Nieuwsbrief van de Nederlandse Vereniging voor Theoretische Informatica

Jan Willem Klop, Jan Rutten, Susanne van Dam (redactie) *

Inhoudsopgave

*CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands. Email: susanne@cwi.nl.

1 Van de Redactie

Beste NVTI-leden,

Graag bieden wij u hierbij het achtste nummer aan van de jaarlijkse NVTI-Nieuwsbrief. Bij het samenstellen hebben we weer de formule van de vorige zeven nummers gevolgd. Zo vindt u naast het programma van de jaarlijkse Theoriedag en de bijgewerkte ledenlijst ook weer enkele bijdragen van collega's met een korte inleiding in hun speciale gebied van expertise. Evenals voorgaande jaren zouden deze Nieuwsbrief en de Theoriedag niet tot stand hebben kunnen komen zonder de financiele steun van onze sponsors: NWO-EW, Elsevier Publishing Company, en de onderzoekscholen IPA en SIKS. Namens de NVTI gemeenschap onze hartelijke dank voor deze middelen die ons voortbestaan mogelijk maken!

De redactie,
Jan Willem Klop (jwk@cwi.nl)
Jan Rutten (janr@cwi.nl)
Susanne van Dam (susanne @cwi.nl)

2 Samenstelling Bestuur

Prof.dr. J.C.M. Baeten (TUE)
Dr. H.L. Bodlaender (UU)
Prof.dr. J.W. Klop (VUA/CWI/KUN) voorzitter
Prof.dr. J.N. Kok (RUL)
Prof.dr. J.-J.Ch. Meyer (UU)
Prof.dr. G.R. Renardel de Lavalette (RUG)
Prof.dr. G. Rozenberg (RUL)
Prof.dr. J.J.M.M. Rutten (CWI/VUA) secretaris
Dr. L. Torenvliet (UvA)

3 Van de voorzitter

Geacht NVTI-lid,

Ook dit jaar heeft het Bestuur zich beijverd om een interessante Theoriedag te organiseren, met prominente sprekers uit binnen- en buitenland. De Theoriedag zal gehouden worden op vrijdag 5 maart, in Hoog-Brabant, Utrecht. We hopen en vertrouwen erop dat het programma voor velen van u weer interessant is. Graag tot ziens op 5 maart in Utrecht!

Jan Willem Klop, voorzitter NVTI

NVTI Lifetime Achievement Award

On Monday, June 30, a special event in the ICALP conference at the TUE was organized, with a two-fold purpose. One was the presentation of the EATCS Lifetime Achievement Award to Grzegorz Rozenberg, from the hands of Mogens Nielsen and Jan van Leeuwen. The other was the presentation of the NVTI Lifetime Achievement Award to N.G. (Dick) de Bruijn, from the hands of Jan Willem Klop. The text of the laudatio for the latter is included below.

LAUDATIO for N.G. (Dick) de Bruijn

June 30, 2003,
Eindhoven

Introduction. 'NVTI' is de Nederlandse Vereniging voor Theoretische Informatica, the Dutch Association for Theoretical Computer Science. It is my task and my pleasure to present Professor de Bruijn, on behalf of the NVTI, with an Award and a Present, symbolizing our recognition and gratitude for his everlasting and profoundly influential contributions to theoretical computer science. This laudatio is an attempt of highlighting some of the achievements of Professor de Bruijn.

First some numbers and statistics. Nicolaas Govert de Bruijn, or shorter, Dick de Bruijn, was born July 9, 1918, which will be next week 85 years ago. He studied mathematics at Leiden University from '36 to '41 and wrote a Ph.D. thesis at the Vrije Universiteit Amsterdam on modular functions. His career started as a researcher at Philips Research Laboratory in Eindhoven from '44 to '46. He was professor of mathematics at the TUD ('46-'52), then at the UVA ('52-'60), then since 1960 at the TUE. Two more numbers: he wrote over 200 papers, the first one in 1937.

Mathematics. This occasion is not primarily meant to deal with his mathematical achievements, but rather those in theoretical computer science. Nevertheless, his mathematical work will have been in some sense a preparation for the later work in computer science, leading up to it and setting the stage. De Bruijn worked in many areas, number theory, analysis, combinatorics. I remember learning as a student in a course on combinatorics the beautiful counting theorem of De Bruijn and Polya. But, I'm not qualified to discuss De Bruijn's mathematical achievements.

Applied type theory. Barendregt has explained to me that de Bruijn is to be considered as the founding father of 'applied type theory'. In 1968 de Bruijn started his celebrated AUTOMATH project, originally meant for the verification of mathematical texts. Notably, the book on analysis by Landau was verified and found correct, but for one mistake. Later, it turned out that AUTOMATH had a much wider scope.

With AUTOMATH, de Bruijn was far ahead of his time, at least a decade. This had some disadvantages. One is the lack of recognition and understanding that the early Automath project encountered sometimes. The other technical disadvantage was that computers were much inferior to what they are now.

The AUTOMATH project had a tremendous impact both on the applied side as well as on the foundational or theoretical side. On the applied side the project delivered the first working system (in 1969) for proof checking, on the basis of dependent types, the Curry-Howard-de Bruijn isomorphism, and the calculus of definitions (PAL). De Bruijn had the crucial insight that typed lambda calculi can be the basis for verification systems. Two of the calculi in Barendregt's lambda cube are members of the AUTOMATH family of calculi, and the designers of the most general system in this cube, Coquand and Huet, refer to de Bruijn as the inspration for their ideas. The initial lack of recognition is nowadays replaced by a general world-wide recognition of the pioneering role of the AUTOMATH project.

Throughout the years De Bruijn has developed AUTOMATH further with numerous ideas and devices. One such idea was his introduction of nameless dummies in lambda calculus (1973), or so-called the Bruijn indices, that solves the perennial problem of renaming bound variables in a deceptively simple and straightforward way. The Bruijn indices are now completely standard in the literature of categorical combinatory logic, and of explicit substitution calculi. This family of calculi is since the past say ten or twelve years extensively investigated but the first calculus of explicit substitution dates according to Lescanne, from a note by de Bruijn, the $C\lambda\xi\phi$-calculus of 1972.

There was the notion of telescopes and segments, a precursor to a treatment of context calculi.

Other achievements. De Bruijn's achievements did not stay restricted to the strategic most important AUTOMATH work. There was much more.

Crystallography and the theory of quasi-crystals. Everyone has seen the beautiful inherently nonperiodic tilings of the plane, discovered by Penrose. De Bruijn gave a deep analysis of their properties by pentagrids. It turned out that his work was not merely in the area of recreational mathematics, but was of direct relevance in crystallography.

By the way, de Bruijn made also many contributions to the less serious area of recreational mathematics, analyzing games like Solitaire and many more.

Term rewriting. One area were I was a close witness to an achievement of Professor de Bruijn was term rewriting. After a survey lecture that I had given here in Eindhoven, with Dick de Bruijn in the audience, he sent me a day later a letter with a ten page typescript, that had remained in his drawers for over ten years with the question whether the theorem had any interest. The note gave a marvellous strong lemma in abstract rewriting, generalizing several known lemma's about confluence, a 'master theorem'. The insight had a strong combinatorial nature, not surprisingly. De Bruijn's lemma was further developed by van Oostrom, and the resulting theorem of decreasing diagrams of De Bruijn and van Oostrom will be without doubt a standard theorem in future textbooks.

Biological memory and the conscious mind. Finally, two more less well-known achievements of de Bruijn should be mentioned. One is his work on modeling biological memory and the conscious mind. The other concerns Mathematical Vernacular, an enterprise, together with Rob Nederpelt, that again follows in a coherent way from de Bruijn's earlier work. Mathematical vernacular is in between natural language and formalized language, and important in interaction between computer and humans. I would not be surprised if the future will show that also in these endeavours Dick de Bruijn was ahead of his times.

This brings me to the Award that symbolically expresses the recognition of our community for de Bruijn's groundbreaking work in so many areas of theoretical computer science. The award is accompanied by a Present. This present is an artistic rendering of what may be the most important datatype in de Bruijn's work (after natural numbers), namely a tree. It is the tree of Pythagoras, designed by emeritus professor in mathematics Koos Verhoeff. It was manufactured by Hans de Koning, Eindhoven. Its branches symbolize the branching out of De Bruijn's many interests. The tree is non-periodic. Further observations about this tree are best left to Professor Dick de Bruijn.

4 Theoriedag 2004

Theoriedag 2004 van de NVTI
(Nederlandse Vereniging voor Theoretische Informatica)

Vrijdag 5 maart 2004
Vergadercentrum Hoog Brabant, Utrecht

Het is ons een genoegen u uit te nodigen tot het bijwonen van de Theoriedag 2004 van de NVTI, de Nederlandse Vereniging voor Theoretische Informatica, die zich ten doel stelt de theoretische informatica te bevorderen en haar beoefening en toepassingen aan te moedigen. De Theoriedag 2004 zal gehouden worden op vrijdag 5 maart 2004, in Vergadercentrum Hoog Brabant te Utrecht, gelegen in winkelcenrum Hoog Catherijne, op enkele minuten loopafstand van CS Utrecht, en is een voortzetting van de reeks jaarlijkse bijeenkomsten van de NVTI die negen jaar geleden met de oprichtingsbijeenkomst begon.

Evenals vorige jaren hebben wij een aantal prominente sprekers uit binnen- en buitenland bereid gevonden deze dag gestalte te geven met voordrachten over recente en belangrijke stromingen in de theoretische informatica. Naast een wetenschappelijke inhoud heeft de dag ook een informatief gedeelte, in de vorm van een algemene vergadering waarin de meest relevante informatie over de NVTI gegeven zal worden, alsmede een presentatie door dr. M. Kas (NWO/EW), die informatie zal verstrekken over de diverse subsidieprogramma's van NWO voor de informatica.

Programma (samenvattingen volgen beneden)

9.30-10.00:	Ontvangst met koffie
10.00-10.10:	Opening
10.10-11.00:	Lezing Prof.dr. U. Schoening (University of Ulm) Titel: Algorithms for satisfiability testing
11.00-11.30:	Koffie
11.30-12.20:	Lezing Dr. P. Gruenwald (CWI) Titel: Two theories of information: Shannon & Kolmogorov
12.20-12.50:	Lezing Dr. M. Kas (NWO/EW) Titel: De omgekeerde wereld. Over de informaticaplannen van het NWO-gebied Exacte Wetenschappen.
12.50-14.10:	Lunch (Zie beneden voor registratie)
14.10-15.00:	Lezing Prof.dr. S. Abramsky (University of Oxford) Titel: A Categorical Semantics of Quantum Protocols.
15.00-15.20:	Thee
15.20-16.10:	Lezing Prof.dr. J. van Benthem (UvA, Stanford University) Titel: Games, logic and computation
16.10-16.40:	Algemene ledenvergadering NVTI

Lunchdeelname

Het is mogelijk aan een georganiseerde lunch deel te nemen; hiervoor is aanmelding verplicht. Dit kan per email of telefonisch bij Susanne van Dam (susanne@cwi.nl, 020-592 4189), tot een week voor de bijeenkomst (27 februari). De kosten kunnen ter plaatse voldaan worden; deze bedragen ongeveer Euro 14. Wij wijzen erop dat in de onmiddellijke nabijheid van de vergaderzaal ook uitstekende lunchfaciliteiten gevonden kunnen worden, voor wie niet aan de georganiseerde lunch wenst deel te nemen.

Lidmaatschap NVTI

Alle leden van de voormalige WTI (Werkgemeenschap Theoretische Informatica) zijn automatisch lid van de NVTI geworden. Aan het lidmaatschap zijn geen kosten verbonden; u krijgt de aankondigingen van de NVTI per email of anderszins toegestuurd. Was u geen lid van de WTI en wilt u lid van de NVTI worden: u kunt zich aanmelden bij Susanne van Dam (susanne@cwi.nl), met vermelding van de relevante gegevens (naam, voorletters, affiliatie indien van toepassing, correspondentieadres, email, URL, telefoonnummer).

Steun

De activiteiten van de NVTI worden mede mogelijk gemaakt door de ondersteuning (financieel en anderszins) van de volgende instellingen: NWO/EW, CWI, Onderzoeksscholen IPA en SIKS, Elsevier Science B.V.

Samenvattingen van de lezingen

Algorithms for Satisfiability Testing
Spreker: Uwe Schoening (University of Ulm)

Despite its NP-completeness, designing good satisfiability testing algorithms (SAT solver) has a lot of practical applications. Such SAT solvers use various heuristics which make them hard to analyze theoretically. We present a couple of theoretically well understood algorithms for k-SAT. It was especially realized during the last years that certain probabilistic algorithms can have better running times than the best known deterministic ones.

Two Theories of Information: Shannon & Kolmogorov
Spreker: Peter Grunwald (CWI)

We introduce, compare and contrast the theories of Shannon information and Kolmogorov complexity. We investigate the extent to which these theories have a common purpose and where they are fundamentally different. We discuss the fundamental relations 'entropy = expected Kolmogorov complexity' and 'Shannon mutual information = expected algorithmic information'. We show how 'universal coding/modeling' (a central idea in practical data compression) may be viewed as a middle ground between the two theories, and how it leads to the 'minimum description length principle', a practically useable theory for statistical inference with arbitarily complex models. (Joint work with P.M.B. Vitanyi.)

A Categorical Semantics of Quantum Protocols
Spreker: Samson Abramsky (University of Oxford, joint work with Bob Coecke)

We study quantum information and computation from a novel point of view. We show how the essential structures found in key quantum information protocols such as teleportation, logic-gate teleportation, and entanglement-swapping can be captured at the abstract level of compact-closed categories with biproducts. This abstract and structural point of view opens up new possibilities for describing and reasoning about quantum systems. It also shows the degrees of axiomatic freedom: we can show what requirements are placed on the (semi)ring of 'scalars' C(I, I), where C is the category and I is the tensor unit, in order to perform various protocols such as teleportation. Our formalism captures both the information-flow aspect of the protocols, and the branching due to quantum indeterminism. This contrasts with the standard accounts, in which the classical information flows are 'outside' the usual quantum-mechanical formalism.

Games, logic and computation
Spreker: Johan van Benthem (UvA, Stanford University)
`http://staff.science.uva.nl/~johan/`

Games are a natural model for interaction and communication. Logic and games go well together, witness the widespread use of 'logic games' for argumentation, semantic evaluation, or model comparison. But the connection runs more deeply. Logic provides a natural fine-structure to game theory. In particular, both strategic and extensive game forms look like models that logicians are used to in studies of actions, knowledge, and belief revision. We will show how key issues in analyzing games link up with recent work in modal logics of knowledge, communication and action. Our examples are (a) reasoning about strategic equilibrium, (b) information update during a game, (c) revision of expectations about the future.

Literature

1999 - ..., "Logic in Games", electronic lecture notes, ILLC Amsterdam & philosophy Stanford (occasional paper versions).

2001A, 'An Essay on Sabotage and Obstruction', ILLC Amsterdam. To appear in D. Hutter ed., "Festschrift for Joerg Siekmann", Springer Verlag.

2001B, 'Dynamic Epistemic Logic', "Bulletin of Economic Research" 53:4, 219-248 (Proceedings LOFT-4, Torino).

2002A, 'Extensive Games as Process Models', in M. Pauly & P. Dekker, eds., special issue of "Journal of Logic, Language and Information" 11, 289-313.

2002, 'One is a Lonely Number: on the logic of communication', Tech Report PP-2002-27, ILLC Amsterdam. To appear in P. Koepke et al. eds., "Colloquium Logicum, Muenster 2001", ASL Publications, Providence.

2003, 'Rational Dynamics and Epistemic Logic in Games', in S. Vannucci, ed., "Logic, Game Theory and Social Choice III", University of Siena, department of political economy, 19-23.

See also the ILLC Preprint server for a bunch of recent papers on the interface of logic and game theory.

5 Mededelingen van de onderzoekscholen

www.win.tue.nl/ipa/

Institute for Programming research and Algorithmics

The research school IPA (Institute for Programming Research and Algorithmics) educates researchers in the field of programming research and algorithmics. This field encompasses the study and development of formalisms, methods and techniques to design, analyse, and construct software systems and components. IPA has three main research areas: Algorithmics & Complexity, Formal Methods, and Software Technology. Researchers from eight universities (University of Nijmegen, Leiden University, Technische Universiteit Eindhoven, University of Twente, Utrecht University, University of Groningen, Vrije Universiteit Amsterdam, and the University of Amsterdam), the CWI and Philips Research (Eindhoven) participate in IPA.

In 1997, IPA was formally accredited by the Royal Dutch Academy of Sciences (KNAW). In 2002, this accreditation was extended for a period of five years.

Activities in 2003

IPA has two multi-day events per year, the Lentedagen and the Herfstdagen, which focus on a particular subject. In the 2002 - 2006 period, each of the Herfstdagen will be dedicated to one of IPA's four so-called application areas: Networked Embedded Systems, Security, Intelligent Algorithms, and Compositional Programming Methods. In 2003, the Herfstdagen were dedicated to Compositional Programming Methods, and the Lentedagen were on Bioinformatics.

On the European front, IPA continued its cooperation in the European Educational Forum (EEF) with the research schools BRICS (Denmark), TUCS (Finland), UKII (United Kingdom), IP (Italy), GEFI (Germany) and FI (France). In the series of four summer schools on the Foundations of Computer Science, organised by the EEF from 2000 - 2003, IPA hosted the final event, which was dedicated to Concurrency.

Lentedagen *April 23 - 25, NH Hotel, Best*
Bioinformatics is an exciting new scientific field that combines molecular biology with methods and approaches from computer science and statistics. The emphasis of the program, composed by Joost Kok (UL) and Peter Hilbers (TU/e), was on new applications and research questions arising in the areas of computer science covered by IPA, but it brought together researchers from all three disciplines.

The Lentedagen took the form of a series of lectures and tutorials, which were clustered into sessions. Each session was dedicated to a different part of the "cycle of life" as it is studied in the life sciences:

DNA

Population **RNA**

Organism **Protein**

Cell

To keep the program self-contained, it opened with a tutorial recalling the key biological notions of the areas covered by the sessions. Abstracts, papers, handouts and presentations, can be found at the web site.
See: `www.win.tue.nl/ipa/archive/springdays2003/`

Herfstdagen *November 17 - 21, 2003, Hotel De Zwaan, Beekbergen*
Compositional Programming Methods is one of the four application areas chosen by IPA as a focus for its research in the period 2002 - 2006. In software engineering the role of software components is becoming a key issue; unfortunately what a component is or should be is not yet well-understood formally. Researchers in IPA aim to make progress in both theory and practice of the construction of software. The leading theme for both sides is compositionality: obtaining larger systems from smaller ones by means of well-understood composition rules.

The Herfstdagen intended to give an overview current research in and around IPA. Various theories of components and composition and theories of component-based design were presented, as well as applications of component models and component-based design. The program contained sessions on: coordination, program algebra, proof systems for object-oriented programming, design patterns, aspect orientation, attribute grammars, run-time composition, generic programming, interface specification, and compositional program restructuring. Shmuel Katz of the Technion (Haifa) was special guest speaker.

The program was composed by Mehmet Aksit (UT), Jan Bergstra (UvA), Marko van Eekelen (KUN), Ruurd Kuiper (TU/e), and Doatse Swierstra (UU). Abstracts, papers, handouts and presentations can be found at the web site: `www.win.tue.nl/ipa/archive/falldays2003`

EEF School on Concurrency *May 19 - 31, Kapellerput, Heeze*
The aim of the school was to provide in-depth knowledge to young scientists on the foundations of Concurrency from a number of approaches. For each approach, training in theoretical foundations was combined with hands-on experience with tools that have been developed within the approach.

The school program contained the following elements: *Process algebra* lecturers: Jos Baeten and Jan Friso Groote (TU/e), tool: μCRL toolset; *Model checking* lecturers: Dennis Dams (Bell Labs) and Dragan Bošnački (TU/e), tool: Spin; *Verification of non-functional properties* lecturers: Kim Larsen and Gerd Behrmann (Aalborg University), tool: Uppaal; *Theorem proving* lecturers: Jozef Hooman and Adriaan de Groot (University of Nijmegen), tool: PVS; *Petri nets* lecturers: Wil van der Aalst and Ella Roubtsova (TU/e), tools: Woflan, Protos, CPN tools. Among the participants were 20 Ph.D. students from 10 different countries. More information on the school, including some pictures, can be found at the school's web page: `www.win.tue.nl/ipa/archive/EEFschool/`.

The companion series of EEF summer schools, the series on Trends in Computer Science, also saw its final event in 2003 with the Trends School on Mobile Computing, hosted by the British research school UKII and held in Edinburgh from July 7 - 12. For more information on EEF and its activities, see: `www.win.tue.nl/EEF/`.

In addition to organising the Lentedagen, the EEF Summerschool, and the Herfstdagen, IPA contributed to the organisation of ICALP2003. This 30th annual meeting of the European Association of Theoretical Computer Science was held at the Technische Universiteit Eindhoven from June 30 - July 4. As usual there were two tracks to the conference: Track A covering Algorithms, Automata, Complexity and Games, and Track B covering Logic, Semantics and Theory of Programming. The proceedings of ICALP2003 have been published by Springer Verlag as LNCS 2719

(Baeten, Lenstra, Parrow, Woeginger (Eds.)). In the weekends surrounding the conference there was an extensive program of pre- and post-conference workshops, several of which were organised by IPA researchers.

In 2003, IPA sponsored two further events: the international symposium on Formal Methods for Components, Objects and their implementation (FMCO 2003, November 4 - 7, Lorentz Center, Leiden), and the 9de Nationale Testdag (November 25, University of Nijmegen).

IPA Ph.D. Defenses in 2003

J.J.D. Aerts *Random Redundant Storage for Video on Demand*
Faculty of Mathematics and Computer Science, TU/e
January 16, IPA-Dissertation Series 2003-01

J.M.W. Visser *Generic Traversal over Typed Source Code Representations*
Faculty of Natural Sciences, Mathematics, and Computer Science, UvA
February 14, IPA-Dissertation Series 2003-03

T.A.C. Willemse *Semantics and Verification in Process Algebras with Data and Timing*
Faculty of Mathematics and Computer Science, TU/e
February 20, IPA-Dissertation Series 2003-05

S.M. Bohte *Spiking Neural Networks*
Faculty of Mathematics and Natural Sciences, UL
March 5, IPA-Dissertation Series 2003-04

M. de Jonge *To Reuse or To Be Reused: Techniques for Component Composition and Construction*
Faculty of Natural Sciences, Mathematics, and Computer Science, UvA
March 6, IPA-Dissertation Series 2003-02

S.V. Nedea *Analysis and Simulations of Catalytic Reactions*
Faculty of Mathematics and Computer Science, TU/e
March 19, IPA-Dissertation Series 2003-06

M.E.M. Lijding *Real-time Scheduling of Tertiary Storage*
Faculty of Electrical Engineering, Mathematics & Computer Science, UT
May 1, IPA-Dissertation Series 2003-07

H.P. Benz *Casual Multimedia Process Annotation – CoMPAs*
Faculty of Electrical Engineering, Mathematics & Computer Science, UT
May 14, IPA-Dissertation Series 2003-08

D.J.P. Leijen *The λ Abroad – A Functional Approach to Software Components*
Faculty of Mathematics and Computer Science, UU
November 4, IPA-Dissertation Series 2003-11

D. Distefano *On Modelchecking the Dynamics of Object-based Software: a Foundational Approach*
Faculty of Electrical Engineering, Mathematics & Computer Science, UT
November 7, IPA-Dissertation Series 2003-09

M.H. ter Beek *Team Automata – A Formal Approach to the Modeling of Collaboration Between System Components*
Faculty of Mathematics and Natural Sciences, UL
December 10, IPA-Dissertation Series 2003-10

Activities in 2004

Although several activities for 2004 are still under development, the themes for the major IPA-events are known: the Lentedagen will be on Hybrid Systems and the Herfstdagen will be dedicated to Intelligent Algorithms.

Lentedagen *April 14-16, 2004, Kapellerput, Heeze.*
More and more systems emerge in which discrete digital controllers control continuous physical processes. Such systems are called "hybrid systems", because they combine discrete and continuous behaviour. Although there are methods for understanding both forms of behaviour separately (continuous behaviour in control science and system theory, and discrete behaviour in computer science), the proper functioning of a hybrid system depends critically on the interaction between the discrete dynamics of the controller and the continuous dynamics of the environment. Hence, the correct and efficient design of such systems requires expertise that is spread over different communities.

Over the past few years, IPA-researchers working in the area of embedded systems have made contact with researchers from communities studying continuous behaviour and started to create a common theoretical basis for the understanding of hybrid systems. The program of the IPA Herfstdagen will present the research currently performed in and around IPA on formalisms, tools, and techniques for the modeling, analysis, validation, and verification of hybrid systems.

More information on the Lentedagen, Herfstdagen, and other upcoming activities will become available through the IPA web pages.

Addresses

Visiting address
Eindhoven University of Technology
Main Building HG 7.22
Den Dolech 2
5612 AZ Eindhoven
The Netherlands

Postal address
IPA, Fac. of Math. and Comp. Sci.
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands

tel. (+31)-40-2474124 (IPA Secretariat)
fax (+31)-40-2475361
e-mail `ipa@tue.nl`
url `www.win.tue.nl/ipa/`

Approximations through relaxations

GERHARD J. WOEGINGER *

Abstract

We discuss polynomial time approximation results for two combinatorial optimization problems (one problem from graph theory, one problem from scheduling theory). The results are based on the technique of rounding the optimal solution of an underlying linear programming relaxation. We analyze these relaxations, their integrality gaps, and the resulting approximation algorithms, and we derive matching worst case instances.

Keywords: Approximation algorithm; worst case analysis; performance guarantee; linear programming relaxation; integrality gap

1 Introduction

Most real-world optimization problems are NP-hard. And most NP-hard problems are difficult to solve to optimality. We conclude: Most real-world optimization problems are difficult to solve to optimality. A standard way of working around this rather pessimistic conclusion is to relax *optimality*, and to satisfy oneself instead with *near-optimal* or *approximate* solutions. This leads us into the area of approximation algorithms for combinatorial optimization problems.

A combinatorial optimization problem consists of a set $\mathcal{I}$ of instances, and a family $\mathcal{F}(I)$ of feasible solutions for every instance $I \in \mathcal{I}$. Every feasible solution $F \in \mathcal{F}(I)$ comes with a non-negative cost $c(F)$. In this paper, we will only consider minimization problems, where the objective is to determine a feasible solution of minimum possible cost. An *approximation algorithm* is an algorithm that for every instance $I \in \mathcal{I}$ returns a near-optimal solution. If it manages to do this in polynomial time, then it is called a *polynomial time* approximation algorithm. An approximation algorithm for a minimization problem is called a *ρ-approximation algorithm*, if it always returns a near-optimal solution with cost at most a factor ρ above the optimal cost. Such a value $\rho \geq 1$ is called a *worst case performance guarantee* of the algorithm.

Approximations through relaxations. One standard approach for designing polynomial time approximation algorithms for a (difficult, NP-hard) optimization problem $\mathcal{P}$ is the following:

(S1) Relax some of the constraints of the hard problem $\mathcal{P}$ to get an easier problem $\mathcal{P}'$ (the so-called *relaxation*).

(S2) Compute in polynomial time an optimal solution S' for this easier relaxed problem $\mathcal{P}'$.

(S3) Translate in polynomial time the solution S' into an approximate solution S for the original problem $\mathcal{P}$.

(S4) Analyze the quality of solution S for $\mathcal{P}$ by comparing its cost to the cost of solution S' for $\mathcal{P}'$.

*gwoegi@win.tue.nl. Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven.

Let C^{Opt} denote the optimal cost of the orginal problem instance, let C^{Rlx} denote the optimal cost of the relaxed instance, and let C^{App} denote the cost of the translated approximate solution. To show that the sketched approach has a performance guarantee of ρ, one usually establishes the following chain of inequalities:

$$C^{Rlx} \leq C^{Opt} \leq C^{App} \leq \rho \cdot C^{Rlx} \leq \rho \cdot C^{Opt}. \tag{1}$$

The first and the last inequality in this chain are trivial, since problem $\mathcal{P}'$ results from problem $\mathcal{P}$ by relaxing constraints. The second inequality is also trivial, since the optimal solution is at least as good as some approximate solution. The third inequality contains the crucial step in the chain, and all the analysis work goes into proving that step. This third inequality relates the relaxed solution to the approximate solution; both solutions are polynomial time computable, and hence their combinatorics will be nice and well-behaved. Thus, the chain yields the desired relation $C^{App} \leq \rho \cdot C^{Opt}$ by analyzing nice, polynomial time computable objects. The analysis avoids touching the orginal NP-hard problem whose combinatorics is messy and complicated and hard to grasp.

Worst case gaps and integrality gaps. Of course, we would like to make the value of the parameter ρ as small as possible: The closer ρ is to 1, the better is the performance of the approximation algorithm. How can we argue that our worst case analysis is complete? How can we argue that we have reached the smallest possible value for ρ? That's usually done by exhibiting a so-called *worst case instance*, that is, an instance I that demonstrates a *worst case gap* of ρ for the approximation algorithm:

$$C_I^{App} = \rho \cdot C_I^{Opt} \qquad \text{and} \qquad C_I^{Opt} = C_I^{Rlx}. \tag{2}$$

Here the left hand equation establishes the gap, and together with the chain (1) it yields the right hand equation. The worst case instance (2) illustrates that our analysis of the *combined* approach (S1)–(S3) is tight. Is this the end of the story? Not necessarily. We could possibly start with the same relaxation step (S1), then solve the relaxation with the same step (S2), and then come up with a completely new (and better) translation step. How can we argue that this is not possible? How can we argue that the value ρ is already the best performance guarantee that we possibly can get out of the considered relaxation? That's usually done by exhibiting an instance J that demonstrates an *integrality gap* of ρ between the original problem and the relaxation:

$$C_J^{Opt} = \rho \cdot C_J^{Rlx} \qquad \text{and} \qquad C_J^{Opt} = C_J^{App}. \tag{3}$$

The equation on the left hand side establishes the gap, and with (1) it yields the equation on the right hand side. In particular, we have $C_J^{App} = \rho \cdot C_J^{Rlx}$. For instance J the third inequality in the chain (1) is tight, and there is no way of proving a better performance guarantee for an approximation algorithm built around the considered relaxation. We stress that such a better approximation algorithm around the relaxation might well exist, but we will never be able to *prove* that its performance guarantee is better than ρ within the framework described above.

For $\rho > 1$, the conditions in (2) and in (3) can not be satisfied by the same instance, as they would be contradictory. Hence, a complete analysis of an approximation algorithm within this framework always must provide *two* separate bad instances, one for the worst case gap and one for the integrality gap.

Overview of this paper. We will illustrate the approach (S1)–(S4) with two examples. The first example (in Section 2) comes from graph theory, and the second example (in Section 3) comes from scheduling theory. For both examples we provide an integer programming formulation, a relaxation, an approximation algorithm, a worst case analysis, and two gap instances. For more information on this field, we refer the reader to the excellent book [6] by Vazirani.

2 The vertex cover problem

As our first example we will discuss the *vertex cover* problem (VC, for short): An input to this problem consists of an undirected graph $G = (V, E)$. A subset $S \subseteq V$ that touches every edge in E is called a *vertex cover* of G, and the goal is to find a vertex cover of minimum cardinality. Problem VC is well-known to be NP-hard (Garey & Johnson [2]).

Exact formulation and relaxation. Consider the following integer programming formulation (4) of VC. Let $v_1, \ldots, v_n$ be an enumeration of the vertices in V. For every vertex v_i, the binary variable x_i decides whether v_i is in the vertex cover (in which case $x_i = 1$) or whether v_i is not in the vertex cover ($x_i = 0$).

$$\begin{array}{lll} \min & \sum_{i=1}^{n} x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for every edge } [v_i, v_j] \in E \\ & x_i \in \{0, 1\} & \text{for } i = 1, \ldots, n \end{array} \tag{4}$$

The constraints state that every edge must be touched by the vertices v_i with $x_i = 1$, that is, by the vertices in the cover. Since VC is an NP-hard problem, also the equivalent integer programming formulation (4) will be NP-hard to solve. Therefore, we relax this formulation to get something simpler: We replace the integrality constraints "$x_i \in \{0, 1\}$" by continuous constraints "$0 \leq x_i \leq 1$":

$$\begin{array}{lll} \min & \sum_{i=1}^{n} x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for every edge } [v_i, v_j] \in E \\ & 0 \leq x_i \leq 1 & \text{for } i = 1, \ldots, n \end{array} \tag{5}$$

Since all variables are now continuous, this relaxation is a standard linear program and can be solved to optimality in polynomial time. We denote the optimal objective value of (4) by C^{Opt}, and the optimal objective value of the LP-relaxation (5) by C^{LP}. Furthermore, we denote by x_i^{LP} the value of variable x_i in the optimal LP-solution.

The approximation algorithm. Now let us translate the LP-solution into a 'nearby' feasible IP-solution. The trouble with the LP-solution is that the variables x_i^{LP} need not be integral, whereas we would like them to be binary. A simple way of resolving this problem is to do threshold rounding: If $x_i^{LP} < 1/2$, then we define a corresponding *rounded* variable $\tilde{x}_i = 0$. And if $x_i^{LP} \geq 1/2$, then we define a corresponding rounded variable $\tilde{x}_i = 1$. The corresponding *rounded* objective value is denoted by $C^{App} = \sum_{i=1}^{n} \tilde{x}_i$.

Let us verify that the rounded values $\tilde{x}_i$ constitute a feasible solution of (4): If they violate some constraint "$\tilde{x}_i + \tilde{x}_j \geq 1$", then $\tilde{x}_i = \tilde{x}_j = 0$ must hold, and thus $x_i^{LP} < 1/2$ and $x_j^{LP} < 1/2$. But this would yield $x_i^{LP} + x_j^{LP} < 1$, and contradict the feasibility of the LP-solution for (5). Hence, the rounded solution indeed is feasible for (4). What about its quality? We observe that the threshold rounding yields

$$\tilde{x}_i \leq 2\, x_i^{LP} \qquad \text{for } i = 1, \ldots, n. \tag{6}$$

By adding up the inequalities (6) for $i = 1, \ldots, n$ we derive that

$$C^{App} = \sum_{i=1}^{n} \tilde{x}_i \leq 2 \sum_{i=1}^{n} x_i = 2\, C^{LP} \leq 2\, C^{Opt}. \tag{7}$$

Hence, the described approximation algorithm has a worst case performance guarantee of at most 2.

Analysis of the two gaps. Is our worst case bound of 2 on the worst case performance guarantee of this algorithm best possible? Yes, it is: Consider the complete bipartite graph where $V = L \cup R$ with $|L| = |R| = n/2$ and $E = \{[u, v] : u \in L, v \in R\}$. Then an optimal vertex cover consists of all vertices in L, and has $C^{Opt} = n/2$. One optimal LP-solution is given by $x_i^{LP} \equiv 1/2$; this leads to $\tilde{x}_i \equiv 1$ and $C^{App} = n$.

Gap 2.1 *There exist instances for VC for which the gap between the optimal objective value and the objective value produced by the rounding algorithm equals* 2.

And what about the integrality gap of the LP-relaxation? Consider the complete graph K_n on n vertices (the graph that contains all possible edges). Then $C^{Opt} = n - 1$, since by omitting two vertices, we would not touch the edge between these two vertices. Morevoer, the LP-relaxation has an optimal solution with $x_i^{LP} \equiv 1/2$ and $C^{LP} = n/2$.

Gap 2.2 *The integrality gap of the linear programming relaxation for problem VC is* 2.

Arora, Bollobás & Lovász [1] show that certain large families of linear programming relaxations for vertex cover all have integrality gaps of at least 2. It is an outstanding open problem to break through this barrier of 2. Håstad [3] has proved that unless P=NP, there cannot be a polynomial time approximation algorithm for vertex cover with performance guarantee strictly less than 7/6.

3 Scheduling with job rejections

In this section, we consider an environment with n jobs $J_1, \ldots, J_n$ and with m *unrelated* parallel machines $M_1, \ldots, M_m$. Job J_j has a processing time p_{ij} on machine M_i, and moreover job J_j has a positive *rejection penalty* f_j. All jobs are available at time 0. Preemption of the jobs is allowed (that is, a job may be arbitrarily interrupted and resumed later on an arbitrary machine). A job may be processed on at most one machine at a time, and every machine may process at most one job at a time. For each job J_j, it must be decided whether to accept or to reject it. The accepted jobs are to be scheduled on the m machines. For the accepted jobs, we pay the makespan of this schedule (that is, the maximum job completion time). For the rejected jobs, we pay their rejection penalties. In other words, the objective value is the preemptive makespan of the accepted jobs plus the total penalty of the rejected jobs. This scheduling problem is denoted by SCHED. Hoogeveen, Skutella & Woeginger [4] have proved that it is NP-hard in the strong sense.

Exact formulation and relaxation. Again, we will start by stating an integer programming formulation. For job J_j, the binary variable y_j decides whether J_j is rejected ($y_j = 0$) or accepted ($y_j = 1$). The continuous variables x_{ij} describe which percentage of job J_j should be processed on machine M_i. The continuous variable C denotes the optimal preemptive makespan for the accepted jobs.

$$
\begin{array}{lll}
\min & C + \sum_{j=1}^{n} (1 - y_j) f_j & \\
\text{s.t.} & \sum_{j=1}^{n} x_{ij} p_{ij} \leq C & \text{for } i = 1, \ldots, m \\
& \sum_{i=1}^{m} x_{ij} p_{ij} \leq C & \text{for } j = 1, \ldots, n \\
& \sum_{i=1}^{m} x_{ij} = y_j & \text{for } j = 1, \ldots, n \\
& x_{ij} \geq 0 & \text{for } i = 1, \ldots, m \text{ and } j = 1, \ldots, n \\
& y_j \in \{0, 1\} & \text{for } j = 1, \ldots, n
\end{array}
\tag{8}
$$

The first family of constraints states that for every machine the total assigned processing time is at most C. The second family of constraints states that the total processing time of any accepted

job cannot exceed C. The third family of constraints connects the binary decision variables y_j to the continuous variables x_{ij}: If a job is accepted ($y_j = 1$), then the percentages x_{ij} should add up to $1 = y_j$. If a job is rejected ($y_j = 0$), then the percentages x_{ij} should add up to $0 = y_j$.

As soon as we have fixed all the values x_{ij}, the remaining makespan minimization problem is essentially makespan minimization in a preemptive open shop. It is well-known [5] that for a preemptive open shop, the smallest value C fulfilling the first and second family of constraints in (8) yields the optimal preemptive makespan. To summarize, the integer program (8) is a complete and correct description of the problem SCHED.

We define the LP-relaxation of the integer programming formulation (8) by replacing the integrality constraints "$y_j \in \{0,1\}$" by continuous constraints "$0 \leq y_j \leq 1$". This LP-relaxation can be solved in polynomial time, and we denote an optimal solution by x_{ij}^{LP}, y_j^{LP}, and C^{LP}.

The approximation algorithm. Now we want to translate the LP-solution into a reasonable feasible solution for the (8). We mainly have to take care of the decision variables y_j; however, this time we also must pay attention to the continuous variables x_{ij}, since their values depend on the values y_j via the third family of constraints in (8).

We *randomly* choose a threshold α from the uniform distribution over the interval $[1/e, 1]$; here as usual $e \approx 2.71828$ denotes the base of the natural logarithm. If $y_j^{LP} \leq \alpha$, then we define a rounded decision variable $\tilde{y}_j := 0$, and otherwise we define $\tilde{y}_j := 1$. Jobs J_j with $\tilde{y}_j = 0$ are rejected in the rounded solution, and we set all their variables $\tilde{x}_{ij} = 0$. Jobs J_j with $\tilde{y}_j = 1$ are accepted in the rounded solution; we set all their variables $\tilde{x}_{ij} := x_{ij}^{LP}/y_j^{LP}$. Finally, we define the rounded makespan by

$$\tilde{C} \;:=\; \max\{\max_{1\leq i\leq m}\sum_{j=1}^{n}\tilde{x}_{ij}p_{ij},\;\; \max_{1\leq j\leq n}\sum_{i=1}^{m}\tilde{x}_{ij}p_{ij}\}. \tag{9}$$

It can be verified that the rounded solution $\tilde{x}_{ij}$, $\tilde{y}_j$, and $\tilde{C}$ constitutes a feasible solution of (8): All variables $\tilde{y}_j$ are binary. For j with $\tilde{y}_j = 0$, the variables $\tilde{x}_{ij}$ add up to 0. For j with $\tilde{y}_j = 1$, the variables $\tilde{x}_{ij}$ add up to $\sum_i x_{ij}^{LP}/y_j^{LP} = 1$. Finally, in (9) the value of $\tilde{C}$ is fixed in order to fulfill the first and the second family of constraints.

Now let us analyze the quality of this rounded solution. For any fixed value of α, the rounded variable $\tilde{x}_{ij}$ is at most a factor of $1/\alpha$ above x_{ij}^{LP}. Hence, by linearity also $\tilde{C}$ is at most a factor of $1/\alpha$ above C^{LP}. Then the expected multiplicative increase in the makespan is at most a factor of

$$\frac{e}{e-1}\int_{1/e}^{1} 1/\alpha \; d\alpha \;=\; \frac{e}{e-1}\;.$$

In the LP-solution, the contribution of job J_j to the total rejection penalty is $(1 - y_j^{LP})f_j$. The expected contribution of J_j to the rejection penalty in the rounded solution is

$$\begin{aligned} f_j \cdot \mathrm{Prob}[y_j^{LP} \leq \alpha] \;&=\; f_j \int_{\max\{1/e,\, y_j^{LP}\}}^{1} \frac{e}{e-1} d\alpha \\ &\leq\; f_j \int_{y_j^{LP}}^{1} \frac{e}{e-1} d\alpha \\ &=\; \frac{e}{e-1}\cdot(1-y_j^{LP})f_j. \end{aligned}$$

All in all, the expected objective value for the rounded solution is at most a factor of $e/(e-1) \approx 1.58$ above the optimal objective value of the LP-relaxation. Hence, our procedure yields a *randomized* polynomial time approximation algorithm with a worst case performance guarantee of $e/(e-1)$.

How can we turn this randomized algorithm into a deterministic algorithm? Well, the only critical values for the threshold parameter α are the values y_j^{LP} with $j = 1, \ldots, n$. All other values of α will yield the same solution as for one of these criticial values. Hence, it is straightforward to derandomize the algorithm in polynomial time: We compute the n rounded solutions that correspond to these n critical values, and we select the solution with smallest objective value.

Analysis of the two gaps. Our next goal is to give a worst case instance for the above approximation algorithm for SCHED. The instance is based on an integer parameter q. There are $m = (q+1)^q - q^q$ machines M_j that are indexed by $j = q^q + 1, \ldots, (q+1)^q$. For every machine M_j, there are two corresponding jobs J_j and J'_j that have infinite processing requirements on all other machines M_i with $i \neq j$; this implies that these two jobs either have to be rejected or have to be processed on M_j. The processing time of job J_j on M_j is $j - q^q$, and its rejection penalty is $(j - q^q)/j$. The processing time of job J'_j on M_j is q^q, and its rejection penalty is q^q/j. Note that the overall processing time of J_j and J'_j is j, and that their overall penalty is 1.

One possible feasible solution accepts all the jobs J'_j and rejects all the jobs J_j. The resulting makespan is $C = q^q$, and the resulting objective value equals

$$q^q + \sum_{j=q^q+1}^{(q+1)^q} (j - q^q)\frac{1}{j} \;=\; (q+1)^q - q^q \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} \;. \tag{10}$$

It can be verified that this in fact is the optimal objective value. Next, assume that the approximation algorithm starts from the following feasible solution for the LP-relaxation: For $j = q^q + 1, \ldots, (q+1)^q$ the two jobs J_j and J'_j both get an acceptance value $y_j^{LP} = q^q/j$. Then on every machine M_j, the overall accepted processing time is $(j - q^q)y_j^{LP} + q^q\, y_j^{LP} = q^q$. The penalty of job J_j plus the penalty of job J'_j equals $1 - y_j^{LP} = (j - q^q)/j$. Hence, the objective value of this LP-solution equals the value in (10).

Consider the rounding step for some fixed threshold α. Since the values $y_j^{LP} = q^q/j$ are decreasing in j, there exists some index k such that for $j \leq k$ the values $y_j^{LP} = q^q/j$ all are rounded up to 1 (and the corresponding jobs are accepted), whereas for $j \geq k+1$ the values $y_j^{LP} = q^q/j$ all are rounded down to 0 (and the corresponding jobs are rejected). Then the makespan becomes k (the load on machine M_k), the total rejection penalty is $(q+1)^q - k$, and the objective value is $(q+1)^q$. Thus, the objective value in the rounded solution always equals $(q+1)^q$, and does not depend on α or k.

The ratio between the optimal objective value in (10) and the approximate objective value of $(q+1)^q$ equals

$$1 - \left(\frac{q}{q+1}\right)^q \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} \;. \tag{11}$$

We estimate the sum in (11) by

$$\int_{q^q+1}^{(q+1)^q+1} \frac{1}{z}\, dz \;\leq\; \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} \;\leq\; \int_{q^q}^{(q+1)^q} \frac{1}{z}\, dz \;.$$

Since the integrals on the left and on the right hand side both converge to 1 when q goes to infinity, the same holds true for the sum in between. Hence, the ratio in (11) behaves roughly like $((q+1)^q - q^q)/(q+1)^q$. For large q, this ratio tends to $(e-1)/e$.

Gap 3.1 *There exist instances of SCHED for which the gap between the optimal makespan and the makespan produced by the rounding algorithm comes arbitrarily close to* $e/(e-1)$.

Our final goal in this section is to get the matching lower bound of $e/(e-1)$ for the integrality gap of the LP-relaxation for SCHED. We use a slight modification of the instance constructed above. Again, we use an integer parameter q, and again there are $m = (q+1)^q - q^q$ machines M_j that are indexed by $j = q^q + 1, \ldots, (q+1)^q$. For every machine M_j, there is one corresponding job J_j. The processing requirements of J_j are $p_{jj} = j$, and $p_{ij} = \infty$ for $i \neq j$. The rejection penalty is uniformly $f_j \equiv 1$.

Consider a 'reasonable' feasible schedule with makespan T: Then the jobs on machines M_j with $j \leq T$ will be accepted, and the jobs on machines M_j with $j > T$ will be rejected. This yields a total rejection penalty of $(q+1)^q - T$, and an optimal objective value of $(q+1)^q$. Next, consider

the following feasible solution for the LP-relaxation: We set $y_j^{LP} = q^q/j$ for $j = q^q+1, \ldots, (q+1)^q$, and we set $C^{LP} = q^q$. The objective value of this solution is equal to

$$q^q + \sum_{j=q^q+1}^{(q+1)^q} (1 - y_j^{LP}) \quad = \quad (q+1)^q - q^q \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} \ .$$

Since this LP-value is equal to the value in (10), and since the optimal value is equal to $(q+1)^q$, the ratio of these two values equals the ratio in (11). The arguments around (11) show that as q becomes large, this ratio tends to $(e-1)/e$.

Gap 3.2 *For problem SCHED, the integrality gap of the LP-relaxation is* $e/(e-1)$.

It would be nice to get a polynomial time approximation algorithm for SCHED with a worst case performance guarantee better than $e/(e-1)$.

References

[1] S. Arora, B. Bollobás, and L. Lovász (2002). Proving integrality gaps without knowing the linear program. *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'2002)*, 313–322.

[2] M.R. Garey and D.S. Johnson (1979). *Computers and Intractability.* W.H. Freeman and Co., New York.

[3] J. Håstad (1999). Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica 182*, 105–142.

[4] H. Hoogeveen, M. Skutella, and G.J. Woeginger (2003). Preemptive scheduling with rejection. *Mathematical Programming 94*, 361–374.

[5] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (1993). Sequencing and scheduling: Algorithms and complexity. In: S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin (eds.) *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 445 522.

[6] V.V. Vazirani (2001). *Approximation Algorithms.* Springer-Verlag, Berlin, Heidelberg.

Domain Independent Hierarchical Clustering

Rudi Cilibrasi, Centrum voor Wiskunde en Informatica

Artificial intelligence has taken hold as a persistent holy grail in computing. Although conversational robots have been so far constrained to the realms of science fiction, there have been incremental steps towards true machine intelligence over the past twenty years. Like space exploration, the dream of artificial intelligence holds a special place in many minds as one of the most inspirational branches of computer science. Science cannot yet take people to vacations on the moon, but just this month we have enjoyed high resolution photographs of Mars sent from our robotic tourist rover. In this article, I will invite you to read about another small step on the road towards practical machine learning.

In the last two years, Ming Li and Paul Vitányi (and others in their groups) devised a formula involving two objects or files. These objects can be any type of information, in the same way a computer's hard disk drive can hold any sort of data. By using a powerful and general mathematical notion termed Kolmogorov Complexity, Li and Vitányi have managed to create a normalized information distance measure. In plain terms, Kolmogorov Complexity can be thought to be the size of a file when compressed by the best conceivable data compression program. Of course, we cannot ever actually know the exact size of any particular file under such an idealized compressor, because we cannot yet (nor ever) build a "perfect compressor." Just the same, it is possible to explore the underlying mathematical structure of such a compressor, and doing so has led to some surprisingly practical results. Li and Vitányi have collected a wealth of information on this theory in their excellent book, *An Introduction to Kolmogorov Complexity and Its Applications.* [1]

A normalized distance is a number between 0 and 1 that represents some idea of similarity between two objects. In the case of a normalized information distance, the objects we are talking about are files stored on a computer. A distance of 0 means two files are identical, whereas a distance of 1 means that two files are totally unrelated to each other. There are many different ways to analyze files. There are many different characteristics that could be compared, each leading to a different notion of distance. A recent theoretical breakthrough was achieved when a formula was discovered that could be said to be a sort of combination of all possible distance metrics. Just as the mathematical ideal of Kolmogorov Complexity can be said to be the size when the ultimate compression program is applied, so such values can be arranged in a simple quotient formula to arrive at the most generally informative normalized information distance in a certain mathematical sense. [2, 3] At first, this formula was of primarily theoretical interest. The formula attained practical relevance when Li realized that the formula may happen to work using real data compression programs in this setting. A formal justification for this step was only discovered later in [5]. Just last year it has received renewed attention with the help of the author. Cilibrasi streamlined the software and made a modified tree search algorithm and packaged the system as a convenient open-source offering. We have created a fast, easy-to-use software package that opens the door to further exploration. The software uses generic compressors instead of true Kolmogorov Complexity to carry through the mathematical formulas to arrive at some real number between 0 and 1 that relates any two files. Several months later this idea was implemented at CWI (Centrum voor Wiskunde en Informatica) using clusters of workstations in parallel, and the results have been consistently intriguing. We have applied this technique in areas as diverse as medicine, astronomy, biology, and network packet analysis.

Intelligence has historically been a characteristic proudly flaunted as synonymous with the human spirit. Yet in the last 30 years we have seen brief flashes of synthetic intelligence in unexpected areas. A great example is the game of chess. Fifteen years ago, chess world champion Garry Kasparov claimed that he would never be beaten by a computer. Just a few years later, it happened. Then again and again. By now, this has happened many times, and many chess fans believe it is inevitable that one day no human will be able to beat the best computer opponent. While these milestones are encouraging to computer scientists around the world, they do little towards helping us build conversational robots. This is because every major

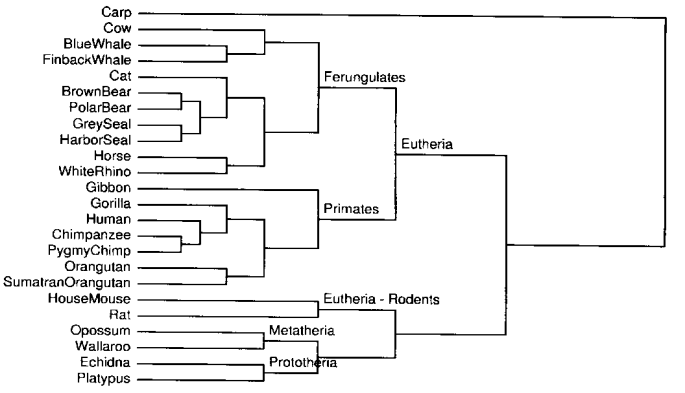
advance in artificial intelligence so far has been limited to a very specific domain. We can make a robot play Connect Four perfectly, or focus a camera, or calculate airflows around a wing. But we have no computer that can yet answer the spoken question, "How many words does this sentence have?" without a preprogrammed cheat-sheet of canned responses. The problem is one of contextual domains: When a computer plays strategy games, the types of questions it must consider are limited to a very small part of the universe. In these thimble-sized worlds, computers reign supreme. But when the context expands out into the fuzzier domains of human knowledge, communication, and creative problem-solving, computer scientists have been unable to mount a decent attempt at giving a computer "common sense." It seems intrinsic to human natural language to have a wide range of possible contexts, and this has stymied the ability of people to use computers effectively forever.

Our new technique of calculating the so-called *Normalized Compression Distance*, or *NCD*, between files is a small step in the direction of reasoning within larger contexts than traditional machine learning methodologies. In essence, the method builds upon the genericity of programs like WinZip and other general purpose data compressors. When you compress a file with WinZip, it is not necessary to tell it anything about the meaning of the file; the only instruction it needs from the user is which file to compress. Rather, it analyses the file and looks for patterns of repetition, blithely unaware of any semantic significance the data may have. The remarkable reality is that this simple-minded sort of counting of repetitions is enough to reach seemingly complex conclusions without preprogrammed domain-specific knowledge. The formula used is quite easy: For any two files, A and B, call the compressed size of each of them C_A and C_B. Then combine the two files through concatenation, and call the size of this when compressed C_{AB}. Then we define $NCD(a,b) = (C_{AB} - min(C_A, C_B))/max(C_A, C_B)$. By using this simple formula for every pair of files in a group, we are able to make a matrix of distances that represent how similar or different any two files are.

One of our favorite examples involves animals. Here, we took the mitochondrial genetic sequences of 24 different animals. Next, we used a compressor called PPMZ and calculated the NCD to get the following matrix of distances:

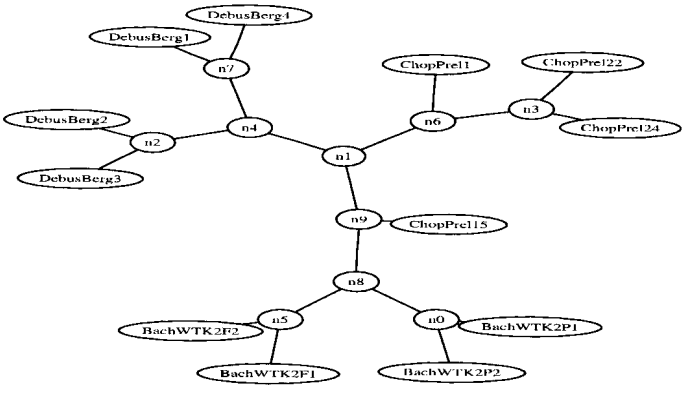
	BlueWhale	Cat	Echidna	Gorilla	Horse	Opossum	PolarBear	SumOrang
	BrownBear	Chimpanzee	FinWhale	GreySeal	HouseMouse	Orangutan	PygmyChimp	Wallaroo
	Carp	Cow	Gibbon	HarborSeal	Human	Platypus	Rat	WhiteRhino
BlueWhale	0.005	0.906	0.943	0.897	0.925	0.883	0.936	0.616	0.928	0.931	0.901	0.898	0.896	0.926	0.920	0.936	0.928	0.929	0.907	0.930	0.927	0.929	0.925	0.902
BrownBear	0.906	0.002	0.943	0.887	0.935	0.906	0.944	0.915	0.939	0.940	0.875	0.872	0.910	0.934	0.930	0.936	0.938	0.937	0.269	0.940	0.935	0.936	0.923	0.915
Carp	0.943	0.943	0.006	0.946	0.954	0.947	0.955	0.952	0.951	0.957	0.949	0.950	0.952	0.956	0.946	0.956	0.953	0.954	0.945	0.960	0.950	0.953	0.942	0.960
Cat	0.897	0.887	0.946	0.003	0.926	0.897	0.942	0.905	0.928	0.931	0.870	0.872	0.885	0.919	0.922	0.933	0.932	0.931	0.885	0.929	0.920	0.934	0.919	0.897
Chimpanzee	0.925	0.935	0.954	0.926	0.006	0.926	0.948	0.926	0.849	0.731	0.925	0.922	0.921	0.943	0.667	0.943	0.841	0.946	0.931	0.441	0.933	0.835	0.934	0.930
Cow	0.883	0.906	0.947	0.897	0.926	0.006	0.936	0.886	0.931	0.927	0.890	0.888	0.893	0.925	0.920	0.933	0.930	0.929	0.905	0.931	0.921	0.930	0.923	0.899
Echidna	0.936	0.944	0.955	0.942	0.948	0.936	0.005	0.936	0.947	0.947	0.940	0.937	0.942	0.941	0.939	0.936	0.947	0.855	0.935	0.949	0.941	0.947	0.929	0.948
FinbackWhale	0.616	0.915	0.952	0.905	0.926	0.885	0.936	0.005	0.930	0.931	0.911	0.908	0.901	0.933	0.922	0.936	0.933	0.934	0.910	0.932	0.928	0.932	0.927	0.902
Gibbon	0.928	0.939	0.951	0.928	0.849	0.931	0.947	0.930	0.005	0.859	0.932	0.930	0.927	0.948	0.844	0.951	0.872	0.952	0.936	0.854	0.939	0.868	0.933	0.929
Gorilla	0.931	0.940	0.957	0.931	0.731	0.927	0.947	0.931	0.859	0.006	0.927	0.929	0.924	0.944	0.737	0.944	0.835	0.943	0.928	0.732	0.938	0.836	0.934	0.929
GreySeal	0.901	0.875	0.949	0.870	0.925	0.890	0.940	0.911	0.932	0.927	0.003	0.399	0.888	0.924	0.922	0.933	0.931	0.936	0.863	0.929	0.922	0.930	0.920	0.898
HarborSeal	0.898	0.872	0.950	0.872	0.922	0.888	0.937	0.908	0.930	0.929	0.399	0.004	0.888	0.922	0.922	0.933	0.932	0.937	0.860	0.930	0.922	0.928	0.919	0.900
Horse	0.896	0.910	0.952	0.885	0.921	0.893	0.942	0.901	0.927	0.924	0.888	0.888	0.003	0.928	0.913	0.937	0.923	0.936	0.903	0.923	0.912	0.924	0.924	0.848
HouseMouse	0.926	0.934	0.956	0.919	0.943	0.925	0.941	0.933	0.948	0.944	0.924	0.922	0.928	0.006	0.932	0.923	0.944	0.930	0.924	0.942	0.860	0.945	0.921	0.928
Human	0.920	0.930	0.946	0.922	0.667	0.920	0.939	0.922	0.844	0.737	0.922	0.922	0.913	0.932	0.005	0.949	0.834	0.949	0.931	0.681	0.938	0.826	0.934	0.929
Opossum	0.936	0.936	0.956	0.933	0.943	0.931	0.936	0.936	0.951	0.944	0.933	0.933	0.937	0.923	0.949	0.006	0.960	0.938	0.939	0.954	0.941	0.960	0.891	0.952
Orangutan	0.928	0.938	0.953	0.932	0.841	0.930	0.947	0.933	0.872	0.835	0.931	0.932	0.923	0.944	0.834	0.960	0.006	0.954	0.933	0.843	0.943	0.885	0.945	0.934
Platypus	0.929	0.937	0.954	0.932	0.946	0.929	0.865	0.934	0.952	0.943	0.936	0.937	0.936	0.930	0.949	0.938	0.954	0.003	0.932	0.948	0.937	0.949	0.920	0.948
PolarBear	0.907	0.269	0.945	0.885	0.931	0.905	0.935	0.910	0.936	0.928	0.863	0.860	0.903	0.924	0.931	0.939	0.933	0.932	0.002	0.942	0.940	0.936	0.927	0.917
PygmyChimp	0.930	0.940	0.960	0.929	0.441	0.931	0.949	0.932	0.854	0.732	0.929	0.930	0.923	0.942	0.681	0.954	0.843	0.948	0.942	0.007	0.935	0.838	0.931	0.929
Rat	0.927	0.935	0.950	0.920	0.933	0.921	0.941	0.928	0.939	0.938	0.922	0.922	0.912	0.860	0.938	0.941	0.943	0.937	0.940	0.935	0.006	0.939	0.922	0.922
SumOrangutan	0.929	0.936	0.953	0.934	0.835	0.930	0.947	0.932	0.868	0.836	0.930	0.928	0.924	0.945	0.826	0.960	0.585	0.949	0.936	0.838	0.939	0.007	0.942	0.937
Wallaroo	0.925	0.923	0.942	0.919	0.934	0.923	0.929	0.927	0.933	0.934	0.920	0.919	0.924	0.921	0.934	0.891	0.945	0.920	0.927	0.931	0.922	0.942	0.005	0.935
WhiteRhino	0.902	0.915	0.960	0.897	0.930	0.899	0.948	0.902	0.929	0.929	0.898	0.900	0.848	0.928	0.929	0.952	0.934	0.948	0.917	0.929	0.922	0.937	0.935	0.032

As you can see, this matrix contains quite a lot of common knowledge about animals. For instance, the distance between a Brown Bear and a Polar Bear is 0.269. This number is substantially lower than average, and suggests that these two different types of bears are in some way highly similar. If we apply a simple tree search to look for the best unrooted binary tree that matches this distance matrix, more relevant correspondences are displayed:

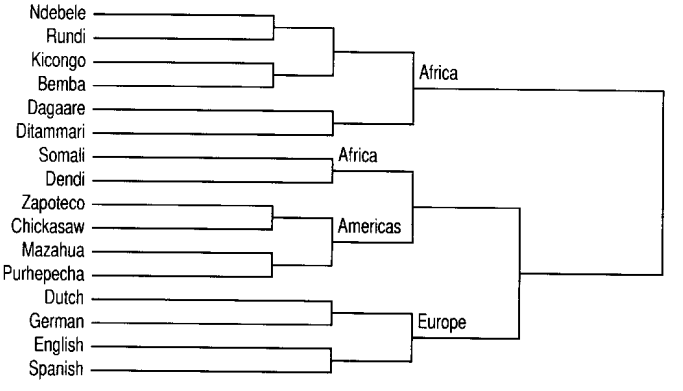


The evolutionary tree shown here is not in itself a new theory. The novel part is that the program that generated this tree was told nothing about biology, evolution, DNA, or any sort of theory; we just fed in the sequence data, asked for a tree, and the computer managed to do the *right thing* without being told explicitly what the question was. This sort of open-ended fuzzy and generic reasoning make this approach unique.

In another experiment, we took twelve MIDI music samples from Debussy, Bach, and Chopin. Here is the result:



Finally, an example using written language samples from the United Nations Declaration of Human Rights:



In our research, we have already applied this method to a wide variety of fields such as astronomy, handwritten digit recognition, heart rhythm identification, music classification, authorship and translation inference, and network packet classification for intrusion detection systems. It's clear to us that we've only begun to scratch the surface of possible applications of this technique, and so now are focussing

our research on trying to understand how this technique can be applied to real and relevant problems in automated deduction and data mining.

If you are interested in finding out more, you can visit

`http://homepages.cwi.nl/~cilibrar/`

to read the first two papers in this line of research, and you may download the software (available as open source for Linux) from

`http://complearn.sourceforge.net/`

Further reading:

References

[1] M. Li and P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 2nd Edition, 1997.

[2] C.H. Bennett, P. Gács, M. Li, P.M.B. Vitányi, and W. Zurek. Information Distance, *IEEE Transactions on Information Theory*, 44:4(1998), 1407–1423.

[3] M. Li, X. Chen, X. Li, B. Ma, P. Vitányi. The similarity metric, *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[4] R. Cilibrasi, R. de Wolf, P. Vitanyi. *Algorithmic Clustering of Music,*

`http://arxiv.org/PS_cache/cs/pdf/0303/0303025.pdf`

[5] R. Cilibrasi, P. Vitanyi. *Clustering by Compression,*

`http://arxiv.org/PS_cache/cs/pdf/0312/0312044.pdf`

Improving the Quality of Protocol Standards: Correcting IEEE 1394.1 FireWire Net Update*

Judi Romijn
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven, The Netherlands
email: j.m.t.romijn@tue.nl

Abstract

The new IEEE 1394.1 FireWire draft standard, which is expected to be finalised this year, contains a new protocol for constructing and maintaining spanning trees in the network topology, called net update. This protocol is complex and merits formal specification and analysis. In the scope of the NWO Vernieuwingsimpuls Project 'Improving the Quality of Protocol Standards', we have taken part in the standardisation process, and have helped the development of this protocol through Promela prototyping (Spin simulation and model checking), PVS protocol derivation and manual proof. Our efforts have resulted in the discovery and correction of many errors, omissions and inconsistencies, as well as the addition of the correctness properties of the protocol to the standard description.

1 Introduction

The NWO Vernieuwingsimpuls 'Improving the Quality of Protocol Standards' which started in December 2001, has taken on its first case study in the net update protocol of the new IEEE FireWire 1394.1 standard. The standard enables building larger networks of IEEE 1394 buses by connecting these with bridges, and the net update protocol maintains spanning trees in these networks. Because of the dynamic 'plug-an-play' character of 1394.1 networks, the behaviour of this protocol is so complex that it is almost impossible to grasp just from reading the description in the standard, let alone understand if and why it is correct.

The members of the NWO project (Goga, Mooij, Romijn, and Wesselink) together with van Langevelde (CWI, Amsterdam) have subjected net update to formal specification and analysis, and have discovered many omissions, inconsistencies and errors. We have participated in the standardisation effort by taking part in the IEEE 1394.1 Ballot Response Committee, and thus were in the position to influence the standard and the development of net update itself. This has resulted in many adjustments of the protocol, the construction of a new protocol, and the addition of the correctness properties in an annex of the standard. The draft standard is being finalised at the time of writing to be sent to the IEEE recirculation ballot for the second time.

In this paper, Section 2 introduces FireWire networks and explains the purpose and nature of net update. Section 3 lists the results of our research: papers, changes in the standard, the new protocol, and the correctness properties.

2 IEEE FireWire and Net Update

The 1394.1 functionality The IEEE 1394 standard [3, 4], specifies how devices, identified by their 64-bit hardware addresses, can be interconnected with cables, without users having to worry about configuring these devices. The IEEE abstraction as presented in the standard consists of *nodes*, identified by 6-bit *physical ids*, connected with a *serial bus*. This abstraction is illustrated in Figure 1.

*This research is supported by the Netherlands Organisation for Scientific Research NWO under project number 016.023.015 "Improving the Quality of Protocol Standards", URL: http://www.win.tue.nl/oas/en_index.html.

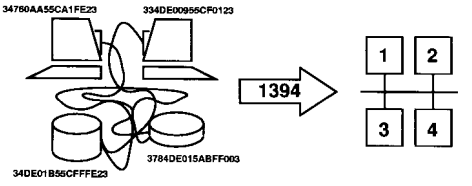


Figure 1: An example IEEE 1394 serial bus

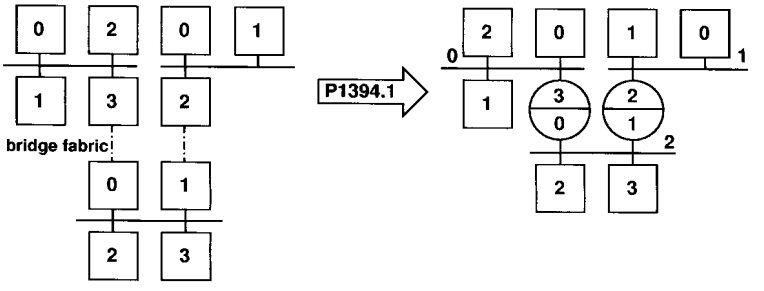


Figure 2: An example IEEE 1394.1 net

In order to meet its high-performance needs, IEEE 1394 restricts serial buses to a maximum of 63 nodes attached, with a maximal cable length of 4.5 meters between each pair of nodes. Larger networks may be built by connecting serial buses with *bridges* which selectively route network traffic originating from a node on one bus destined to a node on another bus.

The IEEE 1394.1 standard [5] specifies how buses can be interconnected with bridges, consisting of a pair of directly coupled nodes called *bridge portals*, without users having to worry about configuring any of these buses. The two portals which constitute one bridge, called each other's *co-portal*, communicate via the *bridge fabric*, a medium which is beyond the 1394 protocol. It is important to note that as a consequence, each of the two co-portals resides on its own separate bus. The abstraction as presented in the standard consists of a *net* of serial buses, where each bus is identified by a 10-bit *bus id*. This abstraction is illustrated in Figure 2. The main restriction on IEEE 1394.1 nets is the maximum of 1023 buses attached.

Spanning trees In order to obtain and maintain the bus id numbering, and use this information for routing messages between different buses, yet another abstraction is necessary: viewing the net as a graph with bridges for edges, and buses for nodes and having spanning trees in this graph structure. The graph abstraction is illustrated in Figure 3. Here, the numbers in the nodes of the graph are the bus ids.

The first notion of spanning tree is the *bus tree*. For each bus id in use, the corresponding bus tree enables the routing of messages destined for that bus id. An example bus tree is shown in Figure 4(a). Here, the dashed edge indicates a bridge that is not part of the spanning tree. Clearly, there must be as

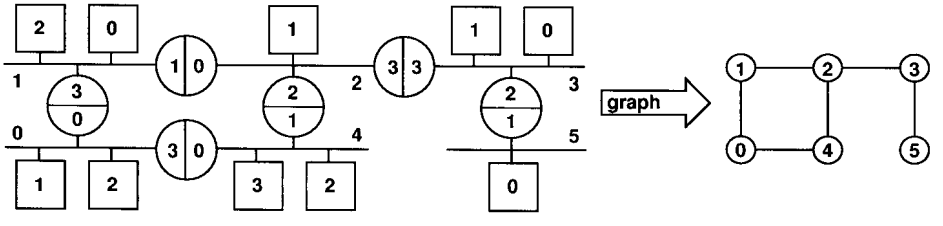
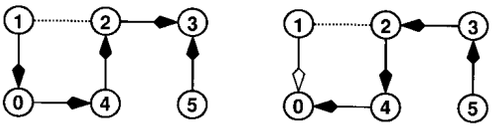


Figure 3: The bus-based graph abstraction



(a) The bus tree for bus id 3

(b) A net tree

Figure 4: Examples of spanning trees for the net in Figure 3

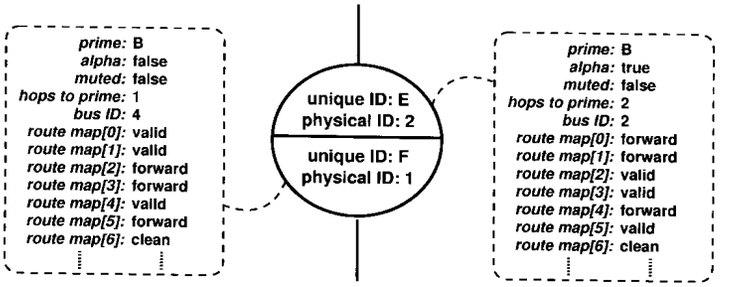


Figure 5: Net update information per bridge portal

many bus trees as the number of bus ids in use.

The second notion of spanning tree is the *net tree*. There is one net tree that gives each bus a route to the root bus where the *prime portal* resides. The prime portal is in charge of assigning bus ids and its unique 64-bit id is the identity of the net. An example net tree is shown in Figure 4(b). Here, the white arrow head indicates the prime portal.

How? The implementation of spanning trees is obtained as follows. Each bridge portal has information that corresponds with that of its co-portal, and that initially spans a net tree consisting of just the edge represented by that bridge, and no bus trees. Upon each topology change on a bus, first an algorithm called *net update* is executed that adjusts the information of all bridge portals attached to the bus. It ensures the maintenance and (if necessary) construction of the net tree as well as the maintenance and (if necessary) desctruction of the bus trees. Whenever the information of one portal in the bridge is changed, this is done in synchronisation with the co-portal, so the information of two portals in one bridge is always consistent. Each adjusted bridge portal ensures that net update is executed on the co-portal's bus as well, which by repetition ensures that all bridges in the net are updated. Upon completion of net update on a bus, if the bus has no bus id and hence no bus tree exists for it in the net, an algorithm called *bus enumeration* is executed to obtain a new bus id and span the corresponding bus tree. The information employed by each bridge portal for the net tree and the bus trees is the following:

prime: the unique identity of the prime portal (64 bits)
alpha: a direction flag for the net tree (1 bit)
mute: a flag indicating if the bridge can be used (1 bit)
hops to prime: the distance to the root bus of the net tree (10 bits)
bus id: the identity of the local bus (10 bits)
route map: for each possible bus id, a four-valued direction variable (clean/valid/forward/dirty, 2046 bits in total)

The net update information is depicted in Figure 5 for the bridge that connects buses 2 and 4 in Figures 3 and 4.

The net update task of collecting information of all bridge portals on the bus, determining which trees are to be destroyed, adjusted, spanned and left alone, and informing each bridge portal of the result, is

performed by one designated portal, the *coordinator*.

Graph view vs. implementation The relation between implementation variables and the graph abstraction is given by the following relations:

- A bridge portal is the prime portal iff *prime* is equal to its unique 64-bit identity and *alpha* is true.
- A bridge represents a directed edge in the net tree iff for both portals, *mute* is false, and for one of the portals *alpha* is false or it is the prime portal, and for the other portal *alpha* is true and it is not the prime portal. The directed bridge is an outgoing edge on the bus where the adjacent portal's *alpha* is true and it is not the prime portal, and it is an incoming edge on the other bus.
- A bridge represents an undirected edge in the net tree iff for both portals, *mute* is true and each portal's *alpha* is false.
- A bridge represents a directed edge in the bus tree for bus id b iff it is a directed edge in the net tree, and *route map*[b] is forward for one portal, and valid for the other portal. The directed bridge is an outgoing edge on the bus with the forward entry, and it is an incoming edge on the other bus.
- A bridge represents an undirected edge in the bus tree for bus id b iff it is a undirected edge in the net tree, and *route map*[b] is valid for both portals.
- A bridge does not represent an edge in the bus tree for bus id b iff for both portals, *route map*[b] is clean or dirty. The value dirty represents a bus id for which the bus tree is being destroyed, it is a temporary value that only occurs during net update.

Any bridge state that is not covered by the above possibilities is an inconsistent state and cannot occur in the net update protocol, except the 'half-directed' edge in a bus tree, where *route map*[b] is valid for one portal and dirty for its co-portal. This is treated as an incoming edge as well on the bus with the valid entry.

Net update algorithm Net update (re)starts executing on a bus whenever the local bus topology has changed (removal or insertion of bridge portals), or when one of the bridge portal's information was adjusted because of net update executing on an adjacent bus. It consists of the following consecutive operations, presented in the graph terminology for sake of clarity and compactness. We assume extra variables per bridge for the net identity, for the distance to the root bus in the net tree, and a destroy flag for each possible bus id.

1. *loop elimination*: if the information of the bridge portals indicates a loop in the topology, then one edge must be taken out of the net tree and bus trees. One of the following applies:
   loop criterion 1 For one net tree identity, there are two outgoing edges on the bus. The one with the largest distance to the root is selected.
   loop criterion 2 For one net tree identity, there is one outgoing edge and one edge with the prime portal on the bus. The outgoing edge is selected.
   action The bridge with the selected edge is undirected in both the net tree and all bus trees.
2. *spanning & destroying trees*: each bridge attached to the bus is adjusted so that a selected net tree is spanned, and each bus tree is either destroyed (if in conflict), spanned (if no conflict but not yet complete) or left unchanged (otherwise).
   net tree criterion The maximal net tree identity for which an outgoing edge or the prime portal is present on the bus is selected. If there is no outgoing edge or prime portal, the unique id of one of the bridge portals is selected at random.
   distance criterion If the selected tree identity is of an outgoing edge, the distance of that bridge is selected and incremented with 1. Otherwise, the selected distance is 0.
   local bus tree criterion One of the bus trees for which there are only incoming edges and undirected edges is selected at random.
   bus tree criterion Each bus tree for which one of the following holds must be destroyed: (1) there are two outgoing edges, (2) there is no outgoing edge and it is not the selected local bus tree, or (3) the destroy flag is true for that bus id at a bridge portal. Each bus tree that is not destroyed for which there is an outgoing edge must be extended.
   net tree action For each bridge portal, the selected net tree identity is copied. If this means a change, then the bridge is directed in the net tree towards the bus. If the bridge is incoming or undirected, it copies the selected distance.

bus tree action For each bridge portal, and each bus tree, if it must be destroyed, the bridge is undirected in the bus tree and the corresponding destroy flag is set to true. If it must be extended and the bridge is undirected in the bus tree, then the bridge is directed in the bus tree towards the bus. If it is the selected local bus tree, then the bridge is directed in the bus tree towards the bus.

spreading net update For each bridge portal that has been adjusted and that is directed in the net tree, net update is started on the neighbour bus.

3. *completion* For each bridge not adjacent to a bus where net update is busy, all destroy flags are set to false.

3 Results

3.1 Papers

In [6], van Langevelde, Romijn and Goga relate experiences with the tool Spin on Promela specifications of parts of net update for maintaining the net tree and the bus trees, respectively. The most important result is the discovery of non-termination of net update and a first attempt at the net panic procedure. The error was detected with Spin simulation in the original net update procedure, but not in subsequent fixes although these contained the same erroneous behaviour.

In [8], Mooij and Wesselink use formal derivation methods [1] to construct an abstract version of the net tree part of net update. This is based on the correctness properties that must be obtained. Currently, the proofs are being checked in PVS, while a front end tool for such proofs to PVS is being developed.

In [7], Mooij, Goga and Wesselink construct an alternative protocol for the net tree part of net update. The alternative is a mixture of 1394.1 and IEEE 802.1 (as described in [9, 10]) functionality. The correctness is proved with techniques from [1].

In [2], Goga and Romijn investigate the feasibility of adjusting Promela specifications in order to restrict random simulations to interesting behaviour. A theoretic approach is given, and it is shown that given some sufficient conditions, the adjusted Promela code does not exhibit new behaviour nor new deadlocks. Using this approach, the non-terminating behaviour can finally be shown with Spin simulation in all erroneous versions of net update.

A paper with manual proofs for the correctness of the net update and bus enumeration protocol at the implementation level is in preparation. Partial correctness has already been shown, termination still needs to be proved.

3.2 Changes in the IEEE standard

The involvement in the standardisation and our simulations, model checking attempts, protocol derivation and manual proofs have led to a number of smaller and larger errors, omissions and inconsistencies to be found and corrected. We mention only the changes in the IEEE standard with a significant impact on the nature and correctness of the protocol.

Synchronised bridges The two portals in one bridge are now required to only change their information in synchronisation with each other. This greatly simplifies the complexity and description of net update.

Unmuting The draft standard did not mention when a bridge without direction must be directed again, this has been included and adjusted many times. The final version has been adjusted so that a better prime portal is kept in more situations.

Portal's tasks The behaviour of a portal and the coordinator task are considered to be two separate processes, which simplifies the description of net update.

Undetected loops Multiple, rapid changes to the net topology may cause a loops that does not meet the loop criterion, hence is not detected and may cause net update to not terminate. A separate net reset protocol called *net panic* has been introduced to remedy this situation, which is described in Section 3.3.

Initial route map The initial value of a portal's route map has been changed many times, from 'vendor dependent' via several different initialisations to the final requirement which is that no bus id is in use.

Bus enumeration Although strictly outside the scope of the net update research, an addressing problem in the bus enumeration protocol was detected, where an alpha portal contacting the prime portal with

1394.1 routing was not aware of the bus id of the prime portal. In the final version, the request is sent by the co-portal of the requesting alpha, once the co-portal's bus (which is nearer to the prime portal) has a bus id. The request is routed via the net tree towards the prime portal, and the response is routed via the appropriate bus tree back towards the co-portal of the requesting alpha, and forwarded over the bridge to its final destination.

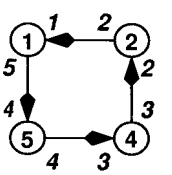
Route map updates Although bridge portals update their route maps in synchronisation during net update, the cleaning of destroyed bus ids upon completion of net update is not synchronised and can cause inconsistencies. The processing of net update information received over the bridge has been slightly adjusted, and in some cases such information requires net update to be started again on the co-portal's bus, once the local bus has completed net update.

Correctness properties A number of correctness properties have been defined, which must hold in a stable situation, and which ensure that the net tree and bus trees have been spanned correctly. These have guided the construction of an abstract version of the protocol as well as the manual proofs for the lower level version, and have been added to the IEEE standard as a normative annex which e.g. enables debugging.

Depending on the outcome of the second recirculation ballot (which is expected to take place in early 2004), the draft standard will be finalised, or must be adjusted for yet another ballot round. In the latter case, it is perhaps possible to add graphical descriptions for the net update protocol, in the form of state machines. In order to conform with accepted IEEE 1394 format, these would have to be complemented with C code, part of which is already given in the standard. Since net update is so complex, it would help to have such graphical and precise additional explanation.

3.3 Net Panic

The basic criterion for detecting a loop in the net topology is that on a bus, there are two alpha portals, i.e. the bus has either outgoing edges or an outgoing edge and the prime portal in one net tree. It is perhaps not surprising that, after some net topology changes, it can also be the case that a loop exists in the network for which the criterion does not hold: all buses on the loop have exactly one outgoing edge in the net tree but the prime portal itself is not there. This can be recognised by observing the information of the distance of each portal to the root bus in the net tree. There must be at least one bus with an outgoing edge with a larger distance than a corresponding incoming edge. See the following figure.



In this situation, the net update algorithm does not terminate, but keeps being executed on consecutive buses in the loop, and the distances keep increasing. This is a major error.

Attempts to fix net update by undirecting the incoming or the outgoing edge failed. Finally, in consultation with the ballot response committee, a separate *net panic* algorithm was added to the net update functionality, with the property of resetting all information in each bridge portal in the net to the initial values. This means that each bridge is directed in the net tree and one of the two portals in each bridge is the prime portal, and no bus tree is in use. Obviously, when net panic completes and more than one bridge is present in the net, the spanning trees are not correct, hence net update must be started again to sort out the net tree, followed by bus enumeration to span new bus trees.
The net panic algorithm consists of the following steps.

0. *Stop bridge functionality:* The portal presents itself on the bus as though it has no 1394.1 functionality. To step 1, 2 or 3, depending on the net panic start condition (see below).
1. *The bus must panic:* The portal broadcasts a *panic* message to all portals on the bus. To step 2.
2. *Observe the bus:* With IEEE 1394 functionality, the portal observes if all portals have now stopped acting as a bridge. If so, to step 3, otherwise back to step 1.

3. *Co-portal must panic:* A *panic* message is sent to the co-portal to inform it that net panic has completed on this bus. To step 4.
4. *Co-portal finished:* As soon as a *panic* message has been received from the co-portal as well (this may have happened during an earlier step), the portal initialises all net update information and starts net update.

The net panic algorithm is started whenever one of the following is true.

Looping condition Upon completion of step 1 of the net update algorithm, it can be determined whether the alternative looping situation mentioned above occurs (this is done by the coordinator). The net panic condition holds if there are two edges with the same net identity, such that one is an outgoing edge and the other is an incoming or undirected edge, and the distance of the former's local portal is larger than the distance of the latter's local portal.
Net panic is started with step 0, and continued with step 1.

Unexpected prime If a portal receives a new net identity which is equal to its own unique identity, but the corresponding distance is not 0, then either non-termination of net update will follow (if a loop exists in the topology) or net update will terminate but in an inconsistent state (otherwise). In either case, net panic must remedy the situation. Net panic is started with step 0, and continued with step 1.

Net size too large In very extreme and rare scenarios, neither of the above two conditions may occur although net update does not terminate. In that case the only way to signal this is the ever-increasing value of the distance to the root bus in the net tree. If during net update, a bridge is adjusted such that its net identity changes, and the distance at the outgoing end exceeds the maximum net size of 1023 buses, net panic must be started. Net panic with is started step 0, and continued with step 1.

Panic message received If a portal receives a *panic* message while it is not executing net panic yet, it must do so. If *panic* was received over the bus, then the distance is copied, net panic is started with step 0, and continued with step 2. If received over the bridge, then the distance is copied and incremented by 1. If the new distance is less than 1023 (maximal net size), then net panic is started with step 0, and continued with step 1, otherwise net panic is started with step 0, and continued with step 3.

Note that the net panic message carries a parameter indicating the distance to the bus on which the algorithm was originally started. In this way the algorithm is guaranteed to stop by respecting the maximum size of the network.

3.4 Correctness Properties

This section contains the correctness properties that hold in stable states and express the specification or goal of net update: correct spanning trees. The manual proof of partial correctness is based on these properties as well as some invariants and behavioural properties which are mostly weakened versions, expressing something about the consistency of states during net update. Termination has not yet been shown, although care has been taken to avoid some obvious sources of non-terminating activity.

Once net panic and net update have completed and the topology is stable, the following holds. The properties are presented at the detailed level of bridge portals, if possible preceded by an abstract description at the abstract graph level (which is *slanted*).
For each bus within the net:

- *All adjacent bridges have the consistent net identity and distance information.*
  All portals have the same *prime*, *hops to prime* and *bus id* value.
- *There is one outgoing bridge iff there is no prime portal on the bus.*
  For exactly one portal, *alpha* is true.
- *For the local bus tree there are only incoming and undirected edges.*
  Ift some portal's *bus id* b is not unassigned, then *route map*[b] is valid for each portal.
- *For each other bus tree, either it is not in use or there is exactly one outgoing edge.*
  For each possible bus id b not equal to any portal's *bus id*, either (1) *route map*[b] is clean for all portals, or (2) *route map*[b] is forward for one portal and valid for all other portals.

For each bridge within the net:

- *The portals in the bridge have consistent net identity and direction information.*
  Both portals have the same *prime* and *mute* value.
- *For each bus tree in use, each directed edge in the net tree corresponds to a directed edge in the bus tree.*
  If *mute* is false, for each bus id b, either *route map*[b] is clean for both portals or it is forward for one portal and valid for the other;
- *Each undirected edge in the net tree corresponds to an undirected edge in the bus tree.*
  If *mute* is true, for each bus id b, *route map*[b] is equal for both portals and it is either clean or valid;
- *The portals in the bridge have a consistent direction.*
  For at most one portal *alpha* is true and the portal's unique id is not equal to *prime*
- *The portals in the bridge have consistent distance information.*
  If *mute* is false, for one portal *alpha* is true, the portal's unique id is not equal to *prime*, and its *hops to prime* is 1 greater than its co-portal's *hops to prime*.

For each portal within a bridge:

- If *mute* is true, *alpha* is true only if the portal's unique id is equal to *prime*
- *The prime portal has distance 0.*
  If the portal's unique id is equal to *prime*, then *alpha* is true and *hops to prime* is 0.

References

[1] W.H.J. Feijen and A.J.M. van Gasteren. *On a method of multiprogramming*. Springer-Verlag, 1999.

[2] N. Goga and J.M.T. Romijn. Guiding Spin simulation. Technical report, 2004. Manuscript. Available on `http://www.win.tue.nl/oas/en\_index.html`.

[3] The Institute of Electrical And Electronics Engineers, Inc. *IEEE Standard for a High Performance Serial Bus*, August 1996. IEEE Std 1394-1995.

[4] The Institute of Electrical And Electronics Engineers, Inc. *IEEE Standard for a High Performance Serial Bus – Amendment 1*, December 2001. IEEE Std 1394a-2000.

[5] The Institute of Electrical And Electronics Engineers, Inc. *IEEE P1394.1 Draft Standard for High Performance Serial Bus Bridges*, November 2003. Version 1.05.

[6] I.A. van Langevelde, J.M.T Romijn, and N. Goga. Founding firewire bridges through promela prototyping. In *17th International Parallel and Distributed Processing Symposium (IPDPS), 8th International Workshop on Formal Methods for Parallel Programming: Theory and Applications (FMPPTA)*. IEEE Computer Society Press, 2003.

[7] A.J. Mooij, N. Goga, and J.W. Wesselink. A distributed spanning tree algorithm for topology-aware networks. In *Proceedings of the Conference on Design, Analysis, and Simulation of Distributed Systems 2004*. The Society for Modeling & Simulation International (SCS), 2004. To appear.

[8] A.J. Mooij and J.W. Wesselink. A formal analysis of a dynamic distributed spanning tree algorithm. Computer Science Report 03-16, Technische Universiteit Eindhoven, Eindhoven, December 2003.

[9] R. Perlman. An algorithm for distributed computation of a spanning tree in an extended LAN. *ACM SIGCOMM Computer Communication Review*, 15(4), September 1985.

[10] R. Perlman. *Interconnections: bridges, routers, switches, and internetworking protocols*. Addison-Wesley, 2000.

Pages 31-41 have been deleted for privacy reasons.

8 Statuten

Artikel 1.
1. De vereniging draagt de naam: "Nederlandse Vereniging voor Theoretische Informatica".
2. Zij heeft haar zetel te Amsterdam.
3. De vereniging is aangegaan voor onbepaalde tijd.
4. De vereniging stelt zich ten doel de theoretische informatica te bevorderen haar beoefening en haar toepassingen aan te moedigen.

Artikel 2.
De vereniging kent gewone leden en ereleden. Ereleden worden benoemd door het bestuur.

Artikel 3.
De vereniging kan niet worden ontbonden dan met toestemming van tenminste drievierde van het aantal gewone leden.

Artikel 4.
Het verenigingsjaar is het kalenderjaar.

Artikel 5.
De vereniging tracht het doel omschreven in artikel 1 te bereiken door
a. het houden van wetenschappelijke vergaderingen en het organiseren van symposia en congressen;
b. het uitgeven van een of meer tijdschriften, waaronder een nieuwsbrief of vergelijkbaar informatiemedium;
c. en verder door alle zodanige wettige middelen als in enige algemene vergadering goedgevonden zal worden.

Artikel 6.
1. Het bestuur schrijft de in artikel 5.a bedoelde bijeenkomsten uit en stelt het programma van elk van deze bijeenkomsten samen.
2. De redacties der tijdschriften als bedoeld in artikel 5.b worden door het bestuur benoemd.

Artikel 7.
Iedere natuurlijke persoon kan lid van de vereniging worden. Instellingen hebben geen stemrecht.

Artikel 8.
Indien enig lid niet langer als zodanig wenst te worden beschouwd, dient hij de ledenadministratie van de vereniging daarvan kennis te geven.

Artikel 9.
Ieder lid ontvangt een exemplaar der statuten, opgenomen in de nieuwsbrief van de vereniging. Een exemplaar van de statuten kan ook opgevraagd worden bij de secretaris. Ieder lid ontvangt de tijdschriften als bedoeld in artikel 5.b.

Artikel 10.
Het bestuur bestaat uit tenminste zes personen die direct door de jaarvergadering worden gekozen, voor een periode van drie jaar. Het bestuur heeft het recht het precieze aantal bestuursleden te bepalen. Bij de samenstelling van het bestuur dient rekening gehouden te worden met de wenselijkheid dat vertegenwoordigers van de verschillende werkgebieden van de theoretische informatica in Nederland in het bestuur worden opgenomen. Het bestuur kiest uit zijn midden de voorzitter, secretaris en penningmeester.

Artikel 11.
Eens per drie jaar vindt een verkiezing plaats van het bestuur door de jaarvergadering. De door de jaarvergadering gekozen bestuursleden hebben een zittingsduur van maximaal twee maal drie jaar. Na deze periode zijn zij niet terstond herkiesbaar, met uitzondering van secretaris en penningmeester. De voorzitter wordt gekozen voor de tijd van drie jaar en is na afloop van zijn ambtstermijn niet onmiddellijk als zodanig herkiesbaar. In zijn functie als bestuurslid blijft het in de vorige alinea bepaalde van kracht.

Artikel 12.
Het bestuur stelt de kandidaten voor voor eventuele vacatures. Kandidaten kunnen ook voorgesteld worden door gewone leden, minstens een maand voor de jaarvergadering via de secretaris. Dit dient schriftelijk te gebeuren op voordracht van tenminste vijftien leden. In het geval dat

het aantal kandidaten gelijk is aan het aantal vacatures worden de gestelde kandidaten door de jaarvergadering in het bestuur gekozen geacht. Indien het aantal kandidaten groter is dan het aantal vacatures wordt op de jaarvergadering door schriftelijke stemming beslist. Ieder aanwezig lid brengt een stem uit op evenveel kandidaten als er vacatures zijn. Van de zo ontstane rangschikking worden de kandidaten met de meeste punten verkozen, tot het aantal vacatures. Hierbij geldt voor de jaarvergadering een quorum van dertig. In het geval dat het aantal aanwezige leden op de jaarvergadering onder het quorum ligt, kiest het zittende bestuur de nieuwe leden. Bij gelijk aantal stemmen geeft de stem van de voorzitter (of indien niet aanwezig, van de secretaris) de doorslag.

Artikel 13.

Het bestuur bepaalt elk jaar het precieze aantal bestuursleden, mits in overeenstemming met artikel 10. In het geval van aftreden of uitbreiding wordt de zo ontstane vacature aangekondigd via mailing of nieuwsbrief, minstens twee maanden voor de eerstvolgende jaarvergadering. Kandidaten voor de ontstane vacatures worden voorgesteld door bestuur en gewone leden zoals bepaald in artikel 12. Bij aftreden van bestuursleden in eerste of tweede jaar van de driejarige cyclus worden de vacatures vervuld op de eerstvolgende jaarvergadering. Bij aftreden in het derde jaar vindt vervulling van de vacatures plaats tegelijk met de algemene driejaarlijkse bestuursverkiezing. Voorts kan het bestuur beslissen om vervanging van een aftredend bestuurslid te laten vervullen tot de eerstvolgende jaarvergadering. Bij uitbreiding van het bestuur in het eerste of tweede jaar van de cyclus worden de vacatures vervuld op de eerstvolgende jaarvergadering. Bij uitbreiding in het derde jaar vindt vervulling van de vacatures plaats tegelijk met de driejaarlijkse bestuursverkiezing. Bij inkrimping stelt het bestuur vast welke leden van het bestuur zullen aftreden.

Artikel 14.

De voorzitter, de secretaris en de penningmeester vormen samen het dagelijks bestuur. De voorzitter leidt alle vergaderingen. Bij afwezigheid wordt hij vervangen door de secretaris en indien ook deze afwezig is door het in jaren oudste aanwezig lid van het bestuur. De secretaris is belast met het houden der notulen van alle huishoudelijke vergaderingen en met het voeren der correspondentie.

Artikel 15.

Het bestuur vergadert zo vaak als de voorzitter dit nodig acht of dit door drie zijner leden wordt gewenst.

Artikel 16.

Minstens eenmaal per jaar wordt door het bestuur een algemene vergadering bijeengeroepen; één van deze vergaderingen wordt expliciet aangeduid met de naam van jaarvergadering; deze vindt plaats op een door het bestuur te bepalen dag en plaats.

Artikel 17.

De jaarvergadering zal steeds gekoppeld zijn aan een wetenschappelijk symposium. De op het algemene gedeelte vaan de jaarvergadering te behandelen onderwerpen zijn

a. Verslag door de secretaris;

b. Rekening en verantwoording van de penningmeester;

c. Verslagen van de redacties der door de vereniging uitgegeven tijdschriften;

d. Eventuele verkiezing van bestuursleden;

e. Wat verder ter tafel komt. Het bestuur is verplicht een bepaald punt op de agenda van een algemene vergadering te plaatsen indien uiterlijk vier weken van te voren tenminste vijftien gewone leden schriftelijk de wens daartoe aan het bestuur te kennen geven.

Artikel 18.

Deze statuten kunnen slechts worden gewijzigd, nadat op een algemene vergadering een commissie voor statutenwijziging is benoemd. Deze commissie doet binnen zes maanden haar voorstellen via het bestuur aan de leden toekomen. Gedurende drie maanden daarna kunnen amendementen schriftelijk worden ingediend bij het bestuur, dat deze ter kennis van de gewone leden brengt, waarna een algemene vergadering de voorstellen en de ingediende amendementen behandelt. Ter vergadering kunnen nieuwe amendementen in behandeling worden genomen, die betrekking hebben op de voorstellen van de commissie of de schriftelijk ingediende amendementen. Eerst wordt over elk der amendementen afzonderlijk gestemd; een amendement kan worden aangenomen met

gewone meerderheid van stemmen. Het al dan niet geamendeerde voorstel wordt daarna in zijn geheel in stemming gebracht, tenzij de vergadering met gewone meerderheid van stemmen besluit tot afzonderlijke stemming over bepaalde artikelen, waarna de resterende artikelen in hun geheel in stemming gebracht worden. In beide gevallen kunnen de voorgestelde wijzigingen slechts worden aangenomen met een meerderheid van tweederde van het aantal uitgebrachte stemmen. Aangenomen statutenwijzigingen treden onmiddellijk in werking.

Artikel 19.

Op een vergadering worden besluiten genomen bij gewone meerderheid van stemmen, tenzij deze statuten anders bepalen. Elk aanwezig gewoon lid heeft daarbij het recht een stem uit te brengen. Stemming over zaken geschiedt mondeling of schriftelijk, die over personen met gesloten briefjes. Uitsluitend bij schriftelijke stemmingen worden blanco stemmen gerekend geldig te zijn uitgebracht.

Artikel 20.

a. De jaarvergadering geeft bij huishoudelijk reglement nadere regels omtrent alle onderwerpen, waarvan de regeling door de statuten wordt vereist, of de jaarvergadering gewenst voorkomt.

b. Het huishoudelijk reglement zal geen bepalingen mogen bevatten die afwijken van of die in strijd zijn met de bepalingen van de wet of van de statuten, tenzij de afwijking door de wet of de statuten wordt toegestaan.

Artikel 21.

In gevallen waarin deze statuten niet voorzien, beslist het bestuur.