the graph (creates a new edge). This work is described in more detail in Section IV-B.

D. Consistency and Satisfiability Checking

1) *Problem description:* One of the prominent logical task is consistency checking. That is, checking whether a data set is internally contradictory. Formally, *inconsistency* is defined as the non-existence of a model for a set of triples while a concept or class is called *unsatisfiable* if it is empty in all models. Inconsistencies or the existence of unsatisfiable concepts usually points to modeling errors. Detecting them automatically can thus be very useful and if done early, at modeling time, avoid publishing inconsistent data.
2) *Standard formulation and solving approach:* From a logical point of view, the detection of these modeling errors is a satisfiability (SAT) problem. The inconsistency checking is a model-finding problem where the task is to find a truth assignment that satisfy a given set of assertions, defined by the triples. A set of assertions can be proven unsatisfiable by model-counting if the number of satisfying truth assignments is equal to zero. A truth assignment is a function that associate a boolean value (true or false) to all the variables mentioned by the set of assertions. Logical proof techniques such as the Tableau calculus [25] and the Davis-Putnam-Loveland procedure [26] are commonly used to approach these problems.

3) *Optimization problem:* An optimization formulation of SAT is obtained by, first, relaxing it into a maximum SAT (MAXSAT) problem where the maximal set of satisfiable assertions is sought. Then, that goal is relaxed into an objective. The optimization thus goes about finding a truth assignment that validate as many of the assertions as possible under the constraint that all the logical assertions get a truth value, true or false, assigned to them. This is a constrained optimization problem.
4) *Solving methods:* There seem to be no optimization algorithms that have been applied to address the problem of consistency and satisfiability checking. The literature however provides a wide range of SAT solver based on soft computing techniques. GASAT [27], for instance, uses evolutionary computing to evolve a population of candidate truth assignments. Similar goals have been achieved by Abbass using a swarm of artificial bees [28]. These algorithms should be directly applicable to Semantic Web data and used to tackle the optimization problem.

E. Mapping

1) *Problem description:* The goal of mapping is to find related resources within a set of triples T. The relation sought depends on the mapping use case. For vocabularies (classes, properties, …), the aim is to arrive at a logical organization. The relations established are inclusion, subsumption and specification, to name only but a few. For general resources describing entities, the aim is to establish equivalence. Several resources can be used to refer the same real world entity, which is, by existence, unique. We extract a generic mapping condition c from these two typical usages, to be further specified according to the data at hand and the desired relations.
2) *Standard formulation and solving approach:* State of the art approaches perform an (almost) exhaustive search over the search space, testing every pair of resources against the matching condition c. This condition is either stipulated by human experts or derived by some inductive method. If the condition is met, a new relation is created. Exemplifying this, the linker tool "Silk" [29] uses manually created linking specification to browse and connect resources found in two different data sets.
3) *Optimization problem:* The matching problem can be formulated as a classification problem, where a mapping can be predicted from the similarity between the extensional information (description) of two resources [30]. The quality of the classification can then be measured in different ways, *e.g.* by using a training and control data sets, and be used as a function to optimize. An alternative way of looking at mappings is to optimize the mapping condition itself, *i.e.* find the best similarity measure between the source and target data sets.
4) *Solving methods:* The size and decentralized nature of the Semantic Web makes it unrealistic to establish mappings between every resources it contains. Even if possible, such mappings may not be useful as triples distant from each other in the graph are also likely to be semantically unrelated. Instead, mapping established among neighbor resources are more likely to be both useful and semantically meaningful. Based on this assumption, Semantic gossiping [31] has been proposed to establish local agreements among the peers of a decentralized systems. Using this form of collective intelligence, semantic interoperability can be achieved at network scale [32] and mappings can be established [33].

Other approaches focus on the size of the search space by trying to explore less of it. GAOM [34] and GOAL [35] are two initiatives making use of genetic algorithms to explore the search space. They do it around two different axes. While GAOM evolves a set of mappings, the optimisation process of GOAL is targeted towards the similarity function between the two data sources. The fitness function of GOAL is a similarity function defined as a weighted sum of different features such as the string distance between the labels of the resources or the number of instances of concepts. The MapPSO [36] algorithm follows a strategy similar to that of GAOM but makes use of Particle Swarm Optimization (PSO) instead of evolutionary computing. A set of candidate solutions explore the search space shaped by the similarity function in a quest for positions with the highest correctness.

The feature used in weighted sum similarity functions may have different importance. Because of that, it is relevant to investigate the composition of the weighted sum serving as a similarity function and allow it to adapt to the data. In [37] we compared the results of different algorithms, including an evolutionary strategy, at performing a sound classification while explaining the results with an adjustment of the weights in the similarity function.

IV. Case Studies

Over the past few years, the Vrije Universiteit Amsterdam has been actively promoting the marriage between Computational Intelligence and the Semantic Web. Our main focus were placed on dissemination events (NatuReS workshop[5], SOKS Symposium[6]) and two flagship projects. In the following sections, we highlight these two attempts at tackling Semantic Web tasks with evolutionary and swarm computing. We recall the main features of the algorithms developed, some results and the lessons learned. Interested parties are invited to consult the related papers for more in depth details about this work.

A. Evolutionary Algorithms for Querying: The Erdf Framework

Querying the Semantic Web is defined as the task of finding triples in a huge, and possibly distributed, graph (set of triples). The solutions to the query have to fit a given query defined by a set of triple patterns, a graph with free variables.

[5]http://natures.few.vu.nl, accessed January 6, 2012,
[6]http://www.few.vu.nl/soks/symposium, accessed January 6, 2012

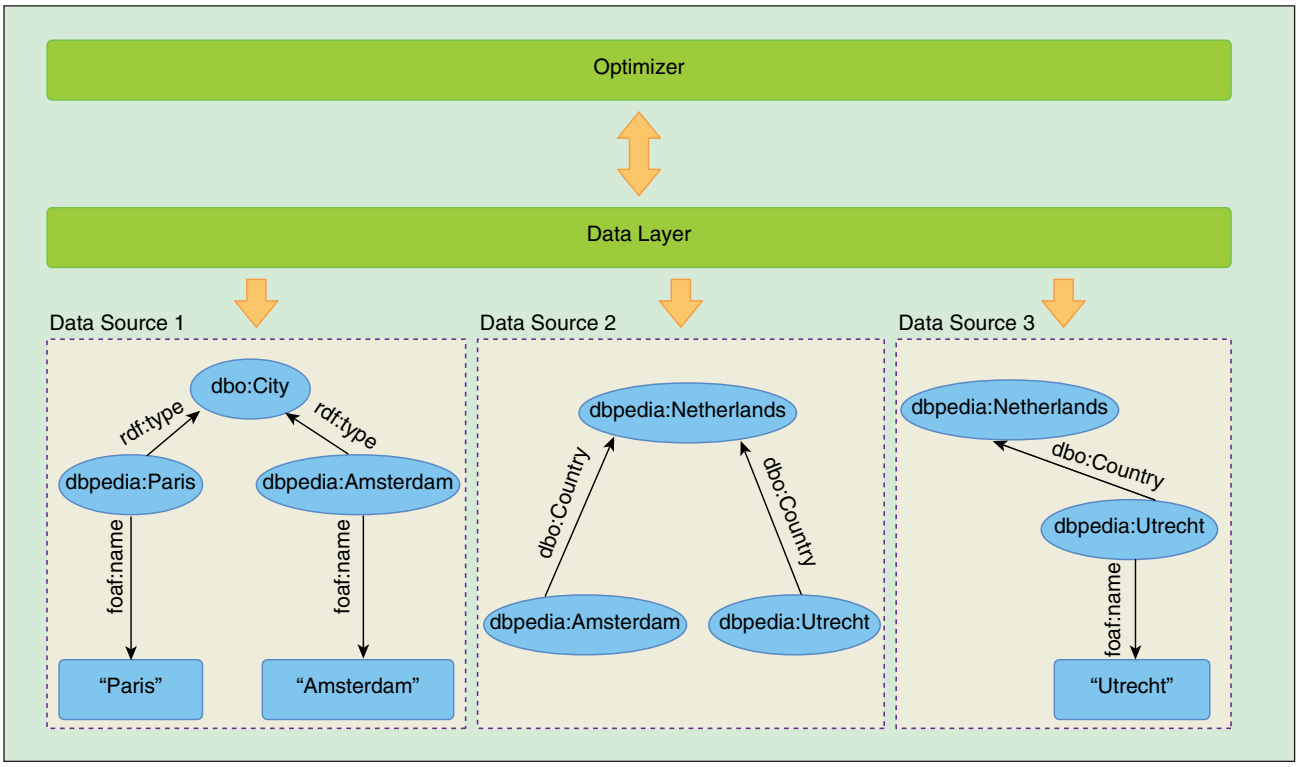


FIGURE 4 eRDF consists in two components: an optimizer and a data layer providing an abstract view over the different data sources.

The traditional way to find answers to the query is to 1) lookup the lists of triples matching the edges of the query and 2) join these partial results. In the example of Figure 3, this consists of retrieving the sets of triples matching $\langle$*, `rdf:type`, `dbo:City`$\rangle$, $\langle$*, `dbo:country`, *$\rangle$ and $\langle$*, `foaf:name`, `"Amsterdam"`$\rangle$. The sets are then joined based on the overlap between their subjects/objects. The result is a final list of every solution to the query. The two shortcomings of this approach are the necessity to pre-fetch all the potentially useful edges and the computational cost of merging the lists.

One can observe that testing if a given solution exists in a graph is easy and only requires checking the presence of the different edges and nodes. The eRDF framework [13], [14], [15] leverage this observation. Its search process is to guess and verify solutions to a SPARQL query. The framework consists of two main components: a data layer and an optimizer[7] (see Figure 4). The data layer provides an abstraction over the data coming from several RDF sources and the optimizer makes use of the data provided by this data layer to build the answers to the queries. Both components are only weakly coupled and may be implemented as two different services. This separation allows the data layer to re-use information across different optimization processes. It also allows the duplication of instances of the optimizer and/or the data layer in order to scale with the number of queries to be solved.

We describe below these two components and then conclude on some results from our prototype implementation.

1) *Data layer:* The data layer is a generic component designed to provide the optimizer with a limited set of primitives over $T = \bigcup_{i=1}^{m} T_i$. With T_i being the triple sets provided by m different data sources. This data layer does not store any "own" data. As in [38], eRDF starts with an empty knowledge base that is filled, at run time, by retrieving the necessary data from the Semantic Web. All the triples are acquired on demand from the different data sources. As a result, the introduced set of triples T is actually an *abstract* set that is never available to the optimizer as such. Data sources may become unavailable and/or require frequent updates. Under these conditions, maintaining a complete, up-to-date, copy of T would be very difficult, if not impossible.

The data layer implements two primitives taking a triple pattern as input. A first one used to verify the validity of the pattern (ASK) and a second one to get a resource matching the triple pattern (GET).

a) *ASK:* A triple $t = \langle s, p, o \rangle$ is considered to be valid if $t \in T$. This first operation is covered by the primitive ASK ($\langle s, p, o \rangle$). Besides telling if the entire triple is valid, the data layer can also be used to test the partial validity of a triple. That is ASK ($\langle *, p, o \rangle$), ASK ($\langle s, *, o \rangle$) and ASK ($\langle s, p, * \rangle$), with * being a wild card allowing any resource to be used. Such ability to test the partial validity

[7]The term "optimizer" is used here in reference to optimization problems, not the optimizer as found in databases systems.

of a triple pattern is used by the optimizer to assign partial rewards to the bindings of candidate solutions.

b) *GET:* The get primitive is used to get a resource that matches a given triple pattern. There are three of them, each returning a result matching one of the possible triple patterns: GET $(\langle *, p, o \rangle)$, GET $(\langle s, *, o \rangle)$, and GET $(\langle s, p, * \rangle)$ while respectively returning a resource to be used as a subject, a predicate or an object. Note that these primitives are aimed at returning one result at a time in a non-deterministic way. This design aspect makes the data layer more robust, allowing the return of results based on the information currently available.

3) *Optimizer:* The optimizer is in charge of generating candidate solutions and checking their validity. We use an evolutionary algorithm [39] for this process. The algorithm is memetic [40], taking care of the evolution of a population of candidate solutions while preserving some knowledge across the different generations. This knowledge is used to gather information about the data being used. It also shares some features with evolutionary strategies as the whole population is seen as parents for the generation of offsprings. A local search and a crossover operator are used to generate new candidate solutions and increase the population by a pre-determined factor. Then, similarly to what is done with "$(\mu + \lambda)$" selection of evolutionary strategies, the parent population and the offspring population are joined and shrunk down to the original population size.

We turned the constraint satisfaction problem of finding a set of bindings that validate all the triple patterns of the query into a constrained optimisation. The problem addressed by eRDF is finding a set of bindings that maximize the number of validated triple patterns. By turning the problem of query resolution into a constrained optimisation problem, eRDF implicitly relaxes all the statements of the query. Basically, the relaxation mimics a query where all the statements would be made optional and for which only the results with a maximum number of statements would be returned. eRDF is non deterministic and does not guarantee completeness. These two traditionally desired features of a query engine are traded for a better response time and an improved robustness. Additionally, the search process can cope with a changing search space (*e.g.* new data available) and can stream back solutions at the end of every generation.

The algorithm is outlined in Algorithm 1. Its three key steps are the *evaluation of the population* (assessing the quality of the candidate solutions, lines 3–4), the *selection of survivors* (selection of solutions to keep for the next generation, lines 5–15) and the *generation of offspring* (creation of new candidate solutions, lines 16–19).

a) *Population evaluation:* The evaluation performs checks on the quality of the population. Based on its bindings β that associate a value to every variable in the query, a candidate solution is given a *fitness* score between 0 and 1. We refer-

ALGORITHM 1 The search starts with a parent population P. The main loop consists in expending the population with new candidate found by a local search and then shrink it down to its original size. Candidate solutions that survived *max_age* generations or have an optimal fitness value are streamed back as results to the request and added to a taboo list τ.

```
1  Initialize population P;
2  while not terminated do
       /* Evaluation */
3      foreach Candidate solution β in P do
4          evaluate(β);
       /* Selection */
5      P ← survivors_selection(P);
6      f* ← best fitness of P;
7      foreach Candidate solution β in P do
8          if fitness(β) = f* then
9              age(β) ← age(β) + 1;
10         else
11             age(β) ← 0;
12         if age(β) = max_age or fitness(β) = 1 then
13             Output β as a result;
14             foreach Triple pattern g in G do
15                 τ ← τ ∪ g|β;
       /* Generation */
16     foreach Candidate solution β in P do
17         β' ← generate(β);
18         if β' ∉ P then
19             P ← P ∪ β';
```

ence by $G|\beta$ the triple set created by instantiating the triple patterns G of the query with the bindings β. The optimization target is to maximize the size of this set while giving more credit to candidate solutions with connected triples (see Equation 1).

$$\text{fitness}(\beta) = \frac{\text{maximal\_component}(G|\beta)}{|G|} * \sum_{g \in G} \frac{\text{reward}(g|\beta)}{|G|}. \quad (1)$$

The function $\text{reward}(g|\beta)$ credits the bindings β with respect to a single triple pattern g from G. This function can be freely customized to achieve different goals under the conditions

TABLE 4 Rewarding scheme for $g|\beta = \langle s, p, o \rangle$. The conditions are mutually exclusive and tested in the order of the table. A get($\langle s, *, o \rangle$) is typically computationally expensive and is thus tested last.

CONDITION	REWARD($G|\beta$)
$\langle S, P, O \rangle \in \tau$	0.25
ASK($\langle$S, P, O$\rangle$) *IS TRUE*	1
GET($\langle$*, P, O$\rangle$) $\neq \varnothing$	0.5
GET($\langle$S, P, *$\rangle$) $\neq \varnothing$	0.5
GET($\langle$S, *, O$\rangle$) $\neq \varnothing$	0.5

that $\text{reward}(g|\beta) \in [0,1]$. A reward of 1 reflects an optimal binding whereas a value of 0 means a non existing assertion. The ordering of the values has also to be respected, that is $\text{reward}(g|\beta) < \text{reward}(g|\beta')$ is equivalent to saying β' is a better solution than β for g. The notion of a candidate solution being better than another one is subject to interpretation and depends on the semantics of the rewarding scheme. For instance, one could consider a rewarding scheme solely based on size and the correctness of the set of triples created by the candidate solution ($G|\beta$). The rewarding scheme implemented in our prototype is reported in Table 4.

Once the entire new population is evaluated, its members are sorted and the population is trimmed down to the initial population size.

b) *Survivor selection:* At this stage, the population is sorted according to the fitness of its individuals and cut down to its default size. The age of the best individual(s) is increased and their optimality is checked. Candidate solutions that survived a *max_age* consecutive generations or that have a fitness equal to 1 are considered to be optimal. Such solutions are sent back to the user and the triples created by their bindings are added to the taboo list τ. This list penalizes the optimizer for re-using triples that have already been used in previously (locally) optimal solutions. This allows for the finding of diverse answers while not prohibiting the re-use of triples.

c) *Offspring generation:* Two strategies are implemented to generate new candidate solutions: 1) using the graph to propagate new values (local search) and 2) combining two candidate solutions (crossover). Both strategies are applied once for every individual in the parent population leading to a maximum increase of total population by a factor of 2. This value stays a theoretical maximum as duplicate candidate solutions are detected and removed from the population at the end of the generation process.

❑ Local search

The local search strategy is a three step process (see Figure 5). In the following ϵ denotes a small value, typically around 0.01, used to avoid having probabilities equal to zero.

1) Every binding has a probability to be changed proportionally to the expected reward gain such change would yield (*c.f.* Equation 2). This drives the evolution towards finding good assignments for pivot nodes before focusing on the branches of the query graph. A desired behavior for answering SPARQL queries.

$$p(v,\beta) \propto |\{g \in G \mid v \in \text{var}(g)\}| - \sum_{g \in G \mid v \in \text{var}(g)} \text{reward}(g|\beta) + \epsilon \quad (2)$$

2) Once a variable has been elected for a value change, an edge is selected to propagate a new assignment. The probability of using a given edge is proportional to its reward (*c.f.* Equation 3). Only the edges having a connection with the variable to be changed are considered in this process.

$$p(g,\beta) \propto 1 - \text{reward}(g \mid \beta) + \epsilon, \forall g \in \{g \in G \mid v \in \text{var}(g)\} \quad (3)$$

3) A GET query is issued using the selected edge and a new value is assigned to the variable. In the example of Figure 5, "Amsterdam" is a literal constant but the model also works using edges between two variables, thereby leading to a cascade effect of variable assignments and a propagation of their efficiency.

❑ Crossover

The crossover operation takes the candidate solution currently considered and another candidate solution randomly picked, as input. The genotypes of these two candidate solutions are combined in order to generate a third, new, candidate solution. For each variable assignment, the binding credited with the highest reward is picked. The decision is random when ties occurs.

3) *Experiments and conclusion:* A prototype implementation of eRDF is available online, under a free license, at https://github.com/cgueret/eRDF and described on http://www.erdf.nl. This implementation has been tested on answering large queries made up of hundreds of statements compromising tens of variables. The results are promising [15], showing that eRDF can effectively scale to such numbers and provide answers to the submitted queries.

Figure 6 contains an excerpt of the results described in [15]. It shows how eRDF performs on a set of requests of varied complexity (radius) and size (number of triple patterns). What can be observed on these graphs is that eRDF generally performs better on complex queries than on simpler ones. Perfect solutions to queries with at least 20 patterns are always found, under 100 seconds (a bit less than 2 minutes). This finding shows how eRDF turns this adverse complexity to its advantage, by using the constraints as guides for the evolutionary process (*c.f.* the local search process). The number of variables is also directly related to the size of the genetic material of the candidate solutions. A complex query leads to richer candidate solutions undergoing a better guided evolution.

In comparison, traditional implementations of querying engines only suffer from an increasing complexity. The number of patterns in the query is related to the number of lookups to be performed whereas the number of variables relates to the quantity of joins to be achieved. As a result, when compared to eRDF

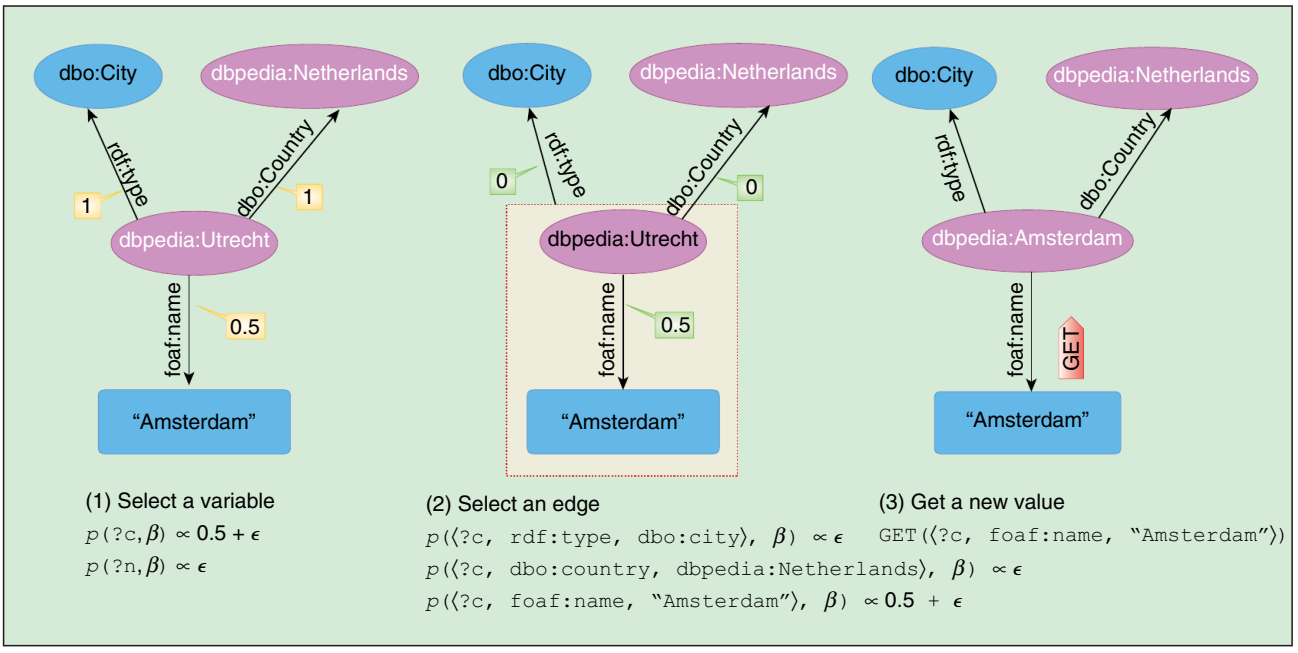


FIGURE 5 The three steps followed by the local search strategy for changing one of the bindings. In this example, the candidate solution {`?c = dbpedia:Utrecht,?n = dbpedia:Netherlands`} is changed into {`?c = dbpedia:Amsterdam,?n = dbpedia:Netherlands`}.

and under a similar time constraint, a standard query engine was found to only be able to solve the simplest queries of the test set.

The development of eRDF is ongoing and new data layers and alternative optimisation strategies are being added and tested as new ideas and use-cases are being found. The main objective of the current investigations is to ensure eRDF performs equally well on both simple and complex queries.

B. Swarm Computing for Logical Entailment

In our second project, we tackled the challenge of distributed Semantic Web entailment with a self-organizing swarm of agents that explore RDF graphs. Our framework is distributed in two ways: the data is stored in distributed dynamic networks of RDF graphs, and the reasoning task is distributed amongst a number of very light-weight agents. In the following, we summarize our work. Interested readers are invited to read [24],[41] and visit http://beast-reasoning.net/ for more details[8].

1) *Semantic Web Entailment:* Logical entailment is defined as the task of finding triples that can be derived based on the semantics of the underlying formalism. We present our reasoning mechanism with the help of RDF(S), but our framework can be extended to OWL 2 RL/RDF rules, or to custom rules that for example add `foaf:knows` relationships between co-authors. The RDF(S) closure of a set of triples T is defined as the materialization of all those entailed triples with respect to RDF(S) model theory [42], and is usually denoted by T^*. It is calculated by applying a set of entailment rules to all triples in the data set until no new triples can be derived. This process can be automated and is (in contrast to our swarm-based alternative) usually performed by an exhaustive search and logical deductions. An RDFS entailment rule consists of two parts: an antecedent with one or two triples as arguments, and a consequent, which is to be added as a new

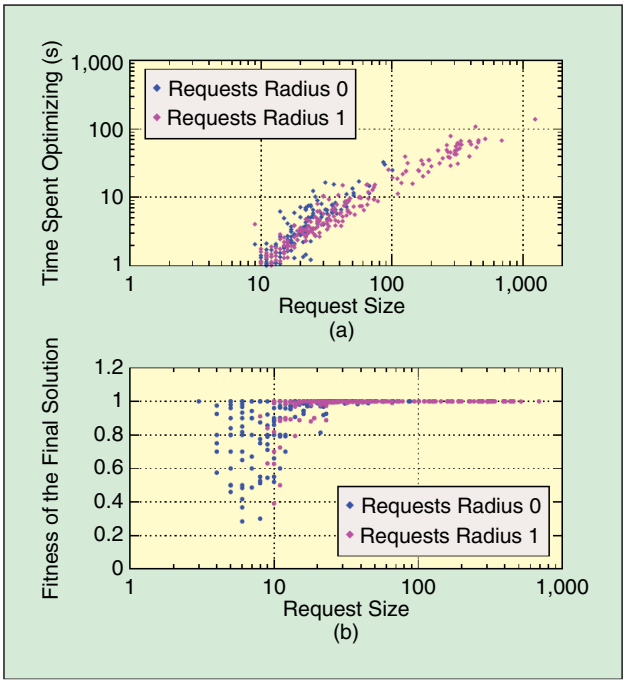


FIGURE 6 Fitness of the first solution streamed back by eRDF and the time the optimization process took to reach it. The results are for requests of different sizes and complexity (radius). (a) Optimization time until a first result is streamed back by the optimizer. (b) Fitness of the first result found, depending on the request size.

[8]Our swarm-based reasoning approach has recently been implemented on top of the fully distributed self-organized semantic storage service "S4" [20]

LISTING 1. Two RDF graphs about publications

```
cg:ISWC08
 pub:title "Anytime Query Answering in
  RDF through Evolutionary Algorithms";
 rdf:type pub:InProceedings;
 pub:author people:Gueret;
 pub:author people:Oren;
 pub:author people:Schlobach;
 pub:cites fvh:SWP.

fvh:SWP
 pub:title "Semantic Web Primer";
 rdf:type pub:Book;
 pub:author people:Antoniou;
 pub:author people:vanHarmelen.
```

triple to the graph. Table 5 lists all RDFS entailment rules with two triples as antecedent (RDFS entailment rules with one triple as antecedent are trivial).

Listing 1 contains two RDF graphs about two publications `cg:ISWC08` and `fvh:SWP`, which are described with the ontology `pub` (for publications) and linked to a `people` data set. On the Semantic Web those two graphs are expected to be administered and served independently by their respective authors.

As shown in Listing 2, the triples in the graphs are enriched with schema information about the ontologies. For example with the information that `pub:InProceedings` is a particular type of `pub:Publication`, the indication that every `people:Person` is also a `people:Agent`, and also that the property `pub:author` can only be used to relate to a `people:Person`.

TABLE 5 Some RDFS entailment rules. These rules are applied to a set of triple to derive and express information which is otherwise implicit.

RULE	IF GRAPH *G* CONTAINS	THEN ADD
rdfs2	*p* `rdfs:domain` *c*. and *s p o*.	*s* `rdf:type` *c*
rdfs3	*p* `rdfs:range` *c*. and *s p o*.	*o* `rdf:type` *c*.
rdfs5	p_1 `rdfs:subPropertyOf` p_2. and p_2 `rdfs:subPropertyOf` p_3.	p_1 `rdfs:subPropertyOf` p_3
rdfs7	p_1 `rdfs:subPropertyOf` p_2. and *s* p_1 *o*.	*s* p_2 *o*.
rdfs9	c_1 `rdfs:subClassOf` c_2. and *s* `rdf:type` c_1.	*s* `rdf:type` c_2
rdfs11	c_1 `rdfs:subClassOf` c_2. and c_2 `rdfs:subClassOf` c_3.	c_1 `rdfs:subClassOf` c_3

LISTING 2. Some RDFS statements

```
pub:InProceedings rdfs:subClassOf
  pub:Publication.
people:Person rdfs:subClassOf
  people:Agent.
pub:author rdfs:range people:Person.
```

Based on the RDFS semantics, it can be derived that `cg:ISWC08` is of type `pub:Publication` (rdfs9), and that all authors are instances of the class `people:Person` (rdfs3), and thus `people:Agent` (rdfs9). The traditional way of calculating these consequences requires the data and the rules to be centralized, and to be treated as if they were one single data set. Given the nature of the Semantic Web as discussed in Section II-B, this assumption is often unrealistic, and undesirable. Instead, we need algorithms that keep the data at their origin and that are able to adapt to dynamic information, with local control.

2) *Reasoning as graph traversal:* The basic idea underlying our approach is that the reasoning task can be decomposed by distributing complementary entailment rules to members of a swarm, so that each individual is responsible for the application of only one rule. Agents thus locally expand the knowledge by traversing RDF graphs and applying their rules on the triples they visit.

Definition 1 (Reasoning as graph traversal): Let T be an RDF graph, N_T the set of all nodes in T and M_T the memory that each agent is associated with. RDF graph traversal reasoning is defined as a triple (t, B_T, M_T), where each $b \in B_T$ is a transition function, referring to a *(reasoning) agent* $rb: M_T \times t \times N_T \rightarrow M_T \times t' \times N_T$ that takes a triple t of the graph as input, moves to an adjacent node, and depending on its memory, adds a new RDF triple t' to the graph.

There are different type of agents, one for each RDF(S) entailment rule. There are two possibilities with regards to their initialization: either the schema is pre-processed and agents are generated for schema triples of corresponding patterns, or agents initialize themselves whenever they encounter a triple of a certain pattern. Let us regard for example rule rdfs3 from Table 5, which defines range restrictions: whenever a triple with the pattern $\langle p,$ `rdfs:range,` $c \rangle$ is encountered, an agent responsible for this piece of schema information is initialized as a function *rb*3 with the associated memory $rb3\{p, c\}$.

Table 6 contains the agents that are needed for RDFS reasoning (in the case where the schema is pre-processed) with their respective inference rules. Reasoning agents *rb*2 and *rb*3 apply the semantics of `rdfs:domain` and `rdfs:range`, while agents *rb*7 and *rb*9 generate the inferences of `rdfs:subPropertyOf` and `rdfs:subClassOf`. They are referred to as domain-agent, range-agent, subproperty-agent and subclass-agent.

TABLE 6 Functioning of reasoning agents.

AGENT	SCHEMA TRIPLE	AGENT MEMORY	IF PATTERN FOUND	THEN ADD
rb2	*p* `rdfs:domain` *c*	rb2 {*p*, *c*}	*s p o*	*s* `rdf:type` *c*
rb3	p `rdfs:range` c	rb3 {*p*, *c*}	*s p o*	*o* `rdf:type` *c*
rb7	p_1 `rdfs:subPropertyOf` p_2	rb7{p_1, p_2}	*s* p_1 *o*	*s* p_2 *o*
rb9	c_1 `rdfs:subClassOf` c_2	rb9{c_1, c_2}	*s* `rdf:type` c_1	*s* `rdf:type` c_2

In our prototype implementation, all schema triples are preprocessed, and the transitive subclass and subproperty closure is calculated before the agents are created. Thus, *rb*5 and *rb*11 are not created. This makes our reasoning safe against ontology hijacking, *i.e.* non-authoritative extensions of ontologies [43]. On the other hand, completeness with respect to the official RDF(S) semantics cannot be guaranteed. The more generic approach (which is shown to converge towards completeness below) is to introduce a subclass-transitivity-agent *rb*11 which searches for connected subclass triples and writes new subclass triples to the graph. These triples can then be used by subclass-agents to update the schema information and to apply it on further triples. The property-transitivity-agent *rb*5 would work accordingly.

a) *Convergence towards the complete closure:* The closure T^* over a data set T contains all triples that follow from the RDF(S) semantics. In our framework, entailment rules are instantiated by the schemata and embodied by agents. Let $b_1, \ldots b_n$ be a swarm with at least one individual per type. The complete closure T^* is derived when the union of the agent outputs $b_1(t_1) \cup \ldots \cup b_n(t_n) \equiv T^*$.

Sketch: to prove that the method converges towards completeness, it has to be shown that all elements of T^* are inferred eventually, *i.e.* that each agent b_m infers the complete closure $b_m(t_m^*)$ of the rule it embodies. Given the agent functions as defined above, an agent infers t_m^* when it visits all triples of the graph that match its inference pattern. This can be achieved by a complete graph traversal, which is possible and can be performed according to different strategies, such as random walk, breadth- or depth-first. It has to be performed repeatedly, as other agents can add relevant triples. T^* is reached when the swarm performed a complete graph traversal without adding a new inferred triples to the graph. Given the possibility of randomly jumping to other nodes within the graph, which also prevents agents from getting stuck in local maxima, the same holds for unconnected (sub-)graphs. When a swarm consists of s members $b_m^1, \ldots b_m^s$ per type, the individuals of one type can infer $b_m(t_m^*)$ collectively.

b) *Example of derivation:* Let us consider the two RDF graphs from our publication example. Figure 7 shows the graph for the first publication. Dashed arrows denote implicit links that are to be derived by reasoning.

We generate one agent per schema triple: a range-agent $rb3_1$ with the memory of `pub:author` and `people:Person` applies the range-triple ⟨`pub:author, rdfs:range, people:Person`⟩, while an agent $rb9_1$ with the memory `people:Person` and `people:Agent` is responsible for the subclass triple ⟨`people:Person, rdfs:subClassOf, people:Agent`⟩. A similar subclass agent is created for the other subclass triples. All agents are distributed randomly over the graph. For example, agent $rb3_1$ starts at node `fvh:SWP` and now has two options. Moving to "SW Primer" leads it to a cul-de-sac, so that it walks back via `cg:ISWC08` towards `cg:Oren`. At node `cg:Oren`, the

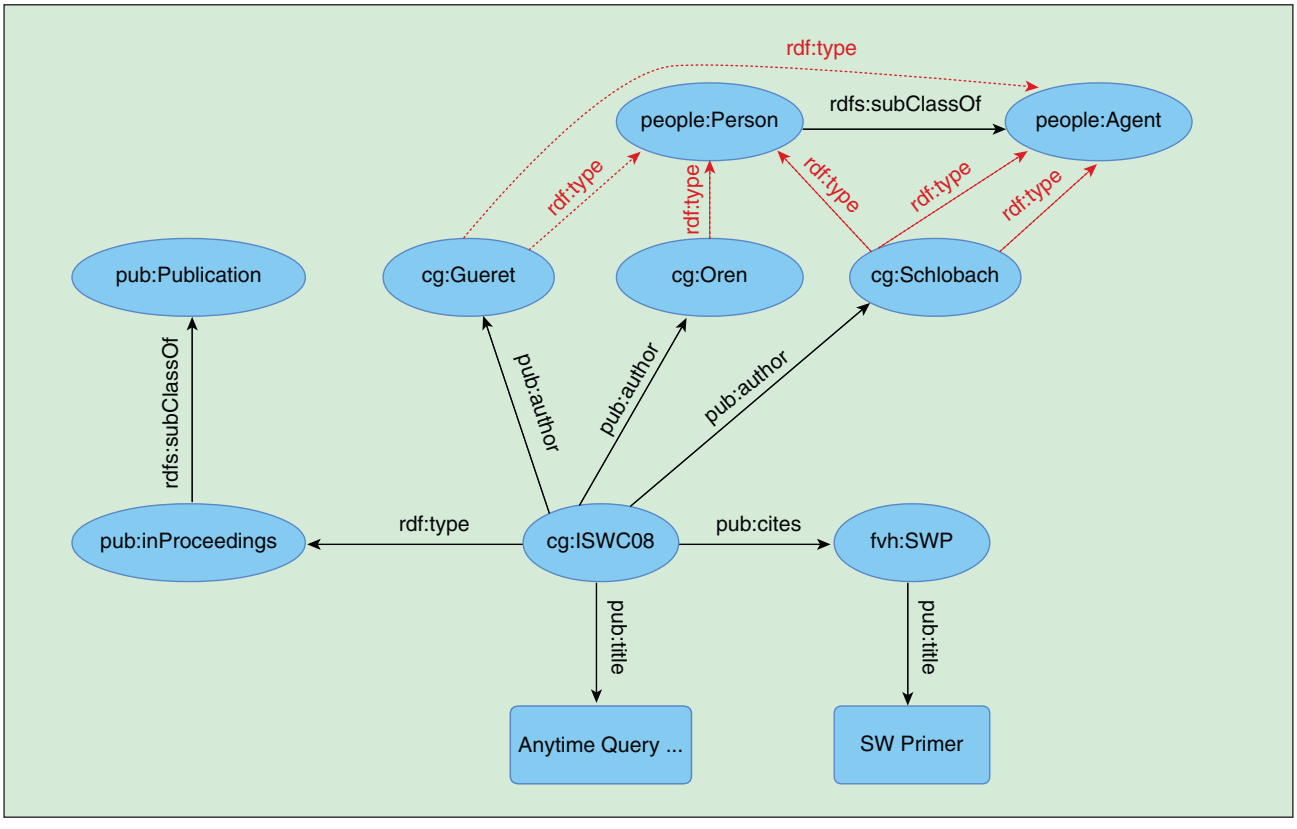


FIGURE 7 An exemplary RDF graph with inferred triples indicated in red.

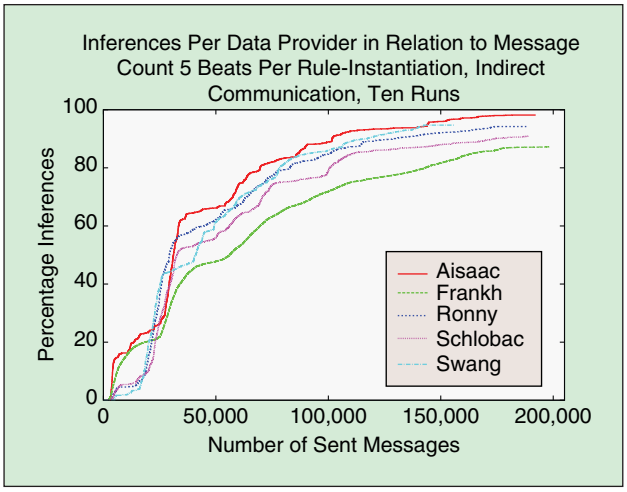


FIGURE 8 The number of inferred triples converge towards closure.

walked path is ⟨`cg:ISWC08, pub:author, cg:Oren`⟩ which means $rb3_1$'s pattern matches the walked triple. It thus adds the triple ⟨`cg:Oren, rdf:type, people:Person`⟩ to the graph. When, after walking other parts of the graph, the subclass agent $rb9_1$ chooses to follow the new `rdf:type` link from `cg:Oren to people:Person`, it finds its memory condition matched, and adds the triple ⟨`cg:Oren, rdf:type, people:Agent`⟩ to the graph, and so forth. This example highlights the challenges faced by the approach: unnecessary paths need to be investigated and the order of visiting agents is important ($rb3_1$ had to be at `cg:Oren` first).

c) *Movement Control:* Our agents operate on an energy metaphor: moving in the graph costs energy and new inferences are rewarding food. This is modeled via a happiness function: the agents are created with an initial happiness value. For every step from RDF node to RDF node made in the graph, the happiness decreases. When an agent finds a new inference, its happiness increases. With this simple mechanism, an agent adapts to its environment: while it is successful and happy, it will probably find even more applications of its rule, whereas an agent that did not find a new inference for a while will either change its rule instantiation, its type, its location or die out of starvation.

In our framework, agents can move to other locations in the network to search for applications of their rules. In our prototype, we apply a simple routing strategy using Bloom filters [44] so that agents are routed to locations that contain elements that they are searching for. Another promising routing strategy would be pheromone based. Distributed graphs can also be bridged by crossing mapping links such as `owl:sameAs`. As per analogy with real ants that find the shortest paths to food sources based on pheromone trails, our swarming agents also leave behind pheromone trails. To choose the next edge to follow, an agent parses all possible ongoing options. If it finds an application of its inference-pattern, it chooses the corresponding path and fires its rule. Otherwise, it categorizes the options into two sets: triples which are promising because they have not been visited before by agents of the same rule instantiation (following other agents is subject to future research) and the triples that already have been visited. If promising options are available, the agent randomly chooses one of them. Otherwise, the ongoing path is chosen with an equation which is inspired by Ant Colony Optimization [45], preferring paths that have been less threaded upon in the past instead of paths that have been frequented.

3) Experiments and conclusion: To prove the concept, we implemented a system based on AgentScape [46], a middle-ware layer that supports large-scale agent systems. It is based on locations, where agents can reside, and active entities that are defined according to the weak notion of agency [47]. Agents can communicate via message passing and migrate from one location to another. In our prototype, each agent is an autonomous agent. Every graph T_i is administered by an agent that is referred to as data provider and linked to other data providers. On each location, there is one data provider which does not migrate. Agents do migrate from location to location and communicate with the local data provider. Reasoning agents migrate to the data, perform local calculations and move on to the next location. Thus, only the agents with their rules and memory are moving in the network. Based on this implementation, we evaluated the feasibility of the approach.

Our experiments have been based on a number of publication files of members of our department. Each of these RDF graphs is a administered by a data provider that resides at a distinct location, e.g. at the computer of the respective colleague, and that is named after the corresponding graph. Based on the employed FOAF and SWRC schemata, we created 195 unique subproperty, subclass, domain and range agents. In the experiments, a swarm that consists of 5 agents per unique rule-instantiation traverses the graph. The swarm operates as described above.

Figure 8 shows how the degree of completeness (*i.e.* the percentage of found inferences in comparison to the output of a standard reasoner) per dataset is rising in relation to the number of sent messages (a message is a request for ongoing options). We can observe a typical anytime behavior: the more messages sent, the more inferences are inferred. In the start-phase, new inferences are found easily, the difficult task is to detect the last remaining inferences. Nevertheless, we can observe that the output converges towards closure.

To conclude, we envisage the Web of Data as an *eternal adaptive anthill* that has decentralized accessible graphs which are constantly reasoned over by restless agents. Because agents can easily deal with added and even deleted triples, we claim that swarm based reasoning is in principle more adaptive and robust than approaches that rely on indices and centralized logical deductions. Also, this reasoning procedure might help in setting up a decentralized publishing model that allows users control over their personal data.

V. Conclusion

The Semantic Web is a complex system made of large-scale, dynamic and potentially incoherent data. State of the art techniques currently employed to deal with this data have not been designed to face such challenges and can not be employed on the actual content of the Semantic Web. Instead, they are applied on curated snapshots of parts of it. In this paper, we discussed the limitations of such an approach and the need to re-consider the basic building blocks of Semantic Web data consumption in light of optimisation problems. In particular, we discussed how the typical tasks of Querying, Storage, Entailment, Consistency checking and Mapping can be rephrased from a logic problem into an optimization problem.

As proven by the work done in this emerging research field, Evolutionary and Swarm computing are particularly suitable solvers for these problems. We sustained this argument by highlighting the current state of the art of the subject and going into details for two approaches we developed to respectively deal with query answering and reasoning. The first addressing the problem of querying with evolutionary computing and the second using swarm computing to compute entailments.

References

[1] M. Dean and G. Schreiber, "Web Ontology Language (OWL): Reference," *W3C Recommendation*, 2004.

[2] D. Fensel and F. v. Harmelen, "Unifying reasoning and search to web scale," *IEEE Internet Comput.*, vol. 11, no. 2, pp. 96–95, 2007.

[3] C. Guéret, S. Wang, P. T. Groth, and S. Schlobach, "Multi-scale analysis of the web of data: A challenge to the complex system's community," *Adv. Complex Syst.*, vol. 14, no. 4, pp. 587–609, 2011.

[4] S. Klarman and V. Gutiérrez-Basulto, "Two-dimensional description logics for context-based semantic interoperability," *in Proc. AAAI*, W. Burgard and D. Roth, Eds. AAAI Press, 2011.

[5] N. Gibbins and N. Shadbolt, *Resource Description Framework (RDF)*, vol. 2004. World Wide Web Consortium, 2009, pp. 4539–4547.

[6] A. Miles and S. Bechhofer, "SKOS," *W3C Recommendation*, 2009.

[7] R. Cyganiak and A. Jentzsch. (2011, Oct.). Linking open data cloud diagram [Online]. Available: http://lod-cloud.net/

[8] J. Bezdek, S. Rajasegarar, M. Moshtaghi, C. Leckie, M. Palaniswami, and T. Havens, "Anomaly detection in environmental monitoring networks [application notes]," *IEEE Comput. Intell. Mag.*, vol. 6, pp. 52–58, May 2011.

[9] F. van Harmelen, A. ten Teije, and H. Wache, "Knowledge engineering rediscovered: Towards reasoning patterns for the semantic web," *in Proc. K-CAP*, Y. Gil and N. F. Noy, Eds. ACM, 2009, pp. 81–88.

[10] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," *W3C Recommendation*, pp. 1–106, Jan. 2008.

[11] J. Broekstra, A. Kampman, and F. Van Harmelen, "Sesame: A generic architecture for storing and querying RDF and RDF schema," *Lect. Notes Comput. Sci.*, vol. 2342, pp. 54–68, 2002.

[12] A. Harth and S. Decker, "Optimized index structures for querying RDF from the web," *in Proc. Web Congr. 2005 LAWEB*, pp. 1–10.

[13] E. Oren, C. Guéret, and S. Schlobach, "Anytime query answering in RDF through evolutionary algorithms," *in Proc. Int. Semantic Web Conf. (ISWC)*. Berlin: Springer-Verlag, 2008, vol. 5318, pp. 98–113.

[14] C. Guéret, E. Oren, S. Schlobach, and M. Schut, "An evolutionary perspective on approximate RDF query answering," *in Proc. Scalable Uncertainty Management*. Berlin: Springer-Verlag, 2008, vol. 5291, pp. 215–228.

[15] C. Guéret, P. Groth, E. Oren, and S. Schlobach. (2010, Oct.). eRDF: A scalable framework for querying the web of data, Vrije Universiteit Amsterdam, Tech. Rep. [Online]. Available: http://dl.dropbox.com/u/2137510/erdf_technicalreport.pdf

[16] H. Stuckenschmidt, R. Vdovjak, G.-J. Houben, and J. Broekstra, "Index structures and algorithms for querying distributed RDF repositories," *in Proc 13th Conf. World Wide Web* (WWW 2004), p. 631.

[17] M. Dorigo and G. D. Caro, "AntNet: Distributed stigmergetic control for communications networks," *J. Artif. Intell. Res.*, vol. 9, pp. 317–365, 1998.

[18] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien, "The dynamics of collective sorting robot-like ants and ant-like robots," in *Proc. 1st Int. Conf. Simulation of Adaptive Behavior on From Animals to Animats*. Cambridge, MA: MIT Press, 1990, pp. 356–363.

[19] R. Tolksdorf and R. Menezes, "Using swarm intelligence in linda systems," *in Proc. 4th Int. Workshop Engineering Societies in the Agents World (ESAW)*. Springer-Verlag, 2003, p. 2004.

[20] H. Mühleisen, A. Augustin, T. Walther, M. Harasic, K. Teymourian, and R. Tolksdorf, "A self-organized semantic storage service," *in Proc. 12th Int. Conf. Information Integration and Web-based Applications and Services*, Paris, France, Nov. 2010, pp. 357–364.

[21] C. Guéret, N. Monmarché, and M. Slimane, "Sharing resources in a p2p network with artificial ants," *J. Math. Model. Algorithms (JMMA)*, vol. 6, pp. 345–360, 2007.

[22] J. Urbani, S. Kotoulas, J. Maassen, F. V. Harmelen, and H. Bal, "Owl reasoning with WebPIE: Calculating the closure of 100 billion triples," *Semantic Web Res. Applicat.*, vol. 6088, pp. 213–227, 2010.

[23] K. Dentler, R. Cornet, A. Ten Teije, and N. De Keizer, "Comparison of reasoners for large ontologies in the OWL 2 EL profile," *Semantic Web J.*, vol. 1, pp. 1–5, 2011.

[24] K. Dentler, C. Guéret, and S. Schlobach, "Semantic web reasoning by swarm intelligence," *in Proc. 5th Int. Workshop Scalable Semantic Web Knowledge Base Systems (SSWS), Collocated at the 8th Int. Semantic Web Conf. (ISWC)*, Oct. 2009.

[25] E. W. Beth, "Semantic entailment and formal derivability," *Proc. Section of Sciences Koninklijke Nederlandse Akademie van Wentenschappen*, vol. 18, no. 13, pp. 309–342, 1955.

[26] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem proving," *Commun. ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[27] F. Lardeux, F. Saubion, and J.-K. Hao, "GASAT: A genetic local search algorithm for the satisfiability problem." *Evol. Comput.*, vol. 14, no. 2, pp. 223–253, 2006.

[28] H. A. Abbass, "A single queen single worker honey bees approach to 3-SAT," *in Proc. Genetic and Evolutionary Computation Conf. (GECCO2001)*.

[29] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Silk—A link discovery framework for the web of data," *Language*, vol. 538, pp. 1–6, 2009.

[30] S. Wang, G. Englebienne, and S. Schlobach, "Learning concept mappings from instance similarity," *in Proc. 7th Int. Conf. The Semantic Web (ISWC '08)*. Berlin: Springer-Verlag, pp. 339–355.

[31] P. Cudré-Mauroux, "Emergent semantics," Ph.D. dissertation, EPFL, Lausanne, 2006.

[32] P. Cudré-Mauroux and K. Aberer, "A necessary condition for semantic interoperability in the large," *in Proc. Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems* (*ODBASE* 2004).

[33] P. Cudré-Mauroux, M. Jost, and H. D. Meer, "idMesh: Graph-based disambiguation of linked data," *Data Manage.*, pp. 591–600, 2009.

[34] J. Wang, Z. Ding, and C. Jiang, "GAOM: Genetic algorithm based ontology matching," in *Proc. 2006 IEEE Asia-Pacific Conf. Services Computing (APSCC'06)*. Washington, DC: IEEE Computer Society, pp. 617–620.

[35] J. Martinez-Gil, E. Alba, and J. F. A. Montes, "Optimizing ontology alignments by using genetic algorithms," in *Proc. Nature Inspired Reasoning for the Semantic Web (NatuReS), (CEUR Workshop*, vol. 419), C. Guéret, P. Hitzler, and S. Schlobach, Eds., Oct. 2008.

[36] J. Bock and J. Hettenhausen, "Discrete particle swarm optimisation for ontology alignment," *Inform. Sci.,* to be published.

[37] S. Wang, G. Englebienne, C. Guéret, S. Schlobach, A. Isaac, and M. Schut, "Similarity features, and their role in concept alignment learning," in *Proc. 4th Int. Conf. Advances in Semantic Processing (SEMAPRO2010)*. IARIA Press.

[38] O. Hartig, C. Bizer, and J.-C. Freytag, "Executing SPARQL Queries over the Web of Linked Data," in *Proc. Informatikhuberlinde* (Lecture Notes in Computer Science, vol. 5823),A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, Eds. Berlin: Springer-Verlag, 2009, pp. 293–309.

[39] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. New York: Springer-Verlag, 2003.

[40] Y.-S. Ong, M. Lim, and X. Chen, "Memetic computation—Past, present & future [research frontier]," *IEEE Comput. Intell. Mag.*, vol. 5, pp. 24–31, May 2010.

[41] K. Dentler. (2009). Semantic web reasoning by swarm intelligence [Online]. Available: http://beast-reasoning.net/thesis.pdf

[42] P. Hayes and B. McBride, "RDF semantics," *W3C Recommendation*, 2004.

[43] A. Hogan, A. Harth, and A. Polleres, "SAOR: Authoritative reasoning for the web," *Semantic Web*, pp. 76–90, 2008.

[44] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[45] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, pp. 28–39, Nov. 2006.

[46] B. J. Overeinder and F. M. T. Brazier, "Scalable middleware environment for agent-based internet applications," *Lect. Notes Comput. Sci.*, vol. 3732, pp. 675–679, 2006.

[47] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowledge Eng. Rev.*, vol. 10, no. 2, pp. 115–152, 1995.

Hannes Mühleisen
Freie Universität Berlin, GERMANY

Kathrin Dentler
Vrije Universiteit Amsterdam, THE NETHERLANDS

Large-Scale Storage and Reasoning for Semantic Data Using Swarms

Digital Object Identifier 10.1109/MCI.2012.2188586
Date of publication: 13 April 2012

1556-603X/12/$31.00©2012IEEE

Abstract–Scalable, adaptive and robust approaches to store and analyze the massive amounts of data expected from Semantic Web applications are needed to bring the Web of Data to its full potential. The solution at hand is to distribute both data and requests onto multiple computers. Apart from storage, the annotation of data with machine-processable semantics is essential for realizing the vision of the Semantic Web. Reasoning on web-scale data faces the same requirements as storage. Swarm-based approaches have been shown to produce near-optimal solutions for hard problems in a completely decentralized way. We propose a novel concept for reasoning within a fully distributed and self-organized storage system that is based on the collective behavior of swarm individuals and does not require any schema replication. We show the general feasibility and efficiency of our approach with a proof-of-concept experiment of storage and reasoning performance. Thereby, we positively answer the research question of whether swarm-based approaches are useful in creating a large-scale distributed storage and reasoning system.

I. Introduction

The success of the Semantic Web leads to ever-growing amounts of data that are being generated, interlinked and consumed. Handling this massive volume is a serious challenge, where scalable, adaptive and robust approaches are needed. Traditional approaches for handling data are often based on large dedicated computer systems which store all required data at one single location and handle all incoming requests from applications and their users. While this is a valid approach for limited amounts of data, it is no longer economically viable for web-scale data due to non-linear increases in hardware costs. Furthermore, robustness of a single system is always limited, making these single-node approaches less suitable for use in the envisioned Web of Data.

The apparent solution is to distribute both data and requests onto multiple computers. In this case, a method to create coherence between these computers is required, designed to make the distributed system appear like a single large unit to its users. Typically, these methods aim to minimize communication costs and to maximize the degree of coherence between nodes. Several methods have been researched and implemented, ranging from master/slave configurations to unstructured and structured Peer-to-Peer systems where all nodes share all responsibilities. Each method represents a trade-off between different dimensions, most commonly scalability, robustness and adaptivity [1]. Scalability is the system's ability to handle increasing amounts of data and requests, robustness is the ability to tolerate failure, and adaptivity is the ability to handle different data characteristics and various request patterns without the need for intervention. Sufficient performance along these three dimensions is required in a storage system for web-scale data.

To realize the vision of the Semantic Web, the annotation of data with machine-processable formal semantics is essential. From these schema annotations, reasoning engines make implicit information explicit, bridging the gap between data and knowledge. When this reasoning process is to be applied to web-scale data, the same requirements regarding scalability emerge, and distributing this task onto many computers is the only viable solution. However, fully distributed reasoning is scarce. Many previous approaches rely on central instances orchestrating the reasoning process, e.g. in [2], which is undesirable since these central nodes are not protected against failure per se, and their failure would inhibit the entire system to perform its reasoning tasks. Alternatively, schema information is replicated in all participating nodes, which is unfeasible in cases where the schema information is very large. Consistency cannot be ensured and any changes to the schema can result in substantial overhead.

Generally, manually configuring and operating large-scale distributed systems that potentially comprise of thousands of nodes is no longer feasible. *Self-organizing* distributed systems are able to operate autonomously [1] and are a promising solution to the challenge of handling distributed systems that provide large-scale storage and analysis for the web of data. The problem addressed by this paper is the design of a method for fully distributed storage and reasoning for Semantic Web data.

One approach to achieve self-organization is the collective behavior of individuals that cooperate in a swarm. The overall goals of the swarm, for example, to find food, are pursued through independent actions of the individuals based on indirect communication methods. From these local actions, a global, intelligent and coordinated behavior emerges [3]. Algorithms inspired by this behavior that aim to imitate the accomplishments of swarms with regard to their self-organization have successfully been applied to solve hard problems such as routing in computer networks [4]. As described above, the challenge of storing, reasoning on and retrieving large-scale semantic data in a distributed setting is a task that can only be sufficiently performed in a truly decentralized way. This makes swarm-based approaches interesting candidates for achieving

the desired self-organization. However, as every distributed system has to trade-off between different goals, these properties come at a cost. Swarm-based approaches trade deterministic guarantees to achieve their advantages. This might make these approaches unfit for use in database-like scenarios where the transactional paradigm has to hold. On the other hand, the extended work on NoSQL storage solutions, which also trade away guarantees in favor of scalability, indicates need for this type of storage [5]. Furthermore, handling web-scale data is a task whose dimensions are yet unclear and it can become necessary to make further compromises. Also, exhaustive results as expected from a classical database make little sense in a web scenario, where the information presented can only be a subset of the available data.

In this paper, we contribute our novel concept for both distributed storage and reasoning on Semantic Web data based on ant-inspired algorithms. We present how the ant's behavior may be adapted for distributed storage and reasoning. We strive to answer our research question of whether swarm-based approaches are useful in creating a large-scale distributed storage and reasoning system for semantic data. We present our concept of such a system in the subsequent Section II, and also contribute a "proof of concept" experiment of the storage and reasoning concepts on a real-world data set in Section III. We discuss related work in Section IV, and conclude this paper in Section V with a discussion of our results showing the feasibility of our approach.

II. Swarm-Based Semantic Storage and Reasoning

Distributed storage, retrieval and reasoning can all be reduced to locating the place where data is to be stored or retrieved to answer queries or to apply reasoning rules. To achieve scalability in these tasks, the location method needs to be as efficient as possible, while maintaining robustness and adaptivity. In the previous chapter, we have argued that self-organization is a method for sufficient performance in all dimensions. However, achieving self-organization with a number of pre-defined algorithms reacting on scenarios is not feasible, as the system will likely face situations not envisioned by its creators. Hence, built-in computational intelligence that is able to adapt itself to new situations is desirable [6].

From the large number of nature-inspired methods that are part of the research in Computational Intelligence, it has been shown that ant foraging is best suited to solve the location problem [7]. In this section, we first introduce basic Semantic Web concepts and technologies. We then describe the brood sorting and foraging methods found in ants and their application to distributed systems. We also show how they can be used to create a distributed storage system with reasoning capabilities. Both ant-based distributed storage and retrieval [8] as well as ant-based reasoning on Semantic Web data [9] have been proposed separately before. We describe how both swarm-based distributed storage and swarm-based reasoning can be combined and extended into a fully distributed and self-organized storage system for Semantic Web data with efficient and fully distributed reasoning. We show how both approaches can benefit from each other, forming a new system capable of performing these tasks with minimal overhead.

TABLE 1 RDFS entailment rules.

RULE	ANTECEDENTS	CONSEQUENT
rdfs2	p *rdfs:domain* c. s p o.	s *rdf:type* c.
rdfs3	p *rdfs:range* c. s p o.	o *rdf:type* c.

A. Semantic Web Concepts and Technologies

Semantic Web research has created the Resource Description Format (RDF) data model. An RDF graph is a directed graph, where the nodes are either URIs, literal values such as strings and integers, or graph-internal identifiers known as blank nodes. Directed and labeled arcs connect these nodes, URIs are also used for these labels. RDF provides a highly generic and flexible data model able to express many other more specific data structures, such as relational or object-oriented data. More formally, let $\mathcal{U}$ be a set of URIs, $\mathcal{L}$ a set of literals and $\mathcal{B}$ a set of blank node identifiers. Any element of the union of these sets $\mathcal{T} = \mathcal{U} \cup \mathcal{L} \cup \mathcal{B}$ is called a RDF *term*. A RDF *triple* is a triple (s, p, o), where $s \in \mathcal{U} \cup \mathcal{B}$, $p \in \mathcal{U}$ and $o \in \mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$. An RDF *graph* is defined as a set of triples [10]. The query language SPARQL has been developed to express complex queries on RDF graphs. For convenient access to the data stored in such a graph, one may use a so-called triple pattern, which may contain variables instead of values. Variables are members of the set $\mathcal{V}$, which is disjoint from $\mathcal{T}$. A *triple pattern* is a member of the set $(\mathcal{T} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{V}) \times (\mathcal{T} \cup \mathcal{V})$. The declarative complex query language SPARQL is based on combining triple patterns and provides many additional features such as ordering and filtering [11].

RDF data (ABox) can be annotated by schema information in the RDF vocabulary description language RDF Schema or the web ontology language OWL (TBox), allowing an automated reasoner to make use of the semantics within an RDF graph. Both ABox and TBox are represented as RDF triples. In many cases (RDF Schema and parts of OWL 2), the semantics expressed by the schema can be calculated using a limited set of pre-defined reasoning or entailment rules. Each reasoning rule consists of two parts: the antecedent(s) and the consequent. The antecedents together specify a graph pattern that is to be matched to an RDF graph, and the consequent specifies how to generate a new triple, i.e. inference. RDF Schema (RDFS) entailment rules have one or two antecedents. If all variables contained in the antecedents are bound to values in the graph, the rule fires and the inference is created. The closure of an RDF graph under the RDFS semantics [10] can be derived by applying all RDFS entailment rules until no new triples are inferred (fixpoint iteration).

As an example, let us consider the RDFS entailment rule for *rdfs:domain* that is shown in Table 1. Whenever an RDF graph contains a triple that states that a given property p has the

rdfs:domain c, and makes use of this property in another triple, it can be inferred that the subject of the other triple is of *rdf:type* c. The entailment rule for *rdfs:range* works accordingly, but refers to the object of a triple that contains the property for which an *rdfs:range* is defined. Let us consider exemplary social network data that makes use of the FOAF vocabulary[1]. Figure 1 shows a limited extract of the corresponding RDF graph. The RDF triple on the bottom states that `Alice foaf:knows Bob`. The two schema triples state that the property `foaf:knows` has both `rdfs:domain` and `rdfs:range foaf:Person`. Based on all triples, it can be inferred that both Alice and Bob are of type `foaf:Person`.

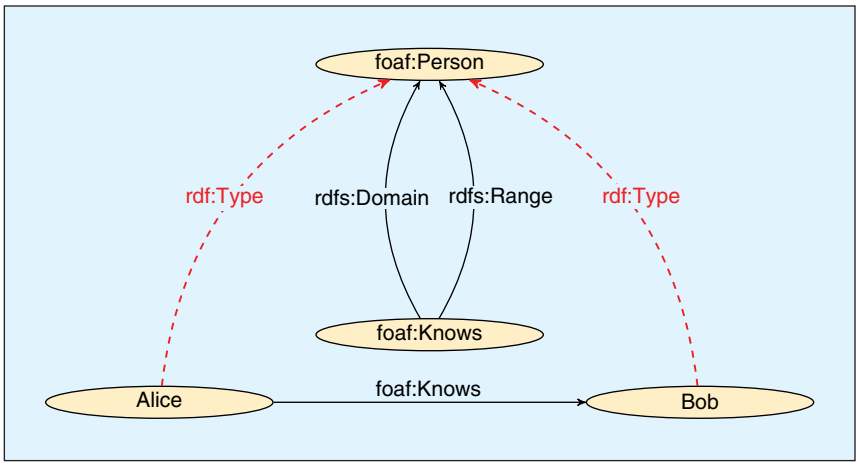


FIGURE 1 An exemplary RDF graph.

B. Swarm Intelligence for Distributed Systems

The basic notion of swarm algorithms is to employ a large number of simple, lightweight individuals. The overall goal of solving the specific problem never depends on single individuals, making those expendable and thus ensuring the robustness of the solution process. In general, each individual only has a *limited view* of its surroundings, e.g. only recognizing their immediate vicinity. Also, individuals only have *limited memory*, forcing them to make decisions based purely on locally available information with the help of *simple rules*. The decision process for a single individual can be described in very few simple behavioral rules. These three principles make swarm algorithms *scalable* with regard to the number of individuals, *robust* to the loss of swarm members and *adaptive* to ever-changing environments. While a single individual has no concept of how to solve the task which the entire swarm is facing, individuals are able to communicate indirectly by changing their environment. A global solution then emerges through the sum of all single independent actions of the individuals.

From a multi-agent perspective, ants are very simple agents which only communicate indirectly via the landscape, which serves as a collective memory. The combination of the independent local actions of the individuals represents a global optimization mechanism based on positive feedback.

The first method that is useful to solve the location problem in a distributed system is the foraging method by which ants search for food in colonies consisting of thousands of individuals. A subgroup of the ants is assigned to search for food and bring it back to the nest. They do so by leaving their nest, randomly changing directions at first. As soon as they encounter food, they carry it back to the nest. On the way back, they leave a chemical pheromone trail behind. Now, other ants who are also searching for food can smell this trail, which leads them to follow it with increasing probability for increasing pheromone intensity. This way, the pheromone intensity of paths to rich food sources is being reinforced, as more and more ants use this path. Ants leaving the nest now no longer have to wander around randomly, but can choose between established paths. Pheromones evaporate over time, so that paths to depleted food sources will disappear [12].

The second ant-inspired method that can be useful for data organization is the method by which ants sort their brood. To optimize care for their brood, ants cluster larvae according to their development stage. To solve this clustering problem, ants developed a fully decentralized method where the ants inspect the larvae nearby and then pick up the most dissimilar one. If they are carrying around a larva, they are inclined to drop it where similar larvae are placed.

The advantage of both methods from a distributed system perspective is that they do not require any shared global data structure, which would have to be reliably maintained at considerable cost. Rather, these methods can be applied in a fully distributed system and still maintain a high degree of efficiency [4]. The tradeoff for this high performance with low requirements is a small degree of failure probability, for which a recovery method has to be designed.

C. Storage and Retrieval

Typically, a distributed storage system consists of a large number of fully independent computers, which are connected by a network. The general goal for this set of computers (nodes) is to provide one single storage service, with data being distributed across all nodes. Since a central gateway to the service provided by the computer network would be a single point of failure, each node should be able to serve requests for all data that is being handled within the system. Storing and retrieving data items is now a problem of locating the subset of nodes within the network which are responsible to store the information. No single node can have all knowledge that is required, since this would be both a bottleneck and Achilles' heel for the system. The problem thus has to be solved through

[1]FOAF: http://xmlns.com/foaf/spec/

cooperation of multiple nodes based on partial information that is locally available. The smaller the subset of involved nodes, the higher the scalability, and typically a logarithmic proportion of the total number of nodes is considered sufficient. We have already proposed the application of ant foraging to solve this problem [8].

The swarm-inspired algorithms are adapted to our distributed system as follows: The internal storage operations are considered the swarm's individuals moving around on a virtual landscape of nodes connected using network technology. Every node is connected to a limited number of other nodes that are its so-called neighbors. The average number of neighbors is equivalent to the degree of the overlay network topology. The data to be stored inside this network is modeled as the ant's food or larvae, respectively. From a Peer-to-Peer perspective, this approach represents a compromise between unstructured and structured networks. Structured, because the pheromone trails represent a shared data structure dramatically improving routing efficiency, and unstructured, because a node's position in the network is dynamic and its routing decisions are not determined by a global law, but rather on a best-effort local heuristic exploiting purely local information.

The ants' brood sorting method is used for write operations. We calculate a so-called numerical routing key for each data item based on a similarity measure. In the general case, the similarity measure is a hash function on the literal key value, e.g. SHA1. Similarity between two items is their numerical distance of the routing key values. Depending on the notion of the distance to be used for the stored data, other similarity measures such as numeric similarity can be used [13].

The routing key then enables the individuals to find an area of the storage network where similar items are stored and where new data items are thus placed. In the case of RDF, graphs to be stored are deconstructed into RDF triples and each triple becomes part of three separate write operations, each time with another triple component (subject, predicate, object) as a routing key. The write operations then move from node to node until they find a number of triples sufficiently similar to the triple that is to be stored, and then they will store it. This leads to clusters of triples that use the same or similar routing keys being placed on the same or neighboring nodes, generating a global degree of organization in the storage network. The details of this process are described in [8]. Since routing is not based on a global law but rather on individual decisions at each storage node, uneven data or request load of single nodes the storage network is unproblematic. Excess data can be moved off to neighboring nodes, and corresponding requests will first be forwarded, with the routing heuristic updating itself to reflect the new location soon [14].

If a node receives a request for information, it creates an internal read operation which is also able to move from node to node similar to the foraging method, thereby regarding RDF triples as food. Triples can be searched for by specifying both fixed values and variables for any triple entry. Thus, any basic graph pattern can be evaluated as a search pattern that is to be matched against the triples. Again using the similarity measure, the read operation is able to find the part of the network where the cluster containing the particular data item is stored, thereby exploiting the locality created by the brood sorting. Successful operations return their result to the waiting application at the node where the query originated and trace back the path they have taken. They use the calculated routing key to intensify virtual pheromones maintained for each connection to another node on the path taken. These pheromones are distinct for each key that has been used to store a data item, the added intensity is dependent on the size of the result set and the path length. While the operation has not yet arrived inside the cluster where the searched triple is located, the routing only has to find this cluster. Hence, pheromone values can also be compressed through aggregation into ranges, effectively limiting the space needed to maintain the pheromones.

As in nature, subsequent operations can read these pheromones and calculate the likelihood of finding results by traversing that particular connection and pheromone values decrease over time to simulate evaporation with a configurable decay rate. Should pheromone values be absent or ambiguous, a random node is chosen as a next hop, with the randomness decreasing as the pheromones get more intense.

The repeated process of requesting different data items from different locations will create a multi-layered network of pheromone paths leading from the nodes where requests were received to the nodes where data was received. The self-optimizing property of the swarm method used will lead to a near-optimal path through the network, as shorter paths are assigned stronger pheromone intensities. The optimal length of the paths is dependent on the properties of the overlay network, which is dependent on the amount of neighbor nodes each node has. Hence, the retrieval costs for often-requested large clusters is close to the shortest path between the requesting node and the node storing the data item. For unpopular and small clusters, the retrieval process can degrade to a random walk, which can incur costs linear to the amount of nodes in the network once. Since it is also possible that a retrieval operation starts a circular movement pattern, its number of steps between nodes is limited by configuration. Should the operation reach the maximum number of steps allowed, it fails and reports back to the node it originated from. It is then able to restart the retrieval operation.

The retrieval process is shown in Figure 2. Here, a request for a triple with the key #B is received at node S2. From the pheromones present for this key, the likelihood for finding matching triples on the connected nodes S1, S5 and S6 can be calculated. According to this probability distribution, a weighted random routing decision is taken, in this case most likely routing the request to node S1, from where it is again likely that the operation will find the searched data item on node S3. However, should this not be the case, every other node is also able to calculate these probabilities, thereby achieving an increasing probability that the operation will find the data item with further hops.

Evaluating complex queries inside this system is still another area of ongoing work [15], but in a first step we have implemented the storage interface used by a general-purpose SPARQL processing engine. The static optimizations used by this engine create a sequence of retrieval operations, which are then executed as series of single retrieval operations for each triple pattern in the query. The results of all retrieval operations are collected and added to a temporary graph, on which the complex query can then be evaluated. While effective, efficiency of this approach can be impaired by large intermediate result sets.

D. Reasoning

Two approaches to infer implicit knowledge are forward and backward chaining of reasoning rules. The idea behind forward chaining is to derive and optionally materialize all possible inferences based on input data, typically when new statements are inserted into a triple store. In contrast, backward chaining of reasoning rules is usually performed during query processing, applying rules which lead to the results that are being queried for in reverse order. Backward chaining thus requires less storage at the cost of longer query processing times. In our usage scenario, we have large amounts of space available to store inferred information and aim at fast query processing. Thus, we focus on forward chaining only.

We previously presented a distributed, forward chaining reasoning method based on swarm intelligence [9]. There, individuals of a self-organizing swarm "walk" on the triples of an RDF graph, aiming to instantiate pattern-based inference rules. To apply this approach to the self-organized semantic storage service, we adapt the members of the swarm so that they no longer employ pheromones to traverse the RDF graph structure, but to find nodes with triples that they can apply their inference rules on. Our storage layer does not differentiate between data (ABox) and schema information (TBox), which are both encoded as RDF triples. Hence, we assume all schema information to be part of the entire set of triples that is stored in the storage network. The forward chaining of reasoning rules can be applied by swarm individuals, assigning each reasoning rule to a number of individuals, who subsequently traverse the network trying to find matches for the antecedents and creating inferences whenever the rule fires. Contrary to the previous approach, we are able to re-use the pheromone trails left behind by the storage operations to efficiently locate the triples required to calculate inferences.

By relying on a storage layer based on a similar concept, the proposed reasoning method requires no additional distributed data structures in the network nor on the nodes and still is able to express increased efficiency. Regarding the expressivity of this approach, methods for sound and complete distributed resolution on the Description Logic $\mathcal{ALC}$ have been proposed [16]. Supporting the same expressiveness in a swarm-based system has also been shown to be feasible together with a formal discussion of the theoretical correctness of the approach [17].

In our system, each node periodically scans the locally stored triples for values that are contained in the triple patterns of the antecedents of the pre-configured rules. Each match becomes a new reasoning operation, which is initialized with the now partially bound triple pattern or basic graph pattern. The reasoning operation now tries to find triples that complete the remaining triple patterns that belong to the basic graph pattern by moving through the network using the previously bound values as routing keys. To find potential matches, the pheromone paths are exploited to efficiently route them to their next destination, even if reaching the destination node requires several hops. Once all variables are bound, the rule fires and the inferred triples are written to the storage network using the write operation. This has the advantage that triples which are already present in the store are not added again, so that duplicates resulting from the reasoning process, for example due to several rules that lead to the same inference, are not added multiple times.

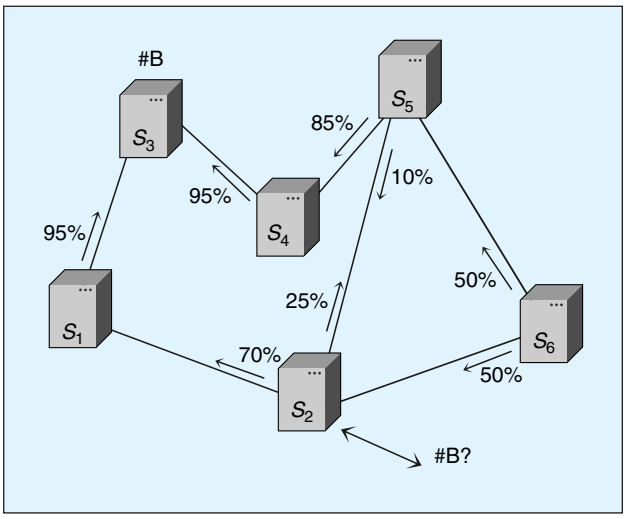


FIGURE 2 Network structure with routing probabilities.

We will demonstrate how the reasoning process is executed using the RDF Schema entailment rules [10]. As RDFS entailment rules with only one antecedent are trivial, we will focus only on rules with two antecedents. Triples matching the antecedents could be stored on different nodes, which requires data exchange between those nodes in order to fire the reasoning rule. This subset of the RDF Schema specification contains `rdfs:domain` and `rdfs:range` entailments, the transitive closure and the implications of `rdfs:subPropertyOf` and `rdfs:subClassOf`. All considered rules contain at least one schema triple, that is antecedents that contain an element from the RDFS namespace. Table 2 shows the different phases of our approach for these inference rules. First, the node-local store is searched in the **init** phase for triples that can be matched to antecedents that are schema triples, and new reasoning operations are generated and initialized based on the found triples. In the **move** phase, these reasoning operations now search for matching triples, first locally and then traversing the storage network. For our employed RDFS rules, one (rdfs2, rdfs3, rdfs7 and rdfs9) or two elements (rdfs5 and rdfs11) need

TABLE 2 Application of RDFS entailment rules.

RULE	INIT	MOVE	INFERENCE
rdfs2	**p** *rdfs:domain* c.	S **p** O.	s *rdf:type* c.
rdfs3	**p** *rdfs:range* c.	S **p** O.	o *rdf:type* c.
rdfs5	P_1 ***rdfs:subPropertyOf*** $\mathbf{P_2}$	$\mathbf{P_2}$ ***rdfs:subPropertyOf*** p_3.	p_1 *rdfs:subPropertyOf* p_3.
rdfs7	$\mathbf{P_1}$ *rdfs:subPropertyOf* p_2.	S $\mathbf{P_1}$ O.	s $_2$ o. p
rdfs9	$\mathbf{c_1}$ *rdfs:subClassOf* c_2.	s *rdf:type* $\mathbf{c_1}$.	s *rdf:type* c_2.
rdfs11	c_1 ***rdfs:subClassOf*** $\mathbf{c_2}$.	$\mathbf{c_2}$ ***rdfs:subClassOf*** c_3.	c_1 *rdfs:subClassOf* c_3.

to match; they are printed in bold and underlined in the table. If a reasoning operation encounters a match, it creates an **inference.**

Regarding our foaf example from Section II-A, in the **init** phase, two reasoning operations are generated: one for the domain axiom and one for the range axiom. In the **move** phase, these reasoning operations search for triples that contain the necessary elements for their inference rules to be applied. Both the `domain` and the `range` reasoning operations now search for occurrences of the predicate `foaf:knows`. Let us assume that data using this predicate does not occur at the local node, so that the reasoning operations migrate to other nodes searching for it. Whenever a match is found, the reasoning operation creates a newly derived triple. For example, the reasoning operation responsible for the domain axiom finds a triple `Alice foaf:knows Bob` and **infers** that the subject of this triple, i.e. `Alice` is of type `foaf:Person`.

ALGORITHM 1 Reasoning Operation—Initialization Phase.

Require: Set of reasoning rules s_r

1: **for all** $r \in s_r$ **do**
2: $p \leftarrow$ *init Pattern*(*antecedent*(r))
3: $s_t \leftarrow$ *localRead*(p)
4: **for all** $t \in s_t$ **do**
5: $p_b \leftarrow$ *bind*(p,t)
6: $r_b \leftarrow r \setminus p \cup p_b$
7: *createInferencingAnt*(r_b)
8: **end for**
9: **end for**

ALGORITHM 2 Reasoning Operation—Move Phase.

Require: Partially bound reasoning rule r_b
Require: Hop limit h_{max}
1: $h \leftarrow 0$
2: $a \leftarrow$ *antecedent*(r_b)
3: $p \leftarrow$ *movePattern*(a)
4: **if** *allBound*(a) **then**
5: **return** *bind*(*consequent*(r_b),a)
6: **end if**
7: **while** $h < h_{max}$ **do**
8: $s_t \leftarrow$ *localRead*(p)
9: **for all** $t \in s_t$ **do**
10: $p_b \leftarrow$ *bind*(p,t)
11: $r_b \leftarrow r \setminus p \cup p_b$
12: *createInferencingAnt*(r_b)
13: **end for**
14: *move*(*routingKey*(p))
15: $h \leftarrow h + 1$
16: **end while**

A more formal definition of the reasoning process is given as pseudocode. The process consists of two phases: First, the initialization phase scans the local storage on every node for ABox (schema) triples, and instantiates reasoning operations that move through the network. This process is given in Algorithm 1. For a set of reasoning rules, the local storage is scanned for triples matching the init pattern of the reasoning rule (Line 3), which is part of the configured rules given in Table 2. For each matching triple, the found values are bound to the pattern, and a new reasoning ant is spawned with this pattern and sent on its way (Line 7).

The process for handling these reasoning ants on every node they visit is given in Algorithm 2. Here, the local storage is checked for triples matching the move pattern (Line 8). For every match on the current node, the match is bound to the antecedent pattern, and a child reasoning ant is spawned (Line 12). If no matches are found, the reasoning ant is routed to the node storing matching data using a routing key from the move pattern. Both move pattern and routing key are also pre-defined in the current reasoning rule set. If the antecedent pattern is fully bound, the resulting triples can be created using the bound values from the antecedent and the static values from the consequent (Line 5). The new inferred triple can now be written using the write operation described above.

Our approach has several advantages: 1) it is fully decentralized, since every node that happens to store a part of any compatible schema will create the corresponding reasoning operations, which do not need to be controlled in any way; 2) it does not require the replication of schema information; 3) it shows "anytime" behavior, generating sound inferences that can be queried for during the reasoning process, with the degree of completeness increasing over time. New triples can be added to the store at any time, leading to new inferences. When triples are deleted, we currently cannot trace and delete the inferences

that they caused, thus only monotonic logics are supported. This problem could be solved by adding a time stamp to each new inference, so that triples that are not re-inferred can be deleted after a while.

The advantages of our approach are gained by relinquishing completeness guarantees. By relying on the pheromone trails also used by the storage operations, the reasoning operation cannot guarantee that a part of a basic graph pattern that may be present somewhere in the network is actually found. However, at web-scale, incomplete reasoning methods have been found to be advantageous due to the gained robustness and scalability, and partial reasoning results are often useful as well [18, 19]. Furthermore, triples that are frequently requested have stronger pheromone paths leading to them, enabling the reasoning operation to find these triples more reliably. This leads to inferences on heavily-used data being calculated more quickly than others, while still maintaining the full adaptivity and robustness of our approach.

E. Stochastic Scalability Analysis

To determine the theoretical performance of routing heuristics, we will now perform a stochastic analysis of our operations. To this matter, we describe the average case performance of the retrieval operation. Consistent with distributed systems research, the unit of cost for this analysis will be hops, that is the amount of transitions of the operation between nodes [20], [21]. Typically, a logarithmic relationship between the amount of nodes in the system and the average hops required to find a data item is required for scalability.

Since a request can be started at every node and results can be on any node in the network, the cost for any retrieval operation is at least the distance between the nodes in the network. In the average case, this distance is the average path length in the network. Disregarding the possibility of the network graph having small-world or scale-free properties, we assume the average path length in random networks as our average distance from start to target node. The average path length in a random network l_{ER} (and also the average distance between nodes) is calculated as follows [22]:

$$l_{ER}(N, \langle k \rangle) = \frac{\ln N - \gamma}{\ln \langle k \rangle} + \frac{1}{2}$$

with N being the number of nodes, γ being the Euler-Mascheroni constant (≈ 0.5772) and $\langle k \rangle$ being the average connectivity in the network (equivalent to the average number of neighbor nodes).

Since our routing method is based on positive feedback, we can assume p_f to be in the range $[0, 0.5]$. For every step on the way from the origin to the destination node, three outcomes of the heuristic-supported routing process are possible: Positive, where the operation got one step closer to its destination, Neutral, where the amount of steps remaining is unchanged, and Negative, where the operation now is one step further away from the destination. Since network connections are defined to be bidirectional, a step in the wrong direction can add at most one additional step to the remaining path length. However, the distribution between neutral and negative outcome is unknown, we therefore introduce a second parameter, p_n. The probabilities for each case are thus as follows:

- $p(positive) = 1 - p_f$
- $p(neutral) = p_f * (1 - p_n)$
- $p(negative) = p_f * p_n$

For a single step in the network, the total impact i on the remaining path length is thus calculated as $i = -(1 - p_f) + (p_f * p_n)$. If the assumption of p_f being at most 0.5 holds, we can see that p_n has to be 1 in order for the improvement i to evaluate to 0. However, it is unlikely that every mistake adds another step to the operation's path, and hence we can safely assume p_n being smaller than 1. If this assumption holds, the improvement i is always negative. Consequently, every routing operation will bring the operation closer to its destination.

The expected value for the average hop count to retrieve an arbitrary element from the network is then the fraction of the average path length by the absolute value of the expected reduction of the remaining path length per hop.

$$\text{hops}(N, \langle k \rangle, p_f, p_n) = \left\lceil \frac{l_{ER}(N, \langle k \rangle)}{|-(1 - p_f) + (p_f * p_n)|} \right\rceil.$$

For example, in a network of 10,000 nodes with an average amount of 10 neighbors, the average path length inside the network is 10. If we assume a high routing error probability of 40%, and a realistic 50/50 distribution between neutral and negative cases, we expect on average 15 hops to reach the node where matching data is stored, or nine more than required from the network structure.

To come back to our stochastic analysis of the average amount of hops required to route any request from the node it was created on to the node storing matching data, we have shown the average amount of hops required to perform this task within our swarm-based system. As we have seen, the average amount of hops is dominated by the average path length, since the probabilities are independent of the network size. Hence, the overhead produced by our swarm-based approach is constant with regard to the network size, and thus our approach can be considered scalable.

F. Network Management

To create the overlay network, we propose a distributed network bootstrap protocol: New nodes are given the address of a "bootstrap" node that is already part of the network. The new node can now retrieve a list of neighbor nodes from the bootstrap node and request its addition to this list. This request is granted if the bootstrap node has not reached its neighbor upper limit as per its configuration yet. This process is then recursively repeated on the newly known nodes until the number of neighbors on the new node has reached the neighbor lower limit, also defined in the node configuration. If the number of bootstrap nodes is limited, this algorithm uses preferential

attachment to create a power-law network structure [23]. Nodes maintain connections to their neighbor nodes, and nodes not responding are removed. If the number of neighbors should fall below the lower limit, the bootstrap process is resumed.

G. Summary

We have designed the operations within the proposed distributed system according to behavior found in ants for foraging and brood sorting. These behavioral descriptions adhere to the general swarm properties with landscape-coordinated actions of large numbers of individuals with limited view. Through the separation of the virtual landscape onto many nodes with a individually managed limited data structure, we have removed global state from the network, which is a major hindrance for scalability, while maintaining statistical efficiency. By design, storage, retrieval and reasoning operations as described above also do not require global state, and every information they require can be calculated on the node the operation is currently executed on. Therefore, the design dimensions for distribution – scalability, robustness and adaptivity are met for storage, retrieval and reasoning with data in the RDF model.

Scalability to the amount of data that can be stored in the network only depends on the sum of storage available on the individual nodes. Should this space become the limiting factor, new nodes can be added using our bootstrap protocol without affecting the entire network. As soon as the new node finishes the bootstrap protocol, the system-inherent randomness will lead to operations being routed to this node. As soon as data is placed on the new node, subsequent retrieval operations will create the pheromone paths leading to this data. This makes this new data efficiently available for retrieval as well as for reasoning operations.

Should individual nodes fail, new operations cannot be routed to these nodes anymore, and operations will have to be routed to the neighbor node with the second-strongest pheromone path from the neighbor list. Again, this does not affect the other nodes in the network, and after a limited time, operations would have created new paths "around" the unresponsive node, expressing the sought-after robustness of the proposed system. To keep the data formerly stored on this node available, a purely local replication scheme may be used.

Finally, adaptivity to skewed data is also possible: For example, if the distribution of routing keys is very uneven in the stored data, the data for this key may likely exceed the storage capacity of a single node. In this case, this node can individually decide to move a portion of the data to neighboring nodes. Pheromone paths will adapt to this new distribution in this area of the network, again keeping all reconfiguration in a very limited area of the network. Furthermore, pheromone paths will become stronger for much sought-after data. Operations requesting this data will exhibit a higher efficiency than operations for less popular data, leading to the designed statistical efficiency of the system.

The trade-off for these characteristics is the potential for failure, which can lead to failed retrieval operations, misplaced data items, and missed inferences. However, retrieval operations can always be restarted until the data is found and misplaced data items can be moved as shown through internal cleanup operations. Reasoning events are repeated periodically, eventually finding most possible inferences. Added operations will create additional load, making the performance of the entire system dependent on its capability to handle large amounts of operations. Should this become a bottleneck, new nodes could be added.

In spite of the conceptional fitness, the emergence of coordinated collective efficient behavior as aspired by their application cannot be proven but only shown in experiments. We present such experiments in the following section.

III. Experimental Results

To test our concept in a preliminary experiment of large-scale storage operations and reasoning in a distributed setting based on swarm intelligence, we have chosen a series of black-box tests in an experimental setup closely resembling the environment where the designed system is to operate. This method was chosen due to the properties of the employed ant algorithms with their inherent randomness, which makes results from simulations difficult to transfer. We have thus implemented our concept as a stand-alone software program, which was then run on a number of independent computing nodes rented from Amazon's Elastic Computing Cloud (EC2). Using our bootstrap protocol, the nodes created a network of connections among them, ensuring that each node is able to reach any other node in the network over at least one path.

For test data, we have used a subset[2] of a crawl from the Web of Data created for the VisiNav system [24] containing a large number of resources annotated using the Friend of a Friend (FOAF) vocabulary[3]. This subset containing ca. 75K triples was written using the storage operation described in the above section, effectively distributing as well as clustering the data over the nodes participating in the storage network. The FOAF vocabulary encoded in RDFS was also written to the network, so that the reasoning ants could be created by the system. This data set was chosen because it contains several characteristics we have identified to be problematic for other—more formal—approaches: First, it contains live web data, which are unchecked and messy, and which can bring logic-grounded reasoners without special optimizations to their limits [25]. Second, since the data is collected from several sources, the distribution of terms is unknown a-priori and potentially skewed [26], making it impossible to configure a conventional large-scale storage system beforehand. Third, a large amount of instance data is using a very small number of classes as defined by the schema, challenging approaches which move all potential matches for inference rules to the node storing the rule.

The relatively small size of the data set was deliberate to allow a large number of repetitions of the experiment, as the swarm algorithms always exhibit a degree of randomness, and multiple repetitions have to be performed in order to create a statistically significant result. As shown in the previous section,

[2]Test data set available at: http://beast-reasoning.net/a.nt
[3]FOAF vocabulary specification: http://xmlns.com/foaf/spec

the potential amount of data that can be handled is directly dependent on the number and storage capacity of the nodes and the throughput of the network connections between them. Hence, a larger data set would not yield additional insight into the system's behavior.

A. Storage and Retrieval

As described above, the main challenge for storage and retrieval operations in a distributed system is to find the node where a data item should be placed or searched for while involving as few nodes as possible. Rather than focusing on evaluating complex queries as described, we issued a retrieval request for a single arbitrary but fixed triple already stored inside the network to every node participating in the storage network. For each retrieval operation, the nodes taking part in the location of the triple were recorded. From the total number of nodes, the number of nodes not able to produce the triple were used to calculate a response success percentage. We have repeated this experiment over network sizes ranging from 20 to 150 nodes.

The results for one of these experiments are given in Figure 3. For the different network sizes and all queried nodes, the average and median number of hops required to find a single triple are plotted. These average values clearly show the average number of nodes to be far less than the number of nodes in the network. The variation in the values between network sizes are attributed to the employed randomness. Also, the response success percentage is 100% almost every time, confirming our expectations for the storage performance of our swarm-based approach. However, for this experiment, the correlation between network size and hops required was not linear. We suspect this to be due to randomness inherent in the swarm algorithms.

Hence, we have repeated the experiments shown in Figure 3 ten times in an effort to sufficiently remove the effects of randomness. Figure 4 shows these results. From the fitted curve, we can observe a linear increase at worst in the number of hops required to retrieve a triple over the network sizes. To determine the overhead created by the swarm-based approach, we have also statically analyzed the network structure created by the bootstrap algorithm between the nodes. We have found that the average path length inside these networks ranged between 1.5 for 20 nodes and 2.5 for 150 nodes. The maximum path length ranged from 3 for 20 nodes to 4 for 150 nodes. An optimal algorithm solving the location problem, for example, enjoying a global view on the network, would have been able to retrieve the data items using these optimal values for the number of hops. Hence, we are able to determine the overhead created using our approach in this experiment to be on average approximately two times the hops required by the "perfect" algorithm. Furthermore, previous research on foraging-based distributed systems has shown that the hop count required scales logarithmically with the network size in simulations [4]. Even though the number of nodes in this experiment was insufficient to prove this behavior for our approach in our limited testbed, a general trend towards logarithmic behavior is visible.

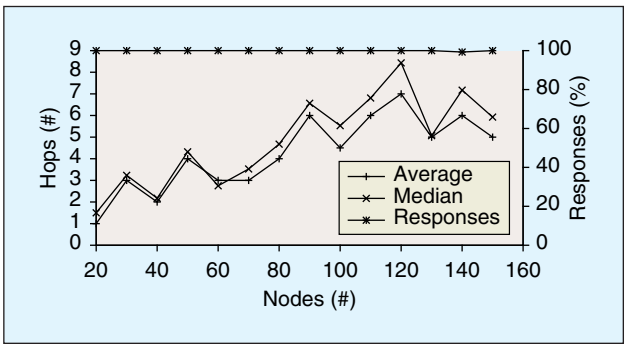


FIGURE 3 Hops required to find a triple for different network sizes.

B. Reasoning

The swarm-based method to perform basic RDFS reasoning as presented in the previous section was evaluated using a different method. From our test data set and the corresponding schema, we have calculated the RDFS closure using a conventional reasoner, which was able to calculate the closure after some problematic statements have been removed. The closure contained approximately the same number of triples as the original data set. Over 87% of the generated statements connected two resources with the `rdf:type` property, since those are most commonly generated according to the employed RDFS inference rules. In our data set, the number of `rdf:type` statements went from 10,173 to 76,734 statements after inferencing. We have focused on `rdf:type` statements in our distributed case, since discerning between static and inferred triples is non-trivial, as they are located in the same storage layer. The comparison of the number of these statements between the data generated by the reasoner and the data generated by the distributed process will yield the degree of completeness achieved by our reasoning process. To this end, each node was extended with a method to allow it to be queried for the number of those statements.

For a test protocol, we completed the process of writing the data set and the schema to the network, which we have limited to 50 nodes for the reasoning experiments. The previous experiment has shown that the number of nodes in the network is not the decisive factor in the system's performance. Then, the

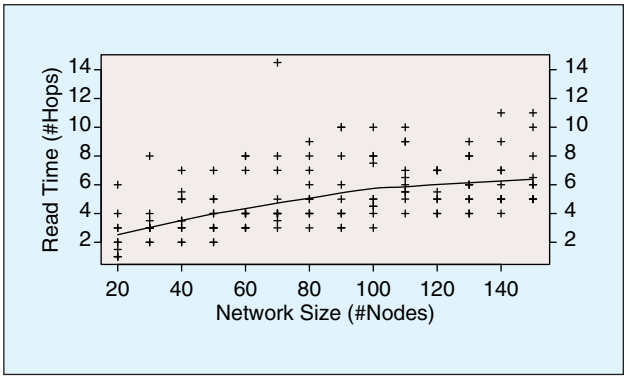


FIGURE 4 Hops required to find a triple for different network sizes—ten test runs.

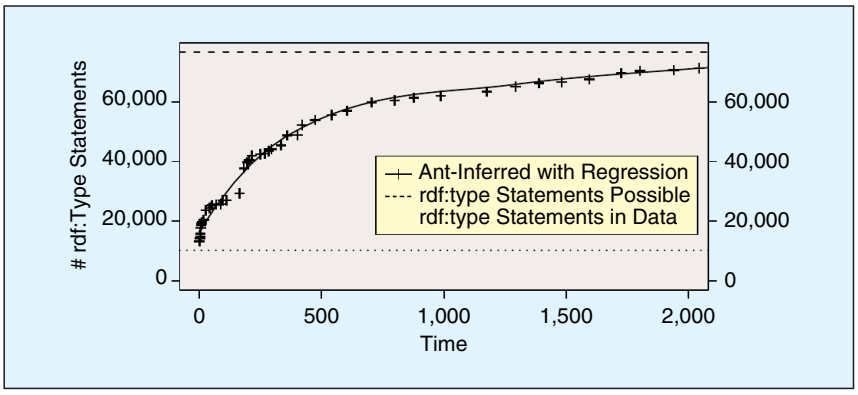


FIGURE 5 Inferred statements over time on 50 nodes.

reasoning process as described was started on each node, generating the corresponding reasoning operations. During this phase, we periodically measured the total number of `rdf:type` statements. Even though the basic scale of the experiment is time, the actual time values are not relevant, since they are highly dependent on the implementation of the system.

Figure 5 plots the results of a single test run, with two lines marking the number of measured statements already in the data set as a baseline, as well as the number of measured statements in the previously calculated full closure of the data set. The plot commences directly after the reasoning operations have been started, showing a discrete measurement along a time scale along with a regression line. The shape of the graph shows the saturation process typical for swarm-based reasoning [9]. We have repeated this experiment several times to remove singular influences by the system-inherent randomness. An aggregation of all experiments is plotted in Figure 6, converging on the same result as shown before, with the measurements exceeding the theoretical limit of possible inferences due to temporarily misplaced duplicates.

IV. Related Work

Our survey of related work is focused on distributed systems that either provide storage and retrieval on RDF data with reasoning capabilities or systems that support distributed reasoning as their only service.

Battré et al. [27] describe a method to perform distributed RDFS forward-chaining reasoning in their BabelPeers system organized using a distributed hash table (DHT). Through the properties of the distributed hash table along with their distribution scheme, they show how all triples required to evaluate the preconditions of the reasoning rules have to be stored together at one of the nodes. Thus, they are able to calculate possible inferences in a completely de-centralized process and then use the same process to materialize triples into the store. To solve the load balancing problem inherent in term-based partitioning, they introduce an overlay tree structure able to split the data with colliding hash values onto several nodes. However, this forces them to replicate schema information across nodes.

Fang et al. [28] presented an approach for distributed reasoning, where they first perform reasoning on the schema (TBox) using a Description Logic (DL) reasoner and then use the schema closure to create reasoning rules that are applied to the instances stored in a DHT (ABox). The schema is assumed to be present on all nodes, allowing any stand-alone reasoner to create the schema closure locally. To apply the reasoning rules to the instances, a "prefetch" operation retrieves the instance data that potentially match the reasoning rules to the local machine. After calculating the inferences, the results are distributed to other nodes, where they trigger further reasoning and reach the closure after multiple iterations of the process. The main issues with this approach are, again, the need for complete schema information on all nodes as well as the prefetch of data. For example, consider all instances in the data using a single RDF class. If this class is also defined to be the sub-class of another class, every node needs to load all the instances of the first class from all other nodes in order to evaluate the rule.

Kaoudi et al. [29] study the trade-offs between distributed forward-chaining and backward-chaining on RDF data on top of a DHT storage network and present their own algorithm on distributed backward-chaining with recursive lookup operations based on values from the query and the set of RDFS reasoning rules matching the query. For example, if the type of a resource is queried, they traverse the subclass hierarchy of the schema potentially stored at various parts in the network, ultimately determining all classes a resource is an instance of. They also argue against the scalability of distributed

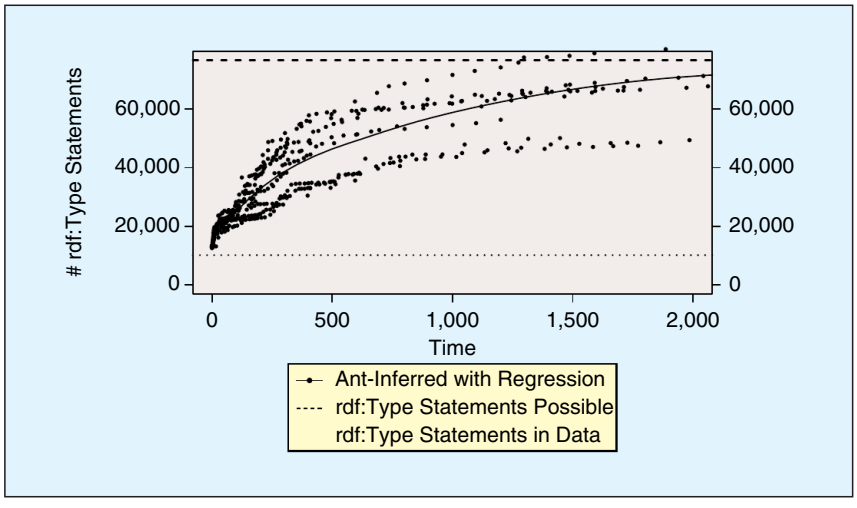


FIGURE 6 Inferred statements—repeated experiments.

forward-chaining in DHTs based on experimental results with a very small data set.

Marvin [30] is a platform for distributed reasoning on a network of loosely coupled nodes. The authors present a divide-conquer-swap strategy and show that the model converges towards completeness. Their routing strategy combines data clustering with randomly exchanging both schema and data triples. On each node, an off-the-shelf reasoner computes the closure. To handle the problem of inferred duplicates which cost memory and bandwidth, the authors propose a "one exit-door" policy, where the responsibility to detect each triple's uniqueness is assigned to a single node. This node uses a Bloom filter to detect previously hosted triples, marks the first occurrence of a triple as the master copy and removes all subsequent copies. For large numbers of nodes, a sub-exit door policy is introduced, where some nodes explicitly route some triples to an exit door. This incurs additional bandwidth costs to send triples to these exit doors.

Kotoulas et al. [26] show that widely employed term-based partitioning (such as in [27], [28] and [29]) limits scalability due to load-balancing problems. They propose a self-organized method to distribute data by letting it semi-randomly flow in the network, which allows clustered neighborhoods to emerge, and implemented it on top of Marvin. Both schema and data triples are moving in the network. A drawback of both Marvin-based approaches is that they solely rely on weighted randomness to ensure that data and schema triples come together at some point. As the number of nodes increases, we expect this to become increasingly less likely.

Urbani et al. [31] propose a scalable and distributed method to compute the RDFS closure of up to 865M triples based on MapReduce. One of the crucial optimizations is to load schema triples into the main memory of all the nodes, as the number of schema triples is usually significantly smaller than the number of data triples, and RDFS rules with two antecedents include at least one schema triple. We deliberately abandon this option, as we aim for an approach that is scalable and adaptive for all kinds of distributions among schema and data triples. Furthermore, replicating the schema information is no longer applicable when dealing with rules that contain two or more data triples as antecedents. In subsequent work[32], the approach was extended to the OWL Horst [33] semantics, able to deal both with required joins between multiple instance triples and multiple required joins per rule. The authors demonstrate the scalability of their approach by calculating the closure of 100 billion triples.

Salvadores et al. [2] have added support for reasoning for minimal RDFS rules in the distributed RDF storage system 4store. They include backward-chaining into the basic retrieval operation in a way very similar to the method presented in [29]. However, they avoid additional retrieval operations by synchronizing all schema information between the participating nodes using a dedicated node. While being able to deliver impressive scalability in experiments, the need for synchronization of schema information jeopardizes the adaptability and the robustness of their approach.

The same limitation applies to the work of Weaver et al. [34], who present a method for parallel RDFS reasoning. It is based on replicating all schema triples to all processing nodes and randomly partitioning the ABox, ignoring triples that extend the RDF Schema. The approach generates duplicates.

Hogan et al. [35] follow a pre-processing approach to scalable reasoning based on a semantics-preserving separation of terminological data. They create a set of stand-alone "template" rules formed from integrating the TBox into the reasoning rules. These rule sets are saturated with dependent rules and indexed for quick access, all aimed at a one-pass calculation of the full closure. They claim that their approach is distributable by distributing said rule sets to multiple nodes. While the template rules use a very similar notion as the reasoning operation presented here, their separation and template generation is based on the entire TBox being present at some point.

V. Conclusion and Future Work

We have explained how swarm self-organization is an interesting candidate to solve the challenges on the way towards web-scale storage, retrieval and reasoning on semantic data. Swarm algorithms already come with many of the properties desired for such systems: They are able to scale to an arbitrary number of operations, they exhibit robustness against failure and can adapt to almost any environment. In this paper, we have investigated our research question of whether swarm-based approaches are useful in creating a large-scale distributed storage and reasoning system. To this end, we have explained how the operations for the storage of new data, for the retrieval of data, and for the reasoning operation can be implemented according to the principles of swarm intelligence, in particular the foraging and brood sorting methods used by ants.

Since the non-deterministic mode of operation within these simulated swarms inhibits a formal proof of our approach, we have shown a stochastic analysis and experiments with black-box tests, where the behavior of the system is compared against a theoretical optimum. The experiments for the storage and retrieval operations measured the number of hops inside the storage network taken to retrieve any single triple. Results showed an almost perfect recall rate and—over several repeated experiments—an at least linear scaling behavior over the number of participating nodes. We have further compared these results with an optimal routing algorithm that always finds the perfect path inside the storage network. The comparison with our experimental results showed a two-fold increase in hops for our approach, which is acceptable in most cases. Thus, we assume our storage and retrieval operations will scale. To answer our research question, swarm-based approaches are indeed useful in creating a distributed storage and reasoning system for Semantic Web data, and we have shown the general feasibility and efficiency of our approach.

The goal of all reasoning operations on Semantic Web data is the generation of inferred statements. Thus, we compared the number of new statements generated by our swarm-based approach to reasoning against the number of inferred

statements calculated with a conventional stand-alone reasoner as a gold standard. These results showed the expected anytime behavior, where sound inferences are generated over time, approximating closure.

By comparing our approach to the related work in distributed storage and reasoning for Semantic Web data, we made two observations: First, distributed reasoning is a heavily-researched topic, answering the need we have identified in our introduction. Second, a wealth of methods ranging from MapReduce to random interactions is employed with impressive results, each either mainly focusing on completeness or on performance. However, these methods still have to be integrated with sufficient capabilities for storing new data and querying both the explicit and inferred statements, which has so far not been achieved in a fully decentralized way. We have presented a novel approach, where the reasoning operations re-use already present data structures for efficiency in a fully distributed way without any replication.

From our experiments that were aimed to show the potential of swarm algorithms for large-scale semantic storage, we are convinced that this idea is promising and merits further research in many directions.

V. Acknowledgments

We would like to thank our anonymous reviewers for their insightful and constructive comments. This research has been partially supported by the "DigiPolis" project funded by the German Federal Ministry of Education and Research (BMBF) under grant number 03WKP07B.

References

[1] C. Prehofer and C. Bettstetter. (2005, July). Self-organization in communication networks: Principles and design paradigms. *IEEE Commun. Mag.* [Online]. 43(7), pp. 78–85. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1470824

[2] M. Salvadores, G. Correndo, S. Harris, N. Gibbins, and N. Shadbolt, "The design and implementation of minimal RDFS backward reasoning in 4store," in *Proc. 8th Extended Semantic Web Conf. (ESWC)*, Heraklion, Greece, G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. D. Leenheer, and J. Pan, Eds. Springer-Verlag, May 29–June 2, 2011.

[3] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[4] G. D. Caro and M. Dorigo. (1998). AntNet: Distributed stigmergetic control for communications networks. *J. Artif. Intell. Res.* [Online]. 9, pp. 317–365. Available: http://dx.doi.org/10.1613/jair.530

[5] W. Vogels. (2008). Eventually consistent. *ACM Queue* [Online]. 6(6), pp. 14–19. Available: http://doi.acm.org/10.1145/1466443.1466448

[6] L. Rutkowski, *Computational Intelligence—Methods And Techniques*. New York: Springer-Verlag, 2008.

[7] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli. (2006, Aug.). Case studies for self-organization in computer science. *J. Syst. Arch.* [Online]. 52(8–9), pp. 443–460. Available: http://linkinghub.elsevier.com/retrieve/pii/S1383762106000166

[8] H. Mühleisen, A. Augustin, T. Walther, M. Harasic, K. Teymourian, and R. Tolksdorf. (2010). A self-organized semantic storage service, in *Proc. 12th Int. Conf. Information Integration and Web-based Applications and Services (iiWAS2010)*. ACM, pp. 357–364 [Online]. Available: http://portal.acm.org/citation.cfm?id=1967542

[9] K. Dentler, C. Guéret, and S. Schlobach, "Semantic web reasoning by swarm intelligence," in *Proc. 5th Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*.

[10] P. Hayes. (2004, Feb.). RDF semantics. World Wide Web Consortium, Recommendation REC-rdf-mt-20040210 [Online]. Available: http://www.w3.org/TR/rdf-mt/

[11] E. Prud'Hommeaux and A. Seaborne, "SPARQL query language for RDF," World Wide Web Consortium, Recommendation REC-rdf-sparql-query-20080115, Jan. 2008.

[12] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford: Oxford Univ. Press, 1999.

[13] H. Mühleisen, T. Walther, and R. Tolksdorf. (2011). Multi-level indexing in a distributed self-organized storage system, in *Proc. IEEE Congr. Evolutionary Computation (CEC)*, pp. 989–994 [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5936494

[14] H. Mühleisen, T. Walther, and R. Tolksdorf. (2011). Data location optimization for a self-organized distributed storage system, in *Proc. 3rd World Congr. Nature and Biologically Inspired Computing (NaBIC)*, IEEE Press [Online]. Available: http: //hannes.muehleisen.org/NaBIC2011-muehleisen-s4-movement.pdf

[15] H. Mühleisen, "Query processing in a self-organized storage system," in *Proc. VLDB2011 PhD Workshop, co-located with 37th Int. Conf. Very Large Databases (VLDB)*, 2011.

[16] A. Schlicht and H. Stuckenschmidt. (2010). Peer-to-peer reasoning for interlinked ontologies. *Int. J. Semantic Computing (Special Issue on Web Scale Reasoning)*, pp. 1–31 [Online]. Available: http://ki.informatik.uni-mannheim.de/fileadmin/publication/Schlicht10p2p.pdf

[17] P. Obermeier, A. Augustin, and R. Tolksdorf, "Towards swarm-based federated web knowledgebases," in *Proc. Workshops der wissenschaftlichen Konferenz Kommunikation in verteilten Systemen* 2011, vol. 10.

[18] D. Fensel and F. van Harmelen. (2007). Unifying reasoning and search to web scale. *IEEE Internet Comput.* [Online]. 11(2), pp. 94–96. Available: http://doi.ieeecomputersociety.org/10.1109/MIC.2007.51

[19] D. Fensel, F. van Harmelen, B. Andersson, P. Brennan, H. Cunningham, E. D. Valle, F. Fischer, Z. Huang, A. Kiryakov, T. K.-I. Lee, L. Schooler, V. Tresp, S. Wesner, M. Witbrock, and N. Zhong, "Towards LarKC: A platform for web-scale reasoning," in *Proc. 2nd IEEE Int. Conf. Semantic Computing (ICSC)*. IEEE Press, 2008, pp. 524–529.

[20] D. Peleg and U. Pincas, "The average hop count measure for virtual path layouts," in *Proc. DISC: Int. Symp. Distributed Computing*, LNCS, 2001.

[21] S. Tang, H. Wang, and P. V. Mieghem. (2008). The effect of peer selection with hopcount or delay constraint on peer-to-peer networking, in *Networking* (Lecture Notes in Computer Science Series, vol. 4982), A. Das, H. K. Pung, F. B.-S. Lee, and L. W.-C. Wong, Eds. Springer-Verlag, pp. 358–365 [Online]. Available: http://dx.doi.org/10.1007/978-3-540-79549-0_31

[22] A. Fronczak, P. Fronczak, and J. A. Hołyst. (2004, Nov.). Average path length in random networks. *Phys. Rev. E* [Online]. 70(5), pp. 056110+. Available: http://dx.doi.org/10.1103/PhysRevE.70.056110

[23] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

[24] A. Harth and P. Buitelaar. (2009). Exploring semantic web data sets with VisiNav, in *Proc. 6th Annu. European Semantic Web Conf. (ESWC2009)*, Poster Session [Online]. Available: http://sw.deri.org/2009/01/visinav/eswc-poster/visinav-poster.pdf

[25] G. Qi, J. Pan, and Q. Ji. (2007). Extending description logics with uncertainty reasoning in possibilistic logic, in *Proc. Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pp. 828–839 [Online]. Available: http://www.springerlink.com/index/j5102m4034j04235.pdf

[26] S. Kotoulas, E. Oren, and F. Van Harmelen. (2010). Mind the data skew: Distributed inferencing by speeddating in elastic regions, in *Proc. 19th Int. Conf. World Wide Web (WWW2010)*, ACM, pp. 531–540 [Online]. Available: http://portal.acm.org/citation.cfm?id=1772745

[27] D. Battré, F. Heine, A. Höing, and O. Kao. (2006). On triple dissemination, forward-chaining, and load balancing in DHT based RDF stores, in *Proc. 4th Int. Workshop Databases, Information Systems and Peer-to-Peer Computing (DBISP2P2006)* (Lecture Notes in Computer Science Series, vol. 4125), G. Moro, S. Bergamaschi, S. Joseph, J.-H. Morin, and A. M. Ouksel, Eds. Springer-Verlag, pp. 343–354 [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71661-7_33

[28] Q. Fang, Y. Zhao, G. Yang, and W. Zheng, "Scalable distributed ontology reasoning using DHT-based partitioning," in *Proc. 7th Int. Semantic Web Conf. (ISWC2008)*, 2008, pp. 91–105.

[29] Z. Kaoudi, I. Miliaraki, and M. Koubarakis. (2010). RDFS reasoning and query answering on top of DHTs, in *Proc. 7th Int. Semantic Web Conf. (ISWC2008)*. Springer-Verlag, pp. 499–516 [Online]. Available: http://www.springerlink.com/index/V6440538394785H6.pdf

[30] E. Oren, S. Kotoulas, G. Anadiotis, R. Siebes, A. Ten Teije, and F. Van Harmelen. (2009). Marvin: Distributed reasoning over large-scale Semantic Web data. *J. Web Semantics* [Online]. 7(4), pp. 305–316. Available: http://linkinghub.elsevier.com/retrieve/pii/S1570826809000444

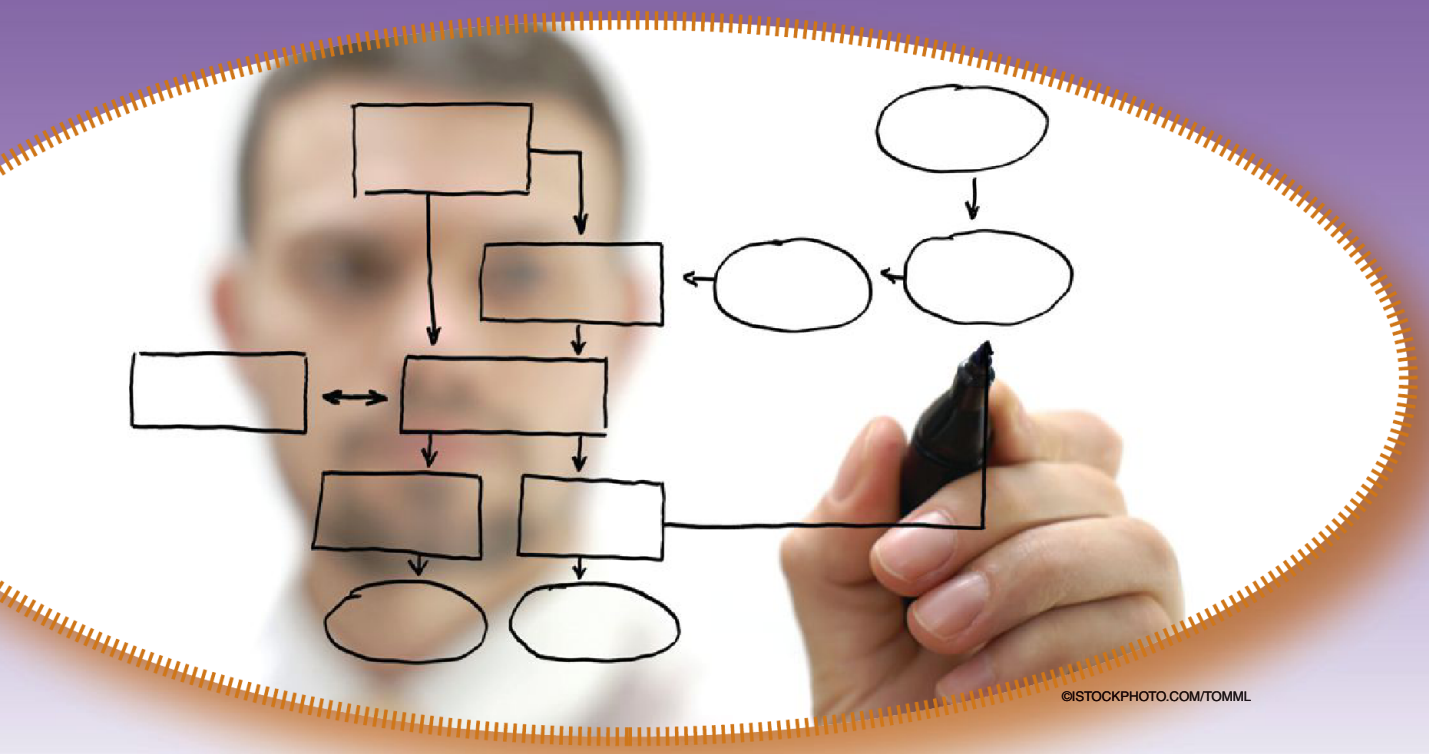
[31] J. Urbani, S. Kotoulas, E. Oren, and F. van Harmelen. (2009). Scalable distributed reasoning using MapReduce, in *Proc. 8th Int. Semantic Web Conf. (ISWC)*, Springer-Verlag, pp. 634–649 [Online]. Available: http://www.springerlink.com/index/M44432748XT110PJ.pdf

[32] J. Urbani, S. Kotoulas, J. Maassen, F. Van Harmelen, and H. Bal. (2010). OWL reasoning with WebPIE: Calculating the closure of 100 billion triples, in *Proc. 9th Int. Semantic Web Conf. (ISWC)*. Springer-Verlag, pp. 213–227 [Online]. Available: http://www.springerlink.com/index/2581664J64961667.pdf

[33] H. Horst. (2005). Combining RDF and part of OWL with rules: Semantics, decidability, complexity, in *Proc. 4th Int. Semantic Web Conf. (ISWC)*, pp. 668–684 [Online]. Available: http://www.springerlink.com/index/366474250nl35412.pdf

[34] J. Weaver and J. Hendler. (2009). Parallel materialization of the finite rdfs closure for hundreds of millions of triples, in *Proc. 8th Int. Semantic Web Conf. (ISWC)*. Springer-Verlag, pp. 682–697 [Online]. Available: http://www.springerlink.com/index/77X71125037K6583.pdf

[35] A. Hogan, J. Pan, and A. Polleres. (2010). SAOR: Template rule optimisations for distributed reasoning over 1 billion linked data triples, in *Proc. 9th Int. Semantic Web Conf. (ISWC)*, vol. 1380, pp. 337–353 [Online]. Available: http://www.springerlink.com/index/M6144754NM475404.pdf



Exploiting Tractable Fuzzy and Crisp Reasoning in Ontology Applications

Abstract–The Semantic Web movement has led to the publication of thousands of ontologies online. These ontologies present and mediate information and knowledge on the Semantic Web. How to provide tractable reasoning services for fuzzy and crisp ontologies has been a pressing research problem over the last five years. In this paper, we present a reusable semantic infrastructure that comprises a tractable reasoning system TrOWL, an ontological search engine ONTOSEARCH2 and a folksonomy extension component Taggr. We show that such an infrastructure can be used to support different ontology applications with tractable reasoning services by making use of tractable profiles in OWL 2 and some of their fuzzy extension.

Jeff Z. Pan, Edward Thomas, Yuan Ren, and Stuart Taylor, University of Aberdeen, UK

I. Introduction

Ontologies play a key role in the Semantic Web, where the W3C recommendation OWL[1] and its successor OWL 2[2] have become the de facto standards for publishing and sharing ontologies online. Increasingly these ontologies are being used by a variety of organizations, covering the definitions of a very wide range of subjects. Fuzzy ontologies are envisioned to be useful in the Semantic Web. On the one hand, ontologies serve as a semantic infrastructure, providing shared understanding of certain domain across different applications, so as to facilitate machine understanding of Web resources. On the other hand, being able to handle fuzzy and imprecise information is crucial to the Web.

Providing tractable reasoning services for OWL ontologies is not a trivial problem—OWL 1 DL has a worst-case computational complexity of NExpTime,

Digital Object Identifier 10.1109/MCI.2012.2188588
Date of publication: 13 April 2012

[1]http://www.w3.org/2004/OWL/
[2]http://www.w3.org/TR/owl2-overview/

while 2NExpTime for OWL 2 DL. This means that increasingly large ontologies require exponentially more computing resources to reason over them. Because of this, OWL 2 includes a number of tractable profiles which have combined complexity of PTIME-complete or better; including OWL 2 EL, OWL 2 QL and OWL 2 RL. However, these profiles all greatly restrict the expressive power of the language. As tool support for these profiles is still limited, it is also very easy for an ontology developer to accidentally exceed the complexity of their target profile by using a construct that is beyond the capability of that language fragment. Approximation ([1]–[7]) has been identified as a potential way to reduce the complexity of reasoning over ontologies in very expressive languages such as OWL 1 DL and OWL 2 DL.

On the front of fuzzy ontology reasoners, there are similar observations. There exists no fuzzy OWL 1 DL ontology reasoners that could be efficient and/or scalable enough to handle the scale of data that the Web provides. Interestingly, there currently exist two fuzzy ontology reasoners, namely the tableaux based fuzzy reasoner FiRE[3] ([8], [9]), which supports a nominal and datatype-free subset of fuzzy-OWL 1 DL, i.e. fuzzy-$\mathcal{SHIN}$, and the mixed integer programming fuzzy reasoner *fuzzyDL*[4], which supports fuzzy-OWL Lite, namely fuzzy-$\mathcal{SHIF}(\mathcal{D})$ [10]. Like their crisp counterparts, fuzzy-$\mathcal{SHIN}$ and fuzzy-$\mathcal{SHIF}(\mathcal{D})$ come with (at least) EXPTIME computational complexity but the reasoners are not as highly optimized as crisp DL reasoners, thus the scalability of the above systems is doubtful. Another approach to reason with fuzzy ontologies is to reduce to crisp ontologies, as implemented in DeLorean [11][5]. However this approach does not reduce the complexity but increases the number of axioms. Following current research developments in crisp DLs, there is an effort on lightweight fuzzy ontology languages. In particular, Straccia [12] extended the DL-Lite ontology language [13] to fuzzy DL-Lite.

In this paper, we present a reusable semantic infrastructure based on tractable OWL 2 profiles. On the one hand, the infrastructure supports a general framework [14] of query languages for fuzzy OWL 2 QL, including the support for threshold queries and general fuzzy queries. On the other hand, the infrastructure supports faithful approximate reasoning ([6], [7]) for OWL 2 DL, based on OWL 2 EL and OWL 2 QL. The rest of the article is organized as follows: In Sec. II we present some use cases of reasoning in fuzzy and crisp ontologies. In Sec. III we present the reusable reasoning infrastructure. In Sec. IV we present a scenario of using the infrastructure in one of the presented use cases in details. Finally in Sec. V we present some evaluation results on the usefulness of the infrastructure on the presented use cases.

II. Use Cases

Our reusable semantic infrastructure supports a variety of reasoning and search services for different real-world application scenarios. Before we go into the details of the technologies developed for our infrastructure, we first illustrate their necessity and versatility by briefly introducing some motivating applications.

A. Mashup

A mashup is a kind of data-integration application, where information regarding input queries is collected from different sources, in different medias, and presented to users in a coherent manner. MusicMash2[6] is an ontology-based semantic mashup application, that is intended to integrate music related content from various folksonomy based tagging systems, linked open data[7], and music metadata Web services. MusicMash2 provides the functionality for users to search for tagged images and videos that are related to artists, albums, and songs.

An application of this nature presents two main problems. The first problem lies with the availability of populated domain ontologies on the Web. The Music Ontology (http://www.musicontology.com/) provides classes and properties for describing music on the Web; however, to instantiate the ontology, MusicMash2 must integrate music meta-data from various sources. The resulting populated ontology may be very large, so a suitable mechanism for reasoning over this ontology must be used—this mechanism must be scalable and efficient enough to handle queries on this ontology with an acceptable response time for users.

The second problem is that searching both within and across folksonomy based systems is an open problem. A naive approach to folksonomy search, such as those provided in most tagging systems[8], results in unacceptable precision for domain specific searches. The lack of search precision is due to the limitations of tagging systems themselves [16]. MusicMash2 addresses this problem by making use of the fuzzy reasoning services, keyword-plus-entailment search (cf. Sec. III-A) and the folksonomy search expansion service (cf. Sec. III-B) of our infrastructure.

B. Process Refinements Validation

During software development, processes are frequently modelled and refined with more and more detailed knowledge of the workflow and business logic of the system. During refinements, the activities in the abstract process are decomposed, and re-organized in the specific process. For example, the following Figure 1 (taken from [17]) shows two models of a hiring process in the standard language BPMN (Business Process Modeling Notation), where the *Specific* model refines the *Abstract* model. The original two activities *Select Applicant* and *Hire Applicant* are decomposed during the procedure of refinement into more fine-grained activities. And the flow structure becomes more complex with the parallel and/or exclusive gateways.

Due to the potentially complex decomposition and restructuring of activities, it is usually difficult to tell if the

[3] http://www.image.ece.ntua.gr/~nsimou
[4] http://gaia.isti.cnr.it/~straccia
[5] http://webdiis.unizar.es/~fbobillo/delorean
[6] http://www.musicmash.org/
[7] http://linkeddata.org/
[8] E.g. YouTube API: http://www.youtube.com/dev/

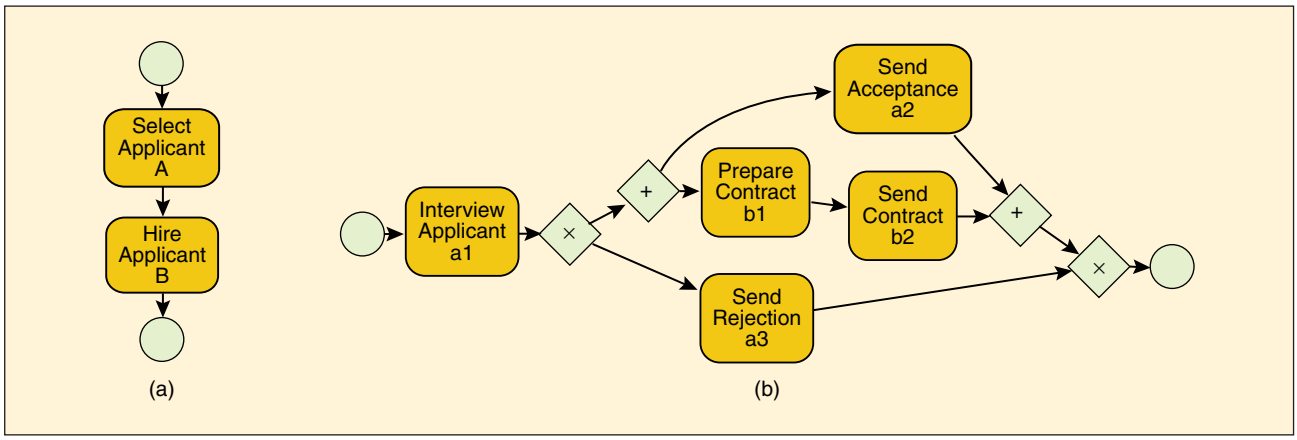


FIGURE 1 A refinement step. (a) Abstract and (b) specific.

refined processes properly capture the intended behavior of the original processes. In [17], an ontology based approach has been developed to reduce a process refinement problem to an ontology concept satisfiability/subsumption checking problem by encoding the relations between activities with ontology axioms.

This application raises an important challenge. The process refinement ontologies use the ontology language OWL 1 DL, and contain general concept inclusions (GCIs) of particular patterns. Due to the EXPTIME complexity, at the time of developing this ontological solution, mainstream reasoners such as Pellet and FaCT++ failed to provide complete classification results in a short period of time. It is crucial to develop new reasoning techniques to handle these GCIs in an efficient and quality-guaranteed manner. This requirement is fulfilled by the faithful approximate reasoning services (cf. Sec. III-C) of our tractable semantic infrastructure.

C. Software Engineering Guidance

The case study of physical device configuration of the MOST project [9] uses ontologies to validate the consistency of configurations of network devices. A typical configuration can be seen in Figure 2 (taken from [18]). As we can see, a configuration usually involves multiple slots, and every slot can host certain types of cards. The types of cards in a same configuration also have certain dependencies and restrictions. All these constraints are encoded as axioms in ontologies as expressive as OWL 2 DL [19].

These ontologies can sometimes be inconsistent, reflecting an invalid configuration of a physical device. To understand how this is manifested in the physical device and provide guidance on how it may be resolved, it is necessary to perform reasoning efficiently in modeling time, and to find justifications for the inconsistency. Furthermore, in order to provide suggestions to engineers, some of the concepts or properties in the ontology need to be closed. Traditional ontology reasoning imposes the Open World Assumption and does not support such services. Closed domain reasoning services are required to address this issue. This requirement is eventually fulfilled by the support of NBox (Negation as failure Box [20]) and approximate reasoning services (cf. Sec. III-C) in our tractable semantic infrastructure.

[9]www.most-project.eu/

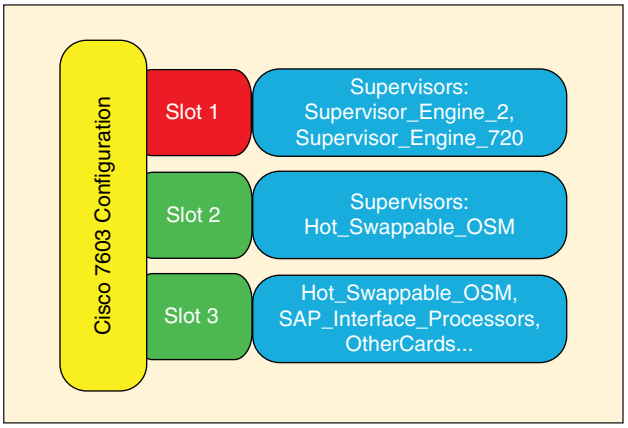


FIGURE 2 A Cisco7603 configuration.

To conclude this section, we summarise the challenges to existing reasoning technologies that are raised by the above use cases:

1) The scalability and efficiency of querying on large scale populated domain ontologies.
2) The precision of folksonomy search, especially for vague and/or ambiguous terms.
3) The efficiency of reasoning in expressive and very expressive ontology languages.
4) The availability of local closed world reasoning services.

In the rest of the article, we will present our proposed solutions to these challenges.

III. A Tractable Semantic Infrastructure

In this section, we present a tractable semantic infrastructure. Our architecture consists of three layers of semantic tools and they can all be accessed by external applications via APIs. The system and its interaction with the use cases are illustrated in the Figure 3.

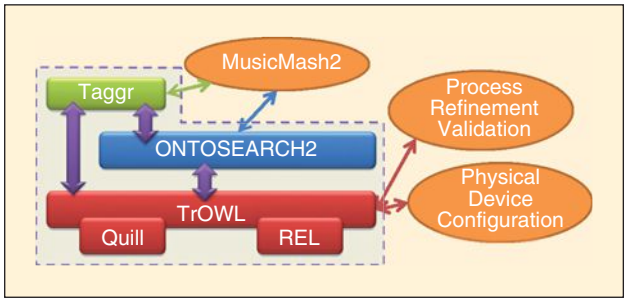


FIGURE 3 Semantic infrastructure and its interaction with use cases.

TrOWL[10] [21] is a tractable reasoning infrastructure of OWL 2. It includes a built-in OWL 2 QL reasoner Quill and a built-in OWL 2 EL reasoner REL. It has been continuously extended with new reasoning services, such as approximation ([6], [7]), forgetting [22], justification, local closed world reasoning (NBox) [20] and stream reasoning ([23], [24]). It can directly support the process refinement validation and physical device configuration use cases.

ONTOSEARCH2[11] ([25, 26]) is an ontological search engine that is based on the TrOWL ontology infrastructure. It supports keyword-plus-entailment search by making use of fuzzy and crisp reasoning services from TrOWL.

Taggr [27] is a folksonomy extension component which has been built on ONTOSEARCH2. MusicMash2 uses both the ONTOSEARCH2 and Taggr.

The architecture includes an ontology repository (as part of TrOWL), where users can submit and share ontologies and a query rewriter that rewrites user queries submitted in SPARQL so that they can be executed on the repository. A useful feature of the repository is that it automatically associates keywords (in values of annotation properties as well as implicit metadata in target ontologies) with concepts, properties and individuals in the ontologies. Default annotation properties include the `rdfs:label`, `rdfs:comment`, `rdfs:seeAlso`, `rdfs:isDefinedBy`, and `owl:versionInfo` properties; we also define the `dc:title`, `dc:description`, and `foaf:name` properties (from Dublin Core[12] and FOAF[13]) as annotation properties. Implicit metadata is drawn from the namespace and ID of each artefact in the ontology.

These keywords are weighted based on ranking factors similar to those used by major search engines[14]. The system uses these scores to calculate the $tf \cdot idf$ [28] for each keyword found within the ontology, and normalises them using a sigmoid function such as the one shown in (1) to a degree between 0 and 1.

$$w(n) = \frac{2}{1.2^{-n} + 1} - 1. \quad (1)$$

[10] http://trowl.eu/
[11] http://www.ontosearch.org/
[12] http://dublincore.org/
[13] http://www.foaf-project.org/
[14] http://www.seomoz.org/article/search-ranking-factors

In Sec. III-A, we first introduce how the knowledge and their degree values can be used to provide fuzzy reasoning services with ONTOSEARCH2. Then in Sec. III-B we show how to use Taggr to extend the search services to folksonomies. Finally we present the details of TrOWL in Sec. III-C to show how the underlying crisp reasoning services are accomplished.

A. Fuzzy Reasoning Services

Being able to handle fuzzy and imprecise information is crucial to the Web. TrOWL also consists of a query engine for fuzzy OWL 2 QL [14]. The query engine supports threshold queries and general fuzzy queries over fuzzy OWL 2 QL ontologies. Users of the query engine can submit fuzzy OWL 2 QL ontologies via the Web interface of ONTOSEARCH2[6], and then submit f-SPARQL [14] queries, such as the following one, against their target ontologies.

```
#TQ#
PREFIX music: <http://musicmash.org/NS/>
SELECT ?x WHERE {
     ?x a music:MusicArtist.
     ?x a music:Popular. #TH# 0.7
     ?x a music:Active. #TH# 0.8
}
```

where #TQ# declares a threshold query, while #TH# is used to specify thresholds for atoms in the query. Therefore this query searches for an instance of `MusicArtist` which is a member of the class `Popular` with degree `0.7`, and a member of the class `Active` with degree `0.8`.

Preliminary evaluations show that performance of the fuzzy OWL 2 QL query engine is in most cases close to the performance of the crisp OWL 2 QL query engine [14].

ONTOSEARCH2 is an ontological search engine that allows users to search its repository with keyword-plus-entailment searches, such as *searching for ontologies in which class X is a sub-class of class Y, and class X is associated with the keywords "Jazz" and "Rock", while class Y is associated with the keyword "Album"*. The search could be represented as the following threshold query:

```
#TQ#
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
schema#>
PREFIX os: <http://www.ontosearch.org/NS/>
PREFIX kw: <http://www.ontosearch.org/KW/>

SELECT ?x WHERE {
?x os:hasKeyword kw:jazz. #TH# 0.5
?x os:hasKeyword kw:rock. #TH# 0.7
?x rdfs:subClassOf ?y.
?y os:hasKeyword kw:album. #TH# 0.8}
```

where kw:jazz, kw:rock, and kw:album are representative individuals for keywords "jazz", "rock", and "album", respectively. The thresholds 0.5, 0.7, and 0.8 can be specified by users. The keyword-plus-entailment searches are enabled by the fuzzy DL-Lite query engine as well as the semantic approximation components. Such services allow both TBox and ABox queries.

B. Relating Folksonomies to Ontologies

The *Taggr* system provides a simplified interface to ONTOSEARCH2 to perform useful operations that are related to folksonomy based systems. It stores a basic ontology (which we refer to as the "tagging database") in the TrOWL repository, capturing the relationships between users, tags and resources in the folksonomy based systems it supports[15].

Taggr allows users to provide a set of arbitrary resources and related tags to be added to the tagging database in the TrOWL repository. A Web service and a traditional Web interface are provided so that users can interact with the tagging database without having to understand the internal representation used by the system.

Taggr also provides an ontology-enabled common interface for folksonomy based systems. It provides the functionality to gather resources and their related tags from the systems that it supports, and populate them to its tagging database from time to time. In case an application requests some resources that Taggr does not know about, it can simply make a call to Taggr to request that it updates its tagging database with resources related to the search.

Furthermore, Taggr allows users to specify which search expansion method(s) [27] and which reference ontology(-ies) to use for the expansion. The extended search will first be evaluated against its tagging database. As the ontological constraints needed for the search expansions require only the expressive power of OWL 2 QL, Taggr can make use of the semantic approximation(s) of the reference OWL DL ontology(-ies) for all entailment checking. Due to the logical properties of semantic approximation, TrOWL can provide sound and complete results for all the needed entailment checking. Details of the search expansion methods go beyond the scope of this paper.

C. Crisp Reasoning Services

TrOWL is a tractable reasoning infrastructure for OWL 2. A syntax checker front-end will briefly examine the syntax of the loaded ontology and decide which built-in reasoners should be used, based on the configuration of the system. There are two major built-in reasoners. Quill provides reasoning services over RDF-DL and OWL 2 QL and REL provides reasoning over OWL 2 EL. Besides, TrOWL also supports full OWL 2 DL reasoning using a plug-in reasoner such as Pellet or FaCT++. The OWL 2 DL reasoning will have the same quality as the plug-in reasoner. Alternatively, TrOWL can also use the approximation facilities provided by REL to deliver sound and practically complete reasoning for OWL 2 DL. Such separation of reasoners make it possible to provide tailored and optimized services, instead of a "one-size-fit-all" reasoner.

1) *Language Transformations:* The transformation from OWL 2 DL to OWL 2 QL is based around the Semantic Approximation technology from OWL DL to DL-Lite [6]. This technology can be easily applied on OWL 2 DL as well. Semantic Approximation adopts the idea of knowledge compilation to pre-compute the materialisation of an OWL 2 DL ontology into OWL 2 QL axioms, by using a heavyweight reasoner. Because the semantics of OWL 2 QL are a subset of, and are hence compatible with, the direct semantics OWL 2 DL, any reasoning results against the approximated OWL 2 QL ontology are always sound with regards to the original OWL 2 DL ontology. Furthermore, it has been shown that for conjunctive query answering, which is the strength of the QL language,

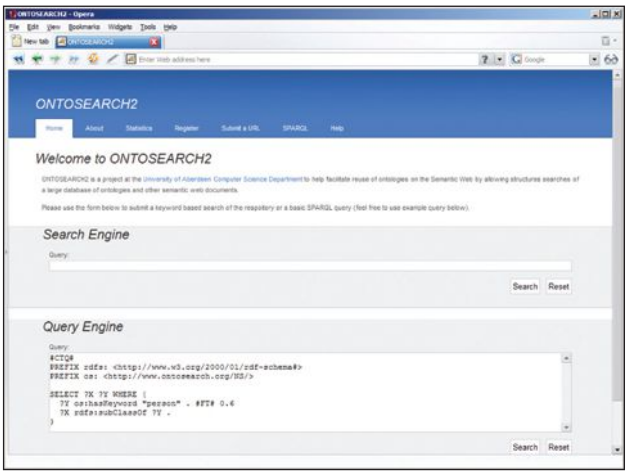


FIGURE 4 ONTOSEARCH2 screenshot.

FIGURE 5 Taggr Beta screenshot.

[15] Taggr currently supports the YouTube and Flickr tagging systems.

FIGURE 6 MusicMash 1.0 screenshot: The original version of MusicMash which does not use any Semantic Web components.

results against the semantic approximation are also complete for queries with no non-distinguished variables, or with non-distinguished variables only in leaf nodes of the query. This already covers a majority of conjunctive queries, including the queries supported by the new SPARQL specification.

The transformation from OWL 2 DL to OWL 2 EL is based on the soundness preserving approximate reasoning approach [7], [29]. This approach represents non-OWL 2 EL concept expressions with fresh named concepts, non-OWL 2 EL role expressions with fresh named roles, and then maintaining their semantics, such as complementary relations, inverse relations, cardinality restrictions, etc. in separate data structures. In the reasoning stage, additional completion rules have been devised to restore such semantics. With this approach, the overall complexity for OWL 2 DL ontologies can be reduced to PTime. Although known to be incomplete, evaluation shows that, such transformation can classify existing benchmarks very efficiently with high recall (over 95%) [7], [29].

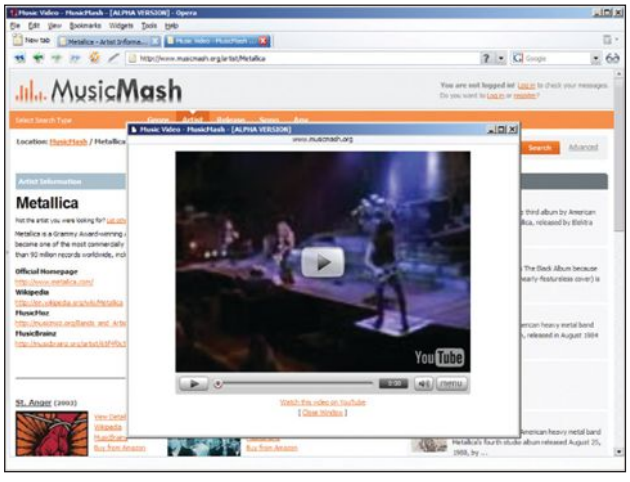


FIGURE 7 MusicMash2 screenshot: The second version of MusicMash built using our infrastructure.

Except for the improved performance of standard reasoning services, the language transformations also enable many other non-standard reasoning services. For example, the support of local closed world reasoning can be reduced to incremental reasoning by restricting the extensions of concepts, and the domains and ranges of properties, which can be further realised more efficiently by transformation to OWL 2 EL as OWL 2 EL has PTIME incremental reasoning capacity. In TrOWL, such closed world reasoning is realised by first specifying an NBox (Negation As Failure Box) in the ontology [20]. An NBox is a set of concept and property expressions. Any predicate (concept or property) in the NBox will be closed.

2) *Lightweight Reasoners:* The Quill reasoner implements a novel and unique database schema for storing normalised representations of OWL 2 QL ontologies. Any conjunctive query can be rewritten into a single, simple SQL query over the underlying database, using the database itself to perform the transitive completion of class and property subsumption with an innovative exploitation of the highly optimized database mechanisms. To support this, new query rewriting and ontology normalisation algorithms have been developed. In its initial testing across large knowledge bases with deep concept hierarchies, such as the DBPedia dataset and the Yago ontology, a significant performance improvement over other DL-Lite query engines is observed. In an extreme case, using the standard query rewriting algorithm PerfectRef over a deep class or property hierarchy can result in a set of hundreds or thousands of conjunctive queries, where the new method only ever results in a single query. Quill supports all reasoning tasks for OWL 2 QL, including consistency and satisfiability checking, and query answering, and by using an OWL 2 DL reasoner it can perform semantic approximation of more expressive ontologies.

The REL reasoner [7] is a Java implementation of an OWL 2 EL reasoner. It extends the EL+ algorithm [30] with the completion rules for OWL 2 EL [31] and further optimizes them. On top of that, the syntactic approximation and reasoning rules are also implemented. This allows REL to provide tractable reasoning for OWL 2 EL ontologies and make up the core component of the soundness-preserving syntactic approximation for OWL 2 DL ontologies. REL is an approximate OWL 2 DL reasoner with high general recall; it is complete for all the HermiT benchmark ontologies. In additional, REL also consists of an OWL 2 EL conjunctive query engine [32], which allows queries over OWL 2 EL ontologies to be answered more efficiently without semantic approximation. Non-standard reasoning services such as local closed world reasoning is also supported by REL. By integrating the idea of truth maintenance systems, REL also supports justification and ontology stream reasoning.

IV. Scenario: Mashing Linked Music Data

In this section, we describe a concrete scenario illustrating how the semantic infrastructure (presented in the previous section) could enhance a typical Web 2.0 application. Let us consider

the story of "Sarah", a keen Web developer with an interest in Web 2.0 and new Web technologies in general.

A. A Web 2.0 Application

Sarah's interest in Web 2.0 had grown from her interest in music. She enjoys using Web 2.0 websites to find multimedia content about her favourite artists. However, she found that it was inconvenient to have to browse many different websites to find content related to a single artist. Meanwhile, she had been reading about Web Services and Mashup as part of her interest in Web development. After reading a few articles on the Web, Sarah decided that she could address this inconvenience by building her own mashup application. The objective of this application is to combine music-related resources—videos, images, biographies and discographies—into a single website. Sarah also decided that if her application were to be of use, it would have to provide accurate search results in a timely fashion. Sarah named her new web application "MusicMash" and began work on the project.

B. Searching Folksonomies

To retrieve video and image content for her new site, Sarah made use of the public Web Service APIs provided by YouTube and Flickr. She quickly developed the first prototype. This version allowed users to perform searches based on an artist's name. MusicMash used the artist's name when making calls to the YouTube and Flickr APIs to retrieve videos tagged with the each word from the artist's name.

Sarah soon found that this early version of MusicMash suffered from a major drawback—that artists' names are often ambiguous search terms in YouTube and Flickr. For example, when she searched for Focus (the Dutch Progressive Rock band), only 5 out of the top 20 results returned by YouTube were relevant to that artist. Sarah noticed that users on YouTube often tagged music videos with both the artist's name and song title. She soon realised that including a song title with the artist's name generated more relevant search results.

Sarah then extended MusicMash to populate a database of music metadata retrieved from the MusicBrainz[16] web service. Using this metadata, Sarah could extend MusicMash to automatically expand a simple artist's name search, to an *artist's name plus song title* search, for each song by that artist. This search expansion technique resulted in an impressive increase in the precision of the search results. However, the volume of API calls needed for the search expansion resulted in an unacceptable amount of time required to return search results.

C. The Switch to Taggr

The performance issues in MusicMash were due to the large number of HTTP requests to Web Service APIs generated by Sarah's search expansion technique. The solution to these issues is to ensure that the APIs perform the search expansion internally hence needing only one HTTP request. Sarah learned of a system called Taggr that provides the same search expansion functionality as MusicMash via a Web Service API. Sarah decided to redevelop MusicMash using the Taggr API, rather than accessing YouTube and Flickr directly.

The Taggr API allows Sarah to input the original search term(s) from the user and some extra parameters to specify how the search expansion should be performed.

The Taggr API allows Sarah to input the original search term(s) from the user and some extra parameters to specify how the search expansion should be performed. More specifically, the parameters indicate whether video or image resources results should be returned; what the input search term(s) should identify, S (a music artist in this case); where to find the extra keywords for the search expansion, T (a song title in this case); and how S and T are related, P (a music artist is the creator of a song). Taggr uses OWL DL ontologies to represent its metadata internally. The parameters S and T should be specified as OWL classes and P as an OWL property. However, Sarah did not know which OWL class was used to identify music artists and song titles on the Web. She follow a link from the Taggr website to ONTOSEARCH2. She then made use of ONTOSEARCH2's ontology search engine to find out which ontologies contained resources relating to music. Sarah typed "music" into the ONTOSEARCH2 search engine and one of the first results returned was from the Music Ontology[17]. After further investigation Sarah found that the Music Ontology was exactly what she needed to describe the classes and properties for music artists, albums and songs.

Sarah then set about trying some searches on Taggr using the concepts `mo:MusicArtist` and `mo:Track,` related via the property `mo:foaf:made;` allowing her to replicate the search expansion from MusicMash. She first used Taggr to check for new keywords that were generated by its search expansion. Sarah tried the keyword "Coldplay" and was surprised to see that Taggr did not provide any new keywords. She then searched ONTOSEARCH2 directly for "Coldplay" and again, no results were returned. Sarah realised that she would have to provide the Music Ontology individuals herself in order for the search expansion to work correctly.

D. From Relational Databases to Ontologies

Since ontology individuals are required by Taggr to replicate the MusicMash search expansion. Sarah decided to drop her database of music metadata in favour of Music Ontology instances stored in the ONTOSEARCH2 repository. ONTOSEARCH2's submission and query engine provided the tools that she needed to insert new individuals and query against them. Sarah decided to populate her ontology using

[16] http://www.musicbrainz.org/

[17] http://purl.org/ontology/mo/

The advantage of this method is that information relating to previously unknown artists can now be added automatically to the Music Ontology in ONTOSEARCH2 and Tagging Database in Taggr.

Web Services that can be easily linked. More specifically, using MusicBrainz API for basic artist, album and song information, she could extend the metadata with other sources that referred to MusicBrainz identifiers, such as Last.fm and DBpedia. This new version of MusicMash was named MusicMash2.

The final two problems left for Sarah to address are when there is no Music Ontology instances relating to a user's search or where there were insufficient resources returned by Taggr. She decided that for any search for which MusicMash2 did not immediately return more than five results to the user, a request would be made to Taggr to populate its tagging database with more resources from tagging systems in its library. Taggr would then send requests to its supported tagging systems to retrieve the first 500 results based on the original search term(s). Similarly when MusicMash2 returns no individuals in the Music Ontology relating to the search, it initiates a background task to retrieve the required information from the Web services in its library. The advantage of this method is that information relating to previously unknown artists can now be added automatically to the Music Ontology in ONTOSEARCH2 and Tagging Database in Taggr. Sarah decided that it was an acceptable trade-off that the first user wait for the information to be retrieved, in order for future searches to be returned in a more acceptable amount of time.

TABLE 1 Artist names used in the evaluation.

SEARCH #	SEARCH TERM	SEARCH #	SEARCH TERM
1	THE BEATLES	9	THURSDAY
2	PORCUPINE TREE	10	PELICAN
3	RADIOHEAD	11	FOCUS
4	METALLICA	12	ISIS
5	EELS	13	YES
6	DOVES	14	KISS
7	FINCH	15	JET
8	STRUNG OUT		

V. Usefulness Evaluation

1) *Mashup:* A typical scenario for MusicMash2 can be illustrated by a user searching for information related to an artist. The user first enters the name of the artist into the search box. On completion of a successful search, MusicMash2 displays the related artist information to the user. This includes a short abstract from DBpedia, the artists discography and links to the artist's homepage and Wikipedia articles. The user can also select the Video Gallery tab to display videos relating to the current artist. The Video Gallery makes use of Taggr to return high precision search results for related videos. An example of an artist's page can be viewed at http://www.musicmash.org/artist/Metallica. In [27], a usefulness evaluation is conducted to compare the YouTube search engine and MusicMash2. 15 artists' names are used as search terms (see Table 1), and the presison of the top 20 results for each search term returned by the two systems is illustrated in Figure 8.

Figure 8 shows that in all but one search, Taggr outperformed YouTube in terms of precision. YouTube returned good results for less ambiguous artist names (searches #1 - 4) but poor results for ambiguous names (searches #5 - 15); i.e., where the artist's name can correspond to many non-music related videos. In searches #13 and #14, Taggr loses some precision for the more ambiguous artist's names "Yes" and "Kiss". In the final search, #15, Taggr performs slightly worse than YouTube, by 15%. The reason for the loss of precision could be because there were instances where the artist's name and song title combinations are themselves ambiguous, and in some cases less effective than YouTube's built-in search facilities. A simple approach to addressing this issue would be to attempt to rank the results by examining the complete set of tags for a resource and checking how many are actually matched by the keywords generated in the search expansion.

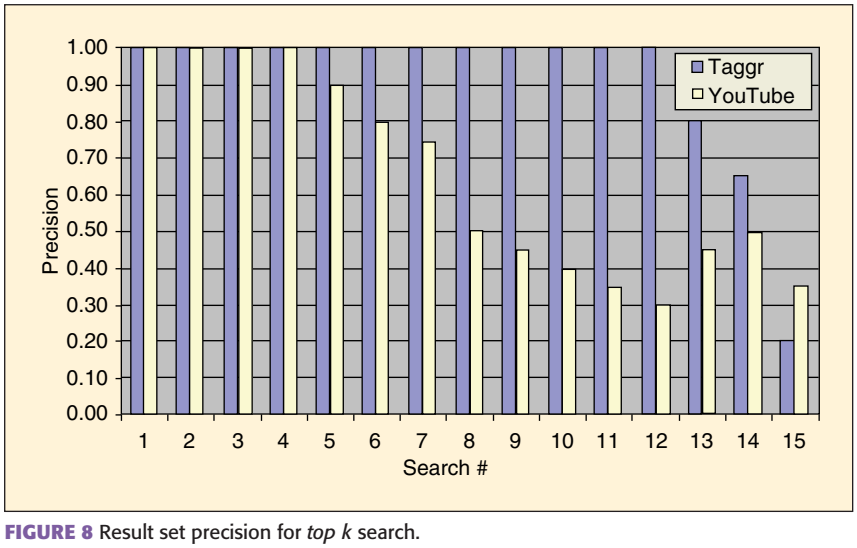


FIGURE 8 Result set precision for *top k* search.

1) Process Refinement Validation: The usefulness on process refinement validation can be examined by comparing the utility of the ontological solution with other solutions, and also comparing the performance of the optimisation of approximation with off-the-shelf reasoners. The completeness and soundness of the solution by using approximation has been theoretically proved in [17]. In [33] a comprehensive empirical study is conducted by SAP to test the usefulness and performance of the

validation methodology and reasoning facilities. The usefulness evaluation shows that, with the overall solution, a process developer can become about 3 times faster than before (productivity) and achieve 1.6 times their original correctness rate (quality). A further performance comparison between REL and Pellet suggested that, when both using ontologies to validate process refinement, the syntactic approximation-based REL solution is significantly faster than the off-the-shelf reasoner Pellet, and more scalable when increasing the length, branching and parallelisms of process models. When compared to non-ontological solutions such as the Petri Net, REL becomes less scalable and efficient when the parallelism of the processes are increased, due to the factorial increase of possible executions of processes caused by parallelism. However, Petri Net is only capable of identifying the 1st invalid activity while an ontology-based solution can find all of them in one go.

2) *Software Engineering Guidance:* The technologies developed in the TrOWL have also been used to support the ontological solution of the physical device configuration case study. Its usefulness can be evaluated by comparing it to the pure model-based state-of-the-art approach used in companies. In [34] COMARCH conducted an empirical evaluation and the results suggest that by using the ontology-integrated solution, the development time is reduced by 76% and bug-fixing time is reduced by 100%, meaning that no error was found in the modeling with the ontological solution. This correctness result is also formally proved in [19], which shows that despite the high expressive power and computational complexity of the physical device ontology, the completeness and soundness of using syntactic approximation implemented in TrOWL can be guaranteed for the considered types of tasks.

VI. Conclusion

In this paper, we have presented a novel tractable semantic infrastructure based on the OWL 2 profiles. The infrastructure provides tractable fuzzy and crisp ontology reasoning services, as well as keyword-plus-entailment search services and tailored support for folksonomy systems, validation services for business process refinement and guidance services for ontology driven software development.

By combining open ontologies with information retrieved from proprietary knowledge bases, we increased the access to this information through open interfaces. Since ONTOSEARCH2, a front end of our infrastructure, is publicly accessible through a standardised interface (SPARQL), it is possible for other applications to be built on top of the ontologies generated by existing applications, such as MusicMash2. Hence, we provide an infrastructure that may stimulate further development and/or population of domain ontologies.

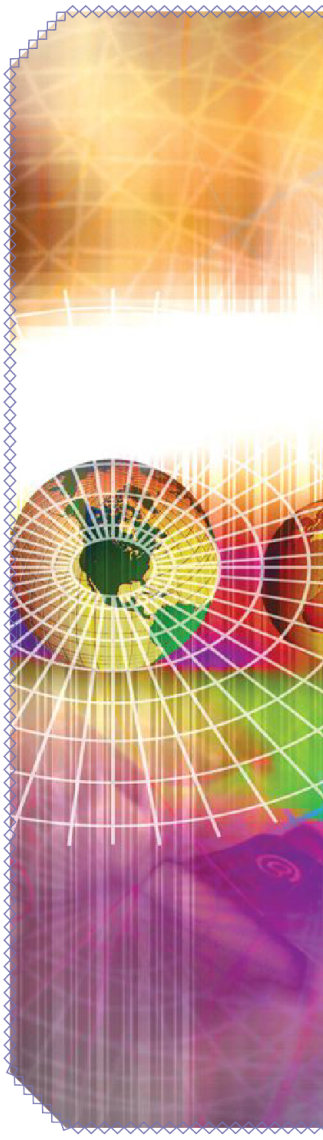
VI. Acknowledgment

The case studies, implementations and evaluations presented in this work have been partially funded by the EU project MOST. The authors would like to thank our MOST colleagues Yuting Zhao, Andreas Friesen, Jens Lemcke, Tirdad Rahmani and Krzysztof Miksa.

References

[1] H. Stuckenschmidt and F. van Harmelen, "Approximating terminological queries," in *Proc. FQAS2002*, pp. 329–343.
[2] H. Wache, P. Groot, and H. Stuckenschmidt, "Scalable instance retrieval for the semantic web by approximation," in *Proc. WISE-2005 Workshop on Scalable Semantic Web Knowledge Base Systems*.
[3] P. Hitzler and D. Vrandecic, "Resolution-based approximate reasoning for OWL DL," in *Proc. 4th Int. Semantic Web Conf. (ISWC2005).*
[4] P. Groot, H. Stuckenschmidt, and H. Wache, "Approximating description logic classification for semantic web reasoning," in *Proc. ESWC2005.*
[5] C. Hurtado, A. Poulovassilis, and P. Wood, "A relaxed approach to RDF querying," in *Proc. 5th Int. Semantic Web Conf. (ISWC-2006).*
[6] J. Z. Pan and E. Thomas, "Approximating OWL-DL ontologies," in *Proc. 22nd Conf. Artificial Intelligence (AAAI-2007)*, pp. 1434–1439.
[7] Y. Ren, J. Z. Pan, and Y. Zhao, "Soundness preserving approximation for TBox reasoning," in *Proc. 25th AAAI Conf. (AAAI2010).*
[8] G. Stoilos, G. Stamou, J. Pan, V. Tzouvaras, and I. Horrocks, "Reasoning with very expressive fuzzy description logics," *J. Artif. Intell. Res.*, vol. 30, no. 5, pp. 273–320, 2007.
[9] G. Stoilos, N. Simou, G. Stamou, and S. Kollias, "Uncertainty and the semantic web," *IEEE Intell. Syst.*, vol. 21, no. 5, pp. 84–87, 2006.
[10] G. Stoilos, G. Stamou, V. Tzouvaras, J. Pan, and I. Horrocks, "Fuzzy owl: Uncertainty and the semantic web," in *Proc. Int. Workshop OWL: Experiences and Directions*, vol. 280, Citeseer, 2005.
[11] F. Bobillo, M. Delgado, and J. Gómez-Romero, "Delorean: A reasoner for fuzzy OWL 2," *Expert Syst. Applicat.*, 2011.
[12] U. Straccia, "Answering vague queries in fuzzy DL-Lite," in *Proc. 11th Int. Conf. Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-2006)*, Citeseer, pp. 2238–2245.
[13] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "DL-Lite: Tractable description logics for ontologies," in *Proc. AAAI* 2005.
[14] J. Z. Pan, G. Stamou, G. Stoilos, S. Taylor, and E. Thomas, "Scalable querying services over fuzzy ontologies," in *Proc. 17th Int. World Wide Web Conf. (WWW2008).*
[15] G. Acampora, V. Loia, and M. Gaeta, "Exploring e-learning knowledge through ontological memetic agents," *IEEE Comput. Int. Mag.*, vol. 5, no. 2, pp. 66–77, 2010.
[16] A. Passant, "Using ontologies to strengthen folksonomies and enrich information retrieval in weblogs," in *Proc. Int. Conf. Weblogs and Social Media (ICWSM2007).*
[17] Y. Ren, G. Gröner, J. Lemcke, T. Rahmani, A. Friesen, Y. Zhao, J. Z. Pan, and S. Staab. (2011). *Process refinement validation and explanation with ontology reasoning,* Univ. Aberdeen, Univ. Koblenz-Landau and AP AG, Tech. Rep. [Online]. Available: http://www.abdn.ac.uk/_csc280/pub/ProcessRefinement.pdf
[18] Y. Zhao, C. Wende, J. Z. Pan, E. Thomas, G. Groener, N. Jekjantuk, Y. Ren, and T. Walter, "D3.4—Guidance tools for language transformations," *MOST Project, Project Deliverable ICT216691/UNIABDN/WP3-D4/D/PU/b1*, 2010.
[19] Y. Ren. (2010). *Syntactic approximation in PDDSL: A completeness guarantee,* Univ. Aberdeen, Tech. Rep. [Online]. Available: http://www.abdn.ac.uk/ csc280/TR/pddsl.pdf
[20] Y. Ren, J. Z. Pan, and Y. Zhao, "Closed world reasoning for OWL2 with NBox," *J. Tsinghua Sci. Technol.*, no. 6, 2010.
[21] E. Thomas, J. Z. Pan, and Y. Ren, "TrOWL: Tractable OWL 2 reasoning infrastructure," in *Proc. Extended Semantic Web Conf. (ESWC2010).*
[22] Z. Wang, K. Wang, R. Topor, and J. Z. Pan, "Forgetting concepts in DL-Lite," in *Proc. 5th European Semantic Web Conf. (ESWC2008)* .
[23] Y. Ren, J. Z. Pan, and Y. Zhao, "Towards scalable reasoning on ontology streams via syntactic approximation," in *Proc. ISWC Workshop on Ontology Dynamics (IWOD2010).*
[24] Y. Ren and J. Z. Pan, "Optimising ontology stream reasoning with truth maintenance system," in *Proc. ACM Conf. Information and Knowledge Management (CIKM 2011)* .
[25] J. Z. Pan, E. Thomas, and D. Sleeman, "ONTOSEARCH2: Searching and querying web ontologies," in *Proc. WWW/Internet* 2006, pp. 211–218.
[26] E. Thomas, J. Z. Pan, and D. Sleeman, "ONTOSEARCH2: Searching ontologies semantically," in *Proc. OWL: Experiences and Directions (OWL-ED2007).*
[27] J. Z. Pan, S. Taylor, and E. Thomas, "Reducing ambiguity in tagging systems with folksonomy search expansion," in *Proc. 6th European Semantic Web Conf. 2008 (ESWC2009).*
[28] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.
[29] Y. Ren, J. Z. Pan, and Y. Zhao, "Towards soundness preserving approximation for ABox reasoning of OWL2," in *Proc. Description Logics Workshop (DL2010).*
[30] F. Baader, C. Lutz, and B. Suntisrivaraporn, "Is tractable reasoning in extensions of the description logic EL useful in practice?" in *Proc. Int. Workshop on Methods for Modalities (M4M-05)* .
[31] F. Baader, S. Brandt, and C. Lutz, "Pushing the EL Envelope," in *Proc. 19th Int. Joint Conf. Artificial Intelligence (IJCAI* 2005).
[32] Y. Zhao, J. Z. Pan, and Y. Ren, "Implementing and evaluating a rule-based approach to querying regular EL+ ontologies," in *Proc. Int. Conf. Hybrid Intelligent Systems (HIS2009).*
[33] A. Friesen, J. Lemcke, D. Oberle, and T. Rahmani, "D6.4—Evaluation of case-study," *MOST Project, Project Deliverable ICT216691/SAP/WP6-D4/D/RE/b1*, 2010.
[34] M. Kasztelnik, K. M. Miksa, and P. Sabina, "D5.4—Evaluation of case-study," *MOST Project, Project Deliverable ICT216691/CMR/WP5-D4/D/RE/b1*, Feb. 2010.

Reasoning with Large Scale Ontologies in Fuzzy *pD** Using MapReduce



Chang Liu, *Shanghai Jiao Tong University, CHINA*

Guilin Qi, *Southeast University, CHINA*

Haofen Wang and Yong Yu, *Shanghai Jiao Tong University, CHINA*

***Abstract*—**The MapReduce framework has proved to be very efficient for data-intensive tasks. Earlier work has successfully applied MapReduce for large scale RDFS/OWL reasoning. In this paper, we move a step forward by considering scalable reasoning on semantic data under fuzzy pD^* semantics (i.e., an extension of OWL pD^* semantics with fuzzy vagueness). To the best of our knowledge, this is the first work to investigate how MapReduce can be applied to solve the scalability issue of fuzzy reasoning in OWL. While most of the optimizations considered by the existing MapReduce framework for pD^* semantics are also applicable for fuzzy pD^* semantics, unique challenges arise when we handle the fuzzy information. Key challenges are identified with solution proposed for each of these challenges. Furthermore, a prototype system is implemented for the evaluation purpose. The experimental results show that the running time of our system is comparable with that of WebPIE, the state-of-the-art inference engine for scalable reasoning in pD^* semantics.

Digital Object Identifier 10.1109/MCI.2012.2188589

Date of publication: 13 April 2012

I. Introduction

The Resource Description Framework (RDF) [1] is one of the major representation standards for the Semantic Web. RDF Schema (RDFS) [2] is used to describe vocabularies used in RDF descriptions. However, RDF and RDFS only provide a very limited expressiveness. In [3], a subset of Ontology Web Language (OWL) [4] vocabulary (e.g., owl:sameAs) was introduced, which extends the RDFS semantics to the pD^* fragment of OWL. The OWL pD^* fragment provides a complete set of entailment rules, guaranteeing that the entailment relationship can be determined within polynomial time under a non-trivial condition (if the target graph is ground). It has become a very promising ontology language for the Semantic Web as it trades off the high computational complexity of OWL Full and the limited expressiveness of RDFS.

Recently, there has been an increasing interest in extending RDF to represent vague information on the Web. Fuzzy RDF allows us to state a triple is true to a certain degree. For example, (*Tom, eat, pizza*) is true with a degree which is at least 0.8. Fuzzy RDFS [5] can handle

A preliminary version of this paper appeared in: C. Liu, G. Qi, H. Wang, Y. Yu. "Large Scale Fuzzy *pD** Reasoning using MapReduce", Presented at the 10th International Semantic Web Conference, Bonn, Germany, 23-27, October, 2011.

1556-603X/12/$31.00©2012IEEE

© IMAGESTATE

class hierarchy and property hierarchy with fuzzy degree. However, both Fuzzy RDF and fuzzy RDFS have limited expressive power to represent information in some real life applications of ontologies, such as biomedicine and multimedia. Therefore, in [6], we extended the OWL pD^* fragment with fuzzy semantics to provide more expressive power than fuzzy RDF(S). However, none of these works focuses on the efficiency issue of the reasoning problem. Since fuzzy RDFS semantics and fuzzy pD^* semantics are focussed on handling large scale semantic data, it is critical to provide a scalable reasoning algorithm for them.

Earlier works (e.g. [7] and [8]) have proved that MapReduce [9] is a very efficient framework to handle the computation of the *closure* of a RDF graph containing up to 100 billion triples under RDFS and pD^* semantics. As such, MapReduce may be helpful to scalable reasoning in fuzzy RDFS semantics and fuzzy pD^* semantics. It turns out that this is a non-trivial problem as the computation of the closure under fuzzy RDFS and fuzzy pD^* semantics requires the computation of the *Best Degree Bound (BDB)* of each triple. The BDB of a triple is the greatest lower bound of the fuzzy degrees of this triple in the closure of a fuzzy RDFS (or fuzzy pD^*) graph. Although most of the optimizations considered by the existing MapReduce framework for pD^* semantics are also applicable for fuzzy pD^* semantics, unique challenges arise when we handle the fuzzy information. For example, [8] proposed an algorithm to handle `TransitiveProperty`. In the fuzzy case, however, we will show in Section III-D3 that calculating the `TransitiveProperty` closure is essentially a variation of the all-pairs shortest path calculation problem, and the algorithm proposed in [8] cannot deal with this problem. Instead efficient algorithms are required to deal with this problem.

In this paper, the proposal is to to build a scalable and efficient reasoning engine for computing closure of a fuzzy RDF graph under fuzzy pD^* semantics. Firstly, a reasoning algorithm is recommended for fuzzy pD^* semantics based on the MapReduce framework. Thereafter MapReduce algorithms is presented for fuzzy pD^* rules together with some novel optimizations. Finally,

TABLE 1 Fuzzy RDFS entailment rules.

CONDITION	CONCLUSION
f-rdfs1	(*v*, *p*, *l*)[*n*]	(*b*$_l$, type, Literal) [1]
f-rdfs1X	(*b*$_l$, type, Literal)[*n*]	(*b*$_l$, type, Literal) [1]
f-rdfs2	(*p*, domain, *u*)[*n*] (*v*, *p*, *w*)[*m*]	(*v*, type, *u*)[*n*⊗*m*]
f-rdfs3	(*p*, range, *w*)[*n*] (*v*, *p*, *w*)[*m*]	(*v*, type, *w*)[*n*⊗*m*]
f-rdfs4a	(*v*, *p*, *w*)[*n*]	(*v*, type, Resource)[1]
f-rdfs4b	(*v*, *p*, *w*)[*n*]	(*w*, type, Resource)[1]
f-rdfs5	(*v*, subPropertyOf, *w*)[*n*] (*w*, subPropertyOf, *u*)[*m*]	(*v*, subPropertyOf, *u*)[*n*⊗*m*]
f-rdfs6	(*v*, type, Property)[1]	(*v*, subPropertyOf, *v*)[1]
f-rdfs7x	(p, subPropertyOf, *q*)[*n*] (*v*, *p*, *w*)[*m*]	(*v*, *q*, *w*)[*n*⊗*m*]
f-rdfs8	(v, type, Class)[*n*]	(*v*, subClassOf, Resource)[*n*]
f-rdfs9	(*v*, subClassOf, *w*)[*n*] (*u*, type, *v*)[*m*]	(*u*, type, *w*)[*n*⊗*m*]
f-rdfs10	(*v*, type, Class)[*n*]	(*v*, subClassOf, *v*)[1]
f-rdfs11	(*v*, subClassOf, *w*)[*n*] (*w*, subClassOf, *u*)[*m*]	(*v*, subClassOf, *u*)[*n*⊗*m*]
f-rdfs12	(*v*, type, ContainerMembershipProperty)[*n*]	(*v*, subPropertyOf, member)[*n*]
f-rdfs13	(*v*, type, Datatype)[*n*]	(*v*, subClassOf, Literal)[1]

a prototype system is implemented to evaluate all these optimizations, and conduct extensive experiments. The experimental results also show that the running time of this proposed system is comparable with that of WebPIE [8], the state-of-the-art inference engine for scalable reasoning in pD^* semantics. This paper is extended and revised from our conference paper [10].

The rest of the paper is organized as follows. In Section II, we introduce the background knowledge about fuzzy pD^* reasoning and the MapReduce framework. Then we propose our MapReduce algorithm for fuzzy pD^* reasoning in Section III. The experiment results are provided in Section IV. Discussions of related work can be found in Section V. Finally we conclude this paper in Section VI.

II. Background Knowledge

In this section, the fuzzy pD^* entailment rule set is introduced in Section II-B, followed by an explanation of the MapReduce framework for reasoning in OWL pD^* fragment in Section II-C.

A. RDF, RDFS and pD* Semantics

An RDF [1] graph is defined as a subset of the set $\mathbf{UB} \times \mathbf{U} \times \mathbf{UBL}$, where **U**, **B** and **L** represent the set of *URI references, Blank nodes*, and *Literals respectively*. The element (s, p, o) of an RDF graph is a *triple*. Intuitively, we use a triple to represent a statement. For example, we can use a triple $(Tom, eat, pizza)$ to state that Tom eats pizza. RDF properties may be considered as relationships between resources. RDF, however, provides no mechanisms for describing the relationships between these properties and other resources. RDF Schema (RDFS) [2], which plays the role of the RDF vocabulary description language, defines classes and properties that may be used to describe classes, properties and other resources.

However, RDF and RDFS only provide a very limited expressiveness. In [3], a subset of Ontology Web Language (OWL) [4] vocabulary (e.g., owl:sameAs) was introduced, which extends the RDFS semantics to the pD^* fragment of OWL. Unlike the standard OWL (DL or Full) semantics which provides the full *"if and only if"* semantics, the OWL pD^* fragment follows RDF(S)'s *"if"* semantics. That is, the OWL pD^* fragment provides a complete set of entailment rules, which guarantees that the entailment relationship can be determined within polynomial time under a non-trivial condition (if the target graph is ground). The OWL vocabulary supported by pD^* semantics include FunctionalProperty, InverseFunctionalProperty, SymmetricProperty, TransitiveProperty, sameAs, inverseOf, equivalentClass, equivalentProperty, hasValueOf, someValuesFrom and allValuesFrom.

B. Fuzzy RDFS and pD* Reasoning

A fuzzy RDF graph is a set of fuzzy triples which are in the form of $t[n]$. Here t is a triple, and $n \in (0, 1]$ is the fuzzy degree of t.

Fuzzy RDFS semantics extends RDFS semantics with fuzzy semantics. The complete and sound entailment rule set of fuzzy RDFS is listed in Table 1. However, fuzzy RDFS has limited expressive power to represent information in some real life applications of ontologies, such as biomedicine and multimedia. Therefore, fuzzy pD^* semantics was proposed in [6] to extends pD^* semantics with fuzzy semantics. To handle the additional vocabularies, we use the fuzzy *P*-entailment rule set listed in Table 2 along with the fuzzy RDFS rules. The notion of a (partial) closure can be easily extended to the fuzzy case.

One of the key notions in both fuzzy RDFS and fuzzy pD^* semantics is called the *Best Degree Bound (BDB)* of a triple. The BDB n of an arbitrary triple t from a fuzzy RDF graph G under fuzzy RDFS semantics is defined to be the largest fuzzy degree n such that $t[n]$ can be derived from G by applying the fuzzy RDFS-entailment rules, or 0 if no such fuzzy triple can be derived. The definition of BDB of a triple under fuzzy pD^* semantics can be similarly defined.

C. MapReduce Framework

MapReduce is a programming model introduced by Google for large scale data processing [9]. A MapReduce program is composed of two user-specified functions, *map* and *reduce*. When the input data is appointed, the *map* function scans the input data and generates intermediate key/value pairs. Then all pairs of key and value are partitioned according to the key and each partition is processed by a *reduce* function.

To illustrate the key factor affecting the performance of the MapReduce program, we briefly introduce the implementation of Hadoop[1] which is an open source MapReduce

[1] http://hadoop.apache.org/

implementation. When a MapReduce program is submitted to the Hadoop job manager, the job manager will first split the input data into several pieces. Then a mapper will be created for each input split, and assigned to an empty map slot (machine). The mappers will generate a lot of key/value pairs. Then the job manager will assign a reducer to each empty reduce slot, and each key/value pair that is outputted by any mapper will be transmitted to the corresponding reducer. When all mappers complete and the data are transferred to the reducers' side, each reducer will first shuffle all the data so that all the key/value pairs with the same key can be grouped in order to be executed together. Then the reducer will process each different key and a list of values at once, and generate the output.

If there is a mapper that is very slow, then the whole program will have to wait for its mapper slot, while other mapper slots which have been completed earlier have to remain idle and cannot be assigned to other mappers. This situation is called skew.

According to these implementations, the following factors will influence the performance of the MapReduce program. They also give important principles to develop efficient MapReduce program.

- On the mapper side, only when all the mappers complete their jobs, then the reducer will start to work. If there is a mapper that is very slow, then the whole program will have to wait for its mapper slot, while other mapper slots which have been completed earlier have to remain idle and cannot be assigned to other mappers. This situation is called *skew*. Since the run time of a mapper is typically linear to the amount of data it is assigned to, this kind of skew is usually caused by unbalanced data partition. In order to improve the performance, we should split the input data as equally as possible.
- Before the reducer is executed, all key/value pairs will be transmitted to the same machine, and a shuffling process must be executed first in order to group key/value pairs with the same key together. This process is the major cost of each MapReduce program. However, for many tasks, the processing algorithm only needs to be executed on the data set in parallel. Thus, the reducers are not needed for these tasks. In this case, the MapReduce framework allows the program to contain only the map side. Thus, the costly data transmission and shuffling process can be avoided to improve the performance.
- On the reducer side, there is also a skew problem. The reducer processing a popular key will run very slowly. Therefore, the mappers' output keys should be carefully designed to ensure that the sizes of all partitions is relatively balanced.
- There is no restriction on how many values there are for the same key. We usually cannot assume that all values sharing a same key can be loaded into the memory. If we force the reducer to load all such values into the memory, in the worst case, it will result in an OutOfMemory error. Thus the reducer should operate on a stream of values instead of a set of values.

III. MapReduce Algorithms For Fuzzy *pD** Reasoning

In this section, we first illustrate how a MapReduce program can be used to apply a fuzzy rule. Then, we give an overview of the challenges when applying the MapReduce framework to

TABLE 2 Fuzzy *P*-entailment rules.

CONDITION	CONCLUSION	
f-rdfp1	(*p*, type, FunctionalProperty) [*n*] (*u*, *p*, *v*) [*m*] (*u*, *p*, *w*) [*l*]	(*v*, sameAs, *w*)[*l*⊗*m*⊗*n*]
f-rdfp2	(*p*, type, InverseFunctionalProperty) [*n*] (*u*, *p*, *w*)[*m*] (*v*, *p*, *w*)[*l*]	(*u*, sameA, *v*)[*l*⊗*m*⊗*n*]
f-rdfp3	(*p*, type, SymmetricProperty)[*n*] (*v*, *p*, *w*)[*m*]	(*w*, *p*, *v*)[*n*⊗*m*]
f-rdfp4	(*p*, type, TransitiveProperty)[*n*] (*u*, *p*, *v*)[*m*] (*v*, *p*, *w*)[*l*]	(*u*, *p*, *w*)[*n*⊗*m*⊗*l*]
f-rdfp5(ab)	(*v*, *p*, *w*)[*n*]	(*v*, sameAs, *v*)[1], (*w*, sameAs, *w*)[1]
f-rdfp6	(*v*, sameAs, *w*)[*n*]	(*w*, sameAs, *v*)[*n*]
f-rdfp7	(*u*, sameAs, *v*)[*n*] (*v*, sameAs, *w*)[*m*]	(*u*, sameAs, *w*)[*n*⊗*m*]
f-rdfp8ax	(*p*, inverseOf, *q*)[*n*] (*v*, *p*, *w*)[*m*]	(*w*, *q*, *v*)[*n*⊗*m*]
f-rdfp8bx	(*p*,inverseOf, *q*)[*n*] (*v*, *q*, *w*)[*m*]	(*w*, *p*, *v*)[*n*⊗*m*]
f-rdfp9	(*v*, type, Class)[*n*] (*v*, sameAs, *w*)[*m*]	(*v*, subClassOf, *w*)[*m*]
f-rdfp10	(*p*, type, Property)[1] (*p*, sameAs, *q*)[*m*]	(*p*, subPropertyOf, *q*)[*m*]
f-rdfp11	(*u*, *p*, *v*)[*n*] (*u*, sameAs, *u*´)[*m*] (*v*, sameAs, *v*´)[*l*]	(*u*´, *p*, *v*´)[*n*⊗*m*⊗*l*]
f-rdfp12(ab)	(*v*, equivalentClass, *w*)[*n*] ⇒(*v*, subClassOf, *w*)[*n*], (*w*, subClassOf, *w*)[*n*]	
f-rdfp12c	(*v*, subClassOf, *w*)[*n*] (*w*, subClassOf, *v*)[*m*]	(*v*, equivalentClass, *w*)[min(*n*, *m*)]
f-rdfp13(ab)	(*v*, equivalentProperty, *w*)[*n*]⇒(*v*, subPropertyOf, *w*)[*n*],	(*w*, subPropertyOf, *w*)[*n*]
f-rdfp13c	(*v*, subPropertyOf, *w*)[*n*] (*w*, subPropertyOf, *v*)[*m*]	(*v*, equivalentClass, *w*)[min(*n*, *m*)]
f-rdfp14a	(*v*, hasValueOf, *w*)[*n*] (*v*, onProperty, *p*)[*m*] (*u*, *p*, *w*)[*l*]	(*u*, type, *v*)[*n*⊗*m*⊗*l*]
f-rdfp14bx	(*v*, hasValueOf, *w*)[*n*] (*v*, onProperty, *p*)[*m*] (*u*, type, *v*)[*l*]	(*u*, *p*, *w*)[*n*⊗*m*⊗*l*]
f-rdfp15	(*v*, someValueFrom, *w*)[*n*] (*v*, onProperty, *p*)[*m*] (*u*, *p*, *x*)[*l*] (*x*, type, *w*)[*k*]	(*u*, type, *v*)[*n*⊗*m*⊗*l*⊗*k*]
f-rdfp16	(*v*, allValuesfrom, *w*)[*m*] (*v*, onProperty, *p*)[*n*] (*u*, type, *v*)[*l*] (*u*, *p*, *x*)[*k*]	(*x*, type, *w*)[*n*⊗*m*⊗*l*⊗*k*]

ALGORITHM 1: Map function for rule f-rdfs2.

```
Input: key, triple
1: if triple.predicate == 'domain' then
2: emit({p=triple.subject}, {flag='L', u=triple.object,
     n=triple.degree};
3: end if
4: emit({p=triple.predicate}, {flag='R', v=triple.subject,
     m=triple.degree};
```

fuzzy pD^* reasoning. Finally, we present our solutions to handle these challenges.

A. Naive MapReduce Algorithms for Fuzzy Rules

We consider rule f-rdfs2 to illustrate our naive MapReduce algorithms:

$$(p, \texttt{domain}, u)[n], (v, p, w)[m] \Rightarrow (v, \texttt{type}, u)[n \otimes m]$$

In this rule, we should find all fuzzy triples that are either in the form of $(p, \texttt{domain}, u)[n]$ or in the form of $(v, p, w)[m]$. A join should be performed over the variable p. The *map* and *reduce* functions are given in Algorithms 1 and 2 respectively. In the *map* function, when a fuzzy triple is in the form of $(p, \texttt{domain}, u)[n]$ (or $(v, p, w)[m]$), the mapper emits p as the key and u (or v) along with the degree n (or m) as the value. The reducer can use the flag in the mapper's output value to identify the content of the value. If the flag is 'L' (or 'R'), the content of the value is the pair (u, n) (or the pair (v, m)). The reducer uses two sets to collect all the u, n pairs and the v, m pairs. After all pairs are collected, the reducer enumerates pairs (u, n) and (v, m) to generate $(u, \texttt{type}, v)[n \otimes m]$ as output.

ALGORITHM 2: Reduce function for rule f-rdfs2.

```
Input: key, iterator values
1: unSet.clear( );
2: vmSet.clear( );
3: for value ∈ values do
4: if value.flag == 'L' then
5: unSet.update(value.u, value.n);
6: else
7: vmSet.update(value.v, value.m);
8: end if
9: end for
10: for i ∈ unSet do
11: for j ∈ vmSet do
12: emit(null, new FuzzyTriple(i.u, 'type', j.v, i.n⊗j.m));
13: end for
14: end for
```

B. Challenges

Even though the fuzzy pD^* entailment rules are quite similar to the pD^* rules, several difficulties arose when we calculated the BDB for each triple by applying the MapReduce framework. The summary of these challenges are as follows:

1) Efficient Implementation

The Naive implementation will encounter some problems and violate the principles to develop efficient MapReduce program introduced in Section II-C. For example, the program for rule f-rdfs2 shown in the last subsection introduces an unnecessary shuffling process and treats the values in the *reduce* function as a set instead of a stream. We will discuss a solution to tackle this problem in Section III-C2 and Section III-D2.

2) Ordering the Rule Applications

In fuzzy pD^* semantics, the reasoner might produce a duplicated triple with different fuzzy degrees before the BDB of such a triple is derived. For example, if the data set contains a fuzzy triple $t[m]$, when a fuzzy triple $t[n]$ with $n > m$ is derived, a duplicated triple is generated. In this case, we should employ a deletion program to reproduce the data set to ensure that for any triple, among all its extended fuzzy triples that can be derived from the MapReduce algorithms, only the one with maximal degrees are kept in the data set. Different orders of rule applications will result in different number of such duplicated triples. To achieve the best performance, we should choose a proper order to reduce the number of duplicated triples. For instance, the subproperty rule (f-rdfs7x) should be applied before the domain rule (f-rdfs2); and the equivalent class rule (f-rdfp12(abc)) should be considered together with the subclass rule (f-rdfs9, f-rdfs10). A solution for this problem will be discussed in Section III-C1 and section III-D1.

3) Shortest Path Calculation

In fuzzy pD^* semantics, there are three rules, i.e., f-rdfs5 (subproperty), f-rdfs11 (subclass) and f-rdfp4 (transitive property), that require iterative calculations. When we treat each fuzzy triple as a weighted edge in the RDF graph, then calculating the closure by applying these three rules is essentially a variation of the all-pairs shortest path calculation problem. We have to find out efficient algorithms for this problem. We will discuss rules f-rdfs5 and f-rdfs11 in section III-C3 and discuss rule f-rdfp4 in section III-D3.

4) SameAs Rule

The most expensive calculation happens when we process the `sameAs` rules f-rdfp 5(ab), 6, 7, 9, 10 and 11. Fully applying all these rules will result in an extremely large fuzzy triple set, which is unnecessary for the application purpose. In Section III-D4 and Section III-D5, we will discuss how to handle these rules efficiently.

C. Algorithms for Handling Fuzzy RDFS Rules

In this subsection, we present MapReduce algorithms for handling the RDFS rules (rules f-rdfs1 to f-rdfs13). Most of these rules have only one fuzzy triple as a condition so that they can function on its own. These rules are easy to handle. Thus we only considered rules with at least two fuzzy triples in its condition, i.e., rules f-rdfs 2, 3, 5, 7x, 9, 11.

Generally speaking, the reasoning engine should iteratively perform the rules to see if a fixpoint is reached. However, after carefully looking at these fuzzy RDFS rules, we find that it is possible to order the rules so that every rule can be executed only once to generate all conclusions.

1) Ordering Rules

Generally speaking, the reasoning engine should iteratively perform the rules to see if a fixpoint is reached. However, after carefully looking at these fuzzy RDFS rules, we find that it is possible to order the rules so that every rule can be executed only once to generate all conclusions. We have the following four phases of rule application.

In the first phase, rules f-rdfs 5, 6, and 7 are executed to produce the whole property hierarchy. In the second phase, the domain and range rules (f-rdfs2 and f-rdfs3) will be performed. In the third phase, we execute rules f-rdfs9, 10 and 11 to produce the whole class hierarchy. Finally, we apply the remaining rules.

In the first and the third phase, the reasoning engine will need to recursively call a MapReduce algorithm to calculate the complete hierarchy. This will greatly influence its efficiency. This problem will be discussed in the following subsections. As we will show, in each phase, the reasoning engine only needs to call one MapReduce program. Some duplicated triples may be derived after applying rules. After executing all the four phases, we should launch a program to remove them to guarantee that each triple appears in the closure only once.

2) Loading Schema Triples into the Memory

As shown in section III-A, a join can be handled by a MapReduce program. However, there are two drawbacks in this MapReduce program. Firstly, there is a very expensive shuffling process before the reducers are launched. Secondly, in each reducer, we have to load all values of key/value pairs generated by mappers with the same key into the memory. As we have discussed in section II-C, if there are too many values, the reducer can easily fail.

We find that in all these rules, there is a fuzzy triple in the condition that is a schema triple. In practice, we can assume that the number of schema triples is relatively small. Therefore, we can load them into the memory, and perform the join in the mappers so that we can avoid the above two drawbacks.

We illustrate this idea by considering rule f-rdfs2. The *map* function is provided in Algorithm 3. Before each mapper starts to process the data, it will load all triples in the form of $(p, \texttt{domain}, w)[n]$, and store them in *domainProperty* which is a Map (in our implementation, we use a hash table) that maps each p to the list of (w, n) pairs. Then the MapReduce program only has to parallelly enumerate every fuzzy triple $(u, p, v)[m]$to output the fuzzy triples $(u, \texttt{type}, w)[n \otimes m]$. Thus we do not need a *reduce* program.

3) Calculating the `subClassOf` and the `subPropertyOf` Closure

Rules f-rdfs5, 6, and 7, and rules f-rdfp12(abc) are relevant to the `subPropertyOf` property while rules f-rdfs9, 10, and 11, and rules f-rdfp13(abc) are relevant to the `subClassOf` property. We only discuss rules that are relevant to `subClassOf` as rules that are relevant to the `subPropertyOf` property can be handled similarly. Since f-rdfs10 only derives a triple $(v, \texttt{subClassOf}, v)[1]$ which will have no affect on other rules, we only consider rules f-rdfs5, 7 and f-rdfp12(abc).

We call the triples in the form of $(u, \texttt{subClassOf}, v)[n]$ to be `subClassOf` triples. It is easy to see that rule f-rdfs11 needs to recursively calculate the transitive closure of `subClassOf` triples. However, since they are schema triples, as we have discussed previously, we can load them into the memory so that we can avoid the reiterative execution of the MapReduce program of rule f-rdfs11. In fact, we can see that calculating the `subClassOf` closure by applying rule f-rdfs11 is indeed a variation of the all-pairs shortest path calculation problem, according to the following property:

Property 1

For any fuzzy triple in the form of $(u, \texttt{subClassOf}, v)[n]$ that can be derived from the original fuzzy RDF graph by only applying rule f-rdfs11, there must be a chain of classes $w_0 = u, w_1, \ldots, w_k = v$ and a list of fuzzy degrees $d_1, \ldots, d_k$ such

ALGORITHM 3: Map function for rule f-rdfs2 (loading schema triples).

```
Input: key, triple
1: if domainProperty.containsKey(triple.predicate) then
2:   for (u,v) ∈ domainProperty.get(triple.predicate) do
3:     emit(new FuzzyTriple(triple.subject, type, u, n⊗triple.
         degree));
4:   end for
5: end if
```

ALGORITHM 4: Calculate the `subClassOf` closure.

```
1: for k ∈ I do
2:   for i ∈ I and w(i, k) > 0 do
3:     for j ∈ I and w(k, j) > 0 do
4:       if w(i,k) ⊗ w(k, j) > w(i, j) then
5:         w(i, j) = w(i, k) ⊗ w(k, j);
6:       end if
7:     end for
8:   end for
9: end for
```

that for every $i = 1, 2, \ldots, k$, $(w_{i-1}, \texttt{subClassOf}, w_i)[d_k]$ is in the original fuzzy graph and $n = d_1 \otimes d_2 \otimes \ldots \otimes d_k$.

This property is easy to prove by induction.[2] Indeed, from fuzzy triples $(w_0, \textbf{subClassOf}, w_i)[d_1 \otimes \ldots \otimes d_i]$ and $(w_i, \textbf{subClassOf}, w_k)[d_{i+1} \otimes \ldots \otimes d_k]$, we can derive $(w_0, \textbf{SubClassOf}, w_k)[d_1 \otimes \ldots \otimes d_k]$ using f-rdfs11.

So we can use the Floyd-Warshall style algorithm given in Algorithm 4 to calculate the closure. In the algorithm, I is the set of all the classes, and $w(i, j)$ is the fuzzy degree of triple $(i, \texttt{subClassOf}, j)$. The algorithm iteratively updates the matrix w. When it stops, the subgraph represented by the matrix $w(i, j)$ is indeed the `subClassOf` closure.

The worst-case running complexity of the algorithm is $O(|I|^3)$, and the algorithm uses $O(|I|^2)$ space to store the matrix w. When $|I|$ goes large, this is unacceptable. However, we can use nested hash map instead of 2-dimension arrays to only store the positive matrix items. Furthermore, since $0 \otimes n = n \otimes 0 = 0$, in line 2 and line 3, we only enumerate those i and j where $w(k, i) > 0$ and $w(k, j) > 0$. In this case, the running time of the algorithm will be greatly reduced.

ALGORITHM 5: Fuzzy pD^* reasoning.

```
1: first_time = true;
2: while true do
3:     derived = apply_fD_rules( );
4:     if derived == 0 and not first_time then
5:         break;
6:     end if;
7:     repeat
8:         derived = apply_fp_rules( );
9:     until derived == 0;
10:    first_time = false;
11:  end while
```

[2] The full proof can be found in our technique report which is available at http://apex.sjtu.edu.cn/apex_wiki/fuzzypd.

After the `subClassOf` closure is computed, rules f-rdfs9 and f-rdfs11 can be applied only once to derive all the fuzzy triples: for rule f-rdfs9 (or f-rdfs11), when we find a fuzzy triple $(i, \texttt{type}, v)[n]$ (or $(i, \texttt{subClassOf}, v)[n]$), we enumerate all classes j with $w(v, j) > 0$ and output a fuzzy triple $(i, \texttt{type}, j)[n \otimes w(v, j)]$ (or $(i, \texttt{subClassOf}, j)[n \otimes w(v, j)]$).

For rule f-rdfp12(abc), since `equivalentClass` triples are also schema triples, we load them into the memory and combine them into the `subClassOf` graph. Specifically, when we load a triple $(i, \texttt{equivalentClass}, j)[n]$ into the memory, *if* $n > w(i, j)$ (*or* $n > w(j, i)$), we update $w(i, j)$ (or $w(j, i)$) to be n. After the closure is calculated, two fuzzy triples $(i, \texttt{equivalentClass}, j)[n]$ and $(j, \texttt{equivalentClass}, i)[n]$ are output for each pair of classes $i, j \in I$, if $n = \min(w(i, j), w(j, i)) > 0$.

D. Algorithms for Fuzzy pD^* Rules

For fuzzy pD^* entailment rules, there is no ordering that can avoid iterative execution. Thus we have to iteratively apply the rules until a fixpoint is reached. In the following, we first give the main reasoning algorithm, we then discuss the optimizations for fuzzy pD^* rules.

1) Overview of the Reasoning Algorithm

Our main reasoning algorithm is Algorithm 5, which can be separated into two phases: the first phase (line 3) applies the fuzzy D rules (from f-rdfs1 to f-rdfs13); and the second phase (lines 7 to line 9) applies the fuzzy P-entailment rules (from rdfp1 to rdfp16). Since some fuzzy P-entailment rules may generate some fuzzy triples having effect on fuzzy D rules, we execute these two phases iteratively (line 2 to line 11) until a fixpoint is reached (line 4 to line 6).

For the fuzzy P-entailment rules, there is no way to avoid a fixpoint iteration. So we employ an iterative algorithm to calculate the closure under P-entailment rules. In each iteration, the program can be separated into five steps. In the first step, all non-iterative rules (rules f-rdfp1, 2, 3, 8) are applied. The second step processes the transitive property (rule f-rdfp4) while the `sameAs` rules (rule f-rdfp6, 7, 10, 11) are applied in the third step. The rules related to `hasValue` are treated in the fourth step, because we can use the optimizations for reasoning in OWL pD^* fragment to compute the closure of these rules in a non-iterative manner. The `someValuesFrom` and `allValuesFrom` rules are applied in the fifth step which needs a fixpoint iteration. In the first and last phases, there are rules that require more than one join, which turns out to be very inefficient. We first discuss how to avoid these multiple joins in Section III-D2. We then discuss the solution to deal with transitive property in Section III-D3. Finally the solution to tackle `sameAs` rules will be discussed in Section III-D4 and Section III-D5.

2) Multiple Joins

For fuzzy P-entailment rules that have more than two triples in their condition, more than one join should be performed in

order to derive the result of such a rule. These rules are f-rdfp 1, 2, 4, 11, 14a, 14bx, 15 and 16. Rule f-rdfp4 and f-rdfp11 will be discussed in the next two subsections, thus we only consider the remaining six rules. After carefully examining these rules, we find that most of the fuzzy triples in the condition part of these rules are schema triples, and the number of assertion triples in each single rule is at most two. Therefore, we can leverage the technique discussed in the last section, i.e., loading schema triples into the memory.

We illustrate this idea by considering rule f-rdfp15:

$$(v, \texttt{someValueFrom}, w)[n], (v, \texttt{onProperty}, p)[m], (u, p, x)[l], (x, \texttt{type}, w)[k] \Rightarrow (u, \texttt{type}, v)[n \otimes m \otimes l \otimes k].$$

In the condition of this rule, $(v, \texttt{someValueFrom}, w)[n]$ and $(v, \texttt{onProperty}, p)[m]$ are schema triples. Thus, before the mappers are launched, they are loaded into the memory and stored in two Maps, e.g. two hash tables: someValuesFrom maps each w to a list of (v, n) pairs, and onProperty maps each p to a list of (v, m) pairs. The *map* function is given in Algorithm 6. In the mapper, we perform two joins. Lines 1 to 5 in Algorithm 6 calculate the join over $(v, \texttt{someValueFrom}, w)[n]$and $(x, \texttt{type}, w)[k]$, resulting in a table with schema $(x, v, n \otimes k)$, and lines 6 to 10 calculate the join over $(v, \texttt{onProperty}, p)[m]$ and $(u, p, x)[l]$, resulting in a table with schema $(x, v, u, m \otimes l)$. Then the mappers output these two tables using (x, v) as the key so that the *reduce* function which is given in Algorithm 7 can perform the final join over these two tables.

ALGORITHM 6: Map function for rule f-rdfp15.

Input: key, triple

```
1:  if triple.predicate == 'type' and sameValuesFrom.
    containsKey(triple.object) then
2:    for (v,n) ∈ sameValuesFrom.get(triple.ojbect) do
3:      emit({triple.subject, v}, {flag='L', degree=n ⊗triple.
        degree});
4:    end for
5:  end if
6:  if onProperty.containsKey(triple.predicate) then
7:    for (v,m) ∈ onProperty.get(triple.predicate) do
8:      emit({triple.object, v}, {flag='R', u=triple.subject,
        degree= m ⊗triple.degree});
9:    end for
10: end if
```

3) Transitive Closure for `TransitiveProperty`

The computation of the transitive closure by applying rule f-rdfp4 is essentially calculating the all-pairs shortest path on the instance graph. To see this point, we consider the following property:

Property 2

Suppose there is a fuzzy triple $(p, \texttt{Type}, \texttt{TransitiveProperty})[n]$in the fuzzy RDF graph G, and $(a, p, b)[m]$ is a fuzzy triple that can be derived from G using only rule f-rdfp4. Then there must be a chain of instances $u_0 = a, u_1, \ldots, u_k = b$ and a list of fuzzy degrees $d_1, \ldots, d_k$ such that $m = d_1 \otimes n \otimes d_2 \otimes \ldots \otimes n \otimes d_k$, and for every $i = 1, 2, \ldots, k$, $(u_{i-1}, p, u_p)[d_i]$ is in the original fuzzy RDF graph. Furthermore, in one of such chains, $u_i \neq u_j$, if $i \neq j$and $i, j \geq 1$.

This property is easy to prove by induction.[3] Indeed, from fuzzy triples $(u_0, p, u_i)[d_1 \otimes \ldots \otimes d_i \otimes n^i]$, $(u_i, p, w_k)[d_{i+1} \otimes \ldots \otimes d_k \otimes n^{k-i}]$ and $(p, \texttt{type}, \texttt{TransitiveProperty})[n]$, we can derive $(u_0, p, w_k)[d_1 \otimes \ldots \otimes d_k \otimes n^{k+1}]$ using f-rdfp4. We use an iterative algorithm to calculate this transitive closure. In each iteration, we execute a MapReduce program using Algorithm 8 as the *map* function and Algorithm 9 as the *reduce* function.

We use `getTransitiveDegree(p)` function to get the maximal *degree* such that $(p, \texttt{type}, \texttt{TransitiveProperty})[degree]$ is in the graph. Since these triples are schema triples, we can load them into the memory before the mappers and reducers are executed. Suppose l is the number of iterations that the transitive closure calculation algorithm already executes. In the *map* function, l is required as input. For any fuzzy triple $(a, p, b)[m]$, there is at least one chain $u_0 = a, u_1, \ldots, u_k = b$ according to Property 2. We use variable *length* to indicate the length of the shortest chain of $(a, p, b)[m]$. At the beginning of the algorithm, for every triple $(a, p, b)[n]$ in the fuzzy RDF graph, *length* is assigned to be one.

ALGORITHM 7: Reduce function for rule f-rdfp15.

Input: key, iterator values

```
1:  nmax = 0;
2:  mSet.clear( );
3:  for value ∈ values do
4:    if value.flag == 'L' then
5:      nmax = value.degree > nmax ? value.degree: nmax;
6:    else
7:      mSet.update(value.u, value.m);
8:    end if
9:  end for
10: for (u,m) ∈ mSet do
11:   emit(null, new FuzzyTriple(u, 'type', key.v, nmax⊗m));
12: end for
```

[3] The full proof can be found in our technique report which is available at http://apex.sjtu.edu.cn/apex_wiki/fuzzypd.

ALGORITHM 8: Map function for rule f-rdfp4.

```
Input: length, triple=(subject, predicate, object)[degree], l
  if getTransitiveDegree(predicate) == 0 then
    return;
  end if
  if length == 2^(l-2) or length == 2^(l-1) then
    emit({predicate, object}, {flag=L, length, subject, degree};
  end if
  if length > 2^(l-2) and length ≤ 2^(l-1) then
    emit({predicate, subject}, {flag=R, length, object, degree};
  end if
```

If $(a,p,b)[m]$ has a chain $u_0,u_1,\ldots,u_k$ with length k, and it can be derived from $(a,p,t)[m_1]$ and $(t,p,b)[m_2]$ in the l-th iteration, then we have $m_1+m_2=m$, $m_1=2^{l-2}$ or 2^{l-1}, and $2^{l-2}<m_2\le 2^{l-1}$. We can show that the integer equation $m_1+m_2=m$ has a unique solution satisfying $m_1=2^{l-2}$ or $m_1=2^{l-1}$, and $2^{l-2}<m_2\le 2^{l-1}$. Thus for such a chain, the triple $(a,p,b)[m]$ will be generated only once. As a consequence, a fuzzy triple will be generated at most as many times as the number of chains it has. In most circumstances, every fuzzy triple will be generated only once.

Furthermore, based on the above discussion, if a fuzzy triple $(a,p,b)[m]$ has a chain with length $2^{l-1}<length\le 2^l$, it will be derived within l iterations. As a consequence, the algorithm will terminate within $\log N$ iterations where N is the number of all instances in the graph.

ALGORITHM 9: Reduce function for rule f-rdfp4.

```
Input: key, iterator values
  left.clear( );
  right.clear( );
  for value ∈ values do
    if value.flag == 'L' then
      left.update(value.subject, {value.degree, value.length});
    else
      right.update(value.object, {value.degree, value.length});
    end if
  end for
  for i ∈ left do
    for j ∈ right do
      newLength = i.length + j.length;
      emit(newLength, new FuzzyTriple(i.subject, key.predicate,
        j.object,
        i.degree ⊗ j.degree ⊗ getTransitiveDegree
        (key.predicate)));
    end for
  end for
```

4) Handling Certain `sameAs` Closure

The rules related to `sameAs` are f-rdfp5(ab), 6, 7, 9, 10 and 11. Rules f-rdfp5(ab) are naive rules which can be implemented directly. The conclusion of Rule f-rdfp9 can be derived by applying rules f-rdfs10 and f-rdfp11. Rule f-rdfp10 allows replacing the predicate with its synonyms. Thus we only consider the rules f-rdfp6 and f-rdfp7, and the following variation of rule f-rdfp11, called f-rdfp11x:

$$(u,p,v)[n],(u,\texttt{sameAs},u')[m],(v,\texttt{sameAs},v')[l],$$
$$(p,\texttt{sameAs},p')[k]\Rightarrow(u',p',v')[n\otimes m\otimes l\otimes k].$$

For convenience, we call a fuzzy triple in the form of $(i,\texttt{sameAs},j)[n]$ a `sameAs` triple. We further call the `sameAs` triples with fuzzy degree 1 the *certain* `sameAs` *triples*, and those with fuzzy degrees less than 1 the *vague* `sameAs` *triples*.

For certain `sameAs` triples, we can employ a technique introduced in [8], called *canonical representation*, to improve the performance. In some real applications, such as the Linking Open Data project, most of the `sameAs` triples are certain when they are used to link different URIs across different datasets. Thus this technique will be useful in practice.

Rules rdfp6 and rdfp7 enforce that `sameAs` is a symmetric and transitive property, thus the `sameAs` closure obtained by applying these two rules is composed of several complete subgraphs. The instances in the same subgraph are all synonyms, so we can assign a unique key, which we call the canonical representation, to all of them. Replacing all the instances by its unique key results in a more compact representation of the RDF graph without loss of completeness of inference.

To achieve this goal, the algorithm can be divided into two phases. In the first phase, we should compute the canonical representation of each instance. In the second phase, we should replace each instance by its canonical representation.

In the first phase, we choose the canonical representation of each instance to be the connected instance with minimal ID. Here, we say instance i is connected with instance j if there is a chain of $i_0,i_1,\ldots,i_k$ where $i_0=i$, $i_k=j$ and $(i_t,\texttt{sameAs},i_{t+1})[1]$ or $(i_{t+1},\texttt{sameAs},i_t)[1]$ belongs to the graph for $t=0,1,\ldots,k-1$. This minimal ID is computed by an algorithm which iteratively call a MapReduce program. After the k-th iteration, the triple set contains all certain sameAs triples $(i,\texttt{sameAs},j)[1.0]$ where i is the instance with minimal ID that is connected with j in the `sameAs` graph and the distance between i and j is at most 2^k. Suppose there are n instances in the `sameAs` graph, then after $\log n$ iterations, the algorithm will find the canonical representations of all instances.

The *map* function and the *reduce* function are given in Algorithm 10 and Algorithm 11 respectively. The mapper processes only certain `sameAs` triples, and after each iteration, the input certain `sameAs` triples will be deleted from the fuzzy triple set. According to our assumption, at the beginning of the k-th iteration, the mapper will only process fuzzy triples in the form of $(u,\texttt{sameAs},v)[1.0]$ where u is the instance with minimal ID

which is connected with v and the distance between u and v is at most 2^{k-1}. When it scans a triple in the form of $(u, \texttt{sameAs}, v)[1.0]$, the mapper will emit both (u, v) and (v, u). Then for each *key*, the reducer will process all values v which is connect with *key* within 2^{k-1} distance. The reducer chooses the canonical representation *min* to be the minimal number in the set containing all values v and the key, and emit a fuzzy triple $(min, \texttt{sameAs}, v)[1.0]$. It is easy to see that *min* should be the minimal ID corresponding to an instance i that is connected to v where the distance between i and v is at most 2^k through instance with ID *key*. Therefore, our algorithm will correctly compute the canonical representation after $\log n$ iterations.

After the first phase, we will compute a canonical representation table which stores $(i, \texttt{sameAs}, CR(i))[1]$ where $CR(i)$ is the canonical representation of i for each instance i. In the second phase, we shall perform a join between this canonical representation table and the whole fuzzy triple set. This join can be simply calculated using a MapReduce program.

5) Handling Vague `SameAs` Closure

However, not all `sameAs` triples are certain `sameAs` triples. The fuzzy pD^* semantics allows using `sameAs` triples to represent the similarity information. In this case, we cannot choose such a canonical representation as illustrated by the following example. Suppose we use the *min* as the t-norm function. Given a fuzzy RDF graph G containing seven triples:

$$(a, \texttt{sameAs}, b)[0.8] \quad (b, \texttt{sameAs}, c)[0.1] \quad (c, \texttt{sameAs}, d)[0.8]$$
$$(a, \texttt{range}, r)[0.9] \quad (u, b, v)[0.9] \quad (c, \texttt{domain}, e)[1] \quad (u', d, v')[0.9].$$

From this graph, we can derive $(v, \texttt{type}, r)[0.8]$. Indeed, we can derive $(b, \texttt{range}, r)[0.8]$ by applying rule f-rdfp11 over $(a, \texttt{sameAs}, b)[0.8]$, $(a, \texttt{range}, r)[0.9]$ and $(r, \texttt{sameAs}, r)[1.0]$. Then we can apply rule f-rdfs3 over $(b, \texttt{range}, r)[0.8]$ and $(u, b, v)[0.9]$ to derive $(v, \texttt{type}, r)[0.8]$.

In this graph, four instances, a, b, c and d are considered as synonyms under the classical pD^* semantics. Suppose we choose c as the canonical representation, then the fuzzy RDF graph is converted into the following graph G' containing four fuzzy triples:

$$(c, \texttt{range}, r)[0.1] \quad (u, c, v)[0.1] \quad (c, \texttt{domain}, e)[1] \quad (u', c, v')[0.8].$$

From this graph, we can derive the fuzzy triple $(v, \texttt{type}, r)[0.1]$, and this is a fuzzy BDB triple from G', which means we cannot derive the fuzzy triple $(v, \texttt{type}, r)[0.8]$. The reason is that after replacing a and b with c, the fuzzy information between a and b, e.g. the fuzzy triple $(a, \texttt{sameAs}, b)[0.8]$, is missing. Furthermore, no matter how we choose the canonical representation, some information will inevitably get lost during the replacement. For this reason, we must store all of these vague sameAs triples and calculate the `sameAs` closure using rules f-rdfp6 and f-rdfp7 to ensure the inference is complete.

Materializing the result of applying rule f-rdfp11x will greatly expand the dataset which may cause fatal efficiency problems. To accelerate the computation, we do not apply rule f-rdfp11x directly. Instead, we modify the algorithms for other rules to consider the effect of rule f-rdfp11x.

ALGORITHM 10: Map function to calculate the canonical representation.

```
Input: key, triple
1: emit(triple.subject, triple.object);
2: emit(triple.object, triple.subject);
```

In the following, we use rule f-rdfs2 mentioned in III-A as an example to illustrate the modification. In rule f-rdfs2, two fuzzy triples join on p. Considering rule f-rdfp11x, if the dataset contains a fuzzy triple $(p, \texttt{sameAs}, p')[n]$, then we can make the following inference by applying f-rdfp11x and f-rdfs2:

$$(p, \texttt{domain}, u)[m], (v, p', w)[k], (p, \texttt{sameAs}, p')[n]$$
$$\Rightarrow (v, \texttt{type}, u)[n \otimes m \otimes k].$$

We use Algorithm 12 to replace Algorithm 1 as the *map* function. The difference is that Algorithm 12 uses a loop between line 2 and line 5. In practice, vague `sameAs` triples are relatively few. Thus we can load them into the memory and compute the `sameAs` closure before the mappers are launched. When the mapper scans a triple in the form of $(p, \texttt{domain}, u)[m]$, it looks up the `sameAs` closure to find the set of fuzzy triples in the form of $(p, \texttt{sameAs}, p')[n]$. For each pair (p', n), the mapper outputs a key p' along with a value {flag='L', u=triple.object, $m \otimes n$}. While processing key p', the reducer will receive all the values of u and $m \otimes n$. Furthermore, the reducer will receive all values of v and k output by the mapper in line 7. Thus the reducer will generate fuzzy triples in the form of $(v, \texttt{type}, u)[n \otimes m \otimes k]$ as desired.

Finally, we discuss the problem of handling `sameAs` triples while processing rules f-rdfp1 and f-rdfp2. We only discuss rule f-rdfp1 since the other can be handled similarly. Consider a fuzzy graph G containing the following $n+1$ fuzzy triples:

ALGORITHM 11: Reduce function to calculate the canonical representation.

```
Input: key, iterator values
1:  min = key;
2:  for v ∈ values do
3:    if v < min then
4:      min = v;
5:    end if
6:  end for
7:  for v ∈ values do
8:    emit(null, new FuzzyTriple(min, sameAs, v, 1.0));
9:  end for
```

The computation of the transitive closure by applying rule f-rdfp4 is essentially calculating the all-pairs shortest path on the instance graph.

ALGORITHM 12: Map function for rules f-rdfs2 and f-rdfp11.

Input: key, triple

```
1: if triple.predicate == 'domain' then
2:   for (subject, sameAs, p') [n] is in the sameAs closure do
3:     m = triple.degree;
4:     emit({p=p'}, {flag='I', u=triple.object, m ⊗ n};
5:   end for
6: end if
7: emit({p=triple.predicate}, {flag='R', v=triple.subject,
     k=triple.degree};
```

$$(a, p, b_1)\ [m_1]\ \ (a, p, b_2)[m_2]\ \ldots\ (a, p, b_n)\ [m_n]$$
$$(p, \texttt{type}, \texttt{FunctionalProperty})[k]$$

By applying rule f-rdfp1, we can derive $n(n-1)/2$ fuzzy triples in the form of $(b_i, \texttt{sameAs}, b_j)[k \otimes m_i \otimes m_j]$.

IV. Experiment

We implemented a prototype system, called FuzzyPD, based on the Hadoop framework[4], which is an open-source Java implementation of MapReduce. Hadoop uses a distributed file system, called HDFS[5] to manage executions details such as data transfer, job scheduling, and error management.

Since there is no system supporting fuzzy pD^* reasoning, we ran our system over the standard LUBM data, and validated it against the WebPIE reasoner to check the correctness of our algorithms. Our system can produce the same results as WebPIE does.

The experiment was conducted in a Hadoop cluster containing 25 nodes. Each node is a PC machine with a 4-core, 2.66GHz, Q8400 CPU, 8GB main-memory, 3TB hard disk. In the cluster, each node is assigned three processes to run *map* tasks, and three process to run *reduce* tasks. So the cluster allows running at most 75 mappers or 75 reducers simultaneously. Each mapper and each reducer can use at most 2GB main-memory.

A. Datasets

Since there is no real fuzzy RDF data available, we generated fuzzy degrees for triples in some crisp benchmark ontologies, i.e., DBPedia [11] core ontology and LUBM ontologies [12]. For DBPedia core ontology, we assigned a randomly chosen fuzzy degree to each triple. For LUBM ontologies, we extended the fuzzy LUBM dataset (called fLUBM dataset) generated for fuzzy DL-Lite semantics in [13]. The fLUBM dataset adds two fuzzy classes, called `Busy` and `Famous`. The fuzzy degrees of triples stating an individual belong to these two fuzzy classes are generated according to the number of courses taught or taken by the individual, and the number of the publications of the individual respectively.

However, since there is no hierarchy among these fuzzy classes, we cannot use fLUBM to test our reasoning algorithm. To tackle this problem, we further added six fuzzy classes, `VeryBusy`, `NormalBusy`, `LessBusy`, `VeryFamous`, `NormalFamous` and `LessFamous`. Given an individual i, suppose its membership degree w.r.t. class `Busy` (the fuzzy degree how i belongs to class `Busy`) is b_i. If $b_i < 0.5$, we added a fuzzy triple $(i,\ \texttt{type},\ \texttt{LessBusy})\ [b_i/0.5]$ into the dataset; if $0.5 \le b_i < 0.7$, we generated a fuzzy triple $(i,\ \texttt{type},\ \texttt{NormalBusy})\ [b_i/0.7]$; otherwise, we generated a fuzzy triple $(i,\ \texttt{type},\ \texttt{VeryBusy})\ [b]$. We added two fuzzy triples, $(\texttt{LessBusy},\ \texttt{subClassOf},\ \texttt{Busy})\ [0.5]$ and $(\texttt{VeryBusy},\ \texttt{subClassOf},\ \texttt{Busy})\ [1.0]$ to the TBox. Similarly, we can generate the fuzzy triples related to `Famous`.

We further added a transitive property call `youngerThan` to test calculation of the transitive closure. In each university ontology, we assigned a randomly generated age to each student. Then we generated n `youngerThan` triples. For each triple, we randomly chose two different students i and j satisfying $age_i < age_j$, and added a fuzzy triple $(i,\ \texttt{youngerThan},\ j)\ [age_i/age_j]$ into the data set.

Finally, we added a TBox triple to assert that `emailAddress` is an inverse functional property. In fact, e-mail is usually used for identifying a person online. Furthermore, for each faculty f, since we know the university from which he got his bachelor degree, we picked one email address e belonging to an undergraduate student in that university, and added a triple $(f,\ \texttt{emailAddress},\ e)\ [d]$ into the data set. Here we assigned the fuzzy degrees d to be either 1.0 or 0.9. Then `sameas` triples were derived using the semantics of `inverseFunctionalProperty`. We set d to be 0.9 with probability 1%, so that a small set of vague `sameas` triples can be generated. Similarly, we can generate other emailAddress related triples for the master and doctoral students similarly. We call the new dataset as fpdLUBM dataset[6].

B. Experimental Results

1) Fuzzy RDFS Reasoning

Since we employ the fuzzy RDFS reasoning algorithm as a subroutine of the fuzzy pD^* reasoning algorithm, the running time of

[4] http://hadoop.apache.org/
[5] http://hadoop.apache.org/hdfs/

[6] The dataset is available at http://apex.sjtu.edu.cn/apex_wiki/fuzzypd.

the former algorithm is definitely smaller than that of the later. In this subsection, we only study the relation between the performance of our algorithm and the number of computing units (mappers and reducers).

We used generated fuzzy version of the DBPedia core ontology which contains 26996983 fuzzy triples. After performing fuzzy RDFS reasoning algorithm, 133656 new fuzzy triples were derived. The running time results are listed in Table 3. The result shows that the running time speedup increases along with the number of computing units used. However, this speedup increase is not as linear as expected. The reason is that there is a warmup overhead of the Hadoop system which is unavoidable no matter how many computing units we used. Furthermore, the running time is relatively smaller than this overhead, thus it is hard to calculate the exact running time from the data we measured. Later in Section IV-B3, we will see the results conducted on larger datasets showing a good linear speedup as expected.

TABLE 3 Scalability for fuzzy RDFS reasoning.

NUMBER OF UNITS	128	64	32	16	8	4	2
TIME (SECONDS)	122.653	136.861	146.393	170.859	282.802	446.917	822.269
SPEEDUP	6.70	6.01	5.62	4.81	2.91	1.84	1.00

2) Comparison with WebPIE

We compared the performance of our system with that of the baseline system WebPIE[7]. We ran both systems over the same dataset fpdLUBM8000. The results are shown in Table 4. Notice that the dataset is a fuzzy dataset, for WebPIE, we simply omitted the fuzzy degree, and submitted all crisp triples to the system. So our system (FuzzyPD) output a little more triples than WebPIE, because our system also updated the fuzzy degrees. The running time difference between our system and WebPIE is from −5 to 20 minutes. However, since a Hadoop job's execution time is affected by the statuses of the machines in the cluster, several minutes' difference between the two systems is within a rational range. Thus we concluded that our system is comparable with the state-of-the-art inference system.

TABLE 4 Experimental results of our system and WebPIE.

NUMBER OF UNIVERSITIES	TIME OF FUZZYPD (MINUTES)	TIME OF WEBPIE (MINUTES)
1000	38.8	41.32
2000	66.97	74.57
4000	110.40	130.87
8000	215.48	210.01

3) Scalability

To test the scalability of our algorithms, we ran two experiments. In the first experiment, we tested the inference time of our system over datasets with different sizes to see the relation between the data volume and the throughput. In the second experiment, we ran our system over fpdLUBM1000 dataset with different number of units (mappers and reducers) to see the relation between the processing units and the throughput. Furthermore, in the second experiment, we set the number of mappers to be the same as the number of reducers. Thus a total number of 128 units means launching 64 mappers and 64 reducers.

The results for the first experiment can be found in table 6. From the table, we can see that the throughput increases significantly while the volume increases. The throughput while processing fpdLUBM8000 dataset is 50% higher than the throughput while processing dataset containing 1000 universities. We attributed this performance gain to the platform startup overhead which is amortized over a larger processing time for large datasets. The platform overhead is also responsible for the non-linear speedup in Table 5 which contains the results of the second test. Figure 1 gives a direct illustration of the overhead effect. In Figure 1, if we subtracted a constant from the time dimension of each data point, then the time is inversely proportional to the number of units. Since the running time should be inversely proportional to the speed, after eliminating the effect of the platform overhead, the system's performance speeds up linearly to the increase of number of units.

[7] We fix some bugs in the source code which will cause performance problem.

V. Related Work

Recently, there have been some works on learning fuzzy OWL ontologies [14], [15]. [5] is the first work to extend RDFS with fuzzy vagueness. In [6], we further proposed the fuzzy pD^* semantics which allows some useful OWL vocabularies, such as

TABLE 5 Scalability for fuzzy *pD reasoning.**

NUMBER OF UNITS	TIME (MINUTES)	SPEEDUP
128	38.80	4.01
64	53.15	2.93
32	91.58	1.70
16	155.47	1.00

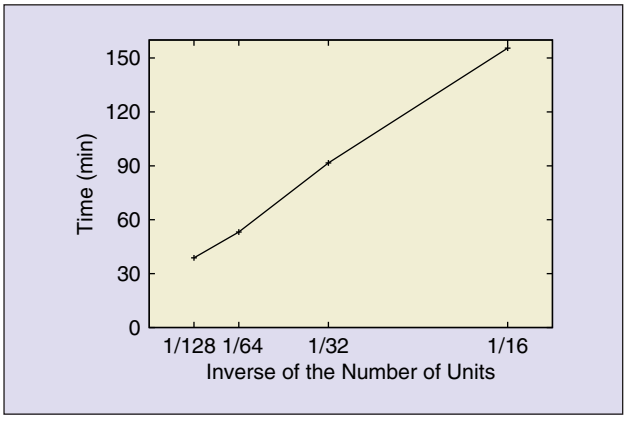


FIGURE 1 Time versus inverse of number of mappers.

TABLE 6 Scalability over data volume.

NUMBER OF UNIVERSITIES	INPUT (MTRIPLES)	OUTPUT (MTRIPLES)	TIME (MINUTES)	THROUGHPUT (KTRIPLES/SECOND)
1000	155.51	92.01	38.8	39.52
2000	310.71	185.97	66.97	46.28
4000	621.46	380.06	110.40	57.37
8000	1243.20	792.54	215.50	61.29

TransitiveProperty and SameAs. In [16] and [17], a more general framework for representing "annotated RDF data" and a query language called AnQL were proposed.

As far as we know, this is the first work that applies the MapReduce framework to tackle large scale reasoning in fuzzy OWL. The only work that tried to deal with large scale fuzzy ontologies was given in Pan et al. in [13]. Their work proposed a framework for fuzzy query answering in fuzzy DL-Lite. In contrast, our work focussed on the inference problem over large scale fuzzy pD^* ontologies.

We will briefly discuss some related work on scalable reasoning in OWL and RDF. None of them takes into account of fuzzy information.

Schlicht and Stuckenschmidt [18] showed peer-to-peer reasoning for the DL $\mathcal{ALC}$ but focusing on distribution rather than performance. Soma and Prasanna [19] presented a technique for parallel OWL inference through data partitioning. Experimental results showed good speedup but only on very small datasets (1M triples) and runtime performance was not reported.

In Weaver and Hendler [20], straightforward parallel RDFS reasoning on a cluster was presented. But this approach split the input to independent partitions. Thus it is only applicable for simple logics, e.g. RDFS without extending the RDFS schema, where the input is independent.

Newman et al. [21] decomposed and merged RDF molecules using MapReduce and Hadoop. They performed SPARQL queries on the data but performance was reported over a dataset of limited size (70,000 triples).

Urbani et al. [7] developed the MapReduce algorithms for materializing RDFS inference results. In [8], they further extended their methods to handle OWL pD^* fragment, and conducted experiment over a dataset containing 100 billion triples.

VI. Conclusion

In this paper, we proposed MapReduce algorithms to process forward inference over large scale data using fuzzy pD^* semantics (i.e. an extension of pD^* semantics with fuzzy vagueness). We first identified the major challenges to handle the fuzzy information when applying the MapReduce framework, and proposed a solution to tackle each of them. Furthermore, we implemented a prototype system for the evaluation purpose. The experimental results show that the running time of our system is comparable with that of WebPIE, the state-of-the-art inference engine for large scale OWL ontologies in pD^* fragment. They show the scalability of our main reasoning algorithm in both the dimensions of data volumes and number of processing units.

As a future work, we will apply our system to some applications, such as Genomics and multimedia data management. In another future work, we will consider other fuzzy semantics, such as the one based on type-2 fuzzy set theory [22], and propose MapReduce algorithms for the new semantics.

VII. Acknowledgments

Guilin Qi is partially supported by NSFC (61003157,60903010), Jiangsu Science Foundation (BK2010412), Excellent Youth Scholars Program of Southeast University under grant 4009001011, the Key Laboratory of Advanced Information Science and Network Technology of Beijing or the Key Laboratory of Information Science & Engineering of Railway Ministry(XDXX1011), the Key Laboratory of Computer Network and Information Integration (Southeast University).

References

[1] RDF. [Online]. Available: http://www.w3.org/RDF/

[2] RDFS. [Online]. Available: http://www.w3.org/TR/rdf-schema/

[3] H. J. Horst, "Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the owl vocabulary," *J. Web Semantics*, vol. 3, nos. 2–3, pp. 79–115, 2005.

[4] OWL. [Online]. Available: http://www.w3.org/TR/owl-features/

[5] U. Straccia, "A minimal deductive system for general fuzzy RDF," in *Proc. RR'09*, 2009, pp. 166–181.

[6] C. Liu, G. Qi, H. Wang, and Y. Yu, "Fuzzy reasoning over RDF data using OWL vocabulary," in *Proc. WI'11*, 2011.

[7] J. Urbani, S. Kotoulas, E. Oren, and F. Van Harmelen, "Scalable distributed reasoning using mapreduce," in *Proc. ISWC'09*, 2009, pp. 374–389.

[8] J. Urbani, S. Kotoulas, J. Maassen, F. Van Harmelen, and H. Bal, "Owl reasoning with webpie: Calculating the closure of 100 billion triples," in *Proc. ESWC'10*, 2010, pp. 213–227.

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. OSDI'04*, 2004, pp. 137–147.

[10] C. Liu, G. Qi, H. Wang, and Y. Yu, "Large scale fuzzy pd* reasoning using mapreduce," in *Proc. ISWC'11*, 2011.

[11] DBPedia. [Online]. Available: http://dbpedia.org/

[12] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A benchmark for owl knowledge base systems," *J. Web Semantics*, vol. 3, no. 2, pp. 158–182, 2005.

[13] J. Z. Pan, G. Stamou, G. Stoilos, S. Taylor, and E. Thomas, "Scalable querying services over fuzzy ontologies," in *Proc. WWW'08*, 2008, pp. 575–584.

[14] F. Zhang, Z. M. Ma, J. Cheng, and X. Meng, "Fuzzy semantic web ontology learning from fuzzy UML model," in *Proc. CIKM'09*, 2005.

[15] G. Da San Martino and A. Sperduti, "Mining structured data," *IEEE Comput. Intell. Mag.*, pp. 42–49, 2010.

[16] U. Straccia, N. Lopes, G. Lukacsy, and A. Polleres, "A general framework for representing and reasoning with annotated semantic web data," in *Proc. AAAI'10*. AAAI Press, 2010, pp. 1437–1442.

[17] U. S. N. Lopes, A. Polleres, and A. Zimmermann, "AnQL: SPARQLing up annotated RDF," in *Proc. ISWC'10*, 2010, pp. 518–533.

[18] A. Schlicht and H. Stuckenschmidt, "Peer-to-peer reasoning for interlinked ontologies," vol. 4, pp. 27–58, 2010.

[19] R. Soma and V. K. Prasanna, "Parallel inferencing for owl knowledge bases," in *Proc. ICPP'08*, 2008, pp. 75–82.

[20] J. Weaver and J. A. Hendler, "Parallel materialization of the finite RDFS closure for hundreds of millions of triples," in *Proc. ISWC'09*, 2009, pp. 682–697.

[21] A. Newman, Y.-F. Li, and J. Hunter, "Scalable semantics—The silver lining of cloud computing," in *Proc. ESCIENCE'08*, 2008.

[22] J. M. Mendel, "Type-2 fuzzy sets, a tribal parody," *IEEE Comput. Intell. Mag.*, pp. 24–27, 2010.

Huajun Chen and Zhaohui Wu
Zhejiang University, China
Philippe Cudré-Mauroux
Massachusetts Institute of Technology, USA

Review Article

Semantic Web Meets Computational Intelligence: State of the Art and Perspectives

I. Introduction

In the early sixties, the concept of *Semantic Network* was firstly introduced as a knowledge representation model by cognitive scientist Allan M. Collins, linguist M. Ross Quillian and psychologist Elizabeth F. Loftus [1]. In 1998, the term *Semantic Web (SW)* was coined by Web inventor Tim Berners-Lee as an extension of the current Web [2]. It was described as a giant global *semantic network* of data that is directly consumable and understandable to machines. In contrast to a *hypertext Web* that indicates texts linked to other texts in other places by hyperlinks, the Semantic Web projects a *hyperdata Web* that indicates data objects linked with other data objects across the Web through formal semantics and ontologies. It enables the formation of *a global web of data* or *open linked data* [3] that interlinks distributed data at a Web-scale. The Semantic Web is led by the World Wide Web Consortium (W3C) as an international collaborative movement [4].

Computational Intelligence (CI) [5] is a set of nature-inspired computational approaches that primarily includes Fuzzy Logic Systems (FLS) [6], Evolutionary Computation (EC) [7] and Artificial Neural Networks (ANN) [8]. Fuzzy logic was introduced as a tool to deal with vagueness and uncertainty that is common for human intelligence. Evolutionary computation could produce highly optimized processes by mimicking the population-based evolution. Neural networks is adept at modeling and learning complex relationships by mimicking the human brain.

The Semantic Web, in its intrinsic nature, creates even more sophisticated problems than hypertext Web. Firstly, the uncertain nature of the Web calls for more expressive languages capable of dealing with fuzziness and vagueness in Web semantics. Secondly, in a highly open, decentralized, and vast Web environment, more efficient computational approaches are required to reduce the computational complexity of a diverse of new problems inherent to the Semantic Web. Typical examples include Web-scale query answering and reasoning, distributed semantic storage, complex ontology alignment across multiple domain boundaries, and massive linked data analysis, etc.


PUBLIC DOMAIN PHOTOS.COM/JOY SHRADER

These insights have triggered a body of researches and innovation with a synergy of the Computational Intelligence and the Semantic Web recently [9][10][11][12][13][14]. For examples, Fuzzy Logic has inspired the design of variant fuzzy extensions of several Semantic Web languages [9][10]; Nature-inspired optimization methods such as Genetic Algorithms (GA) [11], Swarm Intelligence (SI) [12], and Artificial Immune Systems (AIS) [13] have been witnessed in optimizing query answering and reasoning over such a vast, decentralized data space; Artificial Neural Networks are equipped by many Semantic Web applications to improve the learning capability.

This article attempts to survey the state of the art of applying CI approaches into Semantic Web applications. We first identify the vital characteristics or challenges inherent to the Semantic Web including vastness, vagueness, and inconsistency in Section II. We then survey existing researches with CI methods incorporated into the Semantic Web from perspectives of these characteristics. In order to present the literature review, we collected and selected those well-established and most representative works. We classified them into three primary categories of CI methods including Fuzzy Logic, Evolutionary Computation, and Artificial Neural Network, corresponding to Section III, IV, and V respectively. We emphasize on the discussion of perspectives and potential future research directions in this arena in Section VI, followed by a conclusion in Section VII.

II. Semantic Web in a Nutshell

A. The Emergence of a Global Linked Data Space

In a nutshell, the key innovative idea of the Semantic Web is to create a ***Global***

Digital Object Identifier 10.1109/MCI.2012.2188592
Date of publication: 13 April 2012

In 1998, the term *Semantic Web (SW)* was coined by Web inventor Tim Berners-Lee as an extension of the current Web. It was described as a giant global *semantic network* of data that is directly consumable and understandable to machines.

Linked Data Space [3] through formal semantics. Tim Berners-Lee suggested four principles to the question of "how" [15]. The first is to use URIs to identify things. The second is to use HTTP URIs so that these things can be referred to and de-referenced by both people and intelligent agents. The third is to publish machine-understandable information about the things when their URIs are de-referenced, using standard Semantic Web languages such as Resource Description Framework (RDF) [16], Web Ontology Language (OWL) [16], and XML. The fourth is to include links (with semantic annotations) to other related URI-identified things in other places on the Web to improve discovery of related information.

Figure 1 illustrates the new architecture of the Internet inspired by Tim Berners-Lee. It contains three major levels of abstraction: Net, Web, and Graph. The Graph layer is thought of as the Semantic Web. It allows Web users or intelligent agents to create and explore the connections between the things or data objects without the awareness of the boundaries of Web sites.

B. The Semantic Web Languages: RDF and OWL

Succinctly, RDF [16] is a data model based on the idea of making statements about Web resources in the form of triple: <*subject, predicate, object*>. The subject denotes a resource, and the predicate denotes an attribute of the resource or a relationship with other resources denoted by the object. For example, we can represent the notion *"Tom has the symptom headache"* in RDF as a triple: <*Tom, hasSymptom, Headache*>. The triple statement model provides an especially straightforward and simple way of describing arbitrary things and their relations on the Web.

OWL [16] goes beyond RDF in its more expressive ability to represent a full ontology that gives a formal, explicit specification of a shared conceptualization for a domain. An OWL ontology typically defines vocabularies for classes, properties, instances and their operations. The data described by an OWL ontology is interpreted as a set of *individuals*, a set of *property assertions* which relate these individuals to each other, with additional axioms that place constraints on classes and properties. These axioms enable systems to infer additional knowledge based on the knowledge explicitly provided. The W3C-endorsed OWL specification [17] includes three variants of OWL family languages including *OWL Lite*, *OWL DL* and *OWL Full* with different levels of expressiveness. The latest version of OWL is OWL 2 [16] that includes three tractable profiles targeted at different types of applications. For example, OWL 2 EL is particularly useful in applications that contain very large numbers of properties and/or classes, OWL 2 QL is aimed at applications that use very large volumes of instance data and require efficient query answering reasoning, and OWL 2 RL is aimed at applications that require scalable rule-based reasoning.

As content in forms of RDF or OWL can manifest itself as self-descriptive data, more accurate, complete and meaningful results can be obtained with the assistance of automated query and reasoning process.

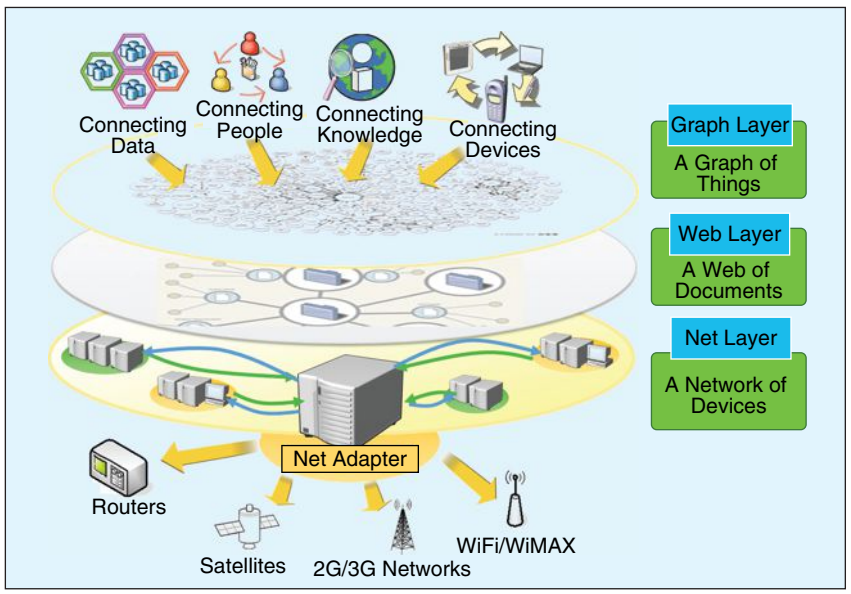


FIGURE 1 Three major levels of abstraction of the Internet: Net, Web, and Graph.

C. Synergy of the Semantic Web and Computational Intelligence

The Semantic Web presents difficult challenges owing to its nature of being decentralized, vast, uncertain, incomplete and inconsistent. We summarize these challenges from three perspectives, on which different CI techniques have been proven to be effective and advantageous.

- **Vastness and Tractability**: The vastness of the Semantic Web is obvious. For example, the W3C Linking Open Data community project [3] has collectively gathered 295 data sets consisting of over 31 billion RDF triples, which are

interlinked by around 504 million RDF links as of September 2011. Any automated query and reasoning systems will have to tackle truly huge inputs in a highly optimized and tractable way.

- **Vagueness and Uncertainty**: Descriptive data on the Web usually comes along with imprecise concepts like "high" or precise concept with uncertain values [18]. The vagueness and uncertainty may exist in user queries, in data content, in matching queries to data content, and in combining disparate data sources with overlapping ontologies. Both current RDF and OWL languages can only handle crispy descriptions which are definitely not satisfactory.
- **Divergence and Inconsistence**: There are always contradictions that inevitably arise when data from divergent sources are combined. In an open system such as the Semantic Web, the aim is to have the agents inter-operate irrespective of the conflicts between their semantics in solving a problem collectively. This requires advanced methods to deal with semantic mapping, ontology alignment, inconsistent reasoning, etc. [19].

In studies related to both CI and the Semantic Web, many attempts have been made to apply variants CI approaches to tackle these challenges. For examples, Evolutionary Computation has been demonstrated to deal with the vastness and tractability issues; Fuzzy Logic has been proven to be effective for the management of vagueness and uncertainty in Web semantics; Artificial Neural Networks have been applied in solving inconsistent issues with regards to data mapping, ontology alignments, and the like. Figure 2 establishes a connection between typical CI approaches and their applications in the Semantic Web, which will be surveyed in detail in the following sections.

III. Semantic Web Meets Fuzzy Logic

One big problem in many Semantic Web applications is to model, store, query, map and reason with fuzziness and vagueness in data semantics [18]. This section first presents several representative fuzzy Semantic Web applications, then introduces the proposed fuzzy languages and their corresponding reasoning methods.

A. Fuzzy Applications in the Semantic Web

Fuzzy concepts are useful in enhancing query or search in the Semantic Web [20][21][22]. For example, a fuzzy concept *Agile Animal* can be used to annotate and index data items about animals in a semantic search engine. A *Tiger* can thus be classified as an *Agile Animal* to a

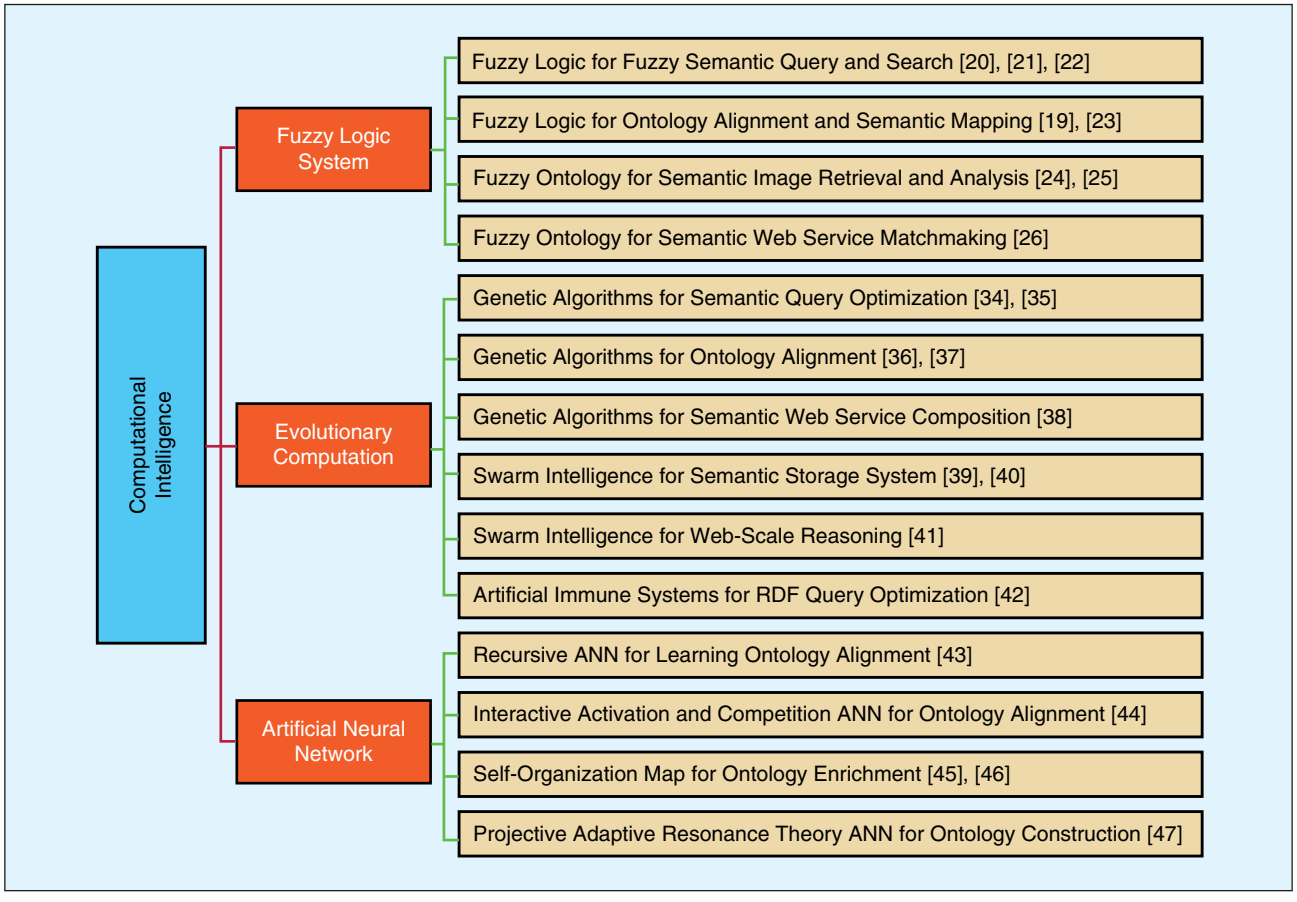


FIGURE 2 An illustration of the landscape of the applications of CI techniques in the Semantic Web.

There are today many proposals for fuzzy extensions to the Semantic Web languages such as fuzzy RDF [27], fuzzy OWL [28] and the most recent fuzzy OWL2 [29].

degree of 0.9. In this way, the degree of relevance to the concept can be determined and ranked while querying and searching based on such a fuzzy semantic index.

Fuzzy approaches have also been applied to resolving semantic conflicts between ontologies [19][23]. Fuzzy semantic mapping can state the generic similarity of two concepts of two different ontologies. We can assert that the objects modeled by the first concept can be also modeled by the second concept to a certain degree. For example, the concept of *TabletComputers* can be mapped to *MobileDevice* to a degree of 0.8. This is very useful in the Semantic Web as the boundaries of semantic mappings between data sources are usually fuzzy.

There are also schemes for using fuzzy ontology to enhance image analysis and retrieval [24][25]. For example, an object *o1* in an image could be *Red* to a degree 0.8 and *closeTo* another object *o2* to a degree 0.6. These fuzzy assertions together with ontology axioms enable agents to recognize objects contained in an image in a fuzzy way so that fuzzy semantic search and analysis can be performed against those images.

The application of fuzzy matchmaking of Semantic Web services is also found in the literature [26]. The matchmaking activity exploits a fuzzy ontology to represent multi-granular information enclosed in the semantic descriptions of a web service using the OWL-S language, an OWL-based semantic markup language for web services. The matchmaking is computed based on a fuzzy distance between user queries and the semantic profiles of matched services.

B. Fuzzy Representation Languages for the Semantic Web

There are today many proposals for fuzzy extensions to the Semantic Web languages such as fuzzy RDF [27], fuzzy OWL [28] and the most recent fuzzy OWL2 [29]. Most of these languages are based on Fuzzy Description Logics [30] [31] that are extended from standard Description Logics (DL) and Fuzzy Set Theory.

At this point we want to make clear that this article is not intended to present a comprehensive and formal specifications on these languages (See [9][10] for such purposes). As an alternative, we use several examples to present the syntax, semantics, and axiom expressions.

We commence from a running scenario in healthcare. As usual, we use $C1, C2\ldots$ to denote concepts and $o1, o2\ldots$ to denote instances. We use DL syntax [31] for simplicity and readability. We firstdefine the *TBox*, which denotes the terminological component of a knowledge base.

$$\begin{aligned} TBox = \{ & C1 \equiv YoungPatient \\ & \quad \cap HeavyWeight \\ & C2 \equiv \exists\, hasSymptom.(HighFever \\ & \quad \cap \exists\, hasSign.WeakPulse)\} \end{aligned}$$

C1 denotes the concept for those young patients with heavy weight. *C2* denotes the concept for those patients who have symptoms of both high fever and weak pulse. With the fuzzy concepts defined in the *Tbox*, we can create a corresponding *ABox*, the assertion component of a knowledge base, with fuzzy assertions like the following ones (see the box at the bottom of the page).

If using t-norm (triangular norm), in order for *o1* to be an instance of *C1* it should hold that:

$$\begin{aligned} C1(o1) &= t(YoungPatient(o1), \\ &\qquad HeavyWeight(o1)) \\ &= t(0.8, 0.6) \end{aligned}$$

Depending on which *t-norm* we use, we can infer different values for *o1* being a *C1* or a *C2*.

For example, if *t* is the product *t-norm* then, *C1(o1)= 0.48*.

We may want to define fuzzy axioms in OWL such as *fuzzy subsumption, fuzzy functional role, fuzzy disjointness*, etc. [10]. For example, to define the *fuzzy subsumption* relation between the concepts of *BodyItching* and *SkinSymptom* with a degree 0.9, we use:

$$(BodyItching \subseteq SkinSymptom) > 0.9$$

Last but not least, we show examples on how to concretely represent fuzzy knowledge in an RDF/XML concrete syntax so as to publish fuzzy ontologies on the Web. There are several approaches to encoding fuzzy knowledge [10][29]. One is to extend OWL construct with the elements degree and *ineqType* that takes values such as *">="*. The following is a simple example.

```
<HighFever rdf : about="#symptom01"
    owlx : ineqType=">=" owlx : degree
    ="0.7" />
```

Another approach is to store fuzzy knowledge in the form of OWL annotations [29], thus avoiding the burden of extending the language. The advantage of using annotation is its strong compatibility with existing tools such as parsers or reasoning engines for non-fuzzy ontologies since these tools can simply ignore fuzzy descriptions encoded in the annotations.

C. Reasoning in a Fuzzy Semantic Web

Being similar to crispy OWL knowledge base, reasoning services in fuzzy OWL

$$\begin{aligned} ABox = & \{(o1 : YoungPatient) > 0.8, (o1 : HeavyWeight) > 0.6 \\ & (o2 : HighFever) > 0.7, ((o1, o2) : hasSymptom)) > 0.9, \\ & \{(o3 : WeakPulse) > 0.5, ((o2, o3) : hasSign)) > 0.8\} \end{aligned}$$

include: *KB satisfiability, concept n-satisfiability, concept subsumption, and entailment*. In addition, two other important reasoning problems are *the Best Truth Value Bound Problem (BTVBP)* that computes the best lower and upper truth value bounds for an axiom, and the *Best Satisfiability Bound Problem (BSBP)* that determines the maximal degree of truth that a concept may have over all individuals in the domain (See [9] for a formal definition on these problem).

Reasoning services over a fuzzy OWL ontology are normally reduced to reasoning over classical fuzzy description logic [30][31], thus one can implement fuzzy DL reasoners to support reasoning for fuzzy extensions of OWL. There exist several fuzzy reasoners particularly designed for OWL. FiRE [18] is tableaux-based fuzzy reasoner that supports a nominal and datatype-free subset of fuzzy-OWL DL. FuzzyDL[28] is a mixed integer programming fuzzy reasoner that supports fuzzy-OWL Lite. Scalability issue has also been investigated in the literatures [22][32]. Particularly, Pan et al. presents algorithms of fuzzy OWL 2 QL for a family of expressive fuzzy query languages for the Quill query engine in the TrOWL infrastructure [22]; Liu et al. report a MapReduce-based framework that enables scalable reasoning for a subset of fuzzy OWL 2 RL on a cloud infrastructure [32].

IV. Semantic Web Meets Evolutionary Computation

The Semantic Web, in entirety, is a large-scale self-organized complex system that is akin to being evolutionary [33]. In this consideration, new adaptive approaches are required to exploit the ever growing amounts of dynamic, multi-dimensional, and evolutionary semantic data at a Web-scale. This section surveys relevant applications of three typical Evolutionary Computation methods including Genetic Algorithms, Swarm Intelligence, and Artificial Immune Systems, which have been proven to be effective and efficient in reducing the complexity of the problem space with regards to web-scale query answering and reasoning.

The Semantic Web, in entirety, is a large-scale self-organized complex system that is akin to being evolutionary.

A. Genetic Algorithms for the Semantic Web

A Genetic Algorithm (GA) is a search heuristic that generates optimization solutions to problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and recombination [11]. GA methods have been investigated in optimizing semantic query processing [34][35], reducing complexity for ontology alignment [36][37], and computing optimal composition of Semantic Web services [38].

eRDF [34] is a GA-based framework for optimizing semantic query evaluation. The problem of semantic query evaluation can be formulated as finding semantic subgraphs that match the pattern of a given semantic query. eRDF firstly generates partial matched subgraphs as the *population* of candidate solutions. To propagate new candidate solutions, it then *mutates* the subgraphs by extending the graphs, or *recombines* two candidate subgraphs to generate new candidate solutions. Subgraphs that survive the fitness selection are considered to be optimal. Alexander Hogenboom et al reports similar approaches to finding optimal orders of join-paths for RDF query optimization [35].

GA methods have also been applied in reducing complexity of ontology alignment when a number of matchers are required to be combined to find optimal mappings from huge number of pairs of entities [36]. The problem consists of finding a best combination of weights for different ontology matchers such as those based on string normalization, string similarity, data type comparison, and linguistic methods, etc. The advantages of using GA methods lay in its capability of learning the best combination of weights for optimum alignment functions without requiring human intervention.

Tizzo, N.P. [37] reports the use of Asynchronous Teams algorithm with genetic agents to optimize the composition of Semantic Web services. There are agents responsible for creating new composition patterns. There are other agents performing the crossover and mutation over these patterns based on genetic algorithms.

B. Swarm Intelligence for the Semantic Web

Swarm Intelligence (SI) is the collective behavior of decentralized, self-organized systems [12]. SI systems are typically made up of a population of simple agents interacting locally with each other and leading to the emergence of "intelligent" global behavior. SI methods have inspired researchers to implement self-organized storage systems for large-scale semantic data sets [39][40] and optimize the decentralized Web-scale reasoning process [41].

Scalable query and reasoning over decentralized semantic storage requires both smart strategies of storing semantic data and highly optimized query and reasoning approaches. Ant-inspired swarm approaches have been used to implement distributed storage and retrieval system for large-scale RDF data sets [39][40]. The idea is to model storage operations on RDF triples as ants moving virtual network of nodes. While writing, the ants move from nodes to nodes until finding a number of RDF triples sufficiently similar to the one to be stored, and store it based on similarity clustering over the triples. While reading, the ants regard RDF triples as food and forage from nodes to nodes until finding the similarity clusters containing the particular result. Successful operations trace back the paths they took and maintain virtual pheromones for each node to node connection. Subsequent

The most representative advantage of CI methods for the Semantic Web is their capability to tackle difficult problems in a highly dynamic and decentralized setting.

operations can make use of these pheromones as heuristic rules to evaluate the possibility of finding results if selecting that particular connection. Results show that this approach can generate optimized storage strategy to improve the overall efficiency of semantic storage system.

Swarm approaches are also feasible for optimizing Web-scale reasoning particularly geared towards a decentralized setting [41]. The idea is that reasoning tasks can be decomposed by distributing inference rules to individuals of a self-organized swarm. All individuals "walk" on the triples of a RDF graph locally to expand their knowledge in a parallel way, thereby optimizing the whole reasoning processes significantly owning to the large number of swarms. One advantage of the approach is that triples can be added and deleted to the store at any time without affecting the inference process due to the random behavior and large number of individuals. This is an important feature because of the highly dynamic nature of the Semantic Web.

C. Artificial Immune System for the Semantic Web

Artificial Immune Systems (AIS) are a type of EC systems inspired by the principles and processes of the vertebrate immune system [13]. It typically exploits the immune system's characteristics of learning and memory to solve a problem. AIS has inspired solutions to optimize query answering in the Semantic Web [42]. The idea is to develop an analogy between semantic queries and antibodies of immune systems. Successful antibodies that are activated by an infection are to be cloned and mutated, thus generating a number of similar antibodies better suited to tackle the infection. Analogously, successful queries that produce relevant results are to be cloned and modified to give rise to various similar queries, each of which may be an improvement and mutation on the original query.

V. Semantic Web Meets Artificial Neural Network

An Artificial Neural Network (ANN) is a computational model inspired by the structure and functional aspects of biological neural networks [7]. They are useful to learn complex relationships or patterns hidden in large scale semantic data. Researchers have used ANN to enhance ontology alignment [43][44], ontology enrichment [45], concept mining [46], automatic ontology construction [47], etc. This section surveys and classifies the usage of ANNs in two typical categories: unsupervised ANN for ontology learning and supervised ANN for ontology alignment. For each category, we select two most representative works from the literature.

A. Supervised ANN for Ontology Alignment

In the literature, supervised ANNs are widely applied in learning semantic mappings between heterogeneous ontologies.

Recursive Neural Network model (RNN) [48], designed to process structured data efficiently, is suitable for use with ontologies which are in a structured data representation too. RNN has been used to model automatic ontology alignment [43]. The idea is to use concept structures and instance relations in an ontology as input to a neural network to learn classifiers, which will be used for aligning concepts of other ontologies.

One problem concerning ontology alignment is to find an optimal configuration that can best satisfy ontology constraints, such as "if a concept $c1$ maps to another concept $c2$ is *true*, then concept $c3$ maps to concept c4 is false." The Interactive Activation and Competition (IAC) neural network [49], designed to solve constraints satisfaction problems in word perception, is used to search for a global optimal solution satisfying as many ontology constraints as possible [44]. The idea is to use a node in the IAC neural network to represent an element mapping hypothesis. The connections between nodes represent constraints between hypotheses. If two hypotheses supports or are against each other, the connection between them is positive or negative respectively. A learning process can thus be applied over the network to learn the optimal solution for constraints satisfaction.

B. Unsupervised ANN for Ontology Learning

In the literature, unsupervised ANNs are found to be used to learn new concepts and instances from domain corpus, in order to enrich or automatically construct an ontology.

Self-Organization Map (SOM) [50] is a type of unsupervised neural network that can produce a low-dimensional representation of the input space of the training samples, called a map. SOM has been used to enrich domain ontologies with concepts and instances extracted from a domain text corpus [45][46]. The idea is to first convert an ontology into a neural representation as an initial state of a self-organization map. Next, the terms representing concepts or instances that are to be added into the ontology are extracted from a domain text corpus by text mining process. The actual ontology enrichment takes place via an unsupervised training of the neural network by exposing the initialized SOM to the terms and their contextual information extracted from the domain corpus based on certain types of similarity metrics.

Another type of unsupervised ANN, called the Projective Adaptive Resonance Theory Neural Network (PART) [51], has also been employed to support automatic ontology construction from web pages [47]. The PART is trained to cluster the collected web pages for the sake of looking for representative terms of each cluster of web pages. The representative terms are input to a Bayesian

network to complete the hierarchy of the ontology.

VI. Perspectives and Potential Research Directions

The idea of using CI techniques to the Semantic Web has been successfully implemented by many researchers. This section points out the most representative advantages and disadvantages of CI methods for the Semantic Web, and proposes several potential directions for this research area.

A. Advantages and Disadvantages of CI methods for the Semantic Web

The most representative advantage of CI methods for the Semantic Web is their capability to tackle difficult problems in a highly dynamic and decentralized setting. Enriching the Web with semantics and enabling Web intelligence are nontrivial missions due to the decentralized nature of the Web. The fact that no central components are on duty imposes autonomous, dynamic, uncertain, and random behaviors on its constituents. Autonomy, uncertainty, and randomness have been well studied in nature-inspired approaches and the CI communities have accumulated a wealth of well-established approaches. These approaches are particularly adept in addressing the challenges of dealing with autonomy, uncertainty, randomness and chaos.

One disadvantage of currently available CI methods is that they are usually targeted at only one or two specific aspects of a problem, and can only resolve problems separately. However, one reality of the Semantic Web is that the problems of vagueness, autonomy, randomness and inconsistence are present simultaneously for many applications. This requires a hybrid and integration of different CI methods, enabling them to work collectively. We extend the discussion in more details in Subsection *D*.

B. Broader Use of Nature-inspired Methods for the Semantic Web

It is noticed that the "marriage" of the two fields is still new, and only a small portion of this vast wealth have been explored in the Semantic Web community. Therefore, it may be interesting to try a broader variant of CI techniques such as ant colony optimization, particle swarm optimization, harmony search, memetic algorithm, multi-valued logic, chaos theory, and many more, into the Semantic Web. For example, Harmony Search (HS) may overcome the drawback of GA's building block theory which works well only if the relationship among variables is carefully considered. It is thus promising to explore HS methods to evaluate more complex semantic queries that may involve a number of variables. Chaos theory is promising to facility content discovery in the Semantic Web. The chaos theory can be used to create useful content focal points from the chaotic mess of semantic data resources distributed across the Web. Instead of looking for repeatable semantic patterns, it looks for high-level correlations or trends by exploring the chaotic nature of the Web.

On the other hand, from the perspective of the Semantic Web, it may also be interesting to investigate how to use different CI techniques to resolve broader problems in the Semantic Web such as ontology evolution, query rewriting, provenance tracking, linked data mining, semantic routing optimization, etc. For examples, linked data mining is a promising field where CI methods can be particularly useful [52], given the astonishing size of the linked data generated so far and the decentralized nature of the data. Provenance tracking is another interesting field that remains unexplored. Provenance tracking requires tracing, recording and querying the paths of data production from one site to another site. Such path relations can be extremely complex in the context of the Semantic Web. Ant-inspired method may be helpful when dealing with such a complexity.

C. Research on Emergent Semantics and Self-Organizing Semantic Web

Considering the complexity of the Semantic Web, it is not far-fetched to regard it as a complex system where the chaotic and loosely-defined nature of the Web need to be tamed by novel approaches. In such a system, global structures embodying global semantic agreement may probably emerge from a multiplicity of pair-wise, local interactions between data sources and agents, generating eventually a self-organizing semantic infrastructure. These statements project the nature-inspired view of "emergent semantics in the Web and evolutionary Semantic Web" [33] that imposes a complex system perspective on the problem of dealing with semantics and intelligence on the Web.

We state that these open new fields study how semantics can emerge and semiotic relations can originate, spread, and evolve over time in a social Web by combining recent advances in a number of nature-inspired methods such as evolutionary computation, chaos theory, and self-organization systems.

D. Hybrid Approaches of EC, NN, and FL for Semantic Web Applications

The reality that the problems of vastness, autonomy, vagueness, randomness, and inconsistence exist simultaneously poses more difficulties for many Semantic Web applications. For example, mapping data across the Web induces the problem that the space of mapping possibilities among a multitude of data sources is especially rich. Meanwhile, resolving the mapping heterogeneity imposes the problem that the meaning of mapping is usually fuzzy. The algorithms need to take care of both uncertainty and scalability issues in many applications. The situation also exists in large-scale query routing and integrative reasoning among intelligent agents that may hold fuzzy knowledge on their status.

The way out of such conundrum may lay in a combination of variant CI techniques. For example, Neural Fuzzy System (NFS) [53] refers to the synthesis of neural network and fuzzy logic. The neuro-fuzzy combination results in a hybrid approach that fuzzy reasoning style is integrated with the learning and connectionist structure of neural networks. For another example, the synthesis of genetic algorithm with fuzzy logic

extends its capability of dealing with fuzzy information in complexity reduction and optimization [54] [55]. We believe it could be a considerable research direction to take in these hybrid approaches of EC-FL [54][55], NN-FL[53], or EC-NN [56] to design systematic solution to tackle the hybrid challenges faced by the Semantic Web community.

VII. Conclusions

The Semantic Web, as a decentralized complex system, is akin to be fuzzy and evolutionary. In this article, we have provided a comprehensive survey on the applications of variant Computational Intelligence methods to enhance a variety of Semantic Web applications. The survey consists of three aspects: fuzzy logic to deal with vagueness and uncertainty in Web semantics; evolutionary computations to deal with the vastness and tractability issues in storing, querying, reasoning and mapping semantic data; artificial neural network to improve the learning capability of the Semantic Web. Based on the survey of the existing approaches in the literature, some potential future research directions in this area have also been discussed and proposed.

Acknowledgments

We thank all of our colleagues including Dr. Jeff Pan, Dr. Tong Yu, and Dr. Chunyin Zhou, for the fruitful discussion. The work of the authors is funded by NSFC61070156, and national 863 program *China Cloud Initiative.*

References

[1] A. M. Collins and M. R. Quillian, "Retrieval time from semantic memory," *J. Verbal Learn. Verbal Behav.*, vol. 8, no. 2, pp. 240–247, 1960.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientif. Amer. Mag.*, 2001.

[3] M. Fischett, "The web turns 20: Linked data gives people power—Part 1 of 4," *Scientif. Amer.*, 2010.

[4] World Wide Web Consortium (W3C). W3C semantic web activity [Online]. Available: http://www.w3.org/2001/sw/

[5] M. J. Er and R. J. Oentaryo, "Computational intelligence: Methods and techniques [Book Review]," *IEEE Comput. Intell. Mag.*, vol. 6, no. 4, pp. 76–78, 2011.

[6] J. Mendel, L. Zadeh, E. Trillas, et al., "What computing with words means to me," *IEEE Comput. Intell. Mag.*, vol. 5, no. 1, pp. 20–26, 2010.

[7] C. A. Coello, "Evolutionary multi-objective optimization: A historical view of the field," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 28–36, 2006.

[8] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[9] T. Lukasiewicz and U. Straccia, "Managing uncertainty and vagueness in description logics for the Semantic Web," *J. Web Semantics*, vol. 6, no. 2, pp. 291–308, 2008.

[10] G. Stoilosa, G. Stamoub, and J. Z. Pan, "Fuzzy extensions of OWL: Logical properties and reduction to fuzzy description logics," *Int. J. Approx. Reason.*, vol. 51, no. 6, pp. 656–679, July 2010.

[11] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[12] C. Blum and D. Merkle, *Swarm Intelligence: Introduction and Applications*. New York: Springer-Verlag, 2008.

[13] D. Dasgupta, "Advances in artificial immune systems," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 40–49, 2006.

[14] G. Acampora, V. Loia, and M. Gaeta, "Exploring e-learning knowledge through ontological memetic agents," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 66–77, 2010.

[15] T. Berners-Lee. Linked data web architecture note, W3C [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html

[16] I. Horrocks, "From SHIQ and RDF to OWL: The making of a web ontology language," *J. Web Semantics: Sci. Services Agents World Wide Web*, vol. 1, no. 1, pp. 7–26, 2003.

[17] W3C Standard Specification. OWL 2 Web ontology language document overview [Online]. Available: http://www.w3.org/TR/owl2-overview/

[18] G. Stoilos, N. Simou, G. Stamou, and S. Kollias, "Uncertainty and the semantic web," *IEEE Intell. Syst.*, vol. 21, no. 5, pp. 84–87, 2006.

[19] A. Ferrara, D. Lorusso, G. Stamou, et al., "Resolution of conflicts among ontology mappings: A fuzzy approach," in *Proc. 3rd Int. Workshop Ontology Matching (OM),* Karlsruhe, Germany, 2008, pp. 121–130.

[20] M. Holi and E. Hyvonen, "Fuzzy view-based semantic search," in *Proc. Asian Semantic Web Conf.*, 2006, pp. 123–135.

[21] L. Zhang, Y. Yu, J. Zhou, et al., "An enhanced model for searching in semantic portals," in *Proc. Int. World Wide Web Conf.* ACM Press, 2005, pp. 453–462.

[22] J. Pan, G. Stamou, G. Stoilos, and E. Thomas, "Scalable querying services over fuzzy ontologies," in *Proc. Int. World Wide Web Conf.*, Beijing, 2008, pp. 340–348.

[23] A. Bahri, R. Bouaziz, and F. Gargouri, "Dealing with similarity relations in fuzzy ontologies," in *Proc. 16th IEEE Int. Conf. Fuzzy Systems*, 2007, pp. 1836–1841.

[24] N. Simou, T. Athanasiadis, G. Stoilos, and S. Kollias, "Image indexing and retrieval using expressive fuzzy description logics," *Signal, Image Video Process.*, vol. 2, no. 4, pp. 321–335, 2008.

[25] M. d'Aquin, J. Lieber, and A. Napoli, "Towards a semantic portal for oncology using a description logic with fuzzy concrete domains," in *Fuzzy Logic and the Semantic Web, Capturing Intelligence*, E. Sanchez, Ed. New York: Elsevier, 2006, pp. 379–393.

[26] G. Fenza, V. Loia, and S. Senatore, "A hybrid approach to Semantic Web services matchmaking," *Int. J. Approx. Reason.*, vol. 4, no. 8, pp. 808–828, 2008.

[27] V. Vaneková, J. Bella, P. Gurský, and T. Horváth, "Fuzzy RDF in the semantic web: Deduction and induction," in *Proc. Workshop Data Analysis*, 2005, pp. 16–29.

[28] G. Stoilos, G. Stamou, V. Tzouvaras, et al., "Fuzzy owl: Uncertainty and the Semantic Web," in *Proc. Int. Workshop OWL: Experiences and Directions*, 2005, pp. 230–242.

[29] B. Fernando and U. Straccia, "Fuzzy ontology representation using OWL 2," *Int. J. Approx. Reason.*, vol. 52, no. 7, pp. 1073–1094.

[30] U. Straccia, "Reasoning within fuzzy description logics," *J. Artif. Intell. Res.*, no. 14, pp. 137–166, 2007.

[31] G. Stoilos, G. Stamou, V. Tzouvaras, et al., "Reasoning with very expressive fuzzy description logics," *J. Artif. Intell. Res.*, vol. 30, no. 5, pp. 273–320, 2007.

[32] C. Liu, G. Qi, H. Wang, and Y. Yu, "Large scale fuzzy pD*reasoning using MapReduce," in *Proc. Int. Semantic Web Conf.*, 2011, pp. 405–420.

[33] S. Staab, "Emergent semantics," *IEEE Intell. Syst.*, vol. 17, no. 1, pp. 78–86, 2002.

[34] E. Oren, C. Guéret, and S. Schlobach, "Anytime query answering in RDF through evolutionary algorithms," *Lect. Notes Comput. Sci.*, vol. 5318, pp. 98–113.

[35] A. Hogenboom, V. L. Milea, F. Frasincar, and U. Kaymak, "Genetic algorithms for RDF query path optimization," in *Proc. Int. Workshop Nature Inspired Reasoning for the Semantic Web (CEUR Workshop Proc.*, vol. 419).

[36] J. Martinez-Gil and E. Alba, "Optimizing ontology alignments by using genetic algorithms," in *Proc. Int. Workshop Nature Inspired Reasoning for the Semantic Web (CEUR Workshop Proc.*, vol. 419).

[37] J. Wang, Z. Ding, and C. Jiang, "GAOM: Genetic algorithm based ontology matching," in *Proc. IEEE Asia-Pacific Conf. Services Computing*, 2006, pp. 617–620.

[38] N. P. Tizzo and J. M. A. Cardozo, "Automatic composition of semantic web services using A-Teams with genetic agents," in *Proc. IEEE Congr. Evolutionary Computation*, June 2011, pp. 370–377.

[39] H. Mühleisen, A. Augustin, T. Walther, et al., "A self-organized semantic storage service," in *Proc. 12th Int. Conf. Information Integration and Web-Based Applications and Services,* Paris, France, Nov. 2010, pp. 357–364.

[40] R. Tolksdorf and R. Menezes, "Using swarm intelligence in linda systems," in *Proc. 4th Int. Workshop Engineering Societies in the Agents World*, 2003, pp. 2004–2014.

[41] K. Dentler, C. Guéret, and S. Schlobach, "Semantic web reasoning by swarm intelligence," in *Proc. 5th Int. Workshop Scalable Semantic Web Knowledge Base Systems*, 2009.

[42] R. Kashif Ali and S. Cayzer, "AIS and semantic query," in *The Semantic Web: Research and Applications (Lecture Notes in Computer Science*, vol. 3532), pp. 1–13, 2005.

[43] A. Chortaras, G. B. Stamou, and A. Stafylopatis, "Learning ontology alignments using recursive neural networks," in *Proc. Int. Conf. Neural Networks (Lecture Notes in Computer Science*, vol. 3697), Poland. Berlin: Springer-Springer-Verlag, 2005, pp. 811–816.

[44] M. Mao, Y. Pengb, and M. Springc, "An adaptive ontology mapping approach with neural network based constraint satisfaction," *J. Web Semantics: Sci., Services Agents World Wide Web*, vol. 8, no. 1, pp. 14–25, Mar. 2010.

[45] S. C. Emil and A. L. Ioan, *Self-organizing Maps in Web Mining and Semantic Web*. Self-Organizing Maps, InTech, 2010.

[46] T. Honkela and M. Pöllä, "Concept mining with self-organizing maps for the semantic web," in *Advances in Self-Organizing Maps* (Lecture Notes in Computer Science, vol. 5629), pp. 98–106, 2009.

[47] R. C. Chen and C. H. Chuang, "Automating construction of a domain ontology using a projective adaptive resonance theory neural network and Bayesian network," *Expert Syst.*, vol. 25, no. 4, pp. 414–430.

[48] D. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. New York: Wiley, 2001.

[49] J. McClelland and D. Rumelhart, *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*. Cambridge, MA: MIT Press, 1988.

[50] S. Haykin, "Self-organizing maps," in *Neural Networks—A Comprehensive Foundation,* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[51] Y. Cao and J. Wu, "Dynamics of projective adaptive resonance theory model: The foundation of PART algorithm. *IEEE Trans. Neural Networks*," vol. 15, no. 2, pp. 245–260.

[52] D. S. Martino and A. Sperduti, "Mining structured data," *IEEE Comput. Intell. Mag.*, vol. 5, no. 1, pp. 42–49, 2010.

[53] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Upper Saddle River, NJ: Prentice Hall, 1996.

[54] O. Cordon, F. Herrera, F. Hoffmann, et al., *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. Singapore: World Scientific, 2002.

[55] A. P. Chen and Y. C. Hsu, "Dynamic physical behavior analysis for financial trading decision support," *IEEE Comput. Intell. Mag.*, vol. 5, no. 4, pp. 19–23, 2011.

[56] P. Durr, C. Mattiussi, and D. Floreano, "Genetic representation and evolvability of modular neural controllers," *IEEE Comput. Intell. Mag.*, vol. 5, no. 3, pp. 10–19, 2010.

Piotr Lipinski
University of Wroclaw, POLAND

Book Review

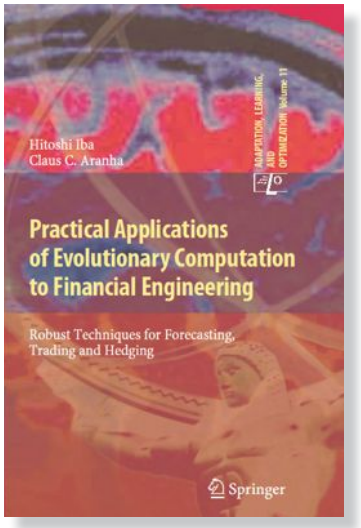
Practical Applications of Evolutionary Computation to Financial Engineering: Robust Techniques for Forecasting, Trading and Hedging, by Hitoshi Iba and Claus C. Aranha (Springer, 2012, 264 pp.) ISBN: 978-3-642-27647-7.

Although computational intelligence is often applied to financial engineering, there are still very few books related to this topic.

Applying computational intelligence to economic and financial data analysis has attracted more and more attention in recent years, mainly due to the increasing amount of economic and financial data available for analysis, the increasing number of different instruments possible for investing and the increasing number of various trader's preferences. This has made many classic approaches inefficient and has constituted an important challenge for emerging computational techniques.

Although computational intelligence is often applied to financial engineering, there are still very few books related to this topic. Most of them are either edited books [3, 4, 5, 7, 8] or advanced monographies [1, 2, 6] intended for academics and computer scientists rather than practitioners and financial engineers. The book written by Iba and Aranha bridges the gap between academics and practitioners. It should be comprehensible not only for computer scientists interested in some applications, but also for financial experts or even market traders interested in advanced methods of automated trading and new tools for market analysis.

This book presents a systematic framework for a few significant problems in financial engineering with some evolutionary approaches to them. The entire book is organized into seven chapters. The first two chapters cover the basics of evolutionary computations. The next chapter contains an introduction to financial engineering. The four consecutive chapters refer to popular financial engineering problems, such as financial data prediction, trend analysis, automated trading, and portfolio optimization.



The book begins with an introduction to evolutionary computation in the first chapter. It includes a short discussion on the basic concepts of the evolutionary approach, a review of four original evolutionary paradigms: Genetic Algorithms (GA), Evolution Strategies (ES), Genetic Programming (GP) and Evolutionary Programming (EP), and more detailed presentations of GA and GP. Finally, the authors explain their choice of GA and GP for solving financial engineering problems.

Chapter 2 addresses some advanced topics in evolutionary computation. The first of them concerns multiobjective optimization (MOO), which is a common issue in financial engineering, usually related to the trade-off between return and risk. The Vector Evaluated Genetic Algorithm (VEGA) is presented as an example of an evolutionary algorithm for finding Pareto fronts. Chapter 2 also introduces memes and memetic algorithms with the Lamarckian and Baldwinian evolution. The baldwinian memetic algorithm is discussed in the context of FX trading rule optimization, where it seems to outperform the classic evolutionary approach. The third part of Chapter 2 focuses on two real-valued algorithms, such as Differential Evolution and Particle Swarm Optimization, which are described in detail. Finally, the authors discuss generating random trees and using them to predict time series data.

Digital Object Identifier 10.1109/MCI.2012.2188593
Date of publication: 13 April 2012

This book offers a good entry point for practitioners interested in solving financial engineering problems by evolutionary computation or financial experts willing to possess some knowledge on evolutionary approaches to economic and financial data analysis.

Basic concepts of financial engineering are introduced in Chapter 3, which explains the differences between the technical and fundamental analysis, as well as briefly presenting some popular financial engineering problems which have been tackled with evolutionary algorithms in recent years, such as price prediction, trend analysis, automated stock trading and portfolio optimization.

Chapter 4 presents financial data prediction. It starts with an example of modeling time series in a GP simulator, the LGPC for Time Series Prediction software, and then proposes the Structured Representation On Genetic Algorithms for NOnlinear Function Fitting (STROGANOFF) approach, which integrates the multiple regression analysis with the GP optimization. Next, the authors present an example which implements STROGANOFF to predict a real-world time series containing price quotations from the Japanese stock market. Finally, they also discuss an extension of STROGANOFF by inductive GP and polynomial neural networks.

Chapter 5 focuses on trend analysis. It begins with some principles of data classification and then describes the Majority Voting Genetic Programming Classifier (MVGPC) for solving it. Next, an example of trend analysis on foreign exchange market data, namely detecting up and down trends in the euro/yen exchange rate, is presented. Finally, possible extensions of MVGPC with weighted voting and different learning methods are pointed out.

In Chapter 6, another popular problem of financial engineering, i.e. trading rule generation for foreign exchange (FX), is presented. It begins with a short historical review of automated trading for FX, which includes such approaches as the EDDIE system by Tsang, the Grammatical Evolution framework by Brabazon and O'Neill, the GA and GP approaches by Dempster and Jones as well as some other frameworks based on DE and PSO. Next, the authors propose three approaches: a price prediction based trading system, which uses the STROGANOFF framework; a GA-GP trading system, which optimizes parameters of technical indices; and a system based on DE and PSO for FX trading, which rates trends based on the relative position of different MA bands and uses them to make a buy, sell, open or close decision with a simple threshold rule.

Chapter 7 is centered on portfolio optimization and proposes two approaches: a simple Array-based GA and a more advanced Memetic Tree-based Genetic Algorithm (MTGA). The first approach is presented for illustration, as, due to its simplicity, it was not capable of solving more complex problems. However, the second approach, being much more advanced, was capable of selecting efficient portfolios on the benchmark data from the NASDAQ and S&P 500 stock market, which were used in the described experiments.

An additional value of this book, which makes it especially interesting for practitioners, includes the software packages described in the appendices which are available to the public. Appendix A refers to the Pareto GA simulator and the GP prediction system discussed in the earlier chapters, as well as the MVGPC system for trend analysis, the STROGANOFF system for time series prediction and the portfolio optimization suite. Appendix B focuses on the GAGPTrader system for automated trading, which may be obtained by the public as a trial version.

This book offers a good entry point for practitioners interested in solving financial engineering problems by evolutionary computation or financial experts willing to possess some knowledge on evolutionary approaches to economic and financial data analysis. One may find, however, that this book cannot be referred to as a regular handbook of evolutionary computation for financial engineering, due to the lack of information on significant research into some evolutionary algorithms, such as evolutionary strategies or grammatical evolution, and their applications to financial data analysis.

References

[1] R. Bauer, *Genetic Algorithms and Investment Strategies*. New York: Wiley, 1994.

[2] A. Brabazon and M. O'Neill, *Biologically Inspired Algorithms for Financial Modelling*. Berlin: Springer-Verlag, 2005.

[3] A. Brabazon and M. O'Neill, *Natural Computing in Computational Finance*. Berlin: Springer-Verlag, 2008.

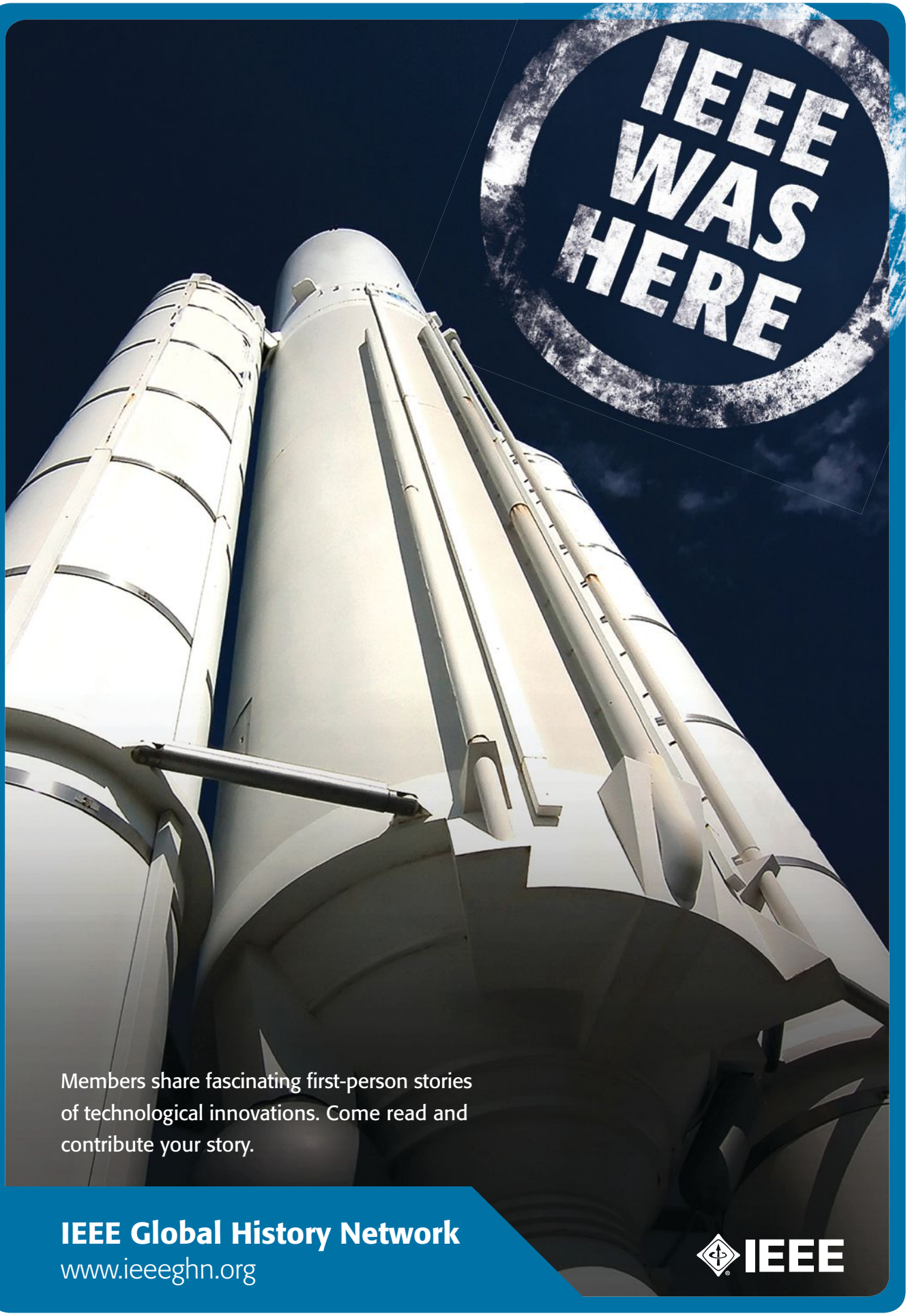
[4] S.-H. Chen, *Evolutionary Computation in Economics and Finance*. Physica-Verlag, 2002.

[5] S.-H. Chen, *Genetic Algorithms and Genetic Programming in Computational Finance*. Norwell, MA: Kluwer, 2002.

[6] A. L. G. Almanza and E. Tsang, *Evolutionary Applications for Financial Prediction: Classification Methods to Gather Patterns Using Genetic Programming*. VDM Verlag, 2011.

[7] J.-P. Rennard, *Handbook of Research on Nature-Inspired Computing for Economics and Management*. IGI Global, 2006.

[8] P. P. Wang, *Computational Intelligence in Economics and Finance*. Berlin: Springer-Verlag, 2003.


IEEE WAS HERE
Members share fascinating first-person stories of technological innovations. Come read and contribute your story.
IEEE Global History Network
www.ieeeghn.org
IEEE

Conference Calendar

Gary B. Fogel
Natural Selection, Inc., USA

* Denotes a CIS-Sponsored Conference
Δ Denotes a CIS Technical Co-Sponsored Conference

*** 2012 IEEE World Congress on Computational Intelligence (IEEE WCCI 2012)**
June 9–15, 2012
Place: Brisbane, Australia
General Chair: Hussein Abbass
http://www.ieee-wcci2012.org

Δ The Third International Conference on Swarm Intelligence 2012 (ICSI 2012)
June 17–20, 2012
Place: Shenzhen, China
http://www.ic-si.org/

Δ 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems 2012 (IPMU 2012)
July 9–13, 2012
Place: Catania, Italy
http://www.ipmu2012.unict.it/

*** 2012 IEEE Symposium on Computational Intelligence for Security and Defense Applications (IEEE CISDA 2012)**
July 11–13, 2012
Place: Ottawa, Canada
General Chair: Rami Abielmona
Website: http://ieee-cisda.org

Δ 9th International Symposium on Neural Networks (ISNN 2012)
July 11–14, 2012
Place: Shenyang, China
http://isnn.mae.cuhk.edu.hk

Digital Object Identifier 10.1109/MCI.2012.2188596
Date of publication: 13 April 2012

Δ International Conference on Brain Inspired Cognitive Systems (BICS 2012)
July 11–14, 2012
Place: Shenyang, China
http://bics2012.mae.cuhk.edu.hk

Δ 8th International Conference on Intelligent Computing (ICIC 2012)
July 25–29, 2012
Place: Huangshan, China
http://www.ic-ic.org/2012/index.htm

Δ North American Fuzzy Information Processing Society 2012 (NAFIPS 2012)
August 6–8, 2012
Place: Berkeley, CA, USA
http://www.ualberta.ca/~reformat/nafips2012/

Δ 4th International Conference on Awareness Science and Technology 2012 (iCAST 2012)
August 21–24, 2012
Place: Seoul, South Korea
http://icast2012.korea.ac.kr/

Δ The 12th UK Workshop on Computational Intelligence (UKCI 2012)
September 5–7, 2012
Place: Edinburgh, UK
http://www.macs.hw.ac.uk/~pdw/UKCI2012.html

Δ 6th IEEE International Conference on Intelligent Systems (IS 2012)
September 6–8, 2012
Place: Sofia, Bulgaria
http://www.ieee-is.org/

*** 2012 IEEE Conference on Computational Intelligence in Games**
September 12–15, 2012
Place: Granada, Spain
General Chair: Antonio Fernandez Leiva
Website: http://geneura.ugr.es/cig2012/

*** 2012 IEEE Computational Intelligence for Measurement Systems (IEEE CIMSA 2012)**
September 19–21, 2012
Place: Tianjin, China
General Chairs: Leonid Perlovsky and Fabio Scotti
http://cimsa2012.ieee-ims.org/

Δ 2012 Symposium on Neural Network Applications in Electrical Engineering (NEUREL 2012)
September 20–22, 2012
Place: Belgrade, Serbia
http://neurel.etf.rs

Δ Annual Conference of the Prognostics and Health Management Society (PHM 2012)
September 23–27, 2012
Place: Minneapolis, MN, USA
http://www.phmsociety.org/events/conference/phm/12

Δ The 1st International Conference on Cognitive Systems and Information Processing 2012 (ICCSI 2012)
September 27–30, 2012
Place: Beijing, China
Website: TBD

*** 2012 IEEE Conference on Development and Learning and Epigenetic Robotics (IEEE ICDL-EpiRob 2012)**
November 7–9, 2012
Place: San Diego, CA, USA
General Co-Chairs: Javier Movellan, Matthew Schlesinger, and Jochen Triesch
Website: TBD

Δ 19th International Conference on Neural Information Processing (ICONIP 2012)
November 26–29, 2012
Place: Doha, Qatar
http://www.iconip2012.org

*** 2013 IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2013)**
April 16–19, 2013
Place: Singapore
General Chair: P.N. Suganthan
http://ieee-ssci.org/

*** 2013 IEEE Congress on Evolutionary Computation (IEEE CEC 2013)**
July 3–6, 2013
Place: Cancun, Mexico
General Chair: Carlos Coello Coello
Website: http://www.cec2013.org/

*** 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2013)**
July 7–10, 2013
Place: Hyderabad, India
General Chair: Nik Pal
Website: TBD

*** 2014 IEEE Conference on Computational Intelligence in Financial Engineering and Economics**
March 27–28, 2014
Place: London, UK
General Chair: Antoaneta Serguieva
Website: TBD

*** 2014 IEEE World Congress on Computational Intelligence (IEEE WCCI 2014)**
July 6–14, 2014
Place: Beijing, China
General Chair: Derong Liu
Website: TBD

Advertisers Index

The Advertisers Index contained in this issue is compiled as a service to our readers and advertisers: the publisher is not liable for errors or omissions although every effort is made to ensure its accuracy. Be sure to let our advertisers know you found them through *IEEE Computational Intelligence Magazine*.

Company	page#	URL	Phone
IEEE Marketing	CVR4	www.ieee.org/ieeexplore	

IEEE Media Advertising Sales Offices

James A. Vick
Staff Director, Advertising Business
+1 212 419 7767; *Fax:* +1 212 419 7589
jv.ieeemedia@ieee.org

Marion Delaney
Advertising Sales Director
+1 415 863 4717; *Fax:* +1 415 863 4717
md.ieeemedia@ieee.org

Susan Schneiderman
Business Development Manager
+1 732 562 3946; *Fax:* +1 732 981 1855
ss.ieeemedia@ieee.org

Product Advertising

Mid-Atlantic
Lisa Rinaldo
+1 732 772 0160; *Fax:* +1 732 772 0164
lr.ieeemedia@ieee.org
NY, NJ, PA, DE, MD, DC, KY, WV

New England/South Central/ Eastern Canada
Jody Estabrook
+1 774 283 4528; *Fax:* +1 774 283 4527
je.ieeemedia@ieee.org
CT, ME, VT, NH, MA, RI, AR, LA, OK, TX.
CANADA: Nova Scotia, Prince Edward Island, Newfoundland,
New Brunswick, Quebec

Southwest
Thomas Flynn
+1 770 645 2944; *Fax:* +1 770 993 4423
tf.ieeemedia@ieee.org
VA, NC, SC, GA, FL, AL, MS, TN

Midwest/Central Canada
Dave Jones
+1 708 442 5633; *Fax*: +1 708 442 7620
dj.ieeemedia@ieee.org
IL, IA, KS, MN, MO, NE, ND, SD,
WI, OH. CANADA: Manitoba,
Saskatchewan, Alberta

Midwest/Ontario, Canada
Will Hamilton
+1 269 381 2156; *Fax:* +1 269 381 2556
wh.ieeemedia@ieee.org
IN, MI. CANADA: Ontario

West Coast/Mountain States/ Western Canada
Marshall Rubin
+1 818 888 2407; *Fax:* +1 818 888 4907
mr.ieeemedia@ieee.org
AZ, CO, HI, NM, NV, UT,
CA, AK, ID, MT, WY, OR, WA
CANADA: British Columbia

Europe/Africa/Middle East
Heleen Vodegel
+1 44 1875 825 700; *Fax:* +1 44 1875 825 701
hv.ieeemedia@ieee.org
Europe, Africa, Middle East

Asia/Far East/Pacific Rim
Susan Schneiderman
+1 732 562 3946; *Fax:* +1 732 981 1855
ss.ieeemedia@ieee.org
Asia, Far East, Pacific Rim, Australia,
New Zealand

Recruitment Advertising

Mid-Atlantic
Lisa Rinaldo
+1 732 772 0160; *Fax:* +1 732 772 0164
lr.ieeemedia@ieee.org
CT, NY, NJ, PA, DE, MD, DC, KY, WV

New England/Eastern Canada
Liza Reich
+1 212 419 7578; *Fax:* +1 212 419 7589
e.reich@ieee.org
ME, VT, NH, MA, RI. CANADA:
Nova Scotia, Prince Edward Island,
Newfoundland, New Brunswick, Quebec

Southeast
Cathy Flynn
+1 770 645 2944; *Fax:* +1 770 993 4423
cf.ieeemedia@ieee.org
VA, NC, SC, GA, FL, AL, MS, TN

Midwest/South Central/Central Canada
Darcy Giovingo
+1 847 498 4520; *Fax:* +1 847 498 5911
dg.ieeemedia@ieee.org
AR, LA, TX, OK, IL, IN, IA, KS, MI,
MN, NE, ND, SD, OH, WI, MO.
CANADA: Ontario, Manitoba,
Saskatchewan, Alberta

West Coast/Mountain States/ Southwest/Asia
Tim Matteson
+1 310 836 4064; *Fax:* +1 310 836 4067
tm.ieeemedia@ieee.org
AK, AZ, CA, CO, HI, ID, MT, NM,
NV, OR, UT, WA, WY.
CANADA: British Columbia

Europe/Africa/Middle East
Heleen Vodegel
+1 44 1875 825 700
Fax: +1 44 1875 825 701
hv.ieeemedia@ieee.org
Europe, Africa, Middle East

Digital Object Identifier 10.1109/MCI.2012.2188598

General Chair
Hussein Abbass, Australia
University of New South Wales
School of Engineering & IT
UNSW@ADFA, Northcott Drive, Canberra
Australia
Tel: +61-2-62688158
Fax: +61-2-62688933
h.abbass@adfa.edu.au

Conference Chairs
IJCNN Cesare Alippi
alippi@elet.polimi.it
FUZZ-IEEE Bernadette Bouchon-Meunier
Bernadette.Bouchon-Meunier@lip6.fr
IEEE CEC Garry Greenwood
greenwd@ece.pdx.edu

Program Chairs
IJCNN Kate Smith-Miles
kate.smith-miles@sci.monash.edu.au
FUZZ-IEEE James M. Keller
KellerJ@missouri.edu
IEEE CEC Xiaodong Li
xiaodong.li@rmit.edu.au

Finance Chair
Gary G. Yen
gyen@okstate.edu

Plenary Session Chair
Danil Prokhorov
dvprokhorov@gmail.com

Invited Session Chair
Chin-Teng Lin
ctlin@mail.nctu.edu.tw

Special Sessions Chairs
IJCNN Brijesh Verma
b.verma@cqu.edu.au
FUZZ-IEEE Laszlo T. Koczy
koczy@tmit.bme.hu
IEEE CEC Mengjie Zhang
mengjie.zhang@ecs.vuw.ac.nz

Tutorial Chairs
IJCNN Haibo He,
he@ele.uri.edu
FUZZ-IEEE Scott Dick
dick@ee.ualberta.ca
IEEE CEC Janet Wiles
janetw@itee.uq.edu.au

Workshop Chair
Slawomir Wesolkowsk
s.wesolkowski@ieee.org

Competitions Chairs
Sung-Bae Cho, Korea
sbcho@yonsei.ac.kr
Philip Hingston
p.hingston@ecu.edu.au
Simon Lucas
sml@essex.ac.uk

Publicity Chairs
Ponnuthurai N. Suganthan
EPNSugan@ntu.edu.sg
Jürgen Branke
juergen.branke@wbs.ac.uk
Kay C Wiese
wiese@cs.sfu.ca

Publications Chairs
Daryl Essam
d.essam@adfa.edu.au
Ruhul Sarker
r.sarker@adfa.edu.au

Registration Chair
Kathryn Merrick
kathryn.merrick@adfa.edu.au

Local Arrangements Chair
Marcus Randall
mrandall@bond.edu.au

2012 IEEE World Congress on Computational Intelligence

June 10-15, 2012
Brisbane Convention & Exhibition Centre
Brisbane, Australia
www.ieee-wcci2012.org

The 2012 IEEE World Congress on Computational Intelligence (IEEE WCCI 2012) is the largest technical event in the field of computational intelligence. It will host three conferences: The 2012 International Joint Conference on Neural Networks (IJCNN 2012), the 2012 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2012), and the 2012 IEEE Congress on Evolutionary Computation (IEEE CEC 2012). WCCI 2012 will provide a stimulating forum for scientists, engineers, educators, and students from all over the world to discuss and present their research findings on computational intelligence.

IEEE WCCI 2012 will be held in sunny Brisbane, a major Australian city with beautiful surroundings, warm weather, and a number of beaches and resorts. Australia is the sixth largest country in the world. It's about the same size as the 48 mainland states of the USA and 50% larger than Europe. Attractions in Brisbane include: Gold Coast, Sunshine Coast, Warner Brothers Movie World, Sea World, Wet'n'Wild Water World, Whitewater World, Dreamworld, Mt Cootha Lookout and Botanic Gardens, museums and culture centres.

Call For Participation

IEEE WCCI 2012 invites researchers in the field of Computational Intelligence, and those wishing to get to understand Computational Intelligence, to participate in the large number of activities during the Congress.

Join us during IEEE WCCI 2012. The week is full of carefully-chosen premium-quality activities.

Hear from our renewed Computational Intelligence Plenary and Invited Speakers in 2012:

- Dan Ashlock, University of Guelph, Canada
- Bernard De Baets, University of Gent, Belgium
- Piero P. Bonissonem Chief Scientist, GE Global Research, USA
- Vladimir Cherkassky, University of Minnesota, USA
- Nikola Kasabov, Auckland University of Technology, New Zealand
- Zbyzek Michalewicz, University of Adelaide, Australia
- Risto Miikkulainen, University of Texas at Austin, USA
- Maria Rifqi, University of Panthéon-Assas, France
- Jennie Si, Arizona State University, USA
- KC Tan, National University of Singapore, Singapore
- Kazuo Tanaka, University of Electro-Communications, Japan
- Lloyd Watts, Chief Technology Officer, Audience, USA
- Xin Yao, University of Birmingham, UK

Attend some of the 35+ tutorials on offer

Attend one of the three workshops on offer

- Entropy, Information and Complexity in Computational Intelligence
- Technology Solutions for Humanitarian Healthcare Applications
- Age-friendly Intelligent Computing

Watch and possibly participate in one of the eleven exciting competitions such as:

- Human vs. Computer Go
- Human-like Bots
- PacMac with Ghosts
- Turing Test Track of the 2012 Mario AI Championship

Three different travel grants and funding schemes:

- IEEE Computational Intelligence Society Students Travel Grants
- Queensland Government Support to Participants from Developing Countries
- Student Travel Fellowships for Spiking Neural Networks (Sponsored by Brain Corp)

Sponsors

IEEE WCCI Sponsor:

IEEE Computational Intelligence Society (CIS)

IJCNN Joint Sponsor:

International Neural Network Society (INNS)

IEEE CEC:

previously co-sponsored by the former Evolutionary Programming Society and the Institution of Engineering and Technology

Important Dates

Early Registration Deadline
April 2, 2012

Conference Dates
June 10-15, 2012

Digital Object Identifier 10.1109/MCI.2012.2188601