Volume 18 Number 3 October 1994 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Profile: Hiroaki Kitano
Object Schema
Causality of Information
Data Reorganization
Compressed Transmission

The Slovene Society Informatika, Ljubljana, Slovenia

Informatica 18 (1994) Number 3, pp. 255–373

Informatica

An International Journal of Computing and Informatics

Basic info about Informatica and back issues may be FTP'd from ftp.arnes.si in magazines/informatica ID: anonymous PASSWORD: <your mail address>
FTP archive may be also accessed with WWW (worldwide web) clients with URL: ftp://ftp.arnes.si/magazines/informatica

Subscription Information: Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 61000 Ljubljana, Slovenia.

The subscription rate for 1994 (Volume 18) is
- DEM 50 (US$ 34) for institutions,
- DEM 25 (US$ 17) for individuals, and
- DEM 10 (US$ 4) for students

plus the mail charge DEM 10 (US$ 4).

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

LaTeX Tech. Support: Borut Žnidar, DALCOM d.o.o., Stegne 27, 61000 Ljubljana, Slovenia.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Žibertova 1, 61000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Department for Computer Science, Jožef Stefan Institute: Tel (+386) 61 1259 199, Fax (+386) 61 219 385, or use the bank account number 900–27620–5159/4 (LB 50101-678-51841 for domestic subscribers only), Ljubljanska banka d.d., Trg Republike 2, 61000 Ljubljana, Slovenia.

According to the opinion of the Ministry for Informing (number 23/216-92 of March 27, 1992), the scientific journal Informatica is a product of informative matter (point 13 of the tariff number 3), for which the tax of traffic amounts to 5%.

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Authomatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences (Janez Peklenik)

The issuing of the Informatica journal is financially supported by the Ministry for Science and Technology, Slovenska 50, 61000 Ljubljana, Slovenia.

PROFILES

The Profile of Professor Robert Trappl explicates his manifold international activity, serving the world community of cybernetics and systems research in the best possible manner. This fact can be easily recognized by looking into volumes of Cybernetics and Systems '94, published by World Scientific, Singapore, on 1912 pages. These volumes include interdisciplinary contributions from various authors of the globe. The Twelfth European Meeting on Cybernetics and Systems Research–1994 (April 5–8) was a genuine Vienna happening in the superlative sense of the word where papers have been presented in lecture rooms of the Catholic Theological Faculty in the old university building.

To understand the various activity of Professor Robert Trappl, the scientific, cultural and the civilian atmosphere of Vienna and the benevolence of its citizens has to be experienced. The realm of Cybernetics and Systems Research belongs to the European philosophical and scientific tradition. For instance, the Vienna Circle (Wiener Kreis) was the group of logical positivists (which included at various times, e.g. R. Carnap and K. Gödel—logisticism, A. Heyting—intuitionism, J. von Neuman—formalism) centered on Vienna University in the 1920s and 1930s. They have added the technical equipment and logical rigour of modern mathematical logic to the empirical tradition. They published its own manifesto, Erkenntis, and series of publications. Its considerable influence on English-language philosophy persisted long after the Circle itself.

Cybernetics and Systems Research is also a continuation of the European philosophy (e.g., E. Husserl, M. Heidegger, etc.) in understanding the global science, humanism and modern technology. It searches new ways of thinking and design, new conceptualism and methodology, new possibilities for humanity and survival. It unites various profiles of researchers, e.g. philosophers, mathematicians, humanists, engineers, economists in a circle of good will and globally flowing information. Professor Robert Trappl is a significant mentor in this world embracing community—constituting the cybernetic paradigm of human, natural and artificial systems.

Robert Trappl

Robert Trappl is professor and head of the Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, Austria.

He holds a Ph.D. in psychology (minor: astronomy), a diploma in sociology, and is an engineer for electrical engineering.

Robert Trappl has been a director of the Austrian Research Institute for Artificial Intelligence since its foundation in 1984, in which he had a leading part. The Institute's current staff amounts to more than 20 scientists (employed or with project contracts), plus some 10 graduate students.

He has held lectures at the University of Vienna as well as the Technical University of Vienna, and on many occasions at universities abroad. He has enjoyed lecturing from pure mathematics to "Theories of Consciousness" or "The Revolution of Cybernetics: From feedback control and systems models to cyberspace and virtual reality" (both held in this academic year). Among his favorite topics is a seminar entitled "Art/ificial Intelligence: The Relationship between Art and AI", which he organizes jointly with a Professor of Music and with a Professor of New Media.

He served as a coordinator of the "Artificial Intelligence" branch of the Computer Science studies offered jointly by these two Universities, and he has been one of the five professors who are members of the studies committee for computer science at the Vienna Technical University. He has also served as the chairman of numerous habilitation committees as well as committees for the appointment of professors. He has acted as a Chairman of the University of Vienna Conciliation Committee since its foundation in 1976, and he has been reelected in this capacity biennially.

He has published more than 100 articles, he is co-author, editor or co-editor of 20 books, the most recent being "Power, Autonomy, Utopia: New Approaches toward Complex Systems", Plenum, New York, "Advanced Topics in Artificial Intelligence", Springer, Heidelberg, and "Cybernetics and Systems '94", World Scientific, Singapore.

He has received several awards, among them

the Award of the Theodor Koerner Foundation, the Award of the Municipality of Vienna, the Innitzer Award for Medical Studies, and the Hoechst Award of the Medical School of the University of Vienna.

Robert Trappl was the first president of the Austrian Society for Cybernetic Studies in 1969 and he has been reelected annually in this capacity. He has acted since 1972 as conference chairman of the biennial European Meetings on Cybernetics and Systems Research, which have developed into the leading international conference in the field, worldwide.

He served as vice president of the International Federation for Systems Research from 1980–1984 and as its President from 1984 to 1988. He has been a member of the Board of Trustees of the Bavarian Research Center for Knowledge-Based Systems (Erlangen-Nuremberg, Munich, Passau), appointed by the Bavarian Minister of Science, since 1989. He has acted as a member of the permanent Committee for the German Conference on Artificial Intelligence since 1993.

He founded the journal "Applied Artificial Intelligence: An International Journal" in 1986 and has acted as its Editor-in-Chief since. He has been Editor-in-Chief of the "Journal of Cybernetics" (since 1980: "Cybernetics and Systems: An International Journal") since 1980. Both journals are published by Taylor & Francis, Washington, DC. In addition, he is serving as Associate Editor or on the Editorial (Advisory) Board of numerous journals - besides "Informatica" - including the "International Journal of General Systems Research", "Systems Research", "Cybernetics and Systems in Management", "Medical Expert Systems", "Artificial Intelligence in Medicine", "Revue Internationale de Systemique", "International Journal of Pattern Recognition and Artificial Intelligence", "Journal of Theoretical and Experimental Artificial Intelligence", "Machine Intelligence Series", "Journal of Information Science and Technology", "AI Communications", "AI & Society".

Moreover, he has served on the Program Committees of numerous international conferences, e.g. ECAI'92.

He has held contracts as a consultant for the OECD, the UNIDO, and the WHO.

He himself has led numerous projects, but he has also acted as a reviewer of project applications handed in at the Austrian Federal Ministry of Science and Research, the Austrian Fund for the Promotion of Scientific Research, the Austrian National Bank, the Volkswagenstiftung in Germany, the Swiss National Science Foundation, and others.

His interest in the relation between Arts and AI has led him into the area of applications of AI methods in cyberspaces of different kinds. He is therefore currently starting a project on intelligent autonomous software agents, which either act in international datanets for the retrieval of information, or can simulate personalities as background of synthetic actors, e.g. in interactive dramas. He has established contacts for cooperation with several research groups abroad, and a fruitful information exchange has already begun.

Artificial Intelligence from its beginning was always supported heavily by "defense ministries or agencies". The main intention was and is to improve warfare. Since the mid-eighties, Robert Trappl proposed ways to use Artificial Intelligence to help in the avoidance of wars, first, by proposing means to decrease the likelihood of international conflicts, e.g. through the improvement of mutual understanding by jointly developing real-world knowledge databases, or by narrowing the North-South gap, e.g. by making medical expert advice available for rural health workers, on portable computers. A prototype of such an expert system was developed at the Austrian Research Institute for Artificial Intelligence.

Second, many crises have led to war, but luckily not all of them. The increased memory capacity, speed of computation, and, most important, the advent of new Artificial Intelligence techniques, could enable decision makers in crisis situations to find out which means have helped to prevent the outbreak of war in comparable historical situations. Robert Trappl is currently leading a project, supported by the Austrian Federal Ministry of Science and Research, with coworkers from Austria, Germany, Switzerland and the U.S.A., in which already existing war and crisis databases are used to investigate the potential of case-based reasoning and other AI techniques in this respect.

Edited by *A.P. Železnikar*

CYBERNETICS AND SYSTEMS RESEARCH ON THEIR WAY TO THE 21st CENTURY

Robert Trappl
Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, and
Austrian Research Institute for Artificial Intelligence of the Austrian Society for Cybernetic Studies

238 papers were presented at the Twelfth European Meeting on Cybernetics and Systems Research, which took place from April 5–8, 1994 in the Main Building of the University of Vienna. The authors of these 238 papers came from 42 countries from all continents except the Antarctica. Altogether more than 400 scientists listened to and discussed the presentations in the 17 symposia. To give an idea about the topics and the distribution of the papers among these topics, see the symposium list below. In addition to these symposia, three plenary lectures were given by outstanding scientists: Prof. Margaret Boden of the University of Sussex talked about "Artificial Intelligence and Creativity", Prof. Stephen Grossberg, Boston University, about "Neural Networks for Learning, Recognition, and Prediction", and Prof. Stuart Umpleby, George Washington University, about "Twenty Years of Second Order Cybernetics".

How come that a conference with this strange title attracts such an international audience and has become "the" leading conference in cybernetics and systems research?

It was in 1971, when we, a group of Austrian scientists and practitioners who had just recently founded the "Österreichische Studiengesellschaft für Kybernetik (Austrian Society for Cybernetic Studies)", decided to invite colleagues from Austria and other countries to a conference for a first exchange of ideas. Since we did not know whether really more than 20 scientists would participate, we named it "Meeting". And though we were not sure if anyone outside of Austria would come, we dared to call it "European"; "international" or "world" seemed too pretentious.

To our surprise, 82 scientists, many of them from abroad, even from the USA and the USSR, joined us, 75 papers were presented. Encouraged by this success, we decided to continue with the meeting every second year.

Looking through the stack of proceedings piled up in the 22 years since then, a rise in the number of papers until 1980, and then a decline can be seen, reaching bottom (119) in 1986. But then the number of papers has again increased, leading to the 238 papers (selected from 295 draft papers submitted) in 1994.

Table 1: Markers, topics and number of papers of symposia at the Twelfth European Meeting on Cybernetics and Systems Research, which took place from April 5–8, 1994 in the Main Building of the University of Vienna.

	Symposium	Papers
A	General Systems Methodology	8
B	Advances in Mathematical Systems Theory	21
C	Fuzzy Systems, Approximate Reasoning and KBS	17
D	Designing and Systems, and Their Education	23
E	Humanity, Architecture and Conceptualization	23
F	Biocybernetics and Mathematical Biology	24
G	Systems and Ecology	6
H	Cybernetics and Informatics in Medicine	9
I	Cybernetics of Socio-Economic Systems	7
J	Systems, Management and Organization	20
K	Cybernetics of Country Development	10
L	Communication and Computers	8
M	Intelligent Autonomous Systems	18
N	Cybernetic Principles of Knowledge Development	9
O	Cybernetics, Systems, and Psychotherapy	6
P	Artificial Neural Networks and Adaptive Systems	14
Q	Artificial Intelligence and Cognitive Science	15
		238

Why has cybernetics and systems research become "fashionable" again? Many causes may have contributed to this change. A prominent one is the fact that the iron curtain, which had

been dividing Europe for more than 40 years, was torn down, democratic governments were established in most of the former socialist countries, also leading to the necessity of a shift from socialist economies to more or less moderate capitalist ones. Cybernetics and systems research are seen as methods and models to help understand the change, to assist in the design of new political and economic systems and to support a smooth transition from here to there, as expressed in many of the papers presented in symposium I, J and K. This is also expressed by the fact that, when ranking the countries by the number of authors, Poland and the Czech Republic are on ranks 4 and 5; USA, Austria and Italy being on rank 1 to 3.

In the Western world, the controllability of many systems, especially social and ecological ones, seemed to decrease. While the share of theoretical papers has in general slightly decreased, a shift to topics as the ones above is manifest (symposia D, E, F, G, O). Often, it even turned out to be necessary to model the vagueness of many variables (symposium C).

Already envisioned by the pioneers of cybernetics, artificial intelligence and its recent offsprings, intelligent autonomous systems and neural networks (already a hot topic in the late 40ies), have gained increasing attention (symposia M, P, Q).

"Cybernetics", as a term rarely used, and even sometimes avoided, in the early 80ies, has also become a buzzword again, especially as "cyberspace": While conventional cybernetics was mainly applied to analyse, model, and sometimes even shape the "real" world, the principles of cybernetics can also be used for the synthesis of virtual worlds or cyberspaces. At the 1992 conference, one plenary lecture was devoted especially to this topic ("Cyberpunk and Cyberspace: The Threatening of the Old and the Creation of a New Reality through Cybernetics"), the 1994 conference saw already a demonstration of World-Wide Web servers, for the "Principia Cybernetica" project of the Free University of Brussels (URL: `http://pespmc1.vub.ac.be/Default.html`) and for the Austrian Research Institute for Artificial Intelligence of the Austrian Society for Cybernetic Studies (URL: `http://www.ai.univie.ac.at/`). It is not risky to forecast that this topic will also play a crucial role in the forthcoming conferences.

Moreover, the European Meetings on Cybernetics and Systems Research have quite unintentionally come to serve also another purpose: They became a prominent place to find partners for joint research projects; many scientists even found new career opportunities.

If this short article has whetted your appetite, contact e-mail sec@ai.univie.ac.at or fax +43-1-5320652, and information about the Thirteenth European Meeting on Cybernetics and Systems Research (EMCSR), April 9–12, 1996, Vienna, is gladly provided.

For those who are planning ahead, please note: 14th EMCSR: April 14 - 17, 1998; 15th EMCSR: April 25–28, 2000.

Parallel Algorithms for the Complete and Restricted Transitive Closure of a Database Relation

Anestis A. Toptsis
Dept. of Computer Science and Mathematics
York University, Atkinson College, Toronto, Ontario, M3J 1P3, Canada
Phone: (416) 736-5232; Fax: (416) 736-5773; Email: anestis@cs.yorku.ca

Keywords: Transitive closure, parallel processing, relational database, load balancing, scalability.

Edited by: Matjaž Gams
Received: July 14, 1993 **Revised:** August 12, 1993 **Accepted:** October 4, 1993

Integration of data and knowledge bases presents a major challenge of efficient processing linear recursive queries on very large data volumes. A notable example is the computation of the transitive closure. Although a plethora of sequential transitive closure algorithms have been developed, parallel transitive closure algorithms are rather scarce. In this paper we present, analyze, and compare six parallel algorithms for the computation of the transitive closure of a database relation. Three of the algorithms compute the complete closure, while the other three are modifications tailored for the computation of the restricted closure. The algorithms make no assumptions about data organization and do not require any preset indices or preprocessed data. A share-nothing parallel architecture is assumed for all algorithms. The analysis and performance evaluation shows that the restricted closure algorithms outperform the complete closure ones by a significant factor. Moreover, four of the six presented algorithms possess load balancing and scalability properties that make them far superior to the conventional transitive closure parallel methods.

1 Introduction

Transitive closure is one of the most important operations in deductive database systems. This stems from the fact that any linear recursive query (i.e. one that has exactly one repetition of its left hand side in its right hand side, such as the clause $ancestor(X,Y) :- ancestor(X,Z), parent(Z,Y)$) can be answered by computing a transitive closure [9]. Transitive closure can be classified in complete and restricted. If none of the variables X and Y are fixed in the left hand side of the above clause, then we have the case where the computation of the complete transitive closure is required. If one of the variables X and Y is instantiated and the other is not (suppose variable X is instantiated to a), then the clause becomes $ancestor(a,Y) :- ancestor(a,Z), parent(Z,Y)$ and we have the case where the computation of the restricted transitive closure is required. Although by far out numbered by their sequential counterparts, several parallel transitive closure algorithms have been proposed ([1], [5], [15],[17], [18]). None of them addresses the issue of computation of the restricted closure. In this paper we present and analyze six parallel transitive closure algorithms, PTC, APTC, GPTC, RPTC, RAPTC, and RGPTC. The first three algorithms (PTC, APTC, GPTC) compute the complete transitive closure of a database relation. The other three (RPTC, RAPTC, and RGPTC) are modifications of the complete closure algorithms and are designed to work best with restricted closure queries. All algorithms compute the closure in logarithmic number of iterations with respect to the depth of the closure. Also, the algorithms make no assumptions about data organization and do not require any preset indices or preprocessed data. The only requirement is that data is stored in a relational database.

A share-nothing parallel architecture is assumed for all algorithms. This is the model typically used in most parallel database operations, including transitive closure.

As shown in Figure 1, the share-nothing model is a typical distributed memory machine with the additional restriction that no two nodes can share a common disk. The organization of the rest of the paper is as follows.

In Section 2, some background and notation are discussed. In Section 3, the parallel complete closure algorithms PTC, APTC, and GPTC are presented. In Section 4, the parallel restricted closure algorithms RPTC, RAPTC, and RGPTC are presented. In Section 5, we analyze the performance of all algorithms. Section 6 presents an analytical performance evaluation and comparison of all presented algorithms. Section 7 summarizes our findings and discusses future research directions.

2 Background

Given a graph $G = (V, E)$, the complete transitive closure of G is a graph $G^+ = (V^+, E^+)$, such that $V^+ = V$, and (x, y) is an edge in G^+ if and only if there is a path from x to y in the original graph G. An example of G and its complete transitive closure are shown in Figure 2.

The restricted transitive closure of G with respect to a vertex n, is a graph $G_n^+ = (V_n^+, E_n^+)$, such that $V_n^+ = V$, and $(x, y) \in E_n^+$ if and only if there is a path from n to y in the original graph G. The restricted closures ${G_a}^+$ and ${G_b}^+$ of G are shown in Figure 3.

In the context of relational database, a two-attribute relation $R(A, B)$ can be perceived as a directed graph whose vertices are labeled with the domain values of R and which has an edge from a vertex x to a vertex y if and only if (x, y) is a tuple of relation R. Then, the complete transitive closure of a relation R is $R^+ = R + R^2 + R^3 + \ldots + R^d$, where d is the depth of the closure, and $R^j = R^{j-1} * R$. We use "+" to denote the relational database union operation and "*" to denote the relational database composition operation. In general, relation R can have any number of attributes, however, only two attributes are of interest for the transitive closure. Therefore, we will assume that relation R has only two attributes, i.e. it is binary. Given two binary relations S and T with attributes A, B and C, D respectively, the composition $S * T$ is a relation containing tuples (a, d) such that (a, k) is a tuple of S, (m, d) is a tuple of T, and $k = m$. The composition $S*T$ can be expressed via the basic relational database operations of Cartesian product, selection, and projection, as $S*T = \pi_{A,D}\sigma_{B=C}(S \times T)$, where $\times$, σ, and π respectively denote the Cartesian product, selection, and projection operations. Typically, the relational join operation is used in place of the subexpression $\sigma_{B=C}(S \times T)$. Since the join (and more, the cartesian product) is reportedly probably the most expensive relational database operation, the composition operation is, subsequently, very expensive. Since the cost of the composition operation is vastly dominated by the cost of the join, hereafter we will treat the terms join and composition as synonyms, and use them interchangeably. When the computation of the transitive closure of a relation R with respect to a starting point 'a' is requested, one can first compute the complete closure $R^+ = R + R^2 + R^3 + \ldots + R^d$ and then perform a selection on the result, i.e. $R_a^+ = \sigma_{A="a"}(R^+)$. Such an approach incurs the overhead of computing the entire R^+ first. It is desirable if selection operations could be "pushed" in the right hand side of the above expression, since the selections would reduce the size of the intermediate result relations R^i, thus reducing the cost of the corresponding join operations. This is basically the approach adopted by most algorithms that are designed with the computation of the restricted closure in mind (e.g. [14]).

3 Complete Closure Algorithms

In this section we present three parallel algorithms, PTC, APTC, GPTC, that compute the complete transitive closure of a binary database relation R. Algorithm PTC is a parallelization of the sequential algorithm SMART [8], [19], [13], which is also given as a reference. Algorithm SMART is considered an important representative of logarithmic transitive closure algorithms, and it has influenced many subsequent algorithms of this kind.

3.1 Algorithm SMART

Algorithm SMART is the following:

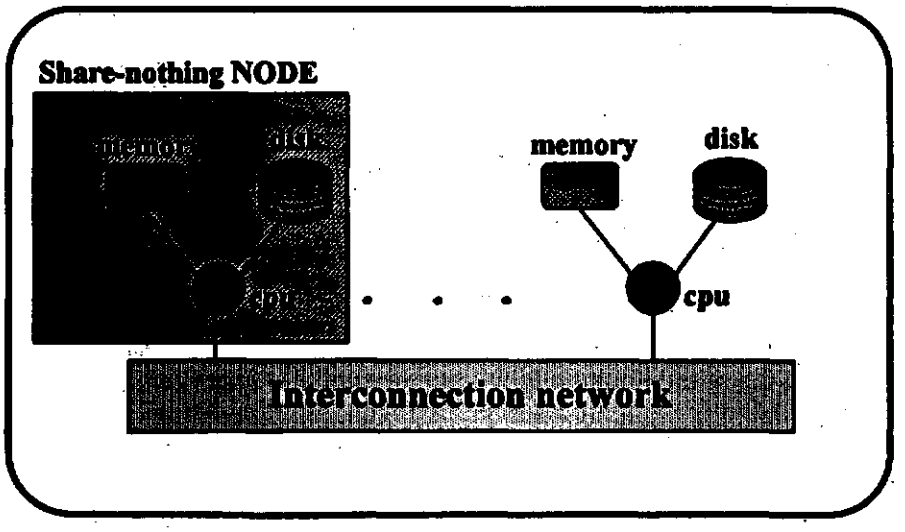

Figure 1: Share-nothing parallel computing model

for $m = 1$ to $\lceil log_2(D+1) \rceil$ do

1. $R^{2^m} := R^{2^{m-1}} * R^{2^{m-1}}$;
2. $T := (R + R^2 + \ldots + R^{2^m-1}) * R^{2^m}$;
3. $E := R^{2^m} + T + E$

endfor.

In the above algorithm, R, T, and E are binary database relations. R is the input relation whose transitive closure is requested. T is an intermediate result relation. Relation E accumulates the transitive closure of R. Algorithm SMART performs $\lceil log_2(D+1) \rceil$ iterations, where D denotes the depth of the complete closure.

3.2 Algorithm PTC

There is:

for $m = 1$ to $\lceil log_2(D+1) \rceil$
  for all $P_i, 1 \leq i \leq 2^{m-1}$
    $R^{i+2^{m-1}} := R^i * R^{2^{m-1}}$
  end for all
endfor.

The "for all P_i" statement in algorithm PTC means that processors P_1, P_2,..., P_i, perform the joins $R^i * R^{2^{m-1}}$, $i = 1, 2, ..., 2^{m-1}$ concurrently. That is, when $m = 1$, processor P_1 performs $R*R$; when $m = 2$ processors P_1 and P_2work concurrently. Processor P_1 performs $R * R^2$ and processor P_2 performs $R^2 * R^2$; etc. The table below shows the joins performed in the first three iterations in algorithms SMART and PTC.

Iter.	SMART	PTC	P
1	(a) $R * R$ (b) $R * R^2$	$R * R$	1
2	(a) $R^2 * R^2$ (b) $(R + R^2 + R^3) * R^4$	$R * R^2$ $R^2 * R^2$	2
3	(a) $R^4 * R^4$ (b) $(R + \ldots + R^7) * R^8$	$R * R^4$ $R^2 * R^4$ $R^3 * R^4$ $R^4 * R^4$	4

Table 1: Algorithms SMART and PTC at work

From Table 1, we can infer that during the m-th iteration, algorithm PTC computes exactly what algorithm SMART computes in step (b) of the $(m-1)$-th iteration together with step (a) of the m-th iteration. For example, algorithm PTC computes $R * R^2$ and $R^2 * R^2$ during iteration 2 and algorithm SMART computes $R * R^2$ during step (b) of iteration 1 and $R^2 * R^2$ during step (a) of iteration 2. Hereafter, when we refer to the

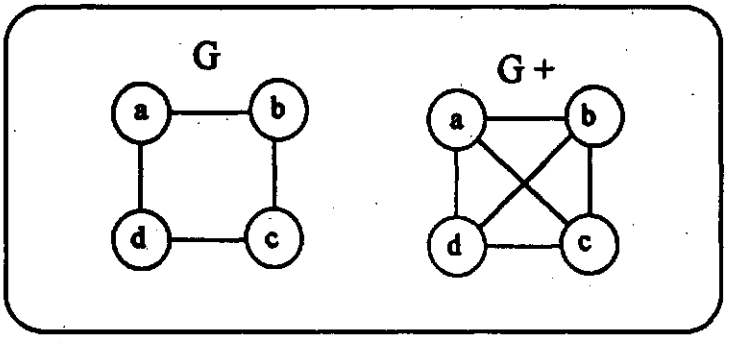

Figure 2: Graph G and its complete transitive closure graph G^+

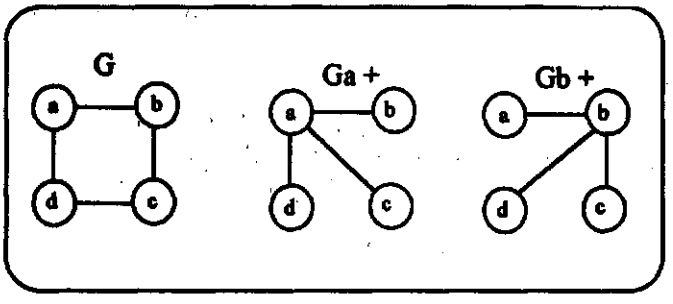

Figure 3: Graph G and its restricted transitive closure graphs G_a^+ and G_b^+

m-th iteration of the algorithms, we mean for algorithm PTC the m-th iteration of this algorithm as shown in Table 1 and for algorithm SMART step (b) of its $(m-1)$-th iteration together with step (a) of its m-th iteration. This setting of the meaning of the m-th iteration is fair, because this way both algorithms manipulate the same data volume and also generate the same tuples within the same iteration. Clearly, the load of each processor in PTC is much less than the load of the single processor used in algorithm SMART. On the other hand, as a result of the parallelism, algorithm PTC suffers the following limitations:

1. (Partial parallelism) During the entire course of execution of algorithm PTC, at least 50 % of the available processors are idle most of the time. Note, assuming that algorithm PTC terminates after k iterations, it uses $2^{\lceil log_2(D+1)\rceil}$ processors at the k-th iteration. Therefore, a machine with at least 2^{m-1} processors is required. However, the algorithm uses only half of these processors in the $(k-1)$-th iteration, and even fewer processors in the earlier iterations. Therefore, while algorithm PTC runs on a $2^{\lceil log_2(D+1)\rceil-1}$-processor machine, it only uses a rather small fraction of these processors during most of its execution.

2. (Non-scalability) Algorithm PTC ignores the availability of additional computing power. In case that $2^{\lceil log_2(D+1)\rceil-1}+z$ processors are available in the system, then z processors are useless as long as algorithm PTC is concerned.

Algorithm APTC, presented next, is an attempt to remedy the above limitations.

3.3 Algorithm APTC

We assume that $r \geq 2^{\lceil log_2(D+1) \rceil - 1}$ processors are available.

Algorithm APTC (All processors PTC):

for $m = 1$ to $\lceil log_2(D+1) \rceil$ do

in_parallel do

$R * R^{2^{m-1}}$ using $\frac{r}{2^{m-1}}$ processors;

$R^2 * R^{2^{m-1}}$ using $\frac{r}{2^{m-1}}$ processors;

...

$R^{2^{m-1}} * R^{2^{m-1}}$ using $\frac{r}{2^{m-1}}$ processors;

end in_parallel

endfor.

Algorithm APTC is a refinement of PTC and performs the same joins as algorithm PTC but in a different way. This different way of performing the joins is an attempt to eliminate the above two weaknesses of algorithm PTC Algorithm APTC has the following characteristics:

1. Algorithm APTC uses all $2^{\lceil log_2(D+1) \rceil - 1}$ processors in the system and it does so in every one on its iterations. This eliminates weaknesses 1 (partial parallelism) mentioned above. Moreover:

2. If more processors are (or become) available, algorithm APTC is able to utilize all of them, by simply dividing the total number of available processors into as many groups as the number of joins that must be performed during any iteration. Therefore, algorithm APTC does not suffer the scalability problem of algorithm PTC. In order to perform the joins $R*R^{2^{m-1}}$, $R^2*R^{2^{m-1}}$, ..., $R^{2^{m-1}}*R^{2^{m-1}}$ in the m-th iteration of APTC, the available processors are partitioned into 2^{m-1} groups. Each group consists of $\frac{r}{2^{m-1}}$ processors and these processors are assigned the task of performing the join $R^j * R^{2^{m-1}}$.

Limitations of algorithm APTC. Although algorithm APTC is able to use all available processors during any of its iterations, the processor load balancing is a major concern. As different groups of processors are assigned to perform different join operations, nothing guarantees that each join requires the same amount of effort. There may be only few joining tuples for the join in a group A of processors, while there may be thousands of joining tuples for the join in another group B. Also, the size of the joining relations $G_i$may widely differ from group to group.

Algorithm GPTC, presented next, is an attempt to remedy this load balancing deficiency.

3.4 Algorithm GPTC

Algorithm GPTC is a refinement of APTC and performs the same joins as algorithm PTC but using a different processor allocation scheme. Like algorithm APTC, algorithm GPTC uses all processors in any of its iterations, and, also, it is able to utilize more processors as they become available. Unlike algorithm APTC, algorithm GPTC allocates the available processors in groups of different sizes. The size of each group of processors depends on the predicted effort required to carry out the join assigned to that group. Specifically, in order to perform the joins $R*R^{2^{m-1}}$, $R^2*R^{2^{m-1}}$, ..., $R^{2^{m-1}}*R^{2^{m-1}}$ in the m-th iteration of GPTC, the available processors are partitioned into 2^{m-1} groups G_1, G_2, ..., $G_{2^{m-1}}$. Each group G_i consists of r_i processors and these r_i processors of group G_i are assigned the task of performing the join $R^i * R^{2^{m-1}}$. The number r_i of processors of group G_i is decided after taking in consideration the amount of data $D_i = |R^i| + |R^{2^{m-1}}|$ which is to be handled by group G_i. The bigger the D_i the more processors are allocated to group G_i. Specifically, for any two groups G_i and G_j, are such that $\frac{D_i}{D_j} = \frac{r_i}{r_j}$. For example, if $D_i = 2 \cdot D_j$ and if group G_i consists of r_i processors, then group G_j consists of $r_j = \frac{r_i}{2}$ processors. Upon completion of the formation of the groups G_1, G_2, ..., $G_{2^{m-1}}$, each group G_i performs the join $R^i * R^{2^{m-1}}$ using a parallel join algorithm. Different groups may use different join algorithms in order to perform the joins faster. The basic idea behind the decision to have a group of processors rather than a single processor to perform a join and at the same time to allocate more processors to perform joins which involve larger amounts of data, is to achieve equal distribution of work among all the processors available in the system. This will eliminate

the load balancing weakness of algorithm APTC, while maintain its full parallelism and scalability properties. We show later that if a hash-based parallel join algorithm is used by all groups, then in the best case all processors of the system *process the same amount of data*, while in the worst case algorithm GPTC coincides with algorithm APTC.

Algorithm GPTC:

We assume that $r \geq 2^{\lceil log_2(D+1) \rceil - 1}$ processors are available. Thus,

for $m = 1$ to k do

Phase I /* Grouping phase */

1. Split the r available processors of the system into groups G_1, G_2, ..., $G_{2^{m-1}}$ such that

   a. Group G_i consists of r_i processors, and

   b. $\frac{|R| + |R^{2^{m-1}}|}{r_1} = \frac{|R^2| + |R^{2^{m-1}}|}{r_2} = \ldots = \frac{|R^{2^{m-1}}| + |R^{2^{m-1}}|}{r_{2^{m-1}}}$, and

   c. $\sum_{j=1}^{2^{m-1}} r_j = r$

2. Designate group G_i to perform the join $R^i * R^{2^{m-1}}$. (This includes sending the reqired data to the processors of group G_i.)

Phase II /* Join phase */

All groups G_i, $1 \leq i \leq 2^{m-1}$, perform $R^i * R^{2^{m-1}}$ concurrently. Each group may use a join algorithm of its choice.

endfor.

The calculation of the size of each group in algorithm GPTC, is easily performed through solution of a simple system of equations. Note, prior to beginning iteration m, the relation sizes $|R|$, $|R^2|$, ..., $|R^{2^{m-1}}|$ are all known, since the corresponding relations have been generated in previous iterations. Note, we would like to have groups G_1, G_2, ..., $G_{2^{m-1}}$, such that the conditions (1b) and (1c) in Phase I of algorithm GPTC are satisfied. Let, $D_i = |R^i| + \left|R^{2^{m-1}}\right|$. Then,

$$(1b) \Rightarrow \frac{D_1}{r_1} = \frac{D_2}{r_2} = \ldots = \frac{D_{2^{m-1}}}{r_{2^{m-1}}}$$

and

$$\frac{D_1}{r_1} = \frac{D_k}{r_k},\ k = 1, 2, \ldots, 2^{m-1} \Rightarrow$$

$$r_k = \frac{D_k}{D_1} \cdot r_1,\ k = 1, 2, \ldots, 2^{m-1} \Rightarrow \text{[by (1c)]}$$

$$r_1 + \frac{D_2}{D_1} \cdot r_1 + \frac{D_3}{D_1} \cdot r_1 + \ldots + \frac{D_{2^{m-1}}}{D_1} \cdot r_1 = r \Rightarrow$$

$$r_1 = \frac{D_1 \cdot r}{\sum_{i=1}^{2^{m-1}} D_i}$$

It is easy to see that $r_k = \frac{D_k \cdot r}{\sum_{i=1}^{2^{m-1}} D_i}$, $k = 1, 2, \ldots, 2^{m-1}$. Recall, $D_k = \left|R^k\right| + \left|R^{2^{m-1}}\right|$, and, therefore, all quantities at the right hand side of the above expression are known prior to the start of iteration m. Therefore, the computation of r_k, $k = 1, 2, \ldots, 2^{m-1}$ can be easily carried out and thus the size of each group of processors be available before the m-th iteration begins.

4 Restricted Closure Algorithms

4.1 Algorithm RPTC

Algorithm RPTC (Restricted Parallel Transitive Closure) computes the restricted closure of a database relation, using multiple processors. The algorithm is inspired by the parallel algorithm PTC [15], which computes the complete transitive closure.

Algorithm RPTC

for $m = 1$ to $\lceil log_2(D+1) \rceil$ do
  for all $P_i, 1 \leq i \leq 2^{m-1}$
    $\pi_B \sigma_A \left(R^{i+2^{m-1}}\right) :=$
    $\pi_B \left(\pi_B \sigma_A \left(R^i\right) * R^{2^{m-1}}\right)$;
  for P_0 *do*
    $R^{2^m} := R^{2^{m-1}} * R^{2^{m-1}}$
endfor.

In algorithm RPTC processors P_1, P_2,..., P_i, perform the joins $R^i * R^{2^{m-1}}$, i =

$1, 2, \ldots, 2^{m-1}$ concurrently. For example, during the second iteration of algorithm RPTC ($m = 2$), three processors work concurrently. One of them performs the composition $\pi_B\sigma_A(R) * R^2$, the other one performs $\pi_B\sigma_A(R^2) * R^2$, and the last one performs $R^2 * R^2$. Note, in the right hand side of the statement in the "for all" loop of algorithm RPTC, the size of relation R^i is reduced by the selection σ_A and the projection π_B prior to performing the composition with relation $R^{2^{m-1}}$. This reduction in size is not possible if the complete closure were to be computed. Nevertheless, algorithm RPTC suffers the same drawbacks as algorithm PTC, namely, partial parallelism and non-scalability.

Algorithm RAPTC, presented next, is an attempt to remedy these deficiencies.

4.2 Algorithm RAPTC

Algorithm RAPTC (Restricted APTC)

for $m = 1$ to $\lceil log_2(D+1) \rceil$ do
In_parallel do

$\pi_B\sigma_A(R) * R^{2^{m-1}}$, using $\frac{r}{2^{m-1}+1}$ processors;

$\pi_B\sigma_A(R^2) * R^{2^{m-1}}$, using $\frac{r}{2^{m-1}+1}$ processors;

...

$\pi_B\sigma_A\left(R^{2^{m-1}}\right) * R^{2^{m-1}}$, using $\frac{r}{2^{m-1}+1}$ processors;

$R^{2^{m-1}} * R^{2^{m-1}}$, using $\frac{r}{2^{m-1}+1}$ processors

end In_parallel
endfor.

Algorithm RAPTC is the restricted closure version of algorithm APTC. It uses all available processors in every iteration. Also, as in algorithm APTC, the processor load balancing is not taken in account. Moreover, in algorithm RAPTC, the issue of load balancing is more important since one of the joins performed in parallel is much more costly than the others. Specifically, in the join $R^{2^{m-1}} * R^{2^{m-1}}$ the size of the joining relations is not reduced as in the join $\pi_B\sigma_A(R^i) * R^{2^{m-1}}$. Therefore, the group of processors that perform $R^{2^{m-1}} * R^{2^{m-1}}$ becomes clearly a bottleneck in the execution time of RAPTC. Algorithm RGPTC, presented next, is an attempt to resolve this load balancing deficiency.

4.3 Algorithm RGPTC

Algorithm RGPTC

We assume that $r \geq 2^{\lceil log_2(D+1) \rceil - 1}$ processors are available. Then,

for $m = 1$ to k do

Phase I /* Grouping phase */

1. Split the r available processors of the system into groups $G_1, G_2, \ldots, G_{2^{m-1}+1}$ such that
   a. Group G_i consists of r_i processors;
   b. $$\frac{|\pi_B\sigma_A(R)| + \left|R^{2^{m-1}}\right|}{r_1} = \frac{|\pi_B\sigma_A(R^2)| + \left|R^{2^{m-1}}\right|}{r_2} = \ldots = \frac{\left|\pi_B\sigma_A(R^{2^{m-1}})\right| + \left|R^{2^{m-1}}\right|}{r_{2^{m-1}}} = \frac{\left|R^{2^{m-1}}\right| + \left|R^{2^{m-1}}\right|}{r_{2^{m-1}+1}};$$
   c. $\sum_{j=1}^{2^{m-1}+1} r_j = r$;
2. Designate group $G_i, i = 1, 2, \ldots, 2^{m-1}$ to perform the join $\pi_B\sigma_A(R^i) * R^{2^{m-1}}$, and group $G_{2^{m-1}+1}$ to perform the join $R^{2^{m-1}} * R^{2^{m-1}}$. (This includes sending the required data to the processors of group G_i.)

PhaseII /* Join phase */

All groups G_i, $1 \leq i \leq 2^{m-1}+1$, perform their joins concurrently. Each group may use a join algorithm of its choice.

endfor.

Algorithm RGPTC (Restricted Group Parallel Transitive Closure) uses all the available processors in the system and it does so in every one on its iterations. In order to perform the joins

$\pi_B\sigma_A(R^i)*R^{2^{m-1}}$, $1 \le i \le 2^{m-1}$, in the m-th iteration of RGPTC, the available processors are partitioned into 2^{m-1} groups G_1, G_2, ..., $G_{2^{m-1}}$. Each group G_i consists of r_i processors and these r_i processors of group G_i are assigned the task of performing the join $\pi_B\sigma_A(R^i)*R^{2^{m-1}}$. The number r_i of processors of group G_i is decided after taking in consideration the amount of data $D_i = |\pi_B\sigma_A(R^i)| + \left|R^{2^{m-1}}\right|$ which is to be handled by group G_i. The bigger the D_i the more processors are allocated to group G_i. This is expressed by step (1.b) of PHASE I in the above algorithm. Upon completion of the formation of the groups G_1, G_2, ..., $G_{2^{m-1}}$, each group G_i performs the join $\pi_B\sigma_A(R^i)*R^{2^{m-1}}$ using a parallel join algorithm. Different groups may use different join algorithms in order to perform the joins faster. The basic idea behind the decision to have a group of processors rather than a single processor to perform a join and at the same time to allocate more processors to perform joins which involve larger amounts of data, is to achieve equal distribution of work among all the processors available in the system. Note, in algorithm RPTC there is only one processor for each join and, also, not all available processors are utilized in every iteration.

5 Analysis

In this section we analyze the algorithms PTC, APTC, GPTC, RPTC, RAPTC, and RGPTC. The main issues taken in account in our analysis are the costs incurred by the join operations and the communication. For comparison purposes, the analysis of algorithm SMART is also provided. The joins can be performed as nested loops join, merge-sort join, or hash-based join. Among these three methods, hash-based join has been reported to perform the best (e.g. [3], [4]). Therefore, for the join operations in our algorithms, a hash based join is assumed. We denote the size of a relation S in number of tuples, by $|S|$. Using a hash join method, two relations S and T can be joined in $\Theta(|S| + |T|)$ time ($|S|$ denotes the number of tuples in relation S). (Note, for nested loops and merge-sort, $\Theta(|S| \cdot |T|)$, and $\Theta(|S| \cdot \log_2|S| + |T| \cdot \log_2|T|)$ units of time are required respectively.) During the m-th iteration of algorithm SMART the joins $R^{2^{m-1}} * R^{2^{m-1}}$ and $(R + R^2 + \ldots + R^{2^m-1}) * R^{2^m}$ are performed. The cost of these joins is

$$C_J^{(m)}(\text{SMART}) = 2 \cdot \left|R^{2^{m-1}}\right| + |R| + |R^2| + \ldots + \left|R^{2^{m-1}-1}\right| + \left|R^{2^{m-1}}\right| \quad (1)$$

During the m-th iteration of algorithm PTC the joins $R * R^{2^{m-1}}$, $R^2 * R^{2^{m-1}}$, ..., $R^{2^{m-1}} * R^{2^{m-1}}$, are performed in parallel. The cost of these joins is

$$C_J^{(m)}(\text{PTC}) = \left|R^{2^{m-1}}\right| + \max\left\{|R|, |R^2|, \ldots, \left|R^{2^{m-1}}\right|\right\} \quad (2)$$

The size $|R^i|$ of the relations R^i in expressions (1) and (2) above, depends on two parameters. One is the join selectivity (JS) and the other is $|R|$. The join selectivity of two relations S and T is expressed as $\text{JS}(S,T) = \frac{|S*T|}{|S| \cdot |T|}$. Therefore, $|S*T| = JS(S,T) \cdot |S| \cdot |T|$. Assuming that JS is constant, we can derive $|R^i| = JS^{i-1} \cdot |R|^i$, for any $i \ge 1$. Equivalently,

$$\left|R^i\right| = A^{i-1} \cdot |R| \quad (3)$$

for any $i \ge 1$, where $A = JS \cdot |R|$. In expression (3), if $A = 1$ then $|R^i| = |R|$. That is, the size of any relation resulting from a join is the same as the size of the initial relation R. If $A > 1$, then the joins generate relations of bigger and bigger size. If $A < 1$, then the joins generate relations of smaller and smaller size. By looking at formula (2) (join cost for m-th iteration of algorithm PTC) we observe the following. When $A = 1$, all processors which participate in the execution of the m-th iteration have to handle the same amount of data. That is, the work is equally distributed among all the active processors. This is a desirable property for any parallel algorithm, including algorithm PTC. When $A > 1$, formula (2) becomes $C_J^{(m)}(\text{PTC}) = 2 \cdot \left|R^{2^{m-1}}\right|$. That is, processor $P_{2^{m-1}}$ has to handle more data than what each of the processors P_1, P_2, ..., $P_{2^{m-1}-1}$ has. This is not a desirable property since the work is not equally distributed among all processors. Similar situation arises when $A < 1$. Below, we

present formulae for the join and communication cost of all discussed algorithms. The derivation of all presented expressions are based on the above discussion. A brief discussion is provided in cases that a derivation is somewhat complicated.

5.1 Join cost

SMART: Join cost of m-th iteration:

$$A = 1: \quad C_J^{(m)}(\text{SMART}) = 2 \cdot |R| + 2^{m-1} \cdot |R| = 2^m \cdot |R|;$$

$$A \neq 1: \quad C_J^{(m)}(\text{SMART}) = |R| \cdot (1 + A + A^2 + \ldots + A^{2^{m-1}-2} + 3 \cdot A^{2^{m-1}-1}).$$

SMART: Join cost over $k = \lceil log_2(D+1) \rceil$ iterations:

$$A = 1: \quad \text{total\_}C_J(\text{SMART}) = |R| \cdot \sum_{m=1}^{k} 2^m = |R| \cdot 2 \cdot \left(2^k - 1\right)$$

$$A \neq 1: \quad \text{total\_}C_J(\text{SMART}) = |R| \cdot \sum_{m=1}^{k} \sum_{j=0}^{2^{m-1}-2} A^j + 3 \cdot |R| \cdot \sum_{m=1}^{k} A^{2^{m-1}-1}$$

PTC: Join cost of m-th iteration:

$$A = 1: \quad C_J^{(m)}(\text{PTC}) = 2 \cdot |R|;$$

$$A > 1: \quad C_J^{(m)}(\text{PTC}) = 2 \cdot |R| \cdot A^{2^{m-1}-1};$$

$$A < 1: \quad C_J^{(m)}(\text{PTC}) = |R| \cdot \left(1 + A^{2^{m-1}-1}\right)$$

PTC: Join cost over $k = \lceil log_2(D+1) \rceil$ iterations:

$$A = 1: \quad \text{total\_}C_J(\text{PTC}) = 2 \cdot k \cdot |R|;$$

$$A > 1: \quad \text{total\_}C_J(\text{PTC}) = 2 \cdot |R| \cdot \sum_{m=1}^{k} A^{2^{m-1}-1};$$

$$A < 1: \quad \text{total\_}C_J(\text{PTC}) = |R| \cdot \left(k + \sum_{m=1}^{k} A^{2^{m-1}-1}\right)$$

APTC: Join cost of m-th iteration. [Same as GPTC (see below).]

APTC: Join cost over $k = \lceil log_2(D+1) \rceil$ iterations. [Same as GPTC (see below).]

GPTC: Join cost of m-th iteration:

We mentioned in the description of algorithm GPTC that each group is free to choose its own join strategy. However, again, in order to simplify our analysis we assume that all groups choose to perform their joins using the same hash-based join algorithm. In this algorithm, in order to join two relations S and T, hashing is used to distribute S and T into buckets S_i and T_i. Then each processor is assigned to join one or more pair (S_i, T_i) of buckets. An important issue which we must consider when S and T are split into buckets is the *data skew* [12]. In partitioning S and T into buckets, the hashing may not result to buckets of equal size. This happens when the joining attributes of S and T carry non distinct values and/or a perfect hashing function cannot be found. Such data is referred to as skewed data [12]. A zero data skew means that all buckets will have approximately the same size (best case) whereas a non zero data skew means that one bucket will be bigger than each of the remaining buckets. Thus, when the buckets are distributed among the processors, the processor which receives the largest bucket becomes the bottleneck in the computation since it will generally take more time to perform its corresponding join. The worst case occurs when both relations S and T each hashes into a single bucket. In such a case, the two buckets are sent to the same processor P and P performs $S * T$ while the remaining processors within the processor-group of P are idle. In such a case we say that the skew of S is equal to 1 and also the skew of T is equal to 1. A data skew of one is quite uncommon, as is a data skew of zero (although zero is more common than one). Typical cases are those with a skew between 3% and 10% [12]. Although there is a fairly rich literature on join operations which take in account the issue of data skew (e.g. [6], [7], [10], [11], [16]), the problem of joining skewed relations has not been satisfactorily solved to date. Also, to the best of our knowledge, the issue of data skew has never been accounted in transitive closure algorithms,

and a zero data skew is typically assumed when the issue arises [e.g. [1]). Unfortunately, our attempts to incorporate data skew in the presented cost expressions were also fruitless, except for the case where data skew = 1. Therefore, we also assume that the data skew is zero in our analysis.

Let $\max\_DS\left(G_i^{(m)}\right)$ be the maximum data size handled by a processor within group G_i in the m-th iteration. Then, $C_J^{(m)}(\text{GPTC}) = \max\left\{\max\_DS\left(G_i^{(m)}\right)\right\}$, $i = 1,2,3,...,2^{m-1}$. When data skew = 0, all buckets are of equal size. Therefore, $\max\_DS\left(G_i^{(m)}\right) = \frac{|R^i| + \left|R^{2^{m-1}}\right|}{r_i}$. Therefore, if A=1, $C_J^{(m)}(\text{GPTC}) = \max\left\{\frac{|R|+|R|}{r_i}\right\} = 2\cdot|R|\cdot\max\left\{\frac{1}{r_i}\right\}, 1 \le i \le 2^{m-1}$. Also, when A=1, $|R| = |R^j|$, for any j, that is, all relations participating in the joins in algorithm GPTC, are of equal size. Therefore, all groups of processors are of equal size and $r_i = \frac{r}{2^{m-1}}$, where r is the total number of processors. Therefore, $G_J^{(m)}(\text{GPTC}) = 2\cdot|R| = \frac{2^m\cdot|R|}{r}$. In summary,

$$A=1:\quad C_J^{(m)}(\text{GPTC}) = \frac{2^m\cdot|R|}{r} = C_J^{(m)}(APTC);$$

$$A\neq 1:\quad C_J^{(m)}(\text{RPTC}) = \frac{|R|\cdot\left(2^{m-1}\cdot A^{2^{m-1}-1} + \frac{A^{2^{m-1}}-1}{A-1}\right)}{r}$$

GPTC: Join cost over $k = \lceil log_2(D+1)\rceil$ iterations, when data skew = 0

$$A=1:\quad \text{total\_}C_J(\text{GPTC}) = \frac{|R|\cdot\left(2^{k+1}-2\right)}{r};$$

$$A\neq 1:\quad \text{total\_}C_J(\text{GPTC}) = \frac{|R|}{r}\cdot\sum_{m=1}^{k}\left(2^{m-1}\cdot A^{2^{m-1}-1}\right) + \frac{|R|}{r}\cdot\sum_{m=1}^{k}\frac{A^{2^{m-1}}-1}{A-1}$$

RPTC: Join cost of m-th iteration:

From the description of algorithm RPTC,

$$C_J^{(m)}(\text{RPTC}) = \max\left\{\max\left\{|\pi_B\sigma_A(R^i)|+\left|R^{2^{m-1}}\right|\right\}, \left|R^{2^{m-1}}\right|\right\}, \quad i = 1,2,\ldots,2^{m-1} \tag{4}$$

We can write,

$$\left|\pi_B\sigma_A\left(R^i\right)\right| = \text{PIF}\cdot\text{SS}\cdot\left|R^i\right| \tag{5}$$

where PIF denotes the *projection improvement factor*, that is, the amount of decrease in the size of a relation T if a projection is imposed on T, and SS denotes the *selection selectivity*, that is, the number of tuples of relation R^i that go into relation $\sigma_A(R^i)$. PIF and SS are introduced since the size of relation $\pi_B\sigma_A(R^i)$ is clearly less than the size of relation R^i, and this should be taken in account in our analysis. Using expressions (4) and (5), it is easy to derive that:

$A=1:$ $C_J^{(m)}(\text{RPTC}) = 2\cdot|R|$ since always must do a $R^i * R^i$ join at one processor.

$A>1:$ $C_J^{(m)}(\text{RPTC}) = 2\cdot|R|\cdot A^{2^{m-1}-1}$. Note, the cost at each iteration is dominated by $R^i * R^i$; and $R^i < R^j$ if $i < j$.

$A<1:$ $C_J^{(m)}(\text{RPTC}) = |R|\cdot\left(\text{PIF}\cdot\text{SS} + A^{2^{m-1}-1}\right)$

RPTC: Join cost over $k = \lceil log_2(D+1)\rceil$ iterations:

$A=1:$ $\text{total\_}C_J(\text{RPTC}) = 2\cdot k\cdot|R|$;

$A>1:$ $\text{total\_}C_J(\text{RPTC}) = 2\cdot|R|\cdot\sum_{m=1}^{k}A^{2^{m-1}-1}$;

$A<1:$ $\text{total\_}C_J(\text{RPTC}) = k\cdot\text{PIF}\cdot\text{SS} + \sum_{m=1}^{k}A^{2^{m-1}-1}$

RAPTC: Join cost of m-th iteration:

From the description of algorithm RAPTC,

$$C_J^{(m)}(\text{RPTC}) = \max\left\{\max\left\{\frac{|\pi_B\sigma_A(R^i)| + \left|R^{2^{m-1}}\right|}{\frac{r}{2^{m-1}+1}}\right\}; \frac{\left|R^{2^{m-1}}\right| + \left|R^{2^{m-1}}\right|}{\frac{r}{2^{m-1}+1}}\right\}; \quad (6)$$

$i = 1, 2, \ldots, 2^{m-1}$

Using $|\pi_B\sigma_A(R^i)| = \text{PIF} \cdot \text{SS} \cdot |R^i|$ (5), as above, we derive:

$$A = 1: \quad C_J^{(m)}(\text{RAPTC}) = \frac{2 \cdot (2^{m-1}+1) \cdot |R|}{r};$$

$$A > 1: \quad C_J^{(m)}(\text{RAPTC}) = \frac{2 \cdot (2^{m-1}+1) \cdot |R| \cdot A^{2^{m-1}-1}}{r};$$

$$A < 1: \quad C_J^{(m)}(\text{RAPTC}) = \frac{(2^{m-1}+1) \cdot \left(\text{PIF} \cdot \text{SS} + A^{2^{m-1}-1}\right) \cdot |R|}{r}$$

RAPTC: Join cost over $k = \lceil log_2(D+1) \rceil$ iterations:

$$A = 1: \quad \text{total\_}C_J(\text{RAPTC}) = \frac{2 \cdot \left(k + 2^k\right) \cdot |R|}{r};$$

$$A > 1: \quad \text{total\_}C_J(\text{RAPTC}) = \frac{2 \cdot |R|}{r} \cdot \sum_{m=1}^{k} \left(2^{m-1}+1\right) \cdot A^{2^{m-1}-1};$$

$$A < 1: \quad \text{total\_}C_J(\text{RAPTC}) = \frac{|R|}{r} \cdot \left(\text{PIF} \cdot \text{SS} \cdot \left(2^k + k + \sum_{m=1}^{k} A^{2^{m-1}-1}\right)\right) + \frac{|R|}{r} \cdot \sum_{m=1}^{k} 2^{m-1} \cdot A^{2^{m-1}-1}$$

RGPTC: Join cost of m-th iteration

Using the methodology for the analysis of algorithms GPTC and RPTC, RAPTC, and assuming a data skew of zero, we can derive the following:

$$A = 1: \quad C_J^{(m)}(\text{RGPTC}) = \frac{2 \cdot (2^{m-1}+1) \cdot |R|}{r};$$

$$A > 1: \quad C_J^{(m)}(\text{RGPTC}) = \frac{2 \cdot (2^{m-1}+1) \cdot |R| \cdot A^{2^{m-1}-1}}{r};$$

$$A < 1: \quad C_J^{(m)}(\text{RGPTC}) = \frac{(2^{m-1}+1) \cdot (\text{PIF} \cdot \text{SS} + 1) \cdot |R|}{r}$$

RGPTC: **Join cost over $k = \lceil log_2(D+1) \rceil$ iterations**

$$A = 1: \quad \text{total\_}C_J(\text{RGPTC}) = \frac{2 \cdot \left(2^k + k\right) \cdot |R|}{r};$$

$$A > 1: \quad \text{total\_}C_J(\text{RGPTC}) = \frac{2 \cdot |R|}{r} \cdot \sum_{m=1}^{k} \left(2^{m-1}+1\right) \cdot A^{2^{m-1}-1};$$

$$A < 1: \quad \text{total\_}C_J(\text{RGPTC}) = \frac{|R|}{r} \cdot \left(2^k + k\right) \cdot (\text{PIF} \cdot \text{SS} + 1)$$

5.2 Communication cost

PTC: Communication cost of m-th iteration

The first few iterations of the communication activity in algorithm PTC are shown in Table 2. In this table, "$P_i : T$ to P_j" means that processor P_i sends relation T to processor P_j.

Iter.	Communication activity
1	No communicaiotn
2	$P_1 : R^2$ to P_2
3	$P_1 : R^3$ to P_3 $P_1 : R^4$ to P_1, P_3, P_4
4	$P_1 : R^5$ to P_5 $P_2 : R^6$ to P_6 $P_3 : R^7$ to P_7 $P_4 : R^8$ to $P_1, ..., P_8$

Table 2: Communication activity in algorithm PTC

From Table 2 we can infer that the amount of data transferred prior to beginning of the m-th iteration ($m \geq 2$) is

$$C_C^{(m)}(PTC) = \left|R^{2^{m-2}+1}\right| + \left|R^{2^{m-2}+2}\right| + ... + \left|R^{2^{m-1}-1}\right| + 2^{m-1} \cdot \left|R^{2^{m-1}}\right|$$

Therefore, it is derived that:

$$A = 1: \quad C_C^{(m)}(PTC) = |R| \cdot (2^m - 2^{m-2} - 1);$$

$$A \neq 1: \quad C_C^{(m)}(PTC) = |R| \cdot A^{2^{m-1}-1} \cdot (2^{m-1} - 1) + |R| \cdot \frac{A^{2^{m-1}-1} - A^{2^{m-2}}}{A-1}$$

PTC: Communication cost over $k = \lceil log_2(D+1) \rceil$ iterations:

$$A = 1: \quad \text{total\_}C_C(PTC) = |R| \cdot \left(2^{k+1} - 2^{k-1} - k - 1\right);$$

$$A \neq 1: \quad \text{total\_}C_C(PTC) = |R| \cdot \sum_{m=2}^{k} A^{2^{m-1}-1} \cdot (2^{m-1}+1) + |R| \cdot \frac{A^{2^{m-1}-1} - A^{2^{m-2}}}{A-1}$$

Note, when $m = 1$, there is no communication in algorithm PTC.

APTC and GPTC: Communication cost of m-th iteration:

The number of tuples involved during the communication or grouping phase of algorithm APTC is the same as in algorithm GPTC. Moreover, this cost is the same as the one incurred by algorithm PTC, since algorithm PTC transfers the same data over the network. Therefore, for all values of A, during the m-th iteration, where $m \geq 2$,

$$C_C^{(m)}(GPTC) = C_C^{(m)}(APTC) = C_C^{(m)}(PTC)$$

When $m = 1$, GPTC distributes R to r processors, while PTC does not incur any communication cost.

APTC and GPTC: Communication cost over $k = \lceil log_2(D+1) \rceil$ iterations:

The total number of tuples involved over k iterations is

$$\text{total\_}C_C(GPTC) = |R| + \text{total\_}C_C(PTC)$$

RPTC: Communication cost of m-th iteration

The first few iterations of the communication activity in algorithm RPTC are shown in Table 3, where "$P_i : T$ to P_j" means that processor P_i sends relation T to processor P_j.

Iter.	Communication activity
1	P_0 : R to P_1
2	P_1 : $\pi_B\sigma_A(R^2)$ to P_2 P_0 : R^2 to P_1, P_2
3	P_1 : $\pi_B\sigma_A(R^3)$ to P_3
	P_2 : $\pi_B\sigma_A(R^4)$ to P_4 P_0 : R^4 to P_1, P_2, P_3, P_4
4	P_1 : $\pi_B\sigma_A(R^5)$ to P_5 P_2 : $\pi_B\sigma_A(R^6)$ to P_6 P_3 : $\pi_B\sigma_A(R^7)$ to P_7 P_4 : $\pi_B\sigma_A(R^8)$ to P_8 P_0 : R^8 to $P_1, P_2, ..., P_8$

Table 3: Communication activity in algorithm RPTC.

Therefore,

$$A = 1: \quad C_C^{(m)}(\text{RPTC}) = |R| \cdot (2^{m+1} + \text{PIF} \cdot \text{SS} \cdot 2^{m-2});$$

$$A \neq 1: \quad C_C^{(m)}(\text{RPTC}) = |R| \cdot A^{2^{m-1}-1} \cdot 2^{m-1} + \text{PIF} \cdot \text{SS} \cdot |R| \cdot \frac{A^{2^{m-1}-1} - A^{2^{m-2}}}{A-1}$$

RPTC: Communication cost over $k = \lceil log_2(D+1) \rceil$ iterations

Total number of tuples involved in communication over k iterations:

$$A = 1: \quad \text{total\_}C_C\,(\text{RPTC}) = |R| \cdot \left(2^k - 1\right) + |R| \cdot \left(2^{k-1} - 1\right) \cdot \text{PIF} \cdot \text{SS};$$

$$A \neq 1: \quad \text{total\_}C_C\,(\text{RPTC}) = |R| \cdot \sum_{m=1}^{k} \left(A^{2^{m-1}-1} \cdot 2^{m-1}\right) + \text{PIF} \cdot \text{SS} \cdot |R| \cdot \sum_{m=1}^{k} \frac{A^{2^{m-1}-1} - A^{2^{m-2}}}{A-1}$$

RAPTC and RGPTC: Communication cost of m-th iteration

From the discussion of the costs of the complete closure versions of these algorithms it follows that

$$\text{total\_}C_C\,(\text{GPTC}) = \text{total\_}C_C\,(\text{APTC}) = \text{total\_}C_C\,(\text{PTC})$$

RAPTC and RGPTC: Communication cost over $k = \lceil log_2(D+1) \rceil$ iterations:

Similar to the above,

$$\text{total\_}C_C\,(\text{GPTC}) = |R| + \text{total\_}C_C\,(\text{PTC})$$

6 Performance Evaluation

We follow the model and formulae set by [13] for evaluating our algorithms. This model is chosen since it is well accepted in the literature, and also provides a single figure for the performance of each algorithm, namely its cost in amount of time. The formulae used in our evaluation as well as their default values, are shown in the APPENDIX at the end of the paper. The algorithms are evaluated in terms of

- size of the initial relation R (whose closure is to be found),
- depth of the transitive closure,
- number of available processors,
- join selectivity,
- selection selectivity, and
- memory capacity per processor.

The results are shown in Figures 4–10. These figures do not show the performance of algorithms APTC and RAPTC because it is not necessary. According to our analysis in the previous section, algorithm APTC has the same join and communication costs as algorithm GPTC. This happens because of our assumption that the data skew is zero. For the restricted closure versions, RAPTC has the same communication cost as algorithm RGPTC. For join costs, when $A \geq 1$ the join cost of RAPTC is the same as the one of RGPTC; when $A < 1$ the join cost of RAPTC is greater than the one of RGPTC. However, the difference only depends on the value of A, and since $A < 1$ and also A is raised to some power, the difference is insignificant. Therefore, the performance of algorithms APTC and RAPTC can be considered identical to the one of GPTC and RGPTC respectively (when data skew = 0), and in order to avoid clattering the Figures, algorithms APTC and RAPTC are not plotted.

In Figure 4, algorithm GPTC clearly outperforms algorithm PTC due to its load balancing features. For relatively small relations (100K tuples) GPTC is about 10 times faster than PTC, and as the relation size increases the performance differential also increases.

Figure 5 shows that RGPTC outperforms algorithm RPTC for all tested sizes of the initial relation R. For smaller R sizes, i.e. up to one million tuples, algorithm RGPTC outperforms RPTC 3.7 times on the average. For larger R sizes, two to ten million tuples, algorithm RGPTC outperforms RPTC 9 times on the average. Additional results, not presented here, have shown that when the size of the original relation R becomes more than 10 million tuples, the performance differential of the two algorithms becomes very severe, and algorithm RGPTC outperforms RPTC 240 times on the average. Overall, the load balancing ability and the scalability of algorithm RGPTC is vividly demonstrated.

Figure 6.(a) demonstrates that as the depth of the closure increases, the performance of PTC deteriorates, both in absolute numbers and in comparison to the performance of algorithm GPTC. This is due to the poor utilization of the available processors by algorithm PTC. On the other hand, the performance of algorithm GPTC is immune to depth increases. This is because of the excellent

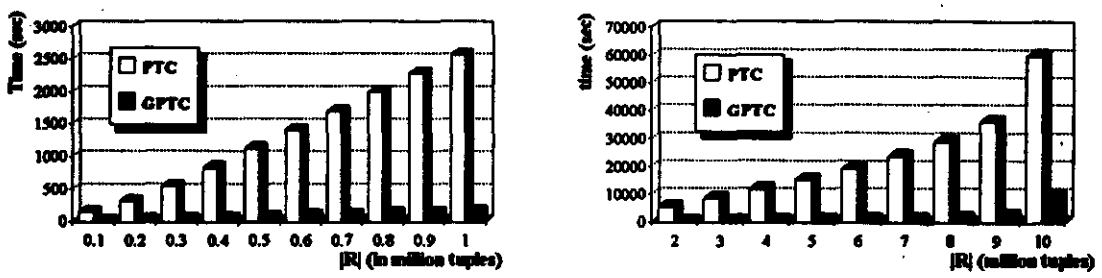

Figure 4: Varying the initial relation size - COMPLETE closure case.

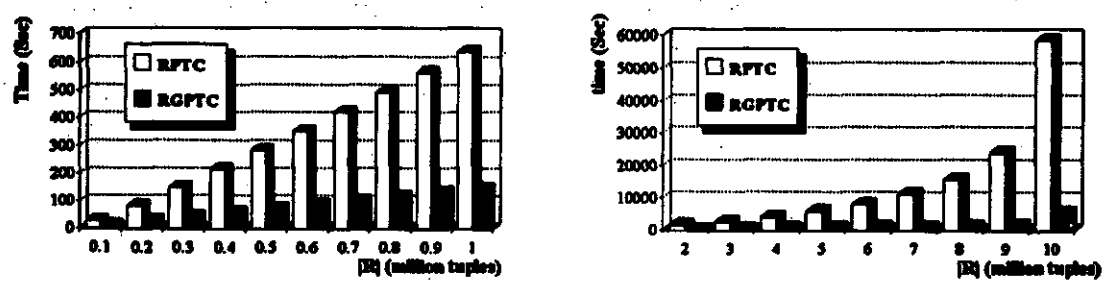

Figure 5: Varying the initial relation size - RESTRICTED closure case.

load balancing ability of GPTC. The way to look at the performance of algorithm GPTC is by looking at its performance first for depth = 1000 and then for depth = 10. Notably, the algorithm is efficient enough at depth = 1000, that it cannot get any better as the depth of the closure decreases to depth = 10.

In Figure 6.(b), algorithm RGPTC uniformly outperforms algorithm RPTC, 4.3 times on the average. The performance differential of the two algorithms is constant due to the fact that we maintain constant join and selection selectivities.

In Figure 7.(a), we observe two things. First, the utilization of all processors by algorithm GPTC makes this algorithm more than 6 times faster than its counterpart algorithm PTC (which utilizes only a portion of the available processors). Second, and more interestingly, we observe that the performance of algorithm GPTC does not improve as more processors are added to the system. At first sight, this is in contradiction with the general claim of high scalability that we have made before during the design of algorithm GPTC. However, the non-scalability phenomenon observed in Figure 7.(a) is due to the fact that algorithm GPTC attains its peak efficiency for a certain number of processors. If more processors become available, and the size of the initial relation R stays fixed, the gains of GPTC due to larger number of processors are out balanced by the communication cost for transferring intermediate result relations. Moreover, note, as the number of processors increases, each processor is assigned smaller amount of work. Figure 7.(a) suggests that the number of processors can become so large that the amount of work assigned per processor is the minimum possible, and thus, addition of more processors does not cause any additional speedup. In this case, the cost of algorithm GPTC is mainly determined by the initialization of processors with the required data.

Figure 7.(b) shows almost the same performance differential as Figure 6.(b). Again, the interesting observation in Figure 7.(b) is that algo-

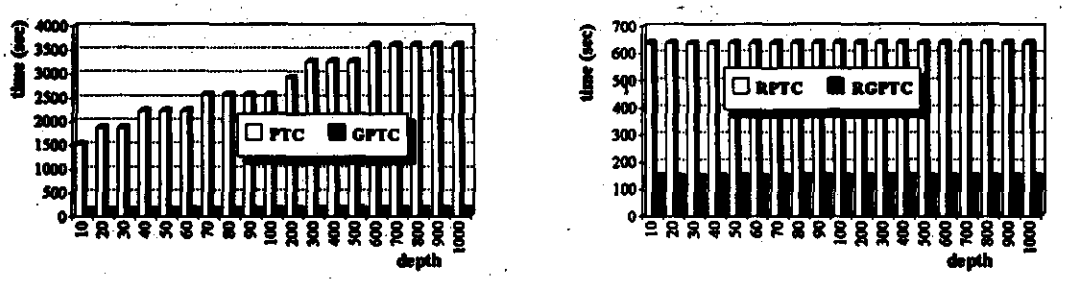

Figure 6: Varying the depth of the transitive closure - (a) COMPLETE closure case; (b) RESTRICTED closure case.

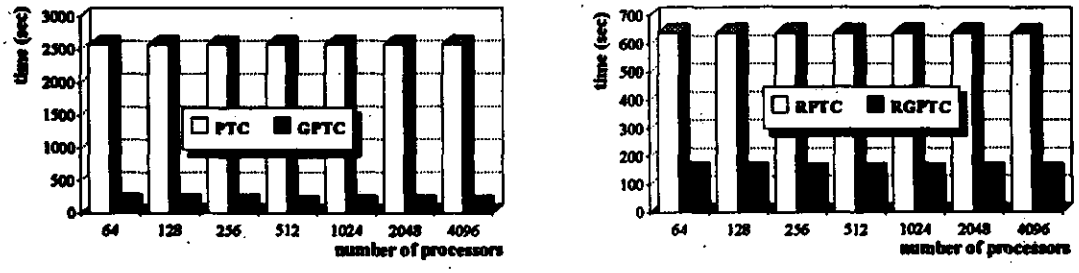

Figure 7: Varying the number of available processors - (a) COMPLETE closure case; (b) RESTRICTED closure case.

rithm RGPTC does not improve its performance as more processors are added to the system. The reasons for this are the same as the ones outlined in the discussion of Figure 7.(a).

Figures 8.(a) and 8.(b) show the effect of join selectivity. In Figure 8.(b), algorithm RGPTC outperforms RPTC about 10 times on the average. Similar observation applies for the complete closure cases. We observe that the worst behavior of algorithm RPTC is for larger values of join selectivity, i.e. when the size of the intermediate result relations is larger. For a join selectivity value of 10^{-5} (not shown here in order to avoid clattering the visibility of the rest of the results), algorithm RGPTC has been found to outperform algorithm RPTC by a factor of 25. In these cases, the same number of processors that must perform joins for smaller relations (i.e. for selectivities 10^{-7} to 10^{-10}), is used to join larger relations. The bottleneck is obviously in the amount of time required for the joins in RPTC. As expected, this is not the case in algorithm RGPTC, where, due to its load balancing and scalability features, large intermediate result relations cause negligible deterioration in its performance.

Figure 9.(a) shows the effect of selection selectivity for the complete closure algorithms. GPTC outperforms PTC about 13 times. The algorithms' performance remains the same for all values of selection selectivity. Note, both GPTC and PTC are complete closure algorithms and, thus, the intermediate result relations are not reduced by any selection operations. Therefore, different values of selection selectivity have no impact on these algorithms.

Figure 9.(b) shows the effect of selection selectivity for the restricted closure case. This parameter also somewhat captures the effect of data skew. Clearly, higher skewed data values imply larger selection selectivities (this is especially true when the skewed value is selected), and, therefore, larger intermediate result relations. Since algorithm RPTC has a preset number of processors available to perform the subsequent joins, its

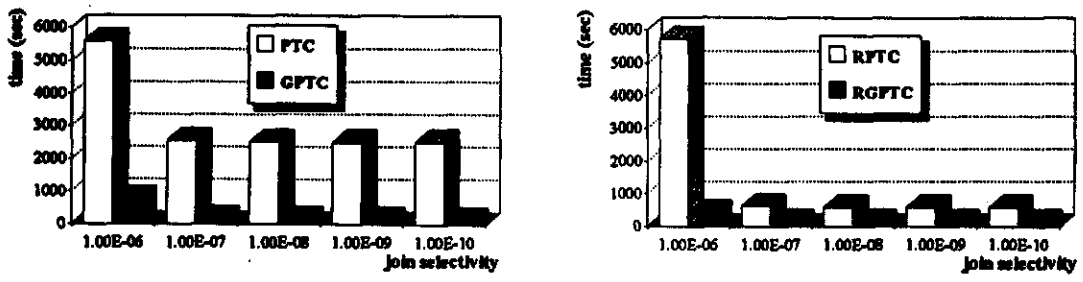

Figure 8: Varying the join selectivity – (a) COMPLETE closure case; (b) RESTRICTED closure case.

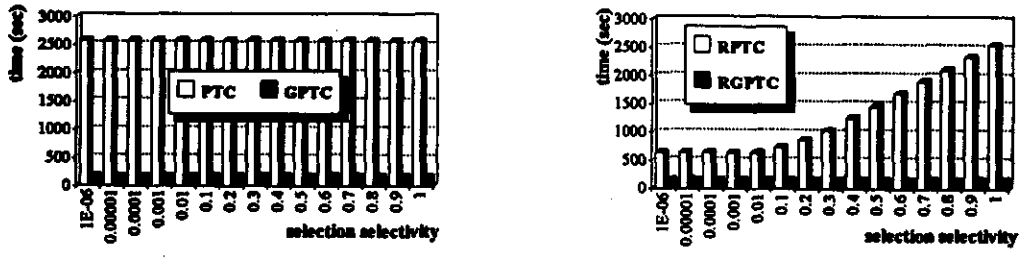

Figure 9: Varying the selection selectivity – (a) COMPLETE closure case; (b) RESTRICTED closure case.

performance deteriorates as the amount of data to be joined increases. This is not the case for algorithm RGPTC, since the number of processors allocated for each join in RGPTC is adjusted dynamically according to the estimated cost of that join.

In Figure 10.(b), algorithm RGPTC outperforms RPTC, 5 times on the average. In the complete closure case (Figure 10.(a)), the performance differential is even steeper. We observe that the performance of algorithm RPTC improves as the memory capacity per processor increases. This is due to the fact that larger memory facilitates more efficient execution of the hash based joins which are performed at the individual nodes. On the other hand, larger memory sizes are indifferent for algorithm RGPTC (a phenomenon that occurred many times in the above figures). This is due to the usage of all processors, algorithm RGPTC assigns small enough data portions at every processor, so that, apparently, all hash based joins can be performed in memory, even with memory sizes of only one megabyte. In this case, additional memory does not make any difference to algorithm RGPTC. Similar comments apply for the performance of algorithm GPTC in Figure 10.(a).

From all the above figures, we draw the following conclusions:

- Algorithms GPTC and RGPTC demonstrate that load balancing and scalability can be achieved in transitive closure operations. The performance of these algorithms especially shows that the considered very expensive complete closure operation is practically feasible for fairly large relations (10 million tuples).
- Load balancing and scalability are important issues in parallel transitive closure algorithms. This is clearly demonstrated by the fact that algorithms GPTC and RGPTC vastly outperform algorithms PTC and RPTC in all Figures.
- Communication cost can become the major bottleneck, regardless of load balancing and

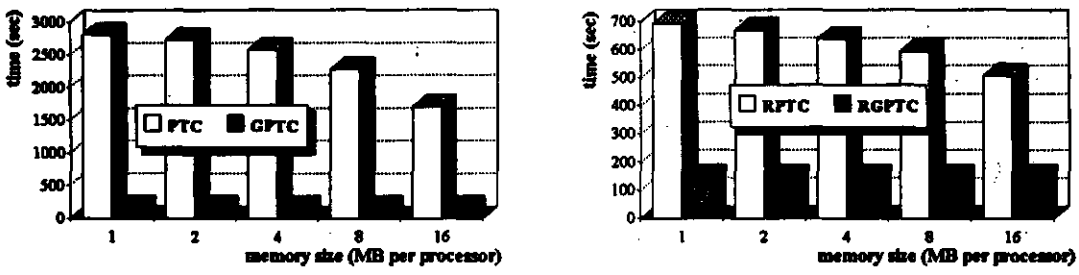

Figure 10: Varying the memory capacity per processor – (a) COMPLETE closure case; (b) RESTRICTED closure case.

scalability ability. This is vividly demonstrated in several occasions where the performance of algorithms GPTC and RGPTC does not improve regardless of the number of allocated processors. This result may, in effect, suggest that the shared-nothing architecture so far typically considered as the de-facto standard for parallel databases, may not be adequate.

- Main memory distribution is an important issue. For example, for a system with 1 GB of main memory, algorithm RPTC will perform better if the system is configured with 128 processors and 8 MB per processor, rather than with 1024 processors and 1 MB per processor.

7 Conclusion

We presented six parallel algorithms that compute the complete and restricted transitive closure of a database relation. Algorithms GPTC and RGPTC exhibit load balancing and scalability properties. An analysis with respect to computation and communication cost was presented for all algorithms, followed by an analytical performance evaluation. The results demonstrated that the scalable and load balancing algorithms consistently outperform the other ones with an average rate of at least 10 times, whereas in some cases the outperfomance fold is in the order of hundreds.

Several issues that are uncovered by the performance evaluation deserve further investigation. The most difficult problem is the reduction of the communication cost which becomes the major bottleneck in algorithms GPTC and RGPTC when the computation cost becomes negligible. We have also seen that the memory distribution is an important factor in algorithms PTC and RPTC. We speculate that in order for these problems to be resolved, the traditional share-nothing parallel computer model need either be augmented with additional features, or a different model must be adopted.

In performing the joins, the hash based join method was assumed. A possible enhancement of the algorithms is to make them more "intelligent" in this respect, that is, to choose a join method dynamically depending of the data that is required to be joined at each iteration. Such an enhancement is worth investigation since, although the hash based join is considered the best general join method, it has been reported that non-hash based join methods may attain better performance for special data formats and distribution.

In our model we assumed a generic distributed memory computer. In practice, distributed memory machines come in well-defined and specific network topologies. Fine tuning of the presented algorithms on specific topologies such as the hypercube will potentially improve the algorithms' performance due to reductions in communication costs that will result by shorter routing distances.

Finally, we consider the full implementation (or simulation) of the algorithms as a very useful project. Such an implementation will allow to carry out experiments for relations with different data distributions and, thus, reveal the effect of skewed data, which resisted our attempts for

incorporation in our analysis.

References

[1] Agrawal, R. and Jagadish, H.V. "Multiprocessor Transitive Closure Algorithms", *Proc. Inter. Symp. on Databases in Parallel and Distributed Systems*, Austin, Texas, 1988, pp. 56–66.

[2] Bancilhon, F. and Ramakrishnan, R. "An Amateur's Introduction to Recursive Query Processing Strategies", *ACM SIGMOD*, 1986, pp. 16–52.

[3] Dewitt, D. J. et. al., "Implementation Techniques for Large Main Memory Database Systems", *ACM SIGMOD*, 1984.

[4] Graefe, Goetz, "Query Evaluation Techniques for Large Databases", *ACM Computing Surveys*, Volume 25, Number 2, 1993, pp. 73–170.

[5] Guh, K.C., and Yu, C.T., "Evaluation of Transitive Closure in Distributed Database Systems", *IEEE Journal on Selected Areas in Communications*, 7, 1989, pp. 399–407.

[6] Heidelberger P., Lakshmi M.S., "A Performance Comparison of Multi-microprocessor and Mainframe Database Architectures", *IEEE Transactions on Software Engineering*, 14, pp. 522–531.

[7] Hua, K.A., Lee, C., "Handling Data Skew in Multiprocessor Database Computers Using Partition Tuning", —, 1990.

[8] Ioannidis, Y.E., "On the Computation of the Transitive Closure of Relational Operators", *12-th VLDB*, Kyoto, Japan, 1986, pp. 403–411.

[9] Jagadish, H.V., Agrawal, R., and Ness, L. "A Study of Transitive Closure as a Recursion Mechanism" *ACM SIGMOD*, May 1987, pp. 331–344.

[10] Keller, A.M., Roy, S., "A Parallel Hash Join Algorithm that Adapts to Data Skew", *SIGMOD Conference*, 1991.

[11] Keller, A.M., Roy, S., "Adaptive Parallel Hash Join in Main-Memory Databases", *First International Conference on Parallel and Distributed Information Systems*, Miami, Florida, 1991, pp. 58–66.

[12] Lakshmi, M. S. and Yu, P. S., "Effect of Skew in Join Performance in Parallel Architectures", *Inter. Symp. on Databases in Parallel and Distributed Systems*, Austin Texas, December 1988.

[13] Lu, H., Mikkilineni, K., and Richrardson, J.P., "Design and Evaluation of Algorithms to Compute the Transitive Closure of a Database Relation", *Proc. IEEE 3-rd Inter. Conf. Data Engineering*, Los Angeles, Feb. 1987, pp. 112–119.

[14] Anestis A. Toptsis, Clement T. Yu, and Peter C. Nelson, "Benchmarking Two Types of Restricted Transitive Closure Algorithms", *Proc. of the 14th Annual Inter. Computer Software and Applications Conference* (IEEE COMPSAC '90), Chicago, Illinois, October 1990, pp. 375–381.

[15] Toptsis, A.A., "Parallel Transitive Closure Computation in Highly Scalable Multiprocessors", *Lecture Notes in Computer Science Series*, Springer-Verlag, No. 497 (1991) : pp. 197–206.

[16] Toptsis, A.A., "Load Balancing in Parallel Hash Join with Data Skew", Proc. of the *First International Conference on Information and Knowledge Management* - CIKM'92, Baltimore, Maryland, November 1992, pp. 248–256.

[17] Toroslu, I.H., and Henschen, L., "An Efficient Transitive Closure Algorithm for Distributed Databases", *Proc. 5th International Conference on Computing and Information* (ICCI '93), Sudbury, Canada, IEEE Computer Society Press, , May 1993 (in press).

[18] Valduriez, P. and Khoshafian, S., "Transitive Closure of Transitively Closed Relations", *Proc. 2-nd Inter. Conf. on Expert Database Systems*, 1988, pp. 177–185.

[19] Valduriez, P. and Boral, H., "Evaluation of Recursive Queries Using Join Indices" *Proc.*

1-st Inter. Conf. on Expert Database Systems, Charleston, 1986, pp. 197–208.

Appendix

Join cost (according to [13]) is the following:

$$\frac{(|R|+|S|)\cdot \mathrm{TS}}{\mathrm{PS}} \cdot t_{\mathrm{read}} +$$

$$(|R|+|S|)\cdot t_{\mathrm{hash}} +$$

$$\frac{(|R|+|S|)\cdot \mathrm{TS} - m}{\mathrm{TS}} \cdot t_{\mathrm{move}} +$$

$$\frac{(|R|+|S|)\cdot \mathrm{TS} - m}{\mathrm{PS}} \cdot t_{\mathrm{write}} +$$

$$\frac{(|R|+|S|)\cdot \mathrm{TS} - m}{\mathrm{PS}} \cdot t_{\mathrm{read}} +$$

$$\frac{(|R|+|S|)\cdot \mathrm{TS} - m}{\mathrm{TS}} \cdot t_{hash} +$$

$$|R|\cdot t_{\mathrm{move}} + |S|\cdot t_{\mathrm{comp}} +$$

$$2\cdot |R|\cdot |S|\cdot \mathrm{JS}\cdot t_{\mathrm{move}} +$$

$$|R|\cdot |S|\cdot \mathrm{JS}\cdot \frac{\mathrm{TS}}{\mathrm{PS}} \cdot t_{\mathrm{write}}$$

Communication cost:

For PTC and RPTC each processor must read the tuples from the intermediate result relation T (unary relation), and send them over the network. There is,

$$\mathrm{RPTC\_Communication\_cost}(T) =$$

$$\frac{T\cdot \mathrm{TS}}{\mathrm{PS}} \cdot t_{\mathrm{read}} + \frac{T\cdot \mathrm{TS}}{\mathrm{BLOCK}} \cdot t_{\mathrm{send}}$$

For GPTC and RGPTC, there is an additional hash cost. Therefore,

$$\mathrm{RGPTC\_Communication\_cost}(T) =$$

$$\frac{T\cdot \mathrm{TS}}{\mathrm{PS}} \cdot t_{\mathrm{read}} + \frac{T\cdot \mathrm{TS}}{\mathrm{BLOCK}} \cdot t_{\mathrm{send}} + T\cdot t_{\mathrm{hash}}$$

In the above, S is the number of tuples from intermediate binary relations, and T is the number of tuples from intermediate unary relations. The meaning of the remaining parameters and their default values are shown below.

- $|R|$: number of tuples in original relation R (default value: 1 million);
- BLOCK: communication block size (default value: 4 KB);
- D: depth of the transitive closure (default value: 100);
- JS: join selectivity (default value: 10^{-7});
- m: available memory per processor (default value: 4 MB);
- PIF: projection improvement factor (default value: 0.5);
- PS: disk page size (for I/O purposes) (default value: 4 KB);
- r: number of processors (default value: 128);
- SS: Selection selectivity (default value: 10^{-3});
- t_{comp}: time to compare two domain values (default value: $3\cdot 10^{-6}$ sec);
- t_{hash}: time to hash a domain value (default value: $9\cdot 10^{-6}$ sec);
- t_{move}: time to move a two-value tuple in memory (default value: $2\cdot 10^{-5}$ sec);
- t_{read}: time to read a page from disk (default value: 0.015 sec);
- TS: tuple size for binary relation (default value: 16 bytes);
- t_{send}: time to transmit one block of data (default value: 0.02 sec); and
- t_{write}: time to write a page to disk (default value: 0.02 sec).

MFM BASED DIAGNOSIS OF TECHNICAL SYSTEMS

Alenka Žnidaršič
Jožef Stefan Institute
Department of Computer Automation and Control
Jamova39, 61111 Ljubljana, Slovenia
Phone: +386 61 1259 199 (int. 606); Fax: +386 61 1219 385
E-mail: alenka.znidarsic@ijs.si

Victor J. Terpstra, Henk B. Verbruggen
Delft University of Technology
Department of Electrical Engineering, Control Laboratory
P.O. Box 5031, 2600 GA Delft, The Netherlands

Keywords: multistrategy learning, advice taking, compilation, operationalization, genetic algorithms

Edited by: Miroslav Kubat
Received: October 8, 1993 **Revised:** January 11, 1994 **Accepted:** February 15, 1994

Detection and diagnosis of faults (FDD) in technical systems represent an important segment of intelligent and fault-tolerant systems. In the article we present the qualitative FDD approach proposed by Larsson and based on Multilevel Flow Modelling representation of the process. The contribution of this article regards evaluation of this method on a simulated water-level process controlled by feedback. The MFM diagnostic expert system, together with the continuous simulation of the process, is implemented in a real-time expert system tool G2. Based on results perspectives for further work will also be given.

1 Introduction

Since many industrial processes become more and more complex fault diagnosis plays an important role in maintenance and on-line monitoring for the purpose of fail-safe plant operation [7]. The techniques of fault detection and diagnosis can be classified to two general categories [4]:

- the mathematical model and
- the knowledge based approaches.

The former make use of the process model, usually in the form of differential equations. The related techniques are based on the concepts of dynamical redundancy and make use of state filtering and parameter estimation.

Since it may often be difficult and time consuming to develop a good mathematical model, knowledge based methods make use of heuristical knowledge derived from human experience and qualitative models. Hereof, two main directions can be recognized [8]: the shallow-knowledge (heuristic) and the deep knowledge (model based) techniques.

In the shallow reasoning approach, diagnostic knowledge is represented mainly in terms of heuristic rules which perform mapping between symptoms and system malfunctions (faults). The rules typically reflect empirical associations derived from experience, rather than a theory of how the device under diagnosis actually works. The shallow diagnostic expert systems have advantages in cases where the expert knowledge in a small field of expertise is available. In this way, we can make this knowledge available to the user. But problems appear in the development of more complex systems, i.e.:

- difficult knowledge acquisition,
- unstructured knowledge requirements,
- knowledge base is highly specialized to the individual process,

- excessive number of rules: difficult to overview in building and updating,
- diagnosability or knowledge - base completeness is not guaranteed: the expert system can diagnose only faults considered in the design of the rule base.

To overcome these disadvantages of the shallow approach the deep knowledge techniques can be used. Rather than assume the existence of an expert experienced in diagnosing the device, we assume the existence of a system description (in qualitative terms): a complete and consistent theory of the correct behaviour of the device [5].

Once the model has been created, it could be used in either of two ways. One way would be to introduce every possible fault, run the model and observe the effects. These observations could be used to create rules linking symptoms to faults. Clearly, this procedure will be feasible in systems with small complexity since the number of possible faults can quickly grow, especially if there is more than one present at the same time. The other way to use a model is to describe how the process is intended to work and failures in the process can be found by noting differences between the intended model and the actual state.

In both cases, the process description (model) and the algorithms for fault detection and diagnosis are separated. It can be said that reasoning about faults is performed on the model. The apparent advantages of such a system are:

- given the device description, the program designer is able to shorten the process of eliciting empirical associations from a human expert,
- the diagnostic reasoning method employed is device independent and
- the ability to reason about unforeseen faults and faults which have never occurred in the process before.

There are several approaches to the qualitative modelling and qualitative reasoning available [3]. In this article we focus our attention on a qualitative FDD approach developed by Larsson [6] and based on the so-called Multilevel Flow Modelling representation (MFM) proposed by Lind (1991). The major part of this article regards evaluation of this method on a simulated water-level process controlled by feedback. Properties of the approach, drawbacks and potentials will be presented and perspectives for further work will be given.

We also try to point out the design cycle of the related MFM diagnostic expert system which consists of the following steps:

- understanding the principles of the process,
- the diagnostic analysis,
- an implementation of the diagnostic expert system using MFM Toolbox,
- testing and validation of the system on the simulated process.

2 Test object: a controlled water column

In the lower part of the setup, there is a large container (see Fig. 1), out of which water is pumped via pipes and valves into the water column. Three valves influence the water flows [1]:

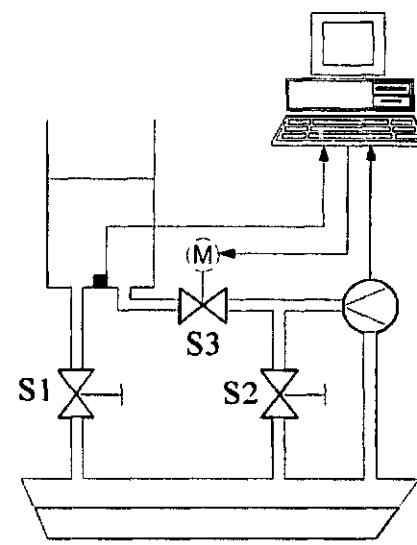

Figure 1: The water-level process

- valve S3 is a control valve controlling the water input to the column. The valve is connected via an electropneumatic transducer to the control computer,

- valve S2 (by-pass valve) is necessary for the pump to prevent an over pressure when S3 is closed,
- valve S1 determines the water outflow from the column.

The water level in the column is measured using a pressure sensor at the bottom of the water column.

3 Review of MFM modelling and diagnostic reasoning

In Multilevel Flow Modelling, a system is modeled as an artifact, i.e. a man-made system constructed with some specific purposes in mind [6].

The three basic types of objects in MFM are:

- goals,
- functions and
- physical components.

The physical components are elements from which a process is constructed (pipes, valves, pumps, etc.). Every component can provide some functions like transport of mass, information or energy, storage of something. A set of interconnected functions serve to realize some goal. Goals in MFM represent what the process should do e.g.: keep the water at the certain level.

Goals, functions and components can be connected with achieve (achieve - by - control), condition and realize relations. An achieve (achieve-by - control) relation can be used to relate a set of flow functions to the corresponding goal. For example, a network of flow functions which describes the water flow through the process (see Fig. 2) is connected to the main goal by an achieve-to-control relation. It means, that the main goal can be achieved by controlling the water flow. Some physical component can provide functions only if some goals are fulfilled first, e.g. the water can be pumped from the container only if the pump works properly. A condition relation is used to connect those conditioned goals to a function. On the lowest level in the MFM graph physical components are in the realize relation with their corresponding functions, e.g. the pump can perform the function of transporting water and it is connected with a realize relation with it.

The MFM model describes how the process is intended to work by using mass or energy balance equations. Every deviation from the balance equation can be a sign that the flow function has error and corresponding alarm states are set-up for it. For describing the mass, energy or information flows of the process, several function types are available. Nearly all flow functions are characterized by one (or more) flow value, which correspond to the real flow of mass or energy.

Based on the MFM graphs, three types of diagnostic methods have been proposed [6]:

- the measurement validation method,
- the alarm analysis method and
- the diagnosis method.

The main aim of the measurement validation algorithm is to find out whether there are inconsistencies among flow values (measurements) in the MFM model. Using available redundancy on the set of measured flow values the MFM model can be divided into internally consistent subgroups. If a flow function with one inconsistent value is discovered, it will be marked and corrected. In case of several conflicting values, the consistent subgroups of measurements will be marked but the flow values will not be corrected. The analysis of an alarm situation can be performed using the alarm analysis algorithm. Every flow function can be performed correctly or not. Its failure state can be defined with one of the following alarms: high flow, low flow, high volume, low volume, leak, etc. The algorithm provides a decision about which of the alarms are directly connected to the faults (primary alarms) and which ones are set up only as a consequence of the primary ones (secondary alarms).

In the terminology of MFM, when one of the goals from the model fails, the fault in the process occurs. The fault diagnosis algorithm provides an explanation for malfunctioning. It is implemented as a search in the MFM graph from the failed goal to the connected networks of functions. When it reaches a single flow function, it uses questions answered by the operator, results of tests performed on measurements or fault propagation rules to find out its failure state. Based on information about states of the flow functions the explanation about a failure situation and remedies are given.

The proposed methods based on MFM are not aimed for diagnosing sensor faults.

4 MFM model of the water-level process

The MFM model of a process is shown in Fig. 2.

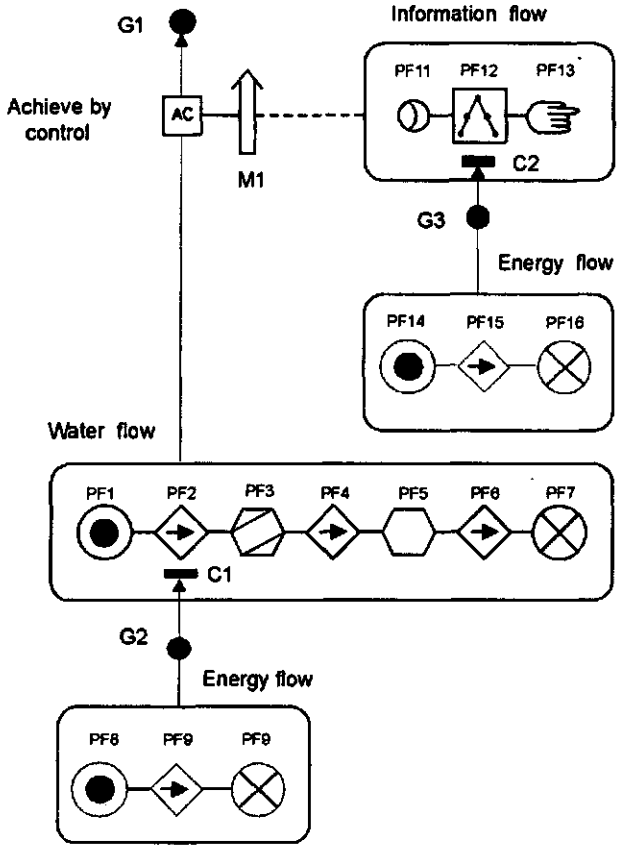

Figure 2: The MFM model of the process

The main goal (G1) is: "Keep the level of the water in the water column at the determined position."

The topmost goal can be achieved by a network of mass flow functions (a water flow). This flow is controlled by a manager function (M1), in this case a PI controller (PF12) acting on the control valve.

The primary flow circuit starts at the source PF1 (water container), continues through the transport function PF2 (pressure source), a balance function PF3, a transport function PF4 (control valve) into the storage PF5 (water column) and through the transport function PF6 (manual valve) back to the sink PF7 (water container).

The water can be pumped from the container if the pressure source works properly. In the MFM model, the transport function PF2 (pump the water) is conditioned (C1) by the subgoal G2: "Keep the pump running." If the subgoal is fulfilled then the transport function is available. An electrical energy needed for the pump running is described as an energy flow from a source PF8 (power supply), via transport PF9, power switch, and to the sink PF10, motor of the pump.

The implementation of the control task (M1) is described as an information flow circuit. Measurements of the water level in the column are provided using an observer function PF11 (sensor). The decision about control action is made by PI control algorithm (PF12) and control output is proceeded to the control valve through the actor function (PF13). The controller works, if the electrical energy is provided for it. Therefore, the subgoal for the controller is: "Keep the controller running." (G3). It is also achieved by a network of flow functions describing energy flow from the source PF14 (power supply), via transport PF15, power switch, and to the sink PF16, the controller.

Some functions are directly connected to the physical components which provide the functions working: PF1 to the water container, PF2 to the pressure source, PF4 to the control valve, PF5 to the water column, PF6 to the manual valve and PF7 to the water container, etc. In the water flow circuit there is also additional balance function (PF3). It is not connected to any of the physical components, but it has to be present because of the syntax reasons.

The MFM model is simplified representation of the real process. The simplification depends on the purpose of the model. We have to be aware that a diagnostic system using a simplified model can not recognize faults in the unmodeled parts of the system.

5 Realization of the diagnostic expert system in G2

G2 is a real-time expert system tool developed at Gensym Corporation. It can be seen as a general programming environment that combines three

paradigms:

- rule - base inference,
- object - oriented programming and
- procedural programming.

It also has a very strong graphical orientation. It consists of several main parts: a knowledge database, a real-time inference engine, a procedure language interpreter, a simulator, a development environment, an operator interface and optional interfaces to external on-line data service [10].

As a support for developing an expert-system for diagnosis based on the MFM methodology, an MFM Toolbox has been developed in G2 [6]. It has two parts:

- a module for developing an MFM model of a process (definition of data structures and graphic elements for building MFM graph),
- a module with a rule base that perform diagnostic reasoning task. Several groups of rules and procedures were implemented: a rule base for syntax control of an MFM models, measurement validation, alarm analysis, consequence propagation and fault diagnosis.

The Toolbox has ben developed by Larsson as part of his thesis work and made available to the Control Laboratory at Delft as part of a mutual research exchange between the Lund and Delft Control Laboratories.

By using the MFM Toolbox it is possible to develop an expert system, which performs diagnostic reasoning for the specific process, in our case for the water-level process. It is assumed that the algorithms for diagnosis are independent of the process description. Therefore, the developer of the expert system needs only to construct an MFM model for his process using the Toolbox.

The MFM graph structure is defined graphically using graphical objects for MFM functions and connections among them. The graphical representation of the MFM model for the water level process is based on the MFM model description (Fig. 2). The construction of the MFM model uses G2's possibilities of graphical creation, cloning and editing of those predefined objects.

In order to enable a diagnostic reasoning, also the values for attributes of flow functions have to be prescribed:

- with a set of rules that transfer the values from a simulated process to the corresponding flow function (on - line) or,
- the user defines the values for each flow function from the model directly using editing of graphical objects (off-line).

As soon as the MFM graph structure is defined, together with the corresponding values for flow function attributes, diagnostic questions and remedies, the diagnostic algorithms are ready to be used.

6 Continuous process simulation

When we talk about purposes of the simulation model, we have to mention the definition of a simulation environment. We must take into account that the main aim of our simulation model is diagnostic system testing. It should be possible to simulate different failure behaviours and to provide data from observable variables. The simulation environment (Fig.3) consists of three different and independent modules inside G2:

- a simulated process module,
- an alarm definition module and
- a fault module.

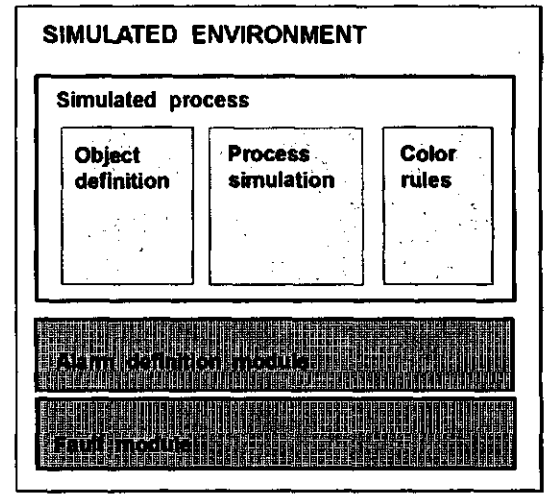

Figure 3: The process simulation environment

The "Simulated process" module represents the behaviour of the water-level process under normal

working conditions in a closed control loop. A physical-structural model, which includes the important physical structure of the system, is used. The process is described in terms of their components and relations that exist between them.

Each component of the system is modelled separately as an object inside G2 with its own behaviour and attributes. The way in which a component behaves, is described by physical equations. Using the components behaviour description, the operation of the whole system is generated by analysing how the components are connected and how they interact within the system.

Measurements of process variables constitute the basic information for diagnosis. The diagnostic methods developed on the MFM models need as an input a set of measured flow signals. But measurements do not always relate directly to the level of process representation (the flow values of the MFM functions), therefore also other types of information must be used. They can be obtained with one of the following methods: sensors, estimation using data transformation (parameter or state estimation methods, statistical methods) or evaluation based on human observations.

Independently from a simulated process module, the alarm definition module has been developed. A procedure for each modelled component have been defined, which prescribes the way for obtaining the data from its observable variables. From the reason, that on the real process only one sensor is available, it is possible to introduce new sensors in the simulation. We refer those sensors as "simulated sensors".

The alarm definition module performs also a detection function. Rules with "crisp alarm limits" - a fixed value where each alarm condition is activated for every modelled physical component - have been used.

The simulated process has been used to test how efficiently the diagnostic system can recognize the possible causes for its malfunctioning. From this point of view it is possible to simulate different failure behaviours of the process.

All possible faults on the physical components are known from the diagnostic analysis of the process. The prescription of boundaries on observable variables, when the components are treated as faulty, derived from experimentation with a process, process simulation and students experiences working with it. For every possible fault a procedure, which introduces this fault into the corresponding physical component, is implemented in the "Fault module". For some faults, it is also possible to define how big a fault is. With activating the procedure the corresponding fault is injected in the simulated process.

7 Experimental results

In order to evaluate the diagnostic system based on MFM for the water-level process (Fig.4), a series of experiments is performed using the simulated process running in parallel with the MFM diagnostic expert system inside G2 [9].

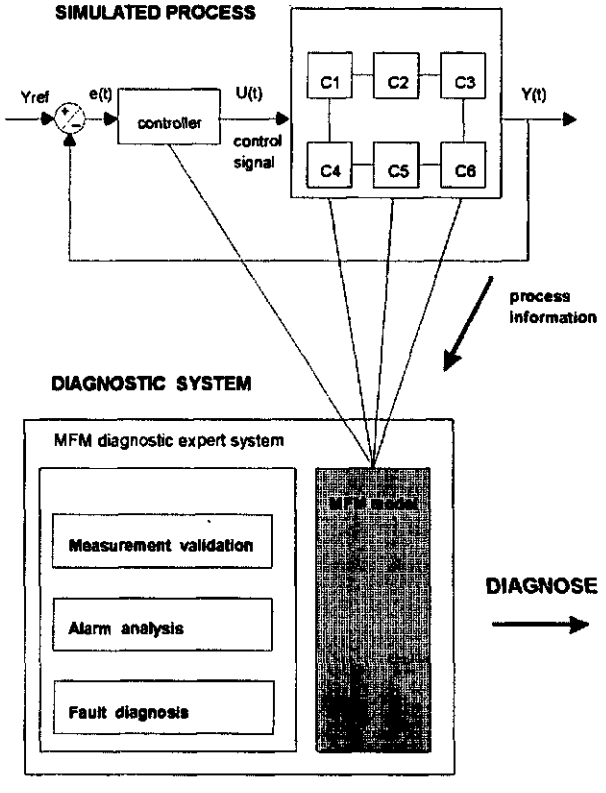

Figure 4: Evaluation of the MFM diagnostic system

The following assumptions have been made:

- all possible faults on the components are known and modelled,
- every physical component may or may not be connected to the real or simulated sensor,
- sensors are functioning correctly and

– process operates in the steady state.

Every experiment consists of the following steps:

1. A set of modeled physical components which are connected with the simulated sensors must be defined before the simulation starts,

2. The process simulation is started with defining the reference value for the water level in the column. Wait until the process is in a steady - state.

3. One single fault or a combination of faults is introduced in the simulated water-level process,

4. MFM diagnostic expert system, which runs in parallel with the simulated process, is used to diagnose the malfunctioning behaviour of the simulated process.

5. The analysis of the diagnostic results is made by comparison of the diagnostic explanation of the diagnostic system and our assumption about possible causes for malfunctioning.

In order to illustrate how the diagnostic expert system responds to the situation in the process let us take the case where a fault is injected into the water container (leak). The measurable (observable) quantities are the water quantity in the container, the flow through the control valve and the water level in the column.

When the main goal was violated (water level is not at the reference value), the diagnostic system starts searching for faults in the connected water flow circuit. The simulated process produces the following symptoms: not enough water in the water container, the flow through the control valve is too low and the water level in the column is below the desired reference value.

Information about the symptoms is transferred to the MFM model as a set of alarms (Fig. 5): the LOCAP on the source function PF1, the low flow (LOFLOW) on the transfer function PF4 and the low volume (LOVOL) on the storage function. The alarm propagation algorithm guesses the alarm states low flow (LOFLOW) for the transport functions PF2 and PF6.

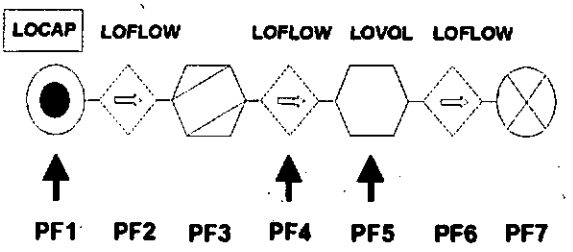

Figure 5: An alarm simulation in the MFM model

Explanation symbols:
arrows show detected/measured symptoms

Based on the described alarm situation, the diagnostic algorithm concludes that only one primary failed function exists (PF1) and that all the others are only consequences of it. The cause for the malfunctioning can be assumption that the fault is present on the water container and the diagnosis is: "The water container is leaking."

In this case the alarm on the transport function PF2 is only consequence of the fault on the source PF1. If we assume, that the operator notices, that the pump is not running because there is no power supply for it, the alarm LOFLOW is set up for the transport function PF9. Based on this additional information, the alarm propagation algorithm can guess, that also the transport function has an alarm LOFLOW, which is then primary-failed. The diagnostic system can find two different faults in the process: "The water container is leaking." and "The fault on the power supply for the pump".

The overview of experimental results is given in a table (Table 1).

8 Discussion

The MFM diagnostic expert system results in only one possible explanation for the process malfunctioning. It does not provide suggestions about all possible faults in the system. Diagnostic precision with which a fault can be identified depends on the number of measurements available from the physical components. If there exists only one sensor in the process, the diagnose is correct only in the case when the physical component attached to this sensor failed. If the number

of measured flow functions in the MFM model increases then the diagnostic accuracy increases as well. Multiple faults can be diagnosed when the alarm situation can not be explained with only one fault.

Another problem encountered in our experiments is referred to the balance function. Even though, theoretically, the fault can propagate through it, the corresponding rules are not included in the MFM Toolbox. Also the balance functions with more than one input or output can not be used.

The treatment of the alarms on the flow function depends on the time interval in which they are transferred to the MFM model. As soon as the symptom in the simulated process is recognised, the corresponding alarm is transferred to the MFM model. The alarm on the flow function should be assigned as a primary failed. Later on another new alarm is discovered in the model. In this case, the fault propagation algorithm guesses also the failure state of the first one only as a consequence of this new alarm. Because the primary failed flow function is covered with a secondary failed as a result of propagation rules, some information about the faults can be lost. The problem is referred as a "loss of diagnostic discrimination". When this problem appears, the diagnosis of malfunctioning is correct but it is not complete.

Furthermore, the concept of goals in the MFM syntax is questionable in case of feedback systems, e.g. if leak on the column is not big, the goal G1 will be maintained by feedback (the controller will force the pump to provide more water). Malfunctioning can be recognized from the control voltage changes, but the diagnostic system does not use this additional information. It will react too late. The MFM diagnostic expert system can be used as an independent system. If a human operator recognizes a process malfunctioning he starts the diagnostic system with defining the goal, which is failed. As an independent module it can be integrated in a supervisory system which performs monitoring of the process. In this case the diagnose is started automatically as a request from the supervisory system without human intervention. Concerning that a system which performs process diagnosis as a combination of automatic tests on measurements and human judgements about the observable states (where measuring is difficult or expensive) might be a solution. It can be a valuable support for the human operator for decision making in managing with fault diagnosis. We can add the diagnostic questions for every flow function. The operator has to concentrate on every physical component systematically and give an answer using observations and experience.

9 Proposed improvements

Based on the results of MFM diagnostic expert system evaluation, the following proposals how to enhance the MFM approach (and also Toolbox) are given:

1. Make the goals of the MFM model active

In the implemented MFM Toolbox, goals of the MFM model do not have immediate use, except as starting points for the diagnostic search and as a connection point between the different layers of functions networks. From this point of view they can be seen as "passive objects". We propose to make goals "active" by defining the list of goal constraints. The constraints can be given as an analytical equations, qualitative equations or heuristical rules.

2. Add "time attribute" to the flow functions

The "loss of diagnostic resolution" problem may be reduced by including the time interval, when the alarms were transferred in the MFM model in the reasoning about faults.

3. Dynamic fault diagnosis

To overcome the disadvantage of being able to diagnose only static problems, a dynamic fault diagnosis by analysing subsequent snapshots is proposed. The process behaviour can be extracted using a pattern recognition approach on the measurement vector. The following process features can be observed using qualitative values: the output response time delay, the curve peaks and time interval between peaks. From those features, middle facts like damping, overshoot, oscillation can be inferred. Various decision rules can be applied to the feature vector to classify the process behaviour into different classes.

4. MFM supervisory layer

Managing with a developed MFM diagnostic system is a human task. It can be started as a request of human operator by indicating a failed

goal. To provide a system more independent, a supervisory system can be developed on the top of the existing MFM diagnostic system. The main tasks of supervisory layer are concerned with:

- analysis of process behaviour,
- testing of goals requirements,
- fault detection in dynamic states and
- activating the diagnose process.

In addition, it can perform communication with the user in form of reports about the process behaviour or demands for additional information from the operator.

10 Conclusions

The diagnostic reasoning based on the Multilevel Flow Models (MFM) is an example of a deep reasoning approach. MFM provides a way of qualitative description of goals, functions and physical components of the process. Because managing with faults concerns also a lot of reasoning about goals, functions and components, the MFM representation of the process can be very suitable for solving diagnostic problems.

The major contribution of the paper regards evaluation of the MFM diagnostic system implemented using MFM Toolbox for a water-level process. For the testing purposes the simulated environment has been developed inside G2 with three independent modules: the simulated process in the closed-loop under normal conditions, the alarm definition module and the fault module which can simulate different types of faults in the process. The proposed diagnostic methods are not aimed for diagnosing sensor faults. The diagnostic experiments have been performed with running a simulated process in parallel with the MFM diagnostic system, which provided diagnostic explanation. The system can diagnose faults in the system correctly if there is enough measured information (sensors) available.

Some diagnostic mistakes are caused because of the balance function, which is not included in the fault propagation rules. In case of multiple faults, problems occur concerning the "loss of the diagnostic resolution".

Furthermore, concept of goals in the MFM syntax is questionable in case of feedback systems. A small fault can be compensated with the controller and reaction of the diagnostic system will be too late.

In order to provide a diagnosis in time a portion of quantitative knowledge should be included in the FDD system, which is a subject for further research.

Acknowledgement

The first author wishes to acknowledge financial support for this research from KFA Jülich and TEMPUS Office Brussels. Authors also acknowledge dr. Larsson from Control laboratory Lund for providing them with software support for MFM methodology.

References

[1] BUTLER, H. (1990): "Model reference Adaptive Control: Bridging the gap between theory and practice." Doctor's thesis: Delft University of Technology, Department of Electrical Engineering (Control Laboratory) Delft, The Netherlands

[2] HIMMELBLAU, D.M. (1987): "Fault Detection and Diagnosis in Chemical and Petrochemical Processes." Elsevier Scientific Publishing Company Amsterdam - Oxford - New York.

[3] HUNT, J.E., M.H.LEE and C.J.PRICE (1992): "An Introduction to Qualitative Model-Based Reasoning." Proceedings of the IFAC/ IFIP/ IMACS International Symposium on Artificial Intelligence in Real-Time Control, Delft University of Technology, Delft, The Netherlands, pp.439 - 454.

[4] ISSERMANN, R. (1984): Process fault detection based on modelling and estimation methods: A survey. Automatica, 20, 387 - 404.

[5] JACKSON, P. (1990):"Introduction to expert systems" (second edition). Addison - Wesley Publishing Company England.

[6] LARSSON,J.E.(1992):"Knowledge - Based Methods for Control Systems". Doctor's the-

	INJECTED FAULTS							"OBSERVABLE" QUANTITIES							DIAGNOSED FAULTS					
	water container	pump	pump	control valve	water column	manual valve	controller	water container	pump	pump	control valve	controller	water column	manual valve	water container	pump	pump	controller	water column	manual valve
			power supply				power supply	water quantity	pressure	power switch (on/off)	water flow	power switch	water level	water flow			power supply	power supply		
1	●												●						●	
					●								●						●	
2											●		●							
					●						●		●							
3	●								●	X	●		●			●				
			●						●	X	●		●				●			
		●			●				●		●		●			●				
			●		●				●	X	●		●				●			
4			●						●				●			●			●	
		●			●				●				●			●			●	
5	●							●			●		●		●					
	●				●			●			●		●		●					
	●		●					●		X	●		●		●		●			
6					●			●	●				●						●	
7				●																
8						●							●	●						●
9				●							●	X	●							
10							●				●	X	●					●		

Table 1: The experimental evaluation of the MFM diagnostic expert system

sis: Department of Automatic Control, Lund Institute of Technology, Lund.

[7] PATTON, R. (1993) Robustness issues in fault-tolerant control. Prepr. Int. Conference on Fault Diagnosis TOOLDIAG93, Toulouse, suppl. vol 3.

[8] TZAFESTAS, S.G. (1989): "System fault diagnosis using the knowledge-based methodology". Chapter 15 from: R.Patton et al.(Ed)Fault Diagnosis in Dynamic Systems:Theory and Applications. Prentice Hall, London. pp. 509 -594.

[9] ŽNIDARŠIČ A. (1993): Model-based diagnosis using MFM models for a water-level process. Working Report, Delft University of Technology, Department of Electrical Engineering (Control Laboratory) Delft, The Netherlands.

[10] G2 Manual (1989), GenSym Co.

ADAPTIVE FILE ALLOCATION IN DISTRIBUTED INFORMATION SYSTEMS

A. Mahmood, H. U. Khan and H. A. Fatmi
Dept. of Electronic and Electrical Engineering
King's College London
The Strand, London WC2R 2LS
U.K.
a.mahmood@bay.cc.kcl.ac.uk

Keywords: File allocation, distributed data management, computer networks, heuristics

Edited by: Miroslav Kubat

Received: September 15,1993 **Revised:** March 14, 1994 **Accepted:** April 1, 1994

The problem of adaptive data allocation in distributed information systems to minimize the overall communication and storage costs is discussed. A key problem of economical estimation of future file utilization pattern is described and an algorithm based on the Gabor-Kolmogorov learning process is presented to estimate the future access and the update patterns. An adaptive algorithm to reallocate data in a computer network is proposed. The reallocation algorithm chooses the optimal allocation using two objective functions and the best fit strategy. Also, a distributed candidate selection algorithm is presented to reduce the number of files and nodes in the reallocation process. The simulation results are presented to demonstrate the accuracy and the efficiency of the proposed algorithms.

1 Introduction

In a distributed database system, files are accessed either for update or for retrieval activities by geographically dispersed users and applications. An important issue in the design of the distributed database systems is determining the file replication level and the allocation of replicated file copies to the participating nodes to achieve satisfactory system performance. This problem is usually known as File Allocation Problem (FAP) and has been proven to be NP-complete [1].

The optimal allocation of files in a computer network is determined on the basis of the anticipated access and update frequencies at each node, the system's limitations, and the designer's requirements. As the file utilization pattern is not static, the resources at the participating nodes and network topology may change, so even a well designed system may become obsolete and suboptimal. Therefore, adaptive data allocation, which involves repeatedly adjusting the file allocation through possible addition and deletion of file copies with the changes in file utilization pattern, is a desirable activity in the distributed databases.

Dynamic FAP was addressed for the first time by Segall [2] in which reallocations of a single copy of a file were allowed at any point in time. Employing the dynamic programming techniques, he modelled the static as well as the stochastic problem as continuous and discrete time Markov decision models. Levin and Morgan [3] extended the static one-period file reallocation model of Segall and Sandell [4] to a multi-period model for file reallocation that takes place at the beginning of each time period. An adaptive file allocation model for star network was presented by Yu et al. [5]. Two algorithms to dynamically reconfigure the distributed database were proposed by Du and Maryanski [6]. The performance of their algorithms depends on the order in which the files are allocated in a computer network with limited resources. Ames [7] proposed dedicated sensors to monitor the file utilization pattern and the reallocation is considered only if the overhead of moving the files is less than the benefit expected.

Pocar [8] proposed a different strategy to deal with single file copy migration problem. He used the Semi-Markov decision model to obtain single-copy migration polices. Hać [9] presented a distributed algorithm to improve the system performance through replication and migration of files and processes. Some of the other work on adaptive file allocation includes [10, 11, 12, 13] and detailed surveys on FAP can be found in [11, 14].

In this paper, an adaptive algorithm to reallocate data in a distributed information system is presented. We are particularly interested in one-period look ahead incremental allocation, i.e. redistributing data in the existing distributed system for the next time period.

The rest of the paper is organized as follows. In section 2, the distributed database monitoring is discussed and an algorithm to predict the future access and update frequencies is given. A distributed algorithm to reduce the solution space is presented in section 3. The adaptive file allocation model and the file reallocation algorithm are presented in section 4. The simulation results are given is section 5 and the paper is concluded in section 6.

2 Estimation of Access and Update Pattern

In most of the cases, the file utilization pattern over the network is not predictable for a long period of time and, therefore, we have to monitor the system behaviour dynamically to determine whether the current file allocation is still optimal. The monitoring of the relevant components and events can be accomplished either through hardware, software or through a combination of both [15, 16, 17, 18].

In the adaptive file allocation activity, the file usage pattern should be monitored to collect the access and update frequencies at each node. In the proposed distributed monitoring system, each node collects two types of statistics: 1) local access and update frequencies for remote files; 2) alien access and update frequencies for local files. The monitoring system also gathers information about the changes in the computer and the network resources.

The statistics collected by the monitoring system can be used to estimate the future access and update frequencies. A good estimation module should be able to predict the future access/update pattern not purely on the basis of current file usage pattern but also on the past history of the access/update pattern. Moreover, it should not be vulnerable to chance fluctuations in usage [19].

The proposed estimation method is based on the Gabor-Kolmogorov Learning Process (GKLP). The GKLP can be described by a stochastic operator upon a function of time [20, 21].

$$O[f(t)] = \sum a_i x_i + \sum\sum b_{ij} x_i x_j + \ldots + \sum \ldots \sum z_{i1\ldots in} x_{i1} \ldots x_{in} \quad (1)$$

Where the first term is a generalized linear predicting filter. The valves of the coefficients $a_i, b_{ij} \ldots$ can be adjusted by minimizing the mean square error between the predicted and the actual values at time $t = 0$, $t = 1$, $t = 2$ and so on. If x_t and y_t are the actual and the corresponding predicted access/update frequencies at time t respectively, then the error E can be defined as:

$$E[e(t)^2] = E[\{x_t - y_t\}^2] \quad (2)$$

where

$$y_t = \sum_{i=0}^{i=t-1} c_i x_i \quad (3)$$

The optimal values of coefficients c_i's are calculated one by one by GKLP, equating them to -1, 0, and +1 and the respective mean square error to E_{-1}, E_0 and E_{+1}. The coefficient c_i is described as:

$$c_i = \frac{E_{-1} - E_{+1}}{2(E_{-1} + E_{+1} + 2E_0)} \quad (4)$$

where

$$E_{-1} = \sum_{i=0}^{i=t-1} [(e_i + x_i)^2] \quad (5)$$

$$E_0 = \sum_{i=0}^{i=t-1} [(e_i)^2] \quad (6)$$

$$E_{+1} = \sum_{i=0}^{i=t-1} [(e_i - x_i)^2] \quad (7)$$

e_i is the partial error at ith iteration and is defined as:

$$e_i = (c_i x_i + x_t - y_t) \quad (8)$$

After adjusting the values of the coefficients, the access and the update frequencies for each node-file combination can be predicted for the time period $t+1$. The coefficients need not be recalculated each time the access and update frequencies are to be estimated. Rather, they should only be recalculated if the error in the predicted and the actual values for the last time period is out of some acceptable error range.

Thus, the information to be stored and transferred from time period t to $t+1$ is the traces of the past history and the values of the coefficients for each node-file combination. As the traces can be collected and processed at each node, the monitoring module should reside on each node. The advantages of such an approach are two fold: the estimation algorithm can be executed in parallel and the volume of data to be transmitted is reduced. Only the revised estimates of the access and update frequencies have to be transmitted to the central node.

3 Candidate Selection

The exponential growth of the computation time with the increase in number of nodes and file means that the problem becomes intractable even for a moderate size network. Therefore, heuristics to reduce the solution space by eliminating certain sites and files from the reallocation process play an important role in the development of fast data allocation algorithms [22]

Based on the theorems given in [13], the following algorithm can be used to find the files which should be considered for reallocation. The notations used in the following sections are given at the end of the paper.

Algorithm Candidate Selection

Step 1:

At each node n DO
  Initialize Df_n and Af_n
  FOR each file $i \notin I_n$ DO
    IF $(a_{ni}V_{ni} \min_{k \in K_i} C_{kn} >$
      $(L_i S_n + \sum_k^N u_{ki}U_{ki}C_{kn}))$ AND
      (copy of i is allowed at n) THEN
        $Af_n \leftarrow Af_n + \{i\}$
  FOR each $i \in I_n$ DO
    IF $((L_i S_n + \sum_k^N u_{ki}U_{ki}C_{kn}) >$
      $\sum_k^N a_{ki}(\min_{l \in K_i} C_{lk} -$
      $\min_{k \in K_i - \{n\}} C_{lk})V_{ki}))$ AND
      (Deletion of i from n is allowed) THEN
        $Df_n \leftarrow Df_n + \{i\}$

Step 2:

Send $Af_{n's}$ and $Df_{n's}$ to a central node

Step 3:

At the selected central node DO
  FOR n= 1 to N DO
  FOR i= 1 to F DO
    IF $i \in Af_n$ THEN $A_i \leftarrow A_i + \{n\}$
    IF $i \in Df_n$ THEN $D_i \leftarrow D_i + \{n\}$
  FOR i= 1 to F DO
    IF $A_i \neq \phi$ THEN $A_f \leftarrow A_f + \{i\}$
    IF $D_i \neq \phi$ THEN $D_f \leftarrow D_f + \{i\}$

In the above algorithm, step 1 is executed in fully distributed fashion. It states that that if the cost of having a local copy of file i is less than the cost of accessing it from the remote site with the minimum cost, then it is beneficial to have a local copy. Also, if the maximum expected benefit of keeping a copy of i at a site n is less than the extra storage and update costs then it is not optimal to have a file copy at n.

The step 1 also uses the fact that certain constraints such as data security, hardware and software compatability etc. can be applied at the earlier stage of data reallocation. These constraints can be expressed logically by a set of allocation matrices, where each entry shows whether a file can be allocated at a node [24]. The constraints specified in the allocation matrices reduce the number of candidate sites for reallocation of a file.

In step 2, all $Af_{n's}$ and $Df_{n's}$ are sent to a central node where the reallocation algorithm to be executed. This node can be selected arbitrarily. In step 3, all the candidate nodes and files are determined at the selected central node.

4 Assumptions and Objective Functions

We assume that the distributed system consists of N nodes and F files. The network is fully connected and each node has its own database management system capable of processing its local as well as remote transactions. Each node main-

tains the accounting information as discussed before, and the reallocation is initiated at fixed time intervals. At the end of each time interval, future access and update frequencies for each node-file combination are predicted and the algorithm *Candidate Selection* is executed.

Moreover, instead of generating a sequence of allocation for the entire database lifetime, the new assignment is generated only for the next time period, assuming that the optimal allocation for each time period will result in an optimal allocation for the multiple periods. It is also assumed that the files are accessed independently.

4.1 The Objective Functions

Let K_i be the set of nodes with a copy of file i. Consider adding a copy of file i at node n. The cost of querying the file i will reduce by the amount

$$R(K_i) - R(K_i') \tag{9}$$

Where

$$R(K_i) = \sum_{m=1}^{N} a_{mi} V_{mi} \min_{k \in K_i} C_{km} \tag{10}$$

$$R(J_i') = \sum_{m=1}^{N} a_{mi} V_{mi} \min_{k \in K_i'} C_{km} \tag{11}$$

$$K_i' = K_i + \{n\} \tag{12}$$

In other words, allocating file i at node n will achieve a maximum *gain* given by (9). On the other hand, allocating i at n will incur extra update cost (CU_i), storage cost (CS_i), and the cost to transfer a copy of file i from a node with minimum cost (T_i) given by (13), (14) and (15) respectively.

$$CU_i = \sum_{k \neq n} u_{ki} U_{ki} Ckn \tag{13}$$

$$CS_i = S_n L_i \tag{14}$$

$$T_i = L_i \min_{K \in J_i} C_{kn} \tag{15}$$

$CU_i + CS_i + T_i$ is defined as the *loss* of allocating file i at site n. The objective is to maximize the benefit, i.e. the difference between the gain and the loss of allocating file i at n. Therefore, the objective to add a copy is to maximize (16) subject to the storage (17), the channel (duplex) (18) & (19) and the processing (20) constraints.

$$Z_{ni}^{a} = R(J_i) - R(J_i') - CU_i - CS_i - T_i \tag{16}$$

Subject to

$$\forall \quad k \quad \sum_{x_{ik}=1} L_i \leq M_k \tag{17}$$

$$\forall \quad k \neq n \quad \sum_{j} (a_{kj} y_{knj} V_{kj} + U_{nj} x_{jk} u_{nj}) \leq \lambda_{nk} \tag{18}$$

$$\forall \quad k \neq n \quad \sum_{j} (a_{kj} y_{nkj} V_{nj} + U_{kj} x_{nj} u_{kj}) \leq \lambda_{kn} \tag{19}$$

$$\forall \quad k \quad \sum_{l} \sum_{x_{ik}=1} p_{klai} a_{li} Y_{kli} + \sum_{k} \sum_{x_{ik}=1} p_{klui} u_{li} \leq p_k \tag{20}$$

On the other hand, deletion of a copy of file i from node n will increase the query cost by an amount given by

$$R(K_i'') - R(K_i) \tag{21}$$

where

$$R(K_i'') = \sum_{k=1}^{N} V_{ki} \min_{m \in Ji''} C_{mk} \tag{22}$$

$$K_i'' = K_i - \{n\} \tag{23}$$

Similar to the copy addition, the objective of copy deletion is to maximize

$$Z_{ni}^{d} = CS_i + UC_i + R(J_i) - R(J_i'') \tag{24}$$

Subject to

$$\forall \quad i \quad \sum_{k} x_{ik} \geq 1 \tag{25}$$

$$\forall \quad k \neq n \quad \sum_{j} (a_{kj} y_{knj} V_{kj} + U_{nj} x_{jk} u_{nj}) \leq \lambda_{nk} \tag{26}$$

$$\forall \quad k \neq n \quad \sum_{j} (a_{kj} y_{nkj} V_{nj} + U_{kj} x_{nj} u_{kj}) \leq \lambda_{kn} \tag{27}$$

$$\forall \quad k \quad \sum_{l} \sum_{x_{ik}=1} p_{klai} a_{li} Y_{kli} + \sum_{k} \sum_{x_{ik}=1} p_{klui} u_{li} \leq p_k \tag{28}$$

Note that the storage constraint for copy deletion is no longer valid as deletion frees the storage space but minimum replication constraint (25) has to be introduced. The benefits associated with an already allocated copy at a node n are given by:

$$R(K_i'') - R(K_i) + CU_i + CS_i \qquad (29)$$

Addition and deletion of a file copy is only beneficial if the respective objective function evaluates to a value greater than zero. Moreover, a beneficial copy can only be added or deleted if such addition or deletion does not violate any of the constraints.

4.2 The Data Reallocation Algorithm

Before executing the reallocation algorithm, the algorithm *Candidate Selection* is executed and the relevant information is transferred to an arbitrarily selected node at which the data reallocation algorithm is executed.

The proposed data reallocation algorithm iteratively improves the allocation. During each iteration, one file copy is considered for addition and deletion. A copy with the maximum positive value of the objective function is added/deleted provided all the constraint are satisfied. If only one copy of a file is present in the network, the benefit of such copy is calculated by considering another feasible node with minimum cost, as if it has a copy of the file. If a beneficial addition is not possible due to constraints, the less beneficial files at that node are deleted to accommodate the most beneficial file.

The proposed algorithm not only generates the new (near) optimal allocation but also generates a serial schedule (a sequences of steps) to transform the present allocation into the new optimal allocation such that no constraint is violated at any time.The algorithm can be described as:

Algorithm Data Reallocation

REPEAT
 $i \leftarrow$ file to consider for addition and deletion
 {during each iteration, next candidate file is considered}
 IF $i \in D_f$ THEN
  REPEAT {To delete beneficial copies of i}
   FOR all $n \in D_i$ Calculate Z^d_{ni}
   Select $m \in D_i$ such that
   $Z^d_{mi} \geq max_{n \in D_i - \{m\}} Z^d_{ni}$
   IF $Z^d_{mi} > 0$ THEN
    Delete i from m. Modify allocation of i
    $D_i \leftarrow D_i - \{m\}$
   ELSE No more beneficial deletion of i
  UNTIL(No more beneficial deletion of i
   OR $D_i = \phi$)

 IF $i \in A_f$ THEN
  REPEAT {To add beneficial copies of i}
   For all $n \in A_f$ Calculate Z^a_{ni}
   Select $m \in A_i$ such that
   $Z^a_{mi} \geq max_{n \in A_i - \{m\}} Z^a_{ni}$
   IF $Z^a_{mi} < 0$ THEN
    No more beneficial addition of i
   ELSE
    IF constraints are not satisfied THEN
     Delete the less beneficial files from m until the combined benefits of keeping the files are greater than Z^a_{mi} and deletion does not violate any of constraints or enough resources are regained. Modify their allocation and add all such file in A_f and calculate their candidate nodes using step 1 of the reallocation algorithm.
    IF constraints for copy addition are satisfied THEN
     Add i at m, modify allocation of i
     $A_i \leftarrow A_i - \{m\}$
    ELSE Restore previous allocation at m
  UNTIL(no more beneficial addition of i
   OR $A_i = \phi$)
UNTIL(No more beneficial addition or deletion
 or deletion OR all $A_{i's}$ and $D_{i's}$ are empty)

The reallocation algorithm given above has the following properties.

Property 1: The algorithm will add or delete a file copy during iteration n if the respective objective function evaluates to a value greater than zero and all the constraints are satisfied.

proof: The proof can be established by examining the reallocation algorithm.

Property 2: If the file allocation at *(n+1)th* iteration is different from that of *nth* iteration, then the total storage and the communication costs at the *(n+1)th* iteration is strictly less than the cost

at *nth* iteration.

Proof: Let K and K' be the file allocation at *nth* and *(n+1)th* iterations respectively such that $K \neq K'$. The objective function will only be evaluated to a positive value if the benefits of adding or deleting a copy are greater than the losses. A positive value of the objective function guarantees the decrease in the total cost. Therefore, the property holds true if $K \neq K'$.

Property 3: If the optimal allocation is obtained at *nth* iteration then the optimal allocation will not change at *(n+1)th* iteration.

Proof: If the optimal allocation is obtained at *nth* iteration then, from the property 2, the value of the objective function cannot be positive. Hence, no file will be added and deleted at *(n+1)th* iteration, otherwise allocation at *nth* iteration is not optimal which is contradiction to the fact that *nth* allocation is optimal.

5 Simulation Results

The proposed algorithms for the prediction of access/update frequencies and for the adaptive file allocation were implemented in Pascal on VAX 4000-100.

Most of the data used in the simulation of the prediction algorithm was taken either from [23] or from stock market traces. Up to 15 values were stored in each access and update history trace, and up to next five values were predicted. The result of the first prediction for 7 different traces are shown in Table 1. Figure 2 show the accuracy of the algorithm when next five values of a trace were predicted without updating the coefficients.

For the simulation of the reallocation algorithm, a network of 5 nodes was considered. The number of files were varied from 4 to 8. The file sizes were set from 2MB to 4MB, whereas the available space at each node was set between 7MB and 11MB. The simulation was performed by ignoring the processing and the channel constraints.

Instead of predicting the file utilization pattern, the access rates were varied either by a fixed percentage (5-25%) or by a random percentage for each node-file combination. The update rates were set randomly between 0% and 30% of the respective access rates.

The file allocations were generated using both the exhaustive search and the proposed algorithms for five different problem sets. Table 2 shows the average percentage accuracy of the proposed algorithm which was calculated by comparing the costs with the optimal ones. The average percentage error varies from 1.42% to 3.96%(see Fig. 2). Table 3 shows one particular instance of the simulation from each problem set. The solutions are within 5% of the optimal and the benefit of up to 19.1% justifies the data reallocation. Table 4 shows the efficiency of the proposed algorithm in terms of the number of times the objective functions were calculated vs. the total number of feasible allocations and the number of files which were added and deleted from the present allocation. In terms of CPU time, the proposed algorithm generated the solution in less than one second for each test case as compared to 5.30 seconds to 663.45 seconds taken by the exhaustive search algorithm. Figure 3 shows the effect of change in the reference rates on the accuracy of the algorithm.

6 Conclusions

A distributed system requires data reallocation to the participating nodes when the various parameters, such as file utilization pattern, and network topology, changes. The utilization patterns are characterized by the access and updates frequencies at each node. These frequencies vary over time, thus an optimal allocation at one period may be nonoptimal at another period.

In this paper, we presented three algorithms. The first algorithm estimates the future access and updates frequencies. As the new information becomes available, the estimates are revised and the assignment adapts itself to the new information. The second algorithm reduces the solution space by eliminating those nodes and files which are most unlikely to change their allocation in the new optimal allocation. The third algorithm generates the (near) optimal allocation. Two objective functions are used to calculate the benefit of adding or removing a file copy. Transition costs incurred by file transmission from one node to another are also taken into account.

The decision to reassign files can be triggered

Data set	Actual value	Predicted value
1	45	45.10
2	130	130.95
3	135	118.59
4	170	177
5	177	165
6	105	107.86
7	135	120

Table 1: Accuracy of the prediction algorithm

only when the net gains can be realized i.e. the new allocation generated by the algorithm is different than that of the present allocation. The simulation results demonstrate the accuracy of the assignments generated by the proposed algorithm is sufficiently close to the optimal with a reduced execution time compared to the exhaustive search algorithm.

Notations

i, j File indices

k, l, m, n Node indices

K_i Set of nodes with a copy of i

I_n Set of files at node k

x_{ik} 1 if file i is assigned at k, 0 otherwise

a_{ki} Access frequency of i from k

u_{ki} Update frequency of i from k

C_{kn} Unit data transmission cost from k to n

L_i Size of file i

S_k Unit storage cost at node k

V_{ki} Average amount of data of file i accessed by node k

U_{ki} Average update message size from node k to file i

p_{knai} Processing required per access of file i from node k at node n

p_{knui} Processing required per update of file i from node k at node n

P_k Processing limit at node k

λ_{kl} Channel limit between k and l

A_f Set of all candidate files which may be added

A_i Set of all candidate nodes which can have a copy of i

D_f Set of all candidate files which can be deleted

D_i Set of all candidate nodes which can relinquish a copy of i

M_k Storage limit at node k

y_{kmi} 1 if node m accesses file i from node k, 0 otherwise

References

[1] K. P. Eswaran: *Placement of records in file and file allocation in a computer network.* IFIPS Information Processing, pp. 304-307 (1974).

[2] A. Segall: *Dynamic file assignment in computer network.* IEEE TOAC, AC-21(2), pp.161-173 (1976).

[3] K. D. Levin and H. L. Morgan: *A dynamic optimization model for distributed databases.* Operation Research, 26(5), pp. 824-835 (1978).

[4] A. Segall and N. R. Sandell: *Dynamic file assignment in a computer network-part II: decentralized control.* IEEE TOAC, AC-23(5), pp. 709-715 (1979).

[5] C. T. Yu, M-k. Siu, K. Lam, and C. H. Chen: *Adaptive file allocation in a star computer network.* IEEE TOSE, SE-11(9), pp. 959-965 (1985).

[6] X. Du and F. J. Maryanski: *Data allocation in a dynamically reconfigurable environment.* Proc. Data Engineering, pp. 74-81 (1988).

[7] J. F. Ames: *Dynamic file allocation in distributed database system.* PhD thesis, Duke University (1977).

[8] H. M. Pocar: *File migration in distributed computer system.* PhD thesis, U.C. Berkeley (1982).

[9] A. Hać: *A distributed algorithm for performance improvement through file replication, file migration and process migration.* IEEE TOSE, 15(11) pp. 1459-1470 (1989).

[10] B. W. Wah: *File placement on distributed computer systems.* IEEE Computer, 17(1), pp. 23-32 (1984).

[11] B. Gavish and O. R. L. Sheng: *Dynamic file migration in distributed computer systems.* Comm. of the ACM, 33(2), pp. 177-189 (1982).

# of files	Change in reference rate (%)						
	5	10	15	20	25	Random (0–30)	Random (-30–+30)
4	96.4	96.35	96.45	96.34	96.43	95.90	94.45
5	100	99.10	100	99.56	100	98.12	93.25
6	99	99.60	99.64	99.65	99.55	98.57	93.29
7	98.7	99.63	99.53	98.68	99.58	96.25	93.29
8	98.7	97.57	100	97.23	96.4	97.84	88.27

Table 2: Accuracy of the proposed algorithm

# of files	Cost without reorganization	Cost after reorganization (proposed alg.)	Optimal cost	Difference (%)	Benefit of reorganization (%)
4	23995	22805	22805	0	5.2
5	53075	44565	44565	0	19.1
6	55450	48170	47180	2.09	15.1
7	58545	50140	50140	0	16.8
8	63625	57550	55000	4.64	10.6

Table 3: Benefit of file reallocation and the accuracy of the proposed algorithm

# of files	Total # of feasible allocations	Ave. # of times the obj. functions were calculated	# of files added and deleted
4	11883	18	7
5	28827	35	10
6	147231	67	13
7	311679	120	14
8	838596	196	20

Table 4: Efficiency of the proposed algorithm in terms of the number of times the objective functions were calculated

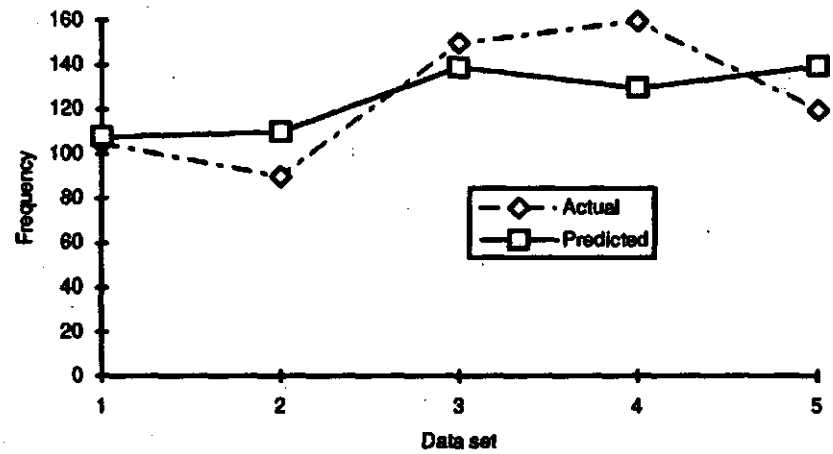

Fig. 1. Accuracy of the prediction algorithm for next five predictions

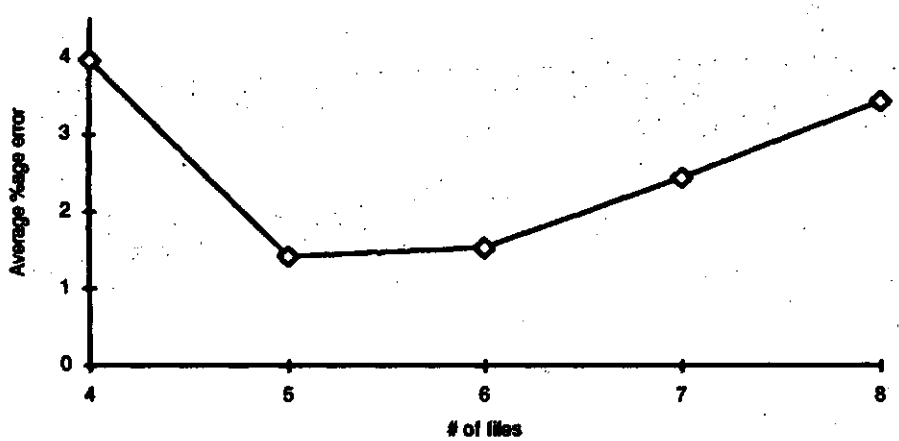

Fig. 2. Average %age error vs. problem size

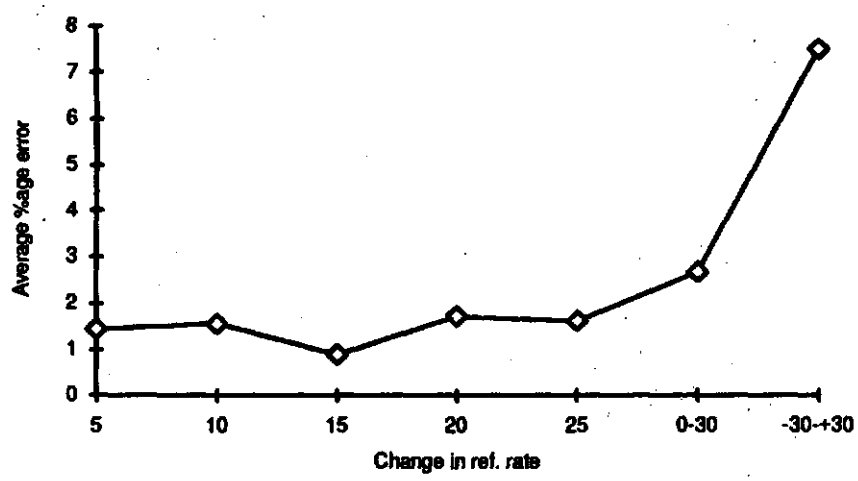

Fig. 3. Average %age error vs. change in reference rate

[12] D. Levin: *Adaptive structuring of distributed databases.* Proc. AFIPS National Computer Conference, pp. 691-696 (1982).

[13] A. Mahmood and H. A. Fatmi: *Adaptive file allocation in a computer network.* int. CISMOD'93, India (1993).

[14] L. W. Dowdy and D. V. Foster: *Comparative models of file assignment problem.* Computing Surveys, 14(2), pp. 287-313 (1982).

[15] J. Joyce, G. Lomow, K. Slind, and B. Unger: *Monitoring distributed systems.* ACM TOCS, 5(2), pp. 121-150 (1987).

[16] R. Snodgrass: *A relational approach to monitoring complex systems.* ACM TOCS, 6(2), pp. 157-196 (1988).

[17] D. Haban and D. A. Wybranietz: *A hybrid monitor for behaviour and performance analysis of distributed systems.* IEEE TOSE, 16(2), pp. 197-221 (1990).

[18] M. Mansouri-Samani and M. Sloman: *Monitoring distributed systems (A survey).* Imperial College research report No. DOC82/23 (1993).

[19] M. Hammer: *Self-adaptive automatic data base design.* Proc. National Computer Conference, pp. 123-129 (1977).

[20] A. N. Kolmogorov: *Interpolation and extrapolation of stationary series.* Bulletin del'Academie des Science del' USSR Series Mathematique, 2, pp. 3-32 (1942).

[21] D. Gabor, W. P. L. Wilby and R. Woodcock: *Universal non-linear filter predictor simulator which optimizes itself by a learning process.* Proc. IEE Part B, pp. 422-433 (1961).

[22] E. Grapa and G. G. Belford: *Some theorems to aid in solving file allocation problem.* Comm. of the ACM, 20(11), pp. 878-882 (1977).

[23] R. G. Brown: *Smoothing, forecasting, and prediction of discrete time series,* NJ:Prentice Hall (1962).

[24] P. Bay and A. Thomasian: *Data allocation heuristics for distributed systems.* INFOCOM, pp. 118-128 (1985).

CONCEPT REPRESENTATION OF THE SOFTWARE TOOL PIDMASTER FOR SYSTEMS MODELING AND CONTROLLERS TUNING

M. Ostroveršnik†, Z. Šehić‡, B. Zupančič‡, M. Šega*

†"Jozef Stefan"Institute,
Jamova 39, 61 000 Ljubljana, Slovenia.
‡Faculty of Electrical and Computer Engineering,
61 000 Ljubljana, Tržaška 25, Slovenia.
*Metronik d.o.o.,
Stegne 21, 61 000 Ljubljana, Slovenia.

Keywords: OOP, Software Engineering, Control System Design, PID Control.

Edited by: Miroslav Kubat
Received: October 22, 1993 **Revised:** January 11, 1994 **Accepted:** February 15, 1994

The work deals with the representation of the concepts of program package PIDMaster for on-site signal analysis, system modeling and controller's synthesis. A user-friendly interface is introduced which enables comfortable work also for less experienced users. The required user-friendliness has been achieved by the use of window interface completed with some elements of the object oriented user interface management system. Central points of the new interface are the so-called movable graphical objects. Their usage in the phase of system parameter estimation is explained thoroughly.

1 Introduction

Despite tremendous development of modern control theory, PID controllers are still widely used in process control applications. Moreover, there are no indications that the situation will change in the near future. The reason why PID is so popular in industrial environment is its ability to master the process dominant dynamics with (in principle) only three parameters, which are relatively easy to understand by process operators [1].

Roughly speaking, there are three categories of PID controllers available on the market [1]:

- PID controllers realized in classical analog technique,
- digital PID controllers,
- knowledge-based PID controllers with the abilities for autotuning and adaptation [1].

Most of the control equipment in industrial plants belongs to the first two categories. Tuning of such a controller is entirely in the domain of process operators and maintainers who approach tuning pragmatically using heuristics and intuition. Unfortunately, from the general point of view, PID controller tuning has remained an art rather than exact science [2]. Therefore, people in industry involved in maintenance need a tool that will help them tune the controller without requiring too detailed knowledge about the control theory. Such a tool is also desirable by control people involved in industrial applications and projects. Therefore, we decided to design and realize a tool that will provide the following:

- "manual" PID tuning on the process model,
- optimization based PID tuning for difficult and nonstandard problems,
- PID tuning by means of tuning rules,
- graphical modeling,
- intuitive and user friendly interface with extensive on-line context-sensitive and hypertext based help,
- ability for easy upgrading without reconstruction of the whole program,

- possibility to work in industrial area (portable, robust, relatively inexpensive).

Two years ago we started the project called PIDMaster. The final result of this project will be a tool which will fulfil previously mentioned goals. Such a tool should provide tuned controller parameters, process models and various progress and status reports, all on the basis of input data files obtained directly by data acquisition equipment, i.e. by measurements of the real process. Input to such a tool will be various data files obtained by data acquisition equipment through direct sensoring of vital variables of a target process. The output will be tuned controller parameters, models of process and various progress and status reports made by user of PIDMaster. In summary, the PIDMaster project has been started with the idea of supporting the above mentioned tasks, preventing huge human errors, offering the possibility of prototyping and thus implicitly learning by example.

2 Project Background

We decided to develop our tool on a PC platform because PC compatible hardware is really widespread and relatively inexpensive. Besides we already had experience with similar projects. Let us mention only two of them, the so-called ANA [3] and SIMCOS [4]. ANA has been made for system analysis and design of controller synthesis [3] whilst SIMCOS [4] has served for simulation of dynamic systems. Both tools were developed for experienced users and are fairly sophisticated. Major drawbacks of these two powerful tools, especially ANA, are relatively poor user interface and lack of project documentation, which caused difficulties with maintenance and upgrading. With the new project we wanted to dodge these problems.

The PIDMaster project differs from our previous projects in the following:

- we established a mixed development team,
- the project lifecycle was strictly divided into phases, and
- IBM's Common User Access (CUA, [5]) standard was used for design and implementation of the user interface.

We established a mixed development team from target domain and computer science experts which can cover both areas without segregation of one area. Actually, in the earlier project phases, target domain experts played dominant role and now, when we are in the implementation phase of the project, computer science component is more stressed. As mentioned above, to overcome the problem with a user interface we decided to use IBM's Common User Access standard, which is roughly supported with Microsoft windowing system Windows 3.x. Since the project is in the implementation phase we haven't had maintenance problems, but we hope that we can surmount such difficulties with thorough project documentation. The documentation includes several documents:

- system requirements document,
- system analysis document,
- design of user and tool interface document,
- design of important objects document, and
- database structure document.

3 PIDMaster Overview

The PIDMaster communicates with environment in three ways (fig. 1, i.e. Signals, Reports, User). Fundamental data inputs are process signals collected with other tools (i.e. Labtech, G-2). The main outputs are various reports that include figures of signals, descriptions of models and controller setups. We cannot eliminate user cooperation, so that all interactions are supervised by user. PIDMaster can import and export signals in various file formats like Matlab binary file format, Labtech ASCII format, the flat ASCII format etc. Reports can be written on any Win 3.x supported printer.

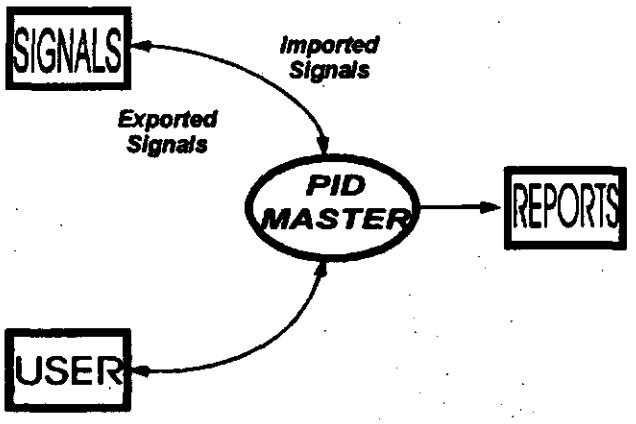

Figure 1: PIDMaster environment

From the user point of view PIDMaster incorporates four major units (fig. 2):

- analysis,
- modeling,
- controller tuning, and
- data import / export unit.

All units are linked via an internal database that includes all imported and/or generated signals, system models and controller setup parameters. All unit input and output formats are compatible, so that results obtained in a previously used unit can be used in the next unit. The most natural way of work is to start with the import of signals via analysis and modeling unit and then to proceed to the controller setup unit. Of course, the user can enter the process of controller tuning at any mentioned point (see fig. 2).

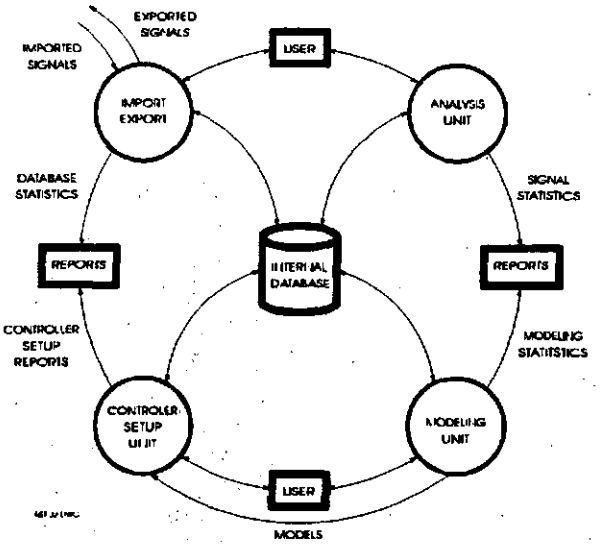

Figure 2: PIDMaster decomposition

The PIDMaster has certain characteristics that are common for tools of its sort and we will mention them only briefly. The main point of distinction regards system modeling from measured data. Since it is almost a way of communication with the user we shall explain it later in user interface section (see section 4).

3.1 Analysis unit

The analysis unit enables the user to display and edit signals on a screen in various ways (y=f(t), or z=f(x(t),y(t))). The tool can calculate standard integral criteria (e.g., Sx, IAE, ISE, ITAE, ITSE), compare two signals and calculate standard deviation, correlation, etc.

3.2 Modeling unit

In the modeling unit eight different mathematical models can be used. The common working pattern consists of selecting the model and entering the values of the parameters. These values can be later improved with a computer supported optimization or computer supported verification. The following models can be used in the process of modeling :

$$M1 : Y(s) = \frac{Ke^{-sT_d}}{1+sT_1}U(s)$$

$$M2 : Y(s) = \frac{Ke^{-sT_d}}{(1+sT_1)^n}U(s)$$

$$
\begin{aligned}
M3 : Y(s) &= \frac{e^{-sT_d}}{sT_1(1+sT_1)^n}U(s) \\
M4 : Y(s) &= \frac{Ke^{-sT_d}}{(1+sT_1)(1+sT_2)}U(s) \quad (1) \\
M5 : Y(s) &= \frac{Ke^2e^{-sT_d}}{s^2+2\xi\omega s+\omega^2}U(s) \\
M6 : Y(s) &= \frac{K(1+sT_0)e^{-sT_d}}{(1+sT_1)(1+sT_2)}U(s) \\
M7 : Y(s) &= \frac{K\omega^2(1+sT_0)e^{-sT_d}}{s^2+2\xi\omega s+\omega^2}U(s) \\
M8 : Y(s) &= \frac{KP_2(s)e^{-sT_d}}{Q_3(s)}U(s)
\end{aligned}
$$

where s, $Y(s)$, $U(s)$, K, T_d, T_i, ξ, ω, $P_2(s)/Q_3$ represent complex variable, Laplace transform of system output, Laplace transform of system input, process gain, pure time delay, time constant, damping factor, natural frequency and general transfer function, respectively parameters for equations (1) can be entered manually or via the so-called engineering method, which encompasses graphical and numerical estimation of parameter values. We believe that the support of the so-called engineering method for the parameter estimation is an important novelty and the main point of distinction between our tool and other available tools of this type. This procedure will hence be briefly described in the next section.

3.3 Controller tuning unit

The user can enter a controller tuning unit directly or via the modeling unit. Here he can select the appropriate controller and tune it's parameters. User can setup parameters directly or with the help of tuning rules (e.g., Ziegler-Nichols, Oppelt, Chien-Crones-Reswick, Rutherford-Aikman, Kessler [6, 7, 8, 9, 10]). It is possible to select both continuous and discrete controllers, different structures (e.g. P, PI, PD, PID) including modified structures (e.g. classical, noninteracting, industrial [10]). Selected parameter values can be optimized and/or verified with computer supported optimization and/or verification. The criterion function can be one of the output error based functions.

4 Engineering Method for System Parameter Estimation (EMSysPE)

Target user group of the PIDMaster are callow computer and control engineers so that we had to pay more attention to design of a friendly and especially forgiving user interface, which should attract new users. We decided to use Microsoft's windowing system and to follow the CUA standard if possible. The PIDMaster interface incorporates two distinct approaches, i.e.:

- diagrams (windows), and
- dialogues.

We use graphical windows for displaying values of signals in the form of diagrams and dialogues for numerical input and output. This is traditional approach alo used in other tools of this type. The main difference of our tool are the so-called movable graphical objects which combine both approaches in a new and more effective way. Movable graphical objects are part of module for estimation of system parameter values.

If we want to model an unknown system in the traditional way, we have to excite it with a step signal on input and observe its response on output. A typical response of a low-pass system is a sort of sloped step. The shape of the sloped step determines the type of the model. Both objects figure out model constants (delay, gain, time constant). That is almost a general algorithm for selecting the type of model and finding its constants. That sounds simple, but there are two problems. First, the algorithm is too general for computer implementation and can not be easily automated. Second, it includes relatively complex calculations that have to be executed several times. We decided to give the user additional support in areas where computers are generally more efficient and reliable (i.e., numerical calculations, memory capacity and reliability, ability of errorless repetition), and we should leave all freedom where man's intuition is essential.

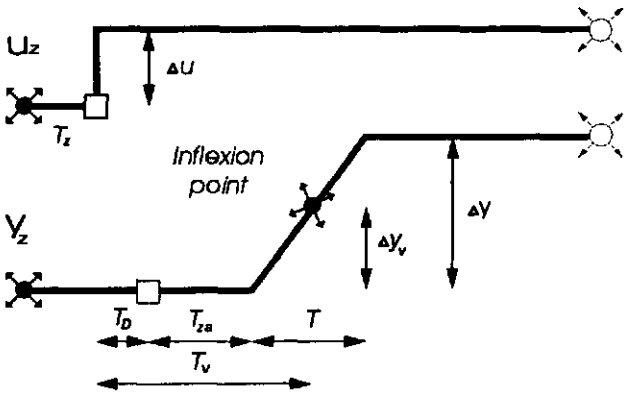

Figure 3: MGO's for model 1

Our concept is based on the so-called moveable graphical objects (MGO). MGO is a graphical object (see fig. 3) composed of markers and lines connecting them. The user can change dimensions or position of MGO simply by manipulating MGOs' markers. The visual appearance of a marker rules its function. The user can handle markers and thus MGO primarily with mouse click and drag. As a backup option we incorporate digital entry too. The system carries out two major MGO groups: simple and complex. The group of simple MGOs incorporates cursor, vertical and horizontal line, slope and parabola. The group of complex MGOs includes four different steps (step (see fig. 3, top), slopped step(see fig. 3, bottom), endless slopped step, slopped step with overshoot). The set of simple MGOs is primarily used in the phase of signal analysis. The set of complex MGOs is used in the system parameter estimation unit. The power of exploiting complex MGOs lies in the so-called engineering method for estimation of system parameters as can be shown in the following example.

Let's try to estimate system parameters with EMSysPE. The model is defined with equation M1 (see (eq. 1)) and we have to find gain, time delay and time constant (i.e., K, T_d and T_1). After selecting the proper model the program enables two MGOs. The first one is always a step and the second is one of the sloped steps. The type of the sloped step depends on the selection of the model. In our example the PIDMaster would give us the picture in fig. 3. As we can see, we have two MGOs, i.e. step and slopped step. The user's task is to properly relate the step on input and the ramp on output measured signal. Hence the user is responsible for correct positioning and dimensions of available objects. The position and dimensions of MGO can be set with the mouse. When the user is sure that both objects are positioned and scaled correctly he can start the process of calculation of model constants. It is the computer that is responsible for that. In our example the computer calculates the model constants from marker positions (e.g., $K = \Delta y/\Delta u, T_d = T_D + T_{za}, T_1 = T$). As can be seen, constants can be calculated directly from positions of markers. Estimated parameters can be saved in a database for later use in this or controller setup unit.

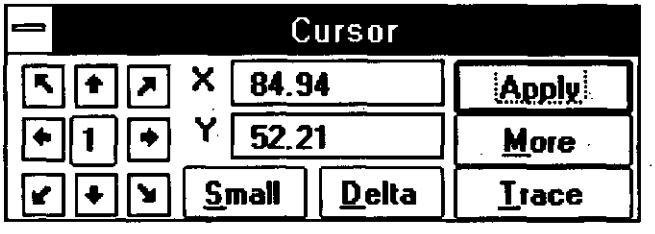

Figure 4: MGO's control window

Parameters can be verified and/or optimized. All intermediate steps can be saved in a log file. A snapshot of actual modeling with the PIDMaster is depicted in fig. 6, where the most important windows and controls can be seen.

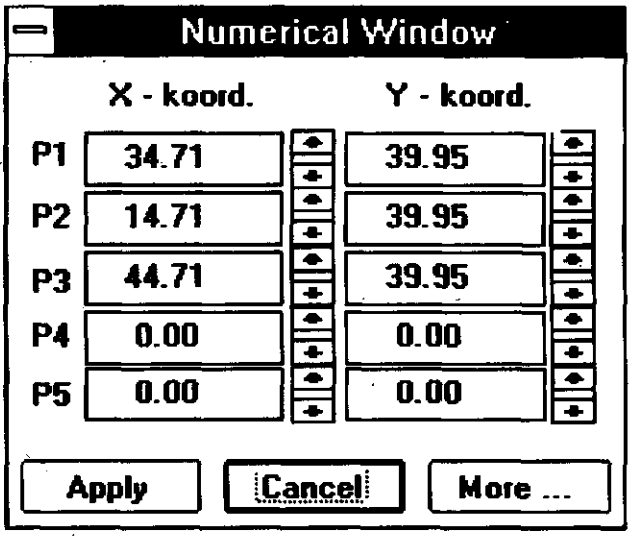

Figure 5: MGO's numerical window

The example shown is quite simple because there is no transitional step between graphical

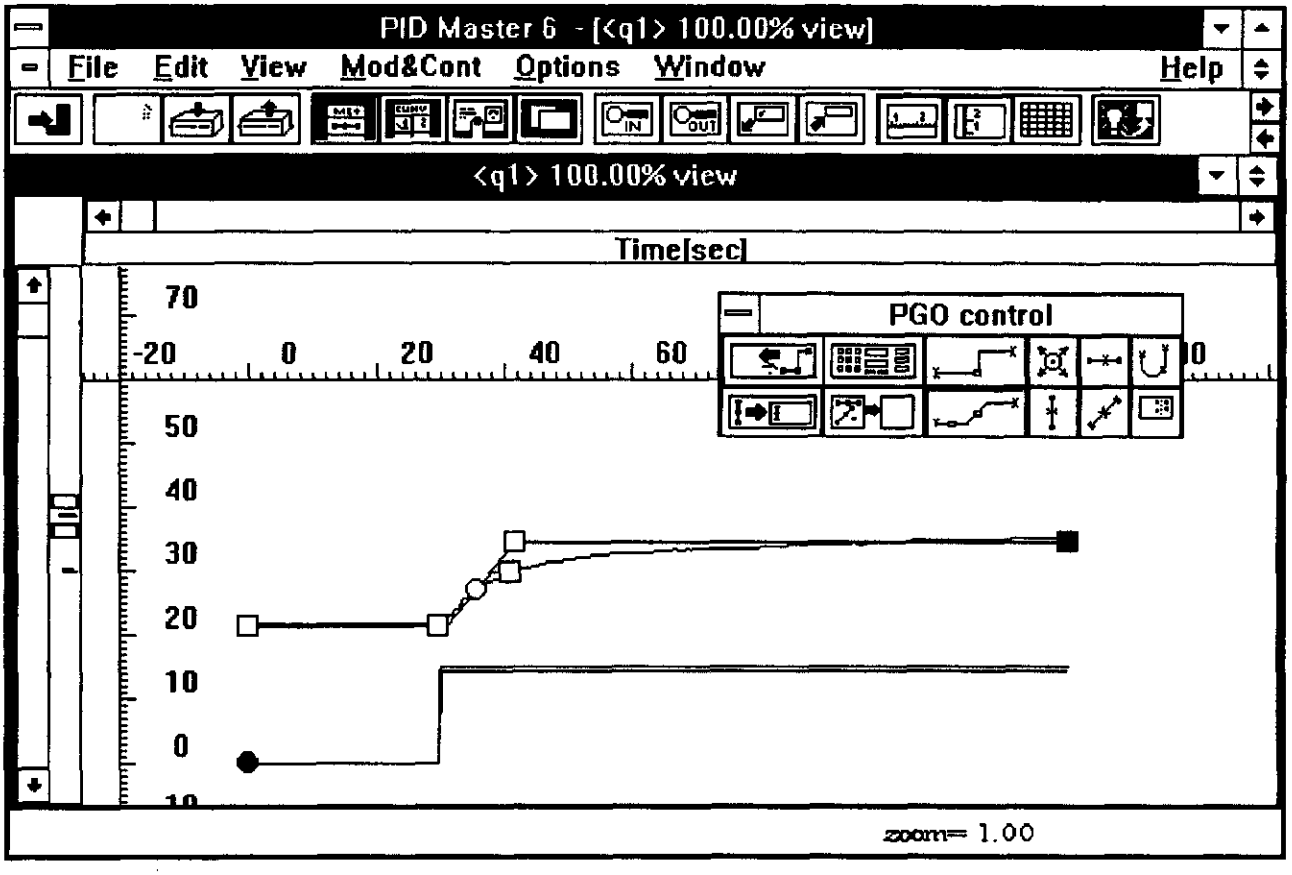

Figure 6: Graphical modeling of model 1

phase and calculation. In case of other models there is always an intermediate step in which one has to decide which possibility to choose. Anyhow the procedure is not so demanding since the user can select a certain value, check the effect and, if not satisfied, select another one, etc.

We believe that graphical estimation of parameters is sufficient for an average user but, nevertheless, we added also an option for numerical positioning of markers. The numerical input method is useful in case of low resolution graphic card or mouse, or in case of repetitive operations. The numerical input is accomplished via two additional modeless dialogues (see fig. 4 and fig. 5). The dialogue in fig. 4 (MGO Control) starts when the user selects an MGO. Values X and Y represent absolute coordinates of the active marker on MGO. New values can be entered directly in edit fields X and Y or via eight buttons. A click on button increases/decreases the selected value in a specified direction. For example, button ($\Rightarrow$) causes the value of field X to increase for delta, the opposite button ($\Leftarrow$) causes the value of field X to decrease for the same delta. The delta value is a user-defined value, with ability to be changed during program execution. Dialogue in fig. 4 enables user to read or change position of the active marker. On the other hand, dialogue in fig. 5 (the numerical window) gives the opportunity to read or change location of all markers in the same time. This dialogue can be activated with a click on button (More) on MGO's control window (fig. 5).

5 Status of Project and Further Improvements

The project is in the implementation phase and the beta version is in testing phase. We have succeeded to completely implement EMSysPE method, while controller setup and tuning is still pending. The significant delay was caused by se-

vere problems with compilers and graphical environment. In case of good acceptance of PIDMaster, we will upgrade our tool from the target domain experts side. The support for real-time signal acquisition can be added, as well as more support for various import/export file formats and support for OLE (object linking and embedding). A module for direct loading of parameters in a particular commercially available controller could also be developed.

References

[1] Åström, K.J., Hang C.C., Persson P., Ho, W.K.: Towards Intelligent PID Control, Automatica, vol 28, p.p. 1-9, 1992.

[2] Gilbert Polonyi, M.J.: PID Controller Tuning Using Standard Forma Optimisation, Control Engineering, p.p. 102-106, March 1989.

[3] Šega, M., Strmčnik, S., Karba, R., Matko, D.: Interactive Program Package ANA for System Analysis and Control Design, Preprints, 3rd IFAC/IFTP Int. Symposium CADCE '85, p.p. Copenhagen, August 1985.

[4] Zupančič, B.: SIMCOS - jezik za simulacijo zveznih in diskretnih sistemov, Fakulteta za elektrotehniko in računalništvo, Univerza v Ljubljani, Ljubljana, 1992.

[5] IBM Systems Application Architecture: Common User Access Advanced Interface Design Guide.

[6] Ziegler, J.G., Nichols, N.B.: Optimum Settings for Automatic Controllers, Trans. ASME, vol. 64, p.p. 759-768, 1942.

[7] Opelt, W.: Kleines Handbuch Technicher Regervorgänge, Verlag Chemie, GmbH, Frankfurt am Main, 1964.

[8] Hofmeister, W.: Prozessregler - Auswahlkkriterien und Einstellungshilfen, VDI - Verlag, GmbH, Düseldorf, 1975.

[9] Isermann, R.: Digital Control Systems, Springler Verlag, Berlin-Heiderberg-New York, 1981.

[10] Kaya, A.: Tunning of PID Controls of Different Structures, Control Engineering, p.p. 28-31, July 88.

EVALUATION OF SOFTWARE QUALITY ATTRIBUTES DURING SOFTWARE DESIGN

C. Wohlin
Dept. of Communication Systems, Lund Institute of Technology, Lund University, Box 118, S-221 00 Lund, Sweden

Keywords: quality evaluation, performance, functional analysis, software reliability, simulation, software metrics, Cleanroom

Edited by: M. Kubat
Received: October 28, 1993 **Revised:** January 18, 1994 **Accepted:** February 28, 1994

This article presents an evaluation method of software quality attributes during software design with a high-level design technique. The attributes considered are real time functional behaviour, performance (in terms of capacity) and reliability. The method is based on transformation of the software design documents and simulation models of hardware architecture in terms of processors, communication channels etc. and the environment in terms of usage of the system. The method provides an opportunity to concentrate on software, architecture and usage of the system one by one and facilitates analysis of the software system long before it is taken into operation, which is particularly valuable for safety-critical software and other complex software systems. This implies that important information concerning both functionality, performance and reliability can be studied early in the development, so that re-design can be performed instead of implementing a poor solution. These early insights become more and more

1 Introduction

The quality of the software in the operational phase is a critical issue. The cost for software failures in the operational phase of software systems today is very high, both in risks and economical terms. This is in particular the case for safety-critical software and other large software systems where the society depends heavily on the software.

The software community is not in control of the quality of the software. Cleanroom Software Engineering [5], [6] and [10] is a promising attempt in dealing with these problems, but it is not enough. Methods are needed for early analysis of functionality, performance as well as reliability.

The dependability of software systems is highly due to getting things right from the beginning. This is essential since the main principles of a software system are often built into the product at an early stage of the software life cycle. The inevitable conclusion from this is that it is necessary to have methods for modelling and analysing the behaviour of the software long before it goes into operation.

A method for evaluation of performance, reliability and functional aspects of software systems at an early stage is presented. The method supports prediction of important quality attributes at the software design phase. The evaluation and prediction is based on the actual software design and simulation models of architecture and user behaviour.

The article presents initially some objectives of the work before discussing some related work. The concepts in the method are then described from a general point of view. The general concepts are exemplified with SDL (Specification and Description Language, standardized by ITU, [3] and [4]), i.e. transformation rules are presented and an example is described to show the applicability of the method. Finally some general conclusions are presented.

2 Objectives

The main objective of this work is to formulate a general (independent of software description technique) method for functional, performance and reliability evaluation at an early stage of software development. The long term objective is to formulate a method that can be applied throughout the software life cycle to evaluate and assess the quality attributes of software systems. The objective is that the principles presented can be used throughout the software life cycle even if the actual level of detail in the models used may vary depending on available information.

The aim is to provide a method for evaluation of functional real time behaviour, performance (in terms of capacity) and reliability of software design descriptions. The method is based on that the software design descriptions are specified with a well-defined language, for example SDL, which can be transformed automatically into a simulation model of the software design. A tool prototype performing the transformations of SDL descriptions into a simulation model of the software has been implemented, [18] and [8]. It must be stressed that SDL is used as an example assuming that SDL is the normal software development method at the company applying the proposed method.

Transformation rules have been formulated for SDL, hence showing that it is possible to actually use the design in the evaluation of quality attributes instead of formulating a separate simulation model of the behaviour of the software. The transformed model is then distributed on a simulation model of the architecture. The input to the system (transformed software design distributed on a simulation model of the architecture) is then modelled in a usage model, which is a simulation model of the anticipated usage of the system. The method consists hence of three separate models: software model, architecture model and usage model.

The software model is a direct transformation of the actual design of the software to be used in the final system. The usage model and the architecture model are formulated in the same language as used in the software design, but these two models are supposed to be simulation models of the actual architecture and of the anticipated behaviour of the users of the system. The three models are hence described with the same description technique which is the same technique as the software is being designed in. The strength of the method is its opportunity to combine the actual software design with simulation models.

The usage model is used as a traffic generator to the system, i.e. it sends signals to the system in a similar way as expected when the system is put into operation. The reliability of the software can be evaluated since failures occur as they would in operation, since the usage model operates with a usage profile which describes the anticipated usage in terms of expected events. The capacity of the system is determined based on the inputs coming into the system and measurements on loads and throughputs. The analysis allows for analysis of bottlenecks in the system as well as delays. The real time functional behaviour is analysed in terms of locating unexpected functional behaviour. In particular, it is possible to find functional behaviour that is a direct consequence of the delays in the system.

The difference between the work presented here and other approaches is the opportunity to combine the software design with simulation models described in the same description technique as the software design. The idea in itself is general and no direct limitations concerning for which design techniques this approach can be applied have been identified. The objective has neither been to formulate a tool set nor to advocate the use of SDL. The major difference with existing approaches is that a special notation has not been used and hence the method is believed to be general and the method aims at more than one quality attribute. This implies that it should be possible to adapt the general idea and formulate transformation rules etc. for other design techniques as well. The aim is to provide a framework and a method supporting early evaluation based on the actual software design as well as other description levels in the future.

The advantages with the proposed scheme can be summarized by:

- the evaluation of quality attributes can be performed at an early stage, i.e. during the design (cf. below with for example statistical usage testing),
- the concepts are general even though transformation rules have to be formulated for

each specific design language,

- the actual software design is included in the evaluation method hence allowing for a good basis for decisions regarding the quality of the software,
- the method aims at analysing performance, reliability and real time functional behaviour hence no separate analysis has to be performed for each quality attribute.

3 Relation to other work

Some approaches resembling the functional and the performance evaluation can be found, see for example [1], [7], [14] and [15]. These methods do neither support reliability evaluation of the software nor do they base the analysis on the actual software design. One of the major differences with the approach presented in this article is its generality. The objective has been to clearly separate method from tools. The approaches presented in [1], [7], [14] and [15] primarily aims at using a "home made" notation for the problem and then building a tool supporting the proposed notation and method. The method presented here takes a more general approach by introducing concepts and ideas, which can be adapted to the design method and tools used in the ordinary software development process. The objective is to make the method independent from any specific description technique and tool environment.

Statistical certification of software, including usage modelling, is currently an area evolving rapidly. Statistical usage testing as a method for certification was initially proposed within the Cleanroom method [5], [6] and [10]. A similar approach is currently used and being developed at AT & T. It is concluded from the projects at AT & T, [9], [11] and [12], that the cost for system test and the overall project cost are reduced considerably. In [11], it is stated that the cost reduction for system test for a "typical" project is 56%, which is 11.5% of the total project cost. Usage modelling from Markov chains are discussed in [13] and [16]. The opportunity to perform early software reliability estimation from high-level design documents using dynamic analysis has been presented in [19]. The reliability certification is normally applied during the test phase, with exception of the work presented in

4 Overview: Modelling concepts

Three models are defined to formalise the modelling and evaluation process. The mapping between reality and models are depicted in figure 1. The models are denoted:

- Software Model
- Usage Model
- Architecture Model

The models, which will be explained below, are independent in the sense that the Software Model, the Usage Model and the Architecture Model are derived independently and they can be combined into an Evaluation/Simulation Model. The simulation is foremost intended to be used during the software design.

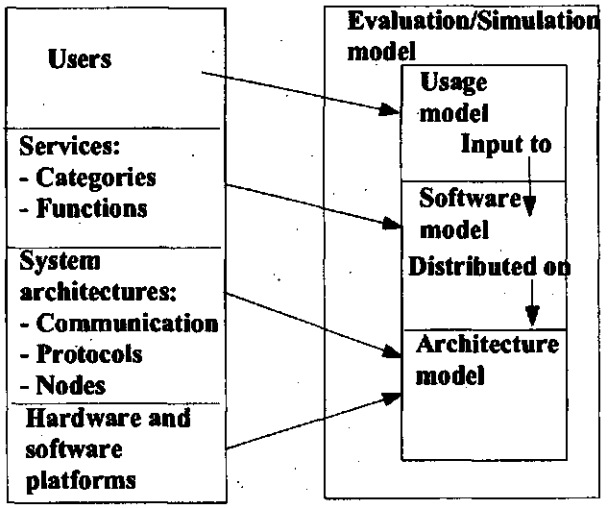

Figure 1: Mapping of the layers of users and system on the modelling concepts.

The Software Model and the Architecture Model are linked to each other through the distribution of the software units in the architecture. This means that the Software Model is allocated to the Architecture Model in a way that describes the actual distribution of the software in the architecture. The Usage Model generates the input to the simulated system (Software Model allocated on the Architecture Model). Thus the three models are connected together into the Evaluation/Simulation Model.

5 Model description

The models are derived in the following manner:

- Software Model
  The software descriptions (specification or design) are transformed to include the real time aspect of the software, which normally is not included in the software design. As part of the transformation the user is requested to add time consumption for executing different concepts of the software design. This addition of time consumption must be made based on prior knowledge or knowledge of the current system. The Software Model describes the application software, i.e. the services that the system provides to the user.

  The content of the Software Model is shown in figure 2.

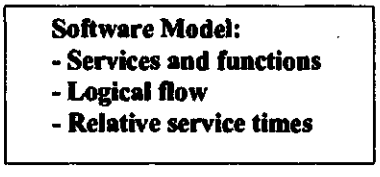

Figure 2: Content of the Software Model.

- Usage Model
  The external usage is not normally described unless a reliability certification is to be made or as input to a performance simulation. The Usage Model must describe the structural usage of the analysis object (which refers to the part of the system being evaluated) and it shall be complemented with a usage profile to allow for reliability certification. The structural usage describes the behaviour of the user without quantification of the usage. The latter is added with the usage profile. Usage modelling with Markov chains is discussed in [13] and [16]. It is, however, favourable if the Usage Model is formulated in the same description technique as used in the software design. This is clearly possible as shown in [17] and in the example below.

  The content of the usage model is shown in figure 3.

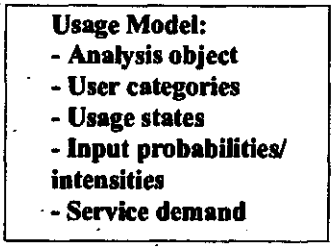

Figure 3: Content of the Usage Model

- Architecture Model
  The architecture is described in terms of a performance simulation model. This type of model is quite common and it is used extensively when doing performance analysis. The aspects to find are those governing the performance behaviour of the architecture. The objective here is to define a performance model of the architecture in the same description technique as has been used in the software design. This is also further discussed in [17] and in the example below.
  The content of the architecture model is shown in figure 4.

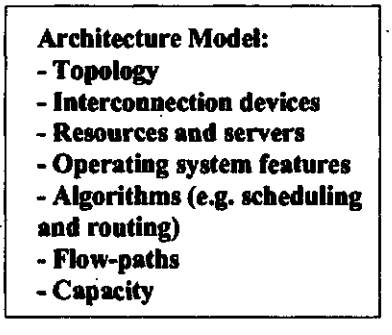

Figure 4: Content of the Architecture Model.

The concepts of the method have hence been described and to illustrate the application of the method SDL is used. First a brief introduction to SDL is given, secondly the transformation rules for SDL are described and finally an example using the method is presented.

6 Brief introduction to SDL

The ITU Specification and Description Language, [2], [3] and [4], known as SDL, was first defined in 1976. It has been extended and reorganised in four study periods since this first definition. These have resulted in new recommendations for the language published every fourth year.

SDL is intended to be well-suited for all systems whose behaviour can be effectively modelled by extended finite-state-machines and where the focus is to be placed especially on interaction aspects. SDL is a unique language which has two different forms, both based on the same semantic model. One is called SDL/GR (graphical representation) and is based on a set of standardized graphical symbols. The other is called SDL/PR (phrase representation) and is based on program-like statements.

The main concepts in SDL are system, blocks, channels, processes and signals. These concepts form the basis for SDL, where system, blocks and channels describe the static structure while the dynamic behaviour is modelled with the processes and its signals. The processes are described by several symbols.

System: Each system is composed of a number of blocks connected by channels. Each block in the system is independent from every other block. Each block may contain one or more processes which describe the behaviour of the block. The only way of communication between processes in two different blocks is by sending signals that are transported by channels. The criteria leading to a certain division of the system into blocks may be to define parts of a manageable size, to create a correspondence with actual software/hardware division, to follow natural functional subdivisions, to minimize interactions, and others.

Block: Within a block, processes can communicate with one another either by signals or shared values. Thus the block provides not only a convenient mechanism for grouping processes, but also, a boundary for the visibility of data. For this reason, care should be taken when defining blocks to ensure that the grouping of processes within a block is a reasonable functional grouping. In most cases it is useful to break the system (or block) into functional units first and then define the processes that go into the block.

Channel: Channels are the communication medium between different blocks of the system or between blocks and the environment.

Signal: Signals can be defined at system level, block level, or in the internal part of a process definition. Signals defined at system level represent signals interchanged with the environment and between system blocks. Signals defined at block level represent signals interchanged between processes of the same block. Signals defined within a process definition can be interchanged between instances of the same process type or between services in the process. Signals are sent along signal routes between processes and on channels between blocks or when interchanged with the environment.

Process: A process is an extended finite-state-machine which defines the dynamic behaviour of a system. The extended finite-state-machine handles data within tasks and decisions. Processes are basically in a state awaiting signals. When a signal is received, the process responds by performing the specific actions that are specified for each type of signal that the process can receive. Processes contain many different states to allow the process to perform different actions when a signal is received. These states provide the memory of the actions that have occurred previously. After all the actions associated with the receipt of a particular signal have been performed, the next state is entered and the process waits for another signal. The basic concepts within a process are further described below.

Processes can either be created at the time the system is created or they can be created as a result of a create request from another process. In addition, processes can live forever or they can stop by performing a stop action. A process definition represents the specification of a type of process; several instances of the same type may be created and exists at the same time; they can execute independently and concurrently.

7 Transformation rules for SDL

7.1 SDL concepts

Each SDL concept or symbol must be associated with parameters that describe its behaviour in terms of performance, to create a Software Model that together with the Architecture Model and the Usage Model shall be put together to the simulation model. Initially the method has been focused on the most common used concepts, denoted Basic concepts (process level), which describes the behaviour within a process.

The parameters that can be associated with the symbols are:

- time, i.e. a delay in terms of an execution time (random or constant),
- probability, i.e. the probability for different outcomes when passing the symbol during the execution,
- signal reception or sending (intensities). The reception of a signal is based on the execution in another part of the analysis object or the environment, while a signal sending is based on the actual execution of the analysis object. This means that no intensity has to be directly associated with the symbol, but the symbol must be associated with a signal reception or sending all the same. The intensity is a consequence of the execution. The arrival intensities for signals arriving from the environment of the analysis object is part of the Usage Model, which models the surrounding of the analysis object.
- dynamic behaviour, i.e. creation or deletion of a process.

An estimate of one of these parameters can either be only one estimate or a combination of several estimates that have been weighed together. The basic concepts are associated with the parameters as follows:

Basic concepts (process level):

State - execution times, i.e. the times it takes to leave and enter a state respectively. Each transaction results in one time leaving a state and one time for entering a state, except when the transaction means that the process terminates its execution.

Input - signal reception and execution time, i.e. the time it takes to remove the signal from the queue and start the execution.

Output - execution time and signal sending. The execution time shall correspond to the time it takes to send the signal, while the signal sending is a direct consequence of reaching the output symbol.

Save - no parameters are associated with the save symbol, since it is assumed that the handling of the queue of a process is carried out by the processor responsible for that specific software process. This means that any delays caused by the queueing discipline shall be modelled by the Architecture Model. The save symbol has to be present after transformation as well to be able to save the right signals in the queue.

Task - execution time, i.e. the time it takes to perform the actions specified in the task.

Decision - execution time and probability. The execution time must correspond to the time it takes to evaluate the statement in the decision symbol and to choose execution path based on the evaluation. In cases when the criterion within the decision box can not be evaluated based on the specification, i.e. the symbol does only contain informal text, a probability for different paths is needed. The probability or probabilities shall model the number of times a certain execution path is chosen, based on the evaluation, compared to the total number of times the symbol is passed. The number of probabilities will be one less than the number of possible paths to leave the decision symbol, since the sum of the probabilities are equal to one.

Create request - execution time and dynamic creation. The execution time models the time it takes to create a new process and initiate all its values. The dynamic creation must be a part of the Software Model, since it is a vital part of the dynamic behaviour of the analysis object. This means that the symbol cannot just be translated to a delay.

Terminate process - execution time and dynamic deletion. The explanation is similar to the one about "create request", with the difference that a process is terminated instead of created.

Timer - Signal sending. The set and the reset command respectively are carried out within a task, which means that the execution time is associated with the task concept and not the timer concept. But the timer also means that a signal is sent to the process itself, and this signal sending has to be modelled in the Software Model. The internal signal sending is performed with the set concept.

Procedure - Execution time. The procedure symbols (call and return) themselves are only associated with execution times, while the result of the procedure call, i.e. the execution of the procedure, is modelled symbol by symbol within the procedure and according to the rules for respective symbol. This means that the procedure call symbol cannot be translated merely to a delay. In other words the procedure call symbol must remain in the Software Model.

Macro - The macro symbol is substituted with

the content of the macro before the execution, which means that it can be disregarded from a performance point of view.

Join and label - Join and label have two main functions, either showing that the flow description continues on another page (in the description) or describing a "goto" in the program. The latter means that a jump is made to another part of the program. Instead of handling these two functions separately, it is assumed that the jump is done instantaneously independent if the jump is done to the next line or to another part of the program. This means that join and label are assumed to have no influence on the performance.

Asterisk - The asterisks are only a shorthand to be used as a wildcard, for example in a save symbol, where it means that all signals not explicitly received shall be saved. This shorthand is merely a simplification when working with SDL and it does not influence the behaviour of the actual implementation.

This discussion, of which performance parameters that have to be associated with the SDL symbols, leads to that transformation rules for each symbol can be formulated.

7.2 Example of some transformation rules

The transformation means that the system design in SDL is transformed into SDL descriptions describing the system from a performance perspective. Based on the parameters identified for each symbol transformation rules can be found. The transformation rules below include in some places symbols that are optional depending on the actual behaviour of the original description. A major objective for all concepts is that, if it is possible to put two or more transformations together it shall be done. For example, the execution times for two symbols after each other shall be put together to one delay. It must also be noted that the whole delay for a symbol is always assumed to be done first, before the actual event described by the symbol occurs. The delay can either be given a specific value or a random time, with some mean value, from some distribution. This means that when the task (set timer) is referred, it shall be possible to incorporate a random or constant delay if wanted. The assignment of values and the c

Basic concepts (process level): The basic concepts describe the dynamic behaviour of a process and they are translated to basic concepts describing their behaviour from a performance perspective. Most of the concepts, as described above, mean some sort of execution time (i.e. delay). The execution time can always be modelled with a task (set timer) and a state in which the process stays until the timer signal is received. Instead of describing this for each concept below a new meta-concept, which will be denoted "delay", will be used.

Definition of *Delay*:
Task (set timer): The timer is set to the time it takes to execute a particular symbol.
State: The "symbol" remains in this state until the timer signal is received.

The delay concept will be a shorthand for a task where a timer is set and a state where the process is waiting until the timer signal is received. The delay concept also includes the possibility of generating a random number corresponding to the execution time. The random number is drawn from a distribution with parameters that model the behaviour of the execution of the symbol. Some examples of how the basic concepts are transformed are:

State
Delay: The state symbol is really two different events that is leaving and entering a state respectively. The entering event is often referred to as "nextstate" from the perspective of the state we are leaving. This means that the delay can be divided into two parts, i.e. one delay for leaving the state and one delay for entering a new state (perhaps the same).

Input
Input: The signal must be received, since it can influence the forthcoming behaviour of the analysis object. *Delay*: The time to take the signal from the queue and to evaluate what to do has to be modelled as a delay.

Save
The save symbol will remain unchanged, since it is needed to be able to handle the queue in an appropriate way, i.e. to save the right signals depending on the state of the process. It must be observed that the process is assumed to be in the last state until it reaches the next state, and the states referred to are the "real" states of the origi-

nal process and not the states that are introduced to cope with the delays. The reason for this is that the queue and the arriving signals shall be handled in an appropriate way, i.e. the signals shall be saved according to the actual system description.

Decision

Delay: The time to evaluate the statement within the decision symbol is modelled with the delay.

Task: If the decision symbol contains informal text a task is needed. The task shall be used to draw a random number from a uniform distribution to compare with the given probabilities for the different execution paths, which model the behaviour of the decision box in cases when the decision can not be evaluated from the original description.

Decision: The decision symbol is either left without any changes or in cases when the symbol contains informal text complemented with a decision criterion. The complement means that a random number is compared with the probabilities for different execution paths and a path is chosen according to the evaluation.

Task

Delay: The time to execute the task is modelled as a delay.

Create request

Delay: This delay models the time it takes to create a new process.

Create request: The new process is created in the SDL description describing the performance aspect as well.

Rules have in a similar way been formulated for all other basic concepts in SDL. The objective is to do the transformation automatically and in a dialogue with the user. The dialogue gives the user possibility to assign values to execution times, probabilities etc. The transformation rules and the concepts are evaluated through an example subsequently.

8 Example

8.1 Introduction

The objective of the example is to provide an illustration of how the proposed method may work and give a flavour of the type of analysis that can be made. This is made by going through some results from the example. It is impossible to go through the example in detail, but more information can be found in [17]. The example has been formulated using SDL (Specification and Description Language), [2], [3] and [4], but any other well-defined design technique could have been used. The example will show results concerning functional behaviour, performance and reliability.

The example describes the communication between two very simple telephone exchanges, which only provide the subscriber with the possibility to call a local call (within the same exchange) or long distance call (to the other exchange). The architecture is modelled with three SDL process types, one describing an exchange, one modelling the communication channel between the exchanges and one handling the administration of the architecture. The services provided by the exchange are designed with seven SDL process types. The usage is modelled with five SDL process types. Some of the process types are created and terminated dynamically, and several instances of the same process type may exist simultaneously. This means that the example in total includes 15 process types.

The system layout of the example including the environment is illustrated in figure 5.

The example can be summarised by:

- 5 processes modelling the behaviour of the subscribers.
  - *A_Subscriber*: models the behaviour of a phoning subscriber.
  - *B_Subscriber*: models the behaviour of a phoned subscriber.
  - *Call_Generator*: responsible for creating *A_Subscribers* as a new call shall start.
  - *Monitor*: responsible for creating *B_Subscribers* when connecting a call.
  - *Creator*: responsible for creating *Monitor* and *Call_Generator*.
- 3 processes modelling the architecture.
  - *Processor*: models the behaviour of the processors executing the software.
  - *Com_Link*: models the communication link between processors.
  - *Arch_Creator*: responsible for creating *Processors* and *Com_Links* according to the layout of the architecture.
- 7 processes designing the services provided by the software.

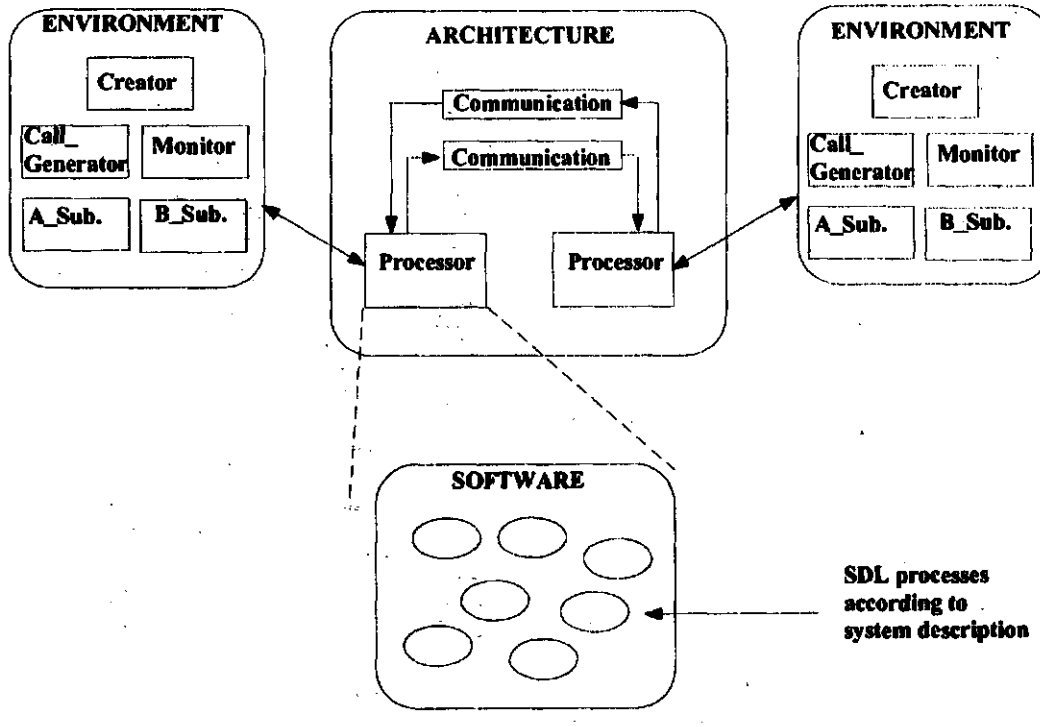

Figure 5: Layout of the example.

- *Statistics*: models the times when statistics about the processors (exchanges) shall be written on a file.
- *A_Handler*: handles the phoning subscribers.
- *B_Handler*: handles the phoned subscribers.
- *Digit_Handler*: responsible for the reception of digits and reserves a code receiver through communication with *Code_Receiver*.
- *Code_Receiver*: responsible for holding code receivers for on-going calls.
- *Soft_Monitor*: responsible for monitoring on-going calls at a processor.
- *Soft_Creator*: responsible for creating the *Soft_Monitors* necessary due to the architecture.

8.2 Software design in SDL

The SDL system description consists, as pointed out, of 7 processes, but before these are discussed in some more detail, the system and block level have to be discussed. The system consists of two blocks. The first block handles all activities that concern subscribers and the second one is a block responsible for collecting the statistics of the telephone exchange. The layout of the SDL design is shown in figure 6 and the processes and their communication are briefly described below.

The statistics block is simple, it only consists of the *Statistics* process of which one instance is created at the system start and it exists the whole life time of the system. The only thing to be noted with the process is that it calls a procedure regularly, which describes the times the statistics are put on a file.

The subscriber block consists of six processes, where one process is created by the system (*Soft_Creator*). This process is responsible for creating two monitors (one for each processor). The *Soft_Monitor* process is the receiver of incoming calls and it creates other processes that handles the subscribers, both A- and B-subscribers. The *B_Handler* process is created by the monitor in cases of a long distance call, otherwise the *B_Handler* process is created by the *A_Handler* process. The *B_Handler* process is quite easy and it handles the communication with the B-subscriber in the environment. The monitor process also creates the two processes controlling code receiving.

The *A_Handler* process creates a *Digit_Handler* process and is responsible for keeping the contact with the A-subscriber in the environment. The

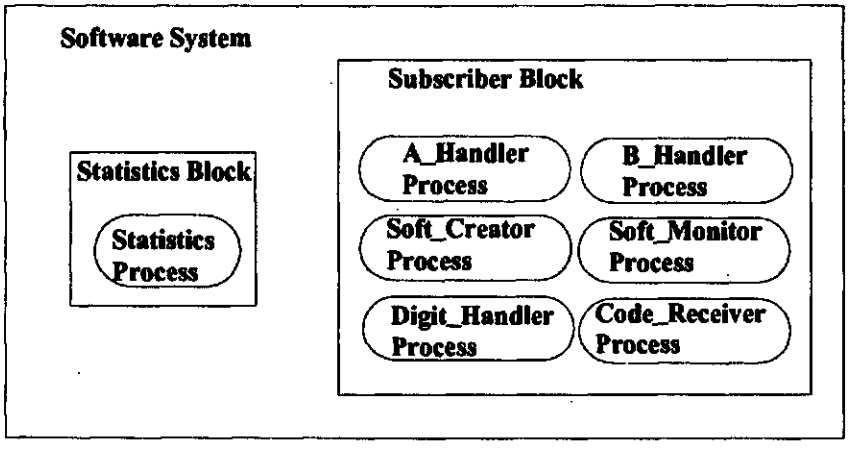

Figure 6: Software processes in the design.

Digit_Handler process checks if there are any code receivers free and if there are any it reserves a code receiver for the incoming call. The call is blocked if the code receivers all are occupied. The *Digit_Handler* is responsible for releasing the code receiver when it has been used.

The *Code_Receiver* process is not modelled in any detail at all. It consists mainly of signal receiving, signal sending and informal text. This is an important aspect, i.e. that the processes may be quite unspecified but still the quality attributes can be evaluated by applying this concept.

The tool support used allows for syntax and semantic analysis of the SDL system and code generation (SDL to C). The generated code was then compiled and linked. The tool allows for functional simulation without taking real time into consideration. A functional simulation was performed but it failed since the SDL system was not complete, i.e. some of the behaviour was not formally specified, it was only given in informal text. The problem with the informal text is in particular decision boxes with informal text since this means that the actual execution is not formally described. The decision boxes with informal text are translated when doing the transformation into a the Software Model.

8.3 Identify objectives of simulation

The objective of the simulation shall not influence the translation technique applied, but it may influence the way the architecture and the usage are described in the model. The objectives of the simulation can lead to that a measurement process has to be included in the simulation system. In this case the objectives are:

- Determine the total execution times of each software process type on the processors. This provides a possibility to identify the software bottleneck, since the available execution time is known through the simulated time. This must not be mixed up with the execution time of the simulation program.
- The load on the communication links is measured.
- Identify any functional problems.
- Detect faults that influence the perceived reliability of the software.

These objectives affect the process modelling the processor, the process modelling the communication link and the Software Model processes. A complement is needed in the Software Model processes to be able to measure for each process type, see below. The latter measurement will require a separate measurement process. These objectives have to be modelled by the user, when describing the architecture, the usage and introducing a measurement process.

8.4 Software Model

The original SDL system is transformed applying the rules formulated for SDL. The transformation results in a new system level, where the original SDL system is a block. This corresponds to the Software Model discussed above. The new system level also contains three new blocks, i.e. one block modelling the architecture, one block modelling the usage and one general block (used

to control the simulation and to make measurements). These three blocks are left without content. These blocks will be complemented with the Architecture Model and the Usage Model as well as general processes governing the simulation. These models have to be formulated by the user, see below. The Software Model and the generated empty blocks are illustrated in figure 7.

All block levels in the original system have been moved one step down in the hierarchical structure. The processes in the Software Model are the result of applying the transformation rules. It must in particular be observed that all symbols with informal text have been removed or replaced, but they are modelled in terms of delays.

As an example of a transformation we will consider the transformation of a task in an SDL process, see figure 8.

The task remains after the transformation if it contains a functional behaviour in which case it influences the outcome of the execution of the process. Independently of the content of the task, a procedure call is added before the execution of the task. This procedure is denoted the delay procedure and one parameter is passed to the procedure, i.e. the delay for the task. The length of the delay has to be determined by the user of the method, as a first approach every symbol of the same type is assigned the same delay. The procedure is also shown in figure 8. The procedure delays the execution for the specified delay by use of the timer concept in SDL. It must be noted that all signals are saved within the procedure, and the reason is of course that the transformation may not alter the original functional behaviour. The transformation corresponds to the rule discussed in section 7. In particular the delay concept is illustrated within the task concept.

8.5 Usage Model, general simulation block and Architecture Model

The formulation of the complete simulation model includes modelling the architecture, the usage and describing the content of the general block. The usage is modelled with five processes describing the behaviour of the subscribers, both A- and B-subscribers, as well as a model of a call generator and a monitor which is responsible for creating the B-subscriber when a call is made, see figure 5 and the brief description above.

The processes in the general block are introduced to govern the simulation and to make measurements according to the objectives of the simulation. The measurements are specified so that they shall cover some usual measurement situations.

Finally, the architecture is modelled. This part is the most difficult in the example, since it includes a general data structure which handles the connections between the different models as well as the routing within the Architecture Model. The actual content of the structure is generated by one process and the process modelling the processor then works on the generated structure. The processor process is formulated so that it can handle the structure independently of the actual generation. The architecture processes are briefly described above. The data structure handling the connection between the models is based on linked lists. The structure is shown in figure 9 and it is briefly described below.

The structure consists of three queue types. The first queue (Proc_Queue) is a queue of the processors, i.e. the Architecture Model processes executing software processes (Software Model processes). This queue is the most central one. This queue is created by *Arch_Creator* and it is not altered during the execution of the simulation. For each processor two other queues are created, where the first one (Contact_i) models the connection of process instances from the Software Model and the Usage Model to the Architecture Model and the second one (Route_i) describes the routing. In the example this means five queues in total, i.e. one central queue with two objects (the two processors) and two queues for each of these objects. The routing queue of each processor is also static, while the queues containing process instances connected to the processors are dynamic, i.e. the contents of the queues are changed as new process instances are created or instances are terminated.

8.6 The simulation system

The modelling results in new SDL graphs describing the architecture, the usage and the general utilities. It is also necessary to alter some of the generated Software Model processes. These are changed due to the objective of measuring the load on the processors for each process type. The

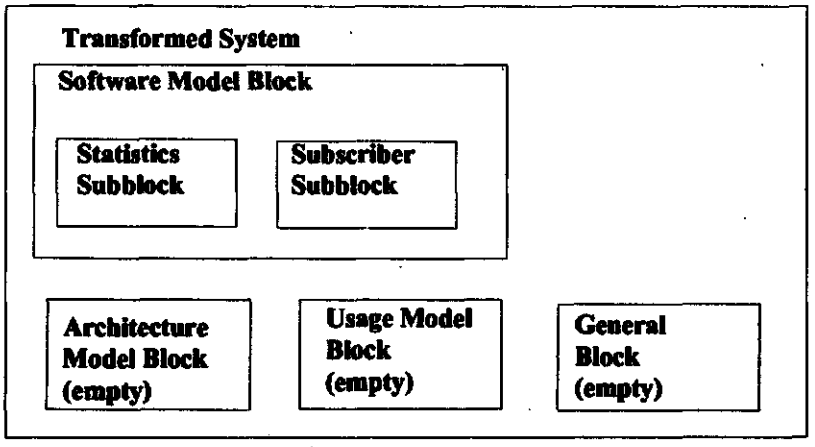

Figure 7: Transformed system.

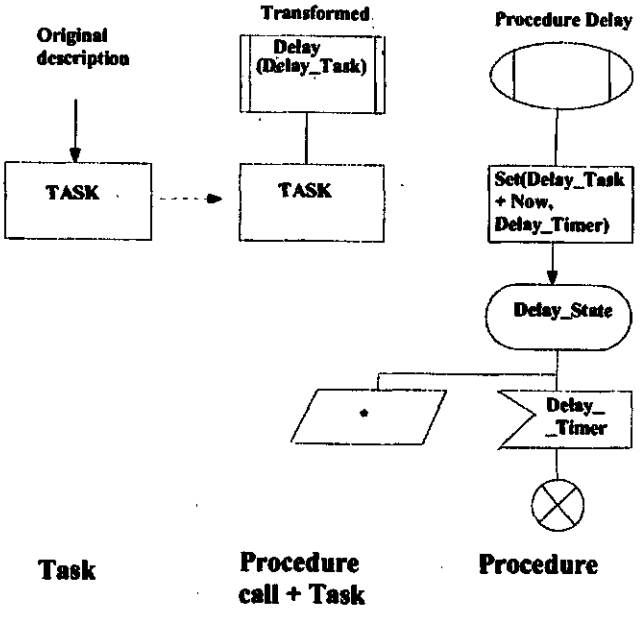

Figure 8: An example of a transformation.

additions are only minor, i.e. the graphs are complemented with a new variable describing the process type, which shall be used to measure the load on the processors for different process types.

8.7 Simulation results

The simulation system can be executed after having been analysed, generated, compiled and linked. It must be noted that the obtained simulation model cannot only be executed for performance and reliability analysis. It can also be used as a real time functional simulator. This means that the methodology provides a way of doing functional simulations in cases where it in the normal case is impossible (see above), i.e. for incomplete system descriptions (that is for example descriptions containing informal text in decision boxes). The transformation and generation facilitates execution of the original not completely specified SDL system from a functional perspective in a real time model environment as well.

The input data are not actual measurement data, but they have, however, been chosen to work as an input set where the relative size between the different inputs are reasonable. The main objective is as pointed out earlier not the actual values, but to show that the simulation actually can be performed based on the proposed concept. The values of the parameters are easily changed since they are declared as external synonyms in SDL.

Three parameters are of particular interest:

– The time, when the A-subscriber thinks he

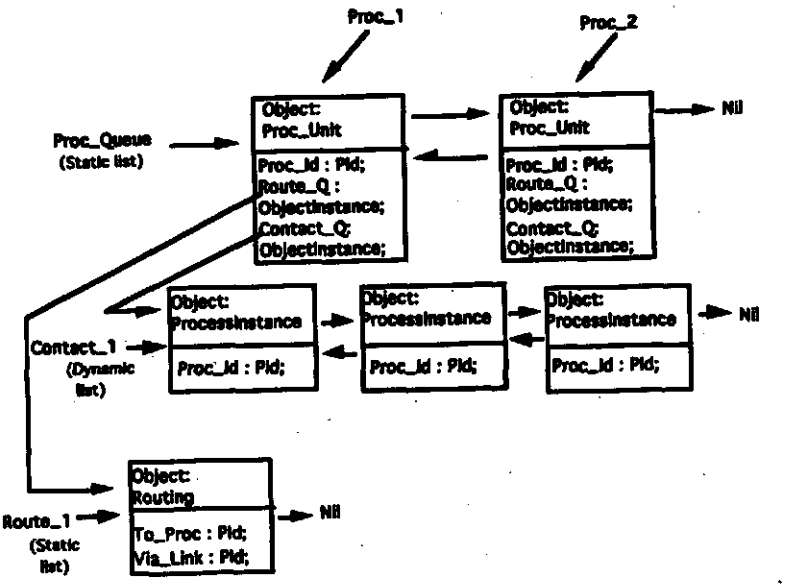

Figure 9: Data structure connecting the modelling concepts.

has waited too long before an answer, is first assigned the value 10, in which case the first output result below was obtained. The value is then changed to be equal to the simulation time.

- The simulation time is set to 1000. This may be too short to obtain real confidence in the output results, but since the actual figures are of minor interest it has been chosen short to obtain the results quickly.
- The mean time between arrival of calls is varied from 10 to 1.5.

The execution of the simulation model of the example gives three valuable results: 1) functional simulation results from a real time model, 2) reliability evaluation in terms of fault detection and 3) performance analysis results.

Functional result

A race between two signals is discovered, i.e. the behaviour of the system becomes different depending on the order of two signals. Due to the delay in the architecture it may happen that a terminate signal has not reached the receiving process instance before it sends a signal to the process that has terminated. This leads to a dynamic failure in the simulation. The original SDL system has been specified so that under some circumstances this will occur. Specifically the problem arises for high loads. A re-design is therefore necessary to cope with this problem, which would have been difficult to find without this simulation.

The functional fault would, without the method, probably not have been found until the load test in the test phase, since timing problems can not be evaluated with "normal" functional simulations. Since the usage profile is input to the simulation, faults will be found according to their probability of occurring in operation. Therefore the method will reveal faults as they are likely to occur in operation hence allowing for a reliability estimate in the same way as can be obtained through operational profile testing [11] and [12], and in statistical usage testing [5] and [6]. The simulation technique proposed advances though the estimation in the software life cycle to the software design phase.

Reliability result

The functional fault found is clearly a reliability problem at high loads. The fault found can hence be regarded as both a functional problem and as a reliability problem. Therefore it can be concluded that the software will at high loads be viewed as not being reliable, while in other cases it will be regarded as being reliable. No other faults were

found in the software during the simulation hence it was concluded that after correcting the functional fault found, the software can be approved, i.e. after regression simulation of the corrected software design.

Performance results

The results from the performance part of the simulation depend on what is specified by the user. The measurements are specified by the user of the system when complementing the generated simulation model with the Architecture Model, the Usage Model and the general processes that govern the simulation (see general block above). The latter has not been discussed in detail, but it is necessary to have some general processes for starting the simulation and perhaps for making measurement. These type of processes are quite simple to formulate and they will not be any problem for the user. In this particular case the load on the communication links was measured as well as the contribution to the total load on the processor by the different software processes. The latter was measured to identify the software processes contributing the most to the load and hence being candidates for re-design.

The results are shown in table 1. The table contains information about the mean time between new calls, the utilization of the two links, the total load on the most used processor, i.e. processor 1. The contribution to the load for the two processes that consume most execution power on processor 1 is also shown in table 1. An example for processor 1, when the mean arrival time between calls is 10; the figure 45 stands for the time process *A_Handler* executes and the time 176 is the total execution time used on the processor, i.e. out of 1000, which is the simulation time.

Some comments to the results in table 1 are worth making, even if the actual figures are of minor interest. It can be seen that the link from processor 2 to processor 1 is utilized more than the other, i.e. link 2. The reason is that the *Statistics* process is only located on processor 1. It can also be seen that process *A_Handler* is the largest contributor to the load. It is responsible for between 26-29% of the load, which means that a re-design of the process perhaps ought to be considered. The *Statistics* process contributes also very much and this is probably a problem, since the statistics in an exchange can be hard to motivate to the subscribers. A solution would be to distribute the statistics to all processors, and this will also cut down the utilization of the communication link between processor 2 and processor 1.

The obtained results show that valuable information relating to software quality can be obtained in the design phase with the proposed evaluation method.

9 Concluding remarks

Well-defined, formal or standardised description techniques provide excellent opportunities for automatic transformations to other representations. The other representation can either be a step in the development life cycle or a special representation for evaluating one or several quality attributes of the system. The presented method can be applied to both functional and object oriented description techniques. The quality attributes of the systems of today are becoming critical issues as the systems are getting larger, more complex and more critical. This means that techniques and methods for analysis of system quality attributes are needed to stay in control of the software system being developed.

This article has considered how software design descriptions and simulation models can be used to evaluate the performance and the reliability as well as the functional behaviour of the system at an early stage. In particular, the method provides an evaluation procedure before the coding and the testing phases.

The method provides a basis for:

- functional simulation in a real time model,
- estimation of software reliability in a simulation environment, when the usage in the simulation model is generated according to the operational profile,
- identifying software bottlenecks at an early stage,
- evaluating different distributions of software processes in an architecture,
- studying the introduction of new services in an existing system (network),
- examining different architectures ability to execute a given software description,

Mean arrival time calls	Utilization link 1	Utilization link 2	Load Pro. 1	Processor 1 A_Handler	Processor 1 Statistics
10	0.06	0.17	0.18	45(176)	51(176)
7	0.10	0.24	0.24	64(240)	61(240)
5	0.16	0.22	0.31	84(306)	72(306)
3	0.24	0.42	0.51	144(509)	106(509)
2	0.35	0.59	0.70	197(698)	140(698)
1.5	0.54	0.86	0.98	283(981)	181(981)

Table 1: Performance simulation results

– identifying system bottlenecks.

These issues will become important aspects as the demands on new services and systems grow in the same time as the requirements on short software development times and higher productivity continue to grow. The above list will be particularly important for safety-critical software systems, where a failure may be catastrophic. Part of the solution to meet the high requirements on functionality, performance and reliability is most certainly to put more emphasis on the early phases of the system life cycle through introduction of well-defined description techniques and methods that support different aspects of the development process. It is believed that methods for automatic translations of software descriptions into other representations will be one of the key issues to cope with the productivity and quality problems of software systems. The presented method provides an opportunity to tackle the problem of early verification of performance, reliability and functionality, as well as for doing capacity dimension

This method, or a similar one, is needed to control the quality attributes before the software system has been coded. The quality of the software must be evaluated and assured during early development. The presented method is a step in the right direction, towards a comprehensive view on quality control of software products.

Acknowledgement

Many thanks to Per Runeson, Q-Labs and Christian Nyberg, Dept. of Communication Systems, Lund, Sweden, for valuable help and constructive improvement suggestions concerning the article.

References

[1] R. L. Bagrodia and C-C. Shen, *MIDAS: Integrated Design and Simulation of Distributed Systems*. IEEE Transactions on Software Engineering, Vol. 17, No. 10, pp. 1042-1058, (1991).

[2] F. Belina F., D. Hogrefe, and A. Sarma, *SDL with Applications from Protocol Specifications*, Prentice-Hall, UK, (1991).

[3] CCITT, *Recommendation Z.100: Specification and Description Language, SDL*, Blue book, Volume X.1, (1988).

[4] CCITT, *SDL Methodology Guidelines*, Appendix I to Z.100, (1992).

[5] R. H. Cobb and H. D. Mills, *Engineering Software Under Statistical Quality Control*, IEEE Software, pp. 44-54, November, (1990).

[6] M. Dyer, *The Cleanroom Approach to Quality Software Development*, John Wiley & Sons, (1992).

[7] E. Heck, D. Hogrefe and B. Müller-Clostermann, *Hierarchical Performance Evaluation Based on Formally Specified Communication Protocols*, IEEE Transactions on Communication, Vol. 40, No. 4, pp. 500-513, (1991).

[8] M. Håcansson and Ö. Persson, *Performance Simulation for SDT*, Master thesis, CODEN:LUTEDX(TETS-5176)/1-66/(1993) & Local 26, Dept. of Communication Systems, Lund Institute of Technology, Lund University, Lund, Sweden, (1993).

[9] B. D. Juhlin, *Implementing Operational Profiles to Measure System Reliability*, Proceedings 3rd International Symposium on Soft-

ware Reliability Engineering, pp. 286-295, Raleigh, North Carolina, USA, (1992).

[10] H. D. Mills, M. Dyer, and R. C. Linger, *Cleanroom Software Engineering*, IEEE Software, pp. 19-24, September, (1987).

[11] J. D. Musa, *Software Reliability Engineering: Determining the Operational Profile*, Technical Report AT & T Bell Laboratories, Murray Hill, NJ 07974, New Jersey, USA, (1992).

[12] J. D. Musa, *Operational Profiles in Software Reliability Engineering*, IEEE Software, pp. 14-32, March, (1993).

[13] P. Runeson and C. Wohlin, *Usage Modelling: The Basis for Statistical Quality Control*, Proceedings 10th Annual Software Reliability Symposium, pp. 77-84, Denver, Colorado, USA, (1992).

[14] M. Véran and D. Potier, *QNAP2: A Portable Environment for Queueing Systems Modelling*, Technical report, Bull, France and INRIA, France, (1984).

[15] W. Whitt, *The Queueing Network Analyzer*, The Bell System Technical Journal, pp. 2779-2815, November, (1983).

[16] J. A. Whittaker and J. H. Poore, *Statistical Testing for Cleanroom Software Engineering*, Proceedings 25th Annual Hawaii International Conference on System Sciences, pp. 428-436, Hawaii, USA, (1992).

[17] C. Wohlin, *Software Reliability and Performance Modelling for Telecommunication Systems*, Dept. of Communication Systems, Lund Institute of Technology, Lund University, Lund, Sweden, Ph.D Dissertation, (1991).

[18] C. Wohlin, *The Performance Prototyping Simulator Concept*, Technical report, IT4-project "Specification with Prototyping and Simulation", TeleLogic, Sweden (1991).

[19] C. Wohlin and P. Runeson, *A Method Proposal for Early Software Reliability Estimations*, Proceedings 3rd International Symposium on Software Reliability Engineering, pp. 156-163, Raleigh, North Carolina, USA, (1992).

SCHEDULING STRATEGIES IN HIGH-LEVEL SYNTHESIS

Jurij Šilc

Jožef Stefan Institute, Laboratory for Computer Architectures
Jamova 39, Ljubljana, Slovenia
E-mail: jurij.silc@ijs.si

Keywords: high-level synthesis, scheduling, allocation

Edited by: Matjaž Gams
Received: May 3, 1993 **Revised:** March 11, 1994 **Accepted:** March 13, 1994

The paper describes objectives of high-level synthesis. It concentrates on operation scheduling strategies and the interaction with the resource allocation. Some transformational and iterative/constructive scheduling algorithms are described. Moreover, a new scheduling/allocation approach is presented and compared with other known algorithms. Finally, some open problems of the high-level synthesis are given.

1 Introduction

The high-level synthesis task is to take a specification of the behavior required of a system and a set of constraints and goals to be satisfied, and to find a structure that implements the behavior while satisfying the goals and constraints. In recent years there has been a trend toward automating synthesis at higher and higher levels of the design hierarchy. There are a number reasons for this: shorter design cycle, fewer errors, the ability to search the design space, documenting the design process, and availability of IC technology to more people [28].

The roots of high-level synthesis can be traced back to the 1960s [15]. During the 1970s most of the effort went into automating tasks at lower levels of the design hierarchy, such as layout. Great progress was made in the development of algorithms and techniques [4, 51]. In the 1980s work on high-level synthesis started to spread from the academic community to industry. High-level synthesis systems are now producing manufacturable chip designs for applications such as signal processing [10], pipelined processors [32], and interfaces [7]. However, there are still many unanswered questions related to such issues as specification, input/output, designer intervention, complex timing constraints, and the relation of synthesis to the overall design and fabrication process.

The paper starts with the description of high-level synthesis structure and then concentrates on scheduling which seems to be the most important step during the synthesis. In particular, some of the basic scheduling techniques are discussed.

2 High-Level Synthesis

Given a *system*, its *structural* description is a specification of a set of components and their interconnections. More recently, however, *behavioral* descriptions of systems are used. Such a description specifies what the system needs to do, i.e. the way that each of the systems components interacts with its environment.

High-level synthesis transforms behavioral description to the structural one. A typical way of describing behavior is to write a program in an ordinary computer language or in a special *hardware description language.*

The first step in high-level synthesis is usually the compilation of the hardware description language into an internal representation. Most approaches use *graph-based* representations that contain both the *data* flow and the *control* flow implied by the specification. Control dependencies are derived directly from the explicit order given in the program and from the compiler's choice of parsing the arithmetic expressions. Data dependencies show the essential ordering of opera-

tions. Some important tasks that should be performed by the compiler at this stage include variable disambiguation, taking care of the scope of variables, converting complex data structures into simple types, and type checking. Moreover, some optimizing transformations may be done at this stage, such as expression simplification. These graphs are given different names in different synthesis systems (e.g. value trace [47], data dependency graph [2], directed acyclic graph [14], control and data flow graph [17]) but are simply different adaptations of similar basic concept. In many systems, the control and data flow graphs are integrated into one structure. In this paper we will use the term *flow graph*. Before proceeding to the second step it is desirable to do some initial optimization of the flow graph, such as dead code elimination, constant propagation, common subexpresion elimination, and loop unrolling.

The second step of the high-level synthesis, which is the core of transforming behavior into structure, includes operation *scheduling* and hardware *allocation*. Since these two tasks are essential in high-level synthesis they have been studied extensively and a variety of algorithms have been published. An excellent overwiev of the different schools of thought has been given in [28]. The scheduling and allocation are closely interrelated. In order to have an optimal design, both tasks should be performed simultaneously [19]. However, due to the time complexity, many systems perform them separately [10, 23, 27, 30, 48, 50] or introduce iteration loops between the two subtasks [17, 33, 35, 45]. Scheduling involves assigning the operation to so-called *control steps*. A control step is the fundamental sequencing unit in synchronous systems; it corresponds to a *clock cycle*. (Different methods for scheduling will be examined in detail in the following sections.) Allocation involves assigning the operations and values to hardware, i.e., providing storage, function units, and communication paths, and specifying their usage. To minimize them together is usually too complex, so in many high-level synthesis systems they are minimized separately. Therefore, allocation is usually further divided into three subtasks – *variable binding*, *operation assignment*, and *data transfer binding*. Variable binding refers to the allocation of registers to data, i.e., values that are generated in one control step and used in another must be assigned to registers. Some systems have a one-to-one correspondence between variables and registers [42], while others allow register sharing for those variables which have disjoint lifetimes [34, 50]. Operation assignment binds operations (e.g., addition) to function units (e.g., an adder or an ALU). Of course, operations can share functional units only if they are mutually exclusive, that is, they are assigned to different control steps. The problem is then to form the minimum number of groups consisting of mutually exclusive operations since this will minimize the number of function units. Data transfer bindings represent the allocation of connections (e.g., busses, multiplexers) between hardware components (i.e., registers and function units) to create the necessary information paths as required by the specification and the schedule. Connections consist of busses and/or multiplexers. Busses offer the advantage of requiring less wiring, but they may be slower than multiplexers. A combination of both is often the best solution.

Once the schedule and allocation have been accomplished, it is necessary to synthesize a *controller* (hardwired or microcoded) that will drive allocated resources as required by the schedule. Finally, the design has to be converted into real hardware. Lower level tools such as *logic* synthesis and *layout* synthesis complete the design.

3 Scheduling Strategies

As noted earlier, a good scheduler is very important to a high-level synthesis system. There are three dimensions along which scheduling algorithms may differ:

1. the objective function and constraints that algorithms use;
2. the interaction between scheduling and allocation;
3. the type of scheduling algorithm used.

3.1 Constraints

Roughly speaking, operation scheduling determines the cost-speed tradeoffs of the design. A *time-constrained* scheduling problem can be defined as follows: *given the maximum number of control steps, find a minimal cost schedule that satisfies the given set of constraints.* Here the

cost may consist of the costs of function units, connections, and registers. Some systems that perform time-constrained scheduling are HAL [34, 35], MAHA [33], and Sehwa [32]. A *resource-constrained* scheduling problem is stated as follows: *given the maximum number of resources, find the fastest schedule that satisfies the given set of constraints.* Until recently, the *resources* included only function units. Lately, connections and registers are also taken into consideration. Some systems that perform resource-constrained scheduling are CMUDA [12, 18, 50], MIMOLA [27, 51], MAHA [33], and Sehwa [32]. The previous two formulations can be combined into a *feasible* scheduling problem [22]: *given a fixed amount of resources and a specified number of time steps, decide if there is a schedule which satisfies all the constraints, and output the solution if it exists.* A system that performs feasible-constarined scheduling is BUD [29].

If the design is subject to a time-constraint, the scheduling algorithm will attempt to parallelize the operations to meet the timing constraint. Conversely, if there is a limit on the cost of resources, the scheduler will serialize operations to meet the resource-constraint.

3.2 Interaction with Allocation

In order to know whether two operations can be scheduled in the same control step, one must know whether they use common hardware resources. Moreover, finding the most efficient possible schedule for the real hardware requires knowing the delays for the different operations, an those can only be found *after* the details of the function units and their interconnections are known. On the other hand, in order to make a good allocation, one must know what operations will be done in parallel, which comes form the schedule. Therefore, scheduling and allocation are strongly interdependent tasks.

The most straightforward approach to this problem is to set some limit (or no limit at all) on the resource cost and then to schedule, as it is done in systems CMUDA [12, 18, 50], Flamel [48], and V [5]. A more flexible approach is to iterate the whole process changing the resource limits until a satisfactory design has been found. This approach is used in MIMOLA [27, 51] and Sehwa [32]. Another approach is to develop the schedule and allocation simultaneously, as in systems HAL [34, 35] and MAHA [33]. Some recent approaches formulate scheduling and allocation together as an optimization problem to be solved by general optimization techniques such as simulated annealing [3, 11, 41] or integer programming [22]. Finally, the allocation can be done first, followed by scheduling, as it is the case in BUD system [29].

3.3 Scheduling Algorithms

The simplest way to perform scheduling is to relegate the task to the user, which is the approach favored by the Silc system [6]. There is, however, a trend toward automated scheduling. Such algorithms can be classified into *transformational* or *iterative/constructive* algorithms.

A transformational type of algorithm starts with an initial schedule (e.g., maximally serial or maximally parallel) and applies transformations to it to obtain other schedules. These algorithms differ in how they choose transformations (e.g., using *exhaustive* search [4], *branch-and-bound* [19], or some *heuristics* [37]).

The other type of algorithms, the iterative/constructive ones, build up a schedule by adding operations one at a time till all the operations have been scheduled. These algorithms differ in how the next operation to be scheduled is chosen and into which control step it is put. The simplest way is to schedule operations *as soon as possible* (ASAP) as is done in the Facet [50], early CMUDA [18], MIMOLA [27, 51], and Flamel [48] systems. ASAP assigns each operation to earliest possible control step such that data and control dependencies allow it to execute. A similar approach is to schedule operations *as late as possible* (ALAP). The problem with ASAP and ALAP scheduling is that when there are limits on resource usage no priority is given to operations on critical paths. Hence, less critical operations can be scheduled first and thus block critical ones [39]. Continuing along the scale of increasing complexity, there are algorithms that use *list scheduling.* For each control step, the operations available to be scheduled into that step are kept in a list, which ordered by some *priority function.* Each operation on the list is scheduled if the resources it need are still free in that step; otherwise it is deferred to the next step. In some cases, this form

of scheduling works nearly as well as branch-and-bound. Schedulers differ in the priority function they use. A priority function may use the length of the *longest path* from the operation to the end of graph [39, 40, 43]. This is approach taken in the BUD system [29]. Elf system [17] uses the *urgency* of the operation, i.e. the length of the shortest path from the operation to nearest local time constraint. In Slicer system [30] the priority function is based on increasing operation *mobilities*, i.e., differences between ASAP and ALAP times of operations. A composite priority is used in MAHA system [33] where the operations on critical paths are scheduled first (and also assigned to function units). Then the other operations are scheduled (and assigned) one at a time according to the least mobility. The HAL system [34, 35] does list scheduling with *force* as a priority. The force between an operation and a particular control step is proportional to the number of operations of the same type that could be scheduled into that step. To conclude, in list scheduling operations that might present more difficult scheduling problems are taken care of first.

In what follows we will briefly describe some known scheduling algorithms. First we give some common definitions. Let $G(V, A)$ be a flow graph, where V is the set of operations and A is the set of dependencies (arcs), which is to be scheduled into s control steps. Let $n = |V|$ and $a = |A|$. Each of the operations is labeled as o_i, $1 \leq i \leq n$. A precedence relation between operations o_i and o_j is denoted by $o_i \rightarrow o_j$, where o_i is immediate predecessor of o_j. The earliest possible start time and the latest possible start time of o_i are S_i and L_i, respectively. There are m types of function units available. A function unit of type t is denoted by F_t. A relation between operation o_i and a function unit F_t is denoted by $o_i \in F_t$, if F_t can perform o_i.

Integer Linear Programming Algorithm
In [22] integer linear programming ILP is used to formulate the feasible scheduling problem. Let the cost of a function unit of type t be c_t and M_t be integer variables denoting the number of function units of type t needed. Finally, let $x_{i\tau}$ be $0-1$ integer variables where $x_{i\tau} = 1$ if o_i is scheduled into control step τ; otherwise, $x_{i\tau} = 0$. Assuming a one-cycle propagation delay for each operation and a nonpipelined execution, the feasible scheduling problem can finally be stated as follows:

$$\sum_{o_i \in F_t} x_{i\tau} \leq M_t, \quad 1 \leq \tau \leq s, 1 \leq t \leq m;$$

$$\sum_{\tau=S_i}^{L_i} x_{i\tau} = 1, \quad 1 \leq i \leq n;$$

$$\sum_{\tau=S_i}^{L_i} \tau * x_{i\tau} - \sum_{\tau=S_k}^{L_k} \tau * x_{k\tau} \leq -1, \quad o_i \rightarrow o_k \in A.$$

The first constraint states that no schedule should have a control step containing more than M_t function units of type t. It is clear that o_i can only be scheduled into a step between S_i and L_i, which is reflected in the second constraint. The third constraint ensures that precedence relations of the flow graph will be preserved. The objective function is a combination of time-constraint objective function $\min \sum_{t=1}^{m} c_t * M_t$ and resource-constraint objective function $\min C_{step}$, where C_{step} is total number of control steps required. This approach allows the user to control the resource-time tradeoff. More explicit resource-time tuning is the advantage of the next algorithm to be presented.

Simulated Annealing Based Algorithm
Another type of transformational feasible scheduler based on the simulated annealing idea is given in [3]. The simulated annealing algorithm can be used for combinatorial optimization problems specified by a finite set of configurations and a cost function defined on all the configurations. The algorithm randomly generates a new configuration which is then accepted or rejected according to a random acceptance rule governed by the parameter analogous to temperature in the physical annealing process [26]. Algorithm starts on an initial configuration which is the schedule obtained by applying ASAP strategy, i.e. the start time of o_i is S_i, for each $o_i \in V$. The function *Cost* evaluates how good a configuration is. It is defined as $Cost(X) = \alpha Area(X) + \beta Time(X)$, where $Area(X)$ is the estimated total area of the resources used and $Time(X)$ is the total execution time corresponding to the given configuration X. The tuning of the algorithm is performed

by taking different values for α and β. For example, if $\alpha \ll \beta$ the algorithm is closer to resource-constrained scheduler (since solutions efficient in speed become more important) while $\alpha \gg \beta$ makes the algorithm more time-constrained. Initially, a high temperature $T_{initial}$ is given in order to accept most new configurations even if they increase the cost. As temperature decreases, less configurations are accepted unless they have improved cost. Given a configuration X, a new configuration Y is generated either by insertion or removal of a register, scheduling an operation to next or previous control step, or by shrinking/expanding a control step. A similar algorithm appears in [11] where it is also reported that the algorithm achieves excellent results. However, it performs scheduling and allocation simultaneously. This is also the characteristics of the approach which is to be presented next.

Force Directed Algorithm Let us first describe a force-directed scheduling algorithm which is based on list scheduling with a force as a priority function. The first step consists of determining the time frames $[S_i, L_i]$ of each operation $o_i \in V$. Let $p_{i\tau}$ denote the probability that o_i will be scheduled into control step $\tau \in [S_i, L_i]$. A useful heuristic is to assume a uniform probability, i.e. $p_{i\tau} = \frac{1}{1+L_i-S_i}$. The next step is to take the summation of the probabilities of each type t of operation for each control step τ: $P(t,\tau) = \sum_{o_i \in F_t} p_{i\tau}$. The final step is to calculate the force $\mathcal{F}$ associated with each operation o_i and bounded to a temporarily reduced time frame $[S_i', L_i']$:

$$\mathcal{F}(S_i', L_i') = \sum_{\tau=S_i'}^{L_i'} \frac{P(t,\tau)}{1+L_i'-S_i'} - \sum_{\tau=S_i}^{L_i} \frac{P(t,\tau)}{1+L_i-S_i},$$

where t is the type of the operation o_i. Once all the forces are calculated, the operation-control step pair with largest negative force (or least positive force) is scheduled. Then P and $\mathcal{F}$ values are updated. The process is repeated untill all operations are scheduled. The scheduling process described above is a part of the HAL system [34, 35]. In particular the scheduling/allocation is performed simultaneously by stepwise refinement in an iterative loop consisting of four phases. The first phase (default allocation) allocates single-function processor to each type of operation. The second phase (preliminary schedule) balances the distribution of similar of operations using force-directed scheduling. The third phase (refined allocation) allocates single and multi-function processors. The last, fourth phase (final schedule) balances the distribution of operations requiring similar processor types.

4 Global Arc Minimization Algorithm

In last section we briefly described three approaches to the scheduling/allocation problem. In this section we present a new algorithm named *Global Arc Minimization* (GAM). The algorithm was developed by the author [44, 45, 46] where both time and resource constrained scheduling algorithms were described.

In this paper, however, we will concentrate only on the time constrained scheduling. We consider situation with $m = 1$, i.e., all function units are of the same type (as it is the case in preliminary scheduling in HAL system). The first step consists of determining the time frames $[S_i, L_i]$ of each operation $o_i \in V$ using ASAP and ALAP schedules. Next, the minimum number f_{min} of function units needed is evaluated. To do this, a method based on the *extended critical parallelism* of flow graph was introduced in [44]. During the execution of the scheduling algorithm the following sets of operations are maintained at each control step $\tau = 1, 2, \ldots, s$:

$\mathbf{O}_{ready}(\tau) := \{o_i | S_i \leq \tau \leq L_i\}$,
$\mathbf{O}_{urgent}(\tau) := \{o_i | S_i = \tau = L_i\}$,
$\mathbf{O}_{deferrable}(\tau) = \mathbf{O}_{ready}(\tau) - \mathbf{O}_{urgent}(\tau)$, and
$\mathbf{O}_{finished}(\tau) := \{o_i \in V | o_i$ has finished at $\tau\}$.

Let $f(\tau)$ denote the number of occupied function units at some control step τ. Hence, there are $f_{min} - f(\tau)$ free function units at τ. Since urgent operations are always taken care of first, in case of $f(\tau) < f_{min}$ some additional operations can be started. Note, that these operations are selected among deferrable ones. Selection was performed according to three priority functions: random selection, increasing execution time selection, and decreasing execution time selection. Since none of these criteria proved to be superior [45], we used the random selection strategy. Therefore, the algorithm is of a list-scheduling type and is given below:

$\tau = 0$
$f(\tau) = 0$
repeat
  if $f(\tau) > 0$ **then**
    $f(\tau) = f(\tau) - |\mathbf{O}_{finished}(\tau)|$
  endif
  $f(\tau) = f(\tau) + |\mathbf{O}_{urgent}(\tau)|$
  if $f(\tau) < f_{min}$ **then**
    Let $\mathbf{O}_{additional}(\tau) \subset \mathbf{O}_{deferrable}(\tau)$,
    where $|\mathbf{O}_{additional}(\tau)| \leq f_{min} - f(\tau)$.
    $f(\tau) = f(\tau) + |\mathbf{O}_{additional}(\tau)|$
  endif
  $\tau = \tau + 1$
until $\tau = s$

After scheduling has been completed the allocation is performed, i.e., the index $\varphi(o_i)$, $1 \leq \varphi(o_i) \leq f$, $f = \max_{1 \leq \tau \leq s} f(\tau)$ of a function unit is computed for each operation $o_i \in V$. Since the communication between operations allocated to different function units is a time consuming operation, the goal is to allocate operations so that the communication time is minimized. Let us call an arc $o_i \rightarrow o_j$ a *global arc* if $\varphi(o_i) \neq \varphi(o_j)$, i.e., if operations o_i and o_j are allocated different function units. In order to minimize communication time an allocation criterion which keeps the number of global arcs as low as possible was successfully applied. Namely, the allocation problem can be transformed into the *weighted bipartite-matching* problem [21]. The global arc minimization algorithm GAM is described in [46].

As we already mentioned, we have designed both time and resource constrained scheduling algorithms. Together these algorithms can be used to solve the feasible scheduling problem. In particular, the total cost of scheduling/allocation can be defined as a function $Cost(s, f, c)$ of the number of control steps s, the number of function units f, and the communication time c. Now, given some default number of control steps s (determined by the critical path of the flow graph), the time constrained scheduling/allocation (as described above) results in f function units and the total communication time c. Next, we iterativelly apply the resource constrained scheduling/allocation with $f - k$, $k = 1, 2, \ldots$ function units. Finally, we chose the most appropriate k, i.e. the scheduling with the lowest $Cost$. One can also iterativelly repeat the whole process starting at higher values of s. (Note, that the process may stop when s equals the sequential execution time of the flow graph.)

Hence, our algorithm iterates the scheduling/allocation process changing the resource limits until a satisfactory design has been found. Recall, that a similar approach has been taken in MIMOLA [27, 51] and Sehwa [32].

5 Experimental results and comparitions

The scheduling/allocation approach GAM has been implemented and tested [45]. The flow graph used in this example implements a fifth-order wave digital elliptic filter.

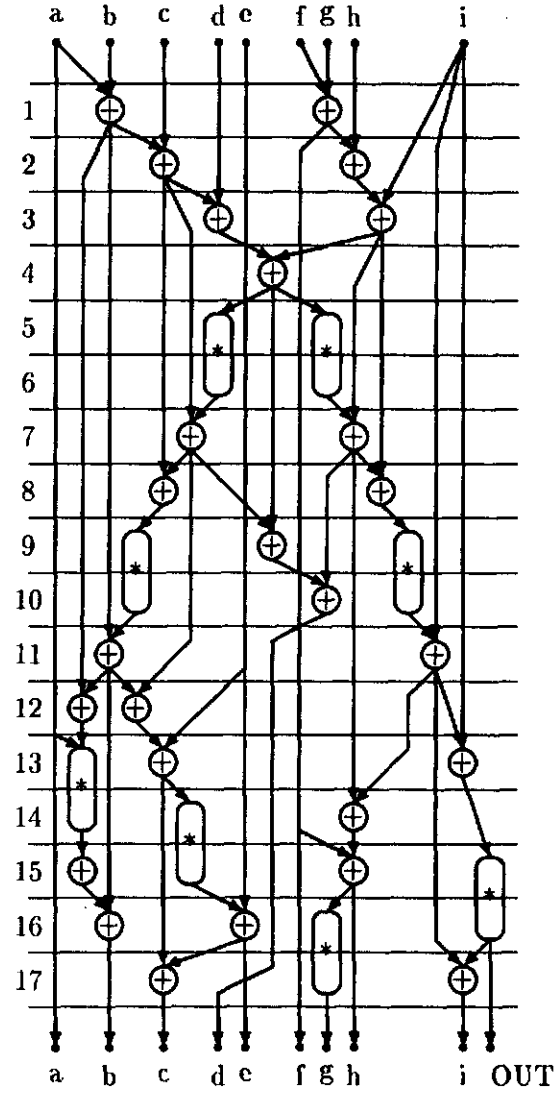

Figure 1: Elliptic filter.

We assume that multipliers require two control

Table 1: Comparision of scheduling results for the elliptic filter.

	ASAP	ALAP	ILP [22]		HAL [35]		GAM	
Adders	4	4	3	2	3	2	2	2
Multipliers	4	3	3	1	3	1	2	1
Control steps	17	17	17	21	17	21	17	21

steps for execution and the adders only one. The critical path length is thus 17 control steps long. Figure 1 shows the results of applying GAM algorithm on the elliptic filter. Note, that in case of time constraint scheduling only 2 multipliers and 2 adders were used.

Finally, Table 1 shows the scheduling results for the elliptic filter using both approaches from section 3 and our GAM algorithm.

6 Concluding remarks

The problem of translating behavioral description of a system into structural a one has been divided into a number of subtasks among which the operation scheduling and hardware allocation are the most important.

The scheduling problem has been researched quite extensively in the past [1, 9, 13, 16, 20, 24]. However, most of the solutions concentrate on systems with homogeneous function units. Moreover, neither of the efforts includes communication overhead. In order to be more realistic the communication delay has to be considered in high-level synthesis. Since allocation involves assigning the operations to hardware, it also determines the communication overhead. Thus, in high-level synthesis the scheduling and allocation are closely interrelated [5, 12, 19, 27, 29, 32, 33, 34, 38, 48, 50, 51]. In order to have an optimal design, both should be performed simultaneously. Due to the time complexity, however, many systems perform them separately, or introduce iteration loops between the two tasks, as it was the case in our GAM scheduling/allocation approach.

We may conclude that the key tasks of scheduling and allocation are relatively well understood since there are a variety of effective techniques that have been applied to them. However, there are many other areas where high-level synthesis must continue to develop if it is to become a useful tool for designing computer systems. Such areas include specification, designer intervention, input/output, complex timing constraints handling, and the relation of synthesis to the overall design and fabrication process [28].

References

[1] Adam T. L., Chandy K. M., and Dickson J. R. (1974) A Comparision of List Schedulers for Parallel Processing Systems. *Comm. ACM*, **17**, 12, p. 685-690.

[2] Allen J. (1985) Computer Architecture for Digital Signal Processing. *Proc. of the IEEE*, **73**, 5, p. 852-873.

[3] Badia R. M., Cortadella J., and Ayguadé E. (1992) Computed-Aided Synthesis of Data-Path by Using a Simulated-Annealing-Based Approach. *Proc. 9th IASTED Int'l Symp. Applied Informatics*, p. 326-329.

[4] Barbacci M. R. (1973) Automated Exploration of the Design Space for Register Transfer (RT) Systems. *Ph.D. Thesis. Carnegie-Mellon University.*

[5] Berstis V. (1989) The V Compiler: Automating Hardware Design. *IEEE Design and Test*, **6**, 2, p. 8-17.

[6] Blackman T. et al. (1985) The Silc Silicon Compiler: Language and Features. *Proc. 22th ACM/IEEE Design Automation Conf.*, p. 232-237.

[7] Borriello G. and Katz R. H. (1987) Synthesis and Optimization of Interface Transducer Logic. *Proc. ICCAD*, p. 274-277.

[8] Brewer F. D. and Gajski D. D. (1987) Knowledge Based Control in Micro-Architecture Design. *Proc. 24th ACM/IEEE Design Automation Conf.*, p. 203-209.

[9] Coffman E. G. and Denning P. J. (1973) Operating Systems Theory. *Prentice-Hall*, Englewood Cliffs.

[10] DeMan H., Rabaey J., Six P., and Claesen L. (1986) Cathedrall II: A Silicon Compiler for Digital Signal Processing. *IEEE Design and Test*, **3**, 6, p. 13-25.

[11] Devadas S. and Newton A. R. (1989) Algorithm for Hardware Allocation in Data Path Synthesis. *IEEE Trans. CAD*, **8**, 7, p. 768-781.

[12] Director S. W., Parker A. C., Siewiorek D. P., and Thomas D. E. (1981) A Design Methodology and Computer Aids for Digital VLSI Systems. *IEEE Trans. Circuits Sys.*, **28**, 7, p. 634-645.

[13] Fernandez E. B. and Bussell B. (1973) Bounds on the Number of Processors and Time for Multiprocessor Optimal Scedules. *IEEE Trans. Computers*, **22**, 8, p. 745-751.

[14] Frank G. A., Franke D. L., and Ingogly W. F. (1985) An Architecture Design and Assessment System. *VLSI Design*, August, p. 30-50.

[15] Friedman T. D. and Yang S. C. (1969) Methods used in an Automatic Logic Design Generator (ALERT). *IEEE Trans. Computers*, **18**, p. 593-614.

[16] Garey M. R., Graham R. L., and Johnson D. S. (1978) Performance Guarantees for Scheduling Algorithms. *Oper. Res.*, **26**, 1, p. 3-21.

[17] Girczyc E. F. and Knight J. P. (1984) An ADA to Standard Cell Hardware Compiler Based on Graph Grammars and Scheduling. *Proc IEEE Int'l Conf. Computer Design*, p. 726-731.

[18] Hafer L. J. and Parker A. C. (1978) Register-Transfer Level Digital Design Automation: The Allocation Process. *Proc 15th ACM/IEEE Design Automation Conf.*, p. 213-219.

[19] Hafer L. J. and Parker A. C. (1983) A Formal Method for the Specification, Analysis, and Design of Register-Transfer Level Digital Logic. *IEEE Trans. CAD*, **2**, 1, p. 4-18.

[20] Hu T. C. (1961) Parallel Sequencing and Assembly Line Problems. *Oper. Res.*, **9**, p. 841-848.

[21] Huang C. Y., Chen Y. S., Lin Y. L. and Hsu Y. C. (1990) Data Path Allocation Based on Bipartite Weihted Matching. *Proc 27th ACM/IEEE Design Automation Conf.*, p. 499-504.

[22] Hwang C. T., Lee J. H., and Hsu Y. C. (1991) A Formal Approach to the Scheduling Problem in High Level Synthesis. *IEEE Trans. CAD*, **10**, 4, p. 464-475.

[23] Jansen K. (1993) The Allocation Problem in Hardware Design. *Discrete Applied Mathematics*, **43**, p. 37-46.

[24] Kasahara H. and Narita S. (1984) Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing. *IEEE Trans. Computers*, **33**, 11, p. 1023-1029.

[25] Kurdahi F. J. and Parker A. C. (1987) REAL: A Program for REgister ALlocation. *Proc 24th ACM/IEEE Design Automation Conf.*, p. 210-215.

[26] van Laarhoven P. M. J. and Aarts E. H. L. (1987) Simulated Annealing: Theory and Applications. *Kluwer Academic Publ. Group*, Dordrecht.

[27] Marwedel P. (1986) A New Synthesis Algorithm for the MIMOLA Software System. *Proc 23rd ACM/IEEE Design Automation Conf.*, p. 271-277.

[28] McFarland M. C., Parker A. C., and Camposano R. (1990) The High-Level Synthesis of Digital Systems. *Proc. of the IEEE*, **78**, 2, p. 301-318.

[29] McFarland M. C. (1986) Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions. *Proc 23rd ACM/IEEE Design Automation Conf.*, p. 474-480.

[30] Pangrle B. M. and Gajski D. D. (1987) Slicer: A State Synthesizer for Intelligent Silicon Compilation. *Proc IEEE Int'l Conf. Computer Design: VLSI in Computers & Processors.*

[31] Pangrle B. M. and Gajski D. D. (1987) Design Tools for Intelligent Silicon Compilation. *IEEE Trans. CAD*, **6**, 6, p. 1098-1112.

[32] Park N. and Parker A. C. (1986) SEHWA: A Program for Synthesis of Pipelines. *Proc 23rd ACM/IEEE Design Automation Conf.*, p. 454-460.

[33] Parker A. C., Pizarro J., and Mlinar M. (1986) MAHA: A Program for Datapath Synthesis. *Proc 23rd ACM/IEEE Design Automation Conf.*, p. 461-466.

[34] Paulin P. G., Knight J. P., and Girczyc E. F. (1986) HAL: A Multi-Paradigm Approach to Automatic Data Path Synthesis. *Proc 23rd ACM/IEEE Design Automation Conf.*, p. 263-270.

[35] Paulin P. G. and Knight J. P. (1987) Force-Directed Scheduling in Automatic Data Path Synthesis. *Proc 24th ACM/IEEE Design Automation Conf.*, p. 195-202.

[36] Paulin P. G. (1989) Scheduling and Binding Algorithms for High-Level Synthesis. *Proc 26th ACM/IEEE Design Automation Conf.*, p. 1-6.

[37] Peng Z. (1986) Synthesis of VLSI Systems with the CAMAD Design Aid. *Proc 23th ACM/IEEE Design Automation Conf.*, p. 278-284.

[38] Potkonjak M. and Rabaey J. (1989) A Scheduling and Resource Allocation Algorithm for Hierarchical Signal Flow Graphs. *Proc 26th ACM/IEEE Design Automation Conf.*, p. 7-12.

[39] Robič B. and Šilc J. (1986) On Choosing a Plan for the Execution of Data Flow Program Graph. *Informatica*, **10**, 3, p. 11-17.

[40] Robič B., Šilc J., and Kolbezen P. (1987) Resource Optimization in Parallel Data Driven Architecture. *Proc 5th IASTED Int'l Symp. Applied Informatics*, p. 86-89.

[41] Robič B., Kolbezen P., and Šilc J. (1992) Area Optimization of Dataflow-Graph Mappings. *Parallel Computing*, **18**, 3, p. 297-311.

[42] Southard J. R. (1983) MacPitts: An Approach to Silicon Compilation. *IEEE Computer*, **16**, 12, p. 74-82.

[43] Šilc J. and Robič B. (1989) Synchronous Dataflow-Based Architecture. *Microprocessing and Microprograming*, **27**, 1-5, p. 315-322.

[44] Šilc J., Robič B., and Patnaik L. M. (1990) Performance Evaluation of an Extended Static Dataflow Architecture. *Computers and Artificial Intelligence*, **9**, 1, p. 43-60.

[45] Šilc J. (1992) Time optimization of asynchronous processing with introduction of some synchronization mechanisms. *Ph.D. Thesis*, University of Ljubljana, Slovenia. (in Slovene)

[46] Šilc J. and Robič B. (1993) Program Partitioning for a Control/Data Driven Computer. *Journal of Computing and Information Technology*, **1**, 1, p. 47-55.

[47] Thomas D. E., Hitchcock C. Y., Kowalski T. J., Rajan J. V. and Walker R. (1983) Automatic Data Path Synthesis. *IEEE Trans. Computers*, **16**, 12, p. 59-70.

[48] Trickey H. (1987) Flamel: A High-Level Hardware Compiler. *IEEE Trans. CAD*, **6**, 2, p. 259-269.

[49] Tsai F. S. and Hsu Y. C. (1992) STAR: An Automatic Data Path Allocator. *IEEE Trans. CAD*, **11**, 9, p. 1053-1064.

[50] Tseng C. J. and Siewiorek D. P. (1986) Automated Synthesis of Data Paths in Digital Systems. *IEEE Trans. CAD*, **5**, 3, p. 379-395.

[51] Zimmermann G. (1980) MDS – The Mimola Design Method. *J. Digital Systems*, **4**, 3, p. 337-369.

FORCE CONTROL OF AN INDUSTRIAL ROBOT WITH ADAPTIVE COMPENSATION OF THE ENVIRONMENT STIFFNESS

Bojan Nemec and Leon Žlajpah
Jožef Stefan Institute, University of Ljubljana
Ljubljana, Jamova 39, Slovenia
bojan.nemec@ijs.si

Keywords: Force control, robot control, adaptive control

Edited by: Jadran Lenarčič
Received: September 15, 1993 **Revised:** March 20, 1994 **Accepted:** March 30, 1994

The paper describes the implementation of the adaptive force control of an industrial robot. The implemented algorithm is a position based hybrid control scheme with adaptation to the environment stiffness. Control scheme is sensitive to the changes in environment stiffness. We solved this problem by the adaptive controller. Implementation problems on the robot controller are also discussed. The proposed control method is easy to implement and can be applied to existing industrial robots fitted with a conventional position controller. The performance of the force controlled manipulator with the proposed control law was tested with the computer simulation and by using the real robot.

1 Introduction

Many of robot industrial applications, such as automated assembly, deburring, teleoperation, etc., require exact control of interaction forces with the environment. The problem of controlling interaction forces has been investigated by many authors. According to Kazerooni [7], active force control strategy can be classified into two major approaches. The first approach force or torque is commanded along those directions constrained by the environment, while position or orientation is commanded in the direction unconstrained by the environment. The above approach was formalized by Mason [9]. Craig and Raibert [12] introduced a hybrid force/position controller by controlling the actuator torque. Whitney [14] proposed damping control where sensed force error is transformed into the commanded velocity of the actuator. A similar approach was used by Paul and Shimano [11]. Some advantages can be obtained if the decoupling of the manipulator is done in the task space, like in the operational space approach introduced by Kathib [6]. The second approach is based on establishing a relationship between the position of the manipulator and interaction forces. Error in position, velocity and force generates joint torque commands. Salisbury [13] introduced the stiffness control approach which acts like a six-dimensional active spring in the tool coordinates. Impedance control which combines stiffness and damping control was introduced by Hogan [5]. Our approach is modified hybrid/position force controller where force error is converted to the position offset. This method is easy to implement and requires no modification at the servo level of the robot controller. The stability and response of the proposed force controller depend on the sensor and environment stiffness. For applications on unknown or changing environment stiffness we propose a simple adaptive controller which adapts the gain of the force control loop to the environment and sensor stiffness.

2 Force control

The problem of compliant control can be broken down into pure position and pure force control. In a direction where the robot task is unconstrained

by the environment, pure position control can be used, while in the direction constrained by the environment pure force control is used. The constrains imposed by the environment are called natural constrains. In order to specify the desired task, artificial constraints are introduced. Natural and artificial constraints together form N-dimensional constrained space C, where N is the number of the Cartesian degrees of freedom. The task of the controller is to map the C space into the manipulator joint movement. The hybrid force control method controls motor torque directly. Here, another approach was used due to the hardware limitation of the controller of our robot. The force control is implemented in the outer loop of the existing position/velocity control [1] and generates new N dimensional input vector $\mathbf{y}_d$ in tool coordinate system

$$\mathbf{y}_d = \mathbf{S}\mathbf{x}_f + (\mathbf{I} - \mathbf{S})\mathbf{x}_p \qquad (1)$$

where $\mathbf{x}_p$ is the desired displacement vector of the robot (translations and orientations), I is the identity matrix and S is the compliance selection matrix [12], and $\mathbf{x}_f$ is

$$\mathbf{x}_f = \mathbf{K}_f(\mathbf{F}_d - \mathbf{F}) \qquad (2)$$

where $\mathbf{K}_f$ is the force controller transfer function and $\mathbf{F}$ and $\mathbf{F}_d$ are the measured force and the desired force, respectively. The compliance selection matrix is defined as a binary NxN-tuple which specifies which degrees of freedom in C are under force control and which are under position control. The first term in the Eq.1 corresponds to the force control loop where the last term is the position (orientation) command vector. The position (orientation) command vector is transformed from the tool coordinate system to the robot base coordinate system and then to the joint coordinates. Joint coordinates $\mathbf{q}_d$ are passed to the position/velocity controller. This transformation can be described by the equation

$$\mathbf{q}_d = \Psi^{-1}(\mathbf{A}(\mathbf{y}_d)) \qquad (3)$$

where Ψ^{-1} describes the transformation from the Cartesian space to the joint angles and $\mathbf{A}$ denotes the transformation from the tool coordinates to the robot base coordinates.

[1] this method is referred to as position based force control

A simple PI controller with the discrete transfer function $K_f\frac{1-\xi z^{-1}}{1-z^{-1}}$ was used for the force controller transfer function. In order to improve the stability, first order anti alias filter was used in the force feedback loop.

2.1 Design of the force controller : Single–joint case

We will first design the closed loop system for the single joint case. Stability analyses will be done in the S domain by a root locus design. A model of the one-joint robot system with DC (AC) motors is presented on Fig. 1. The parameters of the transfer functions were estimated by using the test signals and LS estimation procedure and compared with the known parameters of the system to validate results. For the third joint of our robot the transfer function parameters are as follows:

K_f = 0.06 rad/N	gain of the force control
ξ = 0.997	damping of the force control
K_p = 1400 1/s	gain of the position control
K_u = 15900	gain of the velocity comp.
K_v = 1000 Vs/rad	gain of the P velocity control
K_{vi} = 5000 V/rad	gain of the I velocity control
K_t = 0.031831 V/rad^2	tachometer gain
K_i = 0.23 Nm/A	torque constant
K_b = 0.101 Vs/rad	back EMF constant
R = 0.91 Ω	motor resistance
B_{eff} = 0.0003 Nms/rad	effective damping
H_{eff} = 0.00046 Nms^2/rad	effective inertia
K_{se} = 350 N/rad	sensor/environment stiffness
n = 1/70	gear ratio
a = 50	anti alias filter pole
Ts = 0.01 s	sampling time

The position controller consists of simple K_p gain with feed-forward velocity compensation realized by a digital computer. Since the sampling time of the position control loop is much smaller than the sampling time of the force feedback loop, it is assumed that the position controller is realized with an analog feedback. The force control loop is realized by a digital computer, therefore we will assume zero-order sample/hold element at the input of the position controller. We assume simple model environment, described by the Eq. 4, where q_c and q is the environment contact position and measured position in joint coordinates respectevely.

$$F = K_{se}(q - q_c) \qquad (4)$$

From Fig. 1 we can compute the open loop [2] [3]

[2] open loop with respect to the force loop

[3] we will omit the subsystem index i in the equations for the single joint case

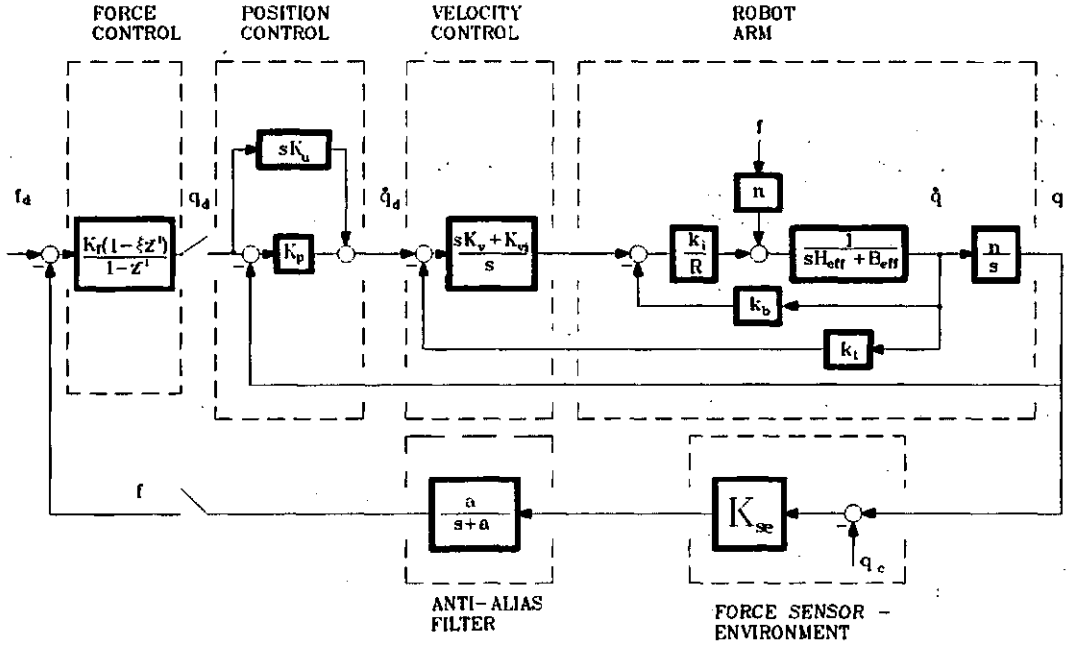

Figure 1: Model of one robot joint with velocity and position control

transfer function for the single joint case in the form

$$q = G_1(s)q_d + G_2(s)f \tag{5}$$

where

$$G_1(s) = \frac{nK_i(K_p + K_u s)(K_{vi} + K_v s)}{W(s)} \tag{6}$$

$$G_2(s) = \frac{n^2 Rs}{W(s)} \tag{7}$$

and

$$\begin{aligned} W(s) = {} & H_{eff}s^3R + \\ & (B_{eff}R + K_iK_vK_t + K_iK_b)s^2 + \\ & K_i(K_{vi}K_t + K_pK_vn)s + K_iK_pK_{vi}n \end{aligned} \tag{8}$$

Discrete PI force control law for the single joint case is

$$q_d(k) = q_d(k-1) + K_f(e(k) - \xi e(k-1)) \tag{9}$$

where $e = (f_d(k) - f(k))$, f and f_d are the measured and the desired joint forces respectively. Factor ξ was chosen to meet the desired dynamic performance of the closed loop system. The overall discrete transfer function of the reduced [4] open loop system with one sample delay in the control loop and first order anti alias filter for the sampling time $T_s = 0.01$ s is thus

$$G(z) = \frac{0.0221z^{-1} - 0.0180z^{-2} - 0.0029z^{-3} - 0.00007z^{-4}}{1 - 1.4488z^{-1} + 0.4736z^{-2} - 0.0005z^{-3} - 0.00004z^{-4}}$$

The above model was used to determine suitable gain for the force control loop via discrete root locus analyses. The root locus for the 3rd joint discrete model of our robot is presented in Fig.2. The gain K_f where system becomes unstable is 0.132 rad/N and suitable gain at dominant damping factor $\zeta = 0.5$ is 0.06 rad/N.

2.2 Design of the force controller : Multi-joint case

Robot dynamics is described by using the Lagrangean formulation, with the Eq. [5]

$$\tau = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{d}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{J}^T\mathbf{F} \tag{10}$$

where τ is the N-dimensional actuator force (torque) vector, $\mathbf{H}(\mathbf{q})$ is the NxN dimensional manipulator and actuator inertia matrix, $\mathbf{d}(\mathbf{q}, \dot{\mathbf{q}})$ is the N-dimensional vector of Coriolis, centrifugal, gravity and friction forces, $\mathbf{J}$ is the manipulator Jacobian and $\mathbf{F}$ is the compliant force in Cartesian coordinates.

[4] The non-reduced system is of 5th order. The system was reduced by canceling non-dominate poles and non-dominate zeroes

[5] for the sake of simplicity we will omit time dependence in the equations that follow

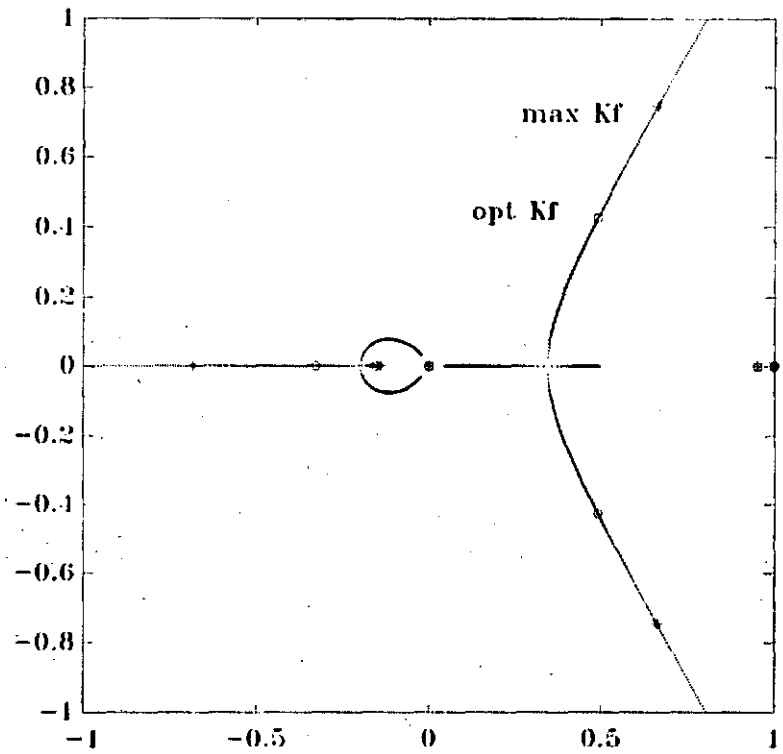

Figure 2: Discrete root locus of the system

In a general case Eq.10 describes a highly nonlinear and strongly coupled multivariable system. A stability analysis with the proposed control law is very difficult, if not impossible for such a system. The following is assumed for the stability analyses

- The gravity is compensated either by mechanical construction of the robot or by online calculation and compensation with a controller.
- the manipulator is operating at low speed, centrifugal and Coriolis forces are therefore negligible.
- the deflections in the position around the desired force are low, because force sensors have high stiffness. This assumption will allow linearisation of the system around the set point.

Furthermore the majority of existing industrial robots have a high gear ratio between the drive motors and the joint. With the above assumptions the matrix of inertia $\mathbf{H}(\mathbf{q})$ can be approximated by a diagonal matrix with constant terms $\mathbf{H}_d$ and $\mathbf{d}(\mathbf{q}, \dot{\mathbf{q}})$ can be approximated with constant damping $\mathbf{B}\dot{\mathbf{q}}$.

$$\tau = \mathbf{H}_d\ddot{\mathbf{q}} + \mathbf{B}\dot{\mathbf{q}} + \mathbf{J}^T\mathbf{F} \qquad (11)$$

For a noncompliant motion Eq.11 describes a decoupled system, which is generally not true in case of compliant motion. First we will analyse the open loop transfer function (Eq.5) for multi joint case. Matrices $\hat{\mathbf{G}}_1$ and $\hat{\mathbf{G}}_2$ are diagonal matrices consisting of the subsystem transfer functions described by Eq.6 and Eq.8 respectively.

$$\mathbf{q} = \hat{\mathbf{G}}_1\mathbf{q}_d + \hat{\mathbf{G}}_2\mathbf{J}^T\mathbf{F} = \hat{\mathbf{G}}_1\mathbf{q}_d + \hat{\mathbf{G}}_2\mathbf{J}^T\mathbf{K}_{se}(\mathbf{x} - \mathbf{x}_c) \qquad (12)$$

where $\mathbf{x}$ is the position and $\mathbf{x}_c$ is the contact position in the Cartesian coordinates. We will define position as position for the desired force plus the deflection from that set point $\mathbf{x} = \mathbf{x}_d + \Delta\mathbf{x}$. Then Eq.12 can be rewritten in the form

$$\begin{aligned} \mathbf{q} &= \hat{\mathbf{G}}_1\mathbf{q}_d + \hat{\mathbf{G}}_2\mathbf{J}^T\mathbf{K}_{se}((\mathbf{x}_d - \mathbf{x}_c) + \Delta\mathbf{x}) \\ &= \hat{\mathbf{G}}_1\mathbf{q}_d + \hat{\mathbf{G}}_2\mathbf{J}^T(\mathbf{F}_d + \mathbf{K}_{se}\Delta\mathbf{x}) \\ &= \hat{\mathbf{G}}_1\mathbf{q}_d + \hat{\mathbf{G}}_2(\mathbf{f}_d + \mathbf{J}^T\mathbf{K}_{se}\mathbf{J}\Delta\mathbf{q}) \end{aligned} \qquad (13)$$

Matrix $\mathbf{J}^T\mathbf{K}_{se}\mathbf{J}$ is the joint stiffness matrix $\mathbf{K}_{sq}$. The control law for cartesian coordinates is in the form

$$\Delta\mathbf{x}_f = \mathbf{K}_f\mathbf{K}_{se}(\mathbf{x}_d - \mathbf{x}_c - \mathbf{x} + \mathbf{x}_c) = \mathbf{K}_f\mathbf{K}_{se}\Delta\mathbf{x} \qquad (14)$$

Next we multiply both sides of the Eq. 14 by Jacobian inverse and assume, that all subsystems

are tuned using inner position and velocity controller to have equal close loop dynamic properties. Then, the matrix $\mathbf{K}_f\mathbf{K}_{se}$ is diagonal matrix with equal terms and control law (Eq. 14) can be rewritten into the form

$$\Delta\mathbf{q}_d = \mathbf{K}_f\mathbf{K}_{se}\Delta\mathbf{q} \qquad (15)$$

Control law for the multi-joint case is thus identical to the control law in single-joint case.

From Eq. 5 and 13, we can see that the dynamics of the multi joint case is thus similar to the dynamics of the single-joint case except that the joint compliance $\mathbf{K}_{sq}$ matrix introduces nonlinearity and cross-coupling between joints. Joint stiffness matrix can be calculated and compensated on-line. This will assure stability of the overall system regarding the assumptions presented at the beginning of this paper section. In our robot with high gear ratio the influence of the last term in the Eq.13 is almost negligible and the results of the single-joint case are also valid for the multi-joint case.

2.3 Adaptation to the variable sensor and environment stiffness

The stability of the proposed force control loop is mainly affected by the environment and sensor stiffness. If the stiffness is not known in advance or is changing during the task, the response of the force control may be to slowness when the expected stiffness is lower than real stiffness. When the real stiffness is greater than the expected stiffness, the response of the robot can be very oscillatory, bouncing and even unstable. This problem can be slightly reduced by diminishing time delays in the force control loop (see the results of [1]), but this may be impossible with some robot controller architectures. The above problem can be efficiently solved by the adaptive control loop. Sensor stiffness can be computed from Eq. 4. Unfortunately, the contact position $\mathbf{x}_c$ vector is usually not known in advance. Differentiating the Eq. 4 poses implementation problems. Robot position signals are usually read from encoders and are not so affected by noise as force signals, which are read as analog values from an A/D converter. Differentiating noisy signals gives less useful results. In [2] averaging was proposed to avoid this problem. Namely, Eq. 4 can be expressed also as $\mathbf{F} = \mathbf{K}_{se}\mathbf{x} - \mathbf{F}_0$, where $\mathbf{F}_0$ is a constant offset if contact position remains unchanged. However, averaging slows the adaptation speed. We propose a state variable filter to solve differentiation problems. In this case, force readings and position vectors are lead to the simple, stable, first order filter with transfer function

$$G_f = \frac{s}{bs+1} \qquad (16)$$

which can be realized by a computer program or by a simple analog circuit. The realization of the filter is presented on Fig. 3. Filtered derivatives $\dot{\mathbf{F}}_f$ and $\dot{\mathbf{x}}_f$ are then used for the estimation of the sensor and environment stiffness. A

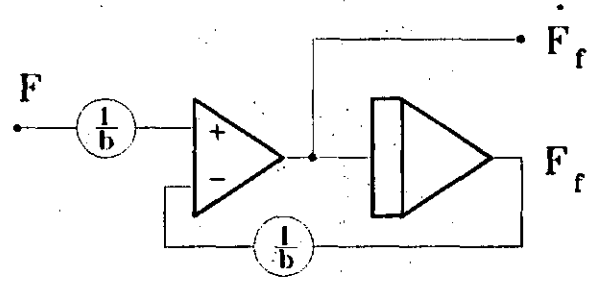

Figure 3: Derivative of an signal obtained by filtering

direct adaptive controller was used in our control scheme. In order to increase the adaptation speed and avoid the computational burden we chose simple reference model in the form

$$\mathbf{F}_m = \hat{\mathbf{G}}\mathbf{K}_0\mathbf{x} \qquad (17)$$

with the desired response. Root locus design was used to determine the required gain $\mathbf{K}_0$ for the desired behaviour of the reference model. Note that sensor-environment stiffness is included in $\mathbf{K}_0$. The aim of the adaptive controller is to minimize the output error between the reference model and the system with variable gain $\mathbf{K}_f$

$$\mathbf{F} = \hat{\mathbf{G}}\mathbf{K}_f\mathbf{K}_{se}\mathbf{x} \qquad (18)$$

It can easily be verified that the proposed adaptive control satisfies the criteria for the perfect linear model following control [8]. The gain K_f [6] for the each subsystem is calculated using RLS

[6] we will omit the subsystem index i in the equations that follow

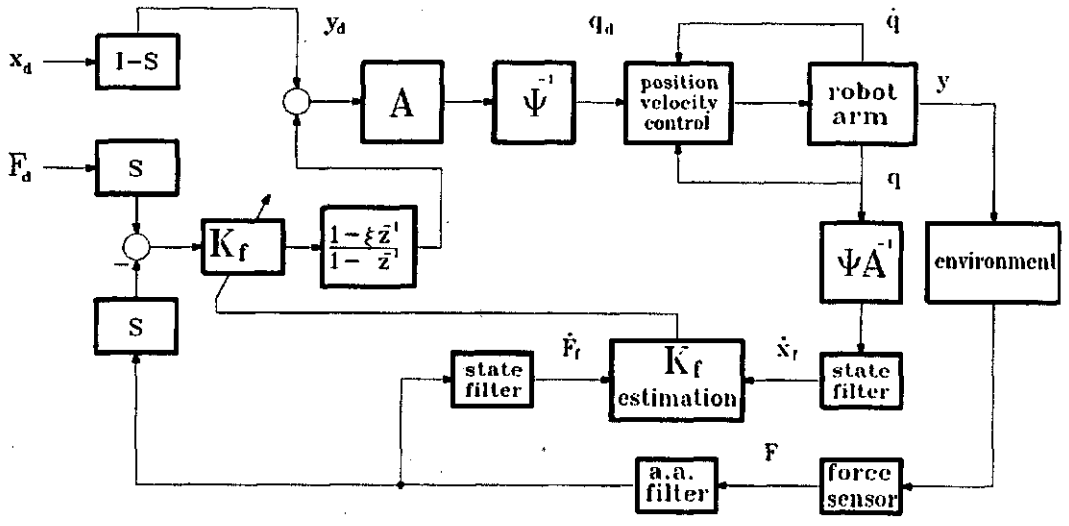

Figure 4: Adaptive control scheme

estimation procedure in the form

$$K_f(k) = K_f(k-1) + \frac{P(k-2)\psi(k-1)}{1+\psi(k-1)P(k-2)\psi(k-1)} e_m(k-1) \quad (19)$$

$$\psi(k-1) = \frac{\dot{F}_f(k-1)}{K_0} \quad (20)$$

$$e_m(k-1) = \dot{x}_f(k-1) - \psi(k-1) \quad (21)$$

$$P(k-1) = (P(k-2) - \frac{P(k-2)\psi(k-1)\psi(k-1)P(k-2)}{1+\psi(k-1)P(k-2)\psi k-1})\lambda \quad (22)$$

λ is the appropriate chosen forgetting factor. Normally, the forgetting factor is set between $\lambda = 0.95$ and $\lambda = 0.99$. However, a large forgetting factor slows down the convergence of the estimation, but the algorithm is more resistant to a sudden change in the estimated parameters due to noise . In our case we get the best results with a forgetting factor of about $\lambda = 0.7$. In this case the behaviour of the RLS estimation will approach the projection estimation algorithm. With a low forgetting factor λ, there is a danger of covariance blowup during the period when the system is insufficiently excited [3]. In our case we solved the problem by identifying only during the period when filtered derivative of the force was above the noise limit.

If the contact position remains unchanged during the adaptation, the adaptive system is linear in the unknown parameter and the stability of such a system can be easily verified under the assumption of a persistently excited system [4]. However, if the contact position changes during the adaptation, it is impossible to estimate both contact position and stiffness of environment. Namely, changes in contact position have the same effect as changes in environment stiffness. To solve this problem we propose to estimate K_f only a few samples after the sensor reaches the obstacle, i.e. to identify only few samples after the filtered force derivative changes from 0 to the ϵ, where ϵ is a suitably chosen constant according to the sensor reading resolution and noise in the measurement.

The adaptive control scheme is presented in Fig.4. Note that at low speed calculation of the robot position in the task space can be replaced by the desired position in the task space. The closed loop behaviour was simulated by using continuous time simulation of the DC motor, gears, velocity controller and sensor, discrete simulation for the position control and trajectory interpolation at a sampling interval of 0.0016 s and discrete simulation of the force control loop at a sampling interval of 0.01 s.

The simulation results for the step response of the proposed control scheme are presented in Fig.5 for the non-adaptive and adaptive controller with filtered signals respectively. Both, the adaptive and non-adaptive controller were tuned for the environment stiffness 1 N/mm, while ac-

tual environment stiffness was 4 N/mm. From the simulation results we can see that the non-adaptive controller starts to bounce when environment stiffness increases and goes to a limit cycle. In contrary, the adaptive controller quickly adjusts to the new environment stiffness.

The simulation results were compared to the measurement obtained on the real robot. The step response of the adaptive and non adaptive controller for the stiff environment are presented in Fig.6. The adaptive controller estimates correct gain and is stable, but some oscillation can be noticed during the impact, which are not obtained in the simulation. This is mainly due to the nonlinear friction and backlash in the gears, which were not included in the simulation.

3 Implementation on the robot controller

The proposed compliance control scheme was implemented on a 6. d.o.f. industrial robot RIKO 106. The architecture of the control system is presented in Fig.7. The main CPU of the robot controller is dedicated to trajectory generation, kinematic transformation and man-machine interface. The axis computer is used for the digital position controller with feed-forward speed compensation and for interfacing with the controller periphery. Because of hardware limitations, force feedback was realized via the main CPU. The sampling interval of the force controller, as well as the sampling interval for the trajectory generation module was set to 0.01 s. The desired trajectory is passed to the axis CPU by a shared VME RAM. The axis computer generates trajectory with sampling time 0.0016 s by polynomial interpolation. Due to the interpolation algorithm and data exchange between the main and the axis CPU a delay of 0.02 s appears in the force feedback loop. RRL robot programming language is implemented on the robot controller [10]. Three additional commands were added to RRL for the compliant motion definition. Natural constrains are defined with command ForceSELect, **FSEL s1 s2 .. s6**. Nonzero parameters s1 .. s6 corresponds to the pure force control in the direction x y z roll pitch yaw, while the zero parameter specifies the pure position control in the tool coordinates. The value of the parameters s1 ..s6 selects the A/D channel where the corresponding force signal appears. The negative parameter reverses the signal input sign. Artificial constraints are defined with command ForceTRACK, **FTRACK c1 .. c6**, where c1 .. c6 is the desired velocity (angular velocity) or force (torque) vector according to the artificial constraints definition. The offset of the sensor and A/D converter, as well as the effect of gravity on the sensor and tool is removed using command **CALIBRATE FS**. Of course, during the calibration, the force sensor should not be in contact with the environment. Calibration activates also adaptation procedure.

4 Example

Force control was tested on the deburring process of an irregulary shaped workpiece. The task of the robot was to apply constant force 70 N in the orthogonal direction of the free movement of the robot and to maintain zero torque at the tool during movement at constant speed 10 mm/s along the X axis of the workpiece. We used a three-dimensional wrist mounted force sensor, developed at our institute. The RRL program for the required task is listed in Fig.8. The response of the robot is presented in Fig.9 for orthogonal force and wrist torques respectively. In the Fig.9 plot between (t=5sec) and (t=9sec) shows the tracking of the sensor when change in the shape of workpiece occur. We can see that the signals are rather chattering. It was found that this is caused mainly by poor resolution, cross-coupling and noise of the sensor. A higher sampling frequency improves the transient response, but does not eliminate chattering from the response.

5 Conclusion

A force control algorithm based on a hybrid control scheme was presented. The main difference between the original method and our approach is that force is controlled by changing the desired position. This approach allows implementation on existing robot controllers with a position and velocity control loop. The limitations of our approach are the following:

- position resolution of the robot controller affects the force resolution of the system. In

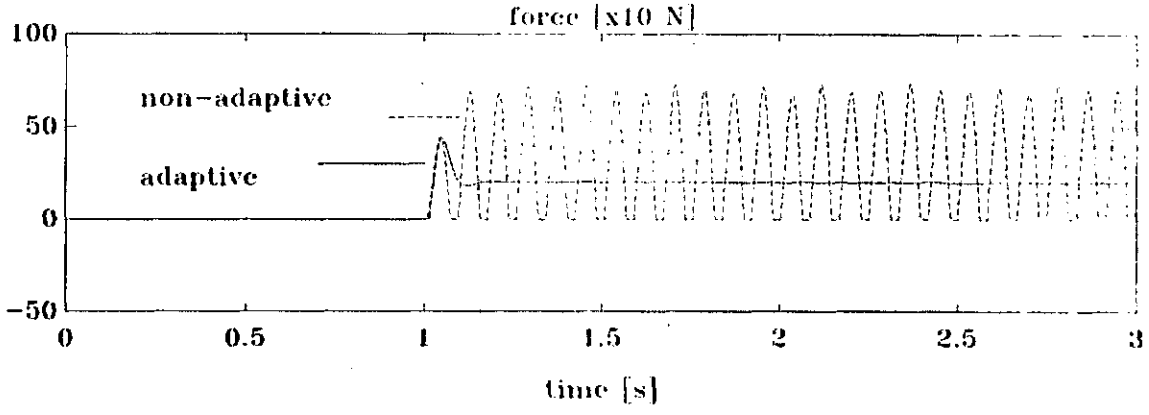

Figure 5: Step response of the simulated adaptive and non-adaptive system

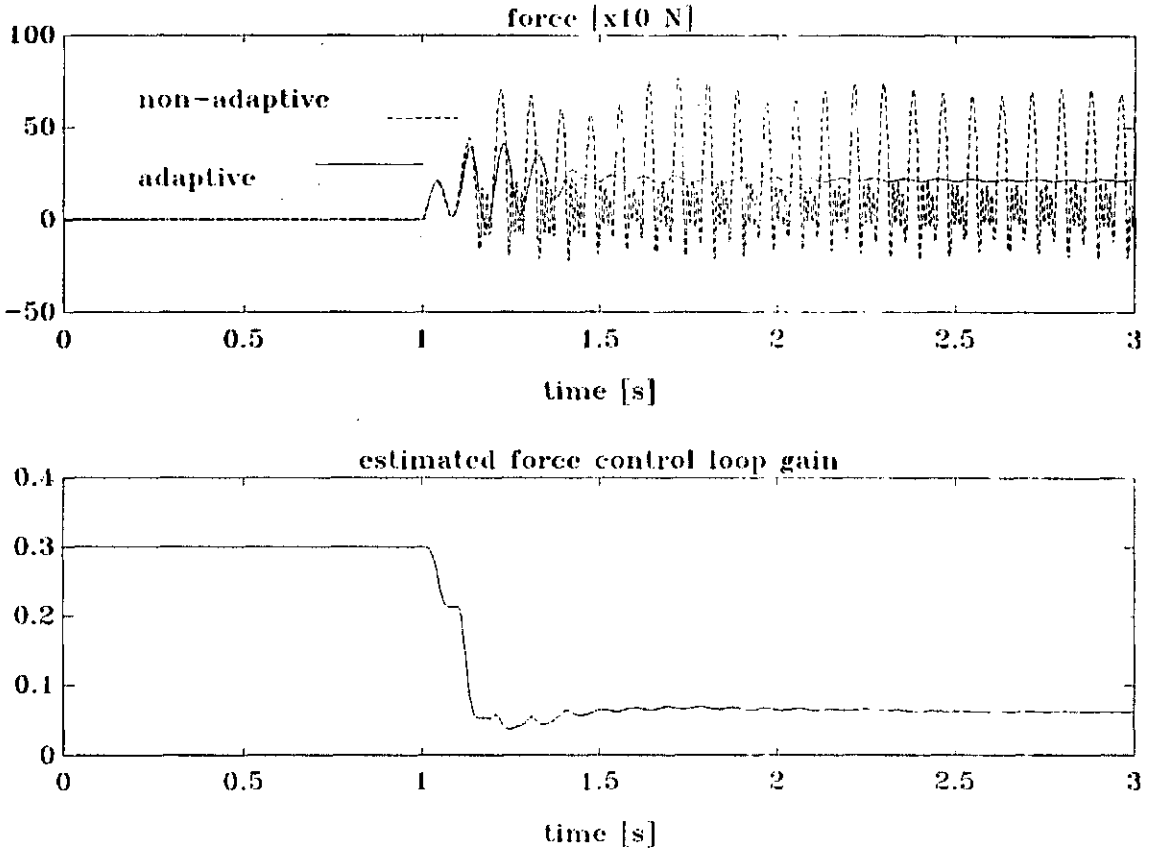

Figure 6: Step response of the adaptive and non-adaptive system and estimated gain of the adaptive controller

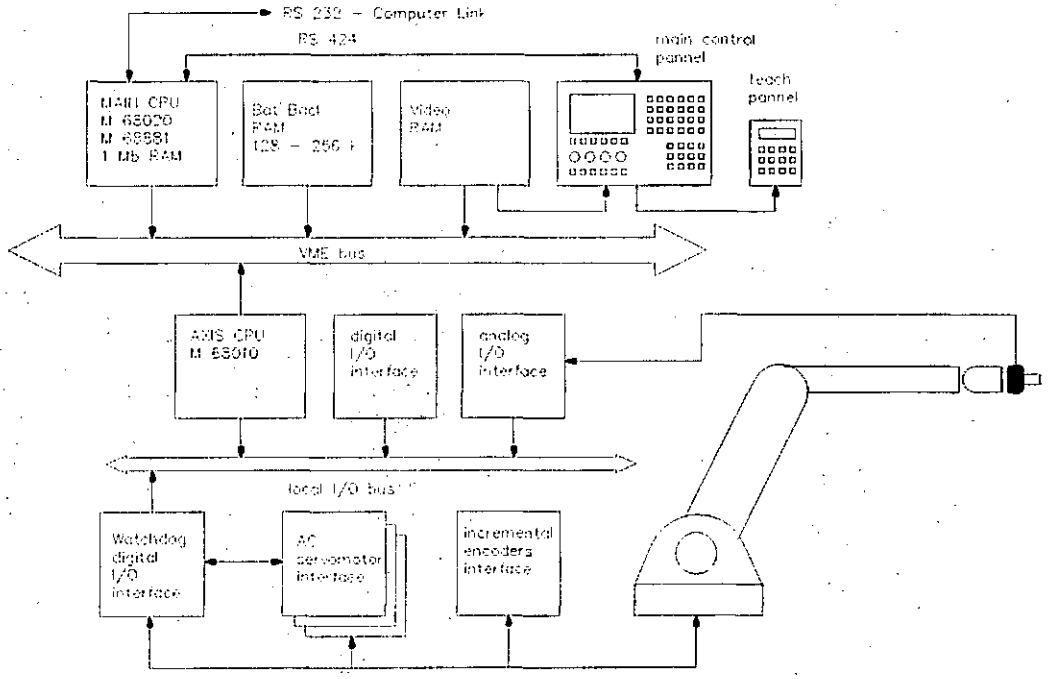

Figure 7: Architecture of the RIKO 106 robot controller

```
1 * RRL sample program for the deburring process
2 *
3 * define maximal, actual speed and toll center point
4 MAXSP = 1000 40
5 SPEED = 50.0
6 TCP 1 = 0 350 0 0 0 0
7 * approach start point of the deburring and calibrate sensor
8 APPRO TO 1 FOR 0 -10 0
9 CALIBRATE FS
10 * natural constraints
11 FSEL 0 1 0 0 2 3
12 * artificial constraints , start deburring, stop on external signal
13 FTRACK 10.0 70.0 0 0 0 0 UNTIL SIG 0
14 DEPART FOR 0 -10 0
15 HOME
```

Figure 8: RRL program for the deburring process

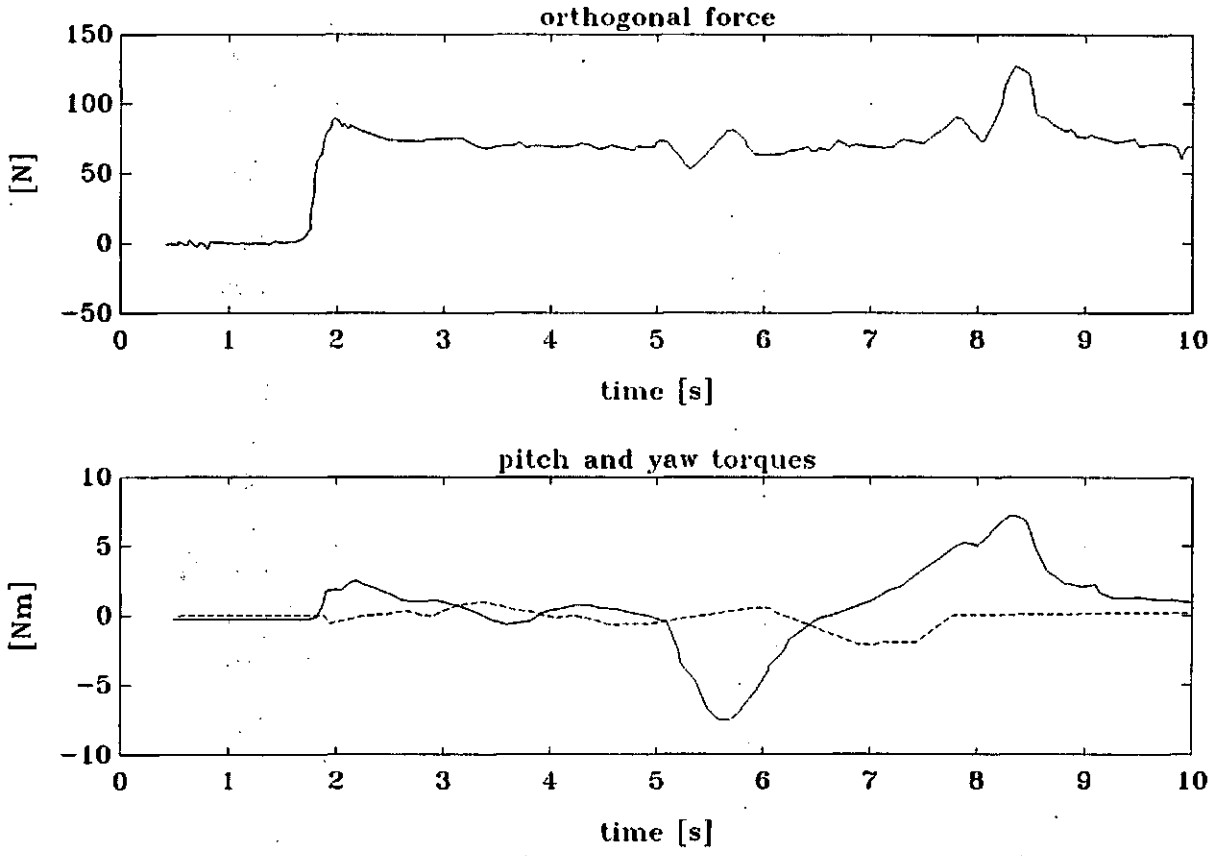

Figure 9: Maintained force of the tool (in the direction normal to the surface) and torques during deburring (in the direction of the movement and orthogonal to the direction of the movement). Note that the shape of the object changes (Fig. 8)

other words, the proposed control scheme will not work with sensors with high stiffness and robots with poor position resolution.

- the sampling interval of the force control loop is the same as the sampling frequency of the trajectory generation module.

Therefore, the proposed method is suitable for compliant tasks at low speed. In the proposed control law the environment stiffness is directly multiplied by the force control gain. Additionally, time delays introduced by the interpolation algorithms and communications between main and axis processor affect the stability of the control algorithm. To avoid this problem we proposed simple direct model reference adaptive controller. A discrete root locus was used for the force controller design. The results were verified with a simulation and compared with the response of the actual robot. The paper shows that the root locus design is also suitable for a multi-joint case in the case of the high gear robot and low speed robot movements.

References

[1] S.D. Eppinger and W.P Seering. On dynamic models of robot force control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 29–34, 1986.

[2] V. Hayward L. Daneshemend and M. Pelletier. Adaptation to environment stiffness in the control of manipulators. *Lecture Notes in Control and Information Science, Experimental Robotics I.*, Vol. 1, No. 1, pages 150–164, 1989.

[3] L.S. Kersenbaum T.R Fortescue and B.E. Ydstie. Implementation of self-tuning regulators with variable forgetting factors. *Automatica*, Vol. 17, No. 6, 1981.

[4] C.G. Goodwin and K.S. Sin. *Adaptive Prediction, Filtering and Control.* Prentice-Hall, 1984.

[5] N. Hogan. Impedance control of industrial robots. *Journal of Robotics and Computer-Integrated Manufacturing*, Vol. 1, No. 1, pages 97–113, 1984.

[6] O. Kathib. A unified approach for the motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, Vol. 2, No. 2, pages 83–106, 1987.

[7] H. Kazerooni, T.B. Sheridan, and P.K. Houpt. Robust compliant motion for manipulators. *IEEE Journal of Robotics and Automation*, Vol. 2, No. 2, pages 83–106, 1986.

[8] Y. Landau. *Adaptive Control.* Marcel Dekker, 1979.

[9] M.T. Mason. Compliance and force control for computer controlled manipulatotrs. *IEEE trans. on System Man and Cybernetics*, Vol. 11, No. 6, 1981.

[10] B. Nemec, A. Ružić, V. Ilc. RRL — an integrated envirnonment for robot programming. *Informatica*, Vol. 16, No. 1, pages 27-33, 1992

[11] R.P. Paul and B. Shimano. Compliance and control. In *Proc. of the Joint Automatic Control Conference*, pages 694–699, July 1976.

[12] M.H. Raibert and J.J. Craig. Hybrid position/force control of manipulators. *Journal of Energy Resources Technology*, Vol. 102, pages 126–133, 1981.

[13] J.K. Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *Proceedings of 19th Conference on Decision and Control*, pages 95–100, 1980.

[14] D.E. Whitney. Force feedback control of manipulator fine motions. *Trans. on ASME and Journals of Dynamics Systems Measurement and Control*, pages 91–97, 1977.

CURRENT STATUS OF THE EDR ELECTRONIC DICTIONARY PROJECT

Hiroshi Suematsu
Japan Electronic Dictionary Research Institute, Ltd. (EDR)
Mita-Kokusai Bldg. Annex, 4–28, Mita 1-chome, Minato-ku, Tokyo, Japan
suematsu@edr.co.jp

Keywords: beta test, bilingual dictionary concept dictionary, cooccurrence dictionary, current status, EDR, EDR corpus, electronic dictionary, project, structure, text base, word dictionary

Edited by: Anton P. Železnikar
Received: January 10, 1994 **Revised:** February 14, 1994 **Accepted:** February 28, 1994

The current status of the EDR electronic dictionary project and its future plan is reported in this paper. The electronic dictionary is not a mere electronic version of a book-form dictionary (i.e. machine-readable dictionary, MRD). It is a dictionary for computer processing of natural languages, e.g. machine translation. It captures lexical information, such as lemmas, parts of speech, and word senses, in a machine-tractable way, to support various natural language processing (NLP) approaches on the level of morphology, syntax, semantics, and pragmatics. A considerable vocabulary size needs to be covered in a consistent way, and implicit information left to the human reader in the case of a book-form dictionary, has to be explicitly stated in a computer-processible manner. In constructing such a large-scale electronic dictionary, a very large-scale corpus is needed to extract linguistic information, and to verify data described in the electronic dictionary. The corpus is gaining more importance along with the recent advocation of example-based machine translation. The EDR electronic dictionary project is one of the earliest efforts of this kind, and has one more year before it ends. EDR is now in the process of compiling and producing the final result of 8 years' work. The EDR Electronic Dictionary, or for short, EDR Dictionary, has reached a level competent enough to be used for R&D, provision of it has started as the beta test. There is a concern, however, about the slow progress in the English section of the Dictionary. Various views and opinions are currently under discussion on the future plan for the EDR Electronic Dictionary.

1 Introduction

The current status of the EDR electronic dictionary project and its future plan is reported in this paper. The *electronic dictionary* is not a mere electronic version of a book-form dictionary (i.e. machine-readable dictionary, MRD). It is a dictionary for computer processing of natural languages, e.g. machine translation. It captures lexical information, such as lemmas, parts of speech, and word senses, in a machine-tractable way, to support various natural language processing (NLP) approaches on the level of morphology, syntax, semantics, and pragmatics. A considerable vocabulary size needs to be covered in a consistent way, and implicit information left to the human reader in the case of a book-form dictionary, has to be explicitly stated in a computer-processible manner.

In constructing such a large-scale electronic dictionary, a very large-scale corpus is needed to extract linguistic information, and to verify data described in the electronic dictionary. The corpus is gaining more importance along with the recent advocation of example-based machine translation.

The EDR electronic dictionary project is one of the earliest efforts of this kind, and has one more year before it ends. EDR is now in the pro-

cess of compiling and producing the final result of 8 years' work. The EDR Electronic Dictionary, or for short, EDR Dictionary, has reached a level competent enough to be used for R&D, provision of it has started as the beta test. There is a concern, however, about the slow progress in the English section of the Dictionary. Various views and opinions are currently under discussion on the future plan for the EDR Electronic Dictionary.

2 EDR Electronic Dictionary

2.1 EDR Electronic Dictionary Project

The EDR electronic dictionary project was launched in 1986 with the aim of developing a fully fledged large-scale electronic dictionary. In April of the same year, with joint funding from the Japan Key Technology Center and eight private corporations, the Japan Electronic Dictionary Research Institute, Ltd. (EDR) was established to promote and oversee the project. A nine-year plan, now in the end of its eighth year, was drawn up for the development of dictionaries for use in NLP, to be collectively called the EDR Electronic Dictionary.

2.2 Overall Structure of the EDR Dictionary and the Role of Each Sub-dictionary

The EDR Dictionary has been developed for the Japanese language, and for English, as a representative of foreign languages. Its components can be broadly divided into three layers. The concept dictionary holds *deep-layer* information; its purpose is to collect knowledge and enable the computer to understand the meanings of words. The Word Dictionary, the Bilingual Dictionary, and the Cooccurrence Dictionary provide *surface-layer* information, and are meant to teach the computer syntactic and morphological behaviors. The Word Dictionary also has the role of linking the *surface* and *deep* layers. The EDR Corpus and the Text Base, which constitute the *resource-layer,* are collections of sample text data meant to serve as materials for the development of the sub-dictionaries. These sub-dictionaries of the EDR Dictionary are related to each other.

The role and the internal structure of each sub-dictionary are as follows:

- *The Word Dictionary* gives the correspondence between words and concepts, as well as grammatical attributes which apply when this correspondence is obtained. The Word Dictionary is a collection of lexical items, each of which consists of the headword information, grammatical information, and semantic information. It contains basic words in general use (200,000 words in each of the Japanese and English languages), as well as technical terms in the information processing field (100,000 words in each language.
- *The Concept Dictionary* is a collection of knowledge the purpose of which is to make the computer understand the concepts indicated by the words in the Word Dictionary. Concepts are defined by listing their relations to other concepts. It is a collection of concept entries, each of which consists of two related concepts, the relation labels which indicate the type of relationship between them, and the certainty factor of the relation. Due to the basic nature of the relationships, the entries are divided into concept classifications and concept descriptions. Concept classifications describe super-sub relations (AKO), while concept descriptions express *deep case* relations. Concept unification, which can be regarded as part of concept classification, represents the merging of two concepts that can be regarded as being in an equivalence relationship when certain values are the same.
- *The Bilingual Dictionary* defines the correspondence between Japanese headword and English headword. It is a collection of translated items, each of which consists of the entry information in the source language and the entry information in the target language (together with auxiliary explanations in some cases), as well as the translation relationships, including the manner of the translation correspondence.
- *The Cooccurrence Dictionary* gives information relating to language expressions and describes expressions between words at the *surface-layer* level. It is a collection of cooccurrence entries, each of which consists of two headwords and a cooccurrence relation label

(giving the type of cooccurrence and the mediation of cooccurrence).

- *The Text Base* is a database containing a large amount of machine-readable text with supplementary information, purchased from an external agency. It is a collection of sentences, and there is an index of the morphemes contained in each sentence; it can provide KWIC search results. It is used in the extraction of words not yet registered in the Word Dictionary, and in the accumulation of information on the frequency of word occurrence.
- *The EDR Corpus* is the result of carefully selecting sentences from the text base (500,000 sentences, in both Japanese and English), to which were appended morphological analysis information, syntactic analysis information, and semantic analysis information. It is used as data in the development and validation of the EDR Dictionary.

3 Current Status and Beta Test of the EDR Electronic Dictionary

3.1 Current Status

The overall status of the EDR Dictionary is that compilation of linguistic data originated in the Japanese language is in progress as planned, while that from the English language is behind schedule.

The progress of the sub-directories in the surface-layer largely depends on that of the Word Dictionary, because they are based on the same vocabulary set. For the Word Dictionary, the Japanese section is at the forefront of improvement of the entire EDR Dictionary, and further refinement will be continued so as to reach a higher level. The work for the English section has been stagnant in idiomatic expressions. The Bilingual Dictionary is being improved for the central vocabulary. For the Cooccurrence Dictionary, Japanese cooccurrence data is being extracted from the EDR Corpus. The English section is behind in this phase.

The sub-dictionaries in the deep layer, i.e. the two parts of the Concept Dictionary, have a similar status. For the Concept Classification, the concepts originating from the Japanese language have been completely refined, while those from the English language have just commenced improvement. And, for the Concept Description, improvement based on the actual data from the EDR Corpus is under way.

For the linguistic data in the resource layer, the Text Base was completed, and the EDR Corpus is in the process of improvement by refining analyzed data of the Japanese sentences. Those of the English sentences are behind schedule in this task.

For the User Support System, the basic functions have been implemented. Full-scale integration and improvement of the system will soon be started.

And, several systems to evaluate the EDR Dictionary have already been put into use.

3.2 Beta Test of the EDR Electronic Dictionary

EDR has just started providing outside users with the EDR Dictionary, as a beta test for the purpose of external evaluation research. The outline of the beta test is explained in this section.

The principal compilation of the EDR Electronic Dictionary was finished in March, 1993, and now we are facing the stage of feasibility evaluation. At this stage, full-scale improvement is planed to be made on the basis of the data obtained through the external evaluation of the Dictionary. Prior to this, EDR provided an avaluating edition of the Dictionary (first edition) for universities and other research institutes, in April, 1992, in order to collect such data.

Today, pursuing a higher degree of oerfection, EDR has decided to expand the scope of the evaluation sites, and to provide them with the latest edition of the EDR Dictionary (second edition) to promote evaluation research on the Dictionary. This is the Beta Test, and through this, EDR expects to attain a considerable amount of the evaluation data.

Evaluation research will last from the date of the contract to the end of December, 1994. During this period, EDR will provide the evaluation edition and the necessary tools free of charge, and the evaluating organization feeds back the evaluation data to EDR. In this way, EDR will be able to attain a higher degree of completion of the Dictio-

nary, and the evaluating organization will be able to take advantage uf using the Dictionary for its own research and for new application plans.

Evaluating organizations will be those who have a strong wish to use a production edition of the EDR Electronic Dictionary, and who are highly reliable on the secrecy matter. For the first provision, the eight investors in EDR and their related companies, major research institutes, and universities in Japan and overseas, approximately in total, are planned to be evaluators. For the second provision, other major manufacturers and software houses, approximately 40 in total are planned.

Up tu now, the transactions for the first provision were finished. However, transactions for the second provision are still in process, and at the end of March, 1994, the sale of the Dictionary will start to the selected institutions.

4 Future Plan for the EDR Electronic Dictionary

The EDR Electronic Dictionary is meant to serve as source data for R&D, and further efforts have to be encouraged for the Dictionary to be widely used. Along this line, the terms and conditions for the sale of the Dictionary are scheduled to ne officially announced at the end of March, 1994. In determining the conditions, the following principles have been adopted:

(1) The terms and conditions should allow as many users as possible.

(2) Special Considerations should be applied to universities and other non-profit organizations.

(3) There should be no difference in treating the users in Japan and abroad.

The creation of a mechanism for the continuous improvement and expansion of the Dictionary after the project is also under discussion. There is a rising demand in Japan for establishing centers which would be counterparts of the *Linguistic Data Consortium* (LCD) and the *Consortium for Lexical Research* (CRL) in the U.S. There is a hope to start building an organization to collect and provide various types of lexical data including the EDR Electronic Dictionary.

A new project is being proposed, which is to involve activities such as the development of electronic dictionaries dealing with phrases and sentences, to research and develop higher-level knowledge bases, and to utilize the EDR Electronic Dictionary. In order to create a worldwide consensus, an international conference and workshop, KB&KS '93[1], has been held in Tokyo in December, 1993, with success.

The EDR Electronic Dictionary project has revealed many problems concerned with electronic dictionaries, which need to be tackled. EDR will put these issues in order, including those that remain unsolved, and make them open so that future R&D will be able to make use of them. Improving and maitaining the English sections of the Dictionary has reached the limit of efforts available within Japan. There is a strong wish that EDR could somehow establish or deepen an appropriate cooperative relationship with relevant organizations in other areas.

References

[1] *EDR Electronic Dictionary Technical Guide*, TR-042, Japan Electronic Dictionary Research Institute, Ltd., Tokyo, 1993.

[1] KB&KS is a shortcut for Knowledge Building and Knowledge Sharing. International Conference on Building and Sharing of Very Large-scale Knowledge Bases was organized by Japan Information Processing Development Center, in December 1-2, 1993, in Keio Plaza Hotel, Tokyo.

Report: IJCAI'93 – a Critical Review

27th August – 3rd September, Chambery, Savoie, France

Matjaž Gams

The purpose of this report is to highlight critical points of the IJCAI'93 event. It is not that the author does not deeply appreciate AI or thinks that IJCAI'93 was not a great event – on the contrary. The main reason for the "critical-advocate approach" is to avoid writing just another favourable report regarding IJCAI.

International Joint Conference on Artificial Intelligence (IJCAI) is the major world-wide AI-related event. Every second time it is held in USA and every second time elsewhere, e.g., in Europe or Australia. Typically, there are a couple of thousands of participants. In 1993, there were more than 2000. Conference chair was Wolfgang Wahlster, program chair Ruzena Bajcsy, and local arrangements chair Jean-Pierre Laurent. Technical program started on Tuesday 31st of August and ended on Friday 3rd of September. There were 16 tutorials and 28 workshops, several panels, exhibitions, invited papers, awards.

The IJCAI distinguished service award for honouring senior scientists in AI was given to Daniel G. Bobrow, editor of Artificial intelligence. The IJCAI award for research excellence of consistently high quality was given to Raymond Reiter. The Computers and Thought Award for outstanding young scientists (below 35 years) was given to Hiroaki Kitano.

Now, to the critical remarks. First, there were two papers with the same first author. Regardless of the fact that the author of both papers is one of brilliant scientists and editors of Informatica, this seems quite extraordinary. Namely, the IJCAI'93 acceptance rate was around 25%, and several very good papers were rejected because of limited quota. It is a common practice that conferences and magazines limit the number of appearances of the same author. For example, Informatica does not accept scientific papers of the same (co)author in two neighbouring issues.

Second, there were typing mistakes and inconsistencies in the proceedings. For example, in a superb paper by Hiroaki Kitano we find (page 814, top paragraph) "... to it.[Waltz, 1988] and ...", later (page 814, third paragraph from top) "... direction[Brooks, 1991] and ...". Most other references in the text – unlike above – have one blank before and a delimiter behind. Another example - Figure 11 and Figure 13 both contain the misspelled word "Accracy". Text in figures is too small and in some cases practically unreadable, e.g., in Figure 4. One can hardly imagine that the invited paper is not carefully read and corrected.

Talking about one of the best papers and presentations at IJCAI'93, it seems really strange that at the end of the presentation participants started leaving in great numbers. Probably the reason for this was the announced dinner event and prolonged lecture - partially due to technical troubles and partially due to extended presentation. Why not postpone the buses?

Overall, organisation was quite well. The influence of French charm was clearly observable, and local arrangements by Jean-Pierre Laurent (one of best-known AI researchers in France and also an editor of Informatica) were very exciting from scientific and social point of view.

If the purpose of this paper is to present critical points about the event, then the refereeing process is clearly a matter of discussion. There were several matters of substance the author of this paper finds debatable. Consider for example, the problem of finding cut-points, i.e. points where to cut continuous-valued attributes into intervals to satisfy a sort of minimal-entropy function. In the proceedings, p.1023, theorem 1 basically states that it is worth to look for cut points only in intervals where examples change classes and not between examples of the same class. It is claimed that the proof of theorem 1 is rather lengthy and thus omitted, while we think that the proof is rather trivial.

Two such cases are presented on the next page. The first row indicates indexes of examples sorted by an attribute A. The examples are presented in the second row and belong to classes "+" or "-". In the third row there is a function calculated at

cut points between examples. The function is calculated as the relative sum of Gini index functions $(1-p_+^2-p_-^2)$ for left and right set of examples regarding the cut point. Weights represent relative frequencies of examples. For example, consider the case with six examples. If we cut between 1 and 2, we obtain 0.40 by calculating

$$\frac{1}{6}\left[1-\left(\frac{1}{1}\right)^2-\left(\frac{0}{1}\right)^2\right]+\frac{5}{6}\left[1-\left(\frac{2}{5}\right)^2-\left(\frac{3}{5}\right)^2\right]$$

The proof we have discussed in our laboratory about 5 years ago is as follows: In the book by Breiman, Friedman, Olshen and Stone (1984) *Classification and Regression Trees* it has been shown that "reasonable" functions such as Gini index or informativity are convex functions (Figure 4.3, 4.4, around p. 100). If functions such as Gini index are indeed convex then the weighted sum of two convex functions is again a convex function. Therefore, the only possibility for minimal cuts is in the intervals where classes of examples change.

```
1    2    3    4    5    6    7    8    9    10   11   12   13   14   15
+    +    -    -    -    -    -    -    -    -    +    +    -    -    -
 0.31 0.23 0.31 0.35 0.37 0.39 0.39 0.39 0.39 0.37 0.39 0.36 0.37 0.38
```

```
1    2    3    4    5    6
+    +    -    -    -    +
 0.40 0.25 0.44 0.50 0.40
```

Fair to say, although the basic proof for two-class problems seems trivial and can be extended to many classes as well, in the discussed paper there are several extensions and additions. Furthermore, we might have overlooked something. In that case I will be happy to publish a correction.

On the other hand, it is hard to believe that the misses we have observed were all results of our misunderstandings or "bugs" that inevitable do happen. The position of a "critical advocate" allows me to raise questions whether the refereeing process is really done as good as it could be.

From the point of critical research directions, two phenomena can be highlighted. First - there is an unhealthy gap between research and applications in AI. During several occasions a question was raised: Can anybody report really successful applications of AI? Typically, majority or participants remained silent. Since there are thousands of AI applications all over the world, it is hard to believe that researchers do not have contacts with developers. We were one of those quite happy to report around fifty small-scale applications in Slovenia, all related to the Bratko's laboratories. In particular, we have presented an application of novel integration methods at the ML and KA workshop organised by George Tecuci (Gams, Karba, Drobnic, Average-case improvements when integrating ML and KA). The major AI application in Slovenia was in use for 2 years at the Acroni Iron and Steel Works Jesenice.

There might be several reasons why most of the papers were seemingly unrelated to real world. Since IJCAI's motto is search for excellence, it is hard to find methods mature enough to be tested in real-life environments and at the same time so intellectually new and attractive to pass the refereeing procedure. The second reason might be the "Publish or perish" frenzy. It seems that certain circles in modern science have accepted that all that counts is the number of publications. In return, this criterion has forced scientists to devote most of their energy to publications, and not to essential scientific research. Remarks of participants working on real-life problems often indicated that one of the key assumptions enabling an original (although not verified) idea was related to the absence of real-life circumstances.

Another critical remark: most prominent presentations, especially the winners of Computers and Thought Awards in recent years (e.g. Brooks in 1991, Kitano in 1993) have presented substantially novel approaches while many or even most of the "normal" papers belonged to the "classical" AI. In that sense and from my personal point of view Hiroaki Kitano's presentation was a truly excellent way to promote new directions in AI and thus avoid the possibility of saturation.

Kitano earned his PhD in 1991. At present, he is a researcher at the Sony Computer Science Laboratory in Japan and the Center for Machine Translation at Carnegie Mellon University. According to the reviewers, Kitano won the prize for novel translation work on a massively parallel computer. Previous approach to translation

used explicit rules of knowledge and was guided by rigid rules. Kitano says that for domains such as translation between two natural languages it is almost impossible to obtain a complete set of knowledge for a given problem. His approach builds on memory as the foundation of intelligence. Each of computer's 32,000 processors contains a few sentences and determines the best match for an input sentence in natural language. Translation is based on extensive use of memory. Statements are parsed into basic parts, translated, and sensibly combined to the translated language. All reasoning is by analogy. The program achieves around 75-percent accuracy by one reports and around 90-percent in others. This puzzle was answeared by prof. Kitano's reply e-mail. He poited our that the accuracy figure 75etc. But all demonstrates superior performance than traditional methods.

His presentation at IJCAI'93 was even better, more profound, overviewing, observing very important changes and providing new directions.

One of the major Kitano's remark is related to AI research directions regarding problem domains. He categorises complete, correct and consistent problem domains as toy domains. Real-world domains are denoted as incomplete, incorrect, inconsistent, and are in addition characterised by human bias, tractability and economics. Real-world domains are by definition very difficult to explain because no simple ideas work well in such circumstances.

According to Kitano, a great part of AI research is devoted to toy problems which are unscalable not only to real-world problems but also to intelligent systems. He proposes designing useful systems (p. 817) which are from start devoted to real-world problems.

Another major Kitano's suggestion is that AI as well as computer science and engineering in general are much more dependent on the development of computer hardware that generally recognised. Bare computer power will son beat the best human chess player. Future computer projects can be characterised by massive computing, massive parallelism, massive data. The progress is still exponential. In a couple of years, a single-wafer chip will achieve the performance of best-today supercomputers.

One of the reasons for my overwhelming support for Kitano's approach lies in the fact that we have been working on similar problems for the last couple of years. In 1991, Gams and Krizman have published in Informatica a paper "The Principle of Multiple Knowledge" describing a cybernetical principle which leads to a conclusion that single systems (similar to Kitano's toy problems/systems) can not achieve good performance in real-life domains nor can they achieve intelligence on digital computers. This seems similar to Kitano's 'collaborative learning'.

Enthusiastic about Kitano's ideas B. Hribovsek and I have implemented two versions of an intelligent interface for VAX/VMS. One was classical and the other based on memory. Both work similarly, e.g., solve similar problems similarly, yet the memory-based program is a couple of times smaller, and therefore, a couple of times easier and faster to write. In addition, substantially more flexible, transportable and adaptable.

The success of memory-based approach is based on dramatic changes in memory capacities in recent years. While translation between natural languages demands special supercomputers, most of every-day human communication can be solved in this way on existing computers. Today, most of "normal" human-computer communication,, e.g., communication with an application package or operating system, can easily be captured in the primary memory of personal computers with response time well below one second.

In many computer and AI-related literature one observes a diagram showing that problems can be solved using programs (processing) or data (memory). For example, using only memory enables better and better performance with more and more memory, yet eventually gets unproportionally more expensive. Therefore, optimal solutions consist of proportional amount of data and proportional amount of program code. However, in the last 20 years human programmers produce basically the same amount or program lines per hour while the memory capacity of PC grew from 4 KB in 1980 to 8 MB in 1990 and are expected to approach 100 MB in 2000. This progress has already substantially moved the optimal approach towards using more memory and less code.

At the end, being critical or not, IJCAI'93 might be one of those rare events that substantially influence the way humans use computers.

CYBERNETICS, SYSTEMS RESEARCH AND INTERDISCIPLINARY RESEARCH

12th European Meeting on Cybernetics and Systems Research–1994

April 5–8, 1994
Vienna
Austria
Sponsored by:
Österreichische Studiengesellschaft für Kybernetik,
in cooperation with
University of Vienna
Department of Medical Cybernetics and Artificial Intelligence, and
International Federation for Systems Research

The plenary address of the Meeting is:

Margaret Boden, ARTIFICIAL INTELLIGENCE AND CREATIVITY, University of Sussex, Brighton, U.K.

Another plenary lecture is:

S.A. Umpleby, TWENTY YEARS OF SECOND ORDER CYBERNETICS, George Washington University, Washington, D.C., U.S.A.

The Programme of the Meeting is organized in 17 parallel Symposia:

A GENERAL SYSTEM METHODOLOGY (G.J. Klir, U.S.A.)

B ADVANCES IN MATHEMATICAL SYSTEMS THEORY (J. Miro, Spain; M. Peschel, Germany; and F. Pichler, Austria)

C FUZZY SYSTEMS, APPROXIMATE REASONING AND KNOWLEDGE-BASED SYSTEMS (C. Carlsson, Finland; K.-P. Adlassnig, Austria; and E.P. Klement, Austria)

D DESIGNING AND SYSTEMS, AND THEIR EDUCATION (B. Banathy, U.S.A.; W. Gasparski, Poland; and G. Goldschmidt, Israel)

E HUMANITY, ARCHITECTURE AND CONCEPTUALIZATION (G. Pask, U.K.; and G. de Zeeuw, The Netherlands)

F BIOCYBERNETICS AND MATHEMATICAL BIOLOGY (L.M. Ricciardi, Italy)

G SYSTEMS AND ECOLOGY (F.J. Radermacher, Germany; and K. Fedra, Austria)

H CYBERNETICS AND INFORMATICS IN MEDICINE (G. Gell, Austria; and G. Porenta, Austria)

I CYBERNETICS OF SOCIO-ECONOMIC SYSTEMS (K. Balkus, U.S.A.; and O. Ladanyi, Austria)

J SYSTEMS, MANAGEMENT AND ORGANIZATION (G. Broekstra, the Netherlands; and R. Hough, U.S.A.)

K CYBERNETICS OF COUNTRY DEVELOPMENT (P. Ballonoff, U.S.A.; T. Koizumi, U.S.A.; and S.A. Umpleby, U.S.A.)

L COMMUNICATION AND COMPUTERS (A.M. Tjoa, Austria)

M INTELLIGENT AUTONOMOUS SYSTEMS (J.W. Rozenblit, U.S.A.; and H. Prähofer, Austria)

N CYBERNETIC PRINCIPLES AND KNOWLEDGE DEVELOPMENT (F. Heylighen, Belgium; and S.A. Umpleby, U.S.A.)

O CYBERNETICS, SYSTEMS, AND PSYCHOTHERAPY (M. Okuyama, Japan; and H. Koizumi, U.S.A.)

P ARTIFICIAL NEURAL NETWORKS AND ADAPTIVE SYSTEMS (S. Grossberg, U.S.A.; and G. Dorffner, Austria)

Q ARTIFICIAL INTELLIGENCE AND COGNITIVE SCIENCE (V. Marik, Chechia; and R. Born, Austria)

In the Proceedings of the Meeting, on more than 1900 pages, four tutorials and 241 papers are published. A workshop within the Symposium H, entitled NEW DEVELOPMENTS IN MEDICAL INFORMATION SYSTEMS: MEDICAL IMAGE FILE FORMATS AND STANDARDIZATION ISSUES (O. Ratib, University Hospital of Geneva, Switzerland) is organized.

Tutorials of the Meeting are the following:

1. B. Stilman, A SYNTACTIC APPROACH TO HEURISTIC NETWORKS: LINGUISTIC GEOMETRY, University of Colorado, U.S.A.

2. C. Carlsson, FUZZY SETS AND IMPRECISE BUT RELEVANT DECISIONS, Abo Akademi University, Abo, Finland.
3. Irina V. Ezhkova, CONTEXTUAL SYSTEMS: A NEW TECHNOLOGY FOR KNOWLEDGE BASED SYSTEM DEVELOPMENT, Russian Academy of Sciences, Moscow, Russia.
4. S.A. Umpleby, TWENTY YEARS OF SECOND ORDER CYBERNETICS, George Washington University, Washington, D.C., U.S.A.

Seven papers from Slovenia are published in the Proceedings:

1. P. Kokol, SOFTWARE SYSTEM DESIGN WITH THE METAPARADIGM, D/385–390, UM[1].
2. D. Ursič and M. Mulej, FROM A GENERAL TO AN INDIVIDUAL PERCEPTION OF AN ORGANIZATION—PROBLEMS ALONG SIMPLIFICATION, E/715–722, UM.
3. D. Lesjak and S. Bobek, TANGIBLE AND INTANGIBLE STRATEGIC IMPACTS OF INFORMATION TECHNOLOGY USAGE IN SLOVENIA, J/1221–1228, UM.
4. D. Savin, GOING GLOBAL PARADIGM IN THE COUNTRIES OF CENTRAL AND EASTERN EUROPE, K/1271–1278, UM.
5. M. Mulej, M. Rebernik, and S. Kajser, TOTAL QUALITY MANAGEMENT—A MAYBE GOOD SUPPORT FOR SYSTEMS THINKING AND INFORMATION IN LESS DEVELOPED AREAS, K/1279–1286, UM.
6. M. Pivka, HOW MANY METRICS ARE REALLY NECESSARY, L/1369–1376, UM.
7. A.P. Železnikar, TOWARDS AN INFORMATIONAL UNDERSTANDING OF KNOWLEDGE, N/1587–1596, Ljubljana, Slovenia.

On the last day of the Meeting, April 8, 1994, the Best Paper Awards will be granted.

Authors of papers are coming from 39 countries of all six continents. The authors' countries are the following:

- Argentina, Australia, Austria;
- Belarus, Belgium, Brazil, Bulgaria;
- Canada, Chechia, China;
- Denmark;
- Finland, France;
- Germany, Greece;
- Hungary;
- India, Iran, Israel, Italy;
- Japan;
- Mexico;
- New Zealand, The Netherlands;
- Poland, Portugal;
- Republics of China (Taiwan), Russia;
- Slovakia, Slovenia, South Africa, Spain, Sweden, Switzerland;
- Turkey;
- Ukraine, U.K., U.S.A.; and
- Venezuela.

The Meeting in Vienna is becoming one of the most significant European and world conferences in the field of systems research and cybernetics. Its European counterpart is held every three years in Namur, Belgium (the next one will be organized in 1995, celebrating the hundredth anniversary of the birth of Norbert Wiener, the founder of modern cybernetics).

A.P. Železnikar

[1]UM is a shortcut for the University of Maribor, Slovenia.

International Conference on Interdisciplinary Research and the 2nd Orwellian Symposium

August 8–11, 1994
Karlovy Vary
Chech Republic
Sponsored by:
The International Institute for Advanced Studies in Systems Research and Cybernetics and Society for Applied Systems Research

The Conference will provide a forum for the presentation and discussion of short reports on the current interdisciplinary research. The 2nd Orwellian Symposium will compare the vision of the totalitarian world described by George Orwell in his classic work "1984" to the technological and social advances in modern life. It will also examine how far we are from this vision and what can we do to prevent a slide into a totalitarian world order.

Some of the topics to be covered are: mechanisms of sociopolitical controls; the impact of the socioeconomic, ecological, political, scientific and other developments on our lives and on our future; erosion of individual and national freedom; invasion of privacy; omission of information, deceptive newscasting and misinformation in mass communication media; social role of television and the impact of TV—viewing on society; local and global increase in criminality; growing divergence between common sense and legal structures; legality versus legitimacy; psychopathology of sociopolitical power; and others.

All submissions and correspondence should be addressed to: Prof. George Lasker, President of the I.I.A.S. and Conference Chairman, School of Computer Science, University of Windsor, Windsor, Ontario, N9B 3P4 Canada.

The address after June 10, 1994 is: Prof. George E. Lasker, Hauptpostlagernd, 70001 Stuttgart, Germany.

A.P. Železnikar

7th International Conference on Systems Research, Informatics and Cybernetics

August 15–21, 1994
Convention Centre, Congresshouse
Baden-Baden, Germany.
Sponsored by:
The International Institute for Advanced Studies in Systems Research and Cybernetics and Society for Applied Systems Research

The Conference will provide a forum for the presentation and discussion of short reports on current systems research in humanities, sciences and engineering. A number of specialized symposia will be held within the Conference to focus on research in computer science, linguistics, cognitive science, psychocybernetics, synergetics, logic, philosophy, management, education and other related areas.

The aim of the conference is to encourage and facilitate the interdisciplinary and transdisciplinary communication and cooperation amongst scientists, engineers, and professionals working in different fields, and to identify and develop those areas of research that will most benefit from such a cooperation. Themes at the Conference include the following topics:

1. Natural/artificial intelligence; AI tools & technologies; knowledge based expert systems; intelligent robotics & computer brainware; systems creativity; knowledge acquisition and representation; cognitive development & knowledge utilization; perception; apperception & concept formation; man/machine vision & image processing; machine learning; reasoning and thought processes; memory; cognitive modelling; cognitive maps & styles; multisensorial integration in biological systems/machines; natural language processing; knowledge base management; pattern analysis.

2. Information Systems: theory/applications; intelligent MIS; management support systems; intelligent decision support systems; innovative management methodologies; systems modelling and simulation; systems research methodologies;

computer aided software development; AI-based diagnostics and forecasting; system complexity; analysis & modelling; application of AI to strategic planning; systems philosophy; current issues in GST; human factors; man-machine interaction; expert approaches to manufacturing; intelligent robotics & advanced automation; impact of automation and computerization.

3. Neural information processing; neural computers; neural networks; neuromorphic systems; cerebromimetic models; neural learning; self-programming in neural nets; artificial neural systems; neural systems research; neuronal modelling; neural controls; self-organizing adaptive systems; neural bases of human creativity & intelligence.

4. Parallel/distributed processing; parallel algorithms; distributed algorithms, reliability $ fault-tolerance; parallel computer architecture; supercomputing; computer graphics; computer animation; image synthesis; software engineering; design development; intelligent controls; virtual reality systems.

5. Quality of human life; aesthetics; ethics, human values; beliefs; family quality and Social development; philosophy and perceptions of contemporary man; current philosophical issues; impact of TV/mass communication media on human life; role models in contemporary society.

6. Education; corporate training; creativity enhancement; instructional development; CAI; intelligent tutoring systems; innovative teaching/learning methods; accelerated learning; superlearning; expanding human potential; self improvement methodologies; ecology of human mind; educational synergetics.

7. Human decision making & problem solving; coping with information overload; dealing with distorted information/misinformation; living with incomplete/insufficient information; improving QOL of the aged; slowing down aging; rejuvenation; burnout, technostress & quality of life; human ecology; environmental quality management; health care systems research.

8. Architecture, human habitat & quality of life; human systems and institutions; society and cultural development; systems research in arts and humanities; literary models; semiotics; psycholinguistics; human communication; human emotions; literature & QOL; fine arts and QOL.

9. Biocybernetics; neurocybernetics; brain/mind research; mental models; biomedical engineering; biomimetic engineering; psychocybernetics; psychotronics; psychosynergetics; mind/brain programming; neurolinguistic programming; motivational research; psychosocial/biosocial resonation; synergetics; ergonomics; somatoinformatics; somatocybernetics.

10. Sociocybernetics, socioinformatics, sociocatalysis & social action; social norms & standards; sociopolitical systems models/processes; social participation and interaction; uses and abuses of power in human affairs; strategies for social/societal healing; sociopolitical conflict resolution; enhancing international cooperation and development; population explosion & its consequences; strategies for peace with freedom and prosperity.

11. Philosophy and methodology of science; logic; mathematics; statistics; operation research; fuzzy sets & systems; possibilistic analysis; information theory; scientific inquiry.

12. Policy design: contemporary perceptions of the future; expert systems for planning and forecasting; analysis of future options; global trends and future scenarios; forecasting future developments; planning for the future; creating better future.

All submissions and correspondence should be addressed to: Prof. George E. Lasker, President of the I.I.A.S. and Conference Chairman, School of Computer Science, University of Windsor, Windsor, Ontario, N9B 3P4 Canada.

The address after June 10, 1994 is: Prof. George E. Lasker, Hauptpostlagernd, 70001 Stuttgart, Germany.

A.P. Železnikar

THE MINISTRY OF SCIENCE AND TECHNOLOGY OF THE REPUBLIC OF SLOVENIA

The Ministry of Science and Technology also includes the Standards and Metrology Institute of the Republic of Slovenia, and the Industrial Property Protection Office of the Republic of Slovenia.

Scientific Research and Development Potential

The statistical data for 1991 showed that there were 230 research and development institutions, organizations or organizational units in Slovenia, of which 73 were independent, 32 were at the universities, and 23 at medical institutions. The remainder were for the most part departments in industry. Altogether, they employed 13,000 people, of whom 5500 were researchers and 4900 expert or technical staff.

In the past 10 years, the number of researchers has almost doubled: the number of Ph.D. graduates increased from 1100 to 1484, while the number of M.Sc.'s rose from 650 to 1121. The 'Young Researchers' (i.e. postgraduate students) programme has greatly helped towards revitalizing research. The average age of researchers has been brought down to 40, with one-fifth of them being younger than 29.

The table below shows the distribution of researchers according to educational level and fields of research:

	Ph.D.	M.Sc.
Natural Sciences	315	217
Engineering-Technology	308	406
Medical Sciences	262	174
Agricultural Sciences	122	69
Social Sciences	278	187
Humanities	199	68
Total	1484	1121

Financing Research and Development

Statistical estimates indicate that US$ 260 million (1.7% of GNP) was spent on research and development in Slovenia in 1991. Half of this comes from public expenditure, mainly the state budget. In the last three years, R&D expenditure by business organizations has stagnated, a result of the current economic crisis. This crisis has led to the financial decline and increased insolvency of firms and companies. These cannot be replaced by the growing number of mainly small businesses. The shortfall was addressed by increased public-sector R&D spending: its share of GNP doubled from the mid-seventies to 0.86% in 1993.

Overall, public funds available for Research & Development are distributed in the following proportions: basic research (35%), applied research (20%), R&D infrastructure (facilities) (20%) and education (25%).

Research Planning

The Science and Technology Council of the Republic of Slovenia, considering initiatives and suggestions from researchers, research organizations, professional associations and government organizations, is preparing the draft of a national research program (NRP). This includes priority topics for the national research policy in basic and applied research, education of expert staff and equipping institutions with research facilities. The NRP also defines the mechanisms for accelerating scientific, technological and similar development in Slovenia. The government will harmonize the NRP with its general development policy, and submit it first to the parliamentary Committee for Science, Technology and Development and after that to parliament as a whole. Parliament approves the NRP each year, thus setting the basis for deciding the level of public support for R&D.

The Ministry of Science and Technology provides organizational support for the NRP, but it is mainly a government institution responsible for controlling expenditure of the R&D budget, in compliance with the NRP and the criteria provided by the Law on Research Activities: International quality standards of groups and projects, relevance to social development, economic efficiency and rationality of the project. The Ministry finances research or co-finances development projects through public bidding and partly finances infrastructure research institutions (national institutes), while it directly finances management and top-level science.

The focal points of R&D policy in Slovenia are:
- maintaining the high level and quality of research activities,
- stimulating cooperation between research and industrial institutions,
- (co)financing and tax assistance for companies engaged in technical development and other applied research projects,
- research training and professional development of leading experts,
- close involvement in international research and development projects,
- establishing and operating facilities for the transfer of technology and experience.

In evaluating the programs and projects, and in deciding on financing, the Ministry works closely with expert organizations and Slovene and foreign experts. In doing this, it takes into consideration mainly the opinions of the research leaders and of expert councils consisting of national research coordinators and recognized experts.

The Ministry of Science and Technology of the Republic of Slovenia. Address: Slovenska c. 50, 61000 Ljubljana. Tel. +386 61 131 11 07, Fax +38 61 132 41 40.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 800 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of Slovenia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the "Jožef Stefan" Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia
Tel.:+386 61 1259 199, Fax.:+386 61 219 385
Tlx.:31 296 JOSTIN SI
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Ines Černe

INFORMATION FOR CONTRIBUTORS

1 Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of the original figures on separate sheets and the text on an IBM PC DOS floppy disk or by e-mail – both in ASCII and the Informatica LaTeX format. Style (attached) and examples of papers can be obtained by e-mail from the Contact Person.

2 News, letters, opinions

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

```
\documentstyle[twoside,informat]{article}

\begin{document}
  \title{TITLE OF THE ARTICLE}
   \author{First Author \\ Address \\ E-mail \\
      AND \\
      Second Author \\ Address \\ E-mail}
  \titleodd{TITLE IN HEADER}
  \authoreven{Author's name in header}
  \keywords{article keywords}
  \edited{editor in charge}
  \received{date}
  \revised{date}
  \accepted{date}
  \abstract{abstract -- around 200 words}
  \maketitle

  \section{Introduction}
    Text of Introduction.

  \section{Subject}
    Text of the Subject.

  \begin{figure}
    An example of a figure over 1 column.
    \caption{Caption of the figure 1}
  \end{figure*}

  \begin{figure*}
    An example of a figure over 2 columns.
    \caption{Caption of the figure 2}
  \end{figure*}

  \begin{thebibliography}{99}
    \bibitem{} First Author: {\sl Title},
      Magazine, Vol. 1, No. 1.
  \end{thebibliography}

\end{document}
```

```
\def\journal{Informatica {\bf 17} page xxx--yyy}
\immediate\write16{'Informatica' V1.2  by B."Z}
\newif\iftitle \titlefalse
\hoffset=11mm \voffset=8mm
\oddsidemargin=-21mm \evensidemargin=-14mm
\topmargin=-33mm
\headheight=17mm \headsep=10mm
\footheight=8.4mm \footskip=52.5mm
\textheight=242mm \textwidth=170mm
\columnsep=5mm \columnseprule=0pt
\twocolumn \sloppy \flushbottom
\parindent 1em
\leftmargini 2em \leftmargin\leftmargini
\leftmarginv .5em \leftmarginvi .5em
\def\labelitemi{\bf --} \def\labelitemii{--}
\def\labelitemiii{--}
\setcounter{secnumdepth}{3}
\def\maketitle{\twocolumn[%
  \vbox{\hsize=\textwidth\Large\bf\raggedright
  \uppercase{\@title}}\vss\bigskip\bigskip \vbox{
  \hsize=\textwidth \@author}\bigskip\smallskip
  \vbox{\hsize=\textwidth {\bf Keywords:}
    \@keywords}
  \bigskip \hbox{{\bf Edited by:} \@edited}
  \smallskip
  \hbox{{\bf Received:}
    \hbox to 10em{ \@received\hss}
    {\bf Revised:}\hbox to 10em{ \@revised\hss}
    {\bf Accepted:}\hbox to 10em{ \@accepted\hss}}
  \bigskip \vbox{\hsize=\textwidth
    \leftskip=3em \rightskip=3em \sl \@abstract}
  \bigskip\bigskip]\titletrue}
\def\maketitleauthor{\twocolumn[%
  \vbox{\hsize=\textwidth \Large\bf\raggedright
  \@title}\vss \bigskip\bigskip
  \vbox{\hsize=\textwidth \@author}
  \bigskip\bigskip] \gdef\@title{}\titletrue}
\def\makeonlytitle{\twocolumn[%
  \vbox{\hsize=\textwidth \Large\bf\raggedright
  \@title}\vss\bigskip\bigskip]
  \gdef\@title{}\titletrue}
\def\@title{} \def\@author{}
\def\@titleH{} \def\@authorH{}
\def\@keywords{} \def\@edited{} \def\@abstract{}
\def\@received{} \def\@revised{} \def\@accepted{}
\def\authoreven#1{\gdef\@authorH{#1}}
\def\titleodd#1{\gdef\@titleH{\uppercase{#1}}}
\def\keywords#1{\gdef\@keywords{#1}}
\def\edited#1{\gdef\@edited{#1}}
\def\received#1{\gdef\@received{#1}}
\def\revised#1{\gdef\@revised{#1}}
\def\accepted#1{\gdef\@accepted{#1}}
\long\def\abstract#1{\gdef\@abstract{#1}}
\def\section{\@startsection {section}
  {1}{\z@}{-3.5ex plus -1ex minus -.2ex}
  {2.3ex plus .2ex}{\Large\bf\raggedright}}
\def\subsection{\@startsection{subsection}
  {2}{\z@}{-3.25ex plus -1ex minus -.2ex}
  {1.5ex plus .2ex}{\large\bf\raggedright}}
\def\subsubsection{\@startsection{subsubsection}
  {3}{\z@}{-3.25ex plus -1ex minus -.2ex}
  {1.5ex plus .2ex}{\normalsize\bf\raggedright}}
\def\@evenhead{\hbox to 3em{\bf\thepage\hss}
  {\small\journal\hfil \iftitle\else
  \@authorH \fi}\global\titlefalse}
\def\@oddhead{{\small\iftitle\else \@titleH \fi
  \hfil\journal}\hbox to 3em{\hss\bf\thepage}
  \global\titlefalse}
\def\@evenfoot{\hfil} \def\@oddfoot{\hfil}
\endinput
```

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 17th year, it is becoming truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

Please, return this questionnaire.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

☐ We intend to cooperate (describe):

☐ Proposals for improvements (describe):

Informatica is published in one volume (4 issues) per year.

If possible, send one copy of Informatica to the local library.

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 61111 Ljubljana, Slovenia.

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

REVIEW REPORT

Basic Instructions

Informatica publishes scientific papers accepted by at least two referees outside the author's country. Each author should submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. The names of the referees should not be revealed to the authors under any circumstances. The names of referees will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees.

It is highly recommended that each referee writes **as many remarks as possible directly on the manuscript**, ranging from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks, and if accepted also to the Contact Person with the accompanying completed Review Reports. The Executive Board will inform the author that the paper is accepted, meaning that it will be published in less than one year after receiving original figures on separate sheets and the text on an IBM PC DOS floppy disk or through e-mail – both in ASCII and the Informatica LaTeX format. Style and examples of papers can be obtained through e-mail from the Contact Person.

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

Date Sent:

Date to be Returned:

Name and Country of Referee:

Name of Editor:

Title:

Authors:

Additional Remarks:

All boxes should be filled with numbers 1-10 with 10 as the highest rated.

The final mark (recommendation) consists of two orthogonal assessments: scientific quality and readability. The readability mark is based on the estimated perception of average reader with faculty education in computer science and informatics. It consists of four subfields, representing if the article is interesting for large audience (interesting), if its scope and approach is enough general (generality), and presentation and language. Therefore, very specific articles with high scientific quality should have approximately similar recommendation as general articles about scientific and educational viewpoints related to computer science and informatics.

☐ SCIENTIFIC QUALITY

☐ Originality

☐ Significance

☐ Relevance

☐ Soundness

☐ Presentation

☐ READABILITY

☐ Interesting

☐ Generality

☐ Presentation

☐ Language

☐ FINAL RECOMMENDATION

☐ Highly recommended

☐ Accept without changes

☐ Accept with minor changes

☐ Accept with major changes

☐ Author should prepare a major revision

☐ Reject

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or Board of Referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board and Board of Referees are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Volume 18 Number 3 October 1994 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Contents: