Cristian S. Calude
Michael J. Dinneen
Gheorghe Păun
Grzegorz Rozenberg
Susan Stepney (Eds.)

LNCS 4135

Unconventional Computation

5th International Conference, UC 2006
York, UK, September 2006
Proceedings

Springer

Cristian S. Calude Michael J. Dinneen
Gheorghe Păun Grzegorz Rozenberg
Susan Stepney (Eds.)

Unconventional Computation

5th International Conference, UC 2006
York, UK, September 4-8, 2006
Proceedings

Springer

Volume Editors

Cristian S. Calude
Michael J. Dinneen
University of Auckland, Department of Computer Science
Private Bag 92019, Auckland, New Zealand
E-mail: {cristian,mjd}@cs.auckland.ac.nz

Gheorghe Păun
Institute of Mathematics of Romanian Academy
P.O. Box 1-764, 014700 Bucureşti, Romania
and
Sevilla University, Department of Computer Science and AI
Avda Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: gpaun@us.es

Grzegorz Rozenberg
Leiden University, Leiden Center of Advanced Computer Science (LIACS)
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
E-mail: rozenber@liacs.nl

Susan Stepney
University of York, Department of Computer Science
Heslington, York, YO10 5DD, UK
E-mail: susan@cs.york.ac.uk

Library of Congress Control Number: 2006931474

CR Subject Classification (1998): F.1, F.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-38593-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-38593-6 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11839132 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 4135

Preface

The 5th International Conference on Unconventional Computation, UC 2006, organized under the auspices of the EATCS by the Centre for Discrete Mathematics and Theoretical Computer Science of the University of Auckland, and the Department of Computer Science of the University of York, was held in York, UK, September 4–8, 2006.

York combines evidence of a history going back to Roman times with a bustling modern city center. The Minster, built on the foundations of the Roman city and an earlier Norman cathedral, is among the finest Gothic cathedrals, and dominates the city. Romans, Vikings, and more recent history are commemorated in a number of top-class museums, as well as being apparent in the architecture of the city.

The series of International Conferences on Unconventional Computation (UC), `https://www.cs.auckland.ac.nz/CDMTCS/conferences/uc/`, is devoted to all aspects of unconventional computation, theory as well as experiments and applications. Typical, but not exclusive, topics are: natural computing including quantum, cellular, molecular, neural and evolutionary computing; chaos and dynamical systems-based computing; and various proposals for computations that go beyond the Turing model.

The first venue of the Unconventional Computation Conference (formerly called Unconventional Models of Computation) was Auckland, New Zealand in 1998; subsequent sites of the conference were Brussels, Belgium in 2000, Kobe, Japan in 2002, and Seville, Spain in 2005.

The titles of volumes of the past UC conferences are the following:

1. C.S. Calude, J. Casti, M.J. Dinneen (eds.). *Unconventional Models of Computation*, Springer, Singapore, 1998, viii + 426 pp. ISBN: 981-3083-69-7.
2. I. Antoniou, C.S. Calude, M.J. Dinneen (eds.). *Unconventional Models of Computation, UMC'2K*, Springer, London, December 2000, xi + 301 pp. ISBN 1-85233-417-0.
3. C.S. Calude, M.J. Dinneen, F. Peper (eds.). *Third International Conference, UMC 2002*, Proceedings Lecture Notes in Computer Science, Vol. 2509, Springer, Heidelberg, 2002, vii + 331 pp. ISBN: 3-540-44311-8.
4. C.S. Calude, M.J. Dinneen, M.J. Pérez-Jiménez, Gh. Păun, G. Rozenberg (eds.). *Proc. 4th International Conference Unconventional Computation*, Lecture Notes in Computer Science, Vol. 3699, Springer, Heidelberg, 2005, xi + 267 pp. ISBN: 3-540-29100-8.

The Steering Committee of the series of International Conferences on Unconventional Computation includes T. Bäck (Leiden, The Netherlands), C.S. Calude (Auckland, New Zealand, Co-chair), L.K. Grover (Murray Hill, NJ, USA), J. van Leeuwen (Utrecht, The Netherlands), S. Lloyd (Cambridge, MA, USA), Gh. Păun

(Bucharest, Romania, and Seville, Spain), T. Toffoli (Boston, MA, USA), C. Torras (Barcelona, Spain), G. Rozenberg (Leiden, The Netherlands, and Boulder, Colorado, USA, Co-chair), A. Salomaa (Turku, Finland).

The five key-note speakers of the conference were:

1. Gerard Dreyfus (ESPCI, Paris, France): *Graph Machines and Their Applications to Computer-Aided Drug Design: A New Approach to Learning from Structured Data*
2. Michael C. Mozer (Department of Computer Science, and Institute of Cogntive Science, University of Colorado, USA): *Rational Models of Cognitive Control*
3. Reidun Twarock (University of York, UK): *Self-Assembly in Viruses*
4. Erik Winfree (Computer Science and Computation & Neural Systems, California Institute of Technology): *Fault-Tolerance in Biochemical Systems*
5. Damien Woods (University College Cork, Ireland): *Optical Computing and Computational Complexity*

UC 2006 included the following tutorials:

1. Andrew Adamatzky, Benjamin De Lacy Costello, Tetsuya Asai (Computing, Engineering and Mathematical Sciences, University of the West of England, Bristol, UK): *Reaction-Diffusion Computers*
2. Cristian S. Calude (University of Auckland, New Zealand): *Computing with Randomness*
3. Nataša Jonoska (University of South Florida, USA), and Darko Stefanovic (University of New Mexico, USA): *Biomolecular Automata*
4. Viv Kendon (University of Leeds, UK): *Quantum Computing*
5. Joseé del R. Millán (Institute for Systems, Informatics and Safety Joint Research Centre, Ispra, Italy): *Brain Signal Analysis*
6. Christof Teuscher (LANL, USA): *To Compute, or not to Compute*

The workshop “From Utopian to Genuine Unconventional Computers” was part of this year's conference.

The Programme Committee thanks the much appreciated work done by the paper reviewers for the conference. These experts were: Nevil Brownlee, Sam Braunstein, Douglas S. Bridges, Matteo Cavaliere, Cristian S. Calude, S. Barry Cooper, Jack Copeland, David Corne, Gabor Csardi, Erzsebet Csuhaj-Varjú, Michael J. Dinneen, Peter Erdi, Marian Gheorghe, Georgy Gimel'farb, James Goodman, Jozef Gruska, Oscar H. Ibarra, Mario de Jesus Pérez-Jiménez, Nataša Jonoska, Jarko Kari, Jan van Leeuwen, Chang Li, Rossella Lupacchini, Joseé del R. Millán, Pablo Moscato, Andrei Păun, Gheorghe Păun, Ion Petre, Vladimir Rogojin, Ulrich Speidel, Susan Stepney, Karl Svozil, Carme Torras, Christof Teuscher, Hiroshi Umeo.

The Programme Committee consisting of J.-P. Banâtre (Rennes, France), S. Braunstein (York, UK), C.S. Calude (Auckland, New Zealand, Co-chair), B. Cooper (Leeds, UK), D. Corne (Exeter, UK), M.J. Dinneen (Auckland, New Zealand, Secretary), P. Érdi (Kalamazoo, MI, USA), E. Goles (Santiago, Chile),

N. Jonoska (Tampa, FL, USA), J. Kari (Turku, Finland), J. van Leeuwen (Utrecht, Netherlands), R. Lupacchini (Bologna, Italy), J. del R. Millan (Ispra, Italy), Gh. Păun (Bucharest, Romania, and Seville, Spain, Co-chair), M.J. Pérez-Jiménez (Seville, Spain), I. Petre (Turku, Finland), P. Prusinkiewicz (Calgary, Canada), C. Teuscher (LANL, Los Alamos, USA), C. Torras (Barcelona, Spain), H. Umeo (Osaka, Japan), S. Stepney (York, UK), K. Svozil (Vienna, Austria), selected 17 papers (out of 36) to be presented as regular contributions.

We extend our thanks to all members of the Conference Committee, particularly to L. Caves, E. Clark, K. Clegg, G. Danks, O. Leyser (Co-chair), F. Polack, S. Stepney (Co-chair), J. Timmis, H. Turner, A. Weeks, J. Wright, for their invaluable organizational work.

We thank the University of York and the Centre for Discrete Mathematics of the University of Auckland for their technical support. The hospitality of our hosts, the Department of Computer Science of the University of York, is much appreciated.

The conference was partially supported by the Department of Biology of the University of York, the Enterprise and Innovation office of of the University of York, Microsoft Research, EPSRC, and the University consortium "White Rose"; we extend to all our gratitude.

It is a great pleasure to acknowledge the fine cooperation with the *Lecture Notes in Computer Science* team of Springer for producing this volume in time for the conference.

June 2006

C.S. Calude
M.J. Dinneen
Gh. Păun
G. Rozenberg
S. Stepney

Lecture Notes in Computer Science

For information about Vols. 1–4055

please contact your bookseller or Springer

Vol. 4185: R. Mizoguchi, Z. Shi, F. Giunchiglia (Eds.), The Semantic Web – ASWC 2006. XX, 778 pages. 2006.

Vol. 4180: M. Kohlhase, OMDoc – An Open Markup Format for Mathematical Documents [version 1.2]. XIX, 428 pages. 2006. (Sublibrary LNAI).

Vol. 4176: S.K. Katsikas, J. Lopez, M. Backes, S. Gritzalis, B. Preneel (Eds.), Information Security. XIV, 548 pages. 2006.

Vol. 4163: H. Bersini, J. Carneiro (Eds.), Artificial Immune Systems. XII, 460 pages. 2006.

Vol. 4162: R. Královič, P. Urzyczyn (Eds.), Mathematical Foundations of Computer Science 2006. XV, 814 pages. 2006.

Vol. 4159: J. Ma, H. Jin, L.T. Yang, J.J.-P. Tsai (Eds.), Ubiquitous Intelligence and Computing. XXII, 1190 pages. 2006.

Vol. 4158: L.T. Yang, H. Jin, J. Ma, T. Ungerer (Eds.), Autonomic and Trusted Computing. XIV, 613 pages. 2006.

Vol. 4155: O. Stock, M. Schaerf (Eds.), Reasoning, Action and Interaction in AI Theories and Systems. XVIII, 343 pages. 2006. (Sublibrary LNAI).

Vol. 4153: N. Zheng, X. Jiang, X. Lan (Eds.), Advances in Machine Vision, Image Processing, and Pattern Analysis. XIII, 506 pages. 2006.

Vol. 4152: Y. Manolopoulos, J. Pokorný, T. Sellis (Eds.), Advances in Databases and Information Systems. XV, 448 pages. 2006.

Vol. 4151: A. Iglesias, N. Takayama (Eds.), Mathematical Software - ICMS 2006. XVII, 452 pages. 2006.

Vol. 4146: J.C. Rajapakse, L. Wong, R. Acharya (Eds.), Pattern Recognition in Bioinformatics. XIV, 186 pages. 2006. (Sublibrary LNBI).

Vol. 4144: T. Ball, R.B. Jones (Eds.), Computer Aided Verification. XV, 564 pages. 2006.

Vol. 4139: T. Salakoski, F. Ginter, S. Pyysalo, T. Pahikkala, Advances in Natural Language Processing. XVI, 771 pages. 2006. (Sublibrary LNAI).

Vol. 4138: X. Cheng, W. Li, T. Znati (Eds.), Wireless Algorithms, Systems, and Applications. XVI, 709 pages. 2006.

Vol. 4137: C. Baier, H. Hermanns (Eds.), CONCUR 2006 – Concurrency Theory. XIII, 525 pages. 2006.

Vol. 4135: C.S. Calude, M.J. Dinneen, G. Păun, G. Rozenberg, S. Stepney (Eds.), Unconventional Computation. X, 267 pages. 2006.

Vol. 4134: K. Yi (Ed.), Static Analysis. XIII, 443 pages. 2006.

Vol. 4133: J. Gratch, M. Young, R. Aylett, D. Ballin, P. Olivier (Eds.), Intelligent Virtual Agents. XIV, 472 pages. 2006. (Sublibrary LNAI).

Vol. 4130: U. Furbach, N. Shankar (Eds.), Automated Reasoning. XV, 680 pages. 2006. (Sublibrary LNAI).

Vol. 4129: D. McGookin, S. Brewster (Eds.), Haptic and Audio Interaction Design. XII, 167 pages. 2006.

Vol. 4128: W.E. Nagel, W.V. Walter, W. Lehner (Eds.), Euro-Par 2006 Parallel Processing. XXXIII, 1221 pages. 2006.

Vol. 4127: E. Damiani, P. Liu (Eds.), Data and Applications Security XX. X, 319 pages. 2006.

Vol. 4124: H. de Meer, J.P. G. Sterbenz (Eds.), Self-Organising Systems. XIV, 261 pages. 2006.

Vol. 4121: A. Biere, C.P. Gomes (Eds.), Theory and Applications of Satisfiability Testing - SAT 2006. XII, 438 pages. 2006.

Vol. 4119: C. Dony, J.L. Knudsen, A. Romanovsky, A. Tripathi (Eds.), Advanced Topics in Exception Handling Components. X, 302 pages. 2006.

Vol. 4117: C. Dwork (Ed.), Advances in Cryptology - CRYPTO 2006. XIII, 621 pages. 2006.

Vol. 4116: R. De Prisco, M. Yung (Eds.), Security and Cryptography for Networks. XI, 366 pages. 2006.

Vol. 4115: D.-S. Huang, K. Li, G.W. Irwin (Eds.), Computational Intelligence and Bioinformatics, Part III. XXI, 803 pages. 2006. (Sublibrary LNBI).

Vol. 4114: D.-S. Huang, K. Li, G.W. Irwin (Eds.), Computational Intelligence, Part II. XXVII, 1337 pages. 2006. (Sublibrary LNAI).

Vol. 4113: D.-S. Huang, K. Li, G.W. Irwin (Eds.), Intelligent Computing, Part I. XXVII, 1331 pages. 2006.

Vol. 4112: D.Z. Chen, D. T. Lee (Eds.), Computing and Combinatorics. XIV, 528 pages. 2006.

Vol. 4111: F.S. de Boer, M.M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects. VIII, 447 pages. 2006.

Vol. 4110: J. Díaz, K. Jansen, J.D.P. Rolim, U. Zwick (Eds.), Approximation, Randomization, and Combinatorial Optimization. XII, 522 pages. 2006.

Vol. 4109: D.-Y. Yeung, J.T. Kwok, A. Fred, F. Roli, D. de Ridder (Eds.), Structural, Syntactic, and Statistical Pattern Recognition. XXI, 939 pages. 2006.

Vol. 4108: J.M. Borwein, W.M. Farmer (Eds.), Mathematical Knowledge Management. VIII, 295 pages. 2006. (Sublibrary LNAI).

Vol. 4106: T.R. Roth-Berghofer, M.H. Göker, H. A. Güvenir (Eds.), Advances in Case-Based Reasoning. XIV, 566 pages. 2006. (Sublibrary LNAI).

Vol. 4104: T. Kunz, S.S. Ravi (Eds.), Ad-Hoc, Mobile, and Wireless Networks. XII, 474 pages. 2006.

Vol. 4099: Q. Yang, G. Webb (Eds.), PRICAI 2006: Trends in Artificial Intelligence. XXVIII, 1263 pages. 2006. (Sublibrary LNAI).

Vol. 4098: F. Pfenning (Ed.), Term Rewriting and Applications. XIII, 415 pages. 2006.

Vol. 4097: X. Zhou, O. Sokolsky, L. Yan, E.-S. Jung, Z. Shao, Y. Mu, D.C. Lee, D. Kim, Y.-S. Jeong, C.-Z. Xu (Eds.), Emerging Directions in Embedded and Ubiquitous Computing. XXVII, 1034 pages. 2006.

Vol. 4096: E. Sha, S.-K. Han, C.-Z. Xu, M.H. Kim, L.T. Yang, B. Xiao (Eds.), Embedded and Ubiquitous Computing. XXIV, 1170 pages. 2006.

Vol. 4095: S. Nolfi, G. Baldassare, R. Calabretta, D. Marocco, D. Parisi, J.C. T. Hallam, O. Miglino, J.-A. Meyer (Eds.), From Animals to Animats 9. XV, 869 pages. 2006. (Sublibrary LNAI).

Vol. 4094: O. H. Ibarra, H.-C. Yen (Eds.), Implementation and Application of Automata. XIII, 291 pages. 2006.

Vol. 4093: X. Li, O.R. Zaïane, Z. Li (Eds.), Advanced Data Mining and Applications. XXI, 1110 pages. 2006. (Sublibrary LNAI).

Vol. 4092: J. Lang, F. Lin, J. Wang (Eds.), Knowledge Science, Engineering and Management. XV, 664 pages. 2006. (Sublibrary LNAI).

Vol. 4091: G.-Z. Yang, T. Jiang, D. Shen, L. Gu, J. Yang (Eds.), Medical Imaging and Augmented Reality. XIII, 399 pages. 2006.

Vol. 4090: S. Spaccapietra, K. Aberer, P. Cudré-Mauroux (Eds.), Journal on Data Semantics VI. XI, 211 pages. 2006.

Vol. 4089: W. Löwe, M. Südholt (Eds.), Software Composition. X, 339 pages. 2006.

Vol. 4088: Z.-Z. Shi, R. Sadananda (Eds.), Agent Computing and Multi-Agent Systems. XVII, 827 pages. 2006. (Sublibrary LNAI).

Vol. 4087: F. Schwenker, S. Marinai (Eds.), Artificial Neural Networks in Pattern Recognition. IX, 299 pages. 2006. (Sublibrary LNAI).

Vol. 4085: J. Misra, T. Nipkow, E. Sekerinski (Eds.), FM 2006: Formal Methods. XV, 620 pages. 2006.

Vol. 4084: M.A. Wimmer, H.J. Scholl, Å. Grönlund, K.V. Andersen (Eds.), Electronic Government. XV, 353 pages. 2006.

Vol. 4083: S. Fischer-Hübner, S. Furnell, C. Lambrinoudakis (Eds.), Trust and Privacy in Digital Business. XIII, 243 pages. 2006.

Vol. 4082: K. Bauknecht, B. Pröll, H. Werthner (Eds.), E-Commerce and Web Technologies. XIII, 243 pages. 2006.

Vol. 4081: A. M. Tjoa, J. Trujillo (Eds.), Data Warehousing and Knowledge Discovery. XVII, 578 pages. 2006.

Vol. 4080: S. Bressan, J. Küng, R. Wagner (Eds.), Database and Expert Systems Applications. XXI, 959 pages. 2006.

Vol. 4079: S. Etalle, M. Truszczyński (Eds.), Logic Programming. XIV, 474 pages. 2006.

Vol. 4077: M.-S. Kim, K. Shimada (Eds.), Geometric Modeling and Processing - GMP 2006. XVI, 696 pages. 2006.

Vol. 4076: F. Hess, S. Pauli, M. Pohst (Eds.), Algorithmic Number Theory. X, 599 pages. 2006.

Vol. 4075: U. Leser, F. Naumann, B. Eckman (Eds.), Data Integration in the Life Sciences. XI, 298 pages. 2006. (Sublibrary LNBI).

Vol. 4074: M. Burmester, A. Yasinsac (Eds.), Secure Mobile Ad-hoc Networks and Sensors. X, 193 pages. 2006.

Vol. 4073: A. Butz, B. Fisher, A. Krüger, P. Olivier (Eds.), Smart Graphics. XI, 263 pages. 2006.

Vol. 4072: M. Harders, G. Székely (Eds.), Biomedical Simulation. XI, 216 pages. 2006.

Vol. 4071: H. Sundaram, M. Naphade, J.R. Smith, Y. Rui (Eds.), Image and Video Retrieval. XII, 547 pages. 2006.

Vol. 4070: C. Priami, X. Hu, Y. Pan, T.Y. Lin (Eds.), Transactions on Computational Systems Biology V. IX, 129 pages. 2006. (Sublibrary LNBI).

Vol. 4069: F.J. Perales, R.B. Fisher (Eds.), Articulated Motion and Deformable Objects. XV, 526 pages. 2006.

Vol. 4068: H. Schärfe, P. Hitzler, P. Øhrstrøm (Eds.), Conceptual Structures: Inspiration and Application. XI, 455 pages. 2006. (Sublibrary LNAI).

Vol. 4067: D. Thomas (Ed.), ECOOP 2006 – Object-Oriented Programming. XIV, 527 pages. 2006.

Vol. 4066: A. Rensink, J. Warmer (Eds.), Model Driven Architecture – Foundations and Applications. XII, 392 pages. 2006.

Vol. 4065: P. Perner (Ed.), Advances in Data Mining. XI, 592 pages. 2006. (Sublibrary LNAI).

Vol. 4064: R. Büschkes, P. Laskov (Eds.), Detection of Intrusions and Malware & Vulnerability Assessment. X, 195 pages. 2006.

Vol. 4063: I. Gorton, G.T. Heineman, I. Crnkovic, H.W. Schmidt, J.A. Stafford, C.A. Szyperski, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 394 pages. 2006.

Vol. 4062: G. Wang, J.F. Peters, A. Skowron, Y. Yao (Eds.), Rough Sets and Knowledge Technology. XX, 810 pages. 2006. (Sublibrary LNAI).

Vol. 4061: K. Miesenberger, J. Klaus, W. Zagler, A.I. Karshmer (Eds.), Computers Helping People with Special Needs. XXIX, 1356 pages. 2006.

Vol. 4060: K. Futatsugi, J.-P. Jouannaud, J. Meseguer (Eds.), Algebra, Meaning, and Computation. XXXVIII, 643 pages. 2006.

Vol. 4059: L. Arge, R. Freivalds (Eds.), Algorithm Theory – SWAT 2006. XII, 436 pages. 2006.

Vol. 4058: L.M. Batten, R. Safavi-Naini (Eds.), Information Security and Privacy. XII, 446 pages. 2006.

Vol. 4057: J.P.W. Pluim, B. Likar, F.A. Gerritsen (Eds.), Biomedical Image Registration. XII, 324 pages. 2006.

Vol. 4056: P. Flocchini, L. Gąsieniec (Eds.), Structural Information and Communication Complexity. X, 357 pages. 2006.

¥385.00元

Table of Contents

Invited Papers

Graph Machines and Their Applications to Computer-Aided Drug Design: A New Approach to Learning from Structured Data
Aurélie Goulon, Arthur Duprat, Gérard Dreyfus 1

Rational Models of Cognitive Control
Michael C. Mozer 20

Fault-Tolerance in Biochemical Systems
Erik Winfree 26

Optical Computing and Computational Complexity
Damien Woods 27

Regular Papers

If a Tree Casts a Shadow Is It Telling the Time?
Russ Abbott 41

Peptide Computing – Universality and Theoretical Model
M. Sakthi Balan, Helmut Jürgensen 57

Handling Markov Chains with Membrane Computing
Mónica Cardona, M. Angels Colomer, Mario J. Pérez-Jiménez, Alba Zaragoza 72

Approximation Classes for Real Number Optimization Problems
Uffe Flarup, Klaus Meer 86

Physical Systems as Constructive Logics
Peter Hines 101

On Spiking Neural P Systems and Partially Blind Counter Machines
Oscar H. Ibarra, Sara Woodworth, Fang Yu, Andrei Păun 113

Chemical Information Processing Devices Constructed Using a Nonlinear Medium with Controlled Excitability
Yasuhiro Igarashi, Jerzy Gorecki, Joanna Natalia Gorecka 130

Flexible Versus Rigid Tile Assembly
Nataša Jonoska, Gregory L. McColm 139

On Pure Catalytic P Systems
Shankara Narayanan Krishna 152

Mapping Non-conventional Extensions of Genetic Programming
W.B. Langdon 166

The Number of Orbits of Periodic Box-Ball Systems
Akihiro Mikoda, Shuichi Inokuchi, Yoshihiro Mizoguchi, Mitsuhiko Fujio 181

The Euclid Abstract Machine: Trisection of the Angle and the Halting Problem
Jerzy Mycka, Francisco Coelho, José Félix Costa 195

$1/f$ Noise in Elementary Cellular Automaton Rule 110
Shigeru Ninagawa 207

A Light-Based Device for Solving the Hamiltonian Path Problem
Mihai Oltean 217

Optimizing Potential Information Transfer with Self-referential Memory
Mikhail Prokopenko, Daniel Polani, Peter Wang 228

On the Power of Bio-Turing Machines
H. Ramesh, Shankara Narayanan Krishna, Raghavan Rama 243

Ergodic Dynamics for Large-Scale Distributed Robot Systems
Dylan A. Shell, Maja J. Matarić 254

Author Index 267

Graph Machines and Their Applications to Computer-Aided Drug Design: A New Approach to Learning from Structured Data

Aurélie Goulon[1], Arthur Duprat[1, 2], and Gérard Dreyfus[1]

[1] Laboratoire d'Électronique,
[2] Laboratoire de Chimie Organique, (CNRS UMR 7084)
École Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
(ESPCI-ParisTech)
10 rue Vauquelin, 75005 PARIS, France
Aurelie.Goulon@espci.fr, Arthur.Duprat@espci.fr,
Gerard.Dreyfus@espci.fr

Abstract. The recent developments of statistical learning focused on *vector machines,* which learn from examples that are described by vectors of features. However, there are many fields where structured data must be handled; therefore, it would be desirable to learn from examples described by *graphs. Graph machines* learn real numbers from graphs. Basically, for each graph, a separate learning machine is built, whose algebraic structure contains the same information as the graph. We describe the training of such machines, and show that virtual leave-one-out, a powerful method for assessing the generalization capabilities of conventional vector machines, can be extended to graph machines. Academic examples are described, together with applications to the prediction of pharmaceutical activities of molecules and to the classification of properties; the potential of graph machines for computer-aided drug design are highlighted.

1 Introduction

Whether neural networks still fall in the category of "unconventional" computational methods is a debatable question, since that technique is well understood and widely used at present; its advantages over conventional regression methods are well documented and mathematically proven. Neural networks are indeed conventional in that they learn from *vector* data: typically, the variables of the neural model are in the form of a vector of numbers. Therefore, before applying learning techniques to neural networks, or any other conventional learning machine (Support Vector Machine, polynomial, multilinear model, etc.), the available data must be turned into a vector of variables; the learning machine then performs a mapping of a set of input vectors to a set of output vectors. In most cases, the output is actually a scalar, so that the mapping is from $\mathbb{R}^n$ to $\mathbb{R}$, where n is the dimension of the input vectors. When modeling a physical process for instance, the factors that have an influence on the quantity to be modeled are known from prior analysis, so that the construction of the vector of

C.S. Calude et al. (Eds.): UC 2006, LNCS 4135, pp. 1 – 19, 2006.

variables is straightforward, requiring simply normalization, and possibly variable selection by statistical methods.

In many cases of interest, however, encoding the data into a vector cannot be performed without information loss. Such is the case whenever the information to be learnt from is structured, i.e. is naturally encoded into a graph. In scene analysis for instance, a scene can be encoded into a graph that describes the relationships between the different parts of the scene. In computer-aided drug design, the purpose of learning is a mapping of the space of molecules to the space of pharmaceutical activities; in most cases, the structure of the molecule explains, to a large extent, its activity. Since molecular structures are readily described by graphs, QSAR (Quantitative Structure-Activity Relationships) aims at mapping the space of the graphs of molecular structures to the space of molecular activities or properties.

In the present paper, we describe an approach to learning that can be termed unconventional insofar as its purpose is a mapping of graphs to real numbers (or vectors) instead of a mapping of vectors to real numbers. The latter quantities may be either real-valued (graph regression) or binary (graph classification). The idea of learning from graphs (and generally structured data) can be traced back to the early days of machine learning, when Recursive Auto-Associative Memories (RAAMs) were designed for providing compact representations of trees [1]. It evolved subsequently to Labeled RAAMs [2], recursive networks [3], and graph machines (for a review of the development of numerical machine-learning from structured data, see [4]).

The first part of the paper is devoted to a description of graph machines and of some didactic, toy problems. It will also be shown that model selection methods that are proved to be efficient for conventional machine learning can be extended to graph machines. The second part of the paper will describe novel applications of graph machines to the prediction and classification of the properties or activities of molecules, a research area known as QSAR/QSPR (Quantitative Structure-Activity/Structure-Property Relationships). We show that graph machines are particularly powerful in that area, because they avoid a major problem of that field: the design, computation and selection of molecular descriptors.

2 Graph Machines

We first provide the definitions and notations for handling acyclic graphs, and the construction of graph machines from general graphs (possibly cyclic). Academic problems are described as illustrations. It is shown that the training and model selection methods developed for vector machines can be extended to graph machines.

2.1 Handling Directed Acyclic Graphs

Definitions: we consider the mapping from a set of acyclic graphs G to a set of real-valued numbers.

For each acyclic graph G_i of G, a parameterized function g^i $\mathbb{R}^n \rightarrow \mathbb{R}$ is constructed, which is intended (i) to encode the structure of the graph [5], and (ii) to provide a prediction of the quantity of interest, e.g. a property or an activity of the molecule, from G_i. It is constructed as follows. A parameterized function $f_{\boldsymbol{\theta}}$ ("node function") is associated to each node. $\boldsymbol{\theta}$ denotes the vector of parameters of the node function. All nodes, except the root node, have the same node function $f_{\boldsymbol{\theta}}$; those functions are combined in such a way that g^i has the same structure as graph G_i: if an edge from node k to node l exists in the graph, then the value of the node function associated to node k is a variable of the node function associated to node l. The root node may be assigned a different function, denoted by $F_{\boldsymbol{\Theta}}$, where $\boldsymbol{\Theta}$ is the vector of parameters of $F_{\boldsymbol{\Theta}}$. If the node functions are neural networks, the g^i's are termed recursive neural networks [3].

Notations: the following notations are used throughout the paper.
We denote by $\mathbf{x}_j$ the (optional) vector of labels that provide information about node j of graph G_i. The size of the label vector is denoted by X_i; it is the same for all nodes of a given graph. Therefore, the parameterized function associated to G_i will be denoted as $g^i_{\boldsymbol{\theta},\boldsymbol{\Theta}}\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\nu_i}\right)$, where ν_i is the number of nodes of graph G_i. If no specific information about the node is necessary, $g^i_{\boldsymbol{\theta},\boldsymbol{\Theta}}$ has no variable: its value depends only on the structure of graph G_i.

We denote by $\mathbf{z}_j$ the vector of variables of the node function $f_{\boldsymbol{\theta}}(\mathbf{z}_j)$ of the non-root node j of graph G_i. Denoting by d_j the in-degree of non-root node j, and defining $M_i = \arg\max_j d_j$, the size of vector $\mathbf{z}_j$ is equal to $D_i = M_i + X_i + 1$. The vectors of variables of the node functions $f_{\boldsymbol{\theta}}(\mathbf{z}_j)$ are constructed as follows: for all j, the first component z_j^0 is equal to 1 (the "bias" if $f_{\boldsymbol{\theta}}(\mathbf{z}_j)$ is a neural network, the constant term if $f_{\boldsymbol{\theta}}(\mathbf{z}_j)$ is an affine function); for node j, of in-degree d_j, components z_j^1 to $z_j^{d_j}$ are the values of the node functions assigned to the parent nodes of node j; if $d_j < M_i$, components $z_j^{d_j+1}$ to $z_j^{M_i}$ are equal to zero; if $X_i \neq 0$, components $z_j^{M_i+1}$ to $z_j^{M_i+X_i}$ are the labels of node j.

We denote by $\mathbf{y}_i$ the vector of variables of the node function $F_{\boldsymbol{\Theta}}\left(\mathbf{y}_i\right)$ of the root node of graph G_i. The size of $\mathbf{y}_i$ is $\Delta_i = d_r + X_r + 1$, where d_r denotes the in-degree of the root node and X_r the size of its vector of labels. y_i^0 is equal to 1 (bias), y_i^1 to $y_i^{d_r}$ are the values of the node functions assigned to the parent nodes of the root node, $y_i^{d_r+1}$ to $y_i^{d_r+X_r}$ are the labels assigned to the root node.

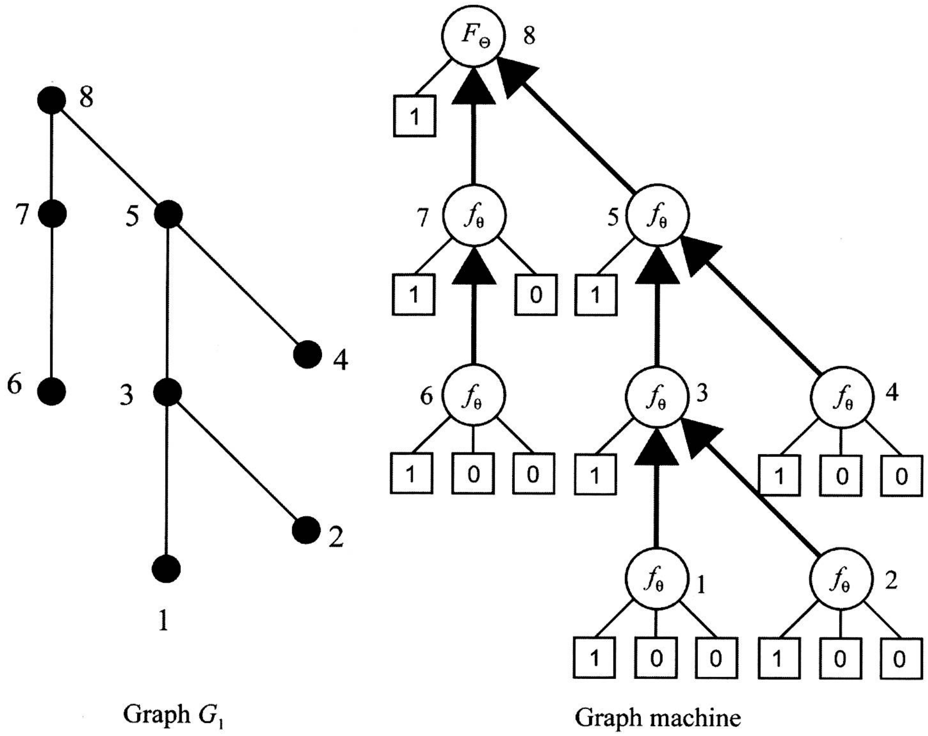
As an example, Fig. 1 shows an acyclic graph G_1 with maximum in-degree $M_1 = 2$; the corresponding graph machine is:

$$g^1_{\boldsymbol{\theta},\boldsymbol{\Theta}}\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_8\right) = F_{\boldsymbol{\Theta}}\left(\mathbf{y}_1\right) = F_{\boldsymbol{\Theta}}\left(\mathbf{x}_8, f_{\boldsymbol{\theta}}\left(\mathbf{x}_7, f_{\boldsymbol{\theta}}\left(\mathbf{z}_6\right), 0\right), f_{\boldsymbol{\theta}}\left(\mathbf{x}_5, f_{\boldsymbol{\theta}}\left(\mathbf{z}_4\right), f_{\boldsymbol{\theta}}\left(\mathbf{x}_3, f_{\boldsymbol{\theta}}\left(\mathbf{z}_2\right), f_{\boldsymbol{\theta}}\left(\mathbf{z}_1\right)\right)\right)\right) \quad (1)$$

If no information about the nodes is required by the problem at hand ($X_1 = 0$), one has $D = 3$, and:

$$\mathbf{z}_1 = \mathbf{z}_2 = \mathbf{z}_4 = \mathbf{z}_6 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^{\mathrm{T}}, \ \mathbf{z}_3 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}\left(\mathbf{z}_1\right) & f_{\boldsymbol{\theta}}\left(\mathbf{z}_2\right) \end{pmatrix}^{\mathrm{T}},$$

$$\mathbf{z}_5 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}\left(\mathbf{z}_3\right) & f_{\boldsymbol{\theta}}\left(\mathbf{z}_4\right) \end{pmatrix}^{\mathrm{T}}, \ \mathbf{z}_7 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}\left(\mathbf{z}_6\right) & 0 \end{pmatrix}^{\mathrm{T}}, \ \mathbf{y}_1 = \begin{pmatrix} 1 & f_{\boldsymbol{\theta}}\left(\mathbf{z}_5\right) & f_{\boldsymbol{\theta}}\left(\mathbf{z}_7\right) \end{pmatrix}^{\mathrm{T}}.$$



Graph G_1

Graph machine

Fig. 1. An acyclic graph and its graph machine

2.2 Cyclic Graphs

Graph machines handle cyclic graphs and parallel edges. To that effect, the initial graph is preprocessed by deleting a number of edges equal to the number of cycles, and all parallel edges but one; moreover, a label is assigned to each node: it is equal to the degree of the node, thereby retaining the information about the original graph structure. Finally, a root node is chosen and the edges are assigned orientations, according to an algorithm described in [6].

2.3 The Training of Graph Machines

Graph machines are trained in the usual framework of empirical risk minimization. A cost function $J(\boldsymbol{\Theta},\boldsymbol{\theta})$ is defined, and its minimum with respect to the parameters is sought, given the available training data. The cost function takes into account the discrepancy between the predictions of the models and the observations present in the training set, and may include regularization terms, e.g.:

$$J\left(\boldsymbol{\theta},\boldsymbol{\Theta}\right)=\sum_{i=1}^{N}\left(y^i-g^i_{\boldsymbol{\theta},\boldsymbol{\Theta}}\right)^2+\lambda_1\left\|\boldsymbol{\theta}\right\|+\lambda_2\left\|\boldsymbol{\Theta}\right\|, \tag{2}$$

where N is the size of the training set, y^i is the value of the i-th observation of the quantity to be modeled, and λ_1 and λ_2 are suitably chosen regularization constants.

Since the parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$ must be identical within each function g^i and across all those functions, one must resort to the so-called *shared weight* trick; the k-th component of the gradient of the cost function can be written as

$$\frac{\partial J(\boldsymbol{\theta},\boldsymbol{\Theta})}{\partial \theta_k} = \sum_{i=1}^{N} \frac{\partial J^i}{\partial \theta_k}, \tag{3}$$

where J^i is the contribution of example i to the cost function. We denote by $n^i_{\theta_k}$ the number of occurrences of parameter θ_k in acyclic graph G_i; if the root is assigned the same parameterized function as the other nodes, then $n^i_{\theta_k}$ is equal to the number of nodes in graph G_i. The shared weight trick consists in setting

$$\frac{\partial J^i}{\partial \theta_k} = \sum_{j=1}^{n^i_{\theta_k}} \frac{\partial J^i}{\partial \theta_{k_j}}, \tag{4}$$

so that one has finally:

$$\frac{\partial J(\boldsymbol{\theta},\boldsymbol{\Theta})}{\partial \theta_k} = \sum_{i=1}^{N} \sum_{j=1}^{n^i_{\theta_k}} \frac{\partial J^i}{\partial \theta_{k_j}}. \tag{5}$$

Relation (5) is subsequently used for minimizing cost function (2) by any suitable gradient descent algorithm (Levenberg-Marquardt, BFGS, conjugate gradient, ...).

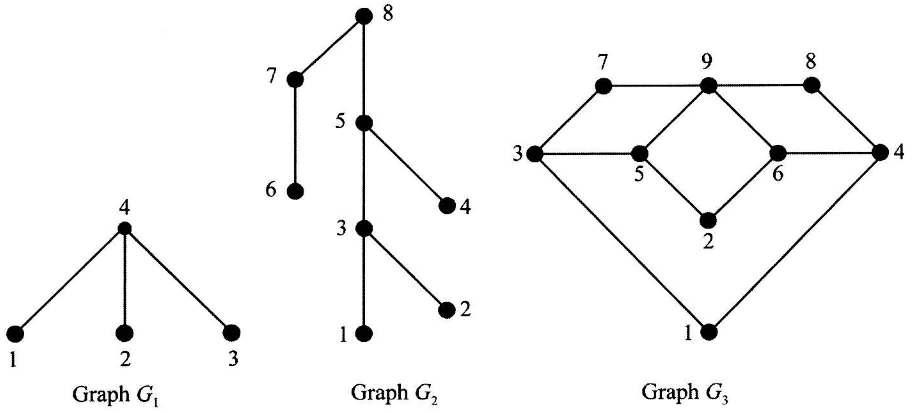
If functions $f_{\boldsymbol{\theta}}$ and $F_{\boldsymbol{\Theta}}$ are neural networks, the usual backpropagation algorithm may be conveniently used for computing the gradient; otherwise, one resorts to numerical estimations thereof.

2.4 Didactic Examples: Learning the Number of Nodes and the Number of Cycles of a Graph

In the present section, two simple examples are provided, whose solutions can be worked out analytically because they are linear. In both cases, we consider the training set made of three graphs, shown on Fig. 2.

Learning the number of nodes of a graph: first, assume that it is desired to learn, from examples, the number of nodes of a graph. Then the desired mapping is: $G_1 \to 4$; $G_2 \to 8$; $G_3 \to 9$. Moreover, generalization should be performed by using the node functions thus obtained in any other graph machine, i.e. to compute the number of nodes of any graph.

The first step consists in constructing directed acyclic graphs (DAGs) from the initial graphs. The construction of the DAGs is obvious for G_1 and G_2. Since graph G_3 has four cycles, four edges must be deleted. Fig. 3 shows the directed acyclic graphs on which the graph machines will be based.



Graph G_1 Graph G_2 Graph G_3

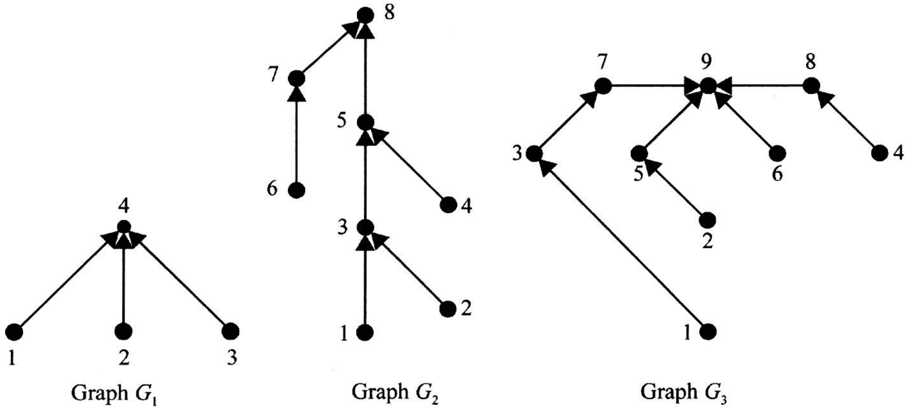
Fig. 2. A training set

The node function $f_\boldsymbol{\theta}$ is sought in the family of affine functions $f_\boldsymbol{\theta}(\mathbf{z}) = \sum_{j=0}^{D-1} \theta_j z_j$, and $F_\boldsymbol{\Theta}$ is taken identical to $f_\boldsymbol{\theta}$. Since the presence or absence of an edge is irrelevant for the computation of the number of nodes, no label is necessary: $X_1 = X_2 = X_3 = 0$. The node functions being the same for all graphs of the training set, we take $D = \max_i M_i + 1 = 5$. Since all edges are equivalent, one has $\theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta$. Therefore, there are actually two independent parameters only.

The obvious solution is $\theta_0 = \theta = 1$. For graph G_1 for instance, one has:

$$g^1_{\boldsymbol{\theta},\boldsymbol{\Theta}}(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4) = f_\boldsymbol{\theta}\left(1, f_\boldsymbol{\theta}(\mathbf{z}_1), f_\boldsymbol{\theta}(\mathbf{z}_2), f_\boldsymbol{\theta}(\mathbf{z}_3), 0\right) = \theta_0 + 3\theta\theta_0 = 4,$$

where $\mathbf{z}_1 = \mathbf{z}_2 = \mathbf{z}_3 = \begin{pmatrix}1 & 0 & 0 & 0 & 0\end{pmatrix}^\mathrm{T}$.



Graph G_1 Graph G_2 Graph G_3

Fig. 3. The acyclic graphs derived from the training set shown on Fig. 2