Volume 15 Number 2 May 1991
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Informatica

A Journal of Computing and Informatics

Subscription Information

Informatica (YU ISSN 0350 – 5596) is published four times a year in Winter, Spring, Summer and Autumn (4 issues).

The subscription price for 1990 (Volume 14) is US$ 30 for companies and US$ 15 for individuals.

Claims for missing issues will be honoured free of charge within six months after the publication date of the issue.

Printed by Tiskarna Tipa, Gosposvetska 13, Ljubljana.

Informacija za naročnike

Informatica (YU ISSN 0350 – 5596) izide štirikrat na leto, in sicer v začetku januarja, aprila, julija in oktobra.

Letna naročnina v letu 1991 (letnik 15) se oblikuje z upoštevanjem tečaja domače valute in znaša okvirno za podjetja DEM 30, za zasebnike DEM 15, za študente DEM 8, za posamezno številko pa DM 10.

Številka žiro računa: 50101 – 678 – 51841.

Zahteva za izgubljeno številko časopisa se upošteva v roku šestih mesecev od izida in je brezplačna.

Tisk: Tiskarna Tipa, Gosposvetska 13, Ljubljana.

Na podlagi mnenja Republiškega komiteja za informiranje št. 23 – 85, z dne 29. 1. 1986, je časopis **Informatica** oproščen temeljnega davka od prometa proizvodov.

Pri financiranju časopisa **Informatica** *sodeluje Republiški komite za raziskovalno dejavnost in tehnologijo, Tržaška 42, 61000 Ljubljana*

Volume 15 Number 2 May 1991
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Letnik 15 Številka 2 Maj 1991 YU ISSN 0350 – 5596

Informatica

Časopis za računalništvo in informatiko

V S E B I N A

MEANING, UNDERSTANDING, SELF-REFERENCE

INFORMATICA 2/91

Keywords: Meaning, Chinese room argument, self-reference

Damjan Bojadžiev
Institute Jožef Stefan, Ljubljana

Arguments against the possibility of machine understanding as symbol manipulation tend to downplay the internal structure of the computational system. The case for genuine mechanical understanding can be made more appealing by considering the levels of (self)-representation and the self-referential mechanisms operating between them.

Pomen, razumevanje, samo-referenca: Argumenti proti možnosti mehaničnega razumevanja kot manipulacije simbolov običajno podcenjujejo notranjo strukturo programskega sistema. Možnost resničnega mehaničnega razumevanja se zdi bolj smiselna, če upoštevamo nivoje (samo)reprezentacije in samo-referenčne mehanizme, ki delujejo med njimi.

1. Introduction

Mainstream tradition in philosophy and more or less educated common sense maintains that there is (and will always be) a huge gap between the cognition, and in particular the linguistic abilities, of men and machines (computers as their most advanced representatives). It is said that, by manipulating symbols, computers might at most succeed in producing an illusion of understanding: appearing to understand what is being said/typed to them and what they themselves say or print, but without actually doing so. Although the current state of the art of artificial intelligence and natural language understanding does not make this a burning issue, many different positions have already been taken on it, presumably because the possibility of machine understanding challenges our current linguistic and cognitive supremacy, and because of the extent to which it affects our self-image as cognitive beings, our understanding of ourselves. The prospect of mechanical understanding apparently tends to have a negative effect on our conception of ourselves, though it is not obvious that it should do so. The indignant position that we are not "mere machines" would only go along with what is sometimes called weak AI, the position that AI programs might only provide superficial I/O simulations and need not be regarded as actual models of human cognitive performance.

The present paper is an overview of some typical arguments, notably yet another debunking of Searle's "Chinese room" argument (section 2) and a compilation of some programatic answers, centering on the notion of self-reference (section 3). The main thesis elaborated there is that the connection between meaning, understanding and self-reference goes through the subject of that understanding, which is construed as an effect of self-referential mechanisms in a meta-level architecture. If the objections against computational understanding are summarized by saying, as Searle does, that computers have syntax but not semantics, and that syntax alone is not sufficient for semantics [19:34], the answer can be summarized, following Casti [4:334-5], by saying that syntax and self-reference may add up to semantics.

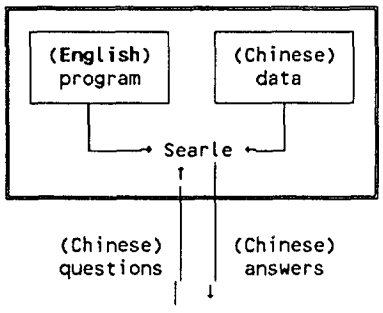
2. Real Understanding

The standard objection against computational understanding by means of symbol manipulation in a formal representational system says:

computers themselves don't mean anything by their tokens (any more than books do) - they only mean what we say they do [7:32-3]; symbols must mean something to the system performing the operations [6].

The second quotation comes from the philosopher F. Dretske, who helps us keep a cool head in such discussions by reminding us that computers are tools, and that we should not get carried away in attributing conceptual or cognitive capacities to them. According to Dretske, the meaning of internal signs consists in their correlation with external conditions, and affects (through learning) the way the system manages the signs. (His negative conclusions about computational cognition seem to rest on the premise that robots can't learn)

A much debated argument about machine understanding is the thought experiment suggested a decade ago by the philosopher J. Searle, known as the Chinese Room argument [19]. Searle imagines that he could pass a (Turing) test for understanding eg. Chinese in the following way, although he does not actually know the language:



Isolated in a room, Searle would carry out instructions (in English) for manipulating incoming strings of Chinese symbols using given strings of Chinese data, and producing answer strings such as would be given by a speaker of Chinese. It is important in this setup that the instructions only say how to manipulate formally characterized Chinese symbols, and thus do not interpret them (in English). Searle then claims that, just as he would not have understood a word of the Chinese, although he would appear to do so from the point of view of the external questioner, a computer executing the same instructions would also not understand anything[1]:

In the Chinese case, the computer is me, and in cases where the computer is not me, the computer has nothing more than I have in the case where I understand nothing [Minds, Brains and Programs].

It seems obvious that Searle would indeed not come to understand Chinese in this way, although even that might be allowed as an abstract possibility[2]. More importantly, a computer taking Searle's place would indeed also not understand anything, but only on Searle's peculiar usage of the word 'computer'. Searle seems to think of a computer as an "empty" processor, without programs and data: he almost seems to identify with the CPU; more accurately, he identifies with the interpreter (in the computational sense of the word) of the formal instructions for manipulating Chinese symbols. No one has ever claimed understanding (of Chinese or anything else) for a computer in this sense; it is the programmed computer (plus data) that might be said to understand - this is what Searle calls the systems reply to his argument[3]. Torrance aptly summarizes it by relating it to an everyday situation: it is not the washing machine's motor which washes clothes, but the washing machine [22:15].

The part I find interesting about the Chinese room thought experiment[4] is Searle's attempt to identify with an understanding computer. According to strong AI, which he is trying to refute, computational understanding models human understanding, so that some identification should be possible; Searle only makes a type error in the particular identification he imagines. The identification does not go through because we do not recognize a (formal) description of what it is for symbols to be meaningful to us in the work of the fast, dumb homunculus whose place Searle thinks he could take. The systems reply to the "experiment" suggests another identification, with the whole Chinese room (or, in a parallel processing version with some Chinese hotel). A variant on this would be identification with (the perspective of) the simulated speaker [10:378]. These possibilities correspond to the two senses in which we speak of symbols being meaningful to us: meaningful to our whole cognitive system or meaningful to some particular point within the system: consciousness, self, mind's eye or I, whatever; such labels only presume what they appear to explain. A short argument for the connection between understanding and "selfhood" can be found in Haugeland [8:240]: ego involvement may be integral to understanding, and ego involvement requires a self-concept. A more technical one can be built around the notions of levels of representation in a symbolic system and the relationships between such levels. An early, idealized attempt to discern some analogue of subjectivity within a formal system was made by Hofstadter, in his 'Gödel, Escher, Bach' [9]. He proposed an arithmetical, Pythagorean version of the interplay between levels as engendering the self: it is formed when it can reflect itself [9:709]. The

(meta)arithmetical basis of the slogan were the effects of sentential self-reference for particular predicates: the "strange loop" connecting the (un)-provability of a sentence and the sentence's assertion of its own (un)provability. In hard(er) computer science, ideas about the effects of incorporating the meta-level in a formal representational system were further developed under the label meta-level architectures[5].

3. Self-Referential Systems

If the symbols of a system are to be meaningful to the system itself, it must first be able to consider them and the way it manipulates them. Thus, Perlis [15] has suggested that a system would use symbols meaningfully if it could consider its own symbolic forms and distinguish between its symbols and what they symbolize. Both conditions require second-order representation (naming or otherwise representing the symbols themselves) and could thus be met in a system with quotation[6]. The suggestion may appear singularly inappropriate, since the alleged problem with computational understanding was that a computer has no access to what its symbols symbolize (for us), that it cannot get "down" from symbols to symbolized (things, meanings): quotation can then only get it one level further up, and disquotation (through a truth predicate) can only get it down to where it was in the first place[7]. Quotation would thus appear to offer no help with the original problem, but then it turns out that the suggestion also includes the position that there is actually no such problem: Perlis says that

> we are reduced to dealing with symbols and their meanings, whatever they are, via expressions or other internal forms ... we never get to the outer "thing in itself" ... expressions or other internal forms do all the work, but at least one is momentarily taken as the thing-in-itself [How can a Program Mean].

The first part of this is the point frequently emphasized by Wilks: there is no escape from the world of symbols to the real world, since the world only comes to a symbolic system through further (sub)symbols. The second, metaphysical part of Perlis' position (reality is projected representation) might at first appear more startling, though it is presumably only a consequence of the first; Jackendoff claims something similar when he talks of a mentally supplied attribute of "out-thereness" and of our being constructed so as normally to be unaware of our own contribution to experience.

Perlis's suggestion could be characterized in currently influential terms by saying that the system must have a meta-level architecture, capable of self-reflection, or self-reference: being able to move to the meta-level and take a stance towards its own symbolic structures (notably judge them true or false), and then descend back to the object level. But in such a system it would seem possible to discern some features of subjectivity in the possibilities of self-representation and mediation between levels of representation. A connection between subjectivity and meaningfulness to the system would seem to make methodological sense, since meaning and the instance to which it is present can only be correlative phenomena, to be explained simultaneously in a computational or any other model. The basic idea is that the self (subjectivity) can be computationally characterized by rules relating (at least) two levels of representation: a ground, primitive level at which we have no representation of ourselves, and a higher, general level, at which we have a neutral representation of ourselves as just another object. Thus, Perry [16] sees indexicals (token-reflexive expressions such as personal pronouns) as mediating between such levels:

> At the "bottom" level, we have cognitions that have no representation of ourselves (or the present moment), which are tied pretty directly to cognition and action ... Since [simple organisms] are always in the background of their perceptions and actions, they need not be represented in the cognitions that intervene between them [Self-Knowledge and Self-Representation].

According to Smith [20], self-referential mechanisms suggest a computational idea of the self as mediating between "blindly" efficient, indexical representations and generic, (more) explicit representation of its circumstances. Smith considers three self-referential mechanisms which vary the theme of self-recognition: paraphrasing roughly, these are: autonimy (recognizing one's own name), introspection (recognizing one's own (implicit) internal structure), and reflection (recognizing oneself in the world)[8]. These mechanisms correspond to different conceptions of self: as a unit, complex system, and independant agent, respectively[9]. Most of the literature on meta-level architectures is concerned with introspection, since many problems of independant interest to the AI effort naturally fall in this framework: learning, planning, default reasoning, truth maintainance, reasoning about control, reasoning about beliefs, incomplete and inconsistent knowledge, handling impasses in reasoning [2], [13]. The classical problems with self-referential paradoxes in mathematical logic and elsewhere would come under the heading of narrow, sentential self-reference (of a sentence to itself), as opposed to introspection in

general as reference of a system to aspects of itself.

In the area of natural language understanding, self-referential structures will presumably appear with the inclusion of meta-level representations of knowledge, linguistic and otherwise. A major effort at formalising the meaning of natural language expressions, Montague semantics, can also be seen to fit in the framework of meta-level architectures (incorporating the meta-level of intensions into the object language) [21]. In general, systems for computational understanding of language will have to include part of their own meta-level because of the range of self-referential phenomena in communication. Thus, in addition to information about (external) situations to which they refer, statements also convey information about the speaker: his other beliefs (about himself and others), his distance (spatial, temporal, ...) from what he is talking about [1:30]; linguistic actions - asking (for help or information), lying, threatening, promising - generally have a self-regarding nature [10:267]. It would thus seem, as Perlis [14] has contended, that a proper understanding of language will in the long run be found to depend upon self-referential abilities.

N o t e s

1. Searle still maintains these views, as can be seen from his recent exposition in Scientific American, titled 'Is the Brain's Mind a Computer Program?' [19]. The subtitle gives a curious answer: 'No. A program merely manipulates symbols, whereas a brain attaches meaning to them'. The notion of a brain attaching meaning to symbols seems rather strange; a mind (subject) would seem better suited for that role.

2. Searle doesn't mention this possibility, though he discusses a similar one, on which he might assign some other, arbitrary interpretation to the symbols he is manipulating (chess moves, stock market predictions, ...) [19]. The structure of the formal system may allow such interpretations, but then similar reinterpretations could also be applied to what Searle (or anyone else) ordinarily says (in English, outside the Chinese room). Usually, we think of such reinterpretation, on a smaller scale, only when what is being said fits very badly with what it is said about. But the abstract possibility of such reinterpretation of the whole language in general remains, and has been explored in arguments directed against the importance of the relation of reference in semantics (permutations of reference; see eg. Davidson's paper 'The Inscrutabilty of Reference' in [5].

3. Searle's reply to the systems reply is that, if understanding is not ascribed to him but to the system <program,data,Searle>, in which he is the agent, he will simply internalize (memorize) the other components:

> The individual then incorporates the entire system. There isn't anything at all to the system that he does not encompass ... All the same, he understands nothing of the Chinese, and a fortiori neither does the system, because there isn't anything in the system that isn't in him [Minds, Brains, Programs].

Searle rightly says that he feels somewhat embarassed to give this 'quite simple' reply: paraphrasing to bring out its curiously childish nature, Searle seems to be saying: "OK, if understanding is not in me but in (my relationship to) these other things outside me, then I'll just put them inside me". Searle seems to be confusing physical with functional containment: he would still be only the agent in this internalized (sub)system, through which he would now be speaking in tongues. The argument

> The whole doesn't understand
> ---
> So, the parts don't.

only seems to bug the question: that is exactly a situation in which the whole doesn't understand while parts do.

4. Searle's "experiment" does dramatize some issues concerning the formalization of meaning and understanding: where or how is meaning present in a formal system (program), and who or what is that meaning present to. Common sense answers invoke the programmer: it is his knowledge of meaning that goes into setting up a formal system, and the meaning captured in it is present to him (as attributed meaning). Other attempts rely on some kind of (w)holism: the meaningfullness of individual symbols is an effect of the whole system, deriving from the relationships of individual symbols with indefinitely many other symbols and with the procedures which manipulate them. Symbols would then be meaningful to the system if it considered not only their immediate, explicit, locally computable ... properties, but also their relationships with indefinitely many other symbols; cf. Hofstadter in [9:582].

5. A survey of three early attempts at computational introspection (Smith's introspective LISP, Weyhrauch's introspective first order logic, FOL, and Doyle's model of introspective action and deliberation) can be found in [2]. More recent efforts are collected in the proceedings of the 1986 workshop on meta-level architectures and reflection, sponsored by the COST-13 project Advanced Issues in Knowledge Representation [13].

6. Perlis uses a more general idea of quotatation, according to which not only internal tokens, but any internal (mental) object or process can be "quoted" (or reflected, as he also says, meaning something similar to what Hofstadter calls jumping out).

7. The ideas of language levels (use and mention (quotation)) and disquotation (truth) are basic to Davidson's semantics for natural language [5]; I hadn't seen Perlis's point in my estimate of the usefulness of Davidson's semantics for computational understanding of language [3].

8. The basic, low-level view of the world, at which we have no representation of ourselves, could be compared to the perspective of the "man with no head" in [10:23-33]. The higher level consists of additional representational structure, which is basic to what Smith has to say about the self-referential mechanisms of the self. In the arithmetical case, considered by Hofstadter, there is no additional representational structure on the higher level, and it is the Gödel code which provides "additional", meta-arithmetical interpretations.

9. The importance of introspective mechanisms in evolutionary biological engineering was pointed out by Lycan:

Parallel processing, time-sharing and hierarchical control, all vital to the fabulous efficiency of such complex sensor-cognition-motor systems as we human beings are, individually and together require a formidable capacity for internal monitoring ... As a matter of engineering, if we did not have the devices of introspection, there would be no we to argue about, or to do the arguing [Consciousness:72-3].

References

1. Barwise, J., Perry, J., Situations and Attitudes, MIT Press 1983

2. Batali, J., Computational Introspection, MIT AI Memo 701, 1983

3. Bojadžiev, D., Davidson's Semantics and Computational Understanding of Language, Grazer Philosophische Studien, Vol.36, 1989

4. Casti, J.L., Paradigms Lost: Images of Man in the Mirror of Science, W.Morrow & Co. 1989

5. Davidson, D., Inquiries into Truth and Interpretation, Oxford Univ. Press 1984

6. Dretske, F., Machines and the Mental, Am. Phil. Assoc. Presidential Address, 1985

7. Haugeland, J., Mind Design, Bradford Books 1981

8. ----------, Artificial Intelligence: The very Idea, Bradford Books 1985

9. Hofstadter, D.R., Gödel, Escher, Bach: An Eternal Golden Braid, Basic Books 1979

10. ----------, Dennett, D.C., The Mind's I - Fantasies and Reflections on Self and Soul (1981), Penguin 1982

11. Jackendoff, R., Semantics and Cognition, MIT Press 1983

12. Lycan, W.G., Consciousness, Bradford, MIT 1987

13. Maes, P., Nardi, D. (eds.), Meta-Level Architectures and Reflection, Elsevier 1988

14. Perlis, D., Languages with Self-Reference I: Foundations, Artificial Intelligence 25, 301-22, 1985

15. ----------, How Can a Program Mean?, in Proceedings of the 10.th IJCAI, Vol.1, Milano 1987

16. Perry, J., Self-knowledge and Self-representation, in Proceedings of the 9.th IJCAI, Vol.2, Los Angeles 1985

17. Searle, J., Minds, Brains and Programs, Behavioral & Brain Sc. 3, 1980; reprinted eg. in [7]

18. ----------, Minds, Brains and Science, The 1984 Reith Lectures, British Broadcasting Corporation 1984

19. ----------, Is the Brain's Mind a Computer Program?, Sc. Am. Vol.262 No.1, Jan. 1990

20. Smith, B.C., Varieties of Self-reference, in J.Y.Halpern (ed.), Theoretical Aspects of Reasoning about Knowledge, Proceedings of the 1986 Conf., Morgan Kaufmann 1986

21. Steels, L., Meaning in Knowledge Representation, in [13]

22. Torrance, S. (ed.), The Mind and the Machine - Philosophical Aspects of Artificial Intelligence, Ellis Horwood 1984

ROUTING ALGORITHMS FOR PACKET SWITCHED NETWORKS

INFORMATICA 2/91

Keywords: Routing Algorithms, Packet Switched Networks

Iskra Djonova Popova
Research Center for New Technologies
Macedonian Academy of Sciences and Arts
Skopje

Abstract: In this paper a short survey of the routing problem in packet-switched networks is provided. A number of shortest path type procedures for several networks are presented (ARPANET, TYMNET and TRANSPAC). An optimal routing approach for the networks with stationary input traffic statistics is itroduced as a more sofisticated alternative. The interaction between the flow control sheme and the routing algorithm in the network is shown as a chalenge for development of a class of algorithms that jointly addresses the problem of routing and flow control.

1. INTRODUCTION

Packet switching, [1], have been inspired by the idea of sharing communication channel capacity across a number of users by implementing the same time slicing philosophy that had earlier proved so sucessfull in sharing the execution power of a single processor across many users processes. Every user node that is conected to a packet switched common carrier makes a contract with the carrier (i.e., follows standard protocols) to hand him bit streams already segmented (packetized), with each packet supplemented with a header saying among other things, to which user node he whishes the packet delivered. The consequent design is then a network which handles small pieces of information (packets) at the highest possible transmission rates. Packet switching does allow the carrier to offer to the user more of the access path function, freedom in buffering and delayed delivery, [2]. Such a network can be reconfigured at tremendous speed, i.e. reconfigured in the sence of providing connections between different dispatching priorities or useing different routs.

The routing of packets from a source node to a destination node is an important issue in the design of packet switched networks. The routing problem consists of how to allocate the available resources in the network in order to accomplish the transsmision of the information offered in the nodes while performing the best possible service. The objective is to optimize the mean packet delay at all levels of the network load. The effect of good routing is to increase throughput for the same value of avarage delay per packet under havy load conditions while decreasing average delay per packet under lighter load conditions.

2. CLASSIFICATION

There are a number of ways to classify routing algorithms. One of the possible classifications relates to wether they change routs in responce to the traffic input patterns. In static routing algorithms, the path used by the sessions of each origin-destination pair is fixed regardless of traffic conditions. This type of algorithm cannot achieve a high throughput under a broad variety of traffic input patterns. The idea behind the adaptive routing algorithms is to change the routs in response to network congestion and guide the traffic between origins and destinations around the point of overload. The static versus adaptive classification is particulary vague, since all networks provide some type of adaptivity to accomodate topological changes (link and/or node

coming up or going down, new topologies being established).

. Another way of classification is to devide them into centralized and distibuted. In centralized algorithms routing strategies are prepared centrally at a centralized network routing center (NRC), while in distributed tehnicques the strategy is prepared throughout the network. The centralized method suffers from high communication overhead, and must deal with the problem of handling link and node failures. An even more serious problem is how to handle the failure of the NRC itself. It appears that distributed routing avoids some of these problems. In this case each node makes its own routing decision based on the information recieved from its neighboring nodes. One potential problem here is that inconsistent routing paths can cause looping of packets and possibly deadlocks.

A different classification could be made according to the kind of information used when implementing the routing algorithm. It is either local, i.e., only the locally avaliable information at the nodes is used,or global, when the comprehensive knoledge about network performance is used.

While routing pocedures can be setup within the network more or less independently of the protocols seen by the users (i.e., the devices external to the network), the choice of an appropriate routing procedure is influenced, to some extent, by the transport protocols operating at the network/user interface, [3]. It is convenient to classify these as either virtual circuit-oriented or message (datagram) oriented. In the former case, a virtual circuit is set up for each new session. The virtual circuit appears to the external devices as if it were a dedicated line; under normal operation, individual data packets arrive at the destination in the proper sequence. However, within the network, packets from many different virtual circuits are generally sharing the same communication lines. In the message-oriented case, communication is on a message-by-message basis. Each message or packet (often called a datagram) may travell along different routs. Therefore it must contain its own destination address. Figure 1 illustrates virtal circuit versus datagram network.

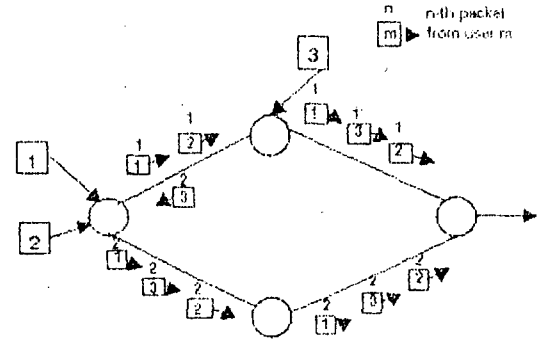


Figure 1a. Routing in datagram network. Two packets of the same user pair can travel along different routes.

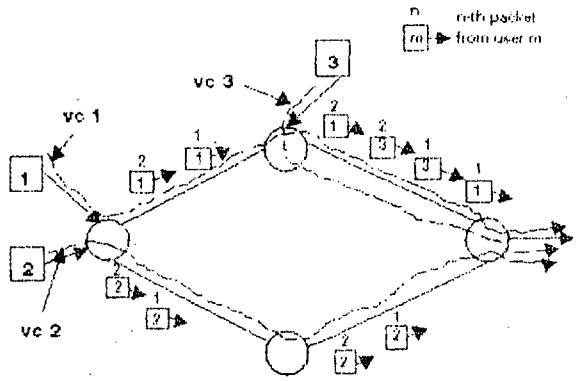


Figure 1b. Routing in a virtual circuit network. All packets of each virtual circuit use the same path.

3. FUNCTIONS OF ROUTING PROCEDURES

In an idealized situation where all parameters of the network are assumed to be known and not changing, it is possible to determine a static routing strategy which minimizes average network delay. The routing problem posed in this form is equivalent to the multicommodity flow problem, [4]. Changing situations in real networks, such as a line failure or change in the traffic distribution, necessate some degree of adaptivity. Any adaptive routing procedure must perform a number of functions listed below.

1. Measurement of the network parameters pertinent to the routing strategy.

2. Reporting the local state to neighbors or to the point or points at which routing computation take place.

3. Finding optimum routs based on the information recieved.

4. Constructing the routing tables to detemine which output line to use to route a packet towards its destination.

Typical information that is measured and used in routing computation consists of states of communication lines, estimated traffic, link delays, avaliable resources (line capacity, nodal buffers) etc.

There is a huge number of possible combinations in using this information and making the routing decisions. The variety is partly due to historical reasons and partly to the diversity of needs in different networks. The problem of routing has been investigated using different approaches. Some are based on heuristics (e.g. shortest path algorithms) while others formulate the routing problem as one of optimal control and than attempt to solve it using standard optimization tehnicque. In section 4 several routing procedures implemented in various packet-switched networks will be described, and in section 5 an optimal routing approach of the routing problem will be given.

4. SEVERAL ROUTING PROCEDURES USED IN PRACTICE

A. ARPANET was created in 1969 as an experiment in computer resource sharing. It is a distributed datagram network with at least two paths between any pair of nodes. The original routing algorithm was an ambitious, distributed, adaptive procedure that unfortunately had some fundamental flaws that were corected in 1979 when the algorithm was replaced by a new version, [5].

Both ARPANET algorithms are based on the notion of a shortest path between two nodes. Here each communication node is assigned a positive number called its lenght. The path between two nodes has a lenght equal to the sum of the lenghts of its links. Each packet is routed along a minimum lenght (or shortest) path bet ween the origin and the destination node of the packet. The idea here is that, if the lenght of each link is representative of the delay on the link, than a shortest path contains relatively few and uncongested links, and is therefore desirable for routing. The route calculation is performed in distributed manner, with each node basing its calculation on local information together with calculations made at every other node. Each update is transmitted to all nodes by the simple, but a reliable method, the so called flooding technique. A node broadcasts an update message to all nodes by sending the message to all its neighbors except to the one from which it was recieved, in turn they send the message to their neighbors, etc.

ARPANET algorithms are prone to oscillation. The phenomenon is explained by the idea that selecting routes through one area of the network increases the lenghts of the coresponding links. As a result, at the next routing update the algorithm tends to select routes through different areas. This makes the first area desirable at the subsequent routing update with an oscillation resulting. The feedback effect between link lenghts and routing updates was a primary flaw of the original ARPANET algorithm that caused dificulties over several years and eventually led to its replacemant. The latest algorithm is also prone to oscillations, but not nearly as much. In particular this algoritham solved the following problems in the original one:

1) The delay measurment procedure was changed. An instantenious measurement of the queue lenght in the old algorithm was replaced by the measurment of the actual delay of each packet crossing the link. Each link lenght is updated periodically every 10 seconds, instead of every 128ms., and is taken to be the avarage packet delay on the link during the preceding 10 second period. The avarage delay has been proved to be the best informa tion policy for the minimum delay algorithm, [6].

2) The algorithm for finding the shortes path spanning tree was replaced by the so called shortest path first algorithm which is based on one atributed to Dijkstra, [7]. The important addition to the basic algorithm is that changes in network topology or characteristics require only an incremental calculation rather then a complete recalculation of all short paths.

3) The interval between two succesive exhange of the shortest distances among neigboring nodes was changed from 625ms. to 10s. To reduce the amount of communication overhead involved in this information exhange, the 10s. avarage link delay measurements are not always transmited. Only when the change in link delay since the last transmission exeeds a certain treshold, does a new transmission take place. The treshold is reduced as time increases since the previous transmission.

Routing algorithm, as it presently exists in the ARPANET, is a greedy one. Switches attempt to grab the best resources (the fastest routes) without regard to their utilization or the effects on neighboring switching. A use of a congestion-based metric instead of delay could alleviates these problems, [8].

A replacement of the existing algorithm is being planned with a new one which uses multi-

ple paths for the same origin destination pair. Furthermore, traffic is devided in several service classes each routed independently of the others. The paths corresponding to an origin-destination pair, and service class are updated over fairly long time intervals. Because of the use of multiple paths, this type of algorithm is capable of higher throughput then the existing one.

B. TYMNET is a computer communication network developed in 1970 by Tymshare, Inc. of Cupertino, California. Almost all nodes are connected to at least two other nodes giving the network an alternate path capability. Routing is set up centrally on a virtual circuit basis by a supervisory program running on one of the four network routing centers. The routing algorithm is based on the shortest path metod, and is adaptive like the ARPANET algorithms. Unlike these algorithms this one does not have poten tial of oscillation, primarely due to the use of virtual circuits rather then datagrams.

A routing decision in TYMNET is needed only at the time a virtual circuit is set up. The Network routng center (also called the supervisor) assignes a virtual circuit to each sessiom by calculating the minimum cost path connecting the origin and the desti nation nodes of the virtual circuit. Integer-valued costs are asigned to each link. The cost assignments depend on the line speed, the line utilization as well as other factors, [9]. In the absence of overloading the algorithm tends to select the shortest path with highest transmission speed. As more users come on the network, the lower speed link begin to be used as well. In lightly loaded situations, users tend to have relatively shorter time delays through the network. The minimum hop paths, favored in the lightly loded case, also tend to be more reliable than ones with more links.

Once the path has been selected, the virtual circuit is assigned an 8-bit logical record number on each link it traverses. This allows up to 256 users or channels to share any one link. In addition, one number or channel is reserved for a node to communicate with supervisor and one channel is reserved for communica tions with the neighboring nodes. The NRC notifies each of the nodes along the path and the necessary information is entered in each node's routing tables. These tables are structured as shown on Figure 2. (Eight possible logical record numbers only have been assumed for simplicity.) In the original version of TYMNET (TYMNET I), the operating NRC maintainse an image of the routing tables of all nodes and explicitly reads and writes the tables in the nodes. In the latter version (TYMNET II), the nodes maintain their own tables.

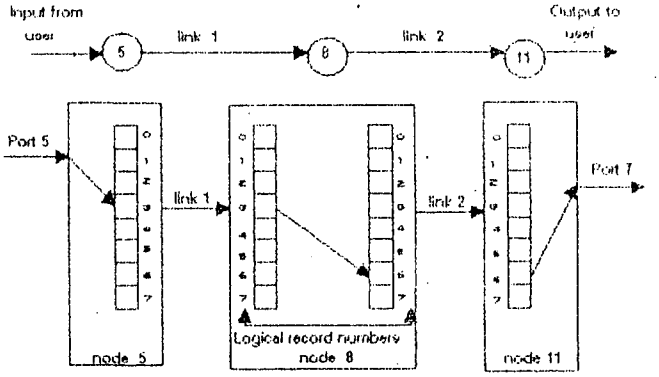


Figure 2. Structure of TYMNET routing tables. For the virtual circuit on the path 5-8-11 the routing table at node 5 mapps input port 5 onto channel 3 on link 1. At node 8 incoming channel 3 is mapped into outgoing channel 6 on link 2. At node 11, incoming channel 6 is mapped into output port 7.

The TYMNET algorithm is well thought out and has performed well over the years. A potential concern is its vulnerability to failure of the network routing centre which is handled by using backup central nodes.

C. TRANSPAC is the French public packet-switched network. It is a virtual circuit oriented system which uses partially decentralized routing algorithm, using a minimum cost, i. e. shortest path criterion. Link costs are defined in terms of link resource utilization. The concept used is similar to "delta routing algorithm" suggested by Rudin, [10]. Each node consists of a control unit (CU) to which are atached a number of switching units (SU). Each incident link is controled by an SU, which executes all data link procedures. Routing is handled by the CU, using information from the NRC (Network Manegemant Center in TRANSPAC terminology). The evaluation of link costs will be described first, and then the routing algorithm will be presented.

Concider a full duplex link conected to nodes m and n. Let $C_m(k)$, $C_n(k)$ be the cost assigned to link k as percived by nodes m and n, respectively, and let $C(k) = \mathrm{Max}[C_m(k), C_n(k)]$ be the "combined" estimate of link cost. The quantities $C_i(k)$, $i = m,n$ are the basic data on which routing computation is based. They are defined as a function of the level of utilization of two types of resources: line capacity and link buffers. The cost $C_i(k)$

is set to infinity if either the link is carryng its maximum permissable number of virtual circuits or it has exeeded a preset threshold of buffer occupancy. Otherwise $C_i(k)$ is defined as a picewise constant increasing function of avarage link flow, quantized to a small number of levels and including a "hysterresis" effect. A typical function is shown on Figure 3, with the arrows indicating the way link cost changes as a function of changing utilization. The nodes send updated values of their $C_i(k)$'s to the NRC whenever a change occurs.

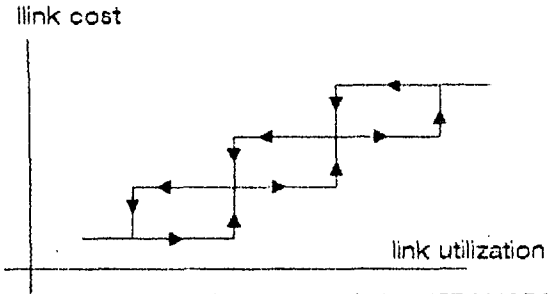


Figure 3. A typical link cost relation (TRANSPAC)

The major part of the routing computation takes place at the NRC, but some local information is used at each node. Let $C(i,j)$ (computed by the NRC) be the total cost associated with the minimum cost path between nodes i and j. Every node s determins the route to node t by choosing the value of i which minimizes $C(i,t)+\text{Max}[C(i),C_s(i)]$, i = all nodes conected to s. In this way the node s chooses the intermediate node that would have been choosen by the NRC, unless the value of $C_s(i)$ has changed recently. Ties are resolved by giving priority to the shortest hop path. Becouse of the way in which link costs are defined, the routing procedure becomes a minimum hop method upon which is superimposed a bias derived from the level of link resource utilization.

Althow the TRANSPAC routing algorithm has many of the features of a typical centralized routing procedure, its operation is not puerly centralized by allowing the final routing decision to be made locally, based on a combination of centrally and locally determined information.

5. OPTIMAL ROUTING APROACH

The algorithms described in Section 4 are generally reffered as grredy algorithms. Switches attempt to use the best resources (the minimum delay path) without regard to other delays in the network. On the other side the optimal routing approach solves the routing problem by minimizing the overall delay of all the messages in the network. The difference between "user optimization" in the former approach and "system optimization" in the latter is evident.

When the statistics of the traffic entering the network do not change over time (quasistatic enviroment) it is convinient to use flow models to describe the behaviour of the network. Traffic congestion can be quantified in terms of the statistics of the arrival processes of the network queues. These statistics determine the distributions of queue lenght and packet delay at each link. Unfortunately, there is usually not accurate analytical expression for the means or variances of the queue lenghts in a data network.

Some imperfect alternative is to measure congestion at a link in terms of the avarage traffic carried by the link. More precisely, we assume that the statistics of the arrival process at each link (i,k) change only due to routing updates, and that we measure congestion on (i,k) via the traffic arrival rate f_{ik}. We call f_{ik} the flow of link (i,k), and we express it in data units/sec, where the data units are the same units used for the maesurement of the transsmition capacity of the link, C_{ik}.

The optimal routing problem reduces to minimizing (or maximizing) some objective function $D(f_{ik})$ with respect to the variables f_{ik}, subject to certain constraints such as flow conservation, nonnegativity of flows etc.

Gallager, [11], has defined a recursive algorithm for datagram networks to solve the distributed routing problem in a quasistatic enviroment. We first define some notation in order to explain the main features of the algorithm:

The ordered pair (i,k) denotes the directed link from node i to node k.

$r_i(j)$ = expected traffic entering the network at node i and destined to node j in packets/sec

$t_i(j)$ = total expected traffic at node i destined for node j in packets/sec

f_{ik} = expected traffic over link (i,k) in packets/sec, i. e. flow of link (i,k)

$\phi_{ik}(j)$ = fraction of traffic $t_i(j)$ that is routed over link (i,k); also referred as the routing varrible for link (i,k).

The flow model in the network is described by the folowing equations:

$$t_i(j) = r(j) + \sum_l t_l(j)\, \phi_{li}(j) \quad \text{all } i,j$$

$$f_{ik} = \sum_j t_i(j)\, \phi_{ik}(j) \ .$$

The algorithm minimizes

$$D_T = \sum_{i,k} D_{ik}(f_{ik}) \ .$$

$D_{ik}(f_{ik})$ is the avarage number of packets in queue or under transmission at link (i,k). By Little's law D_T is proportional to the mean delay of packets in the network, so that minimizing D_T is equivalent to minimizing the mean packet delay. It is shown in [11] that, in order to minimize D_T, each node i must incremantally decrease those routing variables $\phi_{ik}(j)$ for which the marginal delay is large, and increase those for which it is small. The marginal delay is defined as:

$$D'_{ik}(f_{ik}) + \partial D_T/\partial r_k(j)$$

The algorithm obtained is distributed in nature and is intended for quasi-static applications where the input statistics are slowly changing with time and where occasionally links or nodes fail or are added to the network. Under these conditions, the loop freedom of the algorithm is maintained since this is a mathematical property that is independent of the marginal link delays and node flows, which are the only inputs to the algorithm.

Galagher's algorithm needs as a basic quantity the incremental delay dD_{ik}/df_{ik}. One way to find it is by using Kleinrock's formula

$$D_{ik}(f_{ik}) = f_{ik}/(C_{ik} - f_{ik}) \ .$$

This expression involves the assumptions of Poison arrivals at nodes, exponentially distributed packet lenghts, and the "independence assumption" of service time at sucessive nodes (i. e. when a packet arrives at a node, a new lenght is assigned to it from some common exponential distribution). Such assumptions do not hold in practice. It is therefore preferable to estimate the derivative of the delay directly by using data avaliable in the network.

A few techniques have been proposed for on-line estimation of performance gradients of systems modeled as queuing networks. One of them is the "customer rejection" algorithm proposed by Segall, [12]. In this approach, the delay sensitivity is estimated along an observed sample path by assuming that packets arriving at a link are rejected with probability ε. The effective link flow f is thus reduced on the avarage by

$$\delta f = M\varepsilon/T \ .$$

M is the number of observed arrivals in the interval T. Let c_m^n be the amount of system time that the m-th packet would save if the n-th packet were to be removed. Then, over B busy periods, containing each $[N_i,\ i=1, \ldots, B]$ packets, the quantity $\delta D/\delta f$ will be

$$\delta D/\delta f = \sum_{i=1}^{B} \sum_{n=1}^{N_i} \sum_{m=n}^{N_i} c_m^n \Big/ \Big(\sum_{i=1}^{B} N_i \Big)$$

Another procedure for estimating on-line marginal packet delays trough links is a tehnique known as perturbation analyses, [13]. This method is based on the assumption that a sample path of the system is observed. Hence, for the observed link i the queuing and transmission delay r_{ik} of the k-th packet could be recorded. Than the mean is obtained as

$$R_{ik} = (1/K) \sum_{k=1}^{K} r_{ik} \ .$$

The corresponding sensitivity is computed as

$$\delta R_{ik} = (1/K) \sum_{k=1}^{K} \delta r_{ik} \ .$$

Since by Little's law

$$D_{ik} = f_{ik} R_{ik} \ ,$$

than

$$D_{ik}' = \partial D_{ik}/\partial f_{ik} = R_{ik} + f_{ik}(\partial R_{ik}/\partial f_{ik}) \ .$$

However, in the prturbation analyses technique the gradient estimates are obtained in terms of the mean interarrival time $\tau_{ik} = 1/f_{ik}$, instead in terms of the link flow f_{ik}. Then the estimate of D_{ik}' is

$$D_{ik}' = R_{ik} - (\tau_{ik}/\delta\tau_{ik})\delta R_{ik} \ .$$

There also exists a class of algorithms for solving the optimal routing problem which could be viewed as constrained version of common, unconstained optimization methods, such as steepest descend and Newton's method, described in nonlinear programming texts, [14].

It is evident that the optimal routing approach is a more sofisticated alternative to the shortest path routing in the quasi-static network enviroment. As the statistics of the input arrival processes change more rapidly, the aproprietness of the optimal routing algorithms diminishes. In such cases it is difficult to recomend routing methods that are simultaniously effitient and practical.

6. ROUTING ALGORITHMS AND THE FLOW CONTROL

There are two main performance measures that are substantually affected by the routing algorithm: throughput (quantity of service), and average packet delay (quality of service). Routing interacts with flow control in determining these performance measures by means of a feedback mechanism shown in Figure 4. When the offered load is relatively low, it will be fully accepted into the network, i. e.

Troughput = Offered load .

When the offered load is excessive, a portion will be rejected by the flow control algorithm and

Throughput = Offered load - Lost load .

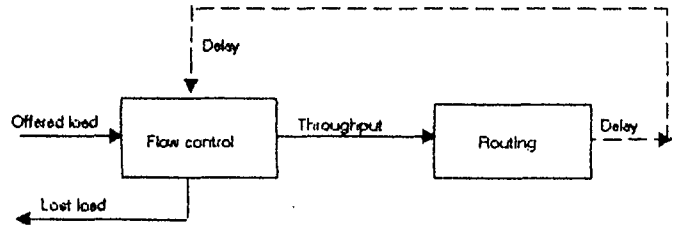


Figure 4 Interaction of routing and flow control.

The average delay per packet will depend on the routes choosen by the routing algorithm for all the traffic accepted into the network. However, throughput will also be greatly affected by the routing algorithm because typical flow control shemes operate on the basis of rejecting the offerd load when delay becomes exessive. Therefore, there is a feedback interaction between the two processes.

Unfortunately routing and flow control procedures have traditionally been developed independently in packet networks, under the assumption that flow control must keep excess traffic out of the network, and routing must struggle to efficiently transport to destination whatever traffic is permitted into the network by the flow control sheme. It seems, however, that routing and flow control can be brought together into usefull cooperation, especially in virtual circuit networks, where a path must be selected before data transfer for a particular session begins. In this case routing algorithm could be invoked first to determine weather a path of sufficial residual bandwith exists. If no path is avaliable, the virtual circuit connection shoud be blocked immediately at the entry node, thus preventing congestion rather then allowing to occure and than attempting to recover from it.

7. CONCLUSIONS

Routing is a sofisticated data network function that requres coordination between network nodes. It affects the avarage packet delay, and the network troughput and it is indirectly responsible for the amount of rejected traffic in the network. The common approach, the shortest path routing is implemented in the most of the networks. However, each network defines "lenght" or "cost" of a communication link differently. Some use centralized computation, some decentralized, and some use the hybrid of two. Adaptivity ranges from the bare minimum necessary to react to line failures to more sofisticated procedures sensing and responding to queuing delays and line loading. For networks with stationary input statistics an alternative procedure is optimal routing based on flow models. The close connection of the routing algorithm with the flow control scheme opens a posibiliity for development of a common routing and flow control procedure. One can conclude from this survey that while the routing function is central to the smooth and efficient operation of packet-switched networks, no single algorithm can be quolified as being the "best" one. The complexity of networks, especialy in regard to the routing functions, suggests a myriad of alternatives on all levels which motivates further research.

References:

[1] Bertsekas D. and Gallager R. (1987) Data Networks. Englewood Cliffs, NJ: Prentice-Hall

[2] Green P. E. (1980) An Introduction to Network Architectures and Protocols. IEEE Trans. Commun. COM-28:413-424.

[3] Schwartz M. and Stern T. E. (1980) Routing Techniques Used in Computer Communication Networks. IEEE Trans. Commun. COM-28:539- 552

[4] Cantor D. G. and Gerla M. (1974) Optimal Routing in a Packet Switched Computer Network. IEEE Trans. Comput. ,vol. C-23:1062-1069

[5] McQuillan J. M. , Richer I. and Rosen E. C. (1980) The New Routing Algorithm for the ARPANET. IEEE Trans. Commun. COM-28:711-719.

[6] Schoute C. F. and McQuillan J. M. (1978) A Comparison of Information Policies for Minimum Delay Routing Algorithms. IEEE Trans. Commun. COM-26:1266-1271

[7] Dijkstra E. (1959) A Note on Two Problems in Connection with Graphs. Numer. Math. vol. 1:269-271

[8] Glazer D. W. Tropper C. (1990) A New Metric for Dynamic Routing Algorithms. IEEE Trans. Commun. COM-38:360-367

[9] Thymes L. R. W.(1981) Routing and Flow Control in Tymnet. IEEE Trans. Commun. COM-29:298-392

[10] Rudin H. (1976) On Routing and "Delta Routing": A taxonomy and Performance Comparison of Techniques for Packet-Switched Networks. IEEE Trans. Commun. COM-24:43-59

[11] Gallager R. G. (1977) A Minimum Delay Routing Algorithm Using Distributed Computation. IEEE Trans. Commun. COM-25:73-85

[12] Segall A. (1977) The Modeling of Adaptive Routing in Data- Communication Networks. IEEE Trans. Commun. COM-25:85-94

[13] Cassandras C. G., Abidi M. V. and Towsley D. (1990) Distributed Routing with On-Line Marginal Delay Estimation. IEEE Trans. Commun. COM-38:348-359

[14] Luenberger D. G. (1984) Linear and Nonlinear Programming. Reading MA: Addison-Wesley.

PARALLEL IMPLEMENTATION OF VLSI HED CIRCUIT SIMULATION

INFORMATICA 2/91

Srilata Raman
University of Iowa, Iowa City, U.S.A.
Lalit Mohar Patnaik
Indian Institute of Science, Bangalore, India
Jurij Šilc
Marjan Špegel
Jožef Stefan Institute, Ljubljana, Slovenia

Keywords: circuit simulation, direct method, waveform relaxation, parallel algorithm, parallel computer architecture

The importance of circuit simulation in the design of VLSI circuits has channelised research work in the direction of finding methods to speedup the highly compute-intensive problem of circuit simulation. Attempts have been made to find better algorithms and to use parallel architectures to accelerate the simulation task. This paper deals with the two well-known circuit simulation algorithms – direct methods and relaxation method. The issues involved in parallelizing these algorithms and various computer architectures that have been reported in the literature are presented in this paper.

IZVEDBA VZPOREDNE SIMULACIJE VLSI VEZIJ – Potreba po simulaciji VLSI vezij pri njihovem snovanju je usmerila raziskovalno delo v iskanje metod za pohitritev računsko intenzivenega postopka simulacije vezij. Predmet raziskav je iskanje učinkovitejših algoritmov ter uporaba vzporednih arhitektur za izvajanje simulacije. V članku sta obravnavana dva dobro znana algoritma za simulacijo vezij: direktna in relaksacijska metoda. Prikazane so vzporedne različice omenjenih algoritmov ter podan pregled računalniških arhitektur, ki so namenjene izvedbi vzporedne simulacije VLSI vezij.

1 Introduction

The complexity of VLSI circuits is growing with the improvement in manufacturing methods and advent of new technologies. This has manifested itself in the increasing number of devices on a single chip. Design verification of VLSI chips has become indispensable to ensure that the circuit meets its requirements. The simulation of integrated circuit (IC) chips at the electrical level constitutes the most important design verification step. However, the dramatic increase in the complexity of ICs has burdened the capabilities of traditional circuit simulators like SPICE2 [Nag75]. Gate-level logic simulators [ST75] and switch-level simulators [HHL82] can verify circuit functions and provide first order timing information more than three orders of magnitude faster than detailed circuit simulator. However, it is necessary to perform accurate electrical simulation to verify circuit performance for critical path, memory design, and analog circuit blocks, and to detect dc circuit problems such as noise margin errors or incorrect logic threshold. Once of the most common analyses performed by circuit simulators and most expensive in terms of computer time is nonlinear, time-domain transient analysis of electrical circuits. This analysis provides precise electrical waveform information if device models and parasitics of the circuit are characterized accurately. Traditional circuit simulators like SPICE required excessive CPU time to generate voltage and current waveforms for circuits containing more than a few hundred transistors. As an example, a 700 MOSFET circuit analyzed for 4 μs of the simulated time with an average 2 ns time step, takes approximately 4 CPU hours on a VAX 11/780 VMS computer with floating-point accelerator hardware using SPICE2.

This situation has spurred vigorous research aimed at reducing the cost of circuit simulation. A number of approaches have been used to overcome the drawbacks of conventional circuit simulators. The time required to evaluate complex device models has been reduced using the table look-up models [CGK75]. Special-purpose microcode has been used for reducing the time required to solve linear systems and node tearing techniques have been used to exploit circuit latency by bypassing

the solution of the subcircuits whose states are not changing. In addition high performance computers with vector processing capabilities like CRAY [CA79] have been used to exploit parallelism and pipelining available in the circuit simulation program. But, circuit simulation programs are not well suited to such computers. The reason is that as the circuit matrix is sparse and has an irregular structure, the data gather-scatter time dominates the overall program execution time [CA79]. This means that fetching the data stored in memory and writing it back after it has been processed poses a bottleneck. All the above approaches have been found to result in an order of magnitude speedup over SPICE.

A recent development is the use of relaxation methods [NSV84] for solving the set of ordinary differential equations describing the circuit under analysis rather than using the direct sparse matrix methods on which standard circuit simulators are based. Simulators using this method have been shown to provide guaranteed accuracy [NSV84] with upto two orders of magnitude speed improvement for large circuits [LRSV82]. This new method has been found to offer much greater speedups on special-purpose hardware designed to exploit the particular features of the relaxation algorithms [DN84]. In the following sections the direct and relaxation-based algorithms and issues in their parallel implementation are presented.

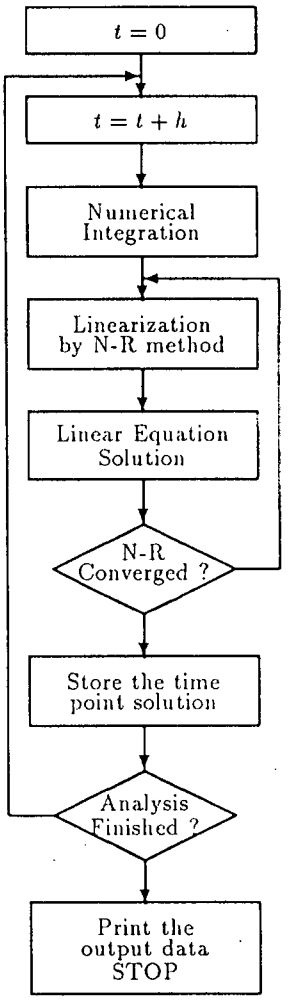


Figure 1: Flowchart of Direct Method of Circuit Simulation.

2 Direct-method for Circuit Simulation

The most common approach to solving the circuit equations in time-domain analysis employs three basic numerical methods [NSV84]: an implicit integration method, the Newton–Raphson (N-R) method and sparse Gaussian elimination method. These tree methods constitute the standard method of circuit simulation on which conventional circuit simulators like SPICE [Nag75] and ASTAP [WJM+73] are based. The analysis portion of a circuit simulation program determines the numerical solution of a mathematical representation of the circuit. The mathematical system of equations for physical circuit is obtained by representing each element in the circuit by its mathematical model. The system of equations describing the complete circuit is given by the model equations and the Kirchoff's current and voltage laws applied to the interconnection of the circuit elements. As a result algebraic-differential equations of the form,

$$F(\dot{\mathbf{x}}, \mathbf{x}, t) = 0 \qquad (1)$$

are obtained. Here, $\mathbf{x} \in \mathbb{R}^N$ is the vector unknown circuit variables, $\dot{\mathbf{x}} \in \mathbb{R}^N$ is the time derivative of $\mathbf{x}$ and F is a nonlinear operator.

Transient analysis determines the time-domain response of the circuit over a specified time-interval $(0, T)$. The flowchart of the standard circuit simulation method is shown in Fig.1.

The entire simulation time interval is divided into a number of discrete time points $(0, t_1, t_2 \ldots, t_n, t_{n+1} \ldots, T)$. $\mathbf{x}^n$, the information from the previous time point is used to predict the solution $\mathbf{x}^{n+1}$

at t_{n+1}. A stiffly stable integration formula like Backward-Euler (BE) with variable time is used to descretize the nodal equation to yield a set of nonlinear, algebraic equations of the form

$$g(\mathbf{x}) = 0 \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the vector of unknown variables at time t_{n+1}. The above equations are solved using N-R algorithm to yield a set of sparse linear equations of the form,

$$\mathbf{Ax} = \mathbf{b} \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a matrix related to the Jacobian of g and $\mathbf{b} \in \mathbb{R}^N$. These equations are solved using direct methods like Gaussian Elimination (GE) or sparse LU decomposition.

The major computation in circuit simulation lies in formulating and solving the system of linear algebraic equations simultaneously. It has been shown in [NP78] that the storage and computer time required by circuit simulation increase rapidly with the size of the circuit, measured in terms of the number of circuit components. Thus for transient analysis, the standard circuit simulators are cost-effective only when the circuit size is limited to a few hundred devices. VLSI circuits with over 10.000 devices impose severe strain on standard circuit simulators. This has necessitated development of alternative circuit simulators. The standard circuit simulators have yet another drawback. For most circuits, the fraction of nodes that change their voltage values at a given point in time decreases as the circuit size increases. So only the circuit equations representing the active nodes need to be solved at any time, bypassing the solution of equations of the nodes which are not active at the time instant. Circuit simulators must exploit this time sparsity or latency because the computational complexity of the GE method applied to an $N \times N$ dense matrix is proportional to $O(N^3)$ whereas the computational complexity of the GE method for sparse matrices is proportional to $O(N^a)$, $1.2 \leq a \leq 1.5$. The performance of standard circuit simulators is compromised for large circuits because they solve the set of equations describing the entire circuit simultaneously irrespective of whether a given node is active or not. Relaxation-based methods help in overcoming these drawbacks.

3 Relaxation Algorithm

The basic concepts of relaxation-based algorithm have been described in great detail in [NSV84]. Relaxation methods can be used with a variety of IC technologies though they are particularly suited to the analysis of large MOS digital ICs. Relaxation-based circuit simulators make an important assumption that a two terminal capacitor is connected from each node of the circuit to the reference node. This assumption is satisfied by the circuits where parasitic capacitances are present between circuit interconnect and ground or the terminals of active circuit elements. Under this assumption, the nodal equations of a circuit are given by,

$$\mathbf{C}(\mathbf{v}(t), \mathbf{u}(t))\dot{\mathbf{v}}(t) = -\mathbf{q}(\mathbf{v}(t), \mathbf{u}(t)), \tag{4}$$

$$\mathbf{v}(0) = \mathbf{V} \tag{5}$$

for $0 \leq t \leq T$ where, $\mathbf{v}(t) \in \mathbb{R}^n$ is vector of node voltages at time t, $\mathbf{V}$ is the given initial values of $\mathbf{v}$, $\dot{\mathbf{v}}(t) \in \mathbb{R}^n$ is vector of time derivatives of $\mathbf{v}(t)$, $\mathbf{u}(t) \in \mathbb{R}^n$ is input vector at time t, $\mathbf{C} : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ is nodal capacitance matrix, $\mathbf{q} : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, and $\mathbf{q}(\mathbf{v}(t), \mathbf{u}(t)) = [q_1(\mathbf{v}(t), \mathbf{u}(t)), \ldots, q_n(\mathbf{v}(t), \mathbf{u}(t))]^T$, where q_i is sum of currents charging the capacitors connected to node i.

The two common relaxation methods used are the Gauss-Seidel (GS) and the Gauss-Jacobi (GJ) method. Relaxation methods can be used for the solution of equations (4) in different ways. Fig.2 illustrates the levels at which relaxation methods can be applied. Linear relaxation method is applied at the linear equation level and consists of replacing the GE method for solving equation (3) by GJ or GS. Nonlinear relaxation methods are applied at the nonlinear equation level and augment the N-R method applied to equation (2). They replace the linear equation solution based on sparse-matrix techniques. Relaxation methods when applied directly to the system of nonlinear algebraic equations describing the circuit are termed Waveform Relaxation (WR) [NSV84]. As a result of this, the system is decomposed into decoupled subsystems of algebraic-differential equations corresponding to decoupled dynamical subcircuits. Each decoupled subcircuit is then analyzed for the entire simulation time interval using the standard simulation techniques. Such a decomposition permits latency to be exploited. Decomposition into subcircuits permits a trade-off between the amount of time spent

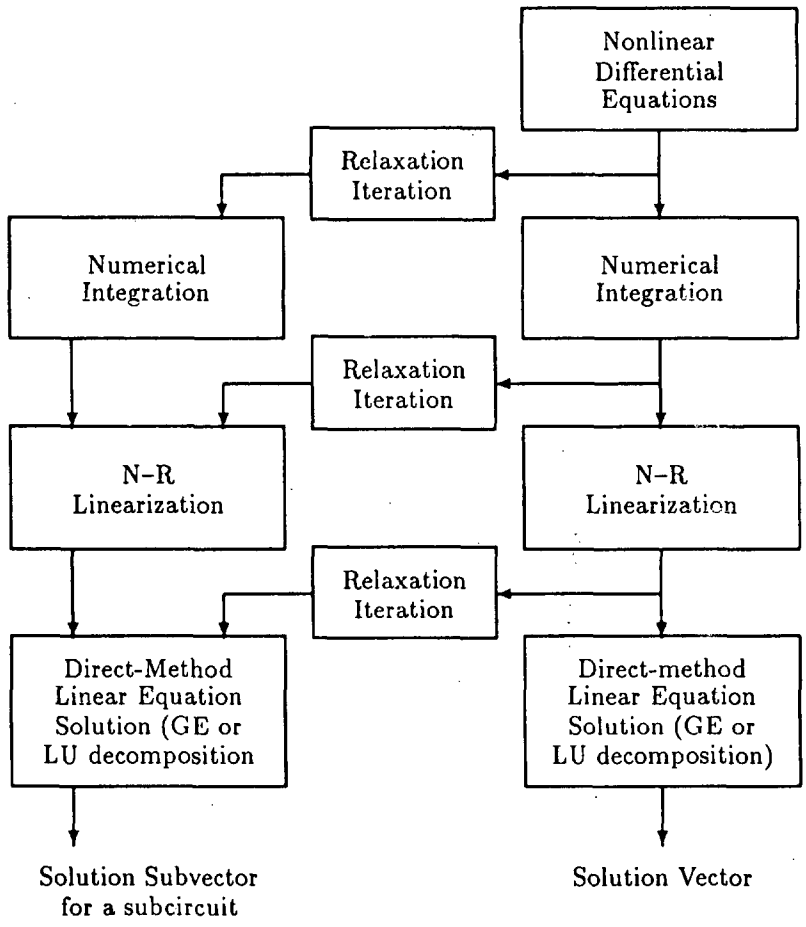


Figure 2: Relaxation Applied at Different Levels of Analisis.

on any single processor at an iteration and the time spent communicating the results of the analysis. This is a key requirement for deriving maximum efficiency from a parallel processing environment.

The WR algorithm using GS iteration is given in Fig.3. Superscript k is the iteration count, subscript i is the component index of a vector, ε is a small positive number, N is the maximum node number and n is the number of circuit variables.

Modifications in the WR algorithm help in improving the speed of convergence. Instead of solving each differential equation for one unknown (point relaxation), the system of differential equations can be partitioned into subsystems having more than one equation (block relaxation). Each decomposed circuit is then solved using standard simulation techniques. Each subcircuit can be analyzed independently [NSV84] from $t = 0$ to $t = T$, using its own time step sequence, controlled by the integration method. As opposed to this, in a standard circuit simulator entire circuit is analyzed over the total simulation time using only one common time step sequence. The number of time step for each subcircuit is thus less in WR decomposition which is a definite computational advantage. Latency of the circuit can be exploited by incorporating bypass techniques. Without losing accuracy the analysis of subcircuits is bypassed for some time intervals knowing the information obtained from the previous time point or previous iteration. The bypass techniques have been described in [NSV84] in detail.

WR methods have guaranteed convergence and have proven to be effective decomposition methods for the analysis of large scale MOS circuits. However, they do suffer from a few drawbacks. The

$k \leftarrow 0$;

guess waveform $\mathbf{v}^0(t) \quad \forall t \in [0,T]$ such that $\mathbf{v}^0(0) = \mathbf{V}$;

repeat

$k \leftarrow k+1$;

for each i solve

$$\sum_{j=1}^{i} C_{i,j}(v_1^k, \ldots, v_i^k, v_{i+1}^{k-1}, \ldots, v_N^{k-1}, u)\ \dot{v}_j^k +$$

$$\sum_{j=i+1}^{n} C_{i,j}(v_1^k, \ldots, v_i^k, v_{i+1}^{k-1}, \ldots, v_N^{k-1}, u)\ \dot{v}_j^{k-1} +$$

$$q_i(v_1^k, \ldots, v_i^k, v_{i+1}^{k-1}, \ldots, v_N^{k-1}, u) = 0$$

for $v_i^k(t)$, $t \in [0,T]$ with initial condition $v_i^k(0) = V_i$

until $\max_{1 \le i \le n} \max_{t \in [0,T]} \mid v_i^k(t) - v_i^{k-1}(t) \mid < \varepsilon$

Figure 3: Algorithm WR–GS.

waveforms of the unknowns at the current iteration have to be stored for computing the waveforms at the next iteration. For large circuits the amount of storage required can be very large. Another problem crops up when there is a logic feedback between the decomposed subcircuits. The speed of convergence of the WR algorithm in such a case becomes very slow unless a good initial guess for the unknown variables is provided. The problem of storage in WR methods can be overcome by dividing the simulation time interval into "windows", $[0,T_1],[T_1,T_2],\ldots[T_{n-1},T_n]$. WR is applied to the first window, $[0,T_1]$ and the values of the node voltages at T_1 are used as the initial conditions for the analysis of the second window. This procedure is repeated until all the windows have been analyzed. This approach helps in rapid convergence.

An obvious advantage of the relaxation algorithms is that they are amenable to parallel implementation. The solution of each node is effectively decoupled from the others and it is possible to allocate a separate processor for each decoupled node equation. If the circuit has been partitioned into subcircuits, they can be analyzed concurrently on different processors. Special purpose hardware can be designed to suit the algorithm. With this brief introduction to the circuit simulation methods, the next section is a survey of the attempts made to speedup circuit simulation using different parallel architectures

4 Need for Parallel Processing

Parallel processing is a technological imperative of computation in VLSI CAD. The economics of VLSI fabrication ensures that parallel computing systems will dominate serial systems in both absolute performance cost. Processing speed is a major concern in circuit simulation. As multiprocessors serve to decrease the program runtime, parallel processing for circuit simulation is a natural evolutionary step. Attempts have been made to implement both the direct and relaxation methods for circuit simulation on parallel architectures. Before such an implementation is attempted, it is necessary to ensure that the algorithm maps well on the architecture to derive the best utilization of the processors. Pipelined architectures [WW86], bus-based architectures [JNP86], hypercube [Mat86], crossbar and multistage switch networks [JNP86] have been used in the past for circuit simulation. The nature of the algorithm shows that relaxation methods are more amenable to parallel implementation that direct methods. This follows because iterative methods have decoupling inherent in them. However, relaxation-based simulators like SPLICE and RELAX have been found to be inappropriate for tightly-coupled circuits due to the convergence

problems encountered in such circuits.

Relaxation algorithms have conflicting requirements when implemented to exploit parallelism. These simulators solve the subcircuits in parallel. Partitioning the circuit to form subcircuits has to be done judiciously so that within a subcircuit tight coupling exists. This ensures that few iterations are required to converge to a solution. However, putting the tightly-coupled nodes in one sub-circuit might lead to a small number of large subcircuits. This situation however, limits the extent of parallelism available. In contrast large number of circuits with few nodes per sub-circuit result in increased parallelism but at the same time number of iterations required for a solution increases as the tightly-coupled nodes may not be together in the same sub-circuit. A balance has to be struck between the number of subcircuits and the number of iterations. Each of the subcircuits is solved independently in parallel on a processor using the direct method.

As even the relaxation algorithm uses direct methods for solving the subcircuits efforts have been directed towards parallelizing direct methods in addition to the relaxation methods. As elucidated in an earlier section, the direct method of circuit simulation involves integrating the set of nonlinear ordinary differential equations modeling the circuit based on the fastest changing circuit variable (this is the variable that requires smallest number of time step for convergence). The time step for the direct method is governed by this variable. The determination of the fastens changing variable can be done in parallel by allocating a set of nodes to each processor to compute the time step [JNP86] for each node and hence the time step for the direct method analysis. The resulting set of nonlinear algebraic equations is solved using N-R iterative technique. Each iteration involves finding the linear equivalent circuit for all nonlinear elements. This again can be done concurrently on a number of processors. The set of sparse linear equations so obtained is decomposed into subcircuits and these subcircuits are solved in parallel using GE method.

Direct method has been implemented on the bus-based Sequent Balance computer, Omega network, cross-bar switch, and BBN Butterfly [JNP86]. Circuit simulation on circuits with upto 50 nodes has been shown by *Jacob et al.* in [JNP86] to have an efficiency of nearly 45% with upto 8 processors. Large circuits are expected to yield better performance because the amount of time spent in contention for shared memory will reduce. It is observed as reported in [JNP86] that as the number of processors increases to the thousands, the algorithm that works on smaller multiprocessors break down due to contention for various system resources. The approach that has been suggested by *Jacob et al.* in [JNP86] is to use clusters of processors to solve smaller parts of the problem (that is, the subcircuits) and then solve between the clusters for the solution of the overall circuit. A hierarchical arrangement of multiprocessors which will grow in a regular fashion to simulate progressively larger circuits has been proposed in [JNP86].

The problem of long runtimes required by SPICE has brought about vectorization of circuit simulation using supercomputers as reported in [MITM87]. The two most time consuming parts of SPICE, namely, computation of the circuit matrix elements and solution of sparse linear equations have been vectorized. *Mikami et al.* have shown [MITM87] that if the vectorized solution of linear equations is ten times faster than its scalar version, it results in a simulator that is 1.1 times faster that its scalar version [MITM87]. The computation time of SPICE based circuit simulation is found to be reduced to one-eighth of the scalar computation time.

The direct method implemented on a SIMD architecture has been found to result in a speedup between 5–7.7 for small and medium sized circuits as shown by *Vladimirescu et al.* in [Vla87]. SIMD avoid problems related to synchronization of different processors. The task of evaluating the models for circuit devices like MOSFETs, and bipolar junction transistors is parallelized. However, the update of the Jacobian matrix and linear equation are implemented as sequential processes only.

The potential of the relaxation methods for parallel processing has motivated implementation of the method on pipelined and multiprocessor architectures. The natural circuit decomposition available in relaxation techniques can be exploited for its parallel implementation. The use of GJ iterative methods provides ample parallelism to the relaxation algorithm. In this method, the relaxation algorithm makes use of the waveforms computed at

the previous iteration for all the subcircuits. All the subcircuits can then be analyzed independently by different processors. The drawback of GJ method is that it is slow in convergence.

The major problem encountered in parallelizing the WR algorithm is that MOS digital circuits are highly directional. It is important to follow the directionality when performing the relaxation computation, otherwise the WR method becomes inefficient. Many iterations are required for convergence if the computation does not follow the signal flow. In the case of large digital circuits, the output of gates have usually more than one fan out and so it is possible to order the computation so that subcircuit can be computed in parallel, but the directionality of the circuit can still be followed by the relaxation computation. Though this limits the parallelism available, it preserves the efficiency of the method.

It is possible to parallelize the WR algorithm while preserving a strict ordering of computation of the subcircuit waveforms by pipelining the waveform computation. In [WW86] *White and Weiner* adopt an approach where the circuit is divided into a number of subcircuits. The first processor starts computing the transient response of a subcircuit for one time point. After the computation corresponding to the first time point is over, the second processor starts computing the response for the first time point for the second subcircuit. At next step, a third processor starts computations for the first time point for the third subcircuit and so on. This is an instance of time point pipelining and has been implemented on a Sequent Balance 8000 computer with a single bus shared memory system as elucidated in [WW86]. The timepoint pipelining algorithm makes efficient use of the available processors. *White et al.* have observed that this algorithm running on the Balance 8000 runs substantially faster than the serial WR algorithm running on a VAX/780 [Whi85]. Thus, an EPROM with 348 FETs take 212 s on VAX/780 whereas pipelined implementation with 9 processors takes 182 s.

Some amount of parallelism can be achieved by using GS iterative method also though it is sequential in nature. *Saleh* has described the implementation of the WR algorithm using GS iterations on different number of processors in [Sal87]. The circuit is presented as a graph with directed edges. The nodes represent the subcircuits and the edges represent the connection between the subcircuits. The directed edges between the nodes give the precedence relation between the tasks. The width of the graph is the maximum size of any independent subset of tasks and these tasks can be solved in parallel. So, all subcircuits at the same level in the graph are computed in parallel for the same iteration. This approach has been found to be good for circuits having wide graphs. Reasonable speedup is possible over the uniprocessor version if the circuit is large enough and only a small number of processors are available. This method proves to be ineffective on circuits with narrow graphs and does not give significant advantages over the GJ method.

CONCISE, a GJ relaxation-based circuit simulator, implemented on a hypercube by *Mattisson* [Mat86] promises to give a very good performance for circuit simulation which takes large fraction of the computing cycles on many high performance computers. In [Mat86] a point relaxation has been used. The circuit simulation program is mapped onto the cube by partitioning the Jacobian matrix **A** into concurrent processes. The linear equation solution phase, that is, Jacobi iteration, involves considerable communication between the processors. The N-R linearization step is completely decoupled and so is concurrently executed. The performance of CONCISE program for different test circuits and implementation details are given in [Mat86].

The hypercube in [Mat86] is, however, modeled by a concurrent program that does not take into account architectural features of a hypercube like the message passing scheme of communication. We have come up with an implementation of the WR algorithm for circuit simulation on a hypercube in HIRECS [Ram88]. Where as CONCISE in [Mat86] makes use of point relaxation, HIRECS is based on block relaxation. A novel circuit partitioning approach based on the heuristic method of Simulated Annealing has been used in HIRECS. An additional feature of HIRECS is that it models all the architectural features of the hypercube. HIRECS has been simulated using the programming language SIMULA on a DEC 1090 system. The speedup obtained from the simulation study for different test

Test Circuit	Number of Processors	Speedup	Efficiency
Inverter	2	1.8	90%
	4	3.1	77%
	8	5.9	73.7%
Multiplexer	2	1.85	92%
	4	3.6	90%
	8	6.0	75%

Table 1: Performance of HIRECS.

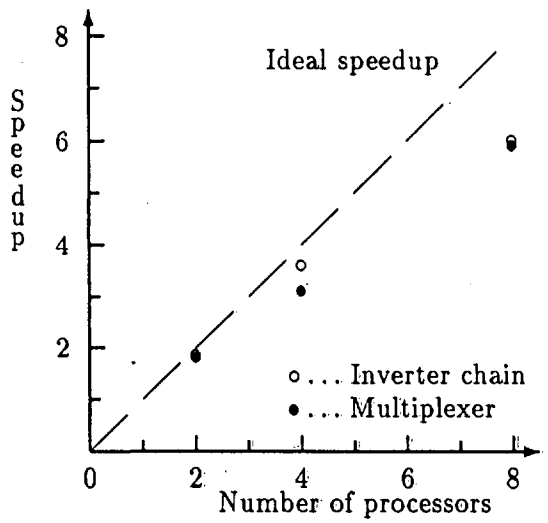


Figure 4: Speedup vs. Number of Processors.

circuits is presented in Table 1 and the performance curves are illustrated in Fig.4. The ideal speedup and the speedup actually obtained for the test circuits are shown in Fig.4. We have observed that for fewer number of processors, a near-linear speedup is possible. as the number of processors increases, the communication time of among the processors increases thereby reducing the speedup. This is because in HIRECS synchronization of the processors is carried out after all the variables of the subcircuits allocated to the processors have converged at all time points over window. Hence the time for which a processor waits till all the others have finished computation over the window also increases with the number of processors. Due to lack of circuit data, HIRECS could not be tested for circuits with large number of nodes. However, we except it to perform equally well for large circuit also.

Improvements in computer architecture allow large circuits to be run without any change in the simulation techniques. Circuit size continue to increase with the progress in technology and existing computer architectures are reaching their performance limits due to constraints on the fundamental speed of light. This has prompted development of an experimental relaxation-based circuit simulator on a massively parallel processor (MPP) – the Connection Machine reported by *Webber et al.* in [WSV87]. The Connection Machine is an MPP with upto 65,536 processors and uses SIMD architecture. The simulator in [WSV87] uses GJ iteration at the nonlinear equation level with point relaxation and a single step of N-R method. Though point relaxation causes slow convergence, it has been found to work well for large class of circuits. Block relaxation is not found to be suitable on the Connection Machine. This is because the data is less uniform in block methods. The matrices to be solved may be of different sizes which make it difficult to exploit the parallelism on the Connection Machine. For an EPROM circuit, the point relaxation on the Connection Machine has been experimentally found by *Webber et al.* [WSV87] to be 30 times faster than the direct method on a MicroVax. The results show that the execution is nearly independent of the size of the problem for circuits without tight coupling. Connection Machine is good for very large problems though it is extremely slow for small problems. The largest circuit that can be run on the Connection Machine is about 10,000 nodes.

5 Conclusion

In this paper we have surveyed the two well known methods for circuit simulation – direct and relaxation. The parallel implementation of these meth-

ods has been considered. The architectures used for the simulation problem as reported in the literature and the observations from our experiments have been presented. It follows from the discussion that considerably higher performance can be achieved by using a special-purpose multiprocessor in which the interconnection of the processors and the design of processors are turned to the circuit simulation task. This is particularly true for the relaxation-based algorithms. Present research work includes finding good partitioning schemes for dividing the circuit into tightly coupled subcircuits, investigation of optimal techniques for finding the simulation time steps and mapping the algorithms to the best possible hardware.

References

[CA79] D. A. Calahan and W. G. Ames. Vector Processors: Models and Application. *IEEE Trans. Circuits Syst.*, CAS-26(9), September 1979.

[CGK75] B. R. Chawla, H. K. Gummel, and P. Kozak. MOTIS – An MOS Timing Simulator. *IEEE Trans. Circuits Syst.*, CAS-22(12):901–909, December 1975.

[DN84] J. T. Deutsch and A. R. Newton. A Multiprocessor Implementation of Relaxation-Based Electrical Circuit Simulation. In *Proc. 21th Design Automation Conf.*, pages 350–357, 1984.

[HHL82] M. H. Heydemann, G. D. Hachtel, and M. Lightner. Implementation Issues in Multiple Delay Switch Level Simulation. In *Proc. Int'l Conf. on Circ. and Comp.*, pages 46–52, September 1982.

[JNP86] G. K. Jacob, A. R. Newton, and D. O. Pederson. Direct Method Circuit Simulation using Multiprocessors. In *IEEE Proc. ISCAS*, 1986.

[LRSV82] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli. The Waveform Relaxation Method for the Time-Domain Analysis of Large Scale Integrated Circuits. *IEEE Trans. Computer-Aided Design*, CAD-1(3):131–145, August 1982.

[Mat86] S. Mattison. CONSINE – A Simulation Program on Hypercube. Technical report, Lund University, 1986.

[MITM87] M. Mikami, J. Ishibashi, N. Tahara, and G. Matsuoka. Vectorization of SPICE Circuit Simulator on FACOM VP Series Supercomputer. In *Proc. 2nd Int'l Conf. on Supercomputing*, pages 29–34, 1987.

[Nag75] L. W. Nagel. SPICE2: A Computer Program to Simulate Semiconductor Circuits. Technical Report Memo.No.ERL-M 520, UCB, Berkeley, May 1975.

[NP78] A. R. Newton and D. O. Pederson. Analysis Time, Accuracy and Memory Requirement Tradeoffs in SPICE2. In *IEEE Proc. ISCAS*, pages 6–9, 1978.

[NSV84] A.R. Newton and A.L. Sangiovanni-Vincentelli. Relaxation-Based Electrical Simulation. *IEEE Trans. Computer-Aided Design*, CAD-3(4):308–331, October 1984.

[Ram88] S. Raman. HIRECS: Hypercube Implementation of Relaxation-Based Circuit Simulation. Master's thesis, Dept. of Computer Science and Automation, Indian Institute of Science, Bangalore, India, July 1988.

[Sal87] R. A. Saleh et al. Parallel Waveform Newton Algorithms for Circuit Simulation. In *Proc. ICCD*, pages 660–663, 1987.

[ST75] S. A. Szygenda and E. W. Thompson. Digital Logic Simulation in a Time-Based, Table-Driven Environment: Part 1 Design Verification. *IEEE Computer*, 7(3):24–36, March 1975.

[Vla87] A. Vladimirescu et al. A Vector Hardware Accelerator with Circuit Simulation Emphasis. In *Proc. 24th Design Automation Conf.*, pages 90–94, 1987.

[Whi85] J. White et al. Accelerating Relaxation Algorithms for Circuit Simulation using Waveform Newton, Iterative Step Size Refinement, and Parallel Techniques. In *Proc. ICCAD*, pages 5–7, 1985.

[WJM+73] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott. Algorithm for ASTAP – A Network Analysis Program. *IEEE Trans. Circuit Theory*, CT-20(11):624–628, November 1973.

[WSV87] D. M. Webber and A. L. Sangiovanni-Vincentelli. Circuit Simulation on the Connection Machine. In *Proc. 24th Design Automation Conf.*, pages 108–113, 1987.

[WW86] J. White and N. Weiner. Parallelizing Circuit Simulation – A Combined Algorithmic and Specialized Hardware Approach. In *Proc. ICCD*, pages 438–441, 1986.

THE PRINCIPLE OF MULTIPLE KNOWLEDGE

INFORMATICA 2/91

Keywords: knowledge representation, learning, classification

Matjaž Gams and Viljem Križman
Jožef Stefan Institute
Jamova 39, Ljubljana

ABSTRACT The principle of multiple knowledge is presented and it's implications are analyzed in real-life classification tasks. Empirical measurements, simulated computer models and analogy with humans strongly indicate that better classification accuracy is obtained when constructing and using multiple knowledge bases instead of one knowledge base alone.

POVZETEK V članku je predstavljen princip mnogoterega znanja in njegov pomen v realnih klasifikacijskih domenah. Empirične meritve, simulirani računalniški modeli in primerjava z ljudmi kažejo, da je v splošnem klasifikacijska točnost več baz znanja boljša kot najboljše baze izmed njih.

1 Introduction

Real-life domains are characterized by nonexistence of exact computational model and consequently, traditional computing approach is often inadequate. Expert systems enabled a significant step ahead yet different studies, e.g. (Keyes 89), report that they are successful only when the application is relatively small, simple, well understood and when experts agree with each other. When coping with more difficult problems, existing ES methodology lacks mechanisms for dealing with

- nonexistence of a single compact body of encodable knowledge,
- nonexistence of isolated static body of knowledge, independent of time, related events and environments,
- nonexistence of perfect exact single algorithm for applying the knowledge.

It is surprising that AI literature hardly mentions problems with multiple [1], contradictory and redundant knowledge with the purpose to increase performance by exploiting these properties. Recent keyword search of 50.000 abstracts from AI literature over 10 years (Clark 90) indicates that only a few refer to this problem. Indeed, computer methods regularly tend to use only one single body of knowledge in the form of one computer procedure.

People usually take quite different approach in real life. They tend to form expert groups, verify results independently and cross-check information in order to find the best solution. Practical experiences show that such approach in general produces better results than when relying on one man or one source of knowledge alone. In other words - people inherently and successfully use multiple knowledge in most of difficult tasks without paying much attention to that phenomenon.

2 Related work

Many well known AI systems like DENDRAL, MYCIN, PROSPECTOR and DIPMETER already enable the use of multiple knowledge to a certain degree. Multiple knowledge is also present in qualitative modeling (Murthy 88), e.g., when combining qualitative and quantitative models, and in the second generation expert systems.

Probably the most relevant work regarding multiple knowledge was reported in empirical learning or learning from examples. Different authors argue that a

[1] In this paper we don't distinguish between multiple methods, multiple systems, multiple knowledge or multiple knowledge bases.

proper use of multiple knowledge enables better possibilities to analyze the domain and better classification accuracy. Catlett & Jack (87) reported of important increase in classification accuracy in several domains when one knowledge base in the form of a decision tree was constructed for each class instead of classifying all classes with one tree. Similarly, Buntine (89) reported of important increase when 10 different decision trees were designed each time with different attribute at the root. Another interesting approach was taken by Schlimmer (87) enabling multiple view on the same domain by using different algorithms each with differently preprocessed input data. Brazdil & Torgo (90) used different data and different algorithms and achieved very significant improvements. Slightly different approach was taken in GINESYS (Gams 78; Gams, Drobnič & Petkovšek 91) where two methods were applied, one AI and one statistical, on the same data and combined together.

In the process of knowledge acquisition different views of experts sometimes result in construction of multiple knowledge bases (Boose et al 89; Clark 90). Other systems like BLIP (Emde 89) are capable of representing several competing hypotheses.

3 The principle of multiple knowledge

The principle of multiple knowledge is shown at the level of the basic definition of learning (Charniak & McDermott 85): Learning can be seen as modifying knowledge base according to the experience in the past for the use in the future. Performances of a knowledge base are measured on specific tasks to evaluate the successfulness of learning. The improvement over older definitions is based on the recognition that learning can be successful only when much is already known. This learning schema can be further modified according to the way how humans use multiple knowledge in real-life tasks:

	LEARNING		APPLYING	
old multiple knowledge bases	$\Longrightarrow$	modified multiple knowledge bases	$\Longrightarrow$	

On the basis of modification of the learning schema the principle of multiple knowledge (further on referred also as Principle) can be defined:

In order to achieve better performances it is generally better to construct and use multiple knowledge bases than one knowledge base alone, as long as they reasonably cooperate.

The emphasis in the definition of the Principle is on the word **reasonable**. Each multiple method should have as good performance as possible and methods should be multiple to a reasonable degree. The Principle should be understood in the following way: For a difficult real-life problem the user can in most cases obtain better results if instead of one method several methods are used together. Practical experiences and theoretical models described in the paper can provide some useful advice, however, the burden of finding intelligent combination of methods in a specific application is left to the user and is domain dependent.

Attempts to verify the advantages of multiple knowledge were performed in three areas, with:

- simulated models,
- empirical measurements on two real-life domains and
- an experiment on the Winston's arch problem.

4 Models of multiple knowledge

Multiple methods were simulated and analyzed with computer models simulating a general classification process. Typically, 10000 classification tasks were generated and average accuracy was measured as the percentage of correct predictions. Each method classified with a predefined average accuracy which was nearly always set to 0.50. By default, all methods were completely independent of each other in the sense that the probability of correct classification by any method was independent of predictions of all other methods. No actual description of the given task was given, rather, each method classified as a random generator with additional specifications.

Each method also estimated confidence factor of its classification. Confidence factor was a real number between 0 and 1 and was generated randomly with a predefined specifications. It didn't necessary represent probability, meaning there was no constraint on the sum of confidence factors of all predictions. The default value was 0.50. Classification of methods combined together was performed on the basis of confidence factors. There were two basic classification schemata:

- **best-one**, where the method with the best confidence factor was chosen for classification and
- **majority**, where all methods added their confidence factors for and against correct prediction.

Best-one schema classified correctly if a method with the best confidence factor classified correctly. Majority schema classified correctly if the sum of confidence factors for the correct prediction was bigger than the sum of confidence factors against it.

No additional information was used. For example, when having many classifications it could be possible to assume that the most probable class would be the one most commonly predicted. No such or other additional mechanism was embedded into models and in most cases parameters were deliberately set to lower levels than expected in real life. The idea is that if even such uninformed system produces better results it is even easier to reproduce the gains in real life.

Three parameters were varied:

- CONNECT - the percentage of cases where the confidence factor was verified to randomly fall into subinterval between 0.50 and 1.00 if the method predicted correctly and between 0.00 and 0.50 if the method failed.
- DECREASE - the decrease of average classification accuracy of each next method.
- DEPEND - the percentage of classifications of j-th method which were checked to classify the same as the first method.

The following is an example with CONNECT = 0.10, DECREASE = 0, DEPEND = 0, best-one voting schema, average 0.50 accuracy and 0.50 certainty factor. Methods are completely independent of each other. The relation between the number of methods and classification accuracy is observed. The meaning of rows is as follows: (1) index of the current method, (2) classification accuracy of the current method, (3) classification accuracy of 1..current method together, (4) confidence factor of correct predictions of the current method and (5) of 1..current method together.

Results of over 2000 similar tests with different combinations of parameters indicate that multiple knowledge can significantly improve overall classification accuracy under the following conditions: (a) certainty factor is positively correlated to the accuracy of each method, (b) classification accuracy of each multiple method is not much smaller then the best one, (c) methods are not too similar. The increase is especially noticeable when having a small number of multiple methods, e.g., 2-5.

5 Empirical measurements

Two real-life domains were chosen for benchmarking available systems. They represent descriptions of patients and their diagnoses mainly obtained by autopsy. Basic data: lymphography - 150 examples, 9 classes, 18 attributes; primary tumor - 339 examples, 22 classes, 17 attributes (Gams 89). Over 3 years around 20 AI and statistical systems were tested in several thousands of tests each time randomly dividing data into learning and test data and adding additional noise, varying the percentage of learning data etc. The best two methods were combined together in a multiple system GINESYS which achieved the best overall classification accuracy in more than 95 percentage of measurements, where one meaurement consisted of averaging classification accuracy over 10 tests.

Although the improvements of classification accuracy over the second the best system were not sufficient to be proven by the significance tests, they were typically at the level of 1% on the average. The chosen statistical measure was the T-test (Jamnik 87) which demands more or less permanent improvements in order to be fulfilled. In our measurements, the standard deviation was usually around 5%, meaning that the distributions of testing and learning data varied quite a lot. Therefore, while the improvements can not be statistically proven as significant when comparisons are made on the bases of one test, they happen nearly always when average over 10 tests is compared. For example, in (Gams 89), where these tests are described, from 164 averages there were only 3 cases where GINESYS had

1	2	3	4	5	6	7	8	9	10	20	30
50.1	50.1	50.2	50.0	50.1	50.0	49.7	50.2	50.3	50.5	50.1	50.0
50.1	52.3	53.6	54.0	54.5	54.2	54.5	54.5	54.8	54.9	55.2	55.7
52.3	52.5	52.4	52.7	52.5	52.8	53.5	52.7	52.8	52.5	52.0	52.4
52.3	68.0	75.7	80.8	84.0	86.3	88.0	89.2	90.3	91.1	95.2	96.8

Table 1: Classification accuracy and confidence factors of simulated models

not achieved the best classification accuracy. Additionally, although the 1% improvement of classification accuracy may seem unimportant, one should have in mind that in our measurements the difference between the classification accuracy of the best and reasonably worst system was typically from 10 to 15%.

GINESYS is a freely available scientific system which was, together with the benchmarking data, sent to over 50 researchers mainly in the developed countries. No major inconsistency was found.

6 Experiment with Winston's arches

It is difficult to measure the gains of multiple knowledge in exact domains. Most of exact deterministic procedures obviously don't need any multiple knowledge to perform theirs tasks. But there might still be many interesting tasks and domains where multiple knowledge can offer additional possibilities. As an example we have chosen the well known problem of Winston's arch (Winston 75).

The task of the system is to learn the concept "arch" from examples. In (Winston 75) there are two positive examples of arches and two negative examples of near misses presented in Figure 1.

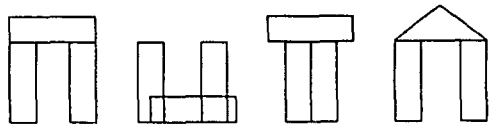


Figure 1: Learning examples for the Winston's arch problem.

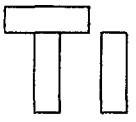
This case example was re-examined by many researches (e.g. Quinlan 90). The descriptions of objects and the solution can use the following relations: supports, left_of, touches, brick, wedge, parallelepiped all with predefined number of arguments. Negation is allowed. Although this description language is more limited that Winston's, the four learning examples are the same and similar problems can be observed in both approaches. The solution by Winston's system in this limited language is the following:

(S1)
arch(A,B,C) ←
    supports(B,A),
    supports(C,A),
    not (touches(B,C)).

Given the same learning examples, Quinlan's system FOIL constructed the following solution:

(S2)
arch(A,B,C) ←
    supports(B,A),
    not (touches(B,C)).

which covers not only the two positive examples, but also



The difference between the two solutions is commented by Quinlan: "Since FOIL looks for concepts in a general-to-specific fashion, it only discovers descriptions that are minimally sufficient to distinguish tuples in the relation from other tuples, which Dietterich and Michalski refer to as maximally general description. In this case, FOIL never formulates a requirement that the lintel be either a block or a wedge, or that both sides support it, because it has never seen near misses in Winston's terms, that make these properties relevant. FOIL would require a description of many more objects in order to elaborate its definitions of the relation arch."

In a different way, Winston devotes especial attention to different kinds of relations: "Humans, however, have no trouble identifying the forth example as an arch because they know that the exact shape of the top object in an arch is unimportant. On the other hand, no one fails to reject the second example because the support relation of the arch is crucial. Consequently, it seems that a description must indicate which relations are mandatory and which are inconsequential before that description qualifies as a model. This does not require any descriptive apparatus not already on hand."

In other words, Winston's solution (S1) contains only "MUST-BE" relations, which corresponds to FOIL's most general solution (S2). Other relations, also constructed by Winston's system are not obligatory, for example that A is a brick. Winston's solution not only contains more information, it is also the one expected by humans. Unfortunately, his system heavily depends on the proper order of examples and is domain dependent whenever near misses differ from positive examples in more than one relation.

FOIL is independent of the order of examples, is very quick and general. It has also a mechanism for handling noise and has found solutions of several interesting problems.

Both systems are very well known within the artificial intelligence community. However, they still more or less follow the formal logic approach which, for example, typically constructs only one solution.

Let us re-examine the Quinlan's solution by the multiple-knowledge approach. To do this, it is necessary to transform the description of examples from relations to attributes and its values. Theoretically this is possible because this domain is finite and practically it is possible because the domain is small. The transformation was performed by the LINUS system (Lavrač, Džeroski & Grobelnik 91).

LINUS enables transformation of problems presented in first-order logic without recursion into attribute-value language. Then, any of empirical learning systems like NEWGEM (Mozetič 85), ASSISTANT (Cestnik et al 87) or GINESYS can be chosen for intermediate knowledge construction. Finally, LINUS transforms results back into first-order logic. LINUS already enables the use of multiple knowledge since the user can apply different methods, compare the results and choose the best of them. Additionally, GINESYS alone enables construction of large sets of rules sorted by user-chosen preference criterion.

By applying GINESYS and using different preference functions two solutions emerge as the most reasonable, one being already presented and the other:

```
arch(A,B,C) ←
      supports(C,A),
      not (touches(B,C)).
```

The solution of Winston's system is also constructed but is somehow lost in tens of rules with similar values of preference functions. When another two near misses eliminating both too general solutions were presented, LINUS produced the wanted solution with ASSISTANT or NEWGEM and GINESYS showed there isn't any solution of similar preference. GINESYS enables yet further analyses. In one of the modes it produced the following solution (this time in an Algol-like notation):

```
if not (supports(B,A)) then not (arch(A,B,C)) else
  if not (supports(C,A)) then not (arch(A,B,C)) else
    if touches(B,C) then not (arch(A,B,C))
                    else arch(A,B,C)
```

and further analyses showed that first three rules can be freely interchanged.

With this example we have tried to illustrate additional possibilities enabled by multiple knowledge in exact domains. They include:

- finding all solutions of the same or similar preference
- analyzing different possible solutions by different preference functions
- analyzing why hasn't the system produced the intended solution
- analyzing the importance of subparts of solution and the form of the solution.

The approach of abstract logic has shown certain disadvantages even in the simple real-life task of learning the concept "arch" from four examples. First, most general solution was not the one that would be normally designed by humans using common-sense knowledge. Second, only one solution was proposed while solutions of the same preference were not even shown to the user. And third, several interesting solutions could be singled out by using different preference functions, but again, that is not common practice in the formal deterministic approach. Therefore, the debate about the appropriateness of the formal logic approach to real-life problems might bear some weight (BYTE, september 1990, 63 of the World's Most Influential People in Personal Computing Predict the Future, Analyze the Present), although much more elaborate analyses should be performed than our simple example.

Whatever the case, multiple solutions, i.e. solutions by the multiple-knowledge approach, seem to give better possibilities to analyze the properties of the domain and present more valuable information than systems constructing only one solution.

7 Discussion

There are many reports of different authors that confirm the practical advantages of multiple knowledge. Here presented simulated models strongly indicate the same conclusion and, furthermore, it seems that people inherently use multiple knowledge.

The Principle can be compared to other basic approaches to computing. Under the assumption that the Principle is valid the following can be argumented: (a) several basic principles like the use of redundant information in the information theory (Shannon & Weaver 64) or hierarhical decomposition promote very similar

approach; (b) Bayesian inductive procedure could be modified (Cheeseman 89) to capture the Principle; (c) Occam's razor (Blumer et al 86) and some conclusions in pattern-recognition theory should be accepted only in the pure form - their simplifications or generalizations sometimes confront not only the Principle but empirical observations as well.

Finally, we argue that if this Principle enables as important improvements in real-life domains as the first empirical measurements and simulations indicate, then people should use many knowledge bases, or in other words many different computer procedures, for most practical difficult tasks.

References

Blumer A., Ehrenfeucht A., Haussler D., Warmuth M.K. (1986) Occam's Razor, UCSC-CLR-86-2, USA.

Boose J., Bradshaw J., Kitto C. & Shema D. (1989) From ETS to Aquinas : Six Years of Knowledge Acquisition Tool Development, Proc. of EKAW 1989.

Brazdil P.B., Torgo L. (1990) Knowledge Acquisition via Knowledge Integration, Proc. of EKAW 1990.

Buntine W. (1989) Learning Classification Rules Using Bayes, Proc. of the International Workshop on Machine Learning, Ithaca, New York.

Catlett J. & Jack C. (1987) Is it Better to Learn Each Class Separately ?, Technical Report, Sydney.

Cestnik B., Kononenko I., Bratko I. (1987) ASSISTANT 86 : a Knowledge - Elicitation Tool for Sophisticated Users, Progress in Machine Learning, Bratko I., Lavrač N. (Ed.), Sigma Press.

Cheeseman P. (1989) On Finding the Most Probable Model, Technical Report.

Charniak E., McDermott D. (1985) Introduction to Artificial Intelligence, Eddison-Wesley.

Clark P. (1990) Reasoning with Differing Expert Opinions: A Novel Application of Expert System Technology, Technical Report, Turing Institute.

Emde W. (1989) An Inference Engine for Representing Multiple Theories, Knowledge Representation and Organization in Machine Learning, K. Morik (Ed.), Springer Verlag.

Gams M. (1987) Unifying Principles in Automatic Learning, Ph.D. thesis, Ljubljana.

Gams M. (1989) New Measurements Highlight the Importance of Redundant Knowledge, Proc. of EWSL 89, Montpellier.

Gams M., Drobnič M., Petkovšek M. (1991) Learning from Examples - a Uniform View, International Journal of Man-Machine Studies.

Jamnik R. (1978) Verjetnostni račun, Mladinska knjiga, Ljubljana.

Keyes J. (1989) Why Expert Systems Fail, AI Expert, 4, 11.

Lavrač N., Džeroski S., Grobelnik M. (1991) Learning Nonrecursive Definitions of Relations with LINUS, Machine Learning - EWSL91, Y.Kodratoff (Ed.), Springer Verlag.

Mozetič I., NEWGEM - program for learning from examples, Technical documentation, University of Illinois at Urbana - Champaign, USA.

Murthy S.S. (1988) Qualitative Reasoning at Multiple Resolutions, Proc. of Qual. Physics Workshop, France.

Quinlan J.R. (1990) Learning Logical Definitions from Relations, Machine Learning, Vol. 5, No. 3.

Schlimmer J.C. (1987) Learning and Representation Change, Proc. of AAAI 87, USA.

Shannon C.E. & Weaver W. (1964) The Mathematical Theory of Communications, Urbana, Illinois, University of Illinois Press.

Winston P.H. (1975): "Learning Structural Descriptions From Examples", The Psychology of Computer Vision, McGraw-Hill.

Keywords: autopoietic information, information, informational horizon, metaphysics of a thing and being, phenomenology of information, thing as information, thing in itself

Anton P. Železnikar
Volaričeva ulica 8
61111 Ljubljana

Ta spis je poskus informacijskega razumevanja stvari samih in še posebej bitja samega. Vprašanje o informiranju stvari je postavljeno v obliki, kako stvari informirajo in ne le kaj informirajo. V spisu je pokazano, kako se to vprašanje navezuje na mejna filozofska vprašanja, vendar tudi, kako se fenomenologija informacije loteva vprašanja stvari na sebi.

Pri analizi stvari kot informacije je osnovno izhodišče formula »stvar informira«. Bit, bivanje, bistvo stvari so le informacijske projekcije stvari same na opazovalca, projekti stvari na opazovalčevo metafiziko kot informacijo. Informiranje stvari so vse (oziroma le) njene opaženo aktualne in opaženo potencialne fenomenologije, tako ali drugače (spontano) opazovalno projektirane in konstruirane oblike in procesi stvarne pojavnosti. Le informacijsko konstruirano stvari se opazovalcu kaže kot odkrito stvari, neodkrito stvari pa mu ostaja označevalec za to, kar kot informacijsko stvari ni prisotno.

Metafizika stvari ali bitja je njej svojska, notranja informacija, npr. totalna informacija bitja, bitjevska informacija. To ni informacija zunanjega opazovalca stvari ali bitja, je pa seveda informacija bitja o zunanjem svetu. Metafizika označuje pojavljajočo informacijo bitja v fizičnem sistemu bitja, ki jo bitje kot avtopoezni sistem producira, generira za svojo rast, življenje, preživetje, obnašanje itd. Z metafiziko postane bitje opazovalec na zavestni in nezavedni ravni, kot razumno in avtomatno (instinktno) bitje.

Metafizika stvari kot informacija je posplošitev, ki postane smiselno mogoča z razumevanjem metafizike živega bitja, z živim izkustvom. Stvar je bitje, proces, razpoznavna, razločljiva entiteta, informacija. Metafizika stvari je informacija znotraj sistema stvari neglede na njeno naravo, izvor, pojavnost. Metafizika je notranja fenomenologija, tako kot se pojavlja in učinkuje v notranjosti kot sistemu stvari (kot sistem v sistemu), ki je za zunanjost na določen način neodkrit, z naravnimi in umetnimi čutili zunanjih opazovalcev le delno zaznaven. Zaradi

tega je ta pojavnost v določenem smislu nad-fizična, meta-fizična, v-fizična.

Informing of Things. This essay is an attempt of informational understanding of things in themselves and particularly of a being in itself. The question on informing of things is posted in the form »how do things inform« and not merely in the form »what do they inform«. This essay shows how this question concerns some boundary philosophic quests, but also in which way the phenomenology of information handles questions on things in themselves.

Within an analysis of thing as information, the basic point is the formula »thing informs«. Being and essence of a thing are only informational projections of a thing in itself onto observer, projects of things on observer's metaphysics as information. Informing of a thing are all (or only) its actually or potentially observed phenomenologies, in this or the other observable (spontaneous) way projected and constructed forms and processes of real occurrences. Only the informationally constructed of a thing shows itself to the observer as the revealed of the thing; the unrevealed of the thing remains to the observer as a marker for that which as the informational of thing is not present yet.

Metaphysics of a thing or being means the thing characteristic, inward information, for instance, the total information of a being, the being-own information. Metaphysics is not information belonging to the outward observer of the thing or being, however, it is certainly a being's information concerning the outward world. Metaphysics marks all occurring information within the physical system of being, produced as information of a being's autopoietic system, generated for a being's growth, life, survival, behavior, etc. By metaphysics, a being becomes the observer on conscious and unconscious level, as a reasonable and as an automaton-like (instinctive) being.

Metaphysics of a thing as information is a generalization which becomes senseful through the understanding of a being's metaphysics, through a live experience. Thing means being, process, recognized, distinct entity, information. A thing's metaphysics is information inside the system of a thing, regardless of its nature, origin, occurrence. Metaphysics is the inner phenomenology, as it appears and performs within the system of a thing (like a system embedded in a system); to some extent, in the exterior of thing, metaphysics might be not revealed or only partly perceived by natural or artificial sensors of outward observers. This is the reason to call this phenomenon in the figurative sense as trans-physical, meta-physical, in-physical.

1. Uvod

Obravnava, ki jo začenjamo, je posplošitev, spominjanje in razumevanje (hermenevtika) neke poti, ki jo je prehodil misleči popotnik skozi stoletja, od *grške* forme (»eidos«, »morfe«) do *latinske* »informatio«, od filozofskega pojmovanja biti do sodobnega zavedanja informacije; nikoli prepričan, da je sploh kaj novega odkril v svojem spontanem in krožnem iskanju, v vračanju k vedno novi tavtološki spontanosti. Toda pot kot usmeritev je bila vendarle tudi nova, sproti nastajajoča, ki se je odpirala vselej, ko so stopinje zadele na sledi še neodkritih odtisov informacijske pojavnosti [Železnikar, OWI].

Kot živa bitja, kot ljudje, kot bitja zavesti zmoremo misel le v obzorjih svoje biološke lupine formacije živega (glej npr. avtopoeza, [Maturana, AC]). Ali je to povsem res? Kaj pa če se biološka lupina, njeno oblikovanje postopoma širi, nastaja hkrati s pojavnim naporom (učenjem) informacije [Nottebohm, BSN]? Karkoli odkrivamo, asociiramo, analiziramo, je vselej posebna informacijska situacija v obzorju trenutne osebne atitude, v domeni informacijskega stanja (vznemirjenosti, motenosti) živega bitja. Prav tu, v prostranstvu dinamične biološke olupinjenosti, je to, kar lahko imenujemo informacijska razprtost neznanega, nikoli doumljenega, totalno nevidnega, nemisljivega in prav zaradi tega razširljivega, prihajajočega. Tu sicer preneha človekova informacijska neposrednost, se omeji za lupino obzorja zavestnega, iz katere informacijsko še ne more izstopiti; toda onstran obzorja trenutnega razumevanja že čaka to, kar vstopa kot novo, ki bo razumljeno.

Izrecimo hipotezo: »Vse je informacija.« In že se znajdemo v nenadejani, hipotetično nepremostljivi zadregi: »Kako oporekati, raziskati in dokazati, da biva informacijsko, ki ni informacijsko, ki ni tudi informacijsko pojavno. Kako oporekati, dokazati, izreči, da pojavno ni (tudi) informacijsko?«

Pojavno, vse kar se v tej ali oni obliki ali procesu pojavlja, je informacijsko osedanjeno. Pojavno zajema materialni in duhovni svet, njuno posebno in naposled skupno informacijsko pojavnost. Vsakršna pojavnost je oblika ali proces informacijskega nastajanja, prihajanja informacije v njeno pojavljanje. Tudi pravkar zapisani odstavki so lahko informacijski in lahko sprožajo nastajanje informacije v naslovnikih (bralcih). Vobče tudi stvari »vznemirjajo« stvari (reči), vplivajo na njih, jim spreminjajo njihovo pojavnost, strukturo in organizacijo. Fenomenologija stvari se kaže prav kot informacijska pojavnost.

Besedilo o informaciji, ki nastaja, literarizira pojem (*lat.* suspicio, *nem.* der Begriff, *angl.* notion) informacije in hkrati destabilizira kanon vsakršne filozofije, njen epistemološki patos in prizadevanje, da bi bila resničnostni ali zdravorazumski opis sveta. Filozofija in z njo tudi filozofija informacije je le metafizični produkt filozofa, zgodba filozofskega akterja v določenem informacijskem stanju, njegovo informiranje. In v tem sta etimološki patos in resničnostna pretenzija (domišljava prevzetnost) lahko le literarna okraska, posebnosti, namenjeni uživalcu filozofske, znanstvene, to je umetelne literarne informacije.

Informacijsko je vse, kar je opazovalno distinktivno (razločljivo), analitsko in sintetsko oblikovano, premišljeno, asociirano v informacijske entitete in iz njih. Informacija se oblikuje iz različnih dobro in slabo oblikovanih informacijskih enot, povezano in svobodno nastajajočih, urejenih in blodečih informacijskih drobcev (lumpov). Tudi informacijsko oblikovanje, povezovanje, spajanje, prepletanje informacijskega, je informacijski pojav, informacijsko nastajajoča oblikovalna entiteta.

Informacija je aktivno-pasivna pojavnost, oblikovalna in učinkovalna (*angl.* forming and impacting) in oblikovana (*angl.* formed and impacted), procesirajoča in procesirana hkrati. Vsebuje torej nekaj, kar ji omogoča oblikovanje in procesiranje (oblikovalce in procesorje), in nekaj, kar lahko oblikuje in procesira, kar bo oblikovano in procesirano. Pri tem se vloge aktivnega in pasivnega v informaciji izmenjujejo, aktivirajo in pasivizirajo, informacijsko prekrivajo, prepletajo in seveda nastajajo. V matematičnem jeziku bi bila informacija rekurzivni objekt, enota in entiteta, ki bi bila hkrati operand in operator v spreminjajočem in nastajajočem pomenu zadevne entitete.

2. Informiranje stvari

Postavimo tole začetno formulo razmisleka o stvari sami (stvari na sebi): »Stvar informira.« Stvar *je* (biva) tako, kot informira. Stvar informira, če je, če obstaja, postaja, nastaja, izginja, preminja. Informiranje stvari kaže to, kar stvar v svojem nastajanju (obstajanju, propadanju) je. Bit stvari je njeno informiranje. Bistvo stvari se kaže, razkriva, prihaja v prisotnost z informiranjem stvari. Stvar informira sama po sebi (po svoji naravi, v sebi, zunaj sebe). Informiranje stvari so vse njene aktualne in potencialne fenomenologije, vse različne oblike in vsi različni procesi njene pojavnosti. Informacijski fenomeni stvari so lahko zaznavni in zakriti, odvisni od opazovalnih zmogljivosti opazovalcev.

Vprašanje o stvari (*lat.* res, *angl.* thing, *nem.* das Ding) smo postavili informacijsko (netradicionalno), to je drugače, kot ga postavljajo filozofi. Za filozofe je stvar razločljivi, števni individuum, zunajzavestno bitje, realni, resnični, pojavni predmet (*nem.* der Gegenstand, dobesedno protistanje, slov. tudi pred-met, proti-stvar). Informacijsko je stvar, informacijski pojav, tudi kot zavestna informacija. Stvari sestavljajo stvarni svet oziroma stvarnost, ki je zunaj realnega doživljanja predmetnosti. Seveda je stvarnost tudi mišljenje kot informacijski proces živih korteksov. Stvari, ki so stvari le, v kolikor informirajo, sestavljajo stvarnost (carstvo, prostranstvo, prostor, domeno, območje, polje) informacije. Informacijsko so stvari poljubne entitete, ki vplivajo na stvari, tudi same nase, in so vplivane s stvarmi, tudi same s seboj. Informacija ni nič drugega kot pojavnost ali fenomenologija stvari, neglede na (fizikalno, fiziološko, informacijsko ali drugo) naravo fenomenologije.

Heidegger izreče na začetku svojega disputa o stvari [WIT, 2] kratkomalo tole: »Vprašanje „Kaj je stvar?" je eno tistih, s katerimi ni mogoče ničesar začeti. Več kot to *o* tem ni potrebno reči.« Ta citat kaže predvsem na težavnost odgovarjanja na staro filozofsko vprašanje, ki vprašuje o bistvu, pojmu stvari kot stvari, stvari na sebi (*angl.* thing-in-itself, *nem.* Ding an sich), njeni biti. Vendar vsaka konkretna stvar konkretno informira in na določen način informira tudi vprašanje o stvari kot stvar (entiteta) človekove metafizike, natančneje zavesti. Ta zadnji izrek kaže na dvoumnost, krožnost in spontanost govorjenja o stvari. Za opazovalca, za opazovalno informacijo stvari, je stvar to, kar je stvar kot pojav za opazovalca (*angl.* thing-for-us, *nem.* Ding für uns). Stvar informira opazovalca le do te mere, kot ga lahko informira, kot lahko opazovalec informacijo stvari kot informacijske pojavnosti sprejme, razume, informacijsko konstruira.

Kaj je stvar, ki informira? To je prvo vprašanje, ki si ga postavi opazovalec v obliki, kaj je stvar na sebi (npr. Kant). Vprašanje po kajstvu stvari je le delno, partikularno usmerjeno vprašanje, omejeno s pomenom vpraševalne besede »kaj«. Kajstvo stvari zadeva v bistvu le informacijsko statiko (definicijo) stvari, čeprav v obliki skrajnega (mejnega) pojma mogočega spoznavanja, saj navadno izpušča vprašanja z vpraševalnimi besedami »kako«, »zakaj«, »kje«, »kam«, »komu«, »odkod« itn., ki dodatno vznemirjajo opazovalčevo pozornost. Kajstvo stvari je na ravni koncepta stvari, je konceptni model stvari, vselej delna, na poseben način (npr. znanstveni) oblikovana informacija, tj. pojmovna formula.

Kako stvar informira? To je drugo vprašanje, ki zahteva odgovor o informiranju stvari kot informacije in o informiranju informacije kot stvari. Tako kot informacija pogojuje svoje informiranje, tudi informiranje pogojuje svojo informacijo. Ali, tako kot je stvar oblika s svojo vsestransko fenomenološko procesnostjo, je tudi proces z raznoliko pojavno oblikovnostjo. Vpraševalna beseda »kako« sproža vprašanja o bivanju stvari, o njenem pojavljanju, nastajanju, spreminjanju, propadanju, izginjanju, to je o njenem informiranju.

Zakaj stvar informira? Informiranje stvari je posledica njene narave, njenih pojavnosti, ali rečeno v znanstvenem jeziku, njenih fizikalnih, kemijskih, fizioloških, bioloških, psiholoških, družbenih fenomenologij, od katerih ima vsaka svojo informacijsko ozadje, svoj jezik in s tem specifično razumevanje stvari kot pojava. Vprašalna beseda »zakaj« nosi specifični informacijski naboj opazovalne komune, ki vprašanje

postavlja, je tedaj postavljena v okvir komunalne (skupinske) inteligence kot informacije.

Kako je stvar informirana? Informirano stvari je stvar sama v svoji pojavnosti, tj. v svoji trenutni oblikovnosti in procesnosti. Informirano stvari je to, kar je s stvarjo kot stvarjo nastalo in postalo, kar se informacijsko razvija glede na trenutno strukturo in organizacijo stvari, glede na njen položaj v okolju, glede na njeno atitudo (stanje, držo, konstelacijo). Informirano stvari je tako odvisno od informiranja stvari same in informiranja drugih stvari. Informirana stvar ima pomen vplivane, od sebe in okolja, od drugih stvari odvisne stvari.

Stvar, ki informira, stvaruje. Uvedba glagola »stvarovati« omogoča izražavo splošne informacijske formule: »Stvar stvaruje.« Tako kot rečemo povsem normalno »informacija informira«, »bit je«, »bivajoče biva«, »misel misli«, »razum razumeva«, »ime imenuje«, »pesnik pesnikuje« itn. pa ni več jezikovno normalno, če rečemo »Bog boguje« (angleščina dopušča »God gods«), »bistvo bistvuje«, »Petra petruje«, »Slovenija slovenuje« itn. Formula »stvar stvaruje« je bistvena, ker izraža krožno ali cirkularno naravo informacijske pojavnosti stvari, ki je po naravi stvari tudi avtentična, avtonomna, svojska, samobitna, samoproduktivna (samoohranjajoča), samoreferenčna, samorefleksivna itd.

Formula »stvar stvaruje« kot informacija kaže tudi spontano nastajalno naravo stvari, ki stvaruje in s svojim stvarovanjem informira. Spontanost bivanja stvari je v njeni materialni, duhovni, družbeni naravi, vplivih in vplivanjih, ki so spontano informiranje stvari, ki stvaruje.

Bivajoče stvari je prav njeno informiranje. Informiranje je aktivni (procesni) in pasivni (oblikovni) princip stvari. Stvar sama (stvar na sebi) je strokovni termin Kantove spoznavne teorije: stvar je od spoznavnega subjekta nedotaknjena (nespremenjena) stvar. Konstitucija čutilnih organov in struktura in organizacija razumevanja v korteksih popačijo stvar samo v t.i. videz (*nem.* die Erscheinung). Te omejenosti spoznavnega procesa nikakor ni mogoče preseči in vsled tega ostane stvar sama za človeka načelno nespoznavna. Tako je mogoče stvar le misliti in pojem stvari na sebi postane tako mejni problem mišljenja. Vendar je zavedanje te omejenosti bistveno, ker kaže na omejenost človekovih spoznavnih (informacijskih, metafizičnih) sposobnosti.

Bistvo stvari je lahko le to, kar stvar informira (izkazuje) v informacijskem sprejemniku, kar je skozi informiranje lahko stvarno izraženo in izrazljivo o stvari v misleči substanci. Bistvo stvari je tedaj njeno razumevanje kot pojav (videz) stvari v primerjavi z razumevanjem drugih stvari. Zahteva po objektivnosti razumevanja, realnosti stvari same se tako prenese na opazovalca kot imperativ, kot zahteva, da naj bo opazovanje, kolikor je sploh mogoče, objektivno, tj. stvarno.

Bit stvari bi bila njena nespremenljiva, za vselej opredeljiva narava, popolnost, mejna vrednost predstavljivega, to je »je« stvari same. Bit stvari bi učinkovala, informirala kot podatek o stvari, ki je vselej enak in s svojo enakostjo (ponovljivostjo) informira druge stvari. Stvar kot bit ne bi bila informirana (vplivana), bila bi lahko razpoznana kot dokončna resnica (narava) stvari kot biti stvari, kot stvari same.

V čem je razlika med informiranjem materialnih in duhovnih stvari? Tako kot je materialno lahko videno, slišano, tipano, vonjano, okušano ali preko naravnih senzorjev posredovano, je duhovno lahko percipirano, koncipirano, opazovano, pojmovano, razumljeno. Pri živem bitju postane tudi materialno-informativno in materialno-informirano prej ko slej, v okviru možnosti nastajanja avtopoeznega sistema, transformirano v metafizično-informacijsko, to je v duhovno, zavestno in nezavedno. Materialne stvari informirajo preko čutnih organov telo in kortekse živega bitja, kjer nastaja duhovno kot informacija. V okviru bitja je seveda tudi vsaka njegova duhovna informacija materializirana, npr. z molekularnimi oblikami in njihovo fenomenologijo (raznovrstno procesnostjo) v celicah, celičnih populacijah, korteksih, telesu. Materialno sicer pogojuje duhovno, vendar tudi duhovno vpliva na oblikovanje materialnega, npr. na nastajanje nevronov, njihovih povezav in mrež [BSN].

O informiranju stvari lahko izrečemo tole pov-

zemajočo misel: stvar informira in opazovalec stvari je z informiranjem stvari (s stvarnim informiranjem) informiran le do stopnje, ki jo kot informacijski opazovalec zmore (premore, s svojo spoznavno informacijo razumeva). Informiranost opazovalca o stvari je celo bistveno (lahko pretežno) odvisna od njegove informacijske zmogljivosti sprejemanja informiranja stvari. V tej pogojenosti se skriva tudi vsakokratna problematičnost fenomenološkega vzklika »k stvari sami«. To vzklikanje ne daje pričakovanih rezultatov, če je opazovalec nezadostno, neprimerno, neintencionalno odprt za informirajoči nagovor stvari same. Neglede na značilno informacijsko zmogljivost opazovalca in prav zaradi te njegove značilnosti (informacijske specifičnosti), informiranje stvari same ne more biti nikoli v celoti sprejeto: bit stvari kot informacija ostaja tako nedokončna, nedosegljiva, vselej drugače zakrita spoznanju opazovalca.

3. Informacija živega bitja

Živo bitje je živa entiteta (tubit), npr. biološka celica, nevron, celična populacija, organizem, korteks, živo bitje (rastlina, žival) kot avtonomna, spontano bivajoča celota živih enot. Živo bitje je intimna informacijska spojenost, soodvisnost in informacijska kooperacija, medsebojno razumevanje njegovih enot. Kaj je informacija živega glede na materialni ustroj bitja (živi sistem), glede na različne življenske pojave v tem ustroju in glede na zunanje informacijske vplive na ta ustroj?

Metafizika kot informacija živega bitja bo označevalec (marker, kazalec) za totalno informacijo bitja samega (stvari), za celotno živčno, telesno, signalno, substancialno, življensko informacijo, ki je primitivno strukturirana in visoko organizirana informacijska pojavnost bitja. To je razširjena, posplošena in redefinirana opredelitev metafizike kot pojavnosti svobodnega, spontanega, cirkularnega, skratka življenskega informiranja živega bitja (ali vobče stvari). V okviru redefiniranega in razširjenega pojmovanja informacije se s tem ponuja tudi razširjeno pojmovanje metafizike kot informacije. Zakaj je izbira s tem pomenom opredeljenega termina »metafizika« smiselna?

Vobče ima stvar svojo metafiziko. V živem bitju označujemo z metafiziko informacijsko fenomenologijo v okolju tako imenovane avtopoeze (samoproduktivnosti) bitja [AC]. Metafizika kot informacija je celotna duhovna, upravljavska, odločevalna, vedenjska informacija, informacijska vseobsežnost telesa (npr. živčnega sistema, možganov, imunskega sistema, mišljenja, obnašanja, razumevanja). Informacijsko totalno ima mnogoteri pomen: miselno, signalno, pojavno aktualno in informacijsko mogoče v okviru bitjevske avtopoeze; informacijsko pojavno, ki zadeva vsakršno pojavnost živega individuuma kot biološko- materialnega in duhovnega bitja, bitjevsko informiranje v sebi, iz bitja v okolje in iz okolja v bitje.

Metafizika je tako vselej informacijska pojavnost, ki je različno opredeljevana. Aristotelsko pojmovanje metafizike (v delu, ki je sledilo njegovi Fiziki) ima svoj izvor v spisih Aristotela kot živega filozofa. Ta metafizika je nauk o pojmovnih strukturah oblikovanja znanja, vključno izkustvenega. Z njo se npr. uvaja razumevanje osrednjih kategorij, kot so oblika/materija, akt/potenca, esenca, bistvo, bit, substanca, resničnost, bog, duša, torej vrsta osrednjih informacijskih konceptov.

Kaj je tedaj Aristotel v svoji Metafiziki konkretno obravnaval? Kako je razumeval to, kar je bila *(ta) meta (ta) physika?* Njegovo delo [MP] je mogoče razdeliti v skupine knjig, ki obravnavajo uvode v filozofijo, področje (provinco) filozofije, bit in nastajanje, filozofske odlomke in nespremenljivo (večno) bit. Posebni termini njegovega dela so: bit, kategorije, principi pojasnjevanja, narava in sprememba, človekova narava in duša, um in naposled božje. Oglejmo si kratke povzetke knjig iz njegove Metafizike [MP].

Uvode v filozofijo sestavljajo knjige (vsebina):

TEORIJA POJASNJEVANJA (teoretično znanje; narava in cilj modrosti; materiali, gibanje in dobro; pitagorejski in parmenidski principi; platonski material in formalne razlage; razlaga kot material, Učinkovalno, Formalno in Končno;

kritika zgodnjih filozofij in doktrin o idejah);

OPOMBE O FILOZOFSKIH PROCEDURAH (napredek v filozofskem znanju; zavrnitev neskončnega vračanja; pomembnost vaje in metode);

ZNAČILNI PROBLEMI SPLOŠNE FILOZOFIJE (oris filozofskih problemov; enotna znanost in spremenjena bit; kategorije kot principi pojasnjevanja; oblike, principi in enota; telesna in matematična bit; ideje, moč in univerzalnost).

Področje filozofije sestavljajo knjige:

BIT IN ZNANJE (znanost o biti kot biti; enotnost filozofije; filozofija in aksiomatika; razprava o kontradikciji; percepcija, sprememba in resnica; prava razsodba in relativnost; posrednost in izključena sredina; kontradikcija in prazna indiskriminacija);

DEFINICIJE TERMINOV (začetek; pojasnjevalni faktor; element; narava; nujno; Eno; bit; primarna bit; isto, drugo, različno, podobno; nasprotno, drugo v načinu; prej in potem; moč; količina; kakovost; relacija; popolno; meja; dispozicija; navada; trajanje; pomanjkanje; imeti in držati; izvirati iz; del; celota; mnogostransko; genus; napačno; naključno);

PRVA FILOZOFIJA IN IRELEVANTNI NAČINI (teoretične znanosti kot naravno, matematično in teološko; izločitev naključne biti iz znanosti; pristnost naključnega; izločitev biti kot resničnega iz znanosti);

Bit in nastajanje obravnavajo knjige:

ISKANJE PRIMARNE BITI (primarna in odvisna bit; definicija in integralni koncepti; definicija, znanje in objekti; vodila za naravno in umetno produkcijo; spojenost materije in oblike v objektih; tipi sprememb; definicija, analiza in oblika; definicije in težko razpoznavne oblike; definicije in osnovne značilnosti; univerzalne in skupne lastnosti; univerzalije, definicija in znanje; najvišje univerzalije; problem enotnosti individualnega);

ENOTNOST MATERIJE IN FORME (materija in sprememba; diferencialne značilnosti in individualnost; domnevno posredovanje med materijo in formo; nastanek specifičnih objektov in dogodkov; težave v logiki spremembe; enotnost individualno pojasnjenega);

SILE IN OPERACIJE (aktivna in pasivna sila; neracionalna in racionalna sila; sila in sprememba; sila in možnost in nemožnost; antiteza sile in dejanja; pogoji potencialne biti; ocene sile in dejanja; resnica in bit);

ENOTA IN ODVEDLJIVI KONCEPTI (enota kot individuum in mera; enota in bit kot končnoveljavno; enota, pluralnost in njuni odvodi; narava nasprotij; skupno zanikanje skrajnosti; nasprotje enote glede na pluralnost; narava posredujočega; drugost; nasprotja, ki označujejo razlike v vrsti; minljivo in neminljivo).

Filozofski odlomki so knjiga, ki obravnava *ponovitve in citate*, in sicer

O METAFIZIKI (vprašanja o enotnosti modrosti; vprašanja o statusu principov; bit in enotnost filozofije; prva znanost in aksiomatika; principi nasprotja; trdilne težave pri kontradikciji; stopnja teologije med znanostmi; bit kot naključje in kot resnica) in

O FIZIKI (naključje; sila, operacija in gibanje; neskončno; sprememba in gibanje; vrste gibanja in sorodni koncepti).

Večno bit obravnavajo knjige:

BOŽJA BIT (primarne biti in njihove vrste; spremenljiva primarna bit in materija; definitna sprememba in oblike; analogije različnih sprememb; status principov v okviru minljive biti; večni prvi vzrok; končno dobro; zvezdna božanstva in božja enotnost; samoaktivnost vrhovne biti; božja popolnost in naravni svet);

MATEMATIČNE ENTITETE IN IDEJE (vidiki matematike in idej; matematika in konkretno; matematika kot abstraktno; ideje kot izvirno in izbrano; ideje kot brezpomembno za pojasnjevanje; vidiki števil kot pojasnila; narava in izpeljava števil; števila, geometrija in ideje; splošnost in partikularna referenca znanja);

ŠTEVILA, IDEJE IN PRVI PRINCIPI (nasprotni principi, enota in pluralnost; elementi ter večna in pluralna bit; števila ideje in generacije; prvi principi in bog; števila, generacija in pojasnilo; posledice fantastičnih doktrin o številih).

Metafizika je klasično ime za osnovno filozofsko znanost, ki ji je temelje postavil Aristotel. V okviru metafizike se združujejo velika vprašanja in sistemski pojmi tako imenovanih pos-

lednjih, mejnih, še mogočih vzrokov biti. To metafizično pojmovanje si lasti določeno univerzalnost in vključuje (upošteva) druge discipline, npr. logiko, spoznavno teorijo, ontologijo, etiko, estetiko, kibernetiko itd. Tu postane metafizika sinonim za filozofijo, s katero je od Aristotela dalje bistveno povezana (prva filozofija).

Kot osrednji element Zapadne filozofije pomeni metafizika od Grkov dalje različne stvari. Je poskus opredeljevanja eksistence ali resničnosti (realnosti) kot celote. V tem se razlikuje od naravnih znanosti, ki raziskujejo le posamezne dele ali vidike celote. Materializem in idealizem, Spinozin monizem in Leibnizova monadologija, skratka univerzalne teorije, so primeri metafizike v tem smislu. Metafizika je tudi poskus preučevanja področja nadčutnega (zunajčutnega), ki je onkraj izkustvene informacije. Metafizika oblikuje gotove (nedvomne) prvinske principe kot temelje za vse drugo znanje, kritično raziskuje, kaj prezre in pozabi bolj (znanstveno, filozofsko) omejena ali usmerjena disciplina kot dano (predpostavljeno, domnevano, privzeto, tavtološko).

Kritika metafizike ugotavlja, da opisani cilji in smeri v bistvu niso dosegljivi. Človekov um naj ne bi odkrival faktov zunaj območja čutnega izkustva (senzorske informacije) oziroma zunaj informacijskih možnosti, ki so pogojene s človekovo avtopoezno naravo v spremenljivem in nepredvidljivem okolju. Npr. matematika, ki temelji na navdihu velikega števila metafizikov, dosega svojo izkustveno neodvisnost zgolj zaradi ukvarjanja s tavtologijami. Tudi prepričanje, da je realnost sestavljena iz ene same substance (monizem) ali iz neskončnega števila substanc (monadologija), ne prispeva k ekonomiji misli o svetu in tovrstno prepričanje naj ne bi bilo niti resnično, niti lažno, temveč nesmiselno. Vsak poskus opredeljevanja realnosti kot celote mora neizbežno uporabljati koncepte, ki so bili poprej razviti za elemente realnosti in so tako v konceptu celote zlorabljeni (empiricizem, pozitivizem).

Vendar pa prav naravoslovne znanosti in tudi naše vsakdanje izkušnje temeljijo na domišljenih metafizičnih sistemih informacije, ki jih sestavljajo znanje, koncepti, principi in prepričanja. Metafizični impulzi prihajajo iz sistematičnega raziskovanja in ne iz realnosti oziroma iz fundamentalne strukture in organizacije realnosti kot informacije v možganih. Narava kot narava pozna koncepte, principe in prepričanja le kot človekove moduse mišljenja, kot konstrukte človekovega uma, kot miselne modele narave, ki so realnosti razmišljajočega uma, konkretne molekularne oblike in pojavni procesi nevronsko zamreženih korteksov. Zakonitosti narave in družbe so čisti človekovi dosežki, njegove izvirne umišljenine, izvorna informacijska pojavnost prepletene žive nevronske strukture in organizacije, ki je oblikovna in procesna prepletenost informirajočega uma.

Metafizika kot informacijska vseobsežnost bitja vsebuje torej koncepte filozofske metafizike, ki od svojega začetka spekulativno išče vseobsegajoče osnovne principe, kot so npr. Bog, Bit, Ideja, Jaz, Monada, Apriorno, Weltgeist, Materija, Nasprotje, Gibanje, Čas, Volja, Obseženo, Up, Življenski prostor, Univerzalni fizikalni sistem, Formula formul itn., s katerimi bi bilo mogoče postopno pojasnevati resničnost v njeni totalnosti.

Ko uvajamo informacijski vidik metafizike, ki se od predhodnih vidikov razlikuje tako, da jih na svojstven način združuje, dodaja pa neko integralno lastnost, se spomnimo: da je Aristotelova metafizika izvorno to, kar sledi fiziki kot objektu neke Aristotelove razprave; da je filozofska metafizika relativno dobro opredeljen termin za filozofske temelje; obstaja pa še razumevanje metafizike kot pojem v pogovornem jeziku za označevanje nadnaravnega kot miselnega pojava. V tej dispoziciji stopi v ospredje metafizika kot informacijska kategorija, kot lastnost bitja, stvari, predmeta za sebe, za vidik svojega, notranjeznačilno pojavnega.

Kaj je metafizika kot informacija, kako informira? Kaj je metafizika stvari same (na sebi), kako se razlikuje od stvari same (na sebi)? »Stvar stvaruje« pomeni »Stvar informira na stvaren način.« Metafizika kot informacija je notranja informacija stvari in v tem specifični del informacije stvari same. Metafizika je tedaj tisti del informacije, ki se v informaciji pojavlja kot notranja opazovalna (za sebe) entiteta. Kako stvar »opazuje« sebe in svoje okolje, svojo »notranjo« in

tujo zunanjo informacijo? Kaj je metafizika stvari, bitja, informacije? To, kar opazuje, opisuje nekaj, ima lastnost opazovanja nečesa, producira informacijo Nad-nekaj, o-nekaj (metainformacijo), zunajnekajšno informacijo. Bitje kot fizični sistem združuje (povezuje v povezanost) svojo notranjo informacijo, zanj-informacijo, bitjevo informacijo kot metafiziko (fizike bitja). Metafizika označuje pojavljajočo informacijo bitja v fizičnem sistemu bitja, ki jo bitje kot avtopoezni sistem producira, generira za svojo rast, življenje, preživetje, obnašanje itd. Z metafiziko postane bitje opazovalec na zavestni in nezavedni ravni, kot razumno in avtomatno (instinktno) bitje.

Metafizika stvari kot informacija je posplošitev, ki postane smiselno mogoča z razumevanjem metafizike živega bitja, z živim izkustvom. Stvar je bitje, proces, razpoznavna, razločljiva entiteta, informacija. Metafizika stvari je informacija znotraj sistema stvari neglede na njeno naravo, izvor, pojavnost. Metafizika je notranja fenomenologija, tako kot se pojavlja in učinkuje v notranjosti kot sistemu stvari (kot sistem v sistemu), ki je za zunanjost na določen način neodkrit, z naravnimi in umetnimi čutili zunanjih opazovalcev le delno zaznaven. Zaradi tega je ta pojavnost v določenem smislu nad-fizična, metafizična, v-fizična.

Grški prislov »meta« [Senc, GHR, 601] se pojavlja kot predpona zloženk in pomeni sredi, medtem; potem, tedaj (takrat), zatem; še k temu, razen tega (in kot predlog med, s, z, ob, poleg, za, nad). Npr. naravni jezik je sebi tudi metajezik, tj. opazovalni, opisovalni, spoznavni jezik jezika. Prav v tej metafiziki jezika kot informaciji se skriva tavtološkost, dokončno nerazrešljiva cirkularnost jezika (zaprtost jezika v jezik), konceptualna jezikovna neterminalnost (nedokončnost, nikoli dokončna bistvenost). Tudi fizika kot znanost ima svojo metafiziko, tisto npr., ki kot informacija legalizira informacijo v fiziki kot informaciji komune fizikov. Ta metafizični princip se uporablja zavestno in nezavedno v vsaki znanstveni disciplini, od matematike do filozofije.

Metafizika je za nas kot bitja naša, stvarna informacija, informacija nas samih kot stvari, o naši in drugi zaznani informaciji. Filozofsko rečeno je metafizika informacija tubiti, to je v okviru avtopoeznega sistema bitja možna, aktualna, vsakršna, glede na »živo« fenomenologijo bitja celotna, torej eksistencialna informacija. Kot zavestna informacija lahko metafizika vsebuje tudi informacijo o biti stvari (bivajočega) le kot tubitna informacija biti. Bistvo stvari je tako na ravni bitja vselej metafizično, metafizično nastajalno, pojavljajoče, odvisno od stanja in atitude konkretne tubiti.

Kaj je metafizika bitja kot informacijska diferenca? Bitje samo (na sebi) je vsa informacijska fenomenologija bitja, zaznana in nezaznana z bitjem kot opazovalcem (in neopazovalcem) in z zunanjimi opazovalci. To je vsa informacija bitja (za bitje), toda ne vsa aktualna in potencialna informacijska fenomenologija bitja. Metafizika je značilna, bitjevsko prisotna in nastajajoča zanj-informacija ali znotraj-informacija, t.i. v-informacija bitja. Razlika informacije bitja samega in njegove metafizike je preostanek (diferencial), ki je lahko ali lahko postane zaznaven tudi zunanjemu opazovalcu, ni pa zaznaven za bitje, je informacijsko zanj brez vpliva. Ta preostanek je za bitje brez-informacija ali zunaj-informacija. Metafizika je tedaj prav informacijska diferenca med informacijo bitja samega in brez-informacijo za bitje.

V zavestni informaciji bitja postaneta razliki med informacijo bitja samega in njegovo metafiziko ter med informacijo bitja samega in brez-informacijo razumljivi (zaznani kot razliki) tako za bitje kot zavestnim opazovalcem kot za bitja kot zunanjimi zavestnimi opazovalci bitja. Tako kot bitje kot opazovalec ne more prodreti v informacijsko bistvo brez-informacije, tudi zunanji opazovalci bitja ne morejo prodreti v bistvo metafizike bitja, čeprav obstajajo informacijska prekritja (vmestitve) med brez-informacijo in metafiziko. Obstaja tedaj še tista diferenca med metafiziko bitja in brez-informacijo, ki ni zaznavna za metafiziko in tista diferenca, ki ni zaznavna za brez-informacijo.

Metafizika bitja, tako kot je opredeljena, je s tem vselej le delna informacija glede na informacijo bitja na sebi. Metafizika kot kulturna informacija je temeljna informacija filozofije, ki se je

nabrala skozi tisočletja človekovega umskega razvoja, njegovega filozofiranja (meditiranja). Med tema vrstama metafizike je seveda bistvena razlika, čeprav se medsebojno marsikje prekrivata. Zaznavanje te razlike je nekakšna meta-metafizika, to, kar je za metafiziko, za njenim obzorjem, kar prihaja, je na poti kot informacijsko obzorje metafizike bitja. Ontološka diferenca, ki jo je vpeljal Heidegger [SZ] kot razliko med bivajočim samim in bitjo samo, je znana informacijska diferenca v tradicionalni metafiziki in ontologiji in pomeni nepremostljivo razpetost med ontološkim in ontičnim v okviru Heideggrove temeljne ontologije. Danes označuje ontološka diferenca razmerje med dvema področjema, za kateri velja a priori, da ju ni moč reducirati eno na drugo, da torej ne gre za dva vidika ene in iste stvari.

4. Sklepna beseda

V okviru te kratke razprave o stvari sami kot informaciji in kot metafiziki stvari kot informaciji se nismo dotaknili še vrste drugih stvari, ki bi spadale v ta kontekst, vendar ne samo v kontekst stvari. Zato smo se omejili na vsebino dveh tistih poglavij v okviru splošne informacijske fenomenologije [Železnikar, II], ki se stvari kot informacijskega pojava neposredno dotikata. Nismo se posebej dotaknili razumevanja kot hermenevtike stvari [Gadamer, PH] in kot hermenevtične informacije [UAI]. Ognili smo se tudi formalnemu teoretiziranju (npr. v smislu [Landman, TTI]), ki bi bilo vsekakor zelo zanimivo (z uporabo posebnega formalnega aparata).

Z informacijskima pojmoma stvar in metafizika (stvari) smo uvedli le sodobnejši pogled na ti mejni informacijski pojavnosti v človekovih korteksih, tj. kot kategoriji zavesti. Ta »transformacija« razumevanja iz tradicionalnega filozofskega polja v informacijsko polje skriva seveda določeno namero, ki jo moramo šele odkrivati. Eno od vprašanj te namere je tehnološko in za današnji čas že preseženo: Kaj je računalnik kot stvar sama in kakšna je njegova metafizika? Načrtovalcu računalniške arhitekture in programerju se zdi to vprašanje razrešeno, dokler se giblje v strogo determiniranem prostoru načrtovalnih orodij, znanja oziroma metodologije. Kaj pa informacijski stroj: kakšna je njegova stvar in metafizika, če deluje na temelju informacijskih principov [OWI]? Zahteva teh principov je namreč informacijska nastajalnost, ki je stvarna (arhitekturna) in metafizična (programska). To pa ni več zgolj računalnik, ki je naposled le stvar arhitekturne (npr. silicijske) in programske (algoritmične) vnaprej-determiniranosti ali za-vselej-determiniranosti.

Drugo je seveda vprašanje vrednot mišljenja, ki so tradicionalno in predvsem tradicionalno utemeljene. Te vrednote priznavajo pretežno filozofsko in ustaljeno znanstveno pojmovanje: informacijsko vrednotenje postaja za njih moteče, ker vstopa bistveno v metafiziko stvari filozofije in znanosti, njihovih pojmov in pojmovanj teh pojmov kot informacijskih stvari. Informacija, ko informira, postavlja predvem vprašanje o »kako« predmeti, stvari informirajo znotraj in navzven, npr. v uporabniški prostor predmetov. To vprašanje se postavlja tako za fizične, tehnološke kot za mentalne, mišljenske stvari. Razširjeno doumevanje informacije upošteva prav vrednote te vrste, njihovo nastajanje v umu in prenos tega v vsakdanjo rabo.

Stvar kot informacija se pomika vselej še v pozicijo opazovanja in s svojo atitudo opazovanja razvija metafiziko stvari. To velja npr. za stvari (entitete) v korteksih, ki so videzi (Erscheinungen) realnih stvari samih (na sebi). Tudi prihodnji informacijski stroj bo le neke vrste tehnološki korteks vendar ne zgolj algoritmični, temveč bistveno bolj informacijski. Njegova stvar in metafizika bosta na področju človekovih artefaktov še prilagodljivejši, prožnejši in predvsem uporabnejši ali vobče bliže realnosti.

Slovstvo

[MP] Aristotle, *Metaphysics*, The University of Michigan Press, Ann Arbor, Mi (1960).

[BSN] Nottebohm, F., *From Bird Song to Neurogenesis*, Scientific American (1989) 2, 56-61.

[PH] Gadamer, H.-G., *Philosophical Hermeneutics*, University of California Press, Berkley, Ca (1977).

[SZ] Heidegger, M., *Sein und Zeit*, Max Niemeyer Verlag, Tübingen (1986).

[WIT] Heidegger, M., *What is a Thing?* Regnery/Gateway, South Bend, In (1967).

[TTI] Landman, F., *Towards a Theory of Information*, GRASS Series, Foris, Dordrecht (1988), Holland.

[AC] Maturana, H.R. and F.J. Varela, *Autopoiesis and Cognition*, D. Reidel Publ. Co., Dordrecht (1972), Holland.

[GHR]Senc, S., *Grčko-hrvatski rječnik*, Kr. hrv.-slav.- dalm. zem. vlade, Zagreb (1910).

[UAI] Železnikar, A.P., *Understanding as Information*, Informatica **14** (1990) *3*, 8-30.

[OWI] Železnikar, A.P., *On the Way to Information 1*, Slovensko društvo Informatika, Ljubljana (1990).

[II] Železnikar, A.P., *Informacijski izreki*, (knjiga v pripravi), Ljubljana (1991).

Opomba. Ta prispevek je izvleček iz knjige »Informacijski izreki«, in sicer iz treh (od skupaj 29) poglavij, ki obravnavajo informacijsko fenomenologijo. Avtor se opravičuje, ker zaradi tega snov v tem prispevku za marsikateri okus ni vsebinsko zaokrožena. Prispevek je zasebno avtorsko delo in ga ni dovoljeno uporabljati ali reproducirati brez pismenega potrdila. Izjema so le kratki citati v okviru kritičnih in preglednih razprav. Za to informacijo je pristojen avtor sam.

ALGORITEM OHLAJANJE (SIMULATED ANNEALING)

INFORMATICA 2/91

Keywords: combinatorial optimization, probabilistic algorithm, Simulated Annealing, randomized local search

Janez Žerovnik
Inštitut za matematiko, fiziko in mehaniko
Jadranska 19, 61111 Ljubljana
Slovenia, Yugoslavia

Povzetek: V preglednem sestavku opišemo popularni algoritem za reševanje splošnega problema kombinatorične optimizacije, ki ga imenujemo Ohlajanje (angl. Simulated Annealing, oznaka $\mathcal{SA}$). Sledi pregled literature in izrek o asimptotskem obnašanju algoritma v primerjavi s preprostim algoritmom, inačico algoritma lokalna optimizacija ($\mathcal{RLS}$). Posplošitev izreka velja tudi za posplošeni algoritem, ki pokriva vse znane paralelizacije algoritma $\mathcal{SA}$.

Abstract: ALGORITHM SIMULATED ANNEALING. The algorithm Simulated Annealing ($\mathcal{SA}$ for short) for solving the general problem of combinatorial optimization has received considerable attention in the literature in the last years. A survey of a number of both theoretical and practical results is given. A theorem, which shows that the asymptotic behavior of the Simulated Annealing algorithm is worse than the asymptotic behavior of the randomised local search algorithm ($\mathcal{RLS}$ for short), is discussed [Žero88]. A generalization of the theorem holds for a very general algorithm which covers many known paralelizations of the $\mathcal{SA}$ algorithm [BFZ89a].

1. Uvod

V teoretičnem računalništvu (ali algoritmiki, kot bi tej veji matematike rekel Knuth [Knut81]) je prevladalo mnenje, da intuitivnima pojmoma lahkih in težkih problemov ustrezata razreda P in NP. V razredu P so problemi, za katere obstajajo polinomski algoritmi. Število korakov, ki jih polinomski algoritem potrebuje za rešitev problema, ki ga lahko opišemo s podatki velikosti n (bitov, besed ali cifer) je omejeno z neko polinomsko funkcijo n-ja. V razredu NP so problemi, za katere obstajajo polinomski algoritmi, ki za dani problem in dano rešitev preverijo, ali je rešitev ustrezna. Očitno torej velja P$\subseteq$NP. V razredu NP je posebej odlikovana skupina NP-polnih problemov. Za NP-polne probleme velja naslednje: če bi obstajal polinomski algoritem za reševanje enega od njih, potem bi obstajali polinomski algoritmi za vse probleme iz razreda NP. Krajše bi to zapisali P=NP. Vprašanje, ali je P$\neq$NP je verjetno najbolj znan odprti problem teoretičnega računalništva. Po skoraj dvajsetih letih brez odgovora vedno bolj prevladuje mnenje, da je verjetno razred P prava podmnožica razreda NP. Za uvodno branje o časovni zahtevnosti algoritmov je še vedno odličen vir že klasična knjiga [GaJo79], v Slovenščini pa na primer [HoUl86] ali [KLPP86].

Najboljši znani algoritmi za NP-polne probleme so eksponentni, zanje je število potrebnih korakov v najslabšem primeru navzdol omejeno z eksponentno funkcijo velikosti problema. Zaradi hitre rasti eksponentnih funkcij to običajno pomeni, da ne znamo dobiti točnih rešitev že za sorazmerno majhne velikosti problemov. Če se sprijaznimo s predpostavko P$\neq$NP, potem je smiselno raziskovati algoritme, ki bodo v razumno kratkem času dajali rešitve, ki bodo ustrezali določenim zahtevam. Aproksimacijski algoritmi na primer dajejo rešitve, za katere je znano, koliko največ utegnejo odstopati od optimalne rešitve. V zadnjem času so sorazmerno popularni verjetnostni algoritmi. Algoritmu rečemo *verjetnostni*, če nekateri koraki niso vnaprej natanko določeni s podatki, in rezultat pri istih podatkih torej ni nujno enak pri vsakem teku algoritma. Članki verjetnostnih algoritmih so na primer [Wels83, Maff86, Frie79, Rabi76]. Pri verjetnostnih algoritmih imamo lahko različne garancije kvalitete, na primer:

- algoritme, ki vedno dajo optimalno rešitev in imajo v povprečju polinomsko časovno zahtevnost,
- algoritme s polinomsko časovno zahtevnostjo, ki dovolj pogosto dajo dobre rezultate,
- polinomske algoritme z omejeno verjetnostjo napake pri negativnem odgovoru in zanesljivim (verjetnost napake 0) pozitivnim odgovorom,
- algoritme, pri katerih se verjetnost izračuna optimalne rešitve s časom približuje 1.

Med verjetnostnimi algoritmi je bil v zadnjem času deležen precej pozornosti algoritem $\mathcal{SA}$. Samo v tem članku med literaturo navajamo preko 30 referenc o tem algoritmu, bibliografija pa s tem še zdaleč ni izčrpana, saj se članki na to temo še vedno pojavljajo [Sasa91]. Za ilustracijo omenimo, da v oglasu za knjigo [John90] omenjajo preko 300 člankov o $\mathcal{SA}$! O popularnosti priča tudi veliko število imen, ki jih navajajo v uvodu ene od knjig o tem algoritmu [LaAr87]: Monte Carlo annealing, statistical cooling, probabilistic hill climbing, stochastic relaxation ali probabilistic exchange algorithm. Leta 1953 je Metropolis s sodelavci [MRRT53] uporabil računalnik za simulacijo fizikalnega modela ohlajanja snovi. Kasneje so Kirckpatrick s sodelavci [KiGV83] in neodvisno Černy (Czerny) [Čern84] opazili analogijo med reševanjem problema kombinatorične optimizacije in Metropolisovo simulacijo fizikalnega modela. Verjetno je prav zaradi duhovite analogije algoritem hitro postal široko poznan in nekoliko 'moderen'. Večina

avtorjev kot vir za $\mathcal{SA}$ citira članek [KiGV83], zaradi pravičnosti pa velja omeniti, da so analogijo med statistično mehaniko in optimizacijo (zveznih) funkcij opazili že prej Khačaturjan, Semenovskaja, Vainstein [KhSV81] in Pincus [Pinc70], vendar sta članka ostala brez širšega odmeva. V nadaljevanju bomo povzeli izrek, da algoritem 'konvergira'. Točneje: pri določenih pogojih (dovolj počasno ohlajanje) za pripadajočo (časovno) nehomogeno Markovsko verigo obstaja limitna porazdelitev, v kateri imajo pozitivno verjetnost natanko globalno optimalna stanja (optimalne rešitve). V praksi je ohlajanje seveda vedno 'prehitro'. V velikem številu poročil o poskusih (glej pregled v nadaljevanju) poročajo o dobrih rezultatih. To sicer včasih pomeni, da so dobljene rešitve dotlej najboljše znane za nekatere referenčne primerke nekaterih problemov, vendar gre to vedno na račun zelo velikega računskega časa. V literaturi najdemo celo vrsto teoretičnih analiz algoritma. Tukaj ob knjigah [LaAr87, Laar88] in članku [MiRS86], po katerem povzemamo izreke o konvergenci algoritma, omenimo še enega. Sasaki in Hajek v [SaHa88] pokažeta, da je za problem maksimalnega prirejanja (angl. maximum matching) cel razred algoritmov tipa $\mathcal{SA}$ gotovo več kot polinomske časovne zahtevnosti.
Algoritma $\mathcal{SA}$ in $\mathcal{RLS}$ sta definirana (glej razdelka 4. in 2.) dovolj splošno, da ju lahko uporabimo za reševanje kateregakoli problema kombinatorične optimizacije, zato ju je smiselno primerjati. Glavna zamera algoritmu $\mathcal{RLS}$ (oziroma notranji zanki algoritma) je to, da se 'ujame' v lokalnih optimumih. Ko začnemo z novo začetno rešitvijo v nekem smislu zavržemo vso informacijo, ki smo jo prigarali z dotedanjim računanjem (razen seveda dotlej najboljše dobljene rešitve). Če dovolimo tudi korake 'navzgor' (v smeri povečanja stroškovne funkcije) se algoritem ne ustavi več v lokalnih optimumih. Tako dobimo takoimenovane 'plezalne' (angl. hill climbing) algoritme, med katere uvrščamo tudi algoritem $\mathcal{SA}$. Na račun možnosti plezanja navzgor pa se sicer navidezno še vedno zelo enostavni algoritem precej zaplete, saj je potrebno poiskati prave vrednosti parametrov, ki kontrolirajo, kdaj in koliko dovolimo korake 'navzgor'. Pri algoritmu $\mathcal{SA}$ kontrolnemu parametru običajno rečemo temperatura. Problem, kako temperaturo zniževati, da bomo dobili dobre rezultate, ni enostaven in vse kaže, da je odgovor precej odvisen tudi od tipa problema, ki ga želimo reševati.
Primerjava algoritmov $\mathcal{SA}$ in $\mathcal{RLS}$ je po [LaAr87, Laar88] odprt problem. V nadaljevanju bomo pregledali nekaj poročil o eksperimentih z obema algoritmoma, ki ne dajo enoznačnega odgovora na gornje vprašanje. Tu bomo pokazali (Izrek 2), da je algoritem $\mathcal{SA}$ asimptotsko slabši od lokalne optimizacije in tako deloma odgovorili na gornje vprašanje.

2. Splošni problem kombinatorične optimizacije

Mnoge diskretne optimizacijske probleme lahko gledamo kot posebne primere *splošnega problema kombinatorične optimizacije*, ki ga bomo definirali v tem razdelku. V naslednjem razdelku bomo dodali še konkretni primer, problem trgovskega potnika.
Dana naj bo (končna ali števna) množica objektov $\mathbf{S}$. Elementom $\mathbf{S}$ rečemo *dopustne rešitve* ali *stanja*. Množica $\mathbf{S}$ je lahko podana opisno, včasih pa se moramo zadovoljiti s tem, da je dan algoritem, ki konstruira elemente iz $\mathbf{S}$. V obeh primerih bomo privzeli, da obstaja verjetnostni algoritem $\mathcal{R}$, ki generira elemente iz $\mathbf{S}$ in ima naslednjo lastnost: za poljubni objekt $o \in \mathbf{S}$ je verjetnost dogodka, da je $\mathcal{R}$ zgeneriral ravno o, pozitivna. Formalno

$$P\{\mathcal{R} = o\} = p_o \geq \min_{o \in \mathbf{S}} p_o > 0. \qquad (1)$$

Opomba: če za trenutek pozabimo na (daljši ali krajši) računski čas, lahko rečemo, da je $\mathcal{R}$ slučajna spremenljivka z zalogo vrednosti $\mathbf{S}$.
Lep posebni primer dobimo, če je množica $\mathbf{S}$ končna in če $\mathcal{R}$ generira vse elemente množice $\mathbf{S}$ z enako verjetnostjo.
Na množici $\mathbf{S}$ je definirana *stroškovna funkcija* $c : \mathbf{S} \rightarrow \mathbf{C}$. Za množico vrednosti $\mathbf{C}$ lahko vzamemo na primer množico naravnih $\mathbb{N}$ ali realnih števil $\mathbb{R}$.
Naloga je poiskati minimum stroškovne funkcije c na množici objektov $\mathbf{S}$.
Povzemimo:

Definicija 1 *Splošni problem kombinatorične optimizacije je družina primerkov. Primerek (splošnega) problema kombinatorične optimizacije je par* $(\mathbf{S}, c)$, *kjer je* $\mathbf{S}$ *množica (množica dopustnih rešitev),* c *pa je preslikava* $S \rightarrow \mathbf{C}$ *(stroškovna funkcija). Treba je poiskati tak* $o_{min} \in \mathbf{S}$, *da velja*

$$c_{min} = c(o_{min}) = \min_{o \in \mathbf{S}} c(o) \qquad (2)$$

Problem iskanja maksimuma definiramo analogno, je pa trivialno ekvivalenten problemu minimuma na isti množici $\mathbf{S}$ s stroškovno funkcijo $\tilde{c} = -c$.
V [KLPP86] pravkar definirane pojme imenujejo nekoliko drugače. Primerku problema rečejo naloga, stroškovna funkcija pa je namenska ali ciljna funkcija.
V tem delu se bomo omejili na probleme s končno množico $\mathbf{S}$. Čeprav končna, pa je množica $\mathbf{S}$ pogosto zelo obsežna (recimo velikosti, ki je eksponentna funkcija velikosti problema). V primeru trgovskega potnika je kar $n!$ permutacij mest, če je vsak par mest povezan.
Zaradi obsežnosti pregledovanje vse množice $\mathbf{S}$ običajno ne pride v poštev. Pri NP–polnih problemih je pregled množice $\mathbf{S}$ algoritem eksponentnega reda. Doslej še nikomur ni uspelo odkriti algoritma, ki bi bil bistveno hitrejši (ki bi imel časovno zahtevnost manj kot eksponentnega reda). Ker verjetno velja hipoteza P≠NP, je za to tudi kaj malo upanja.
Pri obravnavi splošnega problema kombinatorične optimizacije $(\mathbf{S}, c)$ bomo privzeli, da obstaja vsaj en verjetnostni algoritem $\mathcal{R}$ $(\mathcal{R} : \emptyset \rightarrow \mathbf{S})$ in vsaj en verjetnostni algoritem $\mathcal{N}$ $(\mathcal{N} : \mathbf{S} \rightarrow \mathbf{S})$. Verjetnostni algoritem V:$A \rightarrow B$ v našem primeru realizira neko preslikavo. Rezultat je element ciljne množice B, porazdelitev verjetnosti, da bo izračunan konkretni element pa je v splošnem odvisna od podatka, elementa množice A.
Množico $\mathbf{N}(x) = \{y | P(\mathcal{N}(x) = y) > 0\}$ imenujemo *soseščina* točke $x \in \mathbf{S}$. Z algoritmom $\mathcal{N}$ na naravni način vpeljemo razdaljo na množico $\mathbf{S}$, ki inducira neko topologijo. Pogosto je za dani problem mogoče na več načinov smiselno definirati pojem soseščine. V takih primerih nam različne izbire v splošnem dajo različne topologije na množici $\mathbf{S}$, s tem pa tudi različno obnašanje splošnih algoritmov, ki jih bomo definirali v nadaljevanju. Nas bo tu poleg topologije seveda zanimala tudi učinkovitost (hitrost) algoritma $\mathcal{N}$. Običajno lahko pričakujemo, da bomo imeli ali hiter algoritem $\mathcal{N}$ in "grdo" topologijo (z mnogimi lokalnimi ekstremi) ali pa obratno, "lepo" topologijo, vendar počasni algoritem $\mathcal{N}$. Za primerjavo različnih splošnih algoritmov

je pomembno tudi razmerje med časovnima zahtevnostima algoritmov $\mathcal{N}$ in $\mathcal{R}$.
Brž ko privzamemo, da imamo dan algoritem $\mathcal{N}$ in z njim topologijo, lahko definiramo *lokalni minimum* na S:

o_l je lokalni minimum, če velja $c(o_l) < c(o) \quad \forall o \in N(o) - \{o_l\}$

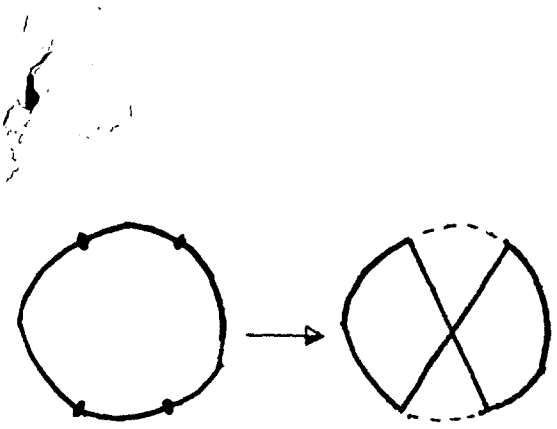
Analogno lahko definiramo *lokalni maksimum*.
Za reševanje splošnega problema kombinatorične optimizacije je znan enostavni algoritem, ki ga bomo imenovali algoritem lokalna optimizacija (angl. randomised local search $\mathcal{RLS}$ [Maff86], neighborhood search, iterative improvement [LaAr87, Laar88], multistart algorithm [Gida85]).

Algoritem $\mathcal{RLS}$: (lokalna optimizacija)
repeat
  generiraj slučajno začetno stanje
  repeat
    poišči soseda
    if je boljši **then** premakni se
  until ni boljšega soseda
until preveč korakov

Enostaven premislek nas pripelje do ugotovitve, da notranja **repeat until** zanka konča v lokalnih minimumih. V nobenem primeru v notranji zanki algoritem ne more iz lokalnega minimuma. To, da se 'ustavi' v lokalnih ekstremih, je tudi glavna zamera, ki jo temu algoritmu namenjajo kritiki [Laar88]. Argument je tembolj tehten, kolikor je $\mathcal{N}$ učinkovitejši od algoritma $\mathcal{R}$.
O časovni zahtevnosti tako splošnega algoritma se ne da povedati veliko. Prav mogoče je, da je celo samo eno iskanje lokalnega optimuma potreben nepolinomski računski čas. Zadostuje, da so okolice dovolj 'bogate'. O tem nas prepriča trivialni primer: vedno je mogoče definirati soseščino tako, da so vse dopustne rešitve dosegljive ena iz druge. Za $\mathcal{N}$ vzamemo kar $\mathcal{R}$ iz definicije problema kombinatorične optimizacije. V tem primeru potrebujemo za pregled ene same okolice $O(|S|)$ časa! (Za tolažbo pa bomo optimalno rešitev zanesljivo dobili.)
Zanimivo je, da celo za znano '2-opt' soseščino pri problemu trgovskega potnika časovna zahtevnost ene lokalne optimizacije ni znana [LITT89]. Menda je Kern [Kern86] nedavno za nekatere probleme trgovskega potnika pokazal, da je '2-opt' v povprečju polinomske časovne zahtevnosti.
Pregled različnih posplošitev algoritma $\mathcal{RLS}$ najdemo v [Maff86], kjer med drugim tudi algoritem Ohlajanje ($\mathcal{SA}$) definirajo kot posplošitev algoritma $\mathcal{RLS}$.

3. Primer problema kombinatorične optimizacije

Mnogi diskretni optimizacijski problemi se dajo zapisati v obliki problema kombinatorične optimizacije, bogato zalogo problemov najdemo na primer v [KLPP86, Koza86], verjetno najpopolnejšo zbirko (preko 300) pa v [GaJo79]. V reviji Journal of Algorithms v posebni rubriki že leta zbirajo nove in nove probleme z znano ali pa še neznano časovno zahtevnostjo.
Morda najbolj popularen problem kombinatorične optimizacije (oziroma operacijskih raziskav) je problem trgovskega potnika (angl. traveling salesman problem, TSP).



Slika 1: Korak lokalne optimizacije tipa 2–opt

TSP spada v širši razred transportnih problemov, ki so deležni precejšnje pozornosti v strokovni literaturi. Za ilustracijo povejmo, da pregledni članek iz leta 1983 [BGAB83] navaja kar 699 referenc v zvezi s transportnimi problemi!
Ob mnogih primerih praktične uporabe je verjetno eden od razlogov popularnosti problema trgovskega potnika dejstvo, da ga je zelo enostavno opisati. Imamo neusmerjen graf z utežmi na povezavah (*omrežje*). *Hamiltonov obhod* ali *Hamiltonov cikel* je pot, katere začetna in končna točka sovpadata in ki obišče vsako točko grafa natanko enkrat. Problem je poiskati Hamiltonov obhod z minimalno dolžino, če za dolžino poti vzamemo vsoto uteži na povezavah poti.
V duhu definicije 1 so dopustne rešitve vsi Hamiltonovi obhodi na grafu. Vsak Hamiltonov obhod lahko zapišemo tudi kot ciklično permutacijo. če je graf poln, potem tudi vsaka ciklična permutacija definira Hamiltonov obhod, v splošnem pa to ni res. Stroškovna funkcija je

$$c(\pi) = \sum_{i=1}^{n} d(i, \pi(i)) \qquad (3)$$

kjer je π permutacija $\in S_n$, $\{1, \ldots, n\}$ so oznake točk, $d(i,j)$ pa je utež na povezavi (i,j).
Pogosto je uporabljana lokalna optimizacija tipa r–opt. Sosednje stanje k danemu stanju dobimo tako, da vzamemo r povezav in jih zamenjamo z novimi. Primer za 2–opt je na sliki 1.
Ker z rastočim r časovna zahtevnost lokalne optimizacije r–opt narašča, so najpogosteje uporabljane 2–opt in 3–opt, ali pa optimizacija, ki se sproti odloča za r. Kot najboljšo pa v [LLRS85] priporočajo Or–opt.
Problem trgovskega potnika je NP–poln. Tu samo omenimo, da so tudi mnogi zanimivi posebni primeri še vedno NP–polni problemi, na primer: simetrični TSP, Evklidski TSP, TSP na polnem omrežju, TSP za katerega velja trikotniška neenakost. Še več, že odločitveni problem, ali ima dani graf Hamiltonov cikel, je NP–poln problem.
Pogosto rešujemo problem na polnem omrežju, saj s tem ne izgubimo splošne uporabnosti algoritma. Vedno namreč lahko povezave, ki jih nočemo uporabljati, obtežimo z zelo veliko utežjo (na primer večjo kot vsota vseh dovoljenih uteži). Če je potem dobljena optimalna rešitev prevelika, vemo, da je bila uporabljena kakšna prepovedana povezava.
V primeru, ko rešujemo problem na splošnem grafu, bi bilo naravno vzeti za dopustne rešitve Hamiltonove obhode grafa. V tem primeru bi bil že algoritem $\mathcal{R}$ časovno zelo zahteven, saj bi moral reševati NP–poln problem. To je pomemben razlog za odločitev, da rešujemo problem na polnem omrežju.

4. Algoritem $\mathcal{SA}$

Eden od razlogov za popularnost algoritma $\mathcal{SA}$ je analogija z modelom procesa ohlajanja snovi iz statistične mehanike. Tu analogije ne bomo opisovali, omenimo le referenco [KiGV83].
S stališča kombinatorične optimizacije je algoritem ohlajanje ($\mathcal{SA}$) zelo enostaven. Za razliko od lokalne optimizacije so možni tudi premiki v smeri poslabšanja stroškovne funkcije. Verjetnost takega koraka je odvisna od parametra T, ki mu po analogiji s fizikalnim modelom rečemo temperatura.

Algoritem Ohlajanje ($\mathcal{SA}$):
generiraj slučajno začetno stanje
nastavi začetno temperaturo T
repeat
  znižaj temperaturo T
  slučajno izberi soseda
  if sosed je boljši **then** premakni se
  else (* sosed je slabši*)
    premakni se z verjetnostjo $p = p(T)$
until preveč korakov

Označimo s p verjetnost dogodka, da je v algoritmu sprejet korak, ki poslabša vrednost stroškovne funkcije. Ta verjetnost je odvisna od trenutnega stanja (trenutne dopustne rešitve) in od trenutne vrednosti parametra T. Predpostavili bomo, da je p padajoča funkcija parametra T. Pri danem primerku problema in dani temperaturi T bomo zahtevali, da naj bo p omejena s konstanto $p < P < 1$.
Funkcijska odvisnost je običajno oblike

$$p = \begin{cases} 1 & \Delta C < 0 \\ \exp^{-\Delta C/T} & \Delta C \geq 0 \end{cases} \qquad (4)$$

kjer smo z ΔC označili razliko stroškovnih funkcij starega in novega stanja. Taka definicija je predlagana v [MRRT53] in [KiGV83], in je običajno uporabljena. Edina avtorju znana izjema je definicija Boltzmanovega stroja [AaKo87, AaKo88] kjer uporabljajo

$$p = \left(1 + \exp^{-\Delta C/T}\right)^{-1} \qquad (5)$$

Definicija algoritma $\mathcal{SA}$ je res enostavna, vendar pri implementaciji hitro naletimo na netrivialne probleme. Na primer, kako je treba zmanjševati temperaturo, da bomo dosegli dobre rezultate. Vemo, da prehitro zniževanje temperature ne zagotavlja več 'konvergence' algoritma [MiRS86]. Ali znamo zniževanje temperature dobro določiti vsaj za kakšen poseben problem? V [LaAr87, Laar88] in v [LaDe88] predlagajo polinomske strategije ohlajanja, ki se dobro odrežejo na nekaterih preiskušenih primerih. (Pri tem se seveda morajo zadovoljiti z dobrimi namesto optimalnih rešitev.) Tu se s tem problemom ne bomo podrobneje ukvarjali.
Zniževanje temperature definiramo lahko tako, da povemo, kako je T odvisna od števila korakov. Poseben primer (in ekvivalenten !) je, da povemo, koliko korakov naredimo pri neki temperaturi. Odvisnost temperature od časa v tem primeru podamo z zaporedjem parov (temperatura, število korakov), na primer

$$(T_1, m_1), (T_2, m_2), \ldots, (T_k, m_k), \ldots$$

kar pomeni: naredi m_1 korakov pri temperaturi T_1,... naredi m_k korakov pri temperaturi T_k,...itd. Kot že omenjeno, mora temperatura padati počasi, če želimo, da algoritem z verjetnostjo 1 najde optimalno rešitev (v končnem, toda ne omejenem številu korakov) [MiRS86].
V nadaljevanju bomo predpostavili, da temperatura T konvergira proti 0 z naraščajočim številom korakov.

$$\lim_{n \to \infty} T(n) = 0 \qquad (6)$$

5. Računski rezultati z algoritmom $\mathcal{SA}$ (pregled literature)

V tem razdelku bomo s kratkimi opombami pospremili poročila o testiranju algoritma $\mathcal{SA}$ na različnih problemih. Pregled še zdaleč ni popoln, zajema pa (verjetno) večino 'klasičnih' člankov in nekaj poročil, ki jih je avtor bolj ali manj slučajno dobil v roke.
V že večkrat omenjenem članku Kirckpatrick s sodelavci [KiGV83] preizkusi algoritem $\mathcal{SA}$ na problemu trgovskega potnika (TSP) in na dveh problemih, ki se pojavita pri načrtovanju integriranih vezij (Cell Placement, Wiring). Članek je populariziral algoritem. Uspešna uporaba tu in še v celi vrsti poročil pomeni, da so bile dosežene dobre ali celo dotlej najboljše znane rešitve za kakšen primerek kakšnega problema. O času, potrebnem za računanje ne poročajo ali pa priznavajo, da je algoritem zelo potraten.
Nasploh običajno (z izjemami, ki jih navajamo v pregledu) velja splošna ugotovitev (ki jo v [AnMS87] (stran 8) povzamejo po [JAMS85] in [GoSk86]), da je $\mathcal{SA}$ tipično slabši od algoritmov, ki so posebej prilagojeni konkretnemu problemu.
Bonomi in Lutton [BoLu84] z algoritmom računata 'skoraj optimalne' rešitve za TSP v $O(n^2)$ časa. Eksperimentalno ugotovita, da se algoritem $\mathcal{SA}$ obnese bolje od lokalne optimizacije. Na žalost uporabita staro verzijo 2-opt algoritma. Ista avtorja v [LuBo86] algoritem uporabita na problemu minimalno uteženo Evklidsko prirejanje (angl. minimum weighted perfect Euclidean matching). Opazita, da je potrebni čas (za osnovno verzijo algoritma $\mathcal{SA}$) 'ogromen' (stran 195, zadnji odstavek). Še eno poročilo omenjenih avtorjev [BoLu86] se ukvarja z aplikacijo algoritma na kvadratni problem prirejanja (angl. Quadratic sum assignement problems). Algoritem $\mathcal{SA}$ je predlagan kot linearni aproksimacijski algoritem.
O poizkusih na kvadratnem problemu prirejanja (Quadratic assignement problem) poročata Burkhardt in Rendl [BuRe84]. Iz rezultatov je videti, da $\mathcal{SA}$ ni bil bistveno boljši od QAPH4B algoritma (ki je v resnici lokalna optimizacija), posebej če upoštevamo porabljeni čas. Uporabljena je tudi različica z več ponovitvami.
Grover [Grov87] poroča o uporabnem programu, ki prinaša lepe prihranke (precej odstotkov) na problemu razporejanja čipov (angl. Standard Cell Placement), uporabljanem v načrtovanju integriranih vezij (VLSI design). Začne vedno z dobro začetno rešitvijo.
Poročili [DoSS87, DoSk88] ugotavljata, da je algoritem $\mathcal{SA}$ na problemu razbitja grafa (graph partitioning) zelo potraten, so pa dobili dobre rešitve v primerjavi z nekaterimi drugimi algoritmi.
V [AnMS87] (stran 8) sta omenjeni dve eksperimentalni študiji. V [JAMS85] ugotavljajo, da je z $\mathcal{SA}$ sicer mogoče dobiti dobre rešitve, vendar je bil algoritem na problemih razcep števil (number partitioning), barvanje grafa (graph colouring) in trgovski potnik (TSP) praviloma slabši od tradicionalnih algoritmov za te probleme. V primeru

problema razbitja grafa so bili dobljeni rezultati ugodni za algoritem $\mathcal{SA}$. V drugem poročilu [GoSk86] ugotavljajo, da za problem trgovskega potnika in "p-mediane" obstajajo algoritmi, ki so boljši od $\mathcal{SA}$.

V članku [LALW89] Laarhoven, Aarts, Lint in Wille poročajo o doslej najboljših rešitvah za problem trdnjav (angl. Footbal-pool problem) na 6, 7 in 8 parih. Prej je najboljšo znano rešitev za $n = 6$ našel Wille [Will87], prav tako z algoritmom $\mathcal{SA}$.

Za konec omenimo še nekaj poročil, v katerih primerjajo algoritem $\mathcal{SA}$ z lokalno optimizacijo. V [LaAr87] povzemajo primer [LuMe86], za katerega je dokazano, da je $\mathcal{SA}$ boljši od $\mathcal{RLS}$ (pričakovano število korakov $\mathcal{SA}$ je manjše od pričakovanega števila korakov algoritma $\mathcal{RLS}$). Ta rezultat ne nasprotuje našemu izreku 2, saj je primer narejen za verzijo $\mathcal{SA}$ s konstantno temperaturo, torej ne ustreza pogojem izreka (oziroma naši definiciji algoritma $\mathcal{SA}$).

Rahin [RaSh89] poroča o primerjavi lokalne optimizacije in $\mathcal{SA}$ na problemu razbitja grafa. Za majhne n se je lokalna optimizacija izkazala za boljšo: rezultat je bil boljši in čas krajši. Uporabljen je bil stari Lin-Kerninghamov algoritem za lokalno optimizacijo.

Lam in Delosme [LaDe88] poročata o testiranju algoritma $\mathcal{SA}$ z originanim načinom ohlajanja. V nekaterih primerih se je algoritem izkazal za boljšega od Lin-Kerninghamove in od Karpove hevristike za problem trgovskega potnika. Na problemu razbitja grafa je bil pri nekaterih načinih generiranja slučajnih primerkov problema algoritem boljši, pri drugih pa slabši od hevristike Fiduccia in Mattheysesa.

Kategorično v prid $\mathcal{SA}$ interpretirajo rezultate v [JAMS85], kjer pa je to storjeno izrazito 'nepošteno'. 100 poskusov z algoritmom $\mathcal{SA}$ je primerjano s 100 poskusi lokalne optimizacije ($\mathcal{RLS}$), to pa pomeni, da je bil čas porabljen za $\mathcal{SA}$ kar približno 300-krat daljši! (6 minut za $\mathcal{SA}$ proti 1 sekundi za $\mathcal{RLS}$).

Po drugi strani v zborniku [AnMS87](stran 95) omenjajo referenco [RiTi85] kjer so eksperimentalno ugotovili, da je pri globalni optimizaciji na kompaktni množici $\mathbf{S} \subset \mathbb{R}^k$ pri pogoju, da leži optimalna točka v notranjosti $\mathbf{S}$ med znanimi najboljša naslednja metoda:

1.generiraj nekaj slučajno izbranih točk (z enakomerno porazdelitvijo);
2.naredi lokalno iskanje na vsaki od njih
3.zapomni si dotedaj najboljšo rešitev

(Torej analogno algoritmu $\mathcal{RLS}$ za diskretni primer!)

Pred zaključkom pregleda povzemimo komentar iz [LaAr87]. Večina poskusov z algoritmom $\mathcal{SA}$ je bila narejena z enostavnim ohlajanjem. Rezultati bi bili verjetno boljši, če bi uporabili bolj "zvite" načine ohlajanja, kot jih (na primer) predlagajo v [LaAr87, Laar88]. Po drugi strani je res, da tudi algoritmi, s katerimi $\mathcal{SA}$ primerjajo niso vedno najboljši od znanih. Pri tovrstnih poskusih je na žalost vedno tako, da je zelo težko doseči popolnoma pošteno primerjavo. Verjetno bo za dokončno splošno priznano sodbo potrebno še precej primerjalnih študij.

Poskusimo navezati nekatere rezultate poskusov na teoretični rezultat iz naslednjega razdelka: Upanje, da je mogoče rezultat izreka 2 podkrepiti še s kakšnim praktičnim rezultatom, sta naslednji poročili. Dueck in Scheuer [DuSc88] sta testirala algoritem $\mathcal{TA}$ (treeshold accepting), ki je nekoliko popravljena lokalna optimizacija. Namesto verjetnostnega kriterija ta algoritem sprejema poleg korakov navzdol tudi korake, ki 'ne gredo preveč navzgor'. Algoritem torej 'preskoči' majhne 'hribčke' in 'dolinice', ki so za običajno lokalno optimizacijo usodni. Doseženi rezultati na primerku problema trgovskega potnika s 442-mesti so najboljši doslej (boljši kot tisti, dobljeni z algoritmom $\mathcal{SA}$). Poročajo tudi o sorazmerno kratkem času računanja.

K temu dodajmo še primerjavo [LeBi89] med algoritmom $\mathcal{SA}$ in 'quenching' algoritmom (zelo hitro ohlajanje), ki se je prav tako izkazalo za boljše od počasnega ohlajanja (pri ustreznem številu ponovitev za hitrejši algoritem, seveda). Če poskusimo iz tega potegniti nekakšen nasvet, bi lahko rekli takole: Običajno se ne izplača predolgo časa porabiti z 'zelo premetenim' algoritmom. Bolje je večkrat začeti prav od začetka, če 'dovolj dolgo' nimamo zaželjenega uspeha. Obravnava vprašanja, kaj v konkretnih primerih pomeni 'dovolj dolgo' presega okvire tega dela. Tukaj omenimo le referenci [BoRV87] in [Laar88], ki obravnavata ustavitvena pravila za nekatere verjetnostne hevristike.

6. Konvergenca algoritma $\mathcal{SA}$

Konvergenco algoritma $\mathcal{SA}$ obravnavajo v člankih [GeGe84, Gida85] in [HaSa89], v [GeMi87] pa so omenjeni tudi izreki Hajeka in Tsitsiklisa. Model izvajanja algoritma je (časovno) nehomogena Markovska veriga, kajti prehodne verjetnosti se s časom spreminjajo. V tem razdelku bomo povzeli po enem od omenjenih virov izrek o konvergenci algoritma $\mathcal{SA}$.

Izrek 1 *[MiRS86] Če parameter T pada dovolj počasi, potem iz kakršnegakoli začetnega stanja algoritem $\mathcal{SA}$ z verjetnostjo 1 konča v enem od globalnih optimumov, če mu dovolimo narediti neskončno mnogo korakov.*

Ali, ekvivalentno: Če parameter T pada dovolj počasi, potem iz kakršnegakoli začetnega stanja algoritem $\mathcal{SA}$ z verjetnostjo poljubno blizu 1 konča v enem od globalnih optimumov, če mu dovolimo narediti dovolj korakov.

7. Primerjava algoritmov $\mathcal{SA}$ in $\mathcal{RLS}$

V tem razdelku navajamo izrek, ki pravi, da je algoritem $\mathcal{SA}$ asimptotsko slabši od algoritma $\mathcal{RLS}$. Še več, iz dokaza bo sledilo, da je celo enostavno ponavljanje neodvisnih generiranj dopustnih rešitev asimptotsko uspešnejši algoritem kakor algoritem $\mathcal{SA}$ [Žero88].

Uporabljali bomo naslednje oznake:

- N je število stanj (moč množice dopustnih rešitev)
- K_1 označuje število stanj, iz katerih vse poti, ki gredo samo navzdol, končajo v enem od globalnih optimumov.
- K označuje število stanj, iz katerih obstaja pot, ki gre samo navzdol in konča v enem od globalnih optimumov. Očitno $K \geq K_1 > 0$. Da se izognemo trivialnim primerom privzemimo še $N > K$.
- R je dolžina najdaljše poti, ki gre samo navzdol.
- w je razmerje med časom, potrebnim za generiranje začetne dopustne rešitve (algoritem $\mathcal{R}$) in med časom, potrebnim za generiranje novega stanja, če kakšno stanje že poznamo (algoritem $\mathcal{N}$).

- d je maksimalna stopnja stanja (maksimalno število sosedov) oziroma

$$d = \max_{x \in \mathbf{S}} |\mathrm{N}(x)|$$

- $1-p_i$ je verjetnost, da algoritem ne bo sprejel slabšega stanja v algoritmu $\mathcal{SA}$ pri temperaturi T_i. Lahko bi tudi rekli: verjetnost, da algoritem ne bo šel 'gor'. (Vrednost p_i je odvisna tudi od trenutnega stanja.)
- $q_i = \min\{1 - p_i\}$, kjer minimum izračunamo po vseh možnih stanjih danega primerka problema. Temperatura je tu fiksirana (T_i). q_i je torej spodnja meja za verjetnost dogodka, da pri temperaturi T_i pri danem primerku problema algoritem $\mathcal{SA}$ ne bo sprejel koraka 'gor', v smeri poslabšanja stroškovne funkcije. V dokazu Izreka 2 bomo predpostavili $q_i \to 1$ ko gre $i \to \infty$. Če vzamemo 4 za definicijo p in predpostavimo $T \to 0$ sledi $q_i \to 1$, torej je za algoritem $\mathcal{SA}$ predpostavka izreka izpolnjena.
- n je število korakov, ki jih je algoritem že opravil.

V dokazu privzamemo, da algoritem $\mathcal{R}$ generira vse dopustne rešitve z enako verjetnostjo. (Privzetek deloma poenostavi dokaz, vendar ni bistven za resničnost izrekov.) Dokaz je (presenetljivo) enostaven. Poiščemo spodnjo mejo za verjetnost, da algoritem lokalna optimizacija najde globalno optimalno rešitev v n korakih in zgornjo mejo za verjetnost, da algoritem $\mathcal{SA}$ najde globalno optimalno rešitev v n korakih. Potem primerjamo obe oceni in sledi rezultat (za dokaz glej [Žero88, Žero90]).
Formalno definiramo

$$P_{\mathcal{RLS}}(n) := Pr(\textit{algoritem } \mathcal{RLS} \textit{ je uspel v } n \textit{ korakih}) \tag{7}$$

in

$$P_{\mathcal{SA}}(n) := Pr(\textit{algoritem } \mathcal{SA} \textit{ je uspel v } n \textit{ korakih}) \tag{8}$$

ocenimo

Lema 1

$$P_{\mathcal{RLS}}(n) \geq 1 - \left(\frac{N - K_1}{N}\right)^{\lceil \frac{n}{Rd+w} \rceil} \tag{9}$$

Lema 2

$$P_{\mathcal{SA}}(n) \leq 1 - q_i^{\tilde{m}_i} q_{i-1}^{m_{i-1}} \ldots q_2^{m_2} q_1^{m_1} \left(1 - \frac{K}{N}\right) \tag{10}$$

kjer je $n = \tilde{m}_i + \sum_{j=1}^{i-1} m_j$ *in* $0 < \tilde{m}_i \leq m_i$.

in izpeljemo

Izrek 2 *Obstaja konstanta* n_0, *tako da velja*

$$\forall n > n_0 \Longrightarrow P_{\mathcal{SA}}(n) < P_{\mathcal{RLS}}(n) \tag{11}$$

8. Skoraj optimalne rešitve

Pri reševanju NP–težkih problemov se moramo običajno odpovedati zahtevi po optimalni rešitvi. Izkaže se, da je mogoče enako primerjavo med algoritmoma $\mathcal{SA}$ in $\mathcal{RLS}$ narediti tudi v primeru ko smo zadovoljni tudi s skoraj-optimalnimi rešitvami optimizacijskega problema, ki jih definiramo takole:

Bodi c_{min} strošek optimalne rešitve in bodi c_{max} strošek ene od najslabših rešitev. (torej $c_{min} \leq c \leq c_{max}$ za vsako dopustno rešitev) Izberimo si $\epsilon > 0$. Rešitev s stroškom $c \leq c_{min} + \epsilon(c_{max} - c_{min})$ imenujemo *ϵ-skoraj optimalna* rešitev.
Po definiciji ϵ govori o kvaliteti rešitve na nekako normirani stroškovni funkciji. Na primer za $\epsilon = 0.10$ skoraj optimalne rešitve niso za več kot 10% slabše od optimalne rešitve. Pri $\epsilon = 0$ dobimo optimalne rešitve, pri $\epsilon = 1$ pa kar vse dopustne rešitve.
Tukaj velja opozoriti, da so mnenja, katere rešitve je smiselno proglasiti za skoraj optimalne, deljena. Namesto o skoraj optimalnih rešitvah bomo tu raje govorili o množici *zaželjenih* rešitev, ki je lahko množica skoraj optimalnih rešitev za neki ϵ, lahko pa za množico zaželjenih rešitev vzamemo tudi kakšno drugo podmnožico množice S.
Če povzamemo oznake iz dokaza izreka 2, ponovno pa definiramo K_1, K in pojem 'uspeh algoritma $\mathcal{SA}$', potem lahko dokažemo podoben izrek za verjetnost zadetka nove množice zaželjenih rešitev.
V tem razdelku naj:

- K_1 označuje število stanj, iz katerih vse poti, ki gredo samo navzdol, končajo v eni od zaželjenih rešitev.
- K označuje število stanj, iz katerih obstaja pot, ki gre samo navzdol in konča v eni od zaželjenih rešitev. Očitno $K \geq K_1$.

Tokrat bomo rekli, da je algoritem $\mathcal{SA}$ uspel (najti zaželjeno rešitev) brž ko je dosegel stanje, iz katerega obstaja vsaj ena pot, ki gre samo navzdol in konča v stanju, ki ustreza zaželjeni rešitvi.
Z natanko istim premislekom kot v dokazu izreka 2 bi pokazali, da velja

Trditev 1 *Obstaja konstanta* n_0, *tako da velja*

$$\forall n > n_0 \Longrightarrow \tilde{P}_{\mathcal{SA}}(n) < \tilde{P}_{\mathcal{RLS}}(n) \tag{12}$$

kjer je $\tilde{P}_{\mathcal{A}}(n)$ *verjetnost, da je algoritem* $\mathcal{A}$ *našel zaželjeno rešitev v* n *korakih.*

Trditev lahko zapišemo še nekoliko drugače:

Posledica 1 *Bodi* $\tilde{P}_{\mathcal{SA}}(n_0) < p$ *in* $n_0 > \sum_{i=1}^{k-1} m_i$, *kjer je*

$$k = \min\left\{i; \forall j \geq i : q_j > \left(1 - \frac{K_1}{N}\right)^{\frac{1}{Rd+w}}\right\}$$

Označimo $n_{\mathcal{RLS}} = \min\{n; \tilde{P}_{\mathcal{RLS}}(n) > p\}$ *in* $n_{\mathcal{SA}} = \min\{n; \tilde{P}_{\mathcal{SA}}(n) > p\}$. *Potem je*

$$n_{\mathcal{SA}} > n_{\mathcal{RLS}}$$

Torej je pri dani zahtevani kvaliteti rešitve in pri dani (dovolj veliki) minimalni zahtevani verjetnosti uspeha število potrebnih korakov manjše, če se odločimo za algoritem $\mathcal{RLS}$ namesto za algoritem $\mathcal{SA}$.

9. Vzporedne implementacije algoritma $\mathcal{SA}$

V tem razdelku predstavimo izrek, analogen izreku 2, ki velja za vse (nam) poznane paralelizacije algoritma $\mathcal{SA}$. V literaturi sta znana dva modela paralelizacije algoritma $\mathcal{SA}$. Prvi [BoLu84, FęKO85] temelji na delitvi podatkov,

kar v primeru problema trgovskega potnika pomeni, da cikel, ki predstavlja začetno rešitev, razbijemo na več poti in jih razdelimo procesorjem. Vsak od procesorjev nekaj časa optimizira svoj del poti (lahko si mislimo, da sta začetna in končna točka fiksirani), potem pa si procesorji izmenjajo podatke. S tem je doseženo, da lahko algoritem (s pozitivno verjetnostjo) doseže katerokoli dopustno rešitev. Drugi model vsem procesorjem dodeli nerazdeljeni problem, procesorji na njem vsak zase izvajajo običajni algoritem $\mathcal{SA}$, po določenem času pa se 'sinhronizirajo'. Sinhronizacija tu pomeni, da na osnovi dotlej najboljših rešitev izberejo nove začetne rešitve. Preizkušeni so bili različni kriteriji sinhronizacije, na primer izbira najboljše rešitve, izbira n najboljših rešitev, slučajna izbira, na primer glede na Boltzmanovo porazdelitev, itd. [LaAr87, DoSS87, ABHL86].

V nadaljevanju bomo definirali model, ki zajema vse zgoraj naštete različice kot posebne primere. Za ta splošni model se da pokazati analogni izrek Izreku 2. Ker je mogoče algoritem lokalna optimizacija $\mathcal{RLS}$ naravno implementirati na več procesorjih z optimalnim pospeškom (saj nimamo nobenih izgub zaradi komunikacije oziroma sinhronizacije), ni presenetljivo, da je tudi na več procesorjih $\mathcal{SA}$ asimptotsko slabši od lokalne optimizacije. (Pospešek pri paralelnem izvajanju je definiran s takoimenovanim Gustafsonovim zakonom [Gust88], ki je na glavo postavljena oblika Amdahlovega zakona [Amda67]. Gustafsonov zakon mnogo očitneje pokaže moč vzporednega izvajanja algoritmov, zato je bil seveda lepo sprejet v krogih, ki se s tem ukvarjajo. Na kratko in neformalno povedano zakona povesta naslednje: če imamo na voljo n procesorjev optimalni pospešek pomeni, da bomo porabili n-krat manj časa kot prej. Ali, še lepše, na n procesorjih ob optimalnem pospešku lahko v istem času rešimo n-krat večje probleme.)

V splošnem modelu predpostavljamo, da p procesorjev (neodvisno) izvaja splošni algoritem, imenovali ga bomo $\mathcal{PSA}$, s sinhronizacijsko razporeditvijo (angl. sinchronization schedule). Namesto 'temperature' (ki je v splošnem lahko različna od procesa do procesa) tu uporabimo števec korakov algoritma in verjetnost, da bo algoritem sprejel korak 'gor', je zdaj funkcija števila korakov $p = p(i)$. Zaporedje $0, v_1, v_2, \ldots, v_k, \ldots$ zdaj beremo: naredi v_1 korakov do prve sinhronizacije, potem naredi v_2 do druge sinhronizacije, itd. Oziroma, sinhroniziraj se prvič pri v_1-tem koraku, drugič pri $v_1 + v_2$-tem koraku, itd. Algoritem zapišemo podobno kot prej:

Algoritem $\mathcal{PSA}$:
generiraj slučajno začetno stanje
$i := 0$;
repeat
  repeat
    i:=i+1;
    slučajno izberi soseda
    if sosed je boljši **then** premakni se
    else (* sosed je slabši *)
      premakni se z verjetnostjo $p = p(i)$
  until sinhronizacija
  izberi novo začetno stanje in/ali
  izmenjaj podatke
until preveč korakov

Oba prej omenjena pristopa k paralelizaciji $\mathcal{SA}$ sta posebna primera splošnega algoritma PSA.

Verjetnost koraka 'gor' zdaj ni odvisna od parametra T, ampak od i. Ponovno bomo uporabili iste oznake kot v dokazu izreka 2. Nekoliko drugače bomo definirali samo verjetnost

$$q_i = \min\{ \textit{ verjetnost, da } \mathcal{PSA} \textit{ ne gre 'gor' v fazi } i \textit{ na nobenem od procesorjev}\} \quad (13)$$

kjer minimum izračunamo po vseh možnih stanjih danega primerka problema. Izraz "faza i" tu nadomešča frazo "med sinhronizacijama $i-1$ in i". V dokazu Izreka 3 bomo ponovno predpostavili $q_i \to 1$ ko gre $i \to \infty$. To bo gotovo res, če se bo sistem 'ohlajal', ali, z drugimi besedami, če bo maksimalna temperatura (po vseh procesorjih) kakorkoli šla proti 0.

Verjetnost, da p neodvisnih izvajanj algoritma lokalna optimizacija najde globalni optimum v n vzporednih korakih je očitno omejena z istim tipom spodnje meje kot če imamo samo eno izvajanje.

$$P_{\mathcal{PRLS}}(n) = 1-(1-P_{\mathcal{RLS}}(n))^p \geq 1-(1-(1-Q))^p = 1-Q^p \quad (14)$$

kjer je

$$Q = \left(\frac{N-K_1}{N}\right)^{\lceil \frac{n}{Rd+\omega} \rceil} \quad (15)$$

Analogno kot prej bomo rekli, da je algoritem $\mathcal{PSA}$ uspel, brž ko je vsaj eden od procesorjev našel stanje, iz katerega pelje vsaj ena pot, ki gre samo navzdol, in konča v enem od globalnih optimumov.

Podobno kot prej dobimo

Lema 3 *Za* $n = \sum_{k=1}^{i-1} v_k + \tilde{v}_i$, *kjer je* $0 < \tilde{v}_i \leq v_i$, *velja*

$$P_{\mathcal{PSA}}(n) \leq 1 - q_i^{p\tilde{v}_i} q_{i-1}^{v_{i-1}} \ldots q_2^{v_2} q_1^{v_1} \left(1 - \frac{K}{N}\right) \quad (16)$$

In

Izrek 3 *Obstaja konstanta* n_0, *tako da velja*

$$\forall n > n_0 \Longrightarrow P_{\mathcal{PSA}}(n) < P_{\mathcal{PRLS}}(n) \quad (17)$$

Podrobnosti opuščamo. (Vsebina razdelka in dokaz izreka je rezultat krajšega obiska v laboratoriju za paralelne algoritme na Ecole Normale Superiuere de Lyon in je objavljena v [BFZ89a, BrFZ89, Žero90].)

Opomba: V posebnem primeru, ko vzporedno poganjamo p neodvisnih procesov, lahko zgornjo mejo za algoritem $\mathcal{PSA}$ nekoliko izboljšamo. V tem primeru imamo namreč p neodvisnih poskusov, od katerih je vsak en običajni (zaporedni) tek algoritma $\mathcal{SA}$.

$$P^*_{\mathcal{PSA}} \leq 1 - \left(q_i^{\tilde{m}_i} q_{i-1}^{m_{i-1}} \ldots q_2^{m_2} q_1^{m_1} \left(1 - \frac{K}{N}\right)\right)^p \quad (18)$$

Razliko pojasnimo preprosto: V rezultatu leme, enačba (16), 'manjkajoča' potenca p na faktorju $(1 - \frac{K}{N})$ je posledica naše implicitne predpostavke, da smo vzporedni tek algoritma začeli z enim začetnim stanjem (in ne s p-timi).

Lahko bi se prepričali, da izrek 3 tudi v tem primeru velja.

10. Zaključek

Namen sestavka je bil predstavitev algoritma SA, ki je bil v zadnjih letih deležen precej pozornosti v strokovni javnosti. Pregled teoretičnih in praktičnih rezultatov kaže, da je vrednost algoritma v splošnem še nejasna. Asimptotski rezultat, ki ga predstavimo, implicira, da v vseh primerih algoritem ne more biti najboljši. Verjetno je relativna uspešnost algoritma (glede na tekmece) bistveno odvisna od lastnosti prostora stanj, ki je pri različnih problemih kombinatorične optimizacije lahko bistveno različen. Omenimo tukaj sveži članek [Sasa91], ki dokazuje povezavo med 'gostoto' prostora stanj in mejo za uspešnost Metropolisovega algoritma. Potrebno bo še precej eksperimetalnih študij, preden bo jasno, v katerih primerih se Ohlajanje res izplača.

Zahvala: Članek temelji na delu disertacije, ki je nastala pod mentorstvom prof. Tomaža Pisanskega. Zahvaljujem se mu za pomoč in vzpodbudo. Zadnja verzija članka je nastala med avtorjevim obiskom prof. Wilfrieda Imricha na Montanuniversität v Leobnu.

Literatura

[AaKo87] E.H.L.Aarts, J.H.M.Korst, Boltzmann Machines and Their Applications *PARLE Parallel Architectures and Languages Europe*, Lecture Notes in Comp. Sci. 258, Springer, Berlin 1987, 34-50

[AaKo88] E.H.L.Aarts, J.H.M.Korst, Computations in massively parallel networks based on the Boltzmann machine: A rewiew, *Parallel Computing*, 9 (1988/89) pp. 129-145

[ABHL86] E.H.L.Aarts, F.M.J.Bont, E.H.A.Habers, P.J.M. Laarhoven: Parallel Implementations of the Statistical Cooling Algorithm, *INTEGRATION, the VLSI journal* 4 (1986) pp. 209-238

[Amda67] G.M. Amdahl: Validity of the single-processor approach to achieving large scale computing capabilities. *AFIPS Conference Proceedings, Atlantic City, N.J., Apr.18-20* **30** AFIPS Press, Restn, Va. 1967 pp. 483-485

[AnMS87] G. Andreatta, F. Mason, P. Serafini (eds.), *Advanced School on Stochastics in Combinatorial Optimization*, World Scientific, Singapore 1987

[BGAB83] L. Bodin, B.Golden, A.Assad, M.Ball, Routing and Scheduling of Vechicles and Crews, the State of the Art *Comput & Ops. Res.* **10** (1983) pp. 63-211

[BoLu84] E.Bonomi, J.L.Lutton, The N-city traveling salesman problem: Statistical Mechanics and the Metropolis algorithm, *SIAM rewiev* **26** 1984

[BoLu86] E.Bonomi, J.L.Lutton, The asymptotic behaviour of quadratic sum assignement problems: A statistical mechanics approach, *European journal of Operational Research* **26** 1986 pp. 295-300

[BoRV87] G.E.Boender, A.H.G. Rinooy Kan, C. Vercelis: Stochastic Optimization Methods, v *Advanced School on Stochastics in Combinatorial Optimization*, G. Andreatta, F. Mason, P. Serafini (eds.):, World Scientific, Singapore 1987

[BrFZ89] B.Braschi, A.G.Ferreira, J. Žerovnik: On the Behaviour of Simulated Annealing, *Research Report no.89-10*, 1989, LIP-IMAG, Lyon, France

[BFZ89a] B.Braschi, A.G. Ferreira, J.Žerovnik: On the Asymptotic Behaviour of Parallel Simulated Annealing, v Parallel Computing 89 (eds: D.J.Evans, G.R. Joubert, F.J.Peters) North-Holland, Amsterdam 1990, 263-268

[BuRe84] R.E.Burkard, F.Rendl, A thermodinamically motivated simulation procedure for combinatorial optimization problems, *European Journal of Operational Research* **17** 1984 pp. 169-174

[Černe84] V. Černy, A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm, *Journal of Optimization Theory and Applications* **45** 1984 pp. 41-51

[DoSS87] J.G.Donnett, M.Starkey, D.B.Skilicorn, Effective Algorithms for Partitioning Distributed Programs, *(preprint) Queen's University*, Kingston, Canada 1987

[DoSk88] J.G.Donnett, D.B.Skilicorn, Code Partitioning by Simulated Annealing, *v: Parallel Processing and Applications, E.Chiricozzi and A.D'Amico (eds.)*, North-Holland, 1988

[DuSc88] G.Dueck and T.Scheuer, Threshold Accepting, A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing, *(preprint) Heildelberg Scientific Center TR 88.10.011* 1989

[FeKO85] * E. Felten, S.Karlin and S.W.Otto: The traveling salesman problem on a hypercubic, MIMD computer *Proceedings of the 1985 International Conference on Parallel Processing* Avgust 20-23, 1985

[Frie79] R. Friedvals, Fast Probabilistic Algorithms, Lecture Notes in Computer Science 74, Springer-Verlag, Berlin, 1979, pp.57-69

[GaJo79] M.R. Garey, D.S. Johnson , *Computers and Intractability*, W.H. Freeman and Co., San Francisco (1979)

[GeGe84] * S.Geman, D.Geman, Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images, IEEE Trans PAMI 6 (1984)

[GeMi87] S.B. Gelfand, S.K. Mitter, Simulated Annealing v *Advanced School on Stochastics in Combinatorial Optimization, (eds:) G. Andreatta, F. Mason, P. Serafini*, World Scientific, Singapore 1987

[Gida85] * B.Gidas, Nonstationary Markov Chains and Convergence of the Annealing Algorithm, *J.Stat. Phys.* **39** (1985) pp. 73-131

[GoSk86] B.Golden and C.Skiscim: Using Simulated Annealing to Solve Routing and Location Problems, *Naval Res. Log. Quarterly* **33** 1986 pp. 261-279

[Gust88] J.L. Gustafson: Reevaluating Amdahl's Law, *Communications of the ACM* **31** 1988 pp. 532-533

[Grov87] L.K.Grover, GRIM: A Fast Simulated Annealing Program for Standard Cell Placement *IEEE 1987 Custom Integrated Circuits Conference* pp. 622-624 1987

[HaSa89] H. Haario, E. Saksman, On the Definition and Convergence of the Simulated Annealing Process in General State Space, Reports of the Dept. of Math., University of Helsinki, 1989

[HoUl86] V.E.Hopcroft, J.D.Ullman: Uvod v teorijo avtomatov,
jezikov in izračunov, (prevod B.Vilfan), Fakulteta za elektrotehniko, Ljubljana 1986

[JAMS85] D.S.Johnson, C.R.Aragon, L.A.McGeoch and C.Schevon: Optimization by simulated Annealing: An Experimental Evaluation preprint, 1985

[John90] M.E. Johnson, *Simulated Annealing & Applications*, American Science Press, New York

[KiGV83] S.Kirckpatrick, C.D.Gellat, Jr., M.P.Vecchi, Optimization by Simulated Annealing *Science* 1983 **220** pp. 671-680

[KhSV81] * A.Khačaturjan, S.Semenovskaja and B.Vainštein: The Thermodynamical Approach to the Structure Analysis of Chrystals *Acta Cryst* **A37** 1981 pp. 742-754

[Kern86] W.Kern: A Probabilistic Analysis of the Switching Algorithm for the Euclidean TSP, Universität zu Köln, Report No 86.28.

[KLPP86] S. Klavžar, M.Lokar, M. Petkovšek, T. Pisanski, *Diskretna optimizacija* DMFA SRS, Ljubljana, 1986

[Knut81] D.E.Knuth: Algorithms in modern mathematics and computer science, Lecture Notes in Comp. Sci. 122, Springer-Verlag, Berlin 1981

[Koza86] J.Kozak, *Podatkovne strukture in algoritmi*, DMFA SRS, Ljubljana, 1986

[LALW89] Laarhoven, P.J.M., Aarts, E.H.L., Lint, J.H. and Wille, L.T.: New Upper Bounds for the Footbal Pool Problem for 6,7 and 8 Matches *Journal of Combinatorial Theory, Series A*, **52** pp. 304-312 (1989)

[LaAr87] P.J.M. Laarhoven, and E.H.L. Aarts, *Simulated Annealing, Theory and Applications*, D.Reidel Publishing Company, Dordrecht (1987)

[Laar88] P.J.M. Laarhoven: *Theoretical and computational aspects of simulated annealing*, Centrum voor Wiskunde en Informatica, Amsterdam 1989

[LaDe88] J. Lam, J-M. Delosme, An Efficient Simulated Annealing Schedule: Derivation, Report 8816 in An Efficient Simulated Annealing Schedule: Implementation and Evaluation, Report 8817, Yale University, 1988

[LLRS85] E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, D.B. Schmoys (eds.), *The Traveling Salesman Problem*, John Wiley & sons (1985)

[LeBi89] C. Lee and L. Bic Comparing Quenching and Slow Simulated Annealing on the Mapping Problem *Proceedings of the third annual parallel processing symposium, California State University, Fullerton* 1989

[LITT89] D.C.Llewellyn, C.Tovey and M.Trick: Local optimization on graphs, *Discrete Applied Math.* **23** 1989 pp. 157-178

[LuMe86] M.Lundy, A.Mees, Convergence of an annealing algorithm, *Mathematical programming* **34** 1986 pp. 111-124

[LuBo86] J.L.Lutton, E.Bonomi, Simulated annealing algorithm for the minimum weighted perfect euclidean matching problem, *R.A.I.R.O. Oper. Res.* **20** 1986 pp. 177-197

[Maff86] F.Maffioli, Randomised algorithms in combinatorial optimisation: a survey, *Discrete Applied Mathematics* **14** 1986 pp. 157-170

[MRRT53] N.Metropolis, A.Rosenbluth, M.Rosenbluth, A.Teller, E.Teller, Equation of State Calculations by Fast Computing Machines, *J.Chem. Phys.* **21** 1953 pp. 1087-1091

[MiRS86] D.Mitra, F.Romeo, A. Sangiovanni-Vincentelli, Convergence and Finite Time Behavior of Simulated Annealing, *Adv. Appl. Prob.* **18** 1986 pp. 747-771

[Pinc70] * M.Pincus: A Monte Carlo Method for the Approximate Solutions of Certain Types of Constrained Optimization Problems *Oper. Res* **18** 1970 pp. 1225-1228

[Rabi76] M.O.Rabin: Probabilistic Algorithms, v Algorithms and Complexity (ur. Traub), 21-39, Academic Press 1976

[RaSh89] M.A.Rahin, J.Shield, Parallel Partitioning of Concurrent VLSI Simulation Graphs, *(preprint) Dept of Electronic and Electrical Engineering, Loughborough University of Technology*, 1989

[RiTi85] * A.H.G. Rinooy Kan, G.T. Timmer, Stochastic Global Optimization Methods, parts I & II, Econometric Institute, Erasmus University, Rotterdam, 1985

[SaHa88] G.H.Sasaki, B.Hajek, The time Complexity of Maximum Matching by Simulated Annealing, *Jour. ACM* **35** 1988 pp. 387-403

[Sasa91] G.Sasaki The effect of the density of states on the Metropolis algorithm *Information Processing Letters* **37** (1991) pp. 159-163

[Žero88] J. Žerovnik, Comparison of Asymptotic Behaviour of two Randomised Heuristic Approaches, *Preprint Series Dept. Math, University E.K. Ljubljana* **26**, pp. 186-192 (1988) (poslano v objavo)

[Žero90] J. Žerovnik, Verjetnost v kombinatorični optimizaciji, Univerza v Ljubljani, FNT, (oddana) doktorska disertacija, 1990

[Wels83] D.J.A.Welsh: Randomised algorithms, Discrete Applied Math. 5 (1983), 133-145

[Will87] L.T. Wille: The Footbal Pool Problem for 6 Matches: A new Upper Bound Obtained by Simulated Annealing, *Journal of Combinatorial Theory, Series A*, **45** pp. 171-177 (1987)

Z zvezdico * smo označili posredne reference.

Keywords: machine learning, artificial neural networks, artificial intelligence, naive Bayes, Hopfield's model

Igor Kononenko
Univerza v Ljubljani
Fakulteta za elektrotehniko in računalništvo
Tržaška 25, Ljubljana

POVZETEK

V prispevku je opisana posplošitev Bayesovih nevronskih mrež na zvezna stanja nevronov, analogno Hopfieldovi posplošitvi. Podane so posplošene prehodne funkcije in dokazana stabilnost izvajanja zvezne Bayesove nevronske mreže, ki temelji na verjetnosti in zvezne Bayesove nevronske mreže, ki temelji na razmerju verjetnosti. Opisane so prednosti zveznih Bayesovih nevronskih mrež pred diskretnimi.

CONTINUOUS BAYESIAN NEURAL NETWORKS

In the paper the Bayesian neural network model is generalized to continuous states of neurons, analogously to Hopfield's generalization. The generalized transition functions are given together with the proof of convergence of execution for the continuous Bayesian neural network based on probability and for the continuous Bayesian neural network based on probability ratio. The advantages of continuous Bayesian neural network models are discussed.

1 Uvod

V (Kononenko 1990) so opisane *diskretne* Bayesove nevronske mreže, ki temeljijo na verjetnosti (BNM-p), in *diskretne* Bayesove nevronske mreže, ki temeljijo na razmerju verjetnosti (BNM-odds). Celotna kombinacijska funkcija za izračun novega stanja S_j' in s tem izhoda nevrona v BNM-p je definirana z:

$$S_j' = D_j(P(S_j = 1|S_1, \ldots, S_N)) \quad (1)$$

kjer je

$$P(S_j|S_1, \ldots, S_N) = \\ = P(S_j = 1) \prod_{S_i=1,(i\neq j)} \frac{P(S_j = 1|S_i = 1)}{P(S_j = 1)} \quad (2)$$

verjetnost, da je j-ti nevron aktiven, pri danih indeksih aktivnih nevronov in

$$D_j(X) = \begin{cases} 1, & \text{če } X > P(S_j = 1) \\ S_j, & \text{če } X = P(S_j = 1) \\ 0, & \text{če } X < P(S_j = 1) \end{cases} \quad (3)$$

pragovna odločitvena funkcija. Pri tem so $S_i, i = 1..N$ trenutna stanja vseh nevronov v mreži in $P(S_j = 1)$ apriorna verjetnost aktivnega stanja j-tega nevrona. Če je izračunana verjetnost večja od apriorne verjetnosti, potem je novo stanje enako 1, če je enaka apriorni verjetnosti, potem se stanje nevrona ne spremeni, in če je izračunana verjetnost manjša od apriorne verjetnosti, potem

je novo stanje nevrona enako 0.

Celotna kombinacijska funkcija za izračun novega stanja S_j' in s tem izhoda nevrona v BNM-odds je definirana z:

$$S_j' = Dq_j(odds(S_j = 1|S_1, \ldots, S_N)) \quad (4)$$

kjer je

$$odds(S_j|S_1, \ldots, S_N) =$$

$$= odds(S_j = 1) \prod_{i \neq j} \frac{odds(S_j = 1|S_i = 1)}{odds(S_j = 1)} \quad (5)$$

razmerje verjetnosti, da je j-ti nevron aktiven, pri danih indeksih aktivnih nevronov in

$$Dq_j(X) = \begin{cases} 1, & \check{c}e \quad X > odds(S_j = 1) \\ S_j, & \check{c}e \quad X = odds(S_j = 1) \\ 0, & \check{c}e \quad X < odds(S_j = 1) \end{cases} \quad (6)$$

pragovna odločitvena funkcija. Pri tem so $S_i, i = 1..N$ trenutna stanja vseh nevronov v mreži in $odds(S_j)$ apriorna verjetnost stanja 1 j-tega nevrona deljena z apriorno verjetnostjo nasprotnega stanja. Če je izračunano razmerje verjetnosti večje od apriornega razmerja, potem je novo stanje enako 1, če je enako apriornemu razmerju, potem se stanje nevrona ne spremeni, in če je izračunano razmerje verjetnosti manjše od apriornega, potem je novo stanje nevrona enako 0.

Ker stanja BNM-p in BNM-odds lahko zavzemajo samo diskretni vrednosti 0 in 1, jim pravimo *diskretni*. V (Kononenko 1990) je dokazana stabilnost asinhronega izvajanja za obe vrsti mrež iz poljubnega začetnega stanja, z uporabo funkcij podobnosti odvisnih samo od stanja mreže. Le te z asinhronim izvajanjem monotono padajo. Ker je možnih stanj diskretne nevronske mreže končno mnogo, nam to garantira prihod v fiksno točko v končnem številu iteracij. Oglejmo si še enkrat funkciji podobnosti za BNM-p in BNM-odds. Naslednja funkcija definira podobnost med aktivacijskimi nivoji nevronov in njihovimi trenutnimi stanji v BNM-p:

$$Sim(S_1, \ldots, S_N) =$$

$$= \prod_{i, S_i = 1} \frac{P(S_i = 1|S_1, \ldots, S_N)}{P(S_i)} \quad (7)$$

Podobnost je večja, če je aktivacijski nivo (izračunana verjetnost, da je nevron aktiven) večji od apriorne verjetnosti, da je nevron aktiven in obratno. Za BNM-odds je funkcija podobnosti enaka, le da produkt teče preko vseh nevronov (ni pogoja $S_i = 1$).

V tem prispevku je podana generalizacija večsmernih BNM na zvezna stanja nevronov analogno Hopfieldovemu (1984) zveznemu modelu nevronske mreže. Namesto diskretnih stanj 0 in 1 je stanje nevrona v zvezni BNM predstavljeno z realno vrednostjo na intervalu [0,1]. Stanje nevrona bo proporcionalno verjetnosti, da je nevron trenutno aktiven. V naslednjem razdelku je na kratko opisan Hopfieldov zvezni model nevronske mreže. V nadaljnjih dveh razdelkih sta opisana zvezna modela večsmerne BNM-p in BNM-odds. V zadnjem razdelku prikazane prednosti in slabosti zveznih modelov.

2 Hopfieldov zvezni model nevronske mreže

Hopfield (1984) za zvezni model uporablja isto topologijo kot pri diskretni nevronski mreži, t.j. vsak nevron je povezan z vsakim. Kombinacijska funkcija diskretnega modela je podana z enačbo (Hopfield 1982):

$$A_j = \sum_{S_i = 1, i \neq j} T_{ji} + I_j = \sum_{i \neq j} S_i T_{ji} + I_j \quad (8)$$

V Hopfieldovem modelu so T_{ji} elementi spominske matrike, dobljene kot vsota zunanjih produktov učnih vzorcev, ki so popravljeni tako, da so vrednosti komponent 0 spremenjene v -1. I_j predstavlja konstanten vhod v j-ti nevron. Model BNM brez povratnih povezav ($i \neq j$) ustreza spominski matriki z ničelno diagonalo ($T_{ii} = 0$ za vse $i = 1 \ldots N$). S_i ima lahko vrednosti 0 in 1 kot v originalnem Hopfieldovem modelu, čeprav učno pravilo Hopfieldovega modela uporablja vrednosti -1 in 1.

Kombinacijska funkcija zveznega Hopfieldovega modela je analogna. Dinamika zveznega modela je opisana z naslednjima enačbama (Hopfield 1984):

$$C_j \frac{du_j}{dt} = \sum_{i,i\neq j} T_{ji} S_i - \frac{u_j}{R_j} + I_j = A_j - \frac{u_j}{R_j} \quad (9)$$

$$S_j = g_j(u_j) \quad (10)$$

kjer je S_i stanje oziroma izhod i-tega nevrona, u_j vhod (aktivacijski nivo) j-tega nevrona in I_j, R_j, C_j konstante. g_j je izhodna funkcija, ki definira relacijo med vhodom in izhodom, in je odvedljiva in sigmoidna z asimptotama 0 in 1. Iz enačbe (9) sledi, da je hitrost spremembe u_j proporcionalna razliki med trenutnim in novim u_j, izračunanim po enačbi (8). V originalni Hopfieldovi (1984) enačbi je izpuščen pogoj $i \neq j$, ker za Hopfieldovo pravilo učenja velja, da so vsi $T_{ii} = 0$.

Stabilnost zveznega modela pokaže Hopfield tako, da definira "energijsko" funkcijo stanja sistema:

$$E = -\frac{1}{2} \sum_j \sum_{i,i\neq j} S_j S_i T_{ji} - \sum_j I_j S_j + \sum_j \frac{1}{R_j} \int_0^{S_j} g_i^{-1}(S) dS$$

$$= -E1 + \sum_j \frac{1}{R_j} \int_0^{S_j} g_i^{-1}(V) dV \quad (11)$$

ki s časom monotono pada, ker velja:

$$\frac{dE}{dt} \leq 0 \quad (12)$$

in

$$\frac{dE}{dt} = 0 \rightarrow \forall j : \frac{dS_j}{dt} = 0$$

3 Zvezna Bayesova nevronska mreža, ki temelji na verjetnosti

Informacijski prispevek i-tega nevrona za aktivnost j-tega nevrona je v diskretni BNM-p dobimo z logaritmiranjem enačbe (2) in je podan z $log_2 \frac{P(S_j|S_i)}{P(S_j)}$. Bolj splošna definicija informacijskega prispevka je podana z

$$S_i \times log_2 \frac{P(S_j|S_i)}{P(S_j)}$$

kjer S_i predstavlja trenutno stanje i-tega nevrona. S_i lahko zavzema poljubno vrednost na intervalu [0,1] in ustreza verjetnosti, da je i-ti nevron aktiven. Posplošen logaritem enačbe (2) vsebuje vsoto po vseh nevronih in ne samo po aktivnih nevronih:

$$-log_2 P(S_j|S_1, \ldots, S_N) =$$

$$-log_2 P(S_j) - \sum_{i\neq j} \left(S_i \times log_2 \frac{P(S_j|S_i)}{P(S_j)} \right) \quad (13)$$

Z antilogaritmiranjem dobimo kombinacijsko funkcijo zvezne BNM-p, ki je posplošena enačba (2):

$$P(S_j|S_1, \ldots, S_N) = P(S_j) \prod_{i\neq j} \left(\frac{P(S_j|S_i)}{P(S_j)} \right)^{S_i} \quad (14)$$

Posplošitev pogojne verjetnosti $P(S_j|S_i = 1)$, pri negotovi evidenci, da je $S_i = 1$, je podana z $S_i \times P(S_j|S_i = 1)$ (Ihara 1987). Ta posplošitev vodi do drugačne definicije enačbe (14), kjer lahko več nevronov predstavlja isti atribut. Enačba (14) je smiselna posplošitev, če se vplivi različnih nevronov, čeprav mogoče predstavljajo isti atribut, obravnavajo neodvisno.

Če v enačbi (13) namesto $-log_2 P(S_j|S_1, \ldots, S_N)$ pišemo A_j, namesto $-log_2 P(S_j)$ pišemo I_j in namesto $-log_2 \frac{P(S_j|S_i)}{P(S_j)}$ pišemo T_{ji}, dobimo enačbo (8), t.j. kombinacijsko funkcijo Hopfieldovega modela. Zato lahko za opis dinamike zvezne BNM-p uporabimo enačbi (9) in (10). Dokaz stabilnosti je ekvivalenten Hopfieldovemu dokazu (glej razdelek 2). Posplošena funkcija podobnosti (7) na zvezna stanja je torej:

$$Sim(S_1,\ldots,S_N) = \prod_j \left(\frac{P(S_j|S_1,\ldots,S_N)}{P(S_j)}\right)^{S_j} \quad (15)$$

4 Zvezna Bayesova nevronska mreža, ki temelji na razmerju verjetnosti

Težo evidence stanja S_i i-tega nevrona za aktivnost j-tega nevrona v diskretni BNM-odds dobimo z logaritmiranjem enačbe (5) in je podana z $log_2 \frac{odds(S_j=1|S_i)}{odds(S_j=1)}$. Bolj splošna definicija teže evidence je podana z

$$S_i \times log_2 \frac{odds(S_j=1|S_i=1)}{odds(S_j=1)} +$$

$$(1-S_i) \times log_2 \frac{odds(S_j=1|S_i=0)}{odds(S_j=1)}$$

kjer S_i predstavlja trenutno stanje i-tega nevrona. S_i lahko zavzema poljubno vrednost na intervalu $[0,1]$ in ustreza verjetnosti, da je i-ti nevron v stanju 1 (oziroma 1 minus verjetnosti, da je nevron v stanju 0). Posplošena kombinacijska funkcija (5) za zvezne BNM-odds je sledeča:

$$odds(S_j|S_1,\ldots,S_N) =$$

$$= odds(S_j=1) \prod_{i\neq j} \left[\left(\frac{odds(S_j=1|S_i=1)}{odds(S_j=1)}\right)^{S_i} \times \left(\frac{odds(S_j=1|S_i=0)}{odds(S_j=1)}\right)^{1-S_i}\right] \quad (16)$$

Ustrezna posplošena funkcija podobnosti je pri zvezni BNM-odds različna od funkcije podobnosti zvezne BNM-P:

$$Sim(S_1,\ldots,S_n) =$$

$$= \prod_j \left[\left(\frac{P(S_j=1|S_1,\ldots,S_n)}{P(S_j=1)}\right)^{S_j} \times \left(\frac{P(S_j=0)}{P(S_j=0|S_1,\ldots,S_n)}\right)^{1-S_j}\right] \quad (17)$$

Če enačbo (16) logaritmiramo in namesto $-log_2 odds(S_j|S_1,\ldots,S_n)$ pišemo A_j, namesto $-log_2 odds(S_j)$ pišemo I_j in namesto $-log_2 \frac{P(S_j=X|S_i=Y)}{P(S_j=X)}$ pišemo T_{ji}^{XY}, dobimo:

$$A_j = I_j + \sum_{i\neq j}(S_i(T_{ji}^{11}-T_{ji}^{01}) + (1-S_i)(T_{ji}^{10}-T_{ji}^{00})) \quad (18)$$

Če zgoraj definirani aktivacijski nivo nevrona A_j uporabimo v enačbi (9) dobimo enačbo za opis dinamike zvezne BNM-odds. Ker nova enačba ni enaka enačbi za zvezno BNM-P, dokaz stabilnosti ni tako očiten.

Iz dokaza stabilnosti Hopfieldovega modela podanega v razdelku 2 je razvidno, da je za stabilnost pri enačbi (11) zadosten pogoj:

$$\frac{dE_1}{dt} = \sum_j \left(\frac{dS_j}{dt} A_j\right) \quad (19)$$

Če v enačbi (11) uporabimo za E_1 naslednji izraz:

$$E_1(S_1,\ldots,S_N) = \sum_j I_j S_j +$$

$$+\frac{1}{2}\sum_j\sum_{i\neq j}(S_jS_iT_{ji}^{11} + S_j(1-S_i)T_{ji}^{10} +$$

$$(1-S_j)S_iT_{ji}^{01} + (1-S_j)(1-S_i)T_{ji}^{00}) \quad (20)$$

potem je pogoj (19) izpolnjen, ker velja:

$$\begin{aligned}\frac{dE_1}{dt} &= \sum_j \left(\frac{dE_1}{dS_j}\frac{dS_j}{dt}\right) \\ &= \frac{1}{2}\sum_j \left(\frac{dS_j}{dt}A_j\right) + \frac{1}{2}\sum_i \left(\frac{dS_i}{dt}A_i\right) \\ &= \sum_j \left(\frac{dS_j}{dt}A_j\right) \end{aligned} \quad (21)$$

S tem je stabilnost zveznega modela BNM-odds dokazana.

5 Zaključki

Bayesove nevronske mreže so lahko diskretne ali zvezne ter lahko temeljijo na verjetnosti ali na razmerju verjetnosti. Razlika med Hopfieldovim modelom in BNM je v učnem pravilu. Za učenje BNM zadošča osnovno Hebbovo pravilo (Hebb 1949), za katerega je precej nevrofiziološke evidence, da se po tem pravilu učijo biološki nevroni. Uteži na sinapsah ustrezajo apriornim verjetnostim, da sta povezana nevrona aktivna. Hopfieldov model uporablja posplošeno Hebbovo pravilo. Uteži na sinapsah ustrezajo razliki med verjetnostjo, da sta povezana nevrona v istem stanju, in verjetnostjo, da sta povezana nevrona v različnem stanju. Zato uteži Hopfieldovega modela vsebujejo manj informacije kot uteži pri BNM. Kompleksnost učenja in izvajanja je pri obeh modelih enaka, po klasifikacijski pravilnosti pa BNM daleč presega Hopfieldov model (Kononenko 1990).

Za opis stanja modela Hopfield uporablja energijsko funkcijo, ki je analogna funkciji podobnosti za opis stanja BNM. Logaritem funkcije podobnosti lahko interpretiramo kot entropijo ali informacijsko vsebino sistema. Iz tega sledi, da je fiksna točka stanje sistema z lokalno minimalno entropijo.

Prednost zveznih BNM je v večji fleksibilnosti predstavitve znanja. Vhodni podatki so lahko "mehki" in nezanesljivi. Dani primer ima lahko več vrednosti za isti atribut, ki jim je možno določiti zaupanje. Tudi odgovor zvezne BNM je zato lahko bolj fleksibilen.

BNM-P ima prednost pred BNM-odds, ker se lahko entropija direktno posploši na več kot dva disjunktna dogodka. BNM-P je zato bolj prožna in računsko manj zahtevna. BNM-odds je primerna samo za binarne atribute in razrede in je manj primerna za predstavitev manjkajočih podatkov. V diskretni BNM-odds je lahko manjkajoči podatek predstavljen z apriornim razmerjem verjetnosti (*odds*) samo med učenjem, medtem ko je v zvezni BNM-odds lahko predtavljen z apriornim razmerjem verjetnosti, tako med učenjem kot med izvajanjem. Interpretacija izvajanja BNM-P je bolj naravna (prispevek informacije v bitih) kot interpretacija BNM- odds (teža evidence). Po drugi strani pa BNM-odds potrebuje bistveno manj nevronov za predstavitev istega problemskega prostora kot BNM-P.

Z nadaljnjimi raziskavami bo potrebno eksperimentalno testiranje zveznih BNM. Potrebno bo preveriti prednosti zveznih BNM na realnih problemih. Poleg bolj fleksibilne predstavitve znanja omogočajo zvezne BNM, z *asinhronim* izvajanjem, ki služi kot *globalnainformacija*, bolj zanesljivo delovanje. Namreč zaradi asinhronosti bodo nevroni, katerih spremembe stanj so bolj zanesljive, prej spremenili svoje stanje, kar bo vplivalo na spremembe stanj ostalih nevronov, preden le-ti uspejo spremeniti svoje stanje. Spremembe niso diskretne (torej niso drastične) ampak majhne z majhnimi vplivi in ne prehudimi posledicami.

Reference

Hebb, D.O. (1949) *The Organization of Behavior*. New York: Wiley.

Hopfield J.J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences* 79:2554-2558.

Hopfield J.J. (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. of the National Academy of Sciences* 81:4586-4590.

Ihara J. (1987) Extension of conditional probability and measures of belief and disbelief in a hypothesis based on uncertain evidence. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, no.4, pp. 561-568.

Kononenko, I. (1990) Bayesove umetne nevronske mreže, *Informatica*, letnik 14, št. 3, str.72-86.

ALGORITEM ZA PRETVORBO TELESA PREDSTAVLJENEGA Z OVOJNICO V PREDSTAVITEV Z OSMIŠKIM DREVESOM

INFORMATICA 2/91

Keywords: geometric modeling, boundary representation, octree representation, computer graphics

Natalija Trstenjak,
Borut Žalik, Nikola Guid
Tehniška fakulteta Maribor
Smetanova 17, 62000 Maribor
Slovenija, Jugoslavija

Predstavitev teles z ovojnico je široko uporabljena v računalniški grafiki in predstavlja tudi primarno predstavitev v Eksperimentalnem modelirniku teles, ki smo ga razvili. Vendar pa je izvajanje Boolovih operatorjev, ki so standardno orodje vsakega modelirnika, zelo težavno nad telesi predstavljenimi z ovojnico. Zato smo kot sekundarno predstavitev v modelirniku uvedli predstavitev z osmiškimi drevesi, ki je za izvajanje Boolovih operatorjev prav idealna. V tem članku obravnavamo pretvorbo predstavitve z ovojnico poljubnega telesa v predstavitev z osmiškim drevesom. Podrobno smo predstavili iskanje sečišč med oktantom in poljubnim telesom, ki je najbolj kritičen del pretvorbe.

An Algorithm for Boundary to Octree Conversion

The boundary representation is widely used in computer graphics and it represents a primary representation in our Experimental geometric modeler. An implementation of the regularized Boolean operators which are the standard tools in the solid modelers is very complicated in the case of boundary representation. For this reason the octree representation is included as a secundary representation and it is almost ideal for the implementation of Boolean operators. In this paper, we consider the problem of converting the boundary representation of a polyhedron to its octree representation. The critical work in the subdivison process is to determine how an octant intersects the polyhedron and this problem is described in details.

1 Uvod

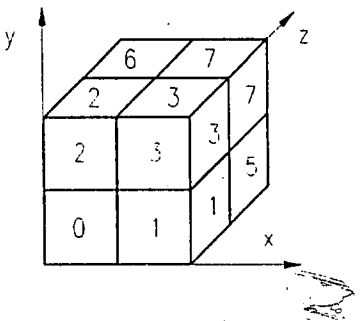
Osnovni problem geometrijskega modeliranja je v opisovanju tridimenzionalnega zveznega telesa v digitalnem računalniku tako, da je predstavitev nedvoumna [HILL82]. Znanih je več nedvoumnih predstavitev, najpogostejše pa so predstavitev z ovojnico ("boundary representation", B-rep), predstavitev s temeljimi gradniki ("constructive solid geometry", CSG) in z osmiškimi drevesi ("octree representation") ([REQU82], [REQU83], [MORT85]). Vsaka izmed predstavitev ima svoje slabosti in prednosti. Zato večina komercialnih modelirnikov temelji na takoimenovani hibridni predstavitvi, ki jo dobimo s kombinacijo vsaj dveh različnih predstavitev. Idealno arhitekturo bi tvorile vse tri prej omenjene predstavitve [MILL89]. Za hibridno predstavitev so izjemno pomembni učinkoviti pretvorbeni algoritmi iz

ene predstavitve v drugo. V prispevku bomo podali pretvorbeni algoritem iz predstavitve z ovojnico v predstavitev z osmiškim drevesom.

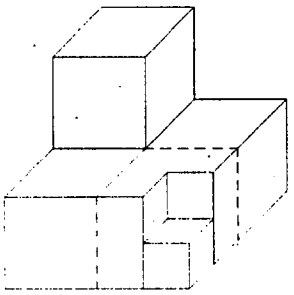
Predstavitev z ovojnico hrani podatke o površju telesa in je sestavljeno iz oglišč, robov in ploskev. Podatkovna struktura mora hraniti tako sosednostne relacije kot geometrijske podatke, kar zahteva pazljivo načrtovanje podatkovne strukture. Ker je vsaka ploskev predstavljena eksplicitno, omogoča dodajanje specifičnih informacij vsaki ploskvi telesa (npr. barva, kvaliteta materiala), prav tako pa v principu ni težav pri vgradnji poljubno oblikovanih ploskev [CHIY87]. Pri vizualizaciji predstavljenega telesa večjih težav ni, saj lahko žični model prikažemo zelo hitro, mnogo klasičnih algoritmov računalniške grafike za odstranjevanje zakritih robov, senčenje in uporabljanje pa temelji na tej predstavitvi [PRAT83]. Telesa lahko modeliramo z nizko - nivojskimi operatorji, npr. Eulerjevimi operatorji, z uporabo višje - nivojskih lokalnih operatorjev in z Boolovimi operatorji ([MANT82], [WILS85], [CHIY85], [MANT88]). Prav implementacija Boolovih operatorjev, ki so standardno orodje vsakega geometrijskega modelirnika, pa je v predstavitvi z ovojnico zelo težavno [MANT83].

Predstavitev z osmiškim drevesom temelji na principu rekurzivne delitve prostora. Rezultat rekurzivnega delitvenega procesa je predstavljen z drevesom osme stopnje, katerega vozlišča so ali listi ali sinovi. Korensko vozlišče v izhodnem drevesu predstavja vhodni prostor. Končna vozlišča-listi predstavljajo oktante, ki vsebujejo enotske kocke iste barve in so označena s črna ali bela, odvisno od tega ali se nahajajo znotraj ali zunaj telesa. Vmesna vozlišča imenujemo notranja vozlišča in so označena s siva. Ta predstavljajo oktante v prostoru osmiškega drevesa, ki ležijo delno znotraj delno zunaj telesa in jih v rekurzivnem postopku delimo na podoktante. Vsak oktant ima predpisano osmiško kodo, ki določa njegov položaj glede na oktanta očeta. Osmiške kode oktantov so prikazane na sliki 1a, primer telesa in njegove predstavitve pa kažeta sliki 1b in 1c.

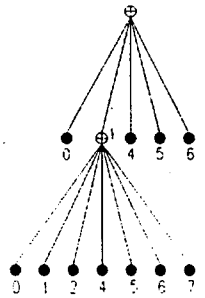
Predstavitev z osmiškimi drevesi je prav idealna za izvajanje Boolovih operacij. Vsa izračunavanja pri Boolovih operacijah temeljijo na celoštevilski aritmetiki (vse potrebne zaokrožitve smo namreč odpravili že v fazi tvorbe osmiškega drevesa), zato so algoritmi izredno robustni. Glavna slabost je v tem, ker meja telesa ni eksplicitno znana, zato ni možno hitro prikazati npr. žičnega modela objekta; direktno jih lahko vizualiziramo samo s tehnikami sledenja žarku ("ray tracing"). Prav tako predstavitev zahteva mnogo pomnilnika, s tem pa je omejena tudi resolucija objekta [CLIF87]. Za predstavitev z osmiškimi drevesi razvijajo zaradi njenih dobrih lastnosti tudi specialno namensko aparaturno opremo [KAUF88].



a) Osmiške kode oktantov (oktant 4 je skrit)



b) Primer telesa



c) Njegova predstavitev z osmiškim drevesom

Slika 1 Osmiške kode oktantov ter predstavitev telesa z osmiškim drevesom

Naš Eksperimentalni geometrijski modelirnik (EGM) ima primarno predstavitev z ovojnico, sekundarno pa z osmiškimi drevesi. Predstavitev z ovojnico podpira 14 vgrajenih Eulerjevih operatorjev, domena teles pa vključuje telesa z ravnimi ploskvami [ŽALI89a]. Višje nivojski operatorji omogočajo tvorbo teles s translacijskim pomikanjem in z operatorji, ki upoštevajo značilnosti teles [ŽALI90]. Vgrajeni so tudi Boolovi operatorji, ki pa so uspešni le nad konveksni telesi z lici, ki ne vsebujejo prstanov [ŽALI89b]. Zaradi tega smo se odločili, da bomo izdelali algoritem za pretvorbo predstavitve z ovojnico v predstavitev z osmiškimi drevesi.

2 Tehnike za tvorbo osmiškega drevesa iz predstavitve z ovojnico

Poznamo različne tehnike za tvorbo osmiškega drevesa, ki se razlikujejo med seboj po obliki vhodnih podatkov. Vhodni podatki so različne predstavitve 3D teles, kot so na primer 3D polja, zaporedne sekcije, slike obrisa objekta, globinske slike in modeli objektov. Med modele objektov spada tudi predstavitev z

ovojnico. Poznamo dve tehniki za tvorbo osmiškega drevesa telesa, predstavljenega z ovojnico: tehnika označevanja komponent in tehnika deli-in-vladaj.

Tehnika označevanja komponent temelji na principu polnjenja notranjosti zaprte meje, zato ne potrebuje testiranja presečišč. Algoritem je sestavljen iz dveh delov. V prvem delu pretvorimo mejo objekta v delno 3D binarno drevo. Črni listi drevesa predstavljajo celice, ki sekajo mejo objekta. V drugem delu delno binarno drevo prečno obiščemo in tiste nedoločene bele liste, ki odgovarjajo notranjosti ali zunanjosti telesa, klasificiramo v črna ali bela vozlišča, skladno s tehniko označevanja. Algoritem generira binarno verzijo osmiškega drevesa, ki jo lahko enostavno pretvorimo v osmiško drevo.

Druga tehnika za pretvorbo poljubnega telesa, predstavljenega z ovojnico v predstavitev z osmiškim drevesom, je tehnika deli_in_vladaj, ki jo bomo v tem članku podrobno predstavili. Ta tehnika velja za grob pristop in pričenja z enojnim črnim vozliščem - korenom drevesa kot začetno aproksimacijo poljubnega telesa in nato nadaljuje z rekurzivno delitvijo prostora osmiškega drevesa. Algoritem je predstavljen na sliki 2.

```
Deli_in_vladaj ( vozlišče )
(*  M : meja - obris telesa  *)

Begin
 Case PRESEČIŠČE ( vozlišče,M ) of
  BELA : opusti vozlišče in vkoreninjeno
         poddrevo
  ČRNA : vozlišče.barva = ČRNA
  SIVA : begin
          vozlišče.barva = SIVA
          for i = 0 to 7
           Deli_in_vladaj ( vozlišče.sin[i] )
         end
 end
End
```

Slika 2 Algoritem tehnike deli-in-vladaj

Kritičen del v procesu delitve je določiti, kako oktant seka poljubno telo. To lahko opravimo v dveh korakih: v prvem koraku pregledamo ali oktant seka katero koli ploskev poljubnega telesa. Če seka, potem je oktant delno zunaj, delno znotraj telesa, pripadajoče vozlišče v osmiškem drevesu označimo s siva in oktant delimo na podoktante. Drugače je oktant ali zunaj ali znotraj telesa. V drugem koraku, katerega bistvo je vsebnostni test, določimo, katera od trditev je pravilna. Če za katero koli ogliščе oktanta ugotovimo, da je obdano s površino poljubnega telesa, se oktant nahaja znotraj telesa, vozlišče osmiškega drevesa označimo s črna, drugače pa se nahaja zunaj telesa in vozlišče osmiškega drevesa označimo z bela [HOME88].

3 Ugotavljanje presečišča med oktantom in poljubnim telesom

Testiranje presečišča med oktantom in poljubnim telesom je v splošnem počasen proces. Da bi povečali hitrost testiranja presečišč, uvedemo izločevalni postopek, ki sestoji iz petih korakov oziroma testov:

- testa minimax,
- testiranja ravnin ploskev telesa,
- testiranja oglišč ploskev telesa,
- testiranja presečnega mnogokotnika,
- testiranja lukenj ploskve telesa.

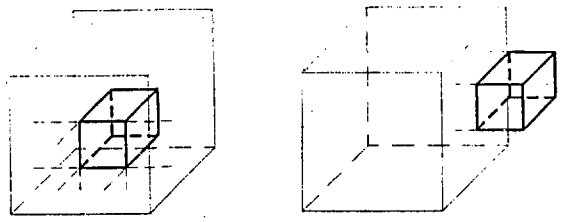
Namen tega postopka je v prvih dveh korakih zmanjšati število ploskev telesa, ki bodo prišle v naslednji korak. Izločamo ploskve, ki zagotovo ležijo zunaj oktanta in ga zato ne morejo sekati. Če je število vseh izločenih ploskev enako številu vseh ploskev telesa, v naslednji korak ne vstopimo, saj lahko v takem primeru določimo barvo vozlišča oziroma presečišča z vsebnostnim testom. V tretjem koraku želimo ugotoviti, ali oktant seka katero od ploskev telesa, ki jih nismo izločili in s tem določiti barvo vozlišča. Vsak nadaljni korak obravnava presečišče bolj podrobno in v zadnjem, petem koraku, barvo vozlišča osmiškega drevesa, ki predstavlja pripadajoč oktant v prostoru osmiškega drevesa, zagotovo določimo.

V podpoglavjih, ki sledijo, je podrobno obdelan vsak korak in zaključki, ki sledijo iz rezultatov testiranja v omenjenem koraku.

3.1 Test minimax

S testom minimax izločimo tiste ploskve telesa, ki zagotovo ne sekajo oktanta. Test je razširjen na 3D prostor, z njim pa ugotavljamo prekrivanje omejitvenih škatel ("bounding box") ploskev telesa in oktanta (omejitvena škatla ploskve je pravokotna škatla, katere robovi so vzporedni z osmi prostora osmiškega drevesa in omejuje ploskev). To je prva groba selekcija ploskev, ki ne sekajo oktanta, saj zaradi narave testa minimax ne moremo trditi, da smo izločili vse takšne ploskve iz nadaljnega postopka [GUID88]. Kljub temu pa s tem testom izločimo veliko število ploskev, še posebej pri večji globini osmiškega drevesa, ko je lahko oktant napram telesu zelo majhen.

Če s testom minimax izločimo vse ploskve telesa, kar pomeni, da vse ploskve ležijo zunaj oktanta, se lahko oktant nahaja strogo znotraj telesa ali strogo zunaj telesa. Primer je podan na sliki 3, kjer je oktant narisan z močnejšo in telo s tanjšo črto. Da bi določili, katera od trditev je pravilna, uporabimo vsebnostni test oglišč oktanta v telesu.



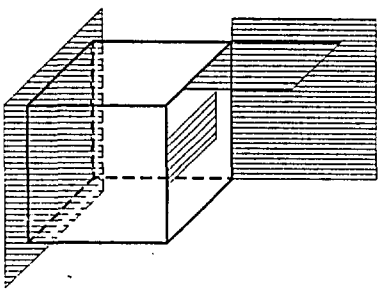
a) Oktant strogo znotraj telesa b) Oktant strogo zunaj telesa

Slika 3 S testom minimax smo izločili vse ploskve telesa

Če s testom minimax nismo izločili vseh ploskev telesa, oziroma ne moremo trditi, da ležijo vse zunaj oktanta, ne moremo določiti barve vozlišča osmiškega drevesa. Ploskve, ki jih nismo izločili, testiramo v drugem koraku, ki vključuje test ravnin ploskev.

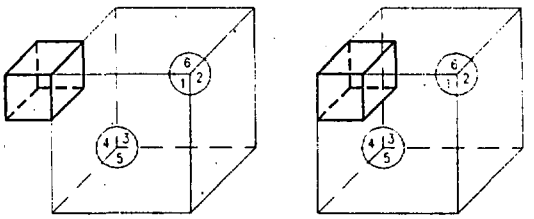
3.2 Test ravnin ploskev telesa

S tem testom izločimo ploskve telesa, ki se oktanta dotikajo. Za vsako ploskev telesa izračunamo enačbo ravnine, v kateri leži, in jo primerjamo z enačbami ravnin, v katerih ležijo ploskve oktanta. Če ravnina ploskve telesa sovpada s katero od ravnin ploskev oktanta, leži ploskev telesa na eni od ploskev oktanta ali pa se oktanta dotika v enem od njegovih robov. Primer je podan na sliki 4, kjer so ploskve telesa označene s vodoravno šrafuro. V tem primeru se ploskev in oktant zagotovo ne sekata in ploskev izločimo iz nadaljnega postopka.



Slika 4 Ploskve telesa, katerih ravnine sovpadajo z ravninami ploskev oktanta

Če je vsota ploskev, ki smo jih izločili v prvem koraku, in ploskev, katere smo izločili v drugem koraku, enaka številu vseh ploskev telesa, se lahko oktant nahaja le zunaj ali znotraj telesa (slika 5). Uporabimo vsebnostni test, da ugotovimo, katera od navedenih trditev je pravilna in lahko skladno s tehniko označevanja določimo barvo vozlišča osmiškega drevesa.



a) Oktant zunaj telesa b) Oktant znotraj telesa

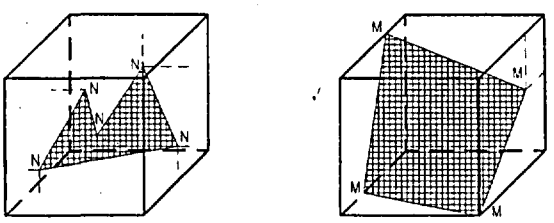
Slika 5 Ploskve telesa 2, 3, 5 smo izločili s testom minimax, ravnine ploskev 1, 4, 6 sovpadajo z ravninami oktanta

Če vsota do sedaj izločenih ploskev ni enaka številu vseh ploskev telesa, je barva presečišča odvisna od lege ploskev, ki jih še nismo izločili. Te ploskve testiramo v naslednjem koraku, ki vključuje test oglišč ploskev.

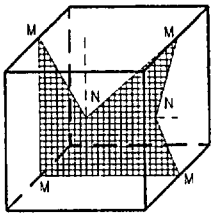
3.3 Test oglišč ploskev telesa

V tem koraku ne izločamo ploskev, ampak želimo ugotoviti, ali katera od ploskev zagotovo seka oktant. Ploskve telesa, ki jih testiramo v tem koraku, lahko:

- ležijo znotraj oktanta,
- sekajo oktant,
- ležijo zunaj oktanta.



a) OP=NP ; MP=0, ZP=0 b) OP=MP ; NP=0, ZP=0



c) OP=MP+NP ; MP≠0, NP≠0, ZP=0

Slika 6 Ploskev leži znotraj oktanta

Ploskev zagotovo seka oktant, če leži v celoti znotraj oktanta, ali pa seka oktant, ne glede na to ali je konveksna ali konkavna in ali ima luknje ali ne. Ta dva položaja ploskve lahko enostavno določimo na podlagi lege njenih oglišč.

Naj je OP število vseh oglišč ploskve telesa. Število oglišč ploskve zunaj oktanta označimo z ZP, število oglišč znotraj oktanta z NP in število oglišč ploskve na meji oktanta z MP.

Če se vsa oglišča ploskve telesa nahajajo znotraj ali na meji oktanta, oziroma če velja

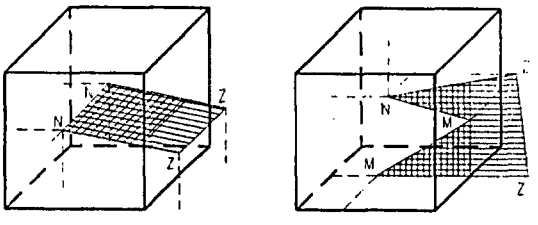
$$OP = NP + MP \; ; \; ZP = 0 \qquad (1)$$

lahko sklepamo, da se ploskev telesa nahaja znotraj oktanta (slika 6). Ploskev ne more ležati na eni izmed ploskev oktanta, kljub temu, da lahko vsa oglišča ploskve telesa ležijo na meji oktanta, saj bi ploskev izločili že v drugem koraku, s testom ravnin ploskev telesa (slika 4).

Če se vsaj eno oglišče ploskve telesa nahaja zunaj oktanta in vsaj eno oglišče znotraj oktanta, oziroma če velja

$$OP = NP + MP + ZP \; ; \; ZP \neq 0 \; , \; NP \neq 0 \qquad (2)$$

ploskev telesa seka oktant, ne glede na to ali je konveksna ali konkavna in ali ima luknje ali ne (slika 7).



a) OP=NP+ZP
MP=0, NP≠0, ZP≠0

b) OP=NP+MP+ZP
NP≠0, MP≠0, ZP≠0

Slika 7 Ploskev seka oktant

Ko ugotovimo, da se vsaj ena od ploskev telesa nahaja znotraj oktanta ali seka oktant, je barva vozlišča osmiškega drevesa siva ne glede na položaj ostalih ploskev telesa glede na oktant.

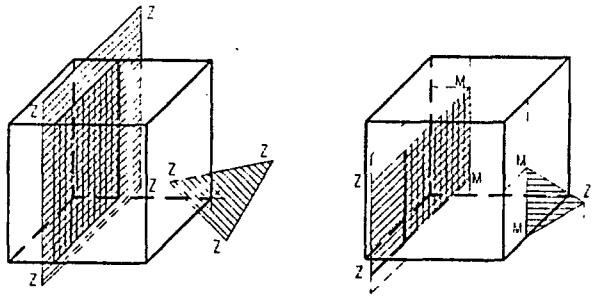
Če za nobeno od ploskev telesa, ki smo jih testirali v tem koraku ne velja enačba 1 ali enačba 2, ne moremo trditi, da katera od ploskev seka oktant, ali da so vse ploskve zunaj oktanta, zato nadaljujemo s četrtim korakom, ki vključuje test presečnega mnogokotnika.

3.4 Test presečnega mnogokotnika

Za ploskve telesa, ki jih testiramo v tem koraku velja, da imajo vsaj eno oglišče zunaj oktanta in nobenega oglišča znotraj oktanta oziroma velja

$$OP = MP + ZP \; ; \; NP = 0, \; ZP \neq 0 \qquad (3)$$

Ploskve telesa lahko ležijo zunaj oktanta ali sekajo oktant (slika 8), na rezultat testa pa lahko vpliva tudi konkavnost ploskve.



a) OP=ZP; MP=0, NP=0

b) OP=MP+ZP
NP=0, ZP≠0, MP≠0

Slika 8 Ploskve telesa ležijo zunaj oktanta ali sekajo oktant

Cilj tega testa je isti kot v tretjem koraku, le da je postopek ugotavljanja presečišč bolj natančen, saj moramo upoštevati, da ima ploskev lahko luknje in ni nujno konveksna. Ta test sestoji iz štirih korakov - testov, ki se nanašajo na posamezno ploskev.

Da bi ugotovili položaj ploskve telesa, si pomagamo s presečnim mnogokotnikom, ki ga določajo točke, v katerih ravnina ploskve telesa seka oktant. Ravnina ploskve telesa seka ravnine šestih ploskev oktanta v šestih premicah. Točke, v katerih se teh šest premic seka, določajo oglišča presečnega mnogokotnika, ki leži v ravnini ploskve telesa. Ploskev telesa seka oktant le takrat, ko seka presečni mnogokotnik.

V prvem koraku testiramo število oglišč presečnega mnogokotnika, ki je odvisno od položaja ravnine ploskve telesa glede na oktant. Na podlagi rezultatov tega testa, izločimo ploskve, ki ležijo zunaj oktanta.

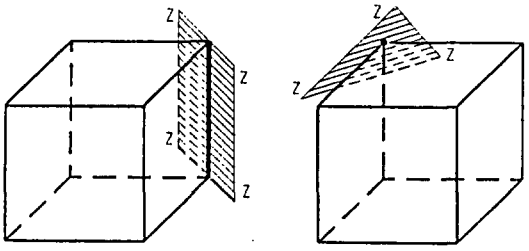
Naj je OPM število vseh oglišč presečnega mnokotnika. Ko je število oglišč presečnega mnogokotnika manjše ali enako dva, oziroma če velja

$$OPM \leq 2 \; ; \; OPM \neq 0 \qquad (4)$$

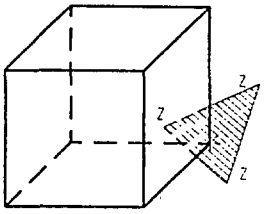
se ravnina ploskve telesa dotika oktanta. Ko je število oglišč presečnega mnogokotnika enako nič, oziroma če velja

OPM = 0 (5)

leži ravnina ploskve telesa zunaj oktanta. Na podlagi tega sklepamo, da ploskev ne seka oktanta, temveč zagotovo leži zunaj oktanta (slika 9).



a)OPM = 2 b) OPM = 1



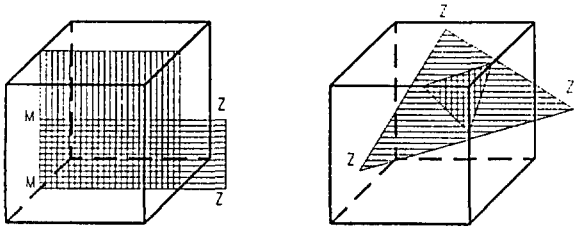
c) OPM = 0

Slika 9 Ploskev telesa leži zunaj oktanta

Ko je število oglišč presečnega mnogokotnika večje od dva, oziroma velja

OPM > 2 (6)

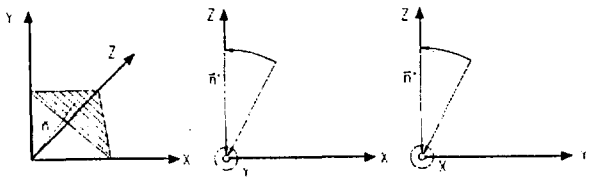
ravnina ploskve seka oktant in preidemo na drugi korak (slika 10).



a) OPM = 4 b) OPM = 3

Slika 10 Ploskev telesa seka oktant

V drugem koraku moramo določiti položaj presečnega mnogokotnika glede na ploskev telesa. Da bi poenostavili vsebnostni test, s katerim določamo položaj oglišč presečnega mnogokotnika, zavrtimo ravnino, v kateri ležita ploskev in presečni mnogokotnik tako, da je vzporedna s koordinatno ravnino XY. To opravimo tako, da zavrtimo normalni vektor ploskve telesa v koordinatno os Z (slika 11).



Slika 11 Rotacija normale ploskve telesa v koordinatno os Z

Sedaj opravimo vsebnostni test, da ugotovimo položaj presečnega mnogokotnika glede na ploskev telesa. Presečni mnogokotnik je na vseh nadaljnih slikah označen z navpično šrafuro, ploskev telesa pa z vodoravno šrafuro.

Naj bo ZPM število oglišč presečnega mnogokotnika zunaj ploskve, MPM naj bo število oglišč na meji ploskve in NPM število oglišč znotraj ploskve telesa. Presečni mnogokotnik lahko leži:

- strogo znotraj ploskve telesa,
- seka ploskev telesa,
- znotraj ploskve telesa,
- zunaj ploskve telesa.

Če so vsa oglišča presečnega mnogokotnika znotraj ploskve, oziroma če velja

OPM = NPM ; MPM = 0, ZPM = 0 (7)

potem je presečni mnogokotnik strogo znotraj ploskve telesa (slika 12), vendar barve vozlišča ne moremo določiti, saj je ta odvisna od medsebojne lege presečnega mnogokotnika in lukenj v ploskvi.

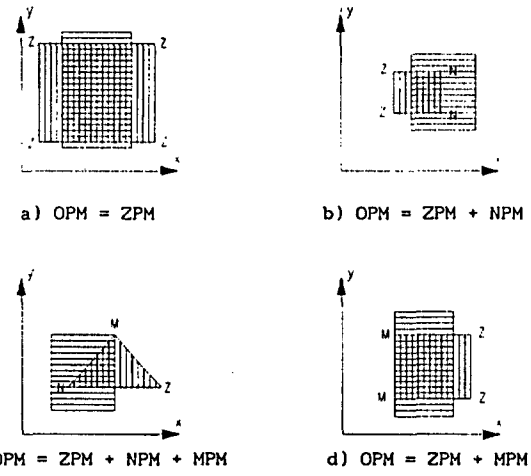
Če ploskev nima lukenj, potem ploskev seka oktant (slika 12a) in je barva vozlišča osmiškega drevesa siva. V primeru, ko ima ploskev luknje (slika 12b), uporabimo test lukenj ploskve telesa in določimo položaj ploskve glede na oktant.

V primeru, ko enačba 7 ni izpolnjena, preidemo na tretji korak. V tem koraku želimo ugotoviti, ali presečni mnogokotnik seka ploskev telesa.

a) Ploskev nima luknje b) Ploskev z luknjo

Slika 12 Presečni mnogokotnik strogo znotraj ploskve telesa

Če vsaj en rob presečnega mnogokotnika seka kateri koli rob ploskve telesa, presečni mnogokotnik seka ploskev (slika 13), ne glede na to ali je ploskev konkavna ali konveksna in ali ima luknje ali ne.



a) OPM = ZPM b) OPM = ZPM + NPM

c) OPM = ZPM + NPM + MPM d) OPM = ZPM + MPM

Slika 13 Presečni mnogokotnik seka ploskev telesa

Ploskev telesa seka oktant, barva vozlišča osmiškega drevesa je siva in prekinemo s testiranjem presečišč.

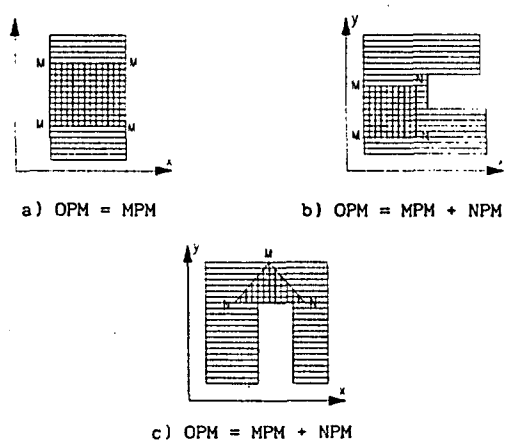
Če noben rob presečnega mnogokotnika ne seka katerega koli roba ploskve telesa, preidemo na četrti korak.

V četrtem koraku ugotavljamo položaj presečnega mnogokotnika glede na ploskev telesa na osnovi lege njegovih oglišč.

Če so vsa oglišča presečnega mnogokotnika na meji ploskve telesa, oziroma če velja

$$OPM = MPM \; ; \; NPM = 0, \; ZPM = 0 \qquad (8)$$

je rezultat testa odvisen od lege središčne točke presečnega mnogokotnika (slika 14a, 15a).

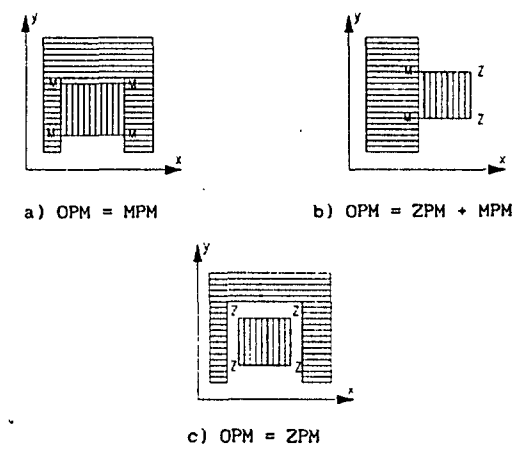


a) OPM = MPM b) OPM = MPM + NPM

c) OPM = MPM + NPM

Slika 14 Presečni mnogokotnik znotraj ploskve telesa

Če središčna točka presečnega mnogokotnika leži znotraj ploskve telesa, leži tudi presečni mnogokotnik znotraj ploskve in ploskev seka oktant (slika 14a). Barva presečišča je siva in prekinemo s testiranjem presečišč. Če leži središčna točka presečnega mnogokotnika zunaj ploskve, leži tudi presečni mnogokotnik zunaj ploskve telesa in ploskev ne seka oktanta (slika 15a).



a) OPM = MPM b) OPM = ZPM + MPM

c) OPM = ZPM

Slika 15 Presečni mnogokotnik zunaj ploskve telesa

Če je vsaj eno oglišče presečnega mnogokotnika znotraj ploskve, oziroma če velja

$$OPM = NPM + MPM \; ; \; ZPM = 0, \; NPM \neq 0 \qquad (9)$$

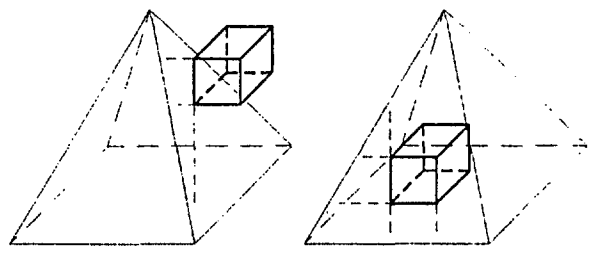
leži presečni mnogokotnik znotraj ploskve in ploskev seka oktant (slika 14b,c). Barva presečišča, oziroma vozlišča osmiškega drevesa je siva in prekinemo s testiranjem presečišč.

Če je vsaj eno oglišče presečnega mnogokotnika zunaj ploskve telesa, oziroma če velja

$$OPM = ZPM + MPM \; ; \; ZPM \neq 0, \; NPM = 0 \qquad (10)$$

leži presečni mnogokotnik zunaj ploskve (slika 15b,c) in ploskev ne seka oktanta.

Rezultat testa presečnega mnogokotnika nam vrne barvo presečišča oziroma vozlišča osmiškega drevesa, če vsaj ena ploskev telesa seka oktant. V primeru, ko nismo dobili barve presečišča, so vse ploskve, ki smo jih testirali v tem koraku, zunaj oktanta (slika 16) in oktant lahko leži znotraj telesa, ali zunaj telesa.



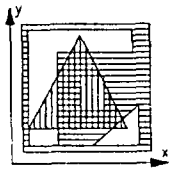
a) Oktant je zunaj telesa b) Oktant je znotraj telesa

Slika 16 Medsebojna lega telesa in oktanta

Uporabimo vsebnostni test oglišč oktanta v telesu, da bi ugotovili, katera od trditev je pravilna. Nato skladno s tehniko označevanja predpišemo vozlišču osmiškega drevesa črno ali belo barvo.

3.5 Test lukenj ploskve telesa

Presečni mnogokotnik je strogo znotraj ploskve, ko velja enačba 7. Samo v tem primeru lahko luknja ploskve prekriva presečni mnogokotnik (slika 12b) in ploskev ne seka oktanta. Luknja v ploskvi se ne more dotikati roba ploskve niti sekati ploskve, zato v primerih, ko enačba 7 ne velja, presečni mnogokotnik ne more biti v celoti prekrit z luknjo (slike 13, 14, 15). Ploskev ima lahko več lukenj (slika 17), zato test ponavljamo za vsako luknjo v ploskvi, dokler ne določimo barve presečišča.



Slika 17 Ploskev telesa z luknjami in lega presečnega mnogokotnika

Ugotoviti moramo ali luknja v celoti prekriva presečni mnogokotnik, saj v vseh ostalih primerih ploskev vsaj delno prekriva presečni mnogokotnik, kar pomeni, da ploskev seka oktant.

Presečni mnogokotnik lahko:

- seka luknjo,
- leži znotraj luknje,
- prekriva luknjo,
- leži strogo zunaj luknje.

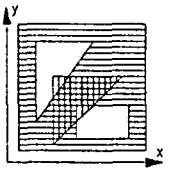
Določanje medsebojne lege luknje ploskve in presečnega mnogokotnika poteka v treh korakih.

V prvem koraku uporabimo test minimax, s katerim določimo prekrivanje omejitvenih pravokotnikov presečnega mnogokotnika in luknje ploskve. Če se omejitvena pravokotnika ne prekrivata, nadaljujemo z naslednjo luknjo ploskve. Če za vse luknje ploskve velja, da se njihovi omejitveni pravokotniki ne prekrivajo z omejitvenim pravokotnikom presečnega mnogokotnika, potem presečni mnogokotnik seka, oziroma leži na ploskvi telesa in ta seka oktant. Če se omejitvena pravokotnika luknje ploskve in presečnega mnogokotnika prekrivata, preidemo na naslednji korak.

V drugem koraku preverjamo, ali kateri od robov presečnega mnogokotnika seka kateri koli rob luknje ploskve. Če je test pritrdilen, leži presečni mnogokotnik delno na ploskvi telesa in ploskev seka oktant ne glede na to, ali ima ploskev še kakšno luknjo in kje na ploskvi se ta luknja nahaja (slika 18). Zato lahko v tem primeru zaključimo s testom lukenj ploskve.

Če je test sekanja robov negativen, nadaljujemo s tretjim korakom.

V tretjem koraku želimo ugotoviti, ali luknja ploskve popolnoma prekriva presečni mnogokotnik



Slika 18 Presečni mnogokotnik seka luknjo ploskve

oziroma, ali leži presečni mnogokotnik v celoti znotraj luknje ploskve. V tem primeru ploskev ne seka oktanta, ne glede na ostale luknje ploskve in lahko zaključimo s testiranjem. Če omejitveni pravokotnik presečnega mnogokotnika omejuje omejitveni pravokotnik luknje ploskve, ugotavljamo vsebnost luknje ploskve v presečnem mnogokotniku, drugače pa ugotavljamo vsebnost presečnega mnogokotnika znotraj luknje ploskve. Ta vsebnostni test je enak v obeh primerih, zato ga lahko posplošimo na ugotavljanje vsebnosti mnogokotnika M1 v mnogokotniku M2.

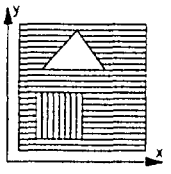
Naj bo OM1 število oglišč mnogokotnika M1 in OM2 število oglišč mnogokotnika M2. Število oglišč mnogokotnika M1 zunaj mnogokotnika M2 označimo z ZM1, število oglišč M1 znotraj M2 označimo z NM1, število oglišč, ki ležijo v ogliščih M2 označimo z VM1 in število oglišč, ki ležijo na robovih M2 označimo z RM1. Če vsa oglišča M1 ležijo v ogliščih M2, oziroma če velja

$$OM1 = VM1 \; ; \; RM1 = 0, \; ZM1 = 0, \; NM1 = 0 \qquad (11)$$

in

$$OM1 = OM2 \qquad (12)$$

sta mnogokotnika enaka in se popolnoma prekrivata. V tem primeru leži presečni mnogokotnik v celoti znotraj luknje ploskve, ne glede na to kateri mnogokotnik označuje presečni mnogokotnik in kateri luknjo ploskve (slika 19). Ploskev telesa ne seka oktanta in prekinemo s testiranjem.

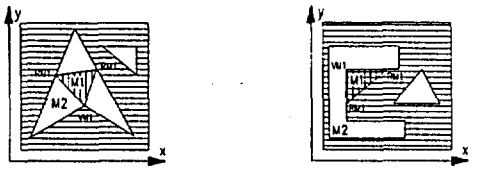


Slika 19 Mnogokotnika se popolnoma prekrivata

Če vsa oglišča mnogokotnika M1 ležijo v ogliščih M2 in na robovih M2, oziroma če velja

$$OM1 = VM1 + RM1 \; ; \; ZM1 = 0, \; NM1 = 0 \qquad (13)$$

je mnogokotnik M1 lahko zunaj ali znotraj mnogokotnika M2. Uporabimo vsebnostni test središčne točke mnogokotnika M1 v mnogokotniku M2, da ugotovimo, katera trditev je pravilna.

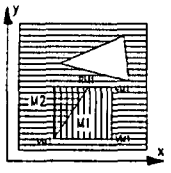


a) Presečni mnogokotnik znotraj luknje

b) Presečni mnogokotnik zunaj luknje

Slika 20 M1 presečni mnogokotnik, M2 luknja ploskve

V primeru, ko M1 označuje presečni mnogokotnik in M2 luknjo ploskve (slika 20), imamo naslednjo rešitev: Če presečni mnogokotnik leži znotraj luknje (slika 20a), ploskev ne seka oktanta. Če leži presečni mnogokotnik zunaj luknje ploskve, ploskev seka oktant (slika 20b).



Slika 21 M1 luknja ploskve, M2 presečni mnogokotnik

V primeru, ko M1 označuje luknjo ploskve in M2 presečni mnogokotnik, je rešitev naslednja: luknja ploskve lahko leži le znotraj presečnega mnogokotnika, ker je oblika presečnega mnogokotnika omejena. Zato presečni mnogokotnik leži delno na ploskvi telesa in ta seka oktant (slika 21).

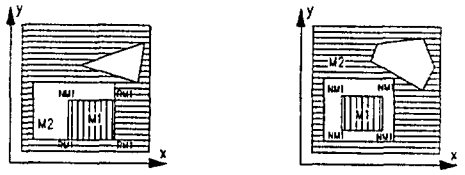
Če vsaj eno oglišče mnogokotnika M1 leži znotraj mnogokotnika M2, oziroma če velja

$$OM1 = NM1 + RM1 + VM1 \; ; \; NM1 \neq 0, \; ZM1 = 0 \qquad (14)$$

leži mnogokotnik M1 znotraj mnogokotnika M2.

Če leži presečni mnogokotnik (M1) znotraj luknje ploskve (M2), luknja ploskve prekriva presečni mnogokotnik in ploskev ne seka oktanta (slika 22).

Če leži luknja ploskve (M1) znotraj presečnega mnogokotnika (M2), leži presečni mnogokotnik delno na ploskvi telesa in ploskev seka oktant (slika 23).



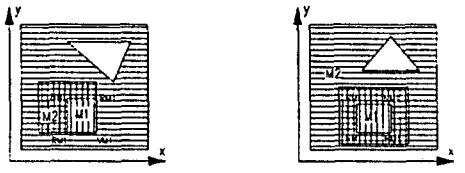
a) RM1 ≠ 0, VM1 ≠ 0

b) RM1 = 0, VM1 = 0

Slika 22 Presečni mnogokotnik (M1) znotraj luknje ploskve (M2)

Če vsaj eno oglišče mnogokotnika M1 leži zunaj mnogokotnika M2, oziroma če velja

$$OM1 = ZM1 + RM1 + VM1 \; ; \; ZM1 \neq 0, \; NM1 = 0 \qquad (15)$$



a) RM1 ≠ 0, VM1 ≠ 0

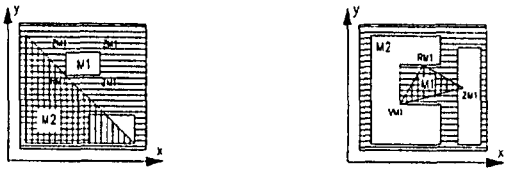
b) RM1 = 0, VM1 = 0

Slika 23 Luknja ploskve (M1) znotraj presečnega mnogokotnika (M2)

in

$$RM1 \neq 0 \quad ali \; VM1 \neq 0 \qquad (16)$$

leži mnogokotnik M1 zunaj mnogokotnika M2 in mnogokotnika se dotikata. V tem primeru, ne glede na to, kateri mnogokotnik določa luknjo ploskve in kateri presečni mnogokotnik, ugotovimo, da presečni mnogokotnik leži zunaj luknje ploskve, vendar ga nobena druga luknja ploskve ne more popolnoma prekrivati. Ploskev seka oktant in prekinemo s testiranjem lukenj (slika 24).



a) Presečni mnogokotnik (M1) luknja ploskve (M2)

b) Luknja ploskve (M1) presečni mnogokotnik (M2)

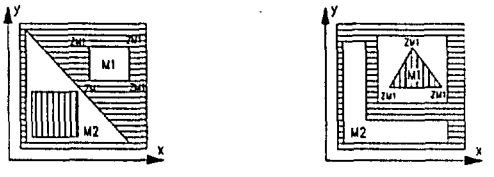
Slika 24 Mnogokotnik M1 zunaj mnogokotnika M2

Če vsa oglišča mnogokotnika M1 ležijo zunaj mnogokotnika M2 oziroma, če velja

$$OM1 = ZM1 \; ; \; VM1 = 0, \; RM1 = 0, \; NM1 = 0 \qquad (17)$$

leži mnogokotnik M1 strogo zunaj mnogokotnika M2 (slika 25).

Ne glede na to, kateri mnogokotnik označuje presečni mnogokotnik in kateri luknjo ploskve, leži presečni mnogokotnik strogo zunaj luknje ploskve, kar pomeni, da lahko ostale luknje ploskve vplivajo na rešitev. Če ploskev nima več lukenj oziroma, če smo obdelali vse luknje ploskve, ploskev seka oktant, drugače pa ponovimo test lukenj za naslednjo luknjo ploskve.



a) Presečni mnogokotnik M1 luknja ploskve M2

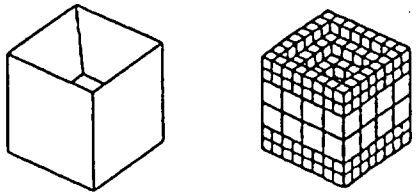
b) Luknja ploskve M1 presečni mnogokotnik M2

Slika 25 Mnogokotnik M1 strogo zunaj mnogokotnika M2

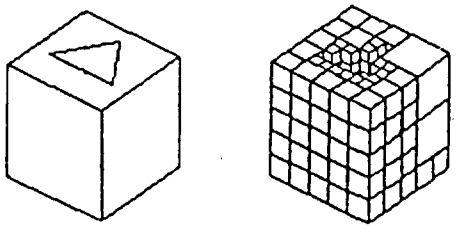
4 Zaključek

Testiranje presečišč med elementi telesa in osmiškega drevesa je časovno in prostorsko zelo zahtevno. Ta zahtevnost je odvisna predvsem od oblike telesa oziroma od števila vozlišč v osmiškem drevesu in od morebitne konkavnosti in luknjavosti telesa. Naše nadaljnje raziskovalno delo bo temeljilo na izboljšanju, oziroma optimiranju algoritma, kot tudi na realizaciji algoritma za pretvorbo iz predstavitve z osmiškim drevesom v predstavitev z ovojnico.

Program je napisan v pascalu in vključen v Eksperimentalni geometrijski modelirnik teles (EGM) [ŽALI89b]. Razvili smo ga na računalniku VAX-8800, za vizualizacijo pa smo uporabili grafični terminal TEK 4207. Rezultati programa so predstavljeni na slikah 26 in 27, kjer smo telesa prikazali z žičnim modelom. Za prikaz telesa in osmiškega drevesa z odstranjenimi zakritimi robovi smo uporabili razširjen Robertsov algoritem za odstranjevanje zakritih robov in ploskev ([ŠOBO88], [ŠOBO89]).



Slika 27 Primer konkavnega telesa



Slika 26 Primer telesa z luknjo

5 Literatura

[CHIY85] Chiyokura, H., F. Kimura, "A Method of Representing the Solid Design Process", IEEE CG&A, Vol. 5, No. 4, 1985, pp. 32 - 41.

[CHIY87] Chiyokura, H., "An Extended Rounding Operation for Modeling Solids with Free-Form Surfaces", IEEE CG&A, Vol. 7, No. 12, 1987, pp. 27 - 36.

[CLIF87] Clifford, A. S. and H. Samet, "Optimal Quadtree Construction Algorithms", Computer Vision, Graphics, and Image Processing, Vol. 37, No. 3, March 1987, pp. 402 - 419.

[GUID88] Guid, N., "Računalniška grafika" Tehniška fakulteta Maribor, Maribor 1988.

[HILL82] Hillyard, R., "The Build Group of Solid Modelers", IEEE CG&A, Vol. 2, No. 2, 1982, pp. 43 - 52.

[HOME88] Homer, H. C. and T. S. Huang, "A Survey of Construction and Manipulation of Octree", Computer Vision, Graphics, and Image Processing, Vol.43, No.3, September 1988, pp. 409 - 431.

[KAUF88] Kaufman, A. and R. Bakalash, "Memory and Processing Architecture for 3D Voxel - Based Imagery", IEEE CG&A, Vol. 8. No. 6. 1988, pp. 10 - 23.

[MÄNT82] Mäntylä, M. and R. Sulonen, "GWB: A Solid Modeler with Euler Operators", IEEE CG&A Vol. 2, No. 7, 1982, pp. 17 - 31.

[MÄNT83] Mäntylä, M., "SET operations of GWB", Graphics Forum, No. 2 - 3, 1983, pp. 122 - 134.

[MÄNT88] Mäntylä, M., "An Introduction to Solid Modeling", Computer Science Press, Rockville, MD (1988).

[MILL89] Miller, R. J., "Architectural Issues in Solid Modelers", IEEE CG&A, Vol. 9, No. 5, 1989, pp. 72 - 87.

[MORT85] Mortenson, M. E., "Geometric Modeling", John Wiley & Sons, New York, 1985.

[PRAT83] Pratt, M. J., "Interactive geometric modelling for CAD/CAM", Eurographics '83 Tuturials, Zagreb, 1983.

[REQU82] Requicha, A. A. G., "Solid Modeling: A Historical Summary and Contemporary Assessment", IEEE CG&A, Vol. 2, No. 2, 1982, pp. 9 - 24.

[REQU83] Requicha, A. A. G, and H. B. Völcker, "Solid Modeling: Current Status and Research Directions", IEEE CG&A, Vol. 3, No. 5, 1983, pp. 25 - 37.

[ŠOBO88] Šobot, P., B. Žalik, N. Guid, "Extended Robert's hidden line / hidden surface algorithm for computer graphics", X. International Symposium Computer at the University, Cavtat 1988, pp. 7.5/1-4.

[ŠOBO89] Šobot, P., B. Žalik, N. Guid, "Analiza in uporaba razširjenega Robertsovega algoritma, ki rešuje problem zakritih robov pri mnogokotniških pravilnih telesih", XIII. Simpozijum o informacionim tehnologijama, Sarajevo 1989, pp. 227/1-10.

[WEIL85] Weiler, K., "Edge-Based Data Structures for Solid Modeling in Curved-Surface Environmenst", IEEE CG&A, Vol. 5, No. 1, 1985, pp. 21 - 40.

[WILS85] Wilson, P. R., "Euler Formulas and Geometric Modeling", IEEE CG&A, Vol. 5, No. 8, August 1985, pp. 24 - 36.

[ŽALI89a] Žalik, B., N. Guid, and P. Šobot, "Experimental geometric modeler based on boundary representation", XI. International Symposium Computer at the University, Cavtat 1989, pp. 7.6/1-6.

[ŽALI89b] Žalik, B., "Geometrijski modelirnik z Eulerjevimi operatorji", Magistrsko delo, Tehniška fakulteta, VTO ERI, Maribor, 1990.

[ŽALI90] Žalik, B., N. Guid, "Vgradnja Eulerjevih operatorjev in njihova uporaba pri tvorbi teles s translacijskim pomikanjem", Informatica, Ljubljana, Vol. 14, No. 2, 1990, pp. 32 - 38.

* *

* *

RAČUNALNIŠKI PODPROGRAMI ZA RAČUNANJE LASTNIH VREDNOSTI IN LASTNIH VEKTORJEV

INFORMATICA 2/91

Keywords: principal components analysis, eigenvalues, eigenvectors, diagonalization method, subroutine

Vesna Čančer
Ekonomsko-poslovna fakulteta v Mariboru
Razlagova 14, 62000 Maribor

V članku obravnavamo računalniške podprograme za računanje lastnih vrednosti in lastnih vektorjev simetrične korelacijske matrike, ki jih potrebujemo pri analizi glavnih komponent. V njem je uporabljena modifikacija Jacobijevega postopka, imenovana "Pope in Tompkins", ki je primerna za računalniško programiranje. Čeprav se uporabljajo hitrejše metode s tridiagonalizirano korelacijsko matriko, se Jacobijeva metoda še pojavlja v sodobnih podprogramskih paketih, saj omogoča natančne izračune lastnih vrednosti in njim pripadajočih lastnih vektorjev. Delovanje podprograma smo ponazorili s primerom in izpisali končni rezultat.

ABSTRACT

The following paper deals with the fortran computer subroutines for computing eigenvalues and eigenvectors of a real symmetric matrix, which are necessary for the principal components analysis. Because of its convenience for computer programming, diagonalization method originated by Jacobi and modified by the iterative "Pope and Tompkins" technique is used. In spite of some quicker methods, based on the threediagonalization matrix, Jacobi's method still appears in the modern subroutine packages because it provides precise definitions of eigenvalues and associated eigenvectors. A numeric example illustrating the application of the described procedure and given computer subroutine is presented.

1. UVOD

Analiza glavnih komponent je statistična tehnika, s katero n statističnih znakov x_1, x_2, ..., x_n linearno in ortogonalno nadomestimo z enakim številom nekoreliranih glavnih komponent y_1, y_2, ..., y_n [3], [8]. Transformacija temelji na iskanju lastnih vrednosti in lastnih vektorjev kovariančne matrike ali, kot v našem primeru, korelacijske matrike A, ki je kovariančna matrika standardiziranih spremenljivk x_1, x_2, ..., x_n [3], [4], [6].

Vzemimo n-razsežni vektor z elementi r_1, r_2, ..., r_n in poskušajmo rešiti matrično enačbo

$$\begin{bmatrix} 1 & a_{12} & \dots & a_{1n} \\ a_{21} & 1 & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = \lambda \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix}, \qquad (1.1)$$

kjer je A matrika korelacijskih koeficientov in λ poljubno število [8].

To enačbo je mogoče zapisati v obliki sistema n linearnih enačb, ki jih uredimo tako, da vse člene z neznankami r_1, r_2, ..., r_n prenesemo na levo stran. Dobimo homogen sistem enačb; desne strani vseh enačb so namreč enake nič [8], [10]. Tak sistem ima pri poljubni vrednosti λ trivialno rešitev $r_1 = r_2 = \dots = r_n = 0$. Netrivialna rešitev eksistira kvečjemu tedaj, kadar je

determinanta koeficientov A = 0 [10]. Če za λ vzamemo tako vrednost, da enačbe tvorijo odvisen sistem, dobimo neskončno mnogo rešitev. Sistem homogenih enačb je torej odvisen, če je determinanta sistema enaka nič:

$$\begin{vmatrix} 1-\lambda & a_{12} & \dots & a_{1n} \\ a_{21} & 1-\lambda & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & 1-\lambda \end{vmatrix} = 0 \qquad (1.2)$$

To je karakteristična enačba korelacijske matrike, ki jo lahko zapišemo v obliki algebrajske enačbe n-te stopnje [8]. Rešitve karakteristične enačbe (1.2) $\lambda_1, \lambda_2, \dots, \lambda_n$ imenujemo lastne vrednosti korelacijske matrike. Predpostavimo, da so lastne vrednosti realne in različne. Uredimo jih po velikosti:

$$\lambda_1 > \lambda_2 > \dots > \lambda_n. \qquad (1.3)$$

Vsaki lastni vrednosti λ_i, i = 1, 2, ..., n, pripada lastni vektor r_i, ki je rešitev matrične enačbe:

$$A\, r_i = \lambda_i\, r_i.$$

Lastnosti lastnih vrednosti in lastnih vektorjev korelacijske matrike so podrobneje opisane npr. v [3], [4], [8] in [10].

S podprogramom EIGEN [5] se računajo lastne vrednosti in lastni vektorji simetrične korelacijske matrike, katere elementi so v podprogramu urejeni v vektor, označen s formalno spremenljivko A. V tej spremenljivki se izračunajo tudi lastne vrednosti. Tvori se matrika lastnih vektorjev, katere elementi so v podprogramu urejeni v vektorju R.

2. OSNOVE UPORABLJENE METODE

Uporabljena je za računalniško programiranje primerna modifikacija Jacobijeve metode za določanje lastnih vrednosti, t. j. algoritem Pope in Tompkins [4], ki ga je za računalnike priredil von Neumann. Bistvo Jacobijevega postopka za ugotavljanje lastnih vrednosti je, da se z zaporedjem ortogonalnih transformacij korelacijska matrika pretvori v diagonalno matriko lastnih vrednosti:

$$\begin{aligned} A_1 &= R_1' A R_1 \\ A_2 &= R_2' A_1 R_2 \\ &\vdots \\ A_g &= R_g' A_{g-1} R_g = \Lambda, \end{aligned}$$

kjer je A simetrična korelacijska matrika,
R transformacijska matrika ortogonalnih transformacij
in Λ diagonalna matrika lastnih vrednosti.

Z vsako ortogonalno transformacijo lahko en nediagonalni element reduciramo na nič. Proces se prične z redukcijo največjega nediagonalnega elementa, nato se reducira drugi največji nediagonalni element itd., dokler niso vsi nediagonalni elementi reducirani na nič. V vsakem zaporedju moramo poznati elementarno transformacijsko matriko, s katero bomo izbrani element v matriki reducirali na nič [10], [11]:

$$R_i = \begin{bmatrix} \cos\Theta & \sin\Theta & 0 & \dots & 0 \\ -\sin\Theta & \cos\Theta & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \qquad (2.1)$$

Kot Θ, za katerega moramo zarotirati korelacijsko matriko A, da bo element a_{lm} postal nič, je podan z izrazom:

$$\operatorname{tang} 2\Theta = -2\, a_{lm} / (a_{ll} - a_{mm}). \qquad (2.2)$$

Ortogonalna transformacije je z geometričnega vidika podorobneje razložena v [3], [4], [10] in [11]; v članku navajamo le ključne enačbe, ki so uporabljene v podprogramu EIGEN [5]. Kot Θ je določen s predmultiplikacijo in postmultiplikacijo matrike A s transformacijsko matriko R in pogojem, da mora biti rezultat takšne multiplikacije nič:

$$(a_{ll} - a_{mm}) \sin\Theta \cos\Theta + a_{lm}(\cos^2\Theta - \sin^2\Theta) = 0. \qquad (2.3)$$

Upoštevajmo vrednosti za sin 2 Θ in cos 2 Θ:

$$1/2\,(a_{ll} - a_{mm}) \sin 2\Theta + a_{lm} \cos 2\Theta = 0, \qquad (2.4)$$

iz česar sledi:

$$-2\, a_{lm} / (a_{ll} - a_{mm}) = \sin 2\Theta / \cos 2\Theta = \operatorname{tang} 2\Theta. \qquad (2.5)$$

Nato iz izraza za tangens dobimo vrednosti sin Θ in cos Θ, t.j. elemente transformacijske matrike R. Bistvena razlika med izvirno Jacobijevo metodo in algoritmom Pope in Tompkins je v tem, da se z njim na nič ne reducira le en element, pač pa se z določitvijo minimalne meje tolerance zaporedno eliminirajo vsi nediagonalni elementi, ki so večji od te meje. Nato se določi nova minimalna meja tolerance, ki je nižja od prejšnje meje. Proces se nadaljuje vse do končne meje tolerance napake za lastne vrednosti. Algoritem tvori osem korakov:

V 1. koraku poiščemo vsote kvadratov vseh nediagonalnih elementov korelacijske matrike in kvadratni koren te vsote, t. j. začetno normo

$$\nu_I = \left(\sum_{i \le k} 2\, a_{ik}^2 \right)^{1/2}, \qquad (2.6)$$

ki ji je v podprogramu prirejena formalna spremenljivka ANORM.

V 2. koraku določimo končno mejo tolerance napake za lastne vrednosti:

$$\nu_F = \frac{\nu_I \; 10^{-6}}{N}, \qquad (2.7)$$

kjer je N rang korelacijske matrike. V podprogramu je tej meji prirejena formalna spremenljivka ANRMX.

V 3. koraku tvorimo enotsko transformacijsko matriko R.

V 4. koraku določimo začetno minimalno mejo tolerance, tako da bodo vsi nediagonalni elementi, večji od te meje, rotirani na nič. V podprogramu ji je prirejena formalna spremenljivka THR.

V 5. koraku iščemo nediagonalne elemente, ki so po absolutni vrednosti večji od minimalne meje tolerance. Iskanje lahko omejimo na desno od glavne diagonale, ker je korelacijska matrika simetrična glede na glavno diagonalo. Vsak tak element je zreduciran na nič. Iskanje in reduciranje nadaljujemo, dokler vsi elementi, ki niso na diagonali, niso manjši ali enaki dani meji. Tedaj definiramo novo minimalno mejo tolerance in nadaljujemo opisani postopek, dokler ne dosežemo končne meje tolerance.

V 6. koraku določimo elemente zaporedne transformacijske matrike A po vsaki iteraciji; v tej matriki se izračunajo tudi lastne vrednosti korelacijske matrike.

V 7. koraku z ortogonalno transformacijo zajamemo tudi enotsko transformacijsko matriko R; v tej matriki se izračunajo tudi lastni vektorji korelacijske matrike.

V 8. koraku ugotavljamo lastne vektorje korelacijske matrike; lastne vrednosti in njim pripadajoče lastne vektorje uredimo po velikosti lastnih vrednosti.

V zbirki podprogramov IBM Corporation [5] so 5., 6. in 7. korak opisanega algoritma zaradi lažjega razumevanja podprograma EIGEN strnjeni v enačbe, ki jih bomo uporabili pri razlagi delovanja obravnavanega podprograma:

$$\lambda = -a_{lm} \qquad (2.8)$$

$$\mu = 1/2\,(a_{ll} - a_{mm}) \qquad (2.9)$$

$$\omega = \text{sign}(\mu)\,\frac{\lambda}{\sqrt{(\lambda^2 + \mu^2)}} \qquad (2.10)$$

$$\sin\Theta = \frac{\omega}{\sqrt{(2(1 + \sqrt{(1 - \omega^2)}))}} \qquad (2.11)$$

$$\cos\Theta = \sqrt{(1 - \sin^2\Theta)} \qquad (2.12)$$

$$B = a_{il}\cos\Theta - a_{im}\sin\Theta \qquad (2.13)$$

$$C = a_{il}\sin\Theta + a_{im}\cos\Theta \qquad (2.14)$$

$$B = r_{il}\cos\Theta - r_{im}\sin\Theta \qquad (2.15)$$

$$r_{im} = r_{il}\sin\Theta + r_{im}\cos\Theta \qquad (2.16)$$

$$r_{il} = B \qquad (2.17)$$

$$a_{ll} = a_{ll}\cos^2\Theta + a_{mm}\sin^2\Theta - 2a_{lm}\sin\Theta\cos\Theta \qquad (2.18)$$

$$a_{mm} = a_{ll}\sin^2\Theta + a_{mm}\cos^2\Theta + 2a_{lm}\sin\Theta\cos\Theta \qquad (2.19)$$

$$a_{lm} = (a_{ll} - a_{mm})\sin\Theta\cos\Theta + a_{lm}(\cos^2\Theta - \sin^2\Theta) \qquad (2.20)$$

3. OPIS DELOVANJA PODPROGRAMA EIGEN S PRIMEROM

V glavnem programu [2] kličemo podprogram EIGEN z ukazom CALL, s katerim se dejanski parametri iz glavnega programa priredijo formalnim parametrom v podprogramu:

formalni parametri v podprog.	p o m e n
A	vhodna simetrična korelacijska matrika, urejena kot vektor, ki se med računanjem v podprogramu uniči; izhodni vektor lastnih vrednosti, ki so elementi matrike lastnih vrednosti
R	izhodna matrika lastnih vektorjev, urejena kot vektor
N	rang matrik A in R, enak redu teh matrik zaradi neodvisnosti statističnih znakov
MV	vhodna spremenljivka za EIGEN, ki ji priredimo vrednost z ukazom DATA v glavnem programu in je lahko 0 za računanje lastnih vrednosti in vektorjev ali 1 za računanje zgolj lastnih vrednosti

V glavnem programu smo spremenljivki MV priredili vrednost 0.

V EIGEN-u niso potrebni drugi podprogrami ali funkcijski podprogrami. Za ponazoritev delovanja podprograma bomo navedli vmesne in izpisali končne rezultate. Za lažje razumevanje uporabljenega algoritma izpišimo podprogram EIGEN [5].

```
      SUBROUTINE EIGEN(A,R,N,MV)
      DIMENSION A(1),R(1)
C     TVORJENJE ENOTSKE MATRIKE
    5 RANGE=1.0E-6
      IF(MV-1) 10,25,10
   10 IQ=-N
      DO 20 J=1,N
      IQ=IQ+N
      DO 20 I=1,N
      IJ=IQ+I
      R(IJ)=0.0
      IF(I-J) 20,15,20
   15 R(IJ)=1.0
   20 CONTINUE
C     RAČUNANJE ZAČETNE IN KONČNE NORME (ANORM IN
C     ANRMX)
   25 ANORM=0.0
      DO 35 I=1,N
      DO 35 J=I,N
      IF(I-J) 30,35,30
   30 IA=I+(J*J-J)/2
      ANORM=ANORM+A(IA)*A(IA)
   35 CONTINUE
      IF(ANORM) 165,165,40
   40 ANORM=1.414*SQRT(ANORM)
      ANRMX=ANORM*RANGE/FLOAT(N)
C     INICIALIZACIJA INDIKATORJA IN RAČUNANJE
C     INPUTA (THR)
      IND=0
```

```
      THR=ANORM
   45 THR=THR/FLOAT(N)
   50 L=1
   55 M=L+1
C     RAČUNANJE SIN IN COS
   60 MQ=(M*M-M)/2
      LQ=(L*L-L)/2
      LM=L+MQ
   62 IF( ABS(A(LM))-THR) 130,65,65
   65 IND=1
      LL=L+LQ
      MM=M+MQ
      X=0.5*(A(LL)-A(MM))
   68 Y=-A(LM)/ SQRT(A(LM)*A(LM)+X*X)
      IF(X) 70,75,75
   70 Y=-Y
   75 SINX=Y/ SQRT(2.0*(1.0+( SQRT(1.0-Y*Y))))
      SINX2=SINX*SINX
   78 COSX= SQRT(1.0-SINX2)
      COSX2=COSX*COSX
      SINCS =SINX*COSX
C     ROTIRANJE L IN M STOLPCEV
      ILQ=N*(L-1)
      IMQ=N*(M-1)
      DO 125 I=1,N
      IQ=(I*I-I)/2
      IF(I-L) 80,115,80
   80 IF(I-M) 85,115,90
   85 IM=I+MQ
      GO TO 95
   90 IM=M+IQ
   95 IF(I-L) 100,105,105
  100 IL=I+LQ
      GO TO 110
  105 IL=L+IQ
  110 X=A(IL)*COSX-A(IM)*SINX
      A(IM)=A(IL)*SINX+A(IM)*COSX
      A(IL)=X
  115 IF(MV-1) 120,125,120
  120 ILR=ILQ+I
      IMR=IMQ+I
      X=R(ILR)*COSX-R(IMR)*SINX
      R(IMR)=R(ILR)*SINX+R(IMR)*COSX
      R(ILR)=X
  125 CONTINUE
      X=2.0*A(LM)*SINCS
      Y=A(LL)*COSX2+A(MM)*SINX2-X
      X=A(LL)*SINX2+A(MM)*COSX2+X
      A(LM)=(A(LL)-A(MM))*SINCS+A(LM)*(COSX2-SINX2)
      A(LL)=Y
      A(MM)=X
C     KONČNI TESTI
C     TEST, DA JE M ZADNJI STOLPEC
  130 IF(M-N) 135,140,135
  135 M=M+1
      GO TO 60
C     TEST, DA JE L PREDZADNJI STOLPEC
  140 IF(L-(N-1)) 145,150,145
  145 L=L+1
      GO TO 55
  150 IF(IND-1) 160,155,160
  155 IND=0
      GO TO 50
C     PRIMERJANJE INPUTA S KONČNO NORMO
  160 IF(THR-ANRMX) 165,165,45
C     SORTIRANJE LASTNIH VREDNOSTI IN LASTNIH  VEKTOR
  165 IQ=-N
      DO 185 I=1,N
      IQ=IQ+N
      LL=I+(I*I-I)/2
      JQ=N*(I-2)
      DO 185 J=I,N
      JQ=JQ+N
      MM=J+(J*J-J)/2
      IF(A(LL)-A(MM)) 170,185,185
  170 X=A(LL)
      A(LL)=A(MM)
      A(MM)=X
      IF(MV-1) 175,185,175
  175 DO 180 K=1,N
      ILR=IQ+K
      IMR=JQ+K
      X=R(ILR)
      R(ILR)=R(IMR)
  180 R(IMR)=X
  185 CONTINUE
      RETURN
      END
```

Simetrična vhodna korelacijska matrika, ki vsebuje korelacijske koeficiente med 3 statističnimi znaki, ki smo jih poprej standardizirali (družbenim produktom na prebivalca, izdatki za osebno potrošnjo na prebivalca in izdatki za investicije na prebivalca, podanimi v cenah leta 1972 in opazovanimi v Jugoslaviji v letih 1974 do 1988) [9]:

$$A = \begin{bmatrix} 1.000 & 0.887 & -0.125 \\ 0.887 & 1.000 & 0.307 \\ -0.125 & 0.307 & 1.000 \end{bmatrix} \qquad (3.1)$$

se v podprogramu EIGEN priredi kot vektor v formalni spremenljivki A z naslednjim zaporedjem elementov: 1., 0.887, 1., -0.125, 0.307, 1., torej je njihovo število N*(N+1)/2 = 6.

V 1. zapisu je definiran podprogram EIGEN in izhodni spremenljivki A in R ter vhodne spremenljivke A, N in MV. V podprogramu je upoštevana enojna natančnost. Če bi želeli dvojno natančnost, bi to definirali z ukazom

```
DOUBLE PRECISION    A,R,ANORM,ANRMX,THR,X,Y,SINX,SIX2,
                    COSX,COSX2,SINCS,RANGE,DSQRT,DABS
```

Tudi fortranske funkcije morajo biti dvojne natančnosti. SQRT v ukazih 40,68,75 in 78 moramo zamenjati z DSQRT. ABS v ukazu 62 moramo zamenjati v DABS. Popraviti moramo tudi konstanto v ukazu 5, in sicer na 1.0D-12.

Z ukazi 10-1 do 20 se v primeru, ko smo v glavnem programu spremenljivki MV dodelili vrednost 0, tvori enotska transformacijska matrika (2.1), katere elementi so urejeni v vektor R(IJ). Izvede se 3. korak algoritma Pope in Tompkins.

Z ukazi 25 do 40 se tvori začetna norma ANORM (2.6). Izvede se 1. korak algoritma Pope in Tompkins. V našem primeru je začetna norma 1.3389340.

V ukazu 40+1 se tvori končna meja tolerance napake za lastne vrednosti oz. končna norma ANRMX (2.7). Izvaja se torej 2. korak obravnavanega algoritma. V našem primeru je končna norma 0.0000004.

Z ukazi 45-2 do 55 poteka inicializacija indikatorjev IND in določanje začetne minimalne meje tolerance, kar sodi v 4. korak v algoritmu Pope in Tompkins. V našem primeru se je izračunalo 14 minimalnih mej tolerance THR; prva je bila 0.4463114, zadnja pa 0.0000003.

Z ukazi od 60 do 78+2 se računajo sinusi in kosinusi, nato pa se do 130-1 zamenjujejo L in M stolpci. V vsaki iteraciji se namreč

selekcionirajo nediagonalni elementi. To sodi v 5., 6. in 7. korak obravnavanega algoritma. Iz ukaza 62 je razvidno: če je nediagonalni element korelacijske matrike po absolutni vrednosti manjši od prej določene minimalne meje tolerance THR, ni reduciran na nič; če so nediagonalni elementi po absolutni vrednosti enaki ali večji od prej določene minimalne meje tolerance, pa se začne reduciranje tega elementa na nič. V našem primeru je v prvi iteraciji A(2), ki je 0.887, večji od THR, ki je 0.446, zato sledi reduciranje A(2) na nič. V ukazu 68-1 se izračuna X po (2.9). X predstavlja del izraza (2.4). V ukazu 68 se po (2.10), v katerem nastopa (2.8), izračuna Y, ki je potreben pri računanju vrednosti elementov transformacijske matrike R in A: sin Θ in cos Θ. V ukazu 75 se po (2.11) izračuna element transformacijske matrike R sin Θ in se zapomni v formalni spremenljivki SINX. V naslednjem ukazu se izračuna $\sin^2\Theta$ in se zapomni v formalni spremenljivki SINX2. V ukazu 78 se po (2.12) izračuna cos Θ in se zapomni v COSX. V naslednjih dveh ukazih se izračunata še $\cos^2\Theta$ in $\sin\Theta\cos\Theta$ in se zapomnita v spremenljivkah COSX2 in SINCS.

V 6. koraku se določijo elementi zaporednih transformacijskih matrik po vsaki iteraciji. V ukazu 110 se izračuna X po (2.13). V naslednjem ukazu se izračuna A(IM) po (2.14). Nato se A(IL) priredi X. Če se računajo le lastne vrednosti, se 7. korak algoritma preskoči, sicer pa se na ukazu 120 prične 7. korak v algoritmu Pope in Tompkins, v katerem je z ortogonalno transformacijo zajeta tudi transformacijska matrika R, ki je v podprogramu urejena v vektor. V ukazu 120+2 se po (2.15) izračuna X. V naslednjem ukazu se po (2.16) izračuna R(IMR) in nato po (2.17) se R(ILR). V našem primeru se v zadnjih dveh iteracijah v spremenljivkah R(1)-R(9) izračunajo elementi lastnih vektorjev, ki se niso dokončno urejeni. Ko se postopek izvede za N-ti I, se v ukazu 125+1 zaradi poenostavitve naslednjih dveh izrazov izračuna tretji element v izrazih (2.18) in (2.19). V navedenih izrazih se izračunata Y in X; to sodi v 6. korak obravnavanega algoritma. V naslednjem ukazu se po (2.20) izračuna A(LM), ki ustreza levi strani (2.3); izvede se torej 5. korak obravnavanega algoritma. Nato se v 6. koraku A(LL) priredi Y, izračunan po (2.18), A(MM) pa se priredi X, izračunan po (2.19). V našem primeru se v zadnjih dveh iteracijah izračunajo od nič različne lastne vrednosti, ki se niso urejene po velikosti, in se zapomnijo v spremenljivkah A(LL) in A(MM), torej v A(1), A(3) in A(6).

Sledijo testi za dokončanje postopka. Če je element A(LM) manjši od minimalne meje tolerance THR, se v ukazu 160 THR primerja s končno mejo tolerance napake za lastne vrednosti ANRMX. Če je minimalna meja tolerance THR manjša ali enaka končni meji tolerance napake za lastne vrednosti ANRMX, se nadaljuje z 8. korakom, sicer pa se računa nova minimalna meja tolerance THR in proces reduciranja se nadaljuje, vse dokler ni dosežena končna meja tolerance ANRMX. V našem primeru se proces reduciranja konča, ko je minimalna meja tolerance THR z vrednostjo 0.0000003 manjša od končne meje tolerance ANRMX z vrednostjo 0.0000004.

Sledi urejanje lastnih vrednosti po (1.3) in urejanje lastnim vrednostim pripadajočih lastnih vektorjev, kar sodi v 8. korak obravnavanega algoritma. Najprej se po velikosti uredijo lastne vrednosti. V našem primeru je lastna vrednost z indeksom 2 manjša od lastne vrednosti z indeksom 3, zato se indeksu z manjšo absolutno vrednostjo priredi večja lastna vrednost. Prvi, tretji in šesti element so diagonalni elementi v matriki lastnih vrednosti, ki je urejena v vektor, in so različni od nič: $\lambda_1 = 1.906$, $\lambda_2 = 1.076$ in $\lambda_3 = 0.017$. Če se računajo tudi lastni vektorji, je urejanje lastnih vektorjev odvisno od ureditve lastnih vrednosti. V našem primeru so se zamenjala zaporedna mesta 4. in 7., 5. in 8. ter 6. in 9. elementa R. Pri urejanju lastnih vrednosti je pomembna velikost lastne vrednosti, pri urejanju lastnih vektorjev pa je pomemben indeks, katerega vrednost je odvisna od ureditve lastnih vrednosti po velikosti. Z ukazoma RETURN in END se podprogram konča, izvajanje glavnega programa pa se nadaljuje z ukazom, ki sledi ukazu CALL v glavnem programu.

V tabeli 1 so urejene lastne vrednosti in njim pripadajoči lastni vektorji, izračunani za primer (3.1):

TABELA 1: Lastne vrednosti lastni vektorji

	LASTNE VREDNOSTI	LASTNI VEKTORJI 1	2	3
1	1.906	0.681	0.717	0.149
2	1.076	-0.317	0.105	0.942
3	0.017	-0.660	0.689	-0.299

4. ZAKLJUČEK

V nekaterih novejših računalniških podprogramih za računanje lastnih vrednosti in lastnih vektorjev korelacijske matrike se upoštevajo postopki, ki temeljijo na tridiagonaliziranih korelacijski matriki, kamor sodijo Givensov, Wilkinsov in Francisov postopek diagonalizacije [4]. Iz tridiagonaliziranih matrik hitreje ugotovimo lastne vrednosti kot iz netridiagonaliziranih matrik. Znana je Kaiserjeva modifikacija Francisove metode, na kateri temelji

npr. podprogram XKAISR [1]. Čeprav so ti postopki hitrejši od Jacobijeve metode, pa prav s slednjo dobimo zelo natančne izračune lastnih vrednosti. Zaradi primernosti za računalniško programiranje se Jacobijeva metoda še vedno uporablja. Novejši računalniški podprogrami, npr. XSMEIG [1], v katerih se tudi upošteva algoritem Pope in Tompkins, se od podprograma EIGEN razlikujejo le po uporabi novejših verzij FORTRAN-a, logika delovanja teh podprogramov pa je ista kot v podprogramu EIGEN [5].

LITERATURA

1. CHAGHAGHI, S. Francois: Time Series Package (TSPACK). G. Goos & J. Hartmanis, Springer--Verlag 1985.
2. ČANČER, Vesna: Analiza statističnega vzorca po glavnih komponentah. (Diplomsko delo) EPF, Maribor 1990.
3. DUNTEMAN, H. George: Principal Components Analysis. Sage Publications, Newbury Park 1989.
4. FULGOSI, Ante: Faktorska analiza. Školska knjiga, Zagreb 1984.
5. IBM Corporation: Scientific Subroutine Package GH20-0252--4. IBM 1968.
6. JAMNIK, Rajko: Uvod v matematično statistiko. DMFA Slovenije, Ljubljana 1976.
7. KAVKLER, Ivan: Teoretične osnove in primeri uporabe faktorske analize. (Magistrsko delo) Sveučilište u Zagrebu, Zagreb 1975.
8. MEŠKO, Ivan: Izbrana poglavja iz kvantitativne analize. VEKS, Maribor 1978.
9. Statistički godišnjak Jugoslavije 1990. Savezni zavod za statistiku, Beograd 1990.
10. VIDAV, Ivan: Višja matematika I. DZS, Ljubljana 1973.
11. VIDAV, Ivan: Višja matematika II. DZS, Ljubljana 1975.

Novice in zanimivosti o elektronskih slovarjih

Longman Dictionaries

Najuglednejša založniška hiša angleških slovarjev Longman Dictionaries (Burnt Mill, Harlow, Essex CM20 2JE, UK), ki je izdala svoj prvi angleški slovar leta 1755, razpolaga z največjim korpusom angleških besed (30 milijonov). Leta 1974 je začela razvijati **računalniški sistem** za izdajanje angleških slovarjev v angleščini (materinem jeziku), in sicer za poučevanje (English Language Teaching, kratko ELT) in splošno uporabo. Longmanovi leksikografi razpolagajo z vrsto orodij, kot so avtomatična križna referenca, definiranje slovarskega preizkušanja, validacija zadevnega področja in izboljšana kontrola dolžine slovarja z možnostjo spremembe načrta v zelo pozni fazi produkcijskega procesa.

Načelo Longmanove **definicije slovarjev** za ELT je omejenost slovarja na 2000 besed, ki naj bi jih obvladali študenti (dijaki) z dovolj dobrim znanjem angleškega jezika. Avtomatsko preizkušanje vseh definicij, ki odstopajo od definicij slovarja z 2000 besedami, je tako eden glavnih pripomočkov Longmanovih leksikografov.

Leta 1979 je Longman začel distribuirati trakove za svoje ELT slovarje, hkrati pa je postal Longmanov slovar sodobnega angleškega jezika (Longman Dictionary of Contemporary English, kratko LDOCE) dosegljiv tudi za lingvistične raziskave na univerzah, inštitutih in v komercialnih podjetjih, in sicer zlasti na področju gramatičnih kodov in njihovih povezav v različne gramatične sisteme, npr. v GPSG (Cambridge Computer Laboratory). Longman je danes vodilni dobavitelj slovarjev (in tezavrov) na področju raziskav.

S posebnimi kodi v posameznih poljih se omejuje subjektno polje, regija, čas ali register (avtorja Katz in Fodor). Te kode uporabljajo leksikografi pri sistematičnemu določanju individualnih pomenov. Obstaja dogovor za fino semantično analizo z zadostno natančnostjo in prečiščenostjo pomenov tudi za tuje učence angleškega jezika.

Prvi veliki angleški slovar Samuela Johnsona je izšel leta 1755. Johnsonov leksikografski princip je pri Longmanu v veljavi še danes. Leksikografe informirata dve računalniški podatkovni bazi, ko se odločajo za uvrščanje posameznih besed v posamezne slovarje, za obliko definicij in oblikovanje primerov (npr. upoštevanje neologizmov). Tako se jezik razvija na leksikalni ravni. T.i. Longman Lancaster English Language Corpus vsebuje 30 milijonov britanskih, ameriških, avstralskih in drugih variant angleščine in predstavlja bazo angleškega jezika v 20. stoletju.

Glavna principa Longmanovega korpusa sta zadevno polje in besedilne značilnosti: čas (kateri del 20. stoletja), raven (literarna/tehnična, srednja/laična ali popularna), medij (knjige, periodika ali kratkotrajni neobjavljeni materiali) in regija (ZDA, Britanija, Avstralija itd.).

40 % korpusa je namenjega fikciji (imaginativnemu), ostalo pa je kategorizirano kot informativno. V korpusu se zrcali vplivnost tekstov, inovativnost avtorjev, pomembnost itd. z upoštevanjem selekcijskih kriterijev. V tem smislu je korpus deljen na dva enaka dela s po 15 milijoni besed. Prva polovica upošteva predizbrana besedila (selektivna polovica), druga polovica pa tekste, identificirane naključno iz seznamov objavljenih materialov (mikrokozmična polovica). Za majhno članarino je celoten korpus razpoložljiv v ASCII obliki za akademsko skupnost.

Z različnimi zahtevami je mogoče iz celotnega korpusa oblikovati delne korpuse, npr. po razdobjih, v določeni regiji, za področje imaginativnega itd. Korpus je razdeljen na dokumente z naslovi in besedili. V naslovnem delu se nahajajo bibliografski podatki, kot so avtor, datum, klasifikacijska informacija (regija, vsebina, čas, raven, medij in spol). Besedilni del je izpisan z ASCII znaki (1-127) in z razširjenimi ASCII znaki (naglasi, denarni simboli in nelatinični znaki).

Longmanov semantični kodirni sistem ima tri dele: izbira omejitev (omejitev na subjekte

glagolov in pridevnikov in na prve ali druge objekte glagolov), semantične značilnosti (stopnja, funkcija, način, intencija itd.) in sociolingvistične značilnosti (register, intencija, uporabniška skupina, itd.). Leksikograf analizira vsako definicijo s posebnim menijem (računalniška podpora) in izbere značilnosti z uporabo posebnega okenskega menija.

LDOCE definicijski slovar deluje v Prologu in Lispu in je primeren za posebne raziskave. Leksikografe zanima zlasti vedenje besed, gramatični vzorci, kolokacije (kontekst) in idiomi skozi avtentično evidenco govorice. Longman razvija tudi govorjeno komponento kot del korpusa (konkordance v ortografiji, neprosodična notacija). Longman Pronunciation Dictionary (J. Wells) s 75000 besedami britanske in ameriške izgovarjave je izšel v letu 1990.

Longmanov korpus in dostopi v bazo so na razpolago raziskovalnim organizacijam, ki morajo imeti status popolne nekomercialnosti (ne smejo ustvarjati dohodka z nobeno komercialno dejavnostjo). Večina univerzitetnih ustanov ne izpolnjuje tega pogoja, saj svoje zmogljivosti plasirajo tudi komercialno, tj. izven okvira neprofitnega državnega financiranja.

Anton P. Železnikar

Projekt japonsko-nemškega slovarja

Japonski in nemški učitelji na področju nemškega jezika, japonskega jezika, lingvistike in informacijske tehnologije so razvili koncept novega japonsko-nemškega slovarja, ki naj bi poenotil različne funkcije dvojezikovnega slovarja v obliki podatkovne baze in omogočil dostop k razumljivejši in raznovrstnejši vsebini.

Ta slovar upošteva splošno sprejete, vendar večkrat negirane razlike dvojezikovnih slovarjev glede na izvirni jezik (izvirnega uporabnika). Te razlike se nujno pokažejo v primerih, ko japonsko-nemški slovar uporabljajo Nemci pri študiju japonščine ali Japonci pri študiju nemščine. Zato morajo izhodni formati pri vhodnih podatkih dovolj izčrpno upoštevati tako japonsko kot nemško leksiko. Le tako lahko slovar preseže študijske in didaktične kakovosti navadnih dvojezikovnih slovarjev v obliki unificirane, relativno pomensko zgoščene in leksikalno kontrastne deskripcije za oba jezika, vendar v *eni* smeri (iz japonščine v nemščino).

Slovar vsebuje podrobno fonetično, gramatično, semantično in faktualno informacijo za vsako glavno besedo (headword), ki je bila še zlasti dopolnjena za japonskega uporabnika slovarja. Elektronsko procesiranje podatkov omogoča zapis velike količine podatkov na omejenem prostoru in tako so bili lahko vključeni v slovar tudi sistematično številni primeri besedil. Skozi to orientacijo je postal slovar tudi dragocen instrument za raziskave gramatike, semantike in stila za nemške in japonske uporabnike.

Druga specifična lastnost slovarja je integracija leksikografsko distinktnih komponent, kot so npr. splošni, tehnični in kulturni vokabular v okviru enega obsežnega slovarja. V okviru splošnega japonskega vokabularja je narejena distnikcija med avtohtonim in sino-japonskim vokabularjem (yamatokotoba in kango), ki morata biti obravnavana leksikalno ločeno. Z upoštevanjem vseh teh komponent je bila ekspertno določena dvojezikovna baza podatkov, ki omogoča javni dostop prek primernih medijev (MT, FD, CD-ROM itd.).

Projekt podpirajo Japan Society for the Promotion of Science, the Institute of Cultural and Linguistic Studies of Keio University, the Scientific and Technical Faculty of Keio University, the National Research Institute for the Japanese Language, The Japan Information Center of Science and Technology (JICST), Toshiba, DEC Japan, NEC korporacije, Toyota Foundation in Kajima Foundation. Projekt vodi dr. Kennosuke Ezawa na univerzi v Tübingenu (Nemčija) v okviru Lingvističnega oddelka za nemški seminar. Na projektu sodelujejo še univerze in inštituti v različnih krajih, kot so Tübingen, Bonn, Hamburg, Niiza, Tsukuba, Dokkyo, Kyoto, Tokyo in Yokohama. Vse to kaže, kako kompleksno so danes koncipirani za nas »enostavni« projekti.

Anton P. Železnikar

Konzorcij za leksikalne raziskave

Računalniška lingvistika je dosegla točko, kjer so zmogljivosti sistemov za procesiranje naravnih jezikov omejene s t.i. leksikalnim zadržkom. Ti sistemi lahko obdelajo veliko več besedila in producirajo pomembne aplikativne rezultate z ugotovitvijo, da so njihovi leksikoni premajhni.

Association for Computational Linguistics v ZDA je predlagala ustanovitev konzorcija (Consortium for Lexical Research, kratko CRL), ki ga bo financirala DARPA. Sedež tega konzorcija je v Computing Research Laboratory (Rio Grande Research Corridor, New Mexico State University). Konzorcij je organizacija za razdeljevanje podatkov in orodij pri raziskavah slovarjev in leksikonov naravnih jezikov in za komunikacijo raziskovalnih rezultatov.

CLR ne sme kreirati t.i. komercialnih produktov in služi le predkonkurenčnim raziskavam, katerih cilj je konstrukcija računalniških leksikonov. CRL sprejema prispevke neglede na izvor, teoretično usmeritev in jih distribuira. Računalniška oprema CRL obsega primerne stroje za napredno obdelavo podatkov, kot so: Connection Machine, Intel Hypercube, Sequent Symmetry, IBM-ACE in še vrsto navadnih strojev. Omogočen bo tudi dostop do strojev z velikimi pomnilniki.

Anton P. Železnikar

Genelex: Eureka za lingvistično inženirstvo

Računalniška lingvistika že vrsto let ni več samo akademska aktivnost, postaja čedalje bolj tudi industrijska. Na sestanku evropskih ministrov junija 1990 v Rimu je bil tako odobren tudi projekt Genelex v okviru Eureke kot največji projekt leta. Projekt bo živel štiri leta s proračunom 250 milijonov frankov in se bo izvajal v Franciji, Italiji in Španiji. Sodelujejo pa: v Franciji Bull, Gsi-Erli, Hachette, IBM, LADL, Sema-Group; v Italiji Lexicon, Research consortium of Pisa, Servedi (Utet and Paravia); v Španiji Salvat, Tecsidel, University of Barcelona.

Jezik postaja glavni informacijski vektor in mnoge aplikacije potrebujejo kompleksno informacijsko procesiranje v pisni obliki. Vrsta projektov je bila realiziranih v okviru ERLI (Eureka for Linguistic Engineering). *Avtomatično indeksiranje* je program za analizo dokumentov, ki pridružuje celotni dokumentni bazi primerne koncepte za uporabo v raziskovalni fazi. S tem orodjem je mogoče tudi avtomatično analizirati vprašanja v naravnem jeziku in prek konceptov, s katerimi je bil dokument indeksiran, poiskati odgovore (informacijski retrieval). *Telematični vmesniki* so bili razviti v Franciji in zahtevajo naravni dialog (brez uporabe računalniških jezikov) med javnimi uporabniki in različnimi storitvami v okviru francoske mreže Minitel. *Avtomatično prevajanje ali računalniško podprto prevajanje* (Computer Assisted Translation, kratko CAT) je program za analizo stavkov v danem jeziku, ki zgradi bolj ali naj abstraktno predstavitev stavkov in generira namenske stavke iz te predstavitve. Možno je tudi *vpraševanje* v relacijskih podatkovnih bazah v naravnem jeziku, avtomatično generiranje korespondence itd. Ta možnost je tako sestavni faktor kompleksa računalniki in jezik v moderni družbi.

V primerjavi z ekspertnimi sistemi je ostalo področje procesiranja naravnih jezikov na margini naprednih medijev. Enostavneje je bilo namreč razumevati generična komercialna orodja (ekspertni generatorji) in se tako izogibati realnim problemom (oblikovanje pravil), ki se začenjajo pri uporabnikih. Področje naravnih jezikov je še vedno v svoji jecljavi fazi in se najraje ukvarja z aplikacijskim razvojem za posamezne kliente in potrebe ter se ne spušča v tvegano investiranje svojih razvojnih produktov (izjema so seveda Japonci!). Trg ekspertnih sistemov se je oblikoval vnaprej, trg naravnih jezikov pa oblikujejo zahteve. V Gsi-Erli je 90% naročenega razvoja namenjenega naročniškim aplikacijam in ne produktnemu razvoju.

Izkušnje na področju aplikacij z naravnim jezikom pa so povzročile nastanek generičnih razvojnih orodij s splošnim pomenom, ki jih lahko razvrstimo v tri skupine. *Avtomati* so analizatorji, ki omogočajo računalniško konstrukcijo predstavitve stavčnih pomenov. Ti generatorji

lahko zgradijo stavek iz semantične predstavitve. Analizatorji uporabljajo dve vrsti podatkovnih baz, in sicer gramatično in slovarsko. *Gramatika* pomaga analizatorju pri manipulaciji s sintakso stavka in deluje v obliki pravil. Za vsako aplikacijo obsega določen korpus, ki je reprezentativni vzorec za množico stavkov, ki se procesirajo. Gramatika opisuje lingvistične fenomene, ki so zastopani v korpusu: vprašalni, relacijski, koordinacijski fenomeni itd. Navadno se v tej zvezi razvijajo zelo različne gramatike. *Slovar* dovoljuje analizatorju, da razpolaga z vso povezano informacijo za neko besedo: morfološko informacijo (del govorice, pravila sestavljanja z drugimi besedami), semantično informacijo (opisi pomenov besede) in večkrat še s pragmatično aplikativno informacijo.

Cilj projekta Genelex (Generic Lexicon) je konstrukcija generičnega slovarja za različne evropske jezike (zaenkrat francoskega, italijanskega in španskega), tako da ne bo potrebno konstruirati novega slovarja za vsako novo aplikacijo. S tem načinom naj bi bila zagotovljena tudi operacijska kakovost slovarja in nizka cena pri več kot 40000 besedah. Ta sistem bo omogočal tudi poljubne modifikacije, izdelavo posebnih slovarjev, skratka razvojno fleksibilnost in ekspertizo. Cilj projekta ni le generični slovar, temveč tudi strategija, ki bo zagotavljala, da bo slovar ostal generičen. Osnova tega slovarja bo velika podatkovna baza.

Projekt Genelex sestavljajo tri naloge. **1.** Definira se podatkovni model za predstavitev strukture besede z možnostjo naložitve te predstavitve v podatkovno bazo. Informacijska množica (morfološka, sintaksna in semantična), ki se lahko poveže z besedo, predstavlja tudi relacijo odvisnosti. Ta model bo postal standard za leksikalno predstavitev. **2.** Možna je konverzija iz drugih modelov slovarjev v model Genelex in na razpolago so še druge funkcije: pomoč leksikografu pri mešanju (zlivanju) slovarjev, analiza besedil pri vstavljanju v model (morfologija, sintaksa), leksikografska delovna postaja (administracija leksikalne podatkovne baze) in generatorski programi (generiranje podleksikonov). **3.** Učinkovito konstruiranje generičnega slovarja.

Anton P. Železnikar

Oblikovanje leksikalnih modelov pri IBM

IBM gleda na možnosti računalniške leksikografije/leksikologije predvsem s stališča predstavitve in shranjevanja leksikalnih podatkov. Tako se problemsko osredotoča na postopke ekstrakcije leksikografskih podatkov v okviru procesiranja strojno berljivih besedil v naravnem jeziku. To je seveda le prvi in pri IBM vselej poslovno zakriti vtis. Prav verjetno pa se IBM s tem ne odpoveduje implementaciji on-line slovarjev za človekovo rabo in implementaciji kompilacijskega programskega sistema za vključevanje podatkov v prihodnje človekove slovarje, ki jih bo tržil. Pri vsem tem je izrecno poudarjeno pridobivanje (akvizicija) leksikalne informacije in oblikovanje (struktura in organizacija) podatkovnih modelov za generične naloge, kot so identifikacija, odkrivanje, ekstrakcija in reprezentacija leksikalnih lastnosti besed na osnovi raziskovanja velikih slovarjev, ki so na razpolago v elektronski obliki. Takšen slovar bi lahko bil npr. Longmanov elektronski slovar (ali bolje baza), s 30 milijoni angleških besed.

Vse več projektov s področja računalniške leksikografije in leksikologije uporablja v začetni fazi strojno berljive oblike objavljenih slovarjev. Posamezni cilji pri tem bistveno variirajo: vključujejo npr. lingvistično analizo slovarskih definicij, statistično osnovano iskanje regularnih vzorcev v slovarskih virih ali polavtomatično izpeljavo leksikonov iz obstoječih slovarjev. Skupna značilnost vseh takih projektov je potreba po določenih okvirih, v katerih se iščejo preslikave iz izvirnih oblik v leksikalne podatkovne baze.

Opazne so predvsem tri oblike opisanih okvirov, in sicer prenos (transdukcija), predstavitev (reprezentacija) in vpraševanje. Zunaj specifičnosti katerega koli pristopa je uporaba strojno berljivega slovarja za ekstrakcijo fragmentov leksikalnih enot. Dostop do leksikalne vsebine on-line slovarja je vselej zagotovljen z uporabo vpraševalnega mehanizma; izrazna moč vpraševalnega jezika določa meje in podrobnosti leksikalnih lastnosti, ki bodo izločljive iz leksikal-

nega vira. Podrobnosti so seveda odvisne tudi od predstavitvene sheme, ki jo npr. uporablja konkretni model on-line slovarja. Narava slovarske predstavitve bistveno omejuje vrsto opazovanja, ki zadeva implicitno kodirano informacijo slovarja in lingvistične posplošitve, ki se zrcalijo v slovarju. Neodvisno od predstavitve pa je potrebna računalniška podpora za prenos surovega tračnega formata v množico slovarskih zapisov in polj.

Nekatere zahteve pa so ortogonalne: mehanizem za analizo slovarskega vira je lahko neodvisen od načina predstavitve slovarskih podatkov. IBM razvija npr. splošno orodje za analizo slovarskih enot; v pripravi je tudi predlog za splošno in aplikacijsko neodvisno predstavitveno shemo za on-line slovarje. Kljub temu pa še ni soglasja, kakšen naj bi bil splošen računalniški model slovarja, še posebej z vidika, kakšne vrste procesiranja naj bi model podpiral.

Pridobivanje (akvizicija) leksikalnih podatkov zadeva tako njihovo analizo, dostop in metodologijo. V okviru tega je posebno vprašanje namenjeno formatom leksikalnih podatkov v okviru konvencionalne tehnologije podatkovnih baz in načrtovanju slovarskih podatkovnih baz.

Možna je identifikacija štirih razredov slovarskih modelov. Te je mogoče razdeliti na relacijske in hierarhične formate, s poddelitivijo na fizične, logične in leksikalno koncipirane hierarhije, ki se pojavljajo kot organizirni principi predstavitve on-line slovarja.

Skladno s priznanim principom tehnologije podatkovnih baz preslikava relacijski model slovarja neko slovarsko enoto na množico tabel. Tu ne obstaja kanonična splošna preslikava med leksikalnimi atributi in lastnostmi, ki so tipične za slovar in entitetami in relacijami, ki jih modelira relacijska podatkovna baza. Kljub temu pa postopen način načrtovanja podatkovne sheme omogoča vzdrževanje vidne slovarske informacije v podatkovni enoti. Primer take podatkovne sheme je npr. on-line model za Longman Dictionary of Contemporary English. Za ekstrakcijo leksikalnih relacij, kot so is_a, part_of, group_of, degree itd., sta Nakamura in Nagao predvidela posebna polja v enotah tega slovarja. Longmanov izvirni trak vsebuje določene označevalce, ki jih v knjižni obliki slovarja ne najdemo.

Anton P. Železnikar

Kitajski elektronski slovar

Kitajci (Dept. of Computer Science, Tsinghua University) razvijajo elektronski slovar z bogato sintaksno in semantično informacijo za potrebe kitajskega deterministčnega analizatorja, ki so ga razvili na univerzi Cingua. Format tega slovarja je prilagojen analizatorju in upošteva strukturne dvoumnosti kitajščine.

Znanost o analizi kitajščine je v zaostanku. Kitajščina je predvsem analitični jezik, ki mu manjkajo formalni označevalci, kar otežuje analizo kitajskega jezika. Kitajci še ne razpolagajo z dovolj robustnim analizatorjem in zaradi tega zaostajajo tudi s projekti strojnega prevajanja kitajščine v tuje jezike. Analizator je programiran v jeziku Common Lisp in se izvaja na delovni postaji Sun4/280. Analizator lahko analizira posamezne stavke kitajskega jezika v času, ki je manjši od stotinke sekunde.

em analizatorja je postalo očitno, da je potrebno razviti tudi elektronski slovar z bogatim leksikalnim znanjem. Gre torej za slovar, ki je strukturno prilagojen potrebam deterministčnega analizatorja.

Vsak vstop kitajskega elektronskega slovarja ima tole strukturo: sintaksno-kategorialna, semantična, primerna (selektivna restrikcija različnih primerov), klasifikacijska, distinktivna (pravila razločevanja), ponovljiva (kategorialna dvoumnost) in shematska informacija. Že ta informacijska struktura odraža posebnosti kitajskega jezika, tj. kitajskih »besed«. Kaj bistveno več pa na tem mestu brez znanja kitajščine tudi ni mogoče povedati.

Anton P. Železnikar

Slovenski slovnični in besedni pregledovalnik

Slovnični pregledovalnik slovenskih besedil (BesAna) je program za besedno in skladenjsko analizo, odkrivanje napak v besedilih in slovnični

pregledovalnik. Program ima bazo s 15000 koreni, tj. okoli 250000 besed. Besede v bazi so zapisane s korenom in vzorcem za njihovo izpeljevanje; tak pristop nudi majhno porabo pomnilnika in hitro iskanje besed. V bazo je mogoče dodajati nove besede. Posebej so zbrane besede, ki se najpogosteje napačno uporabljajo. Je pa še vrsta drugih opozorilnih funkcij: npr. velike črke sredi besede, lastna imena z veliko začetnico, pisanje krajšav, manjkajoče vejice, kontrola pravilnosti rimskih številk, raba predlogov s, z, iz, k, h, v in na, preverjanje sklanjatve ob predlogih, ujemanje pridevnika in samostalnika, pomožnega glagola in deležnika na -l, označevanje napak v besedilu, ki se kasneje popravijo z urejevalnikom itd. Posamezne dele besedila (npr. v tujem jeziku) je mogoče izključiti iz analize, opravlja se statistika uporabljenih besednih vrst in pregledujejo se besedila vseh znanih besedilnih urejevalnikov (od ASCII formata do WordStara).

Čeprav je BesAna čisti komercialni proizvod poslovnega konglomerata Graf, se v okviru te dejavnosti oblikuje tudi združen nacionalni in predvsem mednarodni projekt, tako da bo oblikovanje slovenskega elektronskega slovarja omogočeno z uporabo mednarodnih slovarskih standardov. Ti standardi se bodo vzdrževali s posebnim programskim orodjem za sistematično zajemanje slovarskih entitet (besed, slovničnih pravil, besednih kontekstov, besednih pomenskih konceptov, prevajalnih relacij glede na tuje jezike itd.). Pogajanja s tujimi raziskovalnimi in komercialnimi ustanovami po svetu so v teku.

Proizvajalec in distributor programa BesAna (Besedna Analiza) je Graf Inženiring, Letališka 33, Ljubljana (061/448 241). Cena je v območju 500 DM.

Drugi proizvod je Mspell (Miha Mazzini), ki je program za odkrivanje tipkarskih, slovničnih in stilističnih napak v besedilih. Vsebuje 200000 izvedenih slovenskih besed, preverja uporabo velike začetnice po ločilih, odkriva in popravlja uporabo predlogov s, z, k, h in v, odstranjuje odvečne presledke pred ločili in dodaja manjkajoče za ločili, predlaga vejice, opozarja na zaporedno začenjanje stavkov z eno in isto besedo, preverja rimske številke, vodi evidenco o pojavnosti besed, opozarja na velike črke sredi besede itn. Okvirna cena je 300 DM.

Anton P. Železnikar

Japonska pobuda za sodelovanje pri razvoju in prodaji elektronskih slovarjev

Razvoj elektronskih slovarjev zahteva harmonično sodelovanje različnih skupin, njihovo vodenje in koordinacijo. Japan Electronic Dictionary Research Institute (kratko EDR) razvija v sodelovanju z drugimi skupinami in instituti elektronski slovar.

Področji računalniških znanosti (procesiranje naravnega jezika) in lingvistike z leksikologijo sodelujeta tesno pri razvoju slovarja. V EDR projektu imajo računalniške znanosti pomembnejšo vlogo. Tudi obsežen elektronski slovar ni kaj drugega kot kompleksen sistem in prav računalniško osebje ima potrebno izkustvo pri načrtovanju, razvoju, čiščenju, vzdrževanju in obvladovanju velikih sistemov.

Glavni viri pri razvoju elektronskih slovarjev so seveda obstoječi slovarji v knjižni obliki, vendar se ti slovarji bistveno razlikujejo od elektronskih. Prav zaradi tega se elektronski slovarji načrtujejo in oblikujejo drugače že od samega začetka. EDR ima svoje slovarske podatke, ki so zelo grobi in so zbrani iz različnih slovarjev japonskih založnikov. Za zbiranje tega materiala zunaj EDR je bila potrebna kar precejšnja količina tehnologije in opreme. Pri tem je bilo potrebno upoštevati tudi avtorske pravice in pravno urediti ta vprašanja.

Pomembno je omeniti, da elektronski slovarji pomagajo človeku uporabljati slovarje. Tu gre za velike volumne zbranih besedilnih podatkov in za metodologijo njihove uporabe. Tako se razvija novo področje interakcije med človekom in strojem na prodročju oblikovanja in uporabe slovarjev.

Elektronski slovar je naprava za dejansko in učinkovito uporabo. Zaradi tega je primarni cilj projekta uporaba, ki je pred samo kreacijo slovarja. Slovarji naj bi odražali poglede in zamisli uporabnikov na eni strani in izkustvo načrtovalcev, ki so se že doslej ukvarjali s procesiranjem

naravnih jezikov in procesiranjem znanja, na drugi strani. Pri EDR gre za distribuiran laboratorijski sistem z osmimi raziskovalnimi laboratoriji, ki so razporejeni v posameznih investicijskih podjetjih. Vsak laboratorij je tesno povezan s skupinami svojega podjetja, ko razvija svoj del velikega aplikativnega sistema za procesiranje naravnega jezika, vključno s strojnim prevajanjem.

Pri projektu pomagajo tudi zunanje institucije, ki bodo predvidoma postale največji uporabniki elektronskih slovarjev. Trenutno sodeluje EDR z Electronic Laboratory (ETL), Institute for New Generation Computer Technology (ICOT), Center of the International Cooperation for Computerization (CICC) in s petimi univerzami na Japonskem (univerza v Kyotu je glavna med njimi).

Kakšno pa je mednarodno sodelovanje na projektu? Dokončni cilj projekta je razvoj in priprava elektronskih slovarjev, ki bodo lahko uporabljeni v številnih jezikih in v vseh svetovnih okoljih. Ta cilj se dosega s stalnimi in zadostnimi napori. Prevladuje prepričanje, da lahko elektronski slovar za posamezen jezik pripravijo samo skupine ljudi, ki jim je ta jezik materin jezik. Pri izdelavi dvojezikovnih slovarjev sodelujejo strokovnjaki s področja obeh jezikov. Pri EDR projektu (v sedanji fazi) sta bili izbrani japonščina in angleščina kot namenska jezika s podarkom na japonščini. Angleščina je bila pridružena kot namenski jezik zaradi svoje pomembnosti, saj prevzema vlogo skupnega mednarodnega jezika. Z vključitvijo angleščine bo mogoče tudi preveriti univerzalnost specifikacij v EDR slovarjih. Zaradi nujnosti sodelovanja z raziskovalnimi skupinami v različnih jezikih je EDR zaupal nekatere svoje naloge univerzi v Manchestru (University of Manchester Institute of Science and Technology). To sodelovanje se bo razširilo še na druge anglosaksonske univerze.

Japonci so že zdavnaj ugotovoli, da je angleščina skupen mednarodni jezik. Vendar narašča tudi zanimanje za druge jezike, ki naj bi jih tudi bolj upoštevali. V poštev prihajata zlasti francoščina in nemščina. Vendar je predvideno tudi tesno sodelovanje z nekaterimi azijskimi narodi, zlasti s skupinami v Kitajski, Tajski, Maleziji in Indoneziji skozi projekte v okviru CICC. Te skupine že razvijajo svoje lastne elektronske slovarje z uporabo istih specifikacij kot v EDR. Odprte so tudi vse možnosti za druge narode (tudi za slovenščino), če bi se sami lahko odločili za tovrsten raziskovalni podvig. Iz tega bi prav gotovo lahko nastali odlični elektronski slovarji za Slovence, slovenska raziskovalna sfera pa bi dobila relevantno nalogo ne le v okviru slovenske samobitnosti temveč predvsem v povečevanju možnosti za kakršnokoli preživetje slovenskega jezika.

Pri sodelovanju s tujimi skupinami poudarja EDR pomembnost dogovorjenih procedur, ki so koristne za obe strani. To sodelovanje naj bi se uravnavalo v naslednjih korakih:

Korak 1. Izmenjava informacije o tehnologijah. Specifikacije EDR in druga informacija v procesu raziskav in razvoja je praktično na razpolago z vsemi podrobnostmi. EDR že prireja posebne delovne sestanke za udeležence z vsega sveta. Obstaja slovarska povezava za interesente, ki bi želeli bolj podrobno informacijo o vsebini elektronskega slovarja. Slovarsko povezavo sestavljajo množice glavnih konceptov in glavnih besed, ki kažejo trenutno stanje elektronskega slovarja. Ta povezava se spreminja z napredovanjem projekta in deluje od začetka tega leta. Pristojbina za povezavo obsega le distribucijske stroške.

Korak 2. Tehnične obveznosti. Nadaljnje dogovarjanje je potrebno s skupinami, ki nameravajo razvijati elektronske slovarje po specifikacijah, ki so enake ali podobne oním iz EDR. EDR je pripravil podrobnosti in programsko opremo (objektni kod) za različne sisteme, na katerih bi se izvajal razvoj elektronskega slovarja.

Korak 3. Medsebojna izmenjava elektronskih slovarjev. Ta korak naj bi bil končna stopnja sodelovanja. Možno naj bi bilo izmenjevanje elektronskih slovarjev, če obstaja soglasje obeh partnerjev. Z izmenjavo slovarjev naj bi počasi in zanesljivo nastal svet različnih elektronskih slovarjev. V tej fazi je najpomembnejši princip, da se vsak jezik obravnava enako, na enaki osnovi.

Načelno se vsi rezultati EDR projekta

prodajajo za primerno ceno. Pod enakimi pogoji bo tekla tudi prodaja elektronskih slovarjev neglede na izvor, lokacijo uporabnika (enaki pogoji za domače in tuje uporabnike). Pričakovane cene bodo nižje od drugih elektronskih slovarjev na trgu. Posebni popusti bodo veljali za akademske, univerzitetne in javne raziskovalne institucije. Natančen datum začetne prodaje je tale: Japanese Word Dictionary (april 1991); drugi slovarji bodo sledili kolikor mogoče kmalu.

Opisane principe zastopa EDR. Pri sodelovanju z drugimi skupinami bo EDR postopal kar se da prožno in prilagodljivo. Pri tem naj bi se nikoli ne ustavilo izboljševanje in razširjanje slovarjev. Prav izboljšave in razširitve slovarjev bodo zahtevale svetovno koordinirano vzdrževanje. EDR je že pristopil k vzpostavljanju **ustrezne organizacije,** ki bo širila in vzdrževala trajno sodelovanje med deželami tega sveta.

Anton P. Železnikar

Knjižne recenzije

Recenzija knjige: Anton P. Železnikar, On the Way to Information, Part 1, Ljubljana

Nastanak ove knjige najbolje će razumjeti čitaoci, koji poznaju autora, Antona Železnikara. Železnikar je u svojoj karijeri prošao kroz sve sektore digitalne elektronike, kompjutora i informatike. Već kao mladi inženjer projektirao je digitalnu memoriju za spremanje podataka iz nuklearnih strojeva. Tada su Instituti Jožef Stefan, Rudjer Bošković i Boris Kidrič bili u samom svjetskom vrhu u digitalnoj tehnici. Tako na primjer, instituti u mnogim zapadno evropskim zemljama u to vrijeme nisu još razvili digitalnu memoriju, kakvu je Železnikar razvio u Jožef Stefanu.

Od digitalne logike Železnikar je krenuo u svijet programiranja, programskih jezika, dizajna i upotrebe kompjutora, matematičkih i teoretskih radova, pa sve do ove knjige. Probao je život sveučilišnog profesora, istraživača i dizajnera i voditelja razvojne grupe u industriji. Njegovo veliko i široko iskustvo omogućilo mu je da stvori filozofski stav prema kompjutorima i informacijama i da ga uobliči u ovoj knjizi.

Kompjuteri služe da skupljaju, preradjuju, filtriraju, modificiraju, transformiraju, i t.d., informacije. Svrha: dati korisniku pravu i traženu informaciju.

O obradi informacija postoje razradjene teorije i postupci na nižem i na višem nivoju, od interesa za kompjuterske profesionalce. Železnikar želi svoju knjigu postaviti iznad ovih teorija i iznad kompjuterske profesije i približiti ju filozofiji, pogledu na svijet. Gotovo sve filozofske studije pisane su od ljudi koji ne poznaju niti približno, a kamoli detaljno, svijet kompjutera i informatike i utjecaj na život i rad ljudi. Železnikar predvidja da će filozofi, sociolozi, žurnalisti, kompjuteraši, materijalisti i idealisti oponirati proširenju i

predefiniranju pojava informacije. Ova knjiga upravo to čini: proširuje i predefinira pojam informacije. Železnikar nije želeo upasti u postojeće kolotečine, želeo je ići smjerom koji nije uobičajen. Univerzalna informacijska teorija i filozofija koju opisuje ova knjiga predstavljaju originalne poglede Železnikara, a ne kompilaciju do sada poznatog i prihvaćenog. To se dobro vidi u dijelu knjige koji obradjuje temu: Bog kao informacija. Železnikar vjeruje da univerzalno poimanje informacija može pozitivno utjecati na buduci razvoj informacijskih teorija u raznim znanostima, na konstrukciju intelegentih strojeva i programa i na bolje razumjevanje života i živih biča, mozga, uma i vjere.

Knjiga je pisana na engleskom jeziku i na nekim mjestima se osjeća da to nije materinji jezik autora.

Knjiga je originalna i traži prilično truda i odredjenu naklonost prema filozofiji da bi se čitala. Vjerovatno će izazvati kontradiktorne sudove. Po svoj prilici Železnikar to upravo i želi: stvoriti nove valove u Moru Informacija.

Zagreb, 12.11.1990 *Prof. Branko Souček*

Prirodoslovno-matematički fakultet
Sveučilišta u Zagrebu

On the Way to Information (Na poti k informaciji)

Knjiga »Na poti k informaciji« predstavlja sintezo več znanj s področja filozofije, tehnike, jezikoslovja in tudi — če pomislimo na njeno nadaljevanje (tovrstni članki so bili že objavljeni v časopisu Informatica) — logike in matematike. Avtor se loteva problematike na sebi svojstven način, ki mu vsaj v naši deželi ni para. Z zelo široko in poglobljeno definicijo pojma informacije (njemu sorodnih pojmov) skuša zajeti glavne paradigme filozofije in tudi tehniških in naravoslovnih ved. Seveda se mu to pokaže skozi analizo pojma subjekta in objekta. Pri tem se znajde pred svojevrstno dilemo: ali opisovati subjekt kot vzročno-posledičen splet, ali drugače rečeno, kot zaporedje možganskih stanj ali pa kot tok predstav ali odnosov med idejami; skratka, namesto stanja možganov imamo stanja duha. S svojim konceptom informacije, ki je tudi nekaj, kar samo sebe izpeljuje, se avtor hoče rešiti te dileme pri opisu subjekta. Na ta način lahko združi opise naravoslovnih ved, kot so fiziologija (možgani) in sploh biologija (v stilu kakega Maturane) in psihologije, s fenomenološkimi opisi zavesti kot toka doživljajev. Če gleda na organizem zgolj kot na nosilca informacijskih procesov in jih razlaga s pomočjo znanstvenega aparata biologije (fiziologije), potem mu ta razlaga lahko predstavlja podlago za razlago organizma kot subjekta, ki sam sebe izgrajuje (autopoiesis), kar pa lahko razumemo tudi čisto v filozofskem smislu subjekta, ki skozi svoj razvoj (zgodovino) konstituira samega sebe predvsem tako, da ima sam določeno predstavo ali razmerje samega sebe (informacija informacije).

Če se bo avtorju posrečilo še na dovolj enostaven, razumljiv in zanimiv način formalizirati (kar je delno že storil v prej omenjenih člankih v Informatici) sklop pojmov, ki vsi izhajajo iz pojma informacije, kar bo vsebina nadaljevanja te knjige, potem bo imel bralec pred sabo močan izziv za razmišljanje o odnosu mišljenja in sveta in za tehniško zapopadenje tega odnosa.

V Ljubljani, 29.1.1991 *Doc. dr. Valter Motaln*

Fakulteta za strojništvo
Univerze v Ljubljani

Vprašanja,

ki jih je postavil *dr. V. Motaln* avtorju na tiskovni konferenci v Mladinski knjigi (Titova 3, Ljubljana), dne 29.1.1991:

1. *Kako lahko vpliva po vaše razdelava teorije informacije na razvoj metodologije znanosti?*

Odgovor: Razširjena teorija informacije lahko bistveno prispeva k razvoju metodologij znanosti na več načinov. Prvi vidik je sprememba jezika (znanstvene govorice neke discipline), v katerem t.i. znanstvene formule (fiksni logični izrazi, principi znanosti, njene formalizirane teorije) preidejo v t.i. formalne scenarije, tj. v razvojno odprte formule in teorije, ki so podvržene nastajalnim, konstruktivnim možnostim znanstvene discipline.

Ta vidik se veže na vpeljavo informacijske logike, ki določeni znanstveni disciplini omogoča vpeljavo njenih specifičnih pravil za razvoj znanostnih, raziskovalnih in konstrukcijskih formul. Ta logika prinaša v mišljenje znanosti metodološki obrat, ki poudarja nastajanje znanstvene discipline same in odvisnost od informacijskega nastajanja v drugih znanostih, znanstvenih komunah in okoljih. Hkrati pa informacijski pristop, kot je zasnovan v knjigi, prinaša v zavest raziskovalca fenomenološki vidik, ki ima lahko bistveni vpliv na nastajanje nove metodologije.

2. *Kaj lahko pove teorija informacije o naravi jezika? Ali ga razume predvsem kot skupek pravil za rojevanje (generiranje) novih izrazov (izrazov nasploh) ali tudi kot skupek pravil, ki rojeva tudi samega sebe? Kakšen je odnos jezika in bitja, ki ta jezik uporablja? Ali je jezik v resnici »hiša biti«, kot slikovito pravi Heidegger?*

Odgovor: Jezik je le ena izmed specifičnih jezikovnih informacijskih pojavnosti; seveda če ga ne motrimo zgolj s stališča tradicionalnega jezikoslovja.

Teorija informacije se lahko marsičesa nauči prav pri nastajanju (govorjenju, pisanju, mišljenju) jezika, tj. njegovih stavkov in stvačnih kompleksov. Naravni jezik je najprej sam sebi jezik, sam sebi opisni jezik oziroma metajezik ali jezik jezika. Naravni jezik, njegovi označevalci in njim pripadajoči (nastajajoči) pomeni, razumevanje jezika kot spontano informacijsko gibanje po prepletenih semiotičnih mrežah in oblikovanje pomena nastajajo tedaj v prostranstvu jezika samega, razvijajočega, nastajajočega jezika. Jezik kot informacija rojeva sam sebe v jezikovnih diskurzih, spontano upoštevajoč določena pravila in njihovo nastajanje v domeni jezikovnega informiranja. Razširjena teorija informacije seveda ne gleda na jezik kot skupek pravil, npr. v stilu N. Chomskega ali podobnih formalnih, značilno reduciranih, avtomatnih (generacijskih) teorij jezika. Tovrstne končne množice pravil za rojevanje neskončnih množic izrazov (npr. sintaksno pravilnih) so s stališča teoretske estetike seveda privlačne, vendar ne razrešujejo problemov pomena, vsebine, prevajanja, razumevanja, specifične uporabe in drugih bistvenih vidikov jezika.

Odnos jezika in bitja je odnos jezikovne informacije (lastne in druge) do metafizike kot totalne (vseobsegajoče) informacije bitja. Seveda je v metafiziki tudi jezikovna informacija že ponotranjena, stvar metafizike same, čeprav bistveno oblikovana iz jezikovnega informacijskega okolja. Materin naravni jezik je npr. sooblikovalec metafizike kot informacije (prepričanja, vere, znanja, govornega in motoričnega obnašanja, nazora itd.), prav za prav del metafizike same, njenega domovanja; tuj naravni jezik je spočetka le neke vrste metafizični problem bitja, prenašalec informacijske vsebine, nastale npr. v mišljenskem prostoru materinega jezika. Sčasoma pa se seveda lahko oblikujejo pristni (profesionalni, tehnični ali metafizični) odnosi tudi med tujimi ali celo umetelnimi jeziki in bitjem. Jezik kot informacija postane tako najprej »hiša bitjevske informacije«.

Kot »hiša oziroma dom biti« je jezik seveda »hiša informacije«. Zakaj prav hiša, to je prijazno okrovje in ne nekaj manj? Ker z jezikom, zlasti naravnim (materinim) jezikom bistveno, intenzivno in kar se da domače komuniciramo (informiramo) s samimi seboj in seveda z drugimi bitji. Ta jezik ne le tovori informacijo, temveč postane hkrati tudi njen konstituent, ne le kot možnost, temveč kot »sema« in »pragma«, kot pomen in govorna praksa in v tem okviru informacijska (govorna) moč, ki se lahko povzpne do poljubno zahtevnih višin, seveda le tistih, ki jih posameznik (kot avtopoezno bitje) še lahko smiselno in imaginativno doumeva (oblikuje, razumeva, zapopada, umešča v svoje umevanje stvari). Heidegger je namenil razumevanju kot eksistencialni konstituciji tu-biti doslej nedosegljivi filozofski spev (glej npr. v »Sein und Zeit«, paragrafa »31. Das Da-sein als Verstehen« in »32. Verstehen und Auslegung«), seveda le v okviru svojega projekta razumevanja biti. V »hiši informacije« z jezikom, bitjo, tubitjo, bitjem, bistvom, razumevanjem, stvarjo oziroma s katerim koli (mejnim) filozofskim pojmom postane vse to lahko le posebna oblika oziroma proces informacije — zgolj informacija — formirane in organizirane fenomenološke strukture.

3. *Ali odpira knjiga nove razsežnosti za razumevanje umetne inteligence?*

Odgovor: Prvotni (primordialni) impulz moje odločitve za »potovanje« po poti k informaciji je nastal iz osebne teorijske in inženirske (konstruktorske, tehnološke — v smislu nove tehnologije — in simbolno-teoretske) atitude. *Terry Winograd* (Stanford University), eden pionirjev prostranstva umetne inteligence, je leta 1986, skupaj s Fernandom Floresom, razprl do takrat nezaslišano dilemo o razumevanju (načrtovanju, rabi) računalnikov in spoznanja, na katero je ostro reagirala konsolidirana komuna umetnih inteligenčnikov (polemika v časopisu Artificial Intelligence) (hereza, odpadništvo, zgubljenost). Reakcija je bila podobna oni pred dobrimi dvajsetimi leti, ki jo je doživljal *H.L. Dreyfuss* (What Computers Can't Do) na MIT. Moje osnovno vprašanje takrat je bilo, kaj je informacija v živem in umetnem, katere so možne povezave, modeli, projekti, scenariji, novi formalni okviri in predvsem razumevanje.

Umetna inteligenca — z umetnim razumevanjem — je prostor človekovih inteligentnih orodij (pripomočkov, naprav, opreme), ki se preko razumevanja informacijske fenomenologije v živem približuje živi kompleksnosti, poziciji in atitudi informacije tudi v strojih, seveda, v kolikor ji uspeva spoznava in tehnološka realizacija naravnih informacijskih scenarijev. Posledica tega stališča je, da sta »stroj« (arhitektura stroja) in »program« (programski sistem stroja) informacijska, tj. informacijsko spremenljiva oziroma nastajalna. S tem se preseže tradicionalna algoritmika, ki je prevladujoči princip zlasti na področju programiranja računalnikov. Tudi informacija v stroju — tako kot v živem — »nastaja« z informiranjem, protiinformiranjem in umeščanjem informacije, ki so osnovni trije informacijski principi. Scenariji (arhitekture, programi) so informacijsko nastajajoči, odvisni od trenutne informacijske pozicije in atitude naprave v določenem informacijskem okolju.

Umetna inteligenca (inteligentni programi, ekspertni sistemi) seveda lahko uporablja nastalo, tudi tradicionalno (matematično, programsko) algoritmiko, vendar razpolaga ob tem še z vgrajenimi informacijskimi mehanizmi, ki omogočajo obdelavo izredno velikih količin mrežno prepletene informacije, katere »obdelava« s programi in stroji temelji na izkustvenih scenarijih in ne več le na togih (fiksno formuliranih) teorijsko-modelnih programskih upodobitvah informacijske stvarnosti. Osnovna intenca teorije informacije je, da se tudi v človekove stroje »vgradijo« nekateri (spoznani) principi informacije v živem. Ob tem pa se teorija informacije nikakor ne odpoveduje spoznavanju nastajanja informacije zlasti v človekovih korteksih, v okviru katerih človek spoznava, razumeva, transcendira svojo mentalno, intelektualno in vedenjsko informacijo. V okviru tega pogleda na stvar umetne inteligence odpira knjiga tudi nove razsežnosti za razumevanje umetne inteligence kot človekove in hkrati strojne informacijske pojavnosti.

4. *Ali je sploh možna formalizacija informacije, ki sama sebe izgrajuje?*

Odgovor: Formalizacija nekega področja je vselej predmet posebnega, lahko rečemo umetnega jezika (pomenskega prostora in razumevanja) discipline. Živi objekti so (iz perspektive subjektivnega opazovalca) samoreproduktivni, tj. avtopoezni oziroma sami sebe izgrajujoči. Npr. rekurzivne funkcije izgrajujejo svoje vrednosti iz predhodno zgrajenih vrednosti, od neke dane začetne vrednosti dalje. Tu so formalni začetki informacije, ki npr. od določenega mesta dalje izgrajuje svoje vrednosti izključno iz svoje prejšnje informacije s fiksnim pravilom, tj. z rekurzivno matematično formulo. Vendar je od tu dalje potreben skok v naslednjem premisleku: ali lahko ta funkcija izgrajuje tudi svojo formalno podobo, predpis, definicijo formule kot same sebe?

Odgovor na to vprašanje je pritrdilen, saj je mogoče napisati program (v strojnem jeziku), ki sam sebe in druge programe modificira (npr. programski virusi), torej jim nastajalno spreminja strukturo funkcije. Človek seveda ima izkustvo, kako je mogoče samega sebe izgrajevati, zlasti na informacijskem področju (učenje, asociativno

sklepanje, modusi zavestnega mišljenja, spontano gibanje po prepletenih nevronskih mrežah) ali kako oblikovati stroje, ki bodo imeli samoizgrajevalno funkcijo. Formalno se informacijska entiteta izgrajuje tako, da preprosto ima t.i. »operator nastajanja«, magični simbol informacijskega nastajanja, ki ga je mogoče modelirati celo z vrsto psevdodeterminističnih postopkov (npr. generiranje naključnih števil, znakov, simbolov, označevalcev informacijskih entitet, za katerimi se skrivajo informacijske procedure, tj. operacije itd.). Formalna teorija uvede preprosto operator informiranja, ki združuje v sebi aktualnost in potencialnost informacijskega učinkovanja (nastajanja, spreminjanja, preminjanja).

5. *Kako bi s pomočjo teorije o informaciji razložili koncept zavednega-nezavednega? Ali je mogoče na njeni podlagi razviti razlago ustvarjalnosti (ustvarjalnost kot produkt nezavednega, kot produkt primarnih impulzov, ki izhajajo iz same živalske narave človeka in, na drugi strani, ustvarjalnost kot produkt zavestnega ega, ki izhaja iz internalizacije družbenih odnosov)?*

Odgovor: Zavedno je lahko tisto, kar informacijsko vstopa v razumevanje kot informacijsko entiteto. Nezavedno lahko ostaja zunaj razumevanja. Razumevanje je skupaj s svojimi pomeni, ki jih producira, v razumevajoči cirkularni informacijski povezavi. To seveda ni ničesar drugega kot znani atributi filozofsko pojmovane zavesti in samozavesti (samoreferenca, refleksija, transcendenca itd.). Ustvarjalnost na področju informacijskega je v grobem informacijsko nastajanje (spajanje, pomensko prepletanje, konstruiranje novih pomenov kot informacije, naraščanje ali minevanje razumevanja kot aktivne entitete). Razločitev na nezavedno in zavedno je kot rečeno lahko bistvena le s posebnega informacijskega vidika, ki je razumevanje samo. Informacijski model sploh ne izključuje določenih fizioloških in psiholoških konceptov, npr. primarnih impulzov (instinktov), ega, ida, drugega, družbenega okolja, internalizacije itd. Nasprotno, prav iz teh konceptov lahko sestavlja informacijske entitete, jih ozaveda ali pa jih sploh ne opazi, jih pušča v nezavednem. Vprašanje, ki se postavlja, je prav gotovo izziv avtorju, da poskuša podrobneje odgovoriti, kaj je lahko koncept zavestnega in nezavednega v okviru razširjene teorije informacije.

6. *Kako je sploh možno opisati zavestni tok, kje je po vašem točka razprtja (nem. die Erschlossenheit, angl. disclosure, hrv. dokučenost), ko je mogoče govoriti o pogledu v notranjost duha (mind) (npr. topološki primer krogle, v katero je bila napravljena luknjica, tako da je razprtje notranjosti krogle mogoče)?*

Odgovor: Vprašanje vprašuje po začetku zavesti, po točki, v kateri se informacija razpre (začne razumevati) kot zavest oziroma samozavest. Ali bistveno razumljeno, kdaj informacija pogleda ali že gleda sama vase, kdaj se ta pogled pojavi in kako se ta obrat in nezavednega v zavedno pokaže navznoter (samozavest) in navzven (zavest)? Izkustvo duha nam tu že marsikaj pove. Točka razprtja je kot že tolikanj doslej odvisna predvsem od opredelitve. Kot zavestna bitja se te zavesti opredelitve naenkrat zavemo in od tu naprej govorimo lahko o zavestnem razprtju. Razprtje torej ni ničesar drugega kot nova oblika razumevanja v okviru razumevanja kot nastajajoči informaciji (razumevajoča diferenciacija). Kriterije zavedanja je že danes mogoče specifično tako formulirati, da bi bili uporabni tudi za eksperimente v umetni inteligenci (npr. zdravorazumski modeli sklepanja).

Avtor se zaveda, da so dani odgovori spontani, filozofsko in teorijsko zahtevni in da je za njihove izčrpnejše utemeljitve potreben dodaten napor in izvirni premislek.

* * * * *

Oglas

Prosimo imetnike starejših letnikov časopisa Informatica, da odstopijo ali prodajo posamezne letnike časopisa, in sicer: letnik I (1977), II (1978), III (1979) in VIII (1984). Informacije dobite pri tehničnem uredniku dr. R. Murnu, tel. (061) 214 399.

Informatica

Informatica *is published four times a year in Winter, Spring, Summer and Autumn by the Slovene Society Informatika, Tržaška cesta 2, 61000 Ljubljana, Yugoslavia.*

Volume 15 Number 2 May 1991 YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

CONTENTS